



Data Quality Management in Large-Scale Cyber-Physical Systems

Ahmed Abdulhasan Alwan

School of Architecture, Computing and Engineering

University of East London

A thesis presented for the degree of

Doctor of Philosophy

July 19, 2021

Abstract

Cyber-Physical Systems (CPSs) are cross-domain, multi-model, advance information systems that play a significant role in many large-scale infrastructure sectors of smart cities public services such as traffic control, smart transportation control, and environmental and noise monitoring systems. Such systems, typically, involve a substantial number of sensor nodes and other devices that stream and exchange data in real-time and usually are deployed in uncontrolled, broad environments. Thus, unexpected measurements may occur due to several internal and external factors, including noise, communication errors, and hardware failures, which may compromise these systems quality of data and raise serious concerns related to safety, reliability, performance, and security. In all cases, these unexpected measurements need to be carefully interpreted and managed based on domain knowledge and computational models.

Therefore, in this research, data quality challenges were investigated, and a comprehensive, proof of concept, data quality management system was developed to tackle unaddressed data quality challenges in large-scale CPSs. The data quality management system was designed to address data quality challenges associated with detecting: sensor nodes measurement errors, sensor nodes hardware failures, and mismatches in sensor nodes spatial and temporal contextual attributes. Detecting sensor nodes measurement errors associated with the primary data quality dimensions of accuracy, timeliness, completeness, and consistency in large-scale CPSs were investigated using predictive and anomaly analysis models via utilising statistical and machine-learning techniques. Time-series clustering techniques were investigated as a feasible mean for detecting long-segmental outliers as an indicator of sensor nodes' continuous halting and incipient hardware failures. Furthermore, the quality of the spatial and temporal contextual attributes of sensor nodes observations was investigated using timestamp analysis techniques.

The different components of the data quality management system were tested and calibrated using benchmark time-series collected from a high-quality, temperature sensor network deployed at the University of East London. Furthermore, the effectiveness of the proposed data quality management system was evaluated using a real-world, large-scale environmental monitoring network consisting of more than 200 temperature sensor nodes distributed around London.

The data quality management system achieved high accuracy detection rate using LSTM predictive analysis technique and anomaly detection associated with DBSCAN. It successfully identified timeliness and completeness errors in sensor nodes' measurements using periodicity analysis combined with a rule engine. It achieved up to 100% accuracy in detecting potentially failed sensor nodes using the characteristic-based time-series clustering technique when applied to two days or longer time-series window. Timestamp analysis was adopted effectively for evaluating the quality of temporal and spatial contextual attributes of sensor nodes observations, but only within CPS applications in which using gateway modules is possible.

Acknowledgements

I would like to express my heartfelt appreciation to Professor Allan J Brimicombe, Dr. Mihaela Anca Ciupala, Dr. Paolo Falcarin and Dr. Andres Baravalle for their professional guidance and support throughout this research.

This research was sponsored by the University of East London (UEL) under the PhD scholarship scheme 2017 and through the UEL Research Internship programme 2019. Therefore, I am very grateful for the generous sponsorship and kind support from UEL staff, in particular Richard Bottoms and Charlotte Forbes.

I would like to thank all my friends and staff in the Ministry of Planning - Iraq for their continuous support.

I am deeply grateful for the collaboration with Thingful. Ltd, UK, in particular, I thank Usman Haque for his insight and expertise that assisted the research.

To Suad, My parents and Rawia

Contents

Abstract	i
Acknowledgements	iii
Table of Contents	iv
List of Figures	xi
List of Tables	xxi
List of Abbreviations	xxv
Author's declaration	xxvii
1 Introduction	1
1.1 Cyber-Physical Systems	1
1.2 Smart Cities as Large-Scale CPSs	6
1.3 Research Motivation	11
1.4 The Research Aim	15
1.5 The Research Questions and Objectives	16
1.6 Novelty of Research	18
1.7 Research Boundaries	19
1.8 Research Structure	20
2 Literature Review	22
2.1 Data Quality Concepts and Terminology	23
2.1.1 Accuracy	24

2.1.2	Time-Related Accuracy (Timeliness)	25
2.1.3	Completeness (Completeness)	28
2.1.4	Consistency	30
2.2	Data Quality Challenges in Large-Scale CPSs, a Systematic Literature Review	31
2.2.1	Review Motivation / Introduction	32
2.2.2	Review Process and Methodology	34
2.2.3	Review Conduct and Primary Studies Selection	40
2.2.4	RQ1: Data Quality Challenges in Large-Scale CPSs.	57
2.2.5	RQ2: Data Mining and Data Quality Management in Large-Scale CPSs.	61
2.2.6	RQ3: Unaddressed Data Quality Management Challenges in Large-Scale CPSs and The Research Gap.	67
2.3	The Research Questions	70
2.4	Summary	71
3	System Design (Methodology)	72
3.1	Overview of Research Paradigms	72
3.2	Empirical Research Methods	74
3.2.1	Quantitative	74
3.2.2	Qualitative	74
3.2.3	Mixed-Method (Triangulation)	75
3.3	Empirical Research Strategy	76
3.3.1	Experimental	76
3.3.2	Case Study	77
3.3.3	Choosing the Research Methodology	77
3.4	System Design and Development Phases	81
3.4.1	System Analysis and Design	85
3.4.2	Data Acquisition Unit	87
3.4.3	Data Quality Assessment Unit	91
3.4.4	Selecting Data Analysing Methods	96
3.4.5	Time-Series Decomposition	96

3.5	Online Mode - Predictive Analysis Models	99
3.5.1	Simple Forecasting Methods	101
3.5.2	Holt-Winters Seasonal	102
3.5.3	ARMA, ARIMA and Seasonal ARIMA Models	104
3.5.4	Gaussian Process Regression	110
3.5.5	Long Short-Term Memory Networks	112
3.6	Online Mode – Anomaly Analysis Models	114
3.6.1	Distance-Based Spatial Clustering (K-means)	121
3.6.2	Density-Based Spatial Clustering (DBSCAN)	123
3.7	Online Mode - Timestamp Analysis (Temporal Consistency)	127
3.8	Offline Mode - Time-series Clustering	130
3.8.1	Dynamic Time Warping	134
3.8.2	K-Shape	135
3.8.3	Characteristic-Based Time-Series Clustering	135
3.9	Offline Mode – Timestamp Analysis (Spatial Attributes Consistency)	136
3.10	Summary	142
4	Implementation and Results	143
4.1	Data Acquisition and Data Process	144
4.1.1	Sensor Node Networks	144
4.1.2	Datasets	150
4.1.3	Software Framework	156
4.2	Online-Mode Data Quality Assessment	164
4.2.1	Predictive Analysis Models	165
4.2.2	Anomaly Analysis Models	207
4.2.3	Timestamp Analysis (Temporal Consistency)	220
4.3	Offline-Mode Data Quality Assessment	223
4.3.1	Time-Series Clustering Models	224
4.3.2	Timestamp Analysis Model (Spatial Attributes Consistency)	235
4.4	Discussion and Summary	243
4.4.1	The Data Acquisition Unit (Layers 1 and 2)	243

4.4.2	The Data Quality Assessment Unit	244
5	Conclusions and Future Work	253
5.1	Revisiting the Research Questions and Objectives	254
5.1.1	Review Question-1:	254
5.1.2	Review Question-2:	260
5.1.3	Review Question-3:	260
5.2	Contribution to Knowledge	262
5.3	Conclusions	263
5.4	Future Work	266
	References	267
	Appendices	291
A	Technical and Implementation Details	292
A.1	Sensor Nodes Anatomy and Data Quality	292
A.1.1	Hardware Components	293
A.1.2	Operating System	294
A.2	The Technical Details of the Local Sensor Node Network	296
A.3	Technical Details of The Data Acquisition Unit	301
A.3.1	JSON Parsing and Duplication Prevention Trigger	307
A.3.2	The Periodicity Analysis Rule Engine.	308
A.4	Configuration and programming details	310
A.4.1	Holt-Winters predictive model	310
A.4.2	ARMA and ARIMA predictive models	313
A.4.3	SARIMA predictive model	320
A.4.4	GPR predictive model	322
A.4.5	LSTM predictive model	326
A.4.6	K-means and DBSCAN Partitioning Models	329
A.4.7	DTW and K-Shape Models	335
A.4.8	Characteristics (features)-based Model	338

List of Figures

1.1	Cyber-Physical Systems structure diagram.	3
2.1	A holistic overview of the processes adopted in this systematic literature review.	35
2.2	The number of SLR primary studies by the year of publication, (October 2020).	42
2.3	The number of SLR primary studies by the country of publication.	43
2.4	The ratio of the data quality dimensions addressed by the SLR primary studies.	59
2.5	The key data quality challenges in large-scale CPSs.	60
2.6	The most popular data quality assessment/management methods or techniques in large-scale CPS applications based on the number (left) and the ratio (right) of the SLR studies.	62
2.7	The most popular data mining techniques in large-scale CPSs based on the No. of the SLR studies utilising these techniques.	63
2.8	The key data mining techniques used to assess the main data quality dimensions in large-scale CPSs.	63
2.9	A holistic diagram of the main data quality management/assessment methods and techniques and their associated data quality dimensions that these techniques are addressing, based on the SLR results.	64
3.1	The main processes of the experimental research approach.	77
3.2	The main processes of the case study research approach.	77
3.3	Research approaches based on the interconnection of Research Methods, Design, and paradigms.	78
3.4	The logical sequence to decide the research methodology, methods and techniques.	80

3.5	A holistic overview of the research overall phases.	82
3.6	Adoptive SDLC iterations and the key development phases of the data quality management system (Satzinger et al., 2015, p. 300). . .	83
3.7	A holistic view of the data quality assessment models' development activities (Brimicombe, 2009, p. 89).	84
3.8	The main components of the proof of concept data quality management system. The mode-model structure of the data quality assessment unit is illustrated in Figure 3.11.	86
3.9	Sensor nodes data stream as discrete, time-stamped observations snapshots (Appice et al., 2014).	90
3.10	The high-level process diagram of the data acquisition unit.	92
3.11	The mode-model structure of the data quality assessment unit and data quality dimensions.	95
3.12	Additive time-series decomposition, a time-series of a single real-world, temperature sensor node.	98
3.13	An example (demo data) of a time-series with a seasonality that its magnitude varies with the trend (Brownlee, 2017a).	99
3.14	The high-level processes of a predictive data accuracy assessment model.	100
3.15	Predictive analysis techniques examined in the context of this research.	102
3.16	The high-level process diagram of Holt-Winters seasonal prediction model.	105
3.17	The high-level process diagram of the ARMA, ARIMA and SARIMA prediction models.	109
3.18	A introductory example to Gaussian Processes Regression (Pedregosa et al., 2011).	111
3.19	Sensor node's time-series as a $(n \times 1)$ array of observations.	113
3.20	Time-series transformation (vectorisation) concept.	113
3.21	LSTM time-series transforming process, time-steps = 7 in this example.	114
3.22	The high-level process diagram of the LSTM predictive model. . . .	115
3.23	Spatial autocorrelations among time-series of ideal nearby sensor nodes.	117
3.24	The irregularity in temperature levels around London due to the impact of Urban Heat Islands. The line A to B is presented in Figure 3.25, (MetOffice, 2019).	119

3.25	The heat profile map of London highlighting the impact of urban heat islands, (MetOffice, 2019).	120
3.26	Voronoi tessellation method to describe spatially partitioned two-dimensional data, (Guo et al., 2003, P. 126).	120
3.27	K-means partitional clustering model flowchart diagram illustrating K estimating process using the Silhouette coefficient analysis method.	124
3.28	DBSCAN parameters and data points categorisation (Raschka & Mirjalili, 2017, p. 373).	125
3.29	DBSCAN partitional clustering model flowchart diagram illustrating Eps estimating process using the Silhouette coefficient analysis method.	126
3.30	An example of a typical, real-world time-series with data inconsistency issues.	128
3.31	The different categories of the point (short) outliers, (SURI et al., 2019, p. 180).	131
3.32	Different categories of temporal long outliers, (SURI et al., 2019, p. 180).	132
3.33	An illustration of how DTW warps one time series to another (Salvador & Chan, 2007).	134
3.34	An example of a basic sensor node network topology.	137
3.35	Each sensor node observation may hold up to three different timestamps added from the network components.	139
3.36	The main processes of the timestamp analysis model.	141
4.1	The geographical distribution of the real-world sensor nodes network used as a case study in this research, the Met Office (blue), Open Weather Map (red) and Smart Citizen (green).	145
4.2	The estimated network topology of the large-scale sensor node network.	147
4.3	The deployment map of the local sensor node network at the University of East London.	148
4.4	The topology of the local sensor node network.	149
4.5	The key attributes of the real-world, large-scale sensor nodes dataset.	152
4.6	Time-series of seven days window of sensor nodes observations aggregated by the variations range in their value attribute.	153

4.7	An example of two temperature time-series with long segmental outliers (b and c) comparing with a typical time-series collected from a functional sensor node (a).	154
4.8	Examples of the temporal inconsistencies in time-series of real-world sensor nodes using time-series decomposition, (sensor ID=47qwbfa, 11-23/01/2019, London).	155
4.9	Time-series of 274 sensor nodes categorised according to the range of number of observations (seven days window).	156
4.10	An example of sensor nodes that their coordinates do not reflect their correct location.	157
4.11	The classes and main attributes of the data acquisition software.	159
4.12	The sequence diagram of the data acquisition unit.	160
4.13	Data-pipes details used by the data acquisition software to construct the RESTful Requests URLs.	161
4.14	The JSON parsing process from the destination table into the corresponding observations table, air quality table as an example.	163
4.15	The process diagram of the duplication prevention mechanism of the data acquisition unit.	164
4.16	The algorithms and techniques empirically tested to validate the accuracy of sensor nodes' observations in real-time.	166
4.17	Time series decomposition of the benchmark time-series (Sensor ID: Monnit/493372).	168
4.18	Time series decomposition of the real-world sensor node time-series (ID: Thingful/jcw5m701).	170
4.19	The merge and re-sampling processes applied to the selected dataset to become a time-series with the timestamp attribute as a common index (Pandas, 2020).	171
4.20	The structure of Holt-Winters Python package and its input parameters (Seabold & Perktold, 2010).	171
4.21	Holt-Winters was tested with twenty-five different dataset combinations with one step forward observation shift in each test.	172
4.22	The predictions accuracy of H-W were alternating randomly with each new observation fitted to the model.	172
4.23	The RMSE of the H-W prediction models is rapidly and randomly changing with each newly fitted observation.	173

4.24	H.W regressive model is showing inconsistency in the accuracy of predictions between the ideal (left) and the real-world (right) time-series at the same point in time.	174
4.25	H-W prediction model can not adapt to rapid changes in the trend of the time-series.	174
4.26	The time required by the Holt-Winters seasonal model to render the prediction result of the 25 sequential tests.	175
4.27	Testing the stationarity of ARMA training datasets by dividing each dataset and comparing the mean and the variance values of each sub-set. (left) ideal dataset, (right) real-world dataset.	178
4.28	The results of applying the ADF and KPSS stationarity tests on the ideal dataset.	179
4.29	The results of applying the ADF and KPSS stationarity tests on the real-world dataset.	179
4.30	The ACF and PACF diagrams of the real-world (stationary) dataset.	180
4.31	The values of ARMA parameters of the ideal dataset (a) and the real-world dataset(b) determined using the Grid Search and AIC method.	181
4.32	The values of ARIMA parameters for the ideal dataset (a) and the real-world dataset(b) determined using the Grid Search and AIC method.	182
4.33	The values of SARIMA parameters for the ideal dataset (a) and the real-world dataset(b) determined using the Grid Search and AIC method.	182
4.34	The residua ACF and PACF diagrams of the ARMA (2,2) model applied to the real-world dataset.	183
4.35	The residuals of the ARMA (2,2) models applied to the real-world dataset.	183
4.36	The result of fitting the ARMA (2,1) model with the testing part of ideal dataset.	184
4.37	The results of fitting the ARMA (2,2) model with the testing part of the real-world dataset.	184
4.38	The one-step forecast approach applied to the ARMA models using the real-world dataset.	185

4.39	An overview of the full dataset, SARIMA prediction graph (green) and SARIMA one-step prediction approach using the real-world dataset.	186
4.40	The time required to optimise the ARMA (a), ARIMA (b) and SARIMA (c) models for both the ideal and real-world datasets. . . .	188
4.41	The time required by ARMA (a), ARIMA (b) and SARIMA (c) models to render predictions results.	189
4.42	The structure and parameters of Gaussian Process Regressor package provided by Scikit-learn (Pedregosa et al., 2011).	190
4.43	The result of fitting the ideal dataset to the basic GPR model.	192
4.44	The result of fitting the real-world dataset to the basic GPR model.	193
4.45	The result of fitting the ideal datasets to the GPR model with the optimiser iterations set to 20 and alpha to 5.	194
4.46	The result of fitting the real-world datasets to the GPR model with the optimiser iterations set to 20 and alpha to 5.	195
4.47	The time required by the GPR models to render prediction results of 25 sequential test for the ideal (Blue) and the real-world (Orange) datasets.	197
4.48	The output of the LSTM model fitted with the ideal dataset.	201
4.49	The output of the LSTM model fitted with the real-world dataset.	202
4.50	The time required by the LSTM model to render prediction results of 25 sequential test for the ideal(Orange) and the real-world(Blue) datasets.	203
4.51	The LSTM prediction model using 0.34 C° as the deviation threshold was utilised successfully to detect value attribute anomalies in the real-world dataset.	206
4.52	The accuracy data quality issues (anomalies) detected by the LSTM prediction model when applied on the real-world dataset.	206
4.53	The geographic distribution of all available temperature sensor nodes over a real-scale map (London 04/2020).	209
4.54	K-means main parameters by scikit-learn (Pedregosa et al., 2011).	209
4.55	Silhouette analysis to determine K-means k, the highest score was reached at K=115.	210
4.56	The outcome of applying K-means (K=115) represented by Voronoi tessellation approach plotted over the landscape of London, the blue dots are the centroids of the clusters.	211

4.57	K-means model renders different outcomes after applying it many times on the same dataset.	212
4.58	The time (Sec) required by K-means to render the clustering results applied to the same dataset while testing a range of K (1 to 360). . .	213
4.59	DBSCAN algorithm's main parameters applied using "cluster.DBSCAN" Python package provided by Scikit-learn (Pedregosa et al., 2011). .	214
4.60	DBSCAN highest Silhouette score was obtained at Eps = 1.66 Km. .	215
4.61	DBSCAN clustering result, the blue lines are highlighting high-density regions ($S > 3$) of sensor nodes distribution, the red arrows (added manually) are showing examples of sensor node(s) in low-density areas.	216
4.62	DBSCAN spatially partitioned the available data-points (sensor nodes) into 50 clusters at Eps = 1.66.	216
4.63	The time (Sec) required by DBSCAN to render the clustering results applied to the same dataset while testing a range of Eps (0.2 - 4 Km 40 step).	217
4.64	DBSCAN clusters labels and the number of sensor nodes in each cluster.	218
4.65	The eight of the 47 sensor nodes in cluster-5 that streamed observations between '2020-03-15 23:49:59' and '2020-03-15 23:59:59' of the real-world dataset.	219
4.66	Outlier detection in sensor nodes observations using anomaly analysis and spatial partitioning techniques.	220
4.67	The data stream as a two-dimensional array of C_t, C_{t-1} observations to calculate the interval between every two sequential observations.	221
4.68	Aggregating observations intervals to determine the Threshold Interval t_{Ths} for each sensor node.	222
4.69	An example of the logical expressions (policies) used inside the rule engine.	222
4.70	The result of applying the timestamp analysis approach for temporal consistency assessment on the ideal dataset.	223
4.71	The result of applying the timestamp analysis approach for temporal consistency assessment on the real-world dataset.	223
4.72	The algorithms and techniques adopted and evaluated within the context of the offline unit (module) of the data quality assessment.	224

4.73	The process diagram of the technical steps implemented to fit all available time-series as a 3D array into the Dynamic Time Warping (DTW) and K-Shape time-series clustering models.	227
4.74	DTW and K-Shape were able to successfully differentiate the indoors time series (incipient faults pattern) from other outdoors time-series.	228
4.75	DTW successfully separated time-series with the long segmental outliers from other (typical) time-series when applied to 7-days window real-world dataset.	229
4.76	K-Shape successfully separated time-series with the long segmental outliers from other (typical) time-series when applied to 7-days window real-world dataset.	229
4.77	DTW is not able to differentiate time-series with long segmental outliers from other typical time-series after it was applied to a shorter two days' time-window of real-world time-series.	230
4.78	K-Shape is less able to differentiate time-series with long segmental outliers from other typical time-series after it was applied to a shorter, two days' time-window, of real-world time-series.	231
4.79	The feature-based time-series clustering method was able to differentiate the indoors, incipient faults time series from the other time-series of the ideal dataset.	232
4.80	The process diagram of the technical steps implemented to fit all available time-series to the features-based time-series clustering model.	233
4.81	The feature-based time-series clustering technique successfully differentiated time-series with long segmental outliers when applied to the real-world (seven days' time window). The Graph lines with the same colour belong to the same cluster.	234
4.82	The feature-based time-series clustering technique successfully differentiated time-series with long segmental outliers even when applied to relatively short two days' time-series window of the real-world dataset. The Graph lines with the same colour belong to the same cluster.	234
4.83	The performance of the CPU four cores and the time required to perform the same task by DTW comparing to K-Shape, each graph line represents the performance of a single CPU core.	235
4.84	A time-window from the ideal dataset presenting observation from all available (four) sensor nodes.	236

4.85	Applying $y_{d_t} - y_{d_{t-1}}$ rendered multiple values of G_{dc} and its multiplications for each sensor node.	237
4.86	G_{dc} intervals were approximated, aggregated and aggregated again according to their greatest common divisor to determine the duty-cycle of gateway modules for each sensor nodes.	238
4.87	G_{dc} , the gateway duty-cycles (if any) of the real-world sensor nodes were replicating the duty-cycle of the data acquisition unit.	239
4.88	The geographical distribution of the Met office sensor nodes, highlighting the two sensor nodes of Kensington.	241
4.89	The two sensor nodes of Kensington as a reference to measure the shortest distance between any two sensors in the Met Office sensor node network used in this case-study.	241
4.90	Using the GPS-distance between coordinates calculator (GPS-Coordinates, 2020), revealed that Kensington sensors are separated by 255 meters (0.26 KM).	242
A.1	The main components of a typical wireless sensor node (Vacca, 2015, p. 2-1).	293
A.2	A representation of sensor-nodes components interaction chain (Forster, 2016).	295
A.3	The full network topology of the sensor node networks utilised in this research.	300
A.4	The deployment diagram of the data acquisition software.	305

List of Tables

1.1	Cyber-physical systems cross-domain applications in the context of smart cities.	10
2.1	SLR Review Questions and Objectives.	34
2.2	The SLR Search Terms and Keywords.	36
2.3	The list of the examined digital libraries to be used for the purpose of the SLR.	37
2.4	The list of the most relevant digital libraries used for identifying the SLR primary studies.	38
2.5	The SLR Inclusion and Exclusion Criteria.	38
2.6	The SLR Quality Assessment Questions (Matrix).	40
2.7	The SLR digital libraries, final search strings and the number of identified primary studies.	41
2.8	The final number of primary studies included in the SLR after applying the inclusion and exclusion criteria and fully reviewing all studies.	43
2.9	Primary Studies Referencing Details and their Overall Quality Assessment Score.	45
2.10	Results of the SLR data extraction process addressing data quality main challenges and the proposed solutions in large-scale CPS. . .	47
3.1	Qualitative and quantitative methods in empirical research strategies.	76
3.2	The main attributes of sensor nodes data streams in large-scale CPSs.	90
4.1	The four time-series of the ideal dataset (a snapshot).	151
4.2	The main attributes of the ideal dataset.	151
4.3	ARMA models parameters and requirements.	177

4.4	The RMSE of the ARMA prediction models for the ideal and the real-world datasets.	187
4.5	A list of sensor nodes providers of all sensors used in this case-study.	240
4.6	The evaluation results of the prediction analysis models applied to the ideal and real-world datasets.	246

Acronyms

AIC Akaike Information Criterion. 107

ARIMA Non-seasonal Autoregressive Integrated Moving Average. 108

ARMA Autoregressive Moving Average. 104

BIC Bayesian Information Criterion. 107

CPSs Cyber-Physical Systems. i, 11, 22, 32, 57

DBSCAN Density-Based Spatial Clustering of Applications with Noise. 51–53, 65, 121, 208

DQ Data Quality. 23

DQCs Data Quality Characteristics. 23

DQDs Data Quality Dimensions. 23

ECSs Embedded Computing Systems. 32

GCD Greatest Common Divisor. 236, 251, 259

GPR Gaussian Process Regression. 110

H-Ws Holt-Winters seasonal. 102

HRTSs Hard Real-Time Systems. 4

IoT Internet of Things. xxviii, 5, 48

ISO International Standardization Organisation. 23

LSTM Long Short-Term Memory. 112

RNN Recurrent Neural Networks. 112

SARIMA Seasonal Autoregressive Integrated Moving Average. 108

SD Standard Deviation. 24

SLR Systematic Literature Review. 31, 34

SRTSs Soft Real-Time Systems. 4

SSNs Smart Sensor Networks. 54

TRC Time-Related Accuracy. 25

TSD Time-Series Decomposition. 96

UHI Urban Heat Islands. 118, 207

WSNs Wireless Sensor Networks. 5, 30

Author's declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award. Work submitted for this research degree at the University of East London has not formed part of any other degree either at the University of East London or at another establishment.

Relevant scientific seminars and conferences were regularly attended at which work was often presented:

Publications:

A. A. Alwan, M. A. Ciupala, A. Baravalle and P. Falcarin, "HADES: a Hybrid Anomaly Detection System for Large-Scale Cyber-Physical Systems," 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 2020, pp. 136-142, doi: 10.1109/FMEC49853.2020.9144751.

Alwan A.A., Baravalle A., Ciupala M.A., Falcarin P. (2019) An Open Source Software Architecture for Smart Buildings. In: Arai K., Kapoor S., Bhatia R. (eds) Intelligent Systems and Applications. IntelliSys 2018. Advances in Intelligent Systems and Computing, vol 869. Springer, Cham, doi:10.1007/978-3-030-01057-7_14.

Publications Under Progress:

The Quality of Data in Smart Cities'Cyber-Physical Systems: A Systematic Review.

Predictive Analysis for Data Quality Assessment in Large-Scale Cyber-Physical Systems.

Time-Series Clustering for Fault Detection in large-Scale Cyber-Physical Systems.

The published and the under progress research papers are listed in Appendix ??.

Presentations and conferences attended:

Training Internship: for two months at Thingful. Ltd, an IoT company based in London, (30/04/2019 - 30/06/2019).

Training: Participating in the "5th International Winter School on Big data" at the University of Cambridge, (07/01/2019 - 11/01/2019).

Training: Participating in "O'Reilly Artificial Intelligence training workshops", London, (15/10/2019 - 17/10/2019).

Training: Participating in the summer school of "Building tomorrow society: IoT applications and data management" at the University of Politecnico di Torino / Italy, (16/07/2018 - 20/07/2018).

Conference: Attending the "Intelligent Systems Conference (IntelliSys)", London, (05/09/2019 - 06/09/2019).

Poster: Participating in the AI round table event organised by the University of East London, the poster title is: Large-Scale Cyber-Physical Systems in the Built Environment: Real-Time Data Quality management, (15/11/2018).

Poster: Participating in UEL Excellence and Stuart Hall Scholarship awardees and posters competition, the poster title is: Smart Cities as Cyber-Physical Systems, (23/11/2017).

Word count for the main body of this thesis: \simeq 52K

Signed: _____

Date: _____

Chapter 1

Introduction

“A challenge for our community, ..., is: How can we design cyber-physical systems people can bet their lives on?.”

— Wing (2008)

This chapter is an introduction to cyber-physical systems and their implementations in smart cities’ large-scale applications. It also provides some insights into the research overall context, its aim, objectives and scope.

1.1 Cyber-Physical Systems

Cyber-physical systems (CPSs) are integrated systems engineered to combine computational and physical capabilities effectively using an embedded communication core (Törngren et al., 2017; Platzer, 2019). One of the earliest highlights to the emergence of the concept of ‘cyber-physical systems’ was by Helen Gill in the US National Science Foundation (NSF) Workshop on Cyber-Physical Systems in October 2006 in Austin, Texas (Lee, 2006; Greer et al., 2019).

Helen Gill defined CPSs as *“physical, biological, and engineered systems whose opera-*

tions are integrated, monitored, **and/or controlled** by a computational core. Components are networked at every scale. Computing is “deeply embedded” into every physical component, possibly even into materials. The computational core is an embedded system, usually demands real-time response, and is most often distributed. The behavior of a cyber-physical system is a fully-integrated hybridization of computational (logical) and physical action” (Gill, 2008b, p.3).

According to Sanislav & Miclea (2012)(p.28) "CPS integrates computing, communication and storage capabilities with monitoring **and/or control** of entities in the physical world, and must do so dependably, safety, securely, efficiently and real-time" (Sanislav & Miclea, 2012, p.28).

Based on the above definitions, it possible to highlight some of the fundamental characteristics of CPSs as follows:

Deeply coupled with their environment: CPSs are thoroughly connected with the physical world, CPSs monitor and control physical processes via physical components, such as sensors and actuators combined with cyber components, such as computer control algorithm where both components are connected using many other subsystems such as signal converters and network modules (Gumzej, 2018; Rawat et al., 2015). Generally, the physical world within the context of CPSs paradigm is characterised as follows:

- **Physical system** covers all engineered systems which ideally aligned with CPSs topology of real-time sensing, actuating and control e.g automated production lines and programmable controllers based power grids (Bordel et al., 2017; Gunes et al., 2014; Greer, 2014). CPSs utilises sensor nodes for collecting measurements from the physical system and send them as raw data to the control centre. Based on the received data, the control centre generates feedback and sends it to the actuators which regulate the physical conditions, as shown in Figure 1.1. This

cycle ultimately achieves the self-awareness of the system via its ability to assess and correctly adjust its behaviour and performance in real-time (Zhang et al., 2011; Kounev et al., 2018).

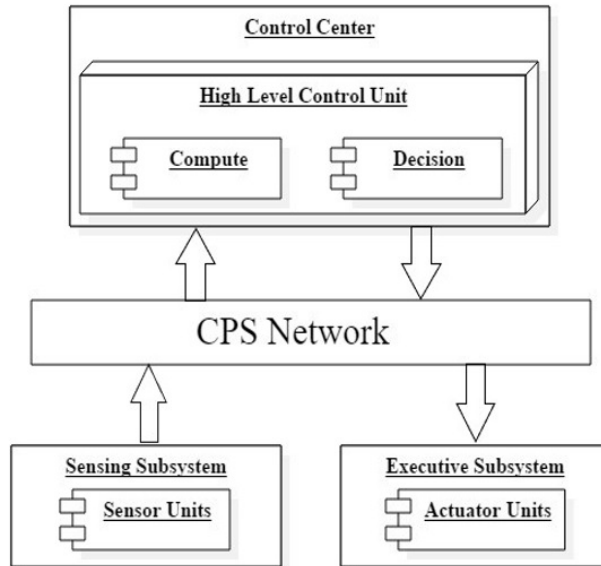


Figure 1.1: Cyber-Physical Systems structure diagram, (Zhang et al., 2011, p. 317).

- **Physical environment** in which CPSs monitor physical phenomena and depending on the application domain might generate feedback to control or support another sub-controlling system, such as in the case of environmental, health monitoring, infrastructure health monitoring systems (Bordel et al., 2017; Greer, 2014) and CPSs-based Building Information Management (BIM) systems (Smarsly et al., 2017). CPSs control (actuation) in such applications could be via physical actuators or could be through software element of control (virtual actuation) (Smarsly et al., 2017). CPSs are typically integrated into the physical environment via wired or wireless sensor networks while the actuators (physical actuators) affect the environment in form of automatic sprinkling pumps, alarms, and light, humidity or temperature regulators (Forster, 2016, p. 7).

Composed of heterogeneous entities: CPSs cover a wide range of applications

which have different degrees of complexity. e.g. a facility management cyber-physical system typically relies on few sensor nodes distributed in a building. In contrast, an environmental monitoring cyber-physical system usually relies on hundred of sensor nodes distributed over a vast geographical area (Gumzej, 2018; Chauhan et al., 2016).

Networked: communications and data exchange are deeply embedded in the design of CPSs. Network integration among the different components (sensor nodes, control unit(s) and actuators) of CPSs is an essential condition for proper functionality (Gumzej, 2018; Gill, 2008a).

Real-time systems: typically, CPSs monitor and adjust their status continuously in a nearly real-time fashion. CPSs real-time requirements depend on the type, size and the level of the system complexity (Hu, 2014, p. 81). In general, real-time systems are expected to function on a timely basis while keeping a significant small and acceptable timeline for response. Thus, the correctness of real-time systems is accounted for the accuracy and the timing for their outcomes (Möller, 2016, p. 97). According to Burns & Wellings (2001)¹ real-time systems can be categorised into:

- **Hard real-time systems (HRTSs):** systems which require responses within very restricted time deadlines, for example, aircraft's control systems, which may lead to a catastrophe in case of missing responses deadlines.
- **Soft real-time systems (SRTSs):** systems (or sub-systems) which are less restricted with responses deadlines, such systems can maintain a relatively normal functionality even after missing few deadlines. For example, data acquisition systems, which typically should collect sensor nodes observations according to a specified frequency, however, such

¹Pages 2-3, also available online: <https://www.cs.york.ac.uk/rts/books/RTSBookThirdEdition.html>

systems may tolerate some intermittent delays.

Cyber-physical systems is an umbrella term that covers many other disciplines associated with robotics, automation, industrial management systems, and the Internet of Things. All these systems share the same high-level functionalities of sensing, controlling and regulating physical processes based on decisions made by a computerised unit and using a communicate means (Fink et al., 2017, p. 133). CPSs and the Internet of Things (IoT) share the same definition and have a lot of common characteristics and many common applications. However, IoT emphasises Internet connectivity while CPSs do not necessarily require Internet connectivity (Törngren et al., 2017, p. 5), therefore, many of IoT projects can be considered as CPS applications (Minerva et al., 2015, p. 24).

Applications that involve a large number of sensor nodes and, or actuators deployed over a broad geographical territory are considered as large-scale CPSs (Rawat et al., 2015, p. 182, 228). For example, applications such as environmental monitoring systems, typically involve a wide network of wireless sensor nodes deployed over a vast geographical area forming a large-scale CPS (Benyuan Liu & Towsley, 2004; Mois et al., 2016; Sanislav et al., 2014; Ahmed et al., 2017). A sensor network may consist of a group of a few to thousands of specialised sensor nodes connected to each other or to an external server via wired or wireless medium (Sohraby et al., 2007, p. 15). *"A sensor network is an infrastructure consisting of sensing, computing, power source, and communication elements, which offer the ability to observe, instrument, and react to events and phenomena in a specified environment"* (Siddesh et al., 2015, p 106).

Wireless Sensor Networks (WSNs) were adopted successfully in environmental monitoring applications based on many small and self-powered sensor nodes connected by an ad-hoc wireless networking protocol (Fitriawan et al., 2017). WSNs is the result of the recent development of networked wireless technology

and the emergence of small, embedded, inexpensive, low-power consumption microprocessors which emerged as an essential mean for monitoring and exploring some phenomena in remote and harsh environments (Alhmiedat, 2015). According to Forster (2016) "*cyber-physical system (CPS) is a newer term for a wireless sensor network...when being integrated into a physical environment*" (Forster, 2016, p 7). The control action (actuation) of a sensor network encompasses three main tasks: coverage control, data source detection, and on request data collection (Cassandras, 2016; Lamnabhi-Lagarrigue et al., 2017).

Sensor nodes within WSNs are low-cost monitoring devices with one major drawback of their limited power capacity, which may limit their service lifetime (Antoo & Mohammed, 2014). Typically, WSNs are deployed in open remote environments. Unexpected measurements may occur due to external effects or internal malfunctions in the sensor nodes. In both cases, these unexpected measurements (anomalies) need to be carefully managed and interpreted based on domain knowledge discovery and computational models (Chen et al., 2018). More details about WSNs limitations and their impact on the quality of data in CPSs are outlined in the next chapter.

1.2 Smart Cities as Large-Scale CPSs

Smart cities are "*Cities using technological solutions to improve the management and efficiency of the urban environment*" (European Commission, 2021). A smart city uses modern technology in everyday urban life such as logistics, transport systems, security, safety, energy efficiency sustainability (Giffinger & Pichler-Milanović, 2007). Smart cities provide services based on collecting and analysing data from the environment using sensors, video cameras and social media means to support data-driven decisions related to how and when to take action by city operators or automatically (Cisco, 2020). Therefore, smart cities can be viewed as large-scale,

heterogeneous CPSs that utilise technologies like the Internet of Things (IoT), surveillance cameras, social media, and others to make informed decisions and drive the innovations of automation in urban areas (Zanni, 2015; Rhee, 2019).

CPSs are an active area of research (Ashibani & Mahmoud, 2017; Lohstroh et al., 2018), with a significant importance to the future of smart cities and the fourth industrial (Industry 4.0) revolution (Haseeb et al., 2019). CPSs are the next generation information systems that integrate communication, computation, and control to achieve higher performing buildings and better public services with more energy-efficient operations and a higher level of automation (Foehr et al., 2017). CPSs are multidisciplinary cross-domain systems that bring together different sectors of smart cities' public services, such as smart transportation management, smart utility management, smart buildings, smart environment management and smart governance, where data sensing, knowledge extraction, and higher automation are critical elements in the future of these services (Wu et al., 2016, p. 303), for example:

- **London Air Quality Network (LAQN)²:** A network of weather and air pollution monitoring devices mounted in cabins distributed in fixed site around London and southeast England to provide a long sequence of independent scientific measurements from the same location. LAQN measurements are used to assess air pollution across London and to track trends over time that support government policies, future planning, and research related to the effect of air pollution on health. The monitored air pollution parameters mainly cover Carbon Monoxide (CO), Nitrogen Dioxide (NO₂), Ozone (O₃), PM₁₀ Particulates, PM_{2.5} Particulates, and Sulphur Dioxide (SO₂) which are normally collected every 15 minutes to provide real-time indicators to other business bodies or public services provides such as Business Improvement Districts (BIDs), TfL and Defra (Environmental Research Group, 2021; De

²London Air Quality Network: <https://www.londonair.org.uk/LondonAir/Default.aspx>

et al., 2017).

- **Smart Santander:**³ proposes a unique city-scale experimental research facility in support of smart applications and services for a smart city. The project mainly supports advance large-scale CPS/IoT research utilising 20,000 sensors, 12,000 of which are deployed in Santander and the rest in Belgrade, Guildford and Lübeck. The sensor networks exploit a variety of urban disciplines using static (fixed) sensors such as environmental, traffic, parking slots and agriculture monitoring and cover a wide range of parameters such as temperature, pollution (CO), noise levels, light density, traffic volumes, road occupancy, vehicle speed, moisture, humidity and wind speed. The project supports mobile sensing via attaching sensors to public service vehicles such as public transport buses. The collected data were used in numerous experiments to enhance the quality of the monitored public services. Furthermore, the collected data were utilised in many augmented reality and participatory sensing applications (Smartsantander, 2021; De et al., 2017).
- **Barcelona Digital City**⁴: is a platform for technological innovation for the city services. The project benefits from a network of 500 kilometres of fiberoptic cables extended within the city utilised as a backbone network for 12 different large-scale CPS/IoT systems including transportation, water, waste and energy management and covering 83 smart applications across the city different urban disciplines. The project involves static (fixed) sensors such as cameras, infrared detectors, air quality sensors, irrigation and water levels monitoring sensors deployed to monitor public transport facilities, parks, traffic and pavement status. The collected data were also shared through an open-source web-based platform (Barcelona Ciutat Digital, 2021; LAURA, 2016; De et al., 2017).

³SmartSantander: <https://www.smartsantander.eu/>

⁴Barcelona Digital City: <https://ajuntament.barcelona.cat/digital/ca>

Most of these large-scale CPSs can be considered as heterogeneous and multi-model information systems which analyse a massive amount of data collected from various devices provided by different manufacturers (Barnaghi et al., 2015).

Typically, smart cities' large-scale CPS applications are designed to sense, process and react to changes in a real-time fashion (Hakiri & Gokhale, 2014). These systems rely on hundreds of sensor nodes and other devices which usually sense and stream readings of various parameters constantly producing a large volume of data known as Big Data (Badidi et al., 2018). The term Big Data describes a massive volume of complex and different types of structured and unstructured data that accumulate in a relatively high velocity (Kale et al., 2019, p. 5). Mining and analysing big data has a significant role in providing a rich source of data about smart cities' utilities and citizens' activities which provide more efficient management, better services and sustainable development (Bibri, 2018, p. 24).

CPSs implementations in different sectors of smart cities, public services are listed in Table 1.1, which also highlights big data and data quality management as common challenges across all of these large-scale CPS applications.

More details about data quality concepts, terminology, dimensions and challenges associated with large-scale CPSs are outlined in chapter-2.

Table 1.1: Cyber-physical systems cross-domain applications in the context of smart cities.

CPS applications / systems	Smart Environment	Smart Transportation	Smart Healthcare	Human activity / Smart spaces	Smart Governance
Smart utility management	✓(1) ⁵	✓(2)	✓(3)	✓(4)	✓(5)
Traffic and road management	✓(6) ⁶	✓(7)	-	✓(8)	✓(9)
Sensors and sensing technology	✓(10) ⁷	✓(11)	✓(12)	✓(13)	✓(14)
Energy management	✓(15) ⁸	✓(16)	-	✓(17)	✓(18)
Common challenges of large-scale CPS in smart cities					
Big Data management	✓(19) ⁹	✓(20)	✓(21)	✓(22)	✓(23)
Data quality management	✓(24) ¹⁰	✓(25)	✓(26)	✓(27)	✓(28)

⁵ 1-Včelák et al. (2017), 2-Mahmood & Zubairi (2019), 3-Goldberg & Zhang (2018), 4-Kim (2017), 5-Zhang et al. (2019).

⁶ 6-Naik et al. (2018), 7-Brincat et al. (2019), 8-Lin et al. (2020), 9-Naik et al. (2018).

⁷ 10-Liu et al. (2017), 11-Herrera-Quintero et al. (2018), 12-Bose et al. (2016), 13-Bonafini et al. (2019), 14-Santos et al. (2017).

⁸ 15-Bisadi et al. (2018), 16-Patel et al. (2018), - , 17-Walia et al. (2019), 18-Minoli et al. (2017).

⁹ 19-Andrés (2016), 20-Rathore et al. (2015), 21-Chen et al. (2018), 22-Lee et al. (2020), 23-Liu et al. (2012).

¹⁰ 24-Fang (2018), 25-Shukla et al. (2016), 26-Larburu et al. (2015), 27-Luo et al. (2019), 28-Lawson & Ramaswamy (2016).

1.3 Research Motivation

Cyber-Physical Systems (CPSs) are designed as a network of computational elements that combine physical input and output mechanisms to interact with the surrounding environment (Robbins & Tanik, 2013, p. 142). CPSs are getting more popular in the context of large-scale, smart cities applications which produce a significant amount of data from numerous devices raising quality-of-service concerns mainly related to real-time big data analysis and data quality management (Sta, 2019; R. et al., 2020; Kim et al., 2019). CPSs are data-driven decision-making system which optimises or control physical processes in the real world, therefore ensuring the quality of the data is crucial for CPSs successful operation. CPSs may cause significant consequences in case of interpreting faulty data, which impact the soundness of their decisions and thus reduce the quality of their service (Sha & Zeadally, 2015).

The quality of data in CPS applications is mainly affected by inaccurate observations that do not represent the real value of the measured phenomena (Geisler et al., 2016) and to delay in receiving observations especially in real-time applications (Bhargavi, 2016, p. 52). Data quality issues may occur in large-scale CPSs due to many reasons such as sensor nodes malfunctions (Labouseur & Matheus, 2017), calibration issues, poor sensor nodes quality, environmental effects, external noise (Okafor et al., 2020), networks or communication errors, and real-time scheduling problems (Sha & Zeadally, 2015; Kim et al., 2016). Furthermore, limitations in communication channels may cause observations' overlooking in sensor networks during data transmission or aggregation processes (Barnaghi et al., 2015; de Aquino et al., 2019).

The challenges of data quality management becomes greater in large-scale CPSs, e.g. in environmental monitoring systems, which rely on various sensors and other

devices connected by extended networks and usually operate under noisy and dynamic conditions (Lawson & Ramaswamy, 2016; Liu et al., 2014; Labouseur & Matheus, 2017). Such applications have enormous technical challenges due to their multiple layers and complex structure that comprises hardware, software, analytical algorithms, business knowledge and communication infrastructure (Togneri et al., 2019). Large-scale environmental monitoring CPS, typically, involve a large number of low-cost sensor nodes deployed in broad geographical terrains forming a large-scale Wireless Sensor Network (WSN) (Okafor et al., 2020; de Aquino et al., 2019; Abid et al., 2015). Failures in sensor nodes and sensor networks are an inevitable event in large-scale CPSs, which may cause severe data missing, produces invalid information and potentially reduce the quality of their service (Li et al., 2019). Sensor nodes in environmental monitoring CPS applications stream observations with various attributes such as temperature, humidity, wind speed besides other contextual attributes such as timestamp and locations coordinates. The impact of the occurrence of quality data issues with any of these attributes (faulty data) scales from making wrong data available to stakeholders in the best case in to tragic consequences in case of failing to support critical CPS decision in case of major environmental events such as floods, water pollution of forest fires (Drăgoicea et al., 2019).

In general, sensor nodes in WSNs have limited computing power, limited storage capacity and limited transmission radius (Lawson & Ramaswamy, 2016; Bhuiyan et al., 2017). Therefore, wireless sensor nodes can not send observations to a remote data destination (the sink) directly. Alternatively, a hub device or other sensor nodes works as a bridge to transfer other sensor node's observations. sensor nodes that are closer to the sink consume more power because they support other sensors to transmit their observations and are expected to have more power failures causing data quality issues (Liao et al., 2019; Togneri et al., 2019). Therefore sensor nodes may determine the network lifetime based on their battery capacity

and may impact the system's quality of information (Du et al., 2016).

Typically, wireless sensors nodes of WSNs are distributed according to a spatial or geographical logic over the targeted environment, (Bhajantri & Pundalik, 2017). Large-scale applications which exchange geographic information may face spatial data quality challenges mainly due to the amount of the delivered data from remote sensing devices which may directly affect the correctness of related spatial analysis and spatial decision making, (Bahl, 2015). Thus, data quality challenges are not only related to observations value attributes but also to mismatches in sensor nodes temporal and spatial contextual attributes (Togneri et al., 2019; Barnaghi et al., 2015).

As illustrated in Table 1.1, data quality challenges are associated with all major large-scale CPSs in the context of smart cities applications, e.g. smart environment monitoring systems (Andrés, 2016) , smart transportation (Rathore et al., 2015), smart healthcare (Chen et al., 2018), human activity and smart spaces management (Lee et al., 2020), and smart governance (Liu et al., 2012).

Based on the result of the systematic literature review illustrated in Chapter 2, it was possible to identify unaddressed data quality management challenges in large-scale CPSs which this research is bridging and for providing evidence to support the research questions as follows:

1.3.0.1 Sensor nodes' Measurement Errors Detection

The SLR primary studies which adopted prediction analysis models as data accuracy assessment techniques are sharing the following limitations:

1. All of the proposed prediction analysis models were based on an assumption that data accuracy issues occur for a short interval of time (point outliers). None of the SLR primary studies proposed a solution to address data ac-

curacy issues associated with long outliers. Long outliers change the time-series' pattern, so the inaccurate observations appear as the standard. In case, a time-series with long outliers is used as the predictive model training dataset. It will compromise the modes' ability to detect data accuracy issues correctly.

2. No systematic method or approach was demonstrated by any of the SLR empirical primary studies on how it was possible to ensure the quality of real-world dataset used to train or calibrate the predictive analysis model.
3. None of the SLR primary studies provided a comparison or a justification for why a particular predictive analysis technique was chosen over another. For example, it is not clear when to apply deep learning neural networks as a predictive technique (Krishna, 2018) instead of linear regression (Okafor et al., 2020).
4. SLR primary studies that investigated anomaly analysis as a solution to evaluate the accuracy of sensor nodes measurements by comparing their observations with different sensor nodes or to a pre-calculated threshold value were based on the assumption that these sensor nodes are spatially correlated. However, this assumption is not necessarily always valid in large-scale CPSs. The spatial continuity among sensor nodes in large-scale CPS applications might be compromised because of the vast distance separating these devices or other factors that disrupt the spatial continuity constraints, as detailed in Section 2.2.5.1 and Section 3.6.

1.3.0.2 Sensor nodes' and Sensors Networks' Failures Detection

The SLR primary studies provided no systematic method or a generic approach for detecting sensor nodes and sensor node networks hardware failures in large-scale CPSs. All proposed failure detection mechanisms were mainly domain-specific

solutions. For example, signal processing techniques were utilised for monitoring the hardware status of a Chinese network of weather radars by Togneri et al. (2019) which can not be applied as a generic solution for hardware failures detection in sensor node networks of large-scale CPSs.

1.3.0.3 Ensures the Quality of Observations' Spatial and Temporal Contextual Attributes

The SLR primary studies revealed that further research is required to address the challenge of ensuring the quality of sensor nodes contextual information of both spatial and temporal attributes. Spatial data quality issues (sensor nodes location) may affect the validity of any related spatial analysis. Furthermore, very limited or no research have practically investigated the possibility of using observations timestamp analysis techniques as a potential solution to improve the quality of sensor nodes spatial contextual information.

The systematic literature review protocol, methodology and data analysis results are detailed in Section 2.2.

1.4 The Research Aim

This research aims to:

To develop a comprehensive data quality management system for large-scale cyber-physical systems and empirically evaluate its validity.

Data quality management is a set of procedures and activities that aim to fulfil data quality requirements by continuously monitoring, measuring, and ensuring data **fitness for use**. The aim of this research, practically, is the development

of a data quality management system to detect data quality issues associated with errors in sensor nodes measurements, sensor nodes hardware failures that affect the quality of their measurements, and to detect mismatches in sensor nodes' observations spatial and temporal contextual attributes¹¹. Furthermore, to evaluate the proposed data quality management system using observations from real-world sensor node networks. The data quality management system must be comprehensive in the context of its ability to simultaneously detecting sensor nodes measurement errors associated with the main data quality dimensions of accuracy, timeliness, completeness and consistency in large-scale CPS applications.

1.5 The Research Questions and Objectives

As detailed in Chapter-2, Section-2.2, the systematic literature review has provided evidence to support the research questions (listed below), thus, a set of objectives were set to address these research questions and ultimately fulfil the aim of the research, as follows:

RQ-1: Is it feasible to develop a proof-of-concept data quality management system for large-scale CPSs that can; (1) detects sensor nodes measurements errors associated with the four main data quality dimensions of accuracy, timeliness, completeness, and consistency, (2) detects hardware failures in sensor nodes and sensors' communication networks and (3) ensures the quality of both spatial and temporal contextual attributes of sensor nodes observations. To address this question the following objectives were set:

Objective 1: To investigate data quality challenges in large-scale cyber-physical systems based on the literature and based on empirical data

¹¹The features of the proposed proof-of-concept data quality management system were determined based on the outcomes of a systematic literature review illustrated in Chapter 2

analysis of observations collected from a real-world, large-scale sensor node network.

Objective 2: To investigate data mining techniques that may support the research aim, such as predictive and anomaly analysis techniques, time-series and timestamp analysis techniques.

Objective 3: To construct, test and evaluate all the required models, components and tools of the proof-of-concept data quality management system to address the research aim. The data quality management system is expected to detect errors in sensors measurements, sensor nodes hardware failures, and mismatches in sensor nodes' spatial and temporal contextual attributes.

RQ-2: Is it possible to empirically evaluate the effectiveness of the proposed data quality management system using a real-world, large-scale sensor node network as a case-study?

Objective 4: To evaluate the effectiveness and performance of the proposed data quality management system utilising a real-world large-scale sensor node network as a case-study.

RQ-3: How to address bias concerns related to the evaluation process of the data quality management system, which emerges due to the presence of data quality issues in the testing or evaluating real-world dataset?

Objective 5: To validate the functionality and performance of the proposed data quality management system using a real-world, high-quality benchmark sensor node network. The benchmark sensor network must comprise high-quality sensors that stream consistent and error-free observations forming long time series of the same parameters collected from the large-scale sensor network.

1.6 Novelty of Research

1. This research delivered a novel, proof of concept, data quality management system which is capable of evaluating sensor nodes' measurements based on the four dominant data quality dimensions, it detects sensor nodes' hardware failures and ensures the quality of observations' spatial and temporal contextual attributes in large-scale CPSs. Such a system can be utilised as a data quality assessment mechanism with compatible industrial or smart-cities scale CPS or IoT applications.
2. This research is an empirical study that utilises a real-world large-scale environment monitoring sensor node network consists of over 200 ambient temperature sensors distributed around London to support its outcomes and conclusions and further validate the robustness of the proposed data quality management system. Furthermore, it brings together advance predictive analysis, spatial partitioning, time-series clustering and timestamp analysis techniques which were successfully utilised to support the aim of this research. Therefore, this research can be used as an academic reference for future research conducted to address emerging data quality management challenges in the context of large-scale CPS applications.
3. This research is one of the very few studies that deliver an empirical data quality assessment solution based on advanced data science and machine-learning models while systematically addressing the bias concerns related to the evaluation process of the used models, which emerges because of the presence of data quality issues in the testing or evaluating real-world dataset. Thus in this research, a high-quality sensor node network was deployed at the University of East London and utilised to produce long, consistent, high-quality data streams of benchmark observations to train and adjust the

different data quality assessment models and to evaluate the performance and accuracy of these models before using them in real-world scenarios.

4. This research is an initiative empirical study that practically addresses data quality challenges associated with the contextual spatial and temporal attributes of sensor nodes observations in large-scale CPSs. It is one of the very few studies that provides and empirically validates a systematic approach for detecting inconsistency in temporal attributes and mismatches in the spatial attributes of sensor nodes' observations in the context of large-scale CPSs.
5. This research provides more insights and empirically exploits many of the large-scale CPSs features, e.g. this research has shown empirically that sensor nodes which are located near to each other (high-density areas) are highly likely to be spatially correlated. In contrast, distant sensor nodes (low-density area) might not fit in any spatial cluster or may form a cluster with other distant sensor nodes which violate their spatial continuity and thus verifying the accuracy of their observations using spatial correlation-based anomaly detection techniques may not render a reliable assessment result.

1.7 Research Boundaries

The following topics are out of the scope of this research:

1. The purpose of this study is to evaluate sensor nodes observations fitness for use in the context of CPSs not to test or evaluate a particular CPS.
2. In this research data quality management was investigated using a relatively low-frequency large-scale environmental monitoring system as a case study. The monitoring system consists of over 200 sensor nodes distributed around

London. The duty cycle of these sensors is around 10 minutes or longer. Therefore, none of the components of the proposed data quality management system was evaluated or tested for higher frequency (shorter duty cycle) CPS applications.

3. This research focuses on data quality management of static large-scale CPSs, where sensor nodes are fixed in specific geographic locations identified in their observations spatial attributes and do not change with time. Mobile CPS applications such as smart vehicles, or smart-wearables that involve mobile sensor nodes that inquire new location with time are out of this research's scope.
4. Investigating the mathematical foundation of the adopted data mining or data analysis models is out of the scope of this research.
5. Implementing technical solutions such as parallel computing, high-performance distributed computing or cloud computing that can provide higher computational power to boost the performance of the data quality management system is out of the scope of this research.

1.8 Research Structure

Chapter-1 is an introduction to cyber-physical systems and their implementations in smart cities large-scale applications. It also provides some insights into overall context of the research, its aim, objectives and scope.

Chapter-2 is an introduction to data quality and the main challenges associated with it in large-scale CPS applications. It incorporates a systematic literature review which analysis the existing literature to draw meaningful conclusions related to data quality challenges in large-scale CPS and identifies the gaps in the current literature that this research is bridging.

Chapter-3 is to describe the research context. It describes the research design, the structure of the data quality management system, data analysis methods, the techniques used to address the research objectives and it explains the adopted logical sequence to conduct the research activities.

Chapter-4 the implementation details and results of testing and evaluating the different components and models of the data quality management systems are presented in this chapter.

Chapter-5 revisits the research questions and illustrates the extent to which these questions were satisfied. It provides more insights into the key components of the data quality management systems. Finally, it outlines the conclusions of the research and future work.

Chapter 2

Literature Review

“... the growing complexity of the I/T infrastructure threatens to undermine the very benefits information technology aims to provide.”

— Horn (2001)

This chapter is an introduction to data quality and the main challenges associated with it in large-scale CPSs. It incorporates a systematic literature review which analyses the existing literature to draw meaningful conclusions related to data quality challenges in large-scale CPSs and identifies the gaps in the current literature that this research is bridging. Furthermore, the systematic literature review provides evidence to support the research questions and establishes the theoretical background to the techniques or methods being used to achieve the research objectives.

This chapter covers the following topics:

- Data quality concepts and terminology.
- Data quality challenges in large-scale CPSs.
- Data mining and data quality management.
- Unaddressed data quality management challenges and research questions.

2.1 Data Quality Concepts and Terminology

Data quality management is a set of procedures and activities aim to fulfil data quality requirements by continuously monitoring, measuring, and ensuring the data fitness for use (Mosley et al., 2009, p. 23-27). According to the International Standardization Organisation (ISO), **quality**, in general, is the "the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs" (ISO 8402, 1994, p. v). In comparison, **Data Quality (DQ)** is defined as data **fitness for the purpose of the intended use** (Juran et al., 1988; Maydanchik, 2007) or its **conformance to requirements** (Crosby, 1979; Batini & Scannapieco, 2016). This definition outlines that data which very well meet some predefined expectations, specifications, or standards are considered to be high-quality data that **fit for use** in a particular application. The concept of fitness to use associated with data quality is also covers how effectively the data describe any events, observations or measurements it was created to represent, where the characteristics of the data are linked to the system used to collect these events (Sebastian-Coleman, 2013).

Data quality can be quantified, measured and monitored using a set of context-dependent parameters or indicators known as Data Quality Characteristics (DQCs) or Dimensions(DQDs) (Wang et al., 1993; Sebastian-Coleman, 2013). More than 200 data quality dimensions have been introduced since the eighties (Guillet & Hamilton, 2007, p. 106). However, these dimensions can be categorised into four core data quality dimensions; accuracy, completeness, timeliness and consistency (Sebastian-Coleman, 2013; Guillet & Hamilton, 2007; Scannapieco et al., 2005; Wand & Wang, 1996), which are ,typically, associated with data quality requirements and mapped to define data quality assessment criteria (Fürber & Hepp, 2010, p. 147-148).

2.1.1 Accuracy

Accuracy is defined as how close observations are to reality (Nguyen et al., 2014). It presents the "degree of conformance of observations and computations with the truth" (Brimicombe, 2009, p. 215). The ISO 5725-1 (1994) defined the term accuracy as the precision and trueness of measurements or test results, as follows:

- **Trueness** is the closeness of agreement between the true value of a measured parameter in comparing with an accepted reference value. Thus, the true value of some measurements can not be known exactly. Alternatively, a reference value of the amusement is accepted. The reference value can be adopted from other measurement methods or selected from a known sample.
- **Precision** is the closeness of agreement between the results of multiple independent tests obtained under specified and controlled conditions. Precision is the distribution of random errors between measurements which can be presented by the Standard Deviation (SD), where higher SD indicates higher **imprecision**. The notion of considering precision is significant in every measurement procedure due to the inherited and unavoidable attribute of random errors which influences the results of measurements and cannot be controlled (ISO 5725-1, 1994).

Accuracy, typically, is the most inspected attribute associated with data quality issues (Kenett & Shmueli, 2016, p. 31), especially in the context of identifying and avoiding measurement errors, where errors are the deviation of observations from the reality and can be in the form of sudden short outliers, systematic or random bias (Brimicombe, 2009, p. 215).

Measuring the accuracy of data requires comparing it to a real-world reference entity that itself was validated for accuracy. The reference entity can be from an outside, but a related source that the measurements can be validated directly

against. If a direct comparison between parallel measurements in large-scale datasets is not possible, then a reference criteria value (threshold value) can be used to validate the accuracy of them. However, in this case, it is possible to identify measurements (observations) that are probably inaccurate but can not determine to what extent inaccurate are they (Sebastian-Coleman, 2013, p. 62-64). In both cases, accuracy characterisation, which is based on the measurements value attribute is known as the "structural-accuracy" or, plainly, "accuracy". However, in real-time systems those collect observations of real-life phenomena, accuracy is not only associated with data quality issues related to the value attribute of observations, but also with the rapidity in which observations are reflecting the reality, which is known as time-related accuracy (Batini & Scannapieco, 2016, p. 23). More about the time-related accuracy is in the next section.

2.1.2 Time-Related Accuracy (Timeliness)

In general, the term "timeliness" is used to refer to the Time-Related Accuracy (TRC) data quality dimension which covers data characteristics associated with time aspects, such as currency, volatility and timeliness (Batini & Scannapieco, 2016, p. 27).

- **Currency** is the degree in which the measurements are current with reality. It defines how up to date the data are, based on an expected update rate or an estimated frequency. Currency is an initial parameter which defines the lifetime limits of data, and it may require manual or automatic techniques to be measured and verified (Loshin, 2011, p. 140).
- **Volatility** is the frequency in which the data is expected to change. Volatility is a critical parameter to evaluate the validity of data, and it is closely associated with currency and timeliness (Sebastian-Coleman, 2013, p. 62). For example, the volatility of particular data such as birth dates is equal to zero

since such data do not change with time. In contrast, the volatility of the attribute value of age is annual, while the volatility of the attribute value of the atmospheric temperature is continuous, which means it remains valid for a relatively short time (Mahanti, 2019).

- **Timeliness (latency)** is the data attribute which implies whether a recorded value is out of date to be useful for a particular usage (Ballou & Pazer, 1985). Data that still available within a specified threshold time frame are timeliness data (Berti-Équille, 2007, p. 118).

The timeliness of real-time data can be very short due to the rapid change (high volatility rate) of data, adding more strain on real-time processing technologies which must analyse the data before it becomes outdated and invalid (Cai & Zhu, 2015). Timeliness is an essential data quality constraint in real-time event processing applications which continuously streams observations and supports time-critical and automated decisions (Aggarwal, 2013, p. 79). Furthermore, timeliness events in information systems can be used as a reliability indicator of the data transmission mechanisms (Sebastian-Coleman, 2013, p. 63). Thus, timeliness is a crucial notion in CPSs which typically must close the loop of sensing, computing and control within a specific processes time frame in order to function properly (Rawat et al., 2015, 147).

According to Ballou et al. (1998), the timeliness of events can be measured based on quantifying the currency and volatility aspects of data using a unified time unit. The measurement of the currency is a function based on the time-related metadata¹ typically associated with data value attributes

¹In the context of CPS in this research, sensor's data value attributes were referred to as observations, while observation's occurrence time or their database receiving time were referred to as time-tags or timestamps.

and defined as the following:

$$\text{Currency} = (\text{DeliveryTime} - \text{AcquisitionTime}) + \text{Age} \quad (2.1)$$

- **AcquisitionTime**: when the data of a real-world event were obtained, e.g. database record's timestamps of newly entered sensor nodes' observations.
- **DeliveryTime**: the time when the data arrived at its usage destination, which can be a computing unit, a digital entity or a customer.
- **(DeliveryTime - AcquisitionTime)**: how long the data have been available in the system.
- **Age**: how old the observations when they were received. Age is the time difference between observations occurrence time and when these observations entered the system, as shown in equation 2.2 (Ballou et al., 1998, p. 468).

$$\text{Age} = (\text{AcquisitionTime} - \text{ObservationTime}) \quad (2.2)$$

It is possible to measure data currency by applying equation 2.1 to the time-related metadata attributes, typically using the most recent metadata. However, this approach is only valid if the frequency (duty-cycle) of data updates is constant. In contrast, in information systems that do not have a fixed data updating frequency, it is possible, e.g., to measure the currency based on the average of data updating frequencies which may exhibit some errors. Other methods can be used based on the characteristics of the information system and the domain application requirements (Batini & Scannapieco, 2016, p. 27-28).

The second data quality criteria required to measure timeliness is volatility.

Volatility is the frequency in which the data is expected to change. Thus, practically, volatility is the inverse of the time interval in which the data still valid. Therefore, timeliness is defined as (Ballou et al., 1998, p. 468):

$$\text{Timeliness} = \max \left\{ 0, 1 - \frac{\text{currency}}{\text{volatility}} \right\} \quad (2.3)$$

Equation 2-3 illustrates that the value of timeliness varies from (0 poor) to (1 excellent). Furthermore, high volatility data are more sensitive to currency comparing with low volatility data which are less sensitive to currency.

2.1.3 Completeness (Completeness)

In general, the data quality completeness dimension is the measure of the presence of all necessary values of a specific variable to complete a particular process (Ballou & Pazer, 1985; Wang & Strong, 1996, p. 32). According to Lemahieu et al. (2018)² and Pipino et al. (2002), completeness can be characterised based on the following:

- Column completeness: at the data unit level, e.g. data records, observations, objects, column completeness is defined as the degree (ratio function) of the existence of missing values in a data column (Pipino et al., 2002). A column is considered to be complete if it has no missing values at all time (Bühmann et al., 2006, p. 188).
- Population completeness: the percentage of the presence of all data entities of a particular reference population (Rula et al., 2016, p. 103), for example, if a dataset contains three of the year seasons only, it considered as a dataset with population incompleteness since the reference population, typically, consists of four seasons.

²Pages 82-84

- Schema completeness: the degree in which structures and attributes of a real-world information system or application domain are covering all schema³ (metadata) features of data (Ehrlinger & Wöß, 2018, p. 23).

Completeness, typically, achieved when all necessary data elements are available to complete a particular process (Jugulum, 2014, p. 34), and can be measured for each of the three types (mentioned above) as the ratio of the number of the incomplete or missing data points, (readings, records, observations or objects) to the number of the expected ones subtracted from 1 as shown in equation 2.4.

$$\text{Completeness (C)} = 1 - \frac{\text{Missing}_{n \text{ data}}}{\text{Expected}(\text{reference})_{n \text{ data}}} \quad (2.4)$$

Equation 2.4 is limited to measure the completeness dimension of static data which do not change with time in regards to completeness or dataset snapshot at a particular point in time.

Pernici & Scannapieco (2003) introduced the notion of **completability** which defines completeness in temporal dynamic information systems that exchange data continuously and their data dynamically evolve with time. Completability (C) is estimated as a function with time (t), as shown in equation 2.5.

$$C = \int_{t_{\text{curr}}}^{t_n} C(t) \quad (2.5)$$

Equation 2.5 measures the completability at the instance t , where t is between t_{init} and t_n , $t \in (t_{\text{init}}, t_n)$, t_{init} is the initial time instant of the collected data. And t_{curr} is the time when the completability was evaluated, t_n is the time of the next expected data update(s) where $t_n > t_{\text{curr}}$.

³A schema in information systems describes data structures, relationships and features which are also known as metadata (Ehrlinger & Wöß, 2018, p. 1)

2.1.4 Consistency

Consistency is how well data comply with integrity constraints, logical rules or specific context without contradictions (Batini & Scannapieco, 2016, p. 23). The consistency can be measured based on the comparison in relation to data from other instances or systems which were produced under similar conditions or using the same production process. Measuring consistency may involve finding logical connection patterns within the data in real-world scenarios while focusing on the consistency of data over time (Sebastian-Coleman, 2013, p. 63).

Sha & Shi (2008) have proposed six consistency models for providing different consistency evaluation levels for data streams from Wide-Networks Sensors (WSNs). However, they argued that these consistency models are application-specific concepts, and they implied that examining the quality of data streams from Wireless Sensor Networks (WSNs) should be mainly based on the trend, frequency and spatial consistency.

Inconsistency in sensor's observations in CPSs is a common data quality issue that may occur among functional or faulty sensor nodes due to many reasons such as unreliable wireless communication, external inference and noise. Therefore, observations redundancies from reliable sensor nodes should be considered to ensure data availability in CPS applications (Li et al., 2016). This approach of using a set of high-quality, reliable sensor nodes as a benchmark for data availability and consistency evaluation was adopted and examined in this research. More details about data quality dimensions and real-world, large-scale CPSs will be detailed in the next sections.

2.2 Data Quality Challenges in Large-Scale CPSs, a Systematic Literature Review

This section incorporates a Systematic Literature Review (SLR) in which the literature systematically aggregated, examined and analysed to: investigate data quality challenges in large-scale CPSs, to identify the most common techniques used to address these challenges and assess the overall effectiveness of these techniques. SLR is defined as an objective and unbiased method for aggregating, analysing and extracting knowledge from the available literature in relation to specific research questions or a given topic using a set of well-defined and repeatable procedures. SLR's are considered as secondary studies synthesised from individual primary studies, such as peer-reviewed research papers, to summarise existing evidence or to identify any gaps in current research for further investigation (Kitchenham et al., 2015, p. 10-11).

The SLR approach was implemented to avoid some of the limitations of the traditional (unsystematic) narrative review approach. According to Green et al. (2006)⁴, unless the researcher is an expert in the field, it is critical to use a systematic approach to produce quality literature reviews. Moreover, to avoid the common drawbacks of the narrative overviews such as subjective selectivity or biased interpretations of the primary studies (Efron & Ravid, 2018, p. 21).

As a result, the SLR was essential to identify the gaps in the current literature concerning data quality management in large-scale CPSs that this research is bridging and for providing evidence to support the research questions. Furthermore, the SLR approach helps to establish the theoretical background of the techniques or methods used to achieve the objectives of this research.

⁴Pages 103-104

2.2.1 Review Motivation / Introduction

CPSs are designed as a network of computational elements that combine physical input and output mechanisms to interact with the surrounding environment (Robbins & Tanik, 2013, p. 142). CPSs can be seen as large and/or heterogeneous Embedded Computing Systems (ECSs)⁵ with high computation and communication capabilities that perform dedicated functions, typically, according to strict real-time constraints (Möller, 2016; Jahromi & Kundur, 2020, p. 7).

CPSs rely on data acquisition from sensor nodes, data processing in the control (computing) unit(s) and data communication with the actuators to regulate the physical environment. This data cycle is necessary for CPSs to meet their operational requirement and ultimately enables the system's self-control and awareness, especially in real-time applications (Pan et al., 2019; Zhang, 2015; Rawat et al., 2015, p. 140). Data, typically, circulate continuously among the different components of CPSs in real-time (Tao et al., 2018, p. 160). Therefore, data has a crucial role in the successful operation of CPSs (Rawat et al., 2015, p. 141), especially considering that CPSs may cause severe consequences in the case of providing decisions based on low-quality data (Sha & Zeadally, 2015; Williams & Tang, 2020; Vaidya et al., 2018).

CPSs might compromise safety constraints and might have life-threatening consequences in cases of receiving incorrect data, missing time deadlines or missing critical readings from sensors in real-time (Grega & Kornecki, 2015, p. 755). Ensuring the quality of data is an open challenge in large-scale CPSs (Farooqi et al., 2018; Peng et al., 2019; Prathiba et al., 2016; Shih et al., 2016; Perez-Castillo et al., 2018) mainly due to the large amount of data that these systems exchange at (near)

⁵An ECS is a specialised physical system controlled by computer-based components which fully encapsulated as core elements in the ECS system itself. The development of ECS, which was associated with computing networking solutions, has led to the emergence of Cyber-Physical Systems as the new generation of information systems (Möller, 2016, p. 94).

real-time, the vast geographical area, and the dynamic and noisy conditions where these systems are usually deployed (Barnaghi et al., 2015, p. 6:3).

This systematic literature review is incorporated to analyse the existing literature to draw meaningful conclusions related to data quality challenges in large-scale CPSs and to provide evidence to support the research questions and establishes the theoretical background to the techniques or methods being used to achieve the research objectives. The next section provides more details about the SLR review process, questions, methodology and results.

2.2.2 Review Process and Methodology

This systematic literature review (SLR) was conducted based on the guidelines proposed by Kitchenham & Charters (2007), which provides an organised and repeatable procedure to perform the SLR based on three main stages: planning, conducting and reporting the review results. A holistic view of the processes adopted to conduct this SLR is illustrated in Figure 2.1.

2.2.2.1 SLR Questions and Objectives

The SLR review questions (RQs) have a significant role in driving the review methodology and identifying the primary studies. Thus, the analysis and synthesis process of the primary studies must extract the data in a way that answers the review questions (Kitchenham & Charters, 2007, p. 9). The SLR review questions are listed in Table 2.1.

Table 2.1: SLR Review Questions and Objectives.

RQ#	SLR Research Question / Objectives
RQ1	What are the most common data quality challenges associated with large-scale CPS applications?
RQ2	Which solutions/methods were adopted to address data quality challenges in large-scale CPSs?
RQ3	What are the unaddressed data quality management challenges in large-scale CPSs?

2.2.2.2 SLR Protocol (Strategy)

The Review protocol is the strategy of implementing a set of specified steps to undertake the SLR. The purpose of the SLR protocol is to narrow down the

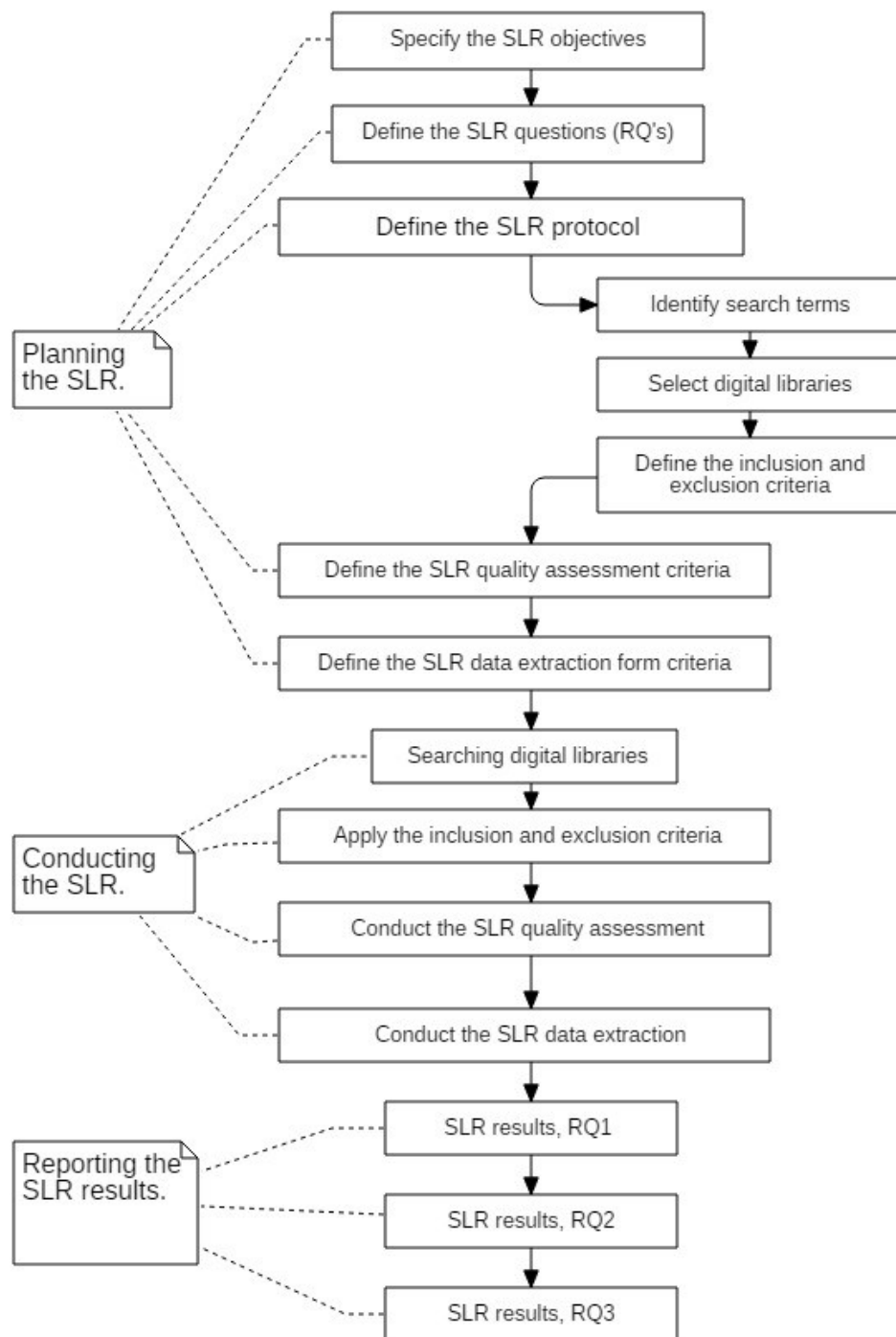


Figure 2.1: A holistic overview of the processes adopted in this systematic literature review.

possibility of researcher bias by pre-defining the review processes and procedures of selecting and analysing the primary studies that will address the research questions. The review protocol involves specifying the research terms (keywords), digital libraries, refinement terms (synonyms for the main search terms), the

quality questionnaire and the data extraction forms (Kitchenham & Charters, 2007; Malhotra, 2015, p. 40-45), as follows:

- **Identifying Search Terms**

Digital libraries must be searched using search terms and keywords to identify the primary studies that will address the review questions. The search terms typically are extracted from the search questions, including any possible alternative terms or synonyms as shown in Table 2.2.

Table 2.2: The SLR Search Terms and Keywords.

Category	Search terms	Level of Abstraction
Primary terms	Quality of Data Data Quality Quality of Information Information Quality	Title and abstract
Secondary terms	Cyber-Physical System Internet of Things Wireless Sensors Network	
Exclusion terms	Social Media Smart-Wearables Vehicle Services Social Sensing	Fully reviewing the study

The search method is based on incorporating the keywords and terms from Table 2.2 using Boolean expressions (OR, AND, NOT...etc) to form Boolean search string, which used to search the pre-selected digital libraries.

Using the term “large-scale” in the Boolean search string has significantly reduced the number of the primary studies rendered by the searched digital libraries furthermore, using it with an OR clause did not affect the outcome of the search results, therefore the term “large-scale” was excluded from the SLR search terms and keywords. However, all the selected studies in this SLR are exploring, proposing, or presenting a data quality management/assessment solution in large-scale CPS/IoT where the term “large-scale” was not used but the study is focusing on a particular large-scale CPS/IoT

application. For example, (Farooqi et al., 2018), (Li et al., 2019) and (Peng et al., 2019) have been selected after applying the inclusion and exclusion criteria and after being fully viewed, none of these studies use the term “large-scale” but still all presents large-scale CPS applications.

- **Selecting Digital Libraries**

Selecting digital libraries is an essential step for identifying relevant primary studies that will address the research questions. It is critical to include many digital libraries in the search process since no signal source can comprehensively provide all relevant primary studies and to ensure resource-dependent search to cover the search topic. The SLR initially included all of the digital libraries listed in Table 2.3.

Table 2.3: The list of the examined digital libraries to be used for the purpose of the SLR.

Interface	Search Screen	Database
EBSCO	https://search.ebscohost.com/Community.aspx?	Academic Search Complete
		British Education Index
		Business Source Complete
		Education Abstracts (H.W. Wilson)
		Educational Administration Abstracts
		Information Science and Technology Abstracts
		Education Research Complete
IEEE Xplore	http://ieeexplore.ieee.org/search/searchresult.jsp	IEEE Xplore digital library
Elsevier	https://www.sciencedirect.com/science/search	ScienceDirect
ProQuest	https://ebookcentral.proquest.com/	eBook Central
ProQuest	https://search.proquest.com/advanced/	ProQuest
ACM	https://dl.acm.org	ACM Digital Library
SAGE	http://journals.sagepub.com/action/doSearch?	SAGE Journals
Emerald	http://www.emeraldinsight.com/action/doSearch?	Emerald Insight
IET	https://digital-library.theiet.org	IET Digital Library

However, most of these digital libraries did not render any significant outcome after being searched using the keywords and terms listed in Table 2.2 combined with Boolean expressions and after applying the inclusion and exclusion criteria listed in Table 2.5. Therefore, the most relevant digital

libraries that some of its rendered primary studies passed the inclusion and exclusion criteria are listed in Table 2.4. Where Table 2.4 shows the list of the most relevant digital libraries searched for identifying the SLR primary studies.

Table 2.4: The list of the most relevant digital libraries used for identifying the SLR primary studies.

ID	Digital Library		Online Search Interface
1	IEEE	IEEE Xplore	https://ieeexplore.ieee.org
2	ACM	ACM Digital Library	https://dl.acm.org
3	IET	IET Digital Library	https://digital-library.theiet.org
4	Elsevier	ScienceDirect	https://www.sciencedirect.com

- **Defining Inclusion and Exclusion Criteria**

The purpose of the inclusion and exclusion section is to define the criteria of selecting which primary studies will be approved for further analysis while excluding other studies that do not satisfy these criteria. The inclusion and exclusion criteria of the SLR are listed in Table 2.5.

Table 2.5: The SLR Inclusion and Exclusion Criteria.

The inclusion criteria are:
- The study is categorized as a peer-reviewed journal and conference paper relevant to the SLR topic and addresses one or more of its review questions.
- The study is relevant to large-scale CPSs or IoT applications.
The exclusion criteria are:
- The study is: an editorial, tutorial, magazine, book, course, poster or it is not a peer-reviewed journal.
- The focus of the study is related to mobile CPSs or IoT.
- The study is written in a different language other than English.
- The full version of the study is not available.
- The study is published before 2014.
- Duplicated studies.

The SLR digital libraries were searched based on the primary studies title and abstract which are typically written in English even if the research paper itself is

written using a different language. The search results did not render any studies which are written in a language other than English. However, since there was no professional interpreter assigned to this research it is important to highlight that the SLR was conducted while focusing mainly on primary studies which are written in English.

The SLR search was limited to studies that were published during and after 2014 because expanding the search to include studies that were published before 2014 did not significantly change the SLR digital libraries search outcome. Furthermore, limiting the search to studies that were published during and after 2014 helped to focus the investigation on the most recent emerging data quality challenges in large-scale CPSs.

2.2.2.3 SLR Quality Assessment

The purpose of the SLR quality assessment is to evaluate the relevance of primary studies that already met the inclusion criteria to the review topic. SLR quality assessment is crucial because it is a further measure to limit the possibility of researcher bias (Kitchenham et al., 2015, p. 81), it presents a repeatable guideline for interpreting the results, and it provides a quantitative numeric mean to determine how strongly the selected primary studies are associated with the SLR objectives via a quality score. Typically, the SLR quality assessment can be implemented by scoring individual primary studies using a quality questionnaire form and based on assessment criteria (Malhotra, 2015, p. 42-43). The SLR primary studies were scored based on the quality assessment questions listed in Table 2.6.

2.2.2.4 SLR Data Extraction Form

The data extraction form summarises and extracts information from the primary studies to answer the review questions. It specifies which primary study addresses

Table 2.6: The SLR Quality Assessment Questions (Matrix).

Q#	Quality assessment questions associated with data quality challenges in large-scale CPS's?	Quality Score		
		Yes	Partially	No
Q1	A review or an empirical study?	1	n/a	0.5
Q2	Is the study combines multi-methods/techniques to address data quality challenges?	1	0.5	n/a
Q3	Is the study justifies the use of these different methods/techniques?	1	n/a	0
Q4	Is there any comparative analysis of the different used methods/techniques?	1	n/a	0
Q5	How many data quality issues associated with the four core data quality dimensions are the study addressing?	No. of Dimensions		
		4	3 - 1	n/a

which of the SLR review questions, analyse the results and identifies the primary study strengths and weaknesses. The structure of the data extraction form used in this SLR is as follows:

- Citation details.
- Study purpose/application.
- Dataset types/details.
- Targeted data quality dimensions.
- Addressed data quality challenges.
- Proposed solutions/methods.

2.2.3 Review Conduct and Primary Studies Selection

The SLR review was conducted using the pre-defined structure highlighted in the review process and methodology section and based on the three main steps: selecting, evaluating and summarising the primary studies as follows:

2.2.3.1 Searching Digital Libraries

The first step to implement the SLR processes and methodology was to identify relevant primary studies by searching the digital libraries listed in Table 2.4 using

search strings developed based on the keywords and terms as specified in Table 2.2 and as shown in Table 2.7⁶.

Table 2.7: The SLR digital libraries, final search strings and the number of identified primary studies.

ID	Digital Library	Action	Boolean Search Strings (10/10/2020)	No. of Papers
1	IEEE Xplore	Search string	((("Document Title": "cyber physical system" OR "internet of things" OR "wireless sensors network") AND "Document Title": "data quality" OR "quality of data" OR "quality of information"))	376
		Filter	Publication Type (Conferences, Journals), Publication Topics (learning (artificial intelligence) Internet of Things data analysis data mining wireless sensor networks Big Data decision making pattern classification data handling optimisation pattern clustering information systems quality of service statistical analysis) Published between (2014 and 2020)	
2	ACM Digital Library	Search string	[Publication Title: "data quality"] OR [Publication Title: "quality of data"] OR [Publication Title: "quality of information"] AND [Publication Date: (01/01/2014 TO 12/31/2020)]	91
		Filter	Published between (2014 and 2020)	
3	IET Digital Library	Search string	("data quality" OR "quality of data" OR "quality of information") AND ("cyber physical system" OR "internet of things")	52
		Filter	Published between (2014 and 2020)	
4	Science Direct	Search string	Articles with these terms ("cyber physical system" OR "internet of things" OR "wireless sensors network") and Title ("data quality" OR "quality of data" OR "quality of information")	23
		Filter	Review articles, Research articles, published between (2014 and 2020)	

2.2.3.2 Applying the Inclusion and Exclusion Criteria

The next step is to determine whether the identified primary studies satisfy all of the pre-defined inclusion and exclusion criteria, listed in Table 2.5. The number of the primary studies included in the SLR after applying the inclusion and exclusion criteria is shown in Table 2.8.

⁶ There are some slight differences among the Boolean search strings used to search different digital libraries, these differences are related to the design of the search interface of the digital libraries and the availability of the primary studies.

2.2.3.3 Conducting the SLR Quality Assessment

The quality assessment (as highlighted in Section 2.2.2.3) is a crucial step to evaluate the relevance of primary studies and scoring them according to the assessment matrix specified in Table 2.6 where the relationship among the quality assessment questions is shown in Equation 2.6.

$$\text{Quality Score} = Q1 + Q2 + Q3 + Q4 + Q5 \quad (2.6)$$

Although the quality assessment of the primary studies does not answer the review questions, it provides an opportunity to understand the trend of most recent studies concerning large-scale CPSs data quality management, Figure 2.2 and their geographical distribution of interest where Figure 2.3 shows the number of SLR studies by the country of publication.

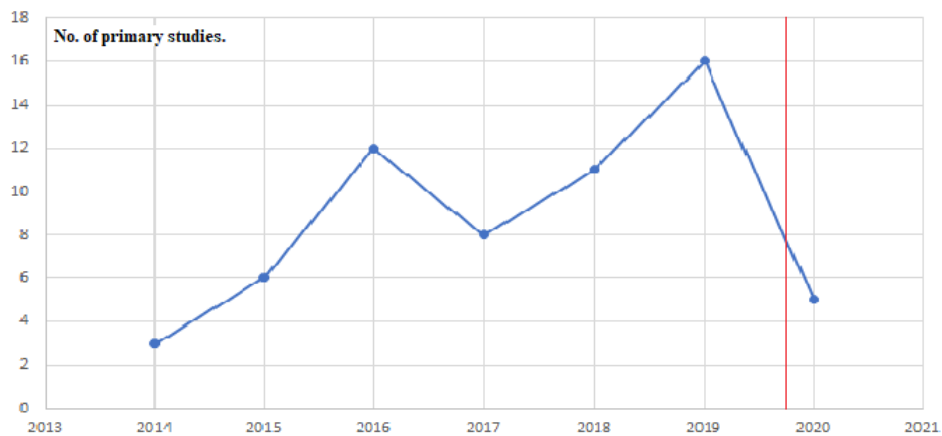


Figure 2.2: The number of SLR primary studies by the year of publication, (October 2020).

The final number of the primary studies included in the SLR after applying the inclusion and exclusion criteria and after fully reviewing all studies is detailed in Table 2.8. The primary studies citation details and their overall quality assessment score are shown in Table 2.9.

The total number of the primary studies included in the SLR after searching the

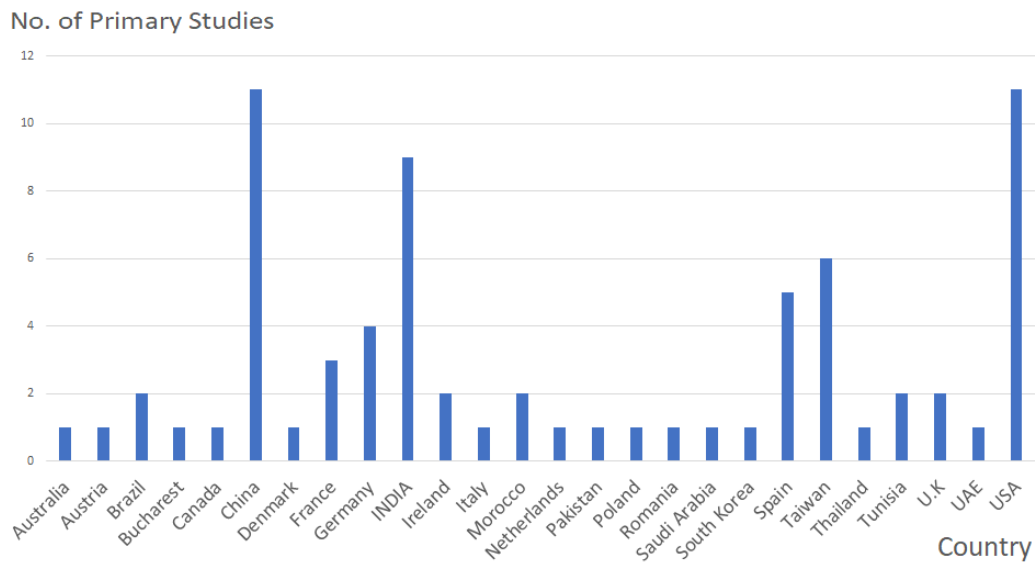


Figure 2.3: The number of SLR primary studies by the country of publication.

Table 2.8: The final number of primary studies included in the SLR after applying the inclusion and exclusion criteria and fully reviewing all studies.

Activity \ Digital Library	IEEE	ACM	IET	Science Direct	Total
Searching digital libraries and applying filters	376	91	52	23	542
Reviewing titles and abstracts	78	26	6	8	118
Fully reviewing all studies	40	13	4	3	60

most relevant digital libraries listed in Table 2.4 using the Boolean search strings listed in Table 2.7 and after applying the inclusion and exclusion criteria is **542** as shown in Table 2.8.

The total number of the primary studies was reduced to **118** after filtering the studies based on reviewing their title and abstract. For example, the case of the Zhiping et al. (2014) primary study which was eliminated based on its title "A novel authentication protocol for mobile nodes in multi-base-station wireless sensor network" which states that the research paper is focusing on mobile nodes which is an elimination criterion as listed in the SLR inclusion and exclusion criteria Table 2.5.

Furthermore, the SLR primary studies were filtered to **60** after full reviewing

all of the 118 studies. For example, the title "Energy-aware quality of information maximisation for wireless sensor networks" of the study by Du et al. (2016) suggests that its focus on enhancing the quality of the information in wireless sensor networks using energy-aware means. However, fully reviewing the research paper revealed that it investigates how to enhance the reliability of wireless sensor networks by reducing the power consumption of its sensors using an optimising algorithm that reduces sensors sampling rate and consequently reduces their internal transmit power consumption. The study does not address any data quality dimension and does not enhance the quality of the information in WSNs. It proposes a trade-off between the quality of information and the energy expenditure in wireless sensor networks using an optimisation technique and thus it was excluded.

2.2.3.4 SLR Data Extraction

The purpose of the data extraction process (as highlighted in section 2.2.2.4) is to quantitatively summarises the information from the primary studies to answer the SLR review questions. Table 2.10 shows the results of the data extraction process.

Table 2.9: Primary Studies Referencing Details and their Overall Quality Assessment Score.

Ref.	Study Identifier	Year	Research type	Approach	Q1	Q2	Q3	Q4	Q5	Score
S1 - Farooqi et al. (2018)	IEEE Conferences	2018	Solution proposal	Framework	1	0.5	1		1	3.5
S2 - Li et al. (2019)	IEEE Conferences	2019	Solution proposal	Method	1	0.5	1	1	1	4.5
S3 - Karkouch et al. (2015)	IEEE Conferences	2015	Review	Guideline	0.5	0.5			1	2
S4 - Peng et al. (2019)	IEEE Conferences	2019	Solution proposal	Framework	1	0.5			1	2.5
S5 - Karkouch et al. (2016)	IEEE Conferences	2016	Solution proposal	Framework	1				2	3
S6 - Kim et al. (2019)	IEEE Journals	2019	Solution proposal	Model	0.5					0.5
S7 - Liao et al. (2019)	IEEE Conferences	2019	Solution proposal	Method	1				1	2
S8 - Perez-Castillo et al. (2018)	IEEE Conferences	2018	Solution proposal	Framework	0.5					0.5
S9 - Larburu et al. (2014)	IEEE Conferences	2014	Solution proposal	Model	0.5	0.5	1		3	5
S10 - Du et al. (2016)	IEEE Journals	2016	Solution proposal	Algorithm	1				1	2
S11 - Rager et al. (2018)	IEEE Journals	2018	Solution proposal	Framework	1				2	3
S12 - Lawson & Ramaswamy (2016)	IEEE Conferences	2016	Solution proposal	Framework	1	0.5			4	5.5
S13 - Liu et al. (2014)	IEEE Journals	2014	Solution proposal	Framework	1	0.5			2	3.5
S14 - Kim et al. (2016)	IEEE Conferences	2016	Solution proposal	Model	1	0.5			1	2.5
S15 - Togneri et al. (2019)	IEEE Conferences	2019	Solution proposal	Framework	1	0.5	1		1	3.5
S16 - Shrivastava et al. (2019)	IEEE Conferences	2019	Solution proposal	Tool	1	0.5	1		1	3.5
S17 - Micic et al. (2017)	IEEE Conferences	2017	Review	Guideline	0.5				1	1.5
S18 - Auger et al. (2016)	IEEE Conferences	2016	Solution proposal	Tool	1	1				2
S19 - Chidean et al. (2016)	IEEE Journals	2016	Solution proposal	Model	1	0.5			1	2.5
S20 - Bahl (2015)	IEEE Conferences	2015	Solution proposal	Framework	0.5				1	1.5
S21 - Auger et al. (2017)	IEEE Conferences	2017	Review	Guideline	0.5					0.5
S22 - Prathiba et al. (2016)	IEEE Conferences	2016	Review	Guideline	0.5					0.5
S23 - Karmakar et al. (2020)	IEEE Journals	2020	Solution proposal	Model	1				1	2
S24 - Bhuiyan et al. (2017)	IEEE Journals	2017	Solution proposal	Method	1	0.5			1	2.5
S25 - Ghosh et al. (2019)	IEEE Conferences	2019	Review	Survey	0.5	1				1.5
S26 - Krishna (2018)	IEEE Conferences	2018	Solution proposal	Tool	1	0.5			3	4.5
S27 - Al-Milli & Almobaideen (2019)	IEEE Conferences	2019	Solution proposal	Algorithm	1	0.5	1		1	3.5
S28 - Xinrui et al. (2019)	IEEE Conferences	2019	Solution proposal	System	1	0.5			2	3.5
S29 - Jayswal & Shukla (2016)	IEEE Conferences	2016	Review	Guideline	0.5	0.5	1			2
S30 - Bhajantri & Pundalik (2017)	IEEE Conferences	2017	Solution proposal	Model	0.5					0.5
S31 - Mylavarapu et al. (2019)	IEEE Conferences	2019	Solution proposal	Tool	1	0.5			1	2.5
S32 - Pełech-Pilichowski (2018)	IEEE Conferences	2018	Evaluation research	Guideline	1	0.5			1	2.5

Table 2.9 continued from previous page

Ref.	Study Identifier	Year	Research type	Approach	Q1	Q2	Q3	Q4	Q5	Score
S33 - Abid et al. (2015)	IEEE Conferences	2015	Solution proposal	Tool	1	0.5			1	2.5
S34 - Pattanavijit et al. (2015)	IEEE Conferences	2015	Solution proposal	Method	1	1	1	1	1	5
S35 - Zhou et al. (2018)	IEEE Conferences	2018	Solution proposal	Model	1				1	2
S36 - Tomescu et al. (2019)	IEEE Conferences	2019	Solution proposal	Tool	1				1	2
S37 - Hanrong Lu et al. (2016)	IEEE Conferences	2016	Solution proposal	Method	1	0.5	1		1	3.5
S38 - Puiu et al. (2016)	IEEE Journals	2016	Solution proposal	Framework	1					1
S39 - Giacobbe et al. (2018)	IEEE Conferences	2018	Solution proposal	Method	1				1	2
S40 - Abid et al. (2017)	IET Journals	2017	Solution proposal	Method	1	0.5			1	2.5
S41 - Sta (2019)	IEEE Conferences	2019	Solution proposal	Model	0.5					0.5
S42 - Nesa et al. (2018)	IEEE Conferences	2018	Solution proposal	Framework	1	0.5			1	2.5
S43 - Barnaghi et al. (2015)	ACM Journals	2015	Review	Guideline	0.5				3	3.5
S44 - Schelter et al. (2018)	ACM journals	2018	Solution proposal	System	1	0.5			3	4.5
S45 - Sha & Zeadally (2015)	ACM Journals	2015	Review	Guideline	0.5					0.5
S46 - Labouseur & Matheus (2017)	ACM Journals	2017	Review	Guideline	0.5					0.5
S47 - Glowalla & Sunyaev (2014)	ACM Journals	2014	Review	Guideline	0.5					0.5
S48 - Geisler et al. (2016)	ACM Journals	2016	Solution proposal	Framework	1	0.5				1.5
S49 - Jain et al. (2020)	ACM Conference	2020	Review	Survey	0.5					0.5
S50 - Januzaj et al. (2019)	ACM Conference	2019	Solution proposal	Method	1					1
S51 - Guo et al. (2018)	ACM Conference	2018	Solution proposal	Model	1				1	2
S52 - Liu et al. (2019)	ACM Conference	2019	Solution proposal	Model	1	0.5			2	3.5
S53 - Zemicheal & Dietterich (2019)	ACM Conference	2019	Solution proposal	Method	1	0.5			1	2.5
S54 - de Aquino et al. (2019)	ACM Journals	2019	Solution proposal	Method	1					1
S55 - Shih et al. (2016)	IET Journals	2016	Review	Guideline	1	0.5			1	2.5
S56 - Auger et al. (2017)	IET Journals	2017	Review	Guideline	0.5				2	2.5
S57 - Wang & Bu (2020)	IET Journals	2020	Solution proposal	Method	1	0.5		1	1	3.5
S58 - R. et al. (2020)	SD Journals	2020	Solution proposal	Method	1				1	2
S59 - Song et al. (2017)	SD Journals	2017	Solution proposal	Model	1					1
S60 - Okafor et al. (2020)	SD Journals	2020	Solution proposal	Model	1	1	1	1	1	5

Table 2.10: Results of the SLR data extraction process addressing data quality main challenges and the proposed solutions in large-scale CPS.

Ref.	Purpose/application	Dimensions	Data quality challenges	Proposed solutions/methods
S1	Weather data quality control.	Accuracy	Automatic verification of data quality, data integrity and scalability in weather data.	Improving the accuracy of data using machine learning models based on the Random Forest Prediction method (Random Forest Regression), which reduces overfitting without increasing the ratio of error.
S2	Data quality enhancement in power terminals.	Completeness	Sensors and sensor networks failures are inevitable events in power IoT systems, which may cause severe data missing.	A one-step forward forecasting model based on the autoregressive-moving-average (ARMA) algorithm was implemented for detecting and mitigating the impact of missing data.
S3	An overview of data outliers detection process.	Accuracy	Improving data quality, focusing on data accuracy in the context of IoT applications.	Outlier detection for enhancing the quality of data more efficiently and systematically in IoT environments.
S4	An anomaly analysis platform to monitor the quality of data in ubiquitous power IoT.	Accuracy	Monitoring the quality of data of ubiquitous power IoT platform considering its high data exchanging rate, diversity of components and the absence of any effective data management mechanism.	Anomaly detection based on the isolated forests integrated unsupervised machine-learning algorithm. For training the ML model, the historical data was reconstructed to form a time series using the sliding time window model.
S5	A data quality reporting framework using graphical editors and models.	Accuracy, Completeness	Data quality is a subjective concept that varies by the purpose or the intended use of the data. There are no standard criteria to define high-quality data which typically diverse in measure attributes and requirements.	A Model-Driven Architecture (MDA) framework developed by Object Management Group (OMG) for software development. It initially developed for data quality management in the context of web applications.
S6	A process-centric framework to improve the quality of streamed sensors data.	-	Improving the quality of data in IoT applications which rely on real-time data streaming sensors and have different data structures.	A proposed data quality management framework based on the Process Reference Model (PRM) which only suitable for offline applications with a well-defined process.

Table 2.10 continued from previous page

Ref.	Purpose/application	Dimensions	Data quality challenges	Proposed solutions/methods
S7	A mechanism to optimise data collection process in WSN while maintaining the level of the quality of information (IoT).	Timeliness	Improving the quality of information by reducing observations delay and enhance the data lifetime in WSN networks. Improving the reliability of WSN and extending its lifetime by reducing its power consumption rates.	A proposed data transmission path planning mechanism named the Energy Harvesting Path Planning Strategy. It manages observations travel path from sensor nodes to the network sink.
S8	A framework for managing data quality in smart connected product (SCP) / IoT environments.	-	The open challenges in SCP/ IoT applications are: data quality standardisation, data quality management especially for applications that collect a significant volume of data from different sources.	A guideline for improving data quality management in SCP environments aligned with ISO/IEC 25012 characteristics and proposed an IoT model based on ISO 8000–62 including the processes of part 8000–61.
S9	A computational model for clinical data quality assessment.	Accuracy, Timeliness, Consistency (Dependability)	Improving telemedicine systems technological context to become data quality-aware systems.	A computational model to assess the quality of context data based on optimising the end-to-end resource configuration chain.
S10	An algorithm to improve the QoI in WSNs	Accuracy	Improving the lifetime of WSNs, enhancing its data transmission rate while maintaining the quality of information (QoI).	Using the proximal optimisation approach (algorithms), which enhances the performance metrics of WSNs.
S11	A QoI framework for WSNs, focusing on completeness and timeliness.	Completeness, Timeliness	Scalability and performance prediction in WSNs concerning the QoI requirements.	Top-K algorithm was adopted for evaluating data completeness metric. Top-k is an image selection algorithm which was implemented to address the non-linear relationship of data completeness with throughput.
S12	A cloud-service framework for optimising the quality of data streams in real-time WSNs.	Accuracy, Timeliness, Completeness, Consistency	Investigating the quality of data of remote environmental sensors data streams in relation to energy efficiency in WSNs.	A cloud-service framework for optimising the quality of data streams in WSNs while assessing their energy efficiency in real-time. The proposed framework dynamically modify and regulate sensors to maintain data quality and energy-efficient operation in WSN.
S13	Energy management of environmental sensors while maintaining the QoI constraints.	Accuracy, Timeliness (latency)	Efficient energy management of environmental monitoring sensors while maintaining the quality-of-information (QoI) in a multitask-oriented environment.	An energy management service compatible with sensors lower layer protocols and over-arching applications, based on signal propagation and processing latency modelling.

Table 2.10 continued from previous page

Ref.	Purpose/application	Dimensions	Data quality challenges	Proposed solutions/methods
S14	Enhancing the quality of the information in real-time decisions-based IoT.	Accuracy (value)	To enhance the quality of the information in real-time decisions-based IoT applications which bring many safety and security challenges related to real-time scheduling problems comparing to traditional applications, especially in data processing and smart devices management.	A scheduling model was proposed to enhance the quality of the information in applications that need multiple data items to make decisions based on quality adjustment algorithms and scheduling policies.
S15	Data quality assurance in IoT applications	Accuracy (value)	Providing higher data quality assurance in regards to data completeness (availability) and consistency(integrity) of IoT sensors data, which usually affected by sensors failures.	Anomaly detection using the Local Outlier Factor algorithm to identify sensors failures and mismatch in sensors spatial contextual information.
S16	Data quality advisor solution for large-scale IoT.	Accuracy (value)	Developing an interactive, large-scale sensors data quality advisor for large-scale, IoT Applications.	A data quality framework that automatically performs data validations. The core of the framework is based on the Direct Acyclic Graph(DAG) model for data quality checks and Scalable Execution Engine (SEE) for executing the validation function.
S17	A data quality assessment framework for heterogeneous data resources.	Accuracy	Meeting the expectations of data accuracy and reliability in large sensor networks is a significant challenge due to the heterogeneous nature of engineering data.	Data quality of sensors observations which form long time series can be examined using outlier detection and trend analysis. However, this approach does not address the challenges of checking and analysing a system of sensors network or a realm of heterogeneous time series simultaneously.
S18	QoI assessment as a service platform for smart cities applications.	Accuracy, Timeliness	Developing an autonomic, collaborative, extensible and configurable solution to cope with the challenge of QoI assessment within smart cities sensing platforms.	The study proposes an Information Quality Assessment solution as a Service (iQAS) based on measuring data attributes such as accuracy and timeliness using filtering and prediction mechanisms for a given application.
S19	Energy efficiency and data quality improvement in large-scale WSNs.	Accuracy	Increasing energy efficiency in WSNs without sacrificing the quality of data.	The study proposes a model for enhancing energy-efficiency in large-scale WSNs by controlling the number of sensors transmissions using the second-order data coupled clustering (SODCC) and the compressive projections principal component analysis (CPPCA) algorithms.

Table 2.10 continued from previous page

Ref.	Purpose/application	Dimensions	Data quality challenges	Proposed solutions/methods
S20	Addressing spatial data quality concerns.	Consistency	Addressing spatial geometric inconsistency and topological inconsistencies in geographic information systems.	A proposed framework for correcting the inconsistency in spatial data based on the Triangular Pyramid Framework for spatial analysis.
S21	An overview of the challenges of sensor streams in large-scale IoT applications.	-	Addressing the Quality of Observation (QoO) challenges between IoT sensors and their observations destination.	The study proposes a cloud-based IoT platform for collecting, processing and delivering sensors observations.
S22	A review of data quality issues in WSNs.	-	The study specified four data quality challenges in WSNs; synchronisation issues, inefficient testing of algorithms, energy management and the lack of novel mathematical modelling.	The study discussed the existing data quality and fault tolerance techniques in WSNs.
S23	Sensors data trust in IoT applications using temporal correlation.	Accuracy	Assessing the trust of sensors data in large-scale IoT applications.	A model for assessing trust in large-scale IoT sensors data using a temporal correlation-based approach and adopting Deep Neural Networks (DNN).
S24	Data quality of event-sensitive monitoring in vibration sensor networks.	Accuracy (value)	Ensuring the quality of data in vibration data-intensive monitoring applications which must deliver high-resolution observations accurately and continuously to the system processing core.	A decentralised control and data reduction algorithm utilising the Goetzal algorithm to address data quality challenges in event sensitive WWSN applications.
S25	A review of outlier detection techniques in WSNs in IoT frameworks.	-	Addressing data quality checking techniques in wireless sensors networks.	The quality of data in large-scale IoT frameworks can be examined using machine learning techniques such as neural Networks, clustering and classification for being powerful methods to detect outliers in sensors data.
S26	IoT reliability in sensors networks systems.	Accuracy, Consistency, Reliability	Optimising sensors coverage and reducing energy consumption in IoT sensing networks.	The study proposes a model which uses the minimum set cover theorem for identifying reliable sensor nodes with more extended sensing sequence of observations, higher accuracy rate and consistency per sensing region to facilitate optimal coverage.
S27	Addressing the issue of missing data in medical IoT applications.	Accuracy	Developing a prediction model for imputing missing data in IoT applications.	The study proposed a prediction model for detecting and estimating missing data in IoT applications using deep learning neural networks.

Table 2.10 continued from previous page

Ref.	Purpose/application	Dimensions	Data quality challenges	Proposed solutions/methods
S28	Data quality control of the Chinese wind radars' network.	Accuracy, Completeness	Ensuring data accuracy and conventional functionality of a large-scale wind radars' network.	The data quality evaluation and detection mechanisms are mainly based on statistical techniques such as standard deviation, correlation coefficient and data acquisition rate of the observation collected from the wind profile radar network.
S29	Reviewing different clustering techniques for detecting outliers in data streams.	Accuracy	Outliers detection in streamed data due to its high-speed, non-stationary, large volume, and attributes diversity comparing to static data sets.	The study concluded that clustering has a fundamental role in data streams mining possess for outliers detection, especially the basic density-based clustering (DBSCAN) algorithms.
S30	Improving the quality of data of WSNs semantic information.	-	Improving the quality of semantic row data in WSNs, and improving sensors spatial and temporal ontology.	A model for providing semantic sensor data through a Semantic Sensor Web (SSW) services to enhance the quality of sensors semantic data using data integration and fusion techniques.
S31	A big data accuracy assessment tool.	Accuracy	Developing a big data quality assessment tool.	The study proposes a data accuracy assessment tool based on machine learning (K-Nearest Neighbors, Logistic Regression and Decision Trees) and Apache Spark for handling large-scale datasets.
S32	Inconsistency analysis in large-scale, non-stationary and inconsistent time series.	Consistency	Interpolation of missing/insufficient data in real-world, large time-series.	The study outlined four different time-series interpolation/predictions methods for short-term statistical time-series analysis using the one step ahead prediction and the moving data window approach.
S33	Anomaly detection based on spatial distribution data in WSNs.	Accuracy of spatial attributes	Detecting abnormalities based on spatial distribution data of sensor nodes and using numerical data outlier detectors in WSNs.	K-nearest neighbours algorithm (KNN) and Euclidian distance were adopted to detect abnormalities from the spatial distribution of data and depending on WSNs Low Energy Adaptive Clustering Hierarchy protocol (LEACH).
S34	Controlling the quality of data in large-scale water-level monitoring system.	Accuracy	Developing a solution to replace DBSCAN (Density-Based Spatial Clustering of Applications with Noise) with a more efficient higher performance clustering algorithm.	A linear-clustering algorithm was developed to replace DBSCAN for data quality control in a large-scale, water-level monitoring system. The experimental results indicated that the performance of the proposed domain-specific outlier detection algorithm is higher than DBSCAN.

Table 2.10 continued from previous page

Ref.	Purpose/application	Dimensions	Data quality challenges	Proposed solutions/methods
S35	Missing data estimation/replication in industrial WSNs.	Availability	Enhancing data availability in the presence of sensor nodes failures in industrial WSNs (IWSNs).	The proposed solution is based on utilising sensor nodes memory space to save measurements from their neighbouring nodes and carry the last observation forward to estimate missing data. This approach is limited to time series with stable trend.
S36	A monitoring system for large-scale sensors networks.	-	Automatically monitoring the infrastructure of large-scale sensors networks (124 stations) deployed over vast geographical terrain (20 sq. km).	The proposed solution is based on a rule engine which reads the system's parameters and compares them against pre-calculated threshold values.
S37	Duplicate records detection in real-world applications.	Duplication	Detecting and cleaning duplicated records to ensures the quality of data and maintains applications performance.	A genetic neural network-based approach for detecting duplicated records.
S38	QoI framework for smart cities applications.	-	Meeting information quality requirements for smart cities scale data analysis applications.	The study proposes a large-scale data analysis framework to provide near real-time machine-interpretable data for smart cities applications. The proposed framework considered many quality measures and fault recovery techniques to enable quality-aware and up-to-date smart city applications.
S39	Evaluating the QoI in IoT as a service in a smart cities scale applications.	Accuracy	Enhancing public information assets using advanced methods to support public administrations services.	The study proposes a quality of information evaluation strategy based on Multiple Criteria Decision Making (MCDM) methods in the context of evaluating the quality of public data and related metadata in the scale of smart cities applications.
S40	Outliers detection in WSNs.	Accuracy	Ensuring the quality of data through outlier detection for identifying intrusion, errors and noise in wireless sensor networks applications.	Density-based outlier detection technique was evaluated using DBSCAN as outlier detection technique for systems with expected normal behaviour.
S41	Data quality evaluation in a large-scale transportation system.	-	Addressing the problem of real-time data analysis and handling imperfections in sensors data of smart cities IoT applications.	Proposing a data integration platform from different sources to interpret the information certainty level using an evidential database based on the evidence theory.

Table 2.10 continued from previous page

Ref.	Purpose/application	Dimensions	Data quality challenges	Proposed solutions/methods
S42	Density-based clustering for outlier detection in WSNs.	Accuracy	Improving the quality of information via outlier measurements, mainly by detecting errors, noise and failures in wireless sensor networks.	A modified density-based spatial clustering of applications with noise (DBSCAN)-OD algorithm was developed based DBSCAN algorithm in order to detect computing and spatial-temporal parameters to identify outliers from standard sensors.
S43	A guideline for data quality challenges in smart cities.	-	Identifying the main data quality challenges in smart cities applications especially issues related to wireless networks energy restrictions, sensors bandwidth or connectivity limitations or for challenges associated with the large data volumes, high data velocity, dynamicity or diversity of types and structures.	The study classified data quality issues in smart cities scale applications into three main types: measurements or precision errors in sensor nodes, external noise or network communication errors and integrity of sensors observations in both spatial and temporal dimensions.
S44	Automating large-scale data quality verification.	Accuracy, Consistency, Completeness	Verifying the quality of data against missing or incorrect information.	The study proposes an automated data quality verifications system. The proposed system adopts a declarative API to combine standard data quality constraints with user pre-defined validation rules and leverages machine learning for anomaly detection using data predictability approach based on historical time series.
S45	A guideline to the main data quality challenges in CPSs.	-	The most significant challenge in CPSs is identifying and filtering faulty data.	The study highlights the need for developing novel algorithms and protocols that can effectively detect and filter erroneous data in CPS applications such as faulty data and information loss models, localized algorithm and the lightweight secure data storage and transmission protocols.
S46	An introduction to dynamic data quality challenges.	Accuracy	Ensuring the quality of dynamic data in IoT applications which typically generated by multi-vendors devices, micro services, automated processes and different types of sensors.	The study highlights that maintaining the quality of dynamic data in IoT applications is an open challenge which provides new research opportunities.
S47	A review of process-driven data quality management.	-	Developing a broadly applicable process-based model for improving and sustaining the quality of data.	The study concludes that further representational analysis is required to enhanced process modelling language for process-driven data quality management (PDDQM) modelling.

Table 2.10 continued from previous page

Ref.	Purpose/application	Dimensions	Data quality challenges	Proposed solutions/methods
S48	Data quality management for data streams.	-	Maintaining the quality of the data stream without affecting the real-time performance of the system.	The study proposes an ontology-based data quality monitoring framework based on the characteristics of relational data stream management to observe data quality values and take counteractions to balance the performance.
S49	A survey about the importance of high-quality data for machine learning.	-	Enhancing the performance and accuracy of machine learning models by ensuring the quality of their training dataset.	The study concluded that researches were focusing on improving the quality of machine learning models. In contrast, insufficient works were conducted to improve the quality of data in the context of its value for machine learning applications.
S50	Distributed data mining for identifying data quality issues.	-	Facilitating data quality analysis of data in their distributed state.	The study proposed a data quality issues identifier based on the knowledge extracted from pre-clustering data in its distributed status. The experimental results showed comparable results with those conducted on the integrated warehoused data.
S51	Data quality challenges in large industrial environment.	Consistency	Addressing the challenge of data inconsistency to enhance the quality of data in large industrial data environment.	A proposed mathematical data quality assessment and monitoring model based on data cleaning, duplicated records detection and traditional data sorting and merging methods.
S52	Data quality assessment for electrical data.	Accuracy, Completeness	To address the challenge of data quality assessment for electricity consumption big data.	The study proposes a model that addresses six data quality assessment indexes including accuracy, completeness and comprehensiveness using time-relevant k-means to detect outliers in voltage curves.
S53	Data quality control for weather data.	Completeness (missing values)	Improving the accuracy of weather data which can be degraded by missing sensors readings.	The study evaluated five strategies for detecting failed sensors and statistically identifying anomalies; Mean imputation, MAP imputation, Reduction, Marginalization and Proportional distribution and concluded that missing values handling algorithms can significantly enhance the reliability of weather systems.
S54	Data quality assessment in smart sensor networks.	-	Smart Sensor Networks (SSNs) rely on sensors with limited resources and usually deployed in remote and harsh environments which impose data quality challenges in IoT applications.	The study proposes a mechanism to reduce memory and network communication overhead and to impose networks delay.

Table 2.10 continued from previous page

Ref.	Purpose/application	Dimensions	Data quality challenges	Proposed solutions/methods
S55	A review of the challenges associated with designing large-scale CPS/IoT applications.	Timeliness	Five main challenges oppose the development of CPS/IoT applications in smart cities applications; middleware development, computation models, fault tolerance, data quality management, and a virtual run-time environment.	The study examined a correlation model among sensors using readings from different sensors to calibrate or verify another sensor's observations when the data are missing.
S56	A survey about the quality of observations within sensors web systems.	Accuracy, Timeliness	Addressing the challenge of ensuring the quality of observations in sensors webs which represent the middleware layer between sensors and applications.	The study identified essential requirements for developing the future adaptive quality of observations aware sensor web solutions including standardisation, the need for a layer-based architecture, mediation, adaptation and reconfiguration.
S57	Enhancing situation awareness in renewable power systems.	Accuracy	Developing situation awareness system for power systems, that can accurately detect anomalies and robust against multiple data corruptions.	This study tackles two primary challenges faced by conventional situation awareness in power systems: 1) accurately detect anomalies using aggregation of random matrix and long short-term memory network. 2) To be robust against multiple data corruptions using a dedicated workflow designed to decrease the impact of data corruptions.
S58	Faulty data detection in cyber-physical systems.	Timeliness	Detecting and filtering faulty data efficiently to improve the quality of the collected data from a system's perspective.	The study proposes an automatic reliability improvement framework of three data quality assessment stages performed on the system input, output and feedback data using machine learning, and operator in the loop approach for detecting faulty-data and improving the reliability of the system.
S59	Data quality management for manufacturing cyber-physical systems.	-	Developing effective managerial policies for controlling the quality of the data generated by improper operations of physical and cyber components of a service-oriented manufacturing CPS.	The study proposes a two-stage optimization model for data quality management of service-oriented manufacturing CPS (SMCPS). Formal semantics of workflow nets (WF-nets) algorithm together with a two-stage optimization model were used to find the optimal policies that balance the system's objectives.

Table 2.10 continued from previous page

Ref.	Purpose/application	Dimensions	Data quality challenges	Proposed solutions/methods
S60	Improving the quality of data of low-cost IoT environmental monitoring networks.	Accuracy	Identifying the main factors that affect the data quality (accuracy) of low-cost IoT sensors in environmental monitoring networks.	The study investigated the use of artificial neural network and linear regression for calibrating low-cost environmental monitoring sensors to improve the accuracy of their readings. These devices are vulnerable to environmental factors such as temperature and humidity; therefore, it is necessary to take these parameters into account when developing the calibration model. The results demonstrated the importance of feature selection process in optimising multi-parameter calibration models.

2.2.4 RQ1: Data Quality Challenges in Large-Scale CPSs.

This section is to answer the first SLR review question (RQ1) listed in Table 2.1.

Cyber-Physical Systems (CPSs) are designed as a network of computational elements that combine physical input and output mechanisms to interact with the surrounding environment (Robbins & Tanik, 2013, p. 142). CPSs are getting more popular in the context of large-scale, smart cities applications which produce a significant amount of data from numerous devices raising quality-of-service concerns mainly related to real-time big data analysis and data quality management (Sta, 2019; R. et al., 2020; Kim et al., 2019).

The quality of data in CPS applications is mainly affected by inaccurate observations that do not represent the real value of the measured phenomena (Geisler et al., 2016). Data quality issues may occur in large-scale CPSs due to many reasons such as sensor nodes malfunctions (Labouseur & Matheus, 2017), calibration issues, poor sensor nodes quality, environmental effects, external noise (Okafor et al., 2020), networks or communication errors, and real-time scheduling problems (Sha & Zeadally, 2015; Kim et al., 2016). Furthermore, limitations in communication channels may cause observations' overlooking in sensor networks during data transmission or aggregation processes (Barnaghi et al., 2015; de Aquino et al., 2019).

The challenges of data quality management becomes greater in large-scale CPSs, e.g. in environmental and noise monitoring systems, which rely on various sensors and other devices connected by extended networks and usually operate under noisy and dynamic conditions (Lawson & Ramaswamy, 2016; Liu et al., 2014; Labouseur & Matheus, 2017). Such applications have enormous technical challenges due to their multiple layers and complex structure that comprises hardware, software, analytical algorithms, business knowledge and communication

infrastructure (Togneri et al., 2019).

Large-scale CPS applications, such as environmental monitoring systems, typically, involve a large number of low-cost sensor nodes deployed in broad geographical terrains forming a large-scale Wireless Sensor Network (WSN) (Okafor et al., 2020), (de Aquino et al., 2019; Abid et al., 2015). Failures in sensor nodes and sensor networks are an inevitable events in large-scale CPSs, which may cause severe data missing, produces invalid information and potentially reduce the quality of their service (Li et al., 2019).

In general, sensor nodes in WSNs have limited computing power, limited storage capacity and limited transmission radius (Lawson & Ramaswamy, 2016; Bhuiyan et al., 2017). Therefore, wireless sensor nodes can not send observations to a remote data destination (the sink) directly. Alternatively, a hub device or other sensor nodes works as a bridge to transfer other sensor node's observations. sensor nodes that are closer to the sink consume more power because they support other sensors to transmit their observations and are expected to have more power failures causing data quality issues (Liao et al., 2019; Togneri et al., 2019). Therefore sensor nodes may determine the network lifetime based on their battery capacity and may impact the system's quality of information (Du et al., 2016).

Typically, wireless sensors nodes of WSNs are distributed according to a spatial or geographical logic over the targeted environment, (Bhajantri & Pundalik, 2017). Large-scale applications which exchange geographic information may face spatial data quality challenges mainly due to the amount of the delivered data from remote sensing devices which may directly affect the correctness of related spatial analysis and spatial decision making, (Bahl, 2015). Thus, data quality challenges are not only related to observations value attributes but also to mismatches in sensor nodes temporal and spatial contextual attributes (Togneri et al., 2019; Barnaghi et al., 2015).

Based on the SLR data extraction process presented in Table 2.10, it is possible

to link all of the addressed data quality challenges in large-scale CPSs in to the following categories:

- Errors in sensor nodes measurements.
- Hardware failures in sensor nodes and communication networks.
- Mismatches in sensor nodes contextual information of both spatial and temporal parameters.

Figure 2.4⁷, shows the main data quality dimensions defined by the SLR data extraction Table 2.10 according to the ratio of the primary studies addressing data quality challenges associated with these dimensions.

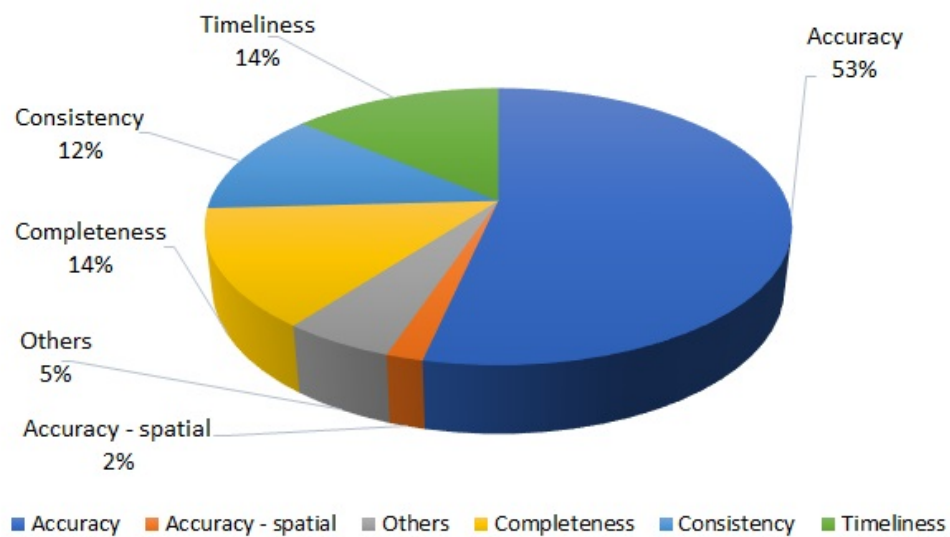


Figure 2.4: The ratio of the data quality dimensions addressed by the SLR primary studies.

Figure 2.5 provides a holistic view of the main data quality management methods in large-scale CPSs, data quality dimensions and the main data quality challenges associated with these systems.

⁷Descriptive studies were excluded.

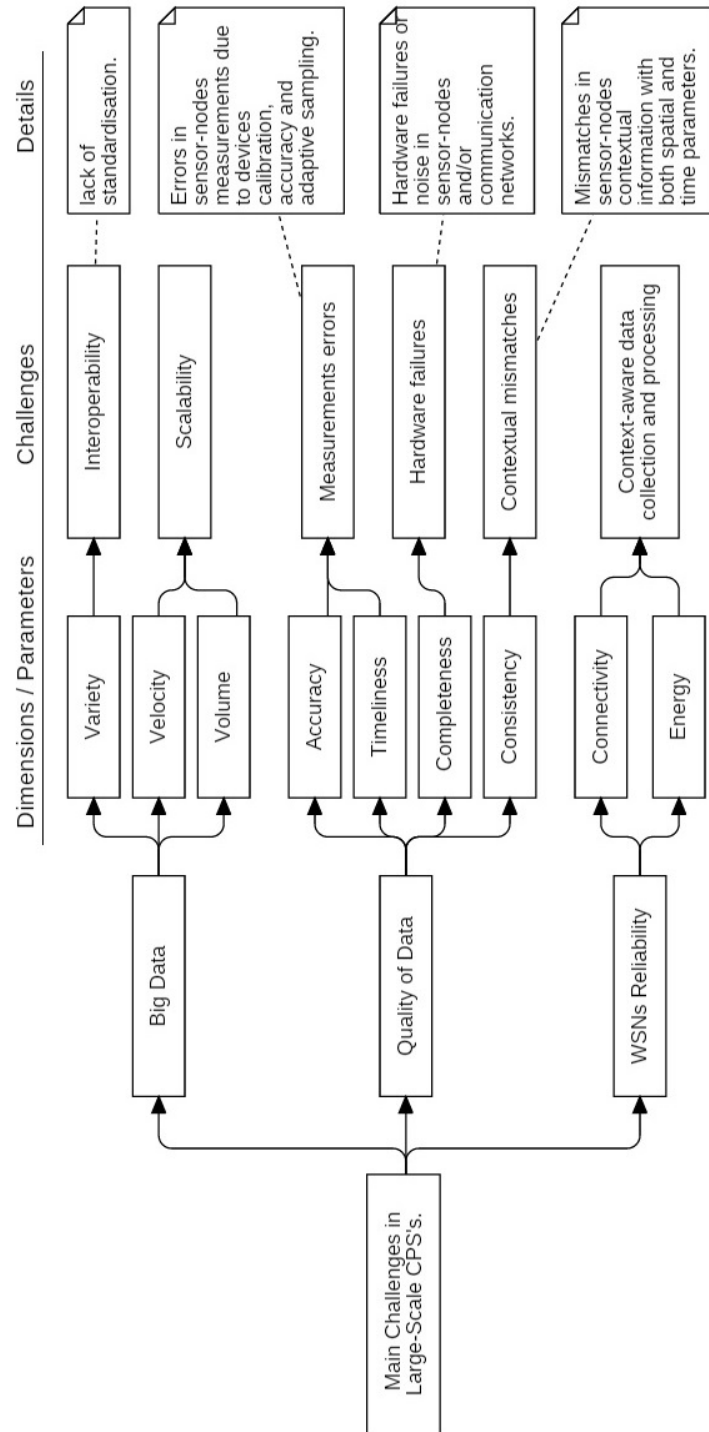


Figure 2.5: The key data quality challenges in large-scale CPSs.

2.2.5 RQ2: Data Mining and Data Quality Management in Large-Scale CPSs.

This section is to answer the second SLR review question (RQ2) listed in Table 2.1 based on the results of the SLR.

Data quality assessment in large-scale CPS applications using traditional methods are no longer efficient due to the heterogeneous large volume of data that these systems typically exchange (Togneri et al., 2019). Thus, such systems, usually, rely on a large number of sensor nodes that stream large volume of data in real-time which requires a high-performance, scalable and flexible tools to effectively provide insight real-time data processing and analysing mechanisms (Kim et al., 2019; Lawson & Ramaswamy, 2016; Geisler et al., 2016; Jayswal & Shukla, 2016).

Based on the results of the SLR data extraction process illustrated in Table 2.10, many statistical, technical and machine-learning models were proposed, tested and evaluated mostly for identifying data quality issues, decreasing their occurrence probability and overcoming their impact on the system. Most of these proposed solutions, methods or models were developed to enhance the reliability of a particular system by improving its data quality based on prior knowledge extracted from the data itself, a process known as Data Mining. Considering the SLR empirical studies only (41 studies), it is possible to categorise all of the adopted data quality assessment/management methods, techniques or solutions into there main groups:

- Data mining.
- Technical solutions/ models.
- Mathematical models.

Figure 2.6 shows the usage ratio of the methods of each of the above groups,

indicating that data mining methods are the most widely used compared to other technical or mathematical techniques.

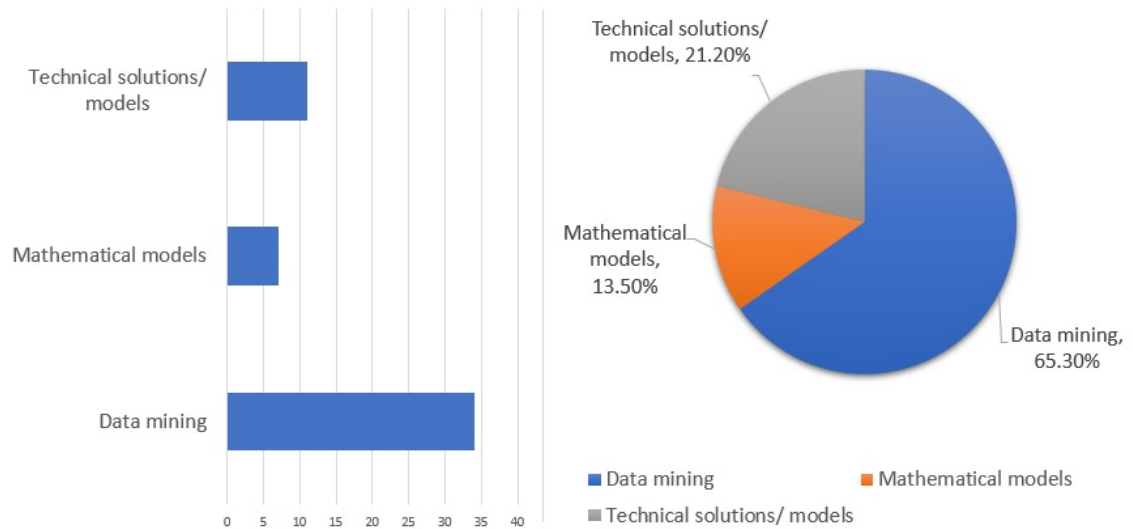


Figure 2.6: The most popular data quality assessment/management methods or techniques in large-scale CPS applications based on the number (left) and the ratio (right) of the SLR studies.

Data mining is the process of auto-discovering knowledge, patterns or models from large volumes of data using advance data analysis methods (Black, 2019, p. 12). Data mining techniques are essential for data analysis in large-scale CPSs which rely on sensor node networks that, typically, stream a continuous flow of spatiotemporal⁸ data at a relatively high-speed and dynamicity (Appice et al., 2014, p. 2-3).

Focusing on the SLR primary studies that adopted data mining methods/techniques for tackling data quality challenges in large-scale CPSs reveals that these methods are mainly divided into statistical and machine-learning based methods. Furthermore, it reveals that most popular data mining techniques used for data mining in large-scale CPSs are anomaly analysis, predictive analysis and clustering analysis, as shown in Figure 2.7.

Moreover, these three leading data mining techniques were applied to address

⁸Spatiotemporal data are sensor nodes observations of events that occur in a given place at a particular time.

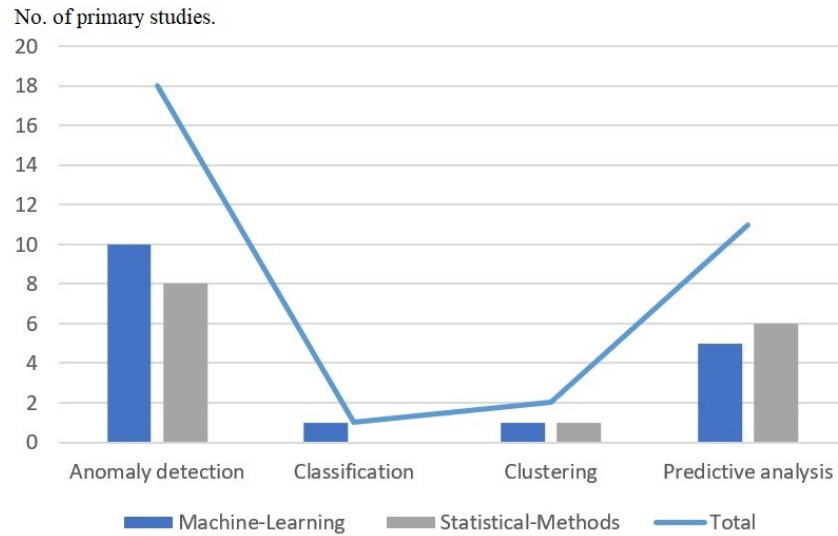


Figure 2.7: The most popular data mining techniques in large-scale CPSs based on the No. of the SLR studies utilising these techniques.

various data quality issues associated with the main data quality dimensions, as shown in Figure 2.8.

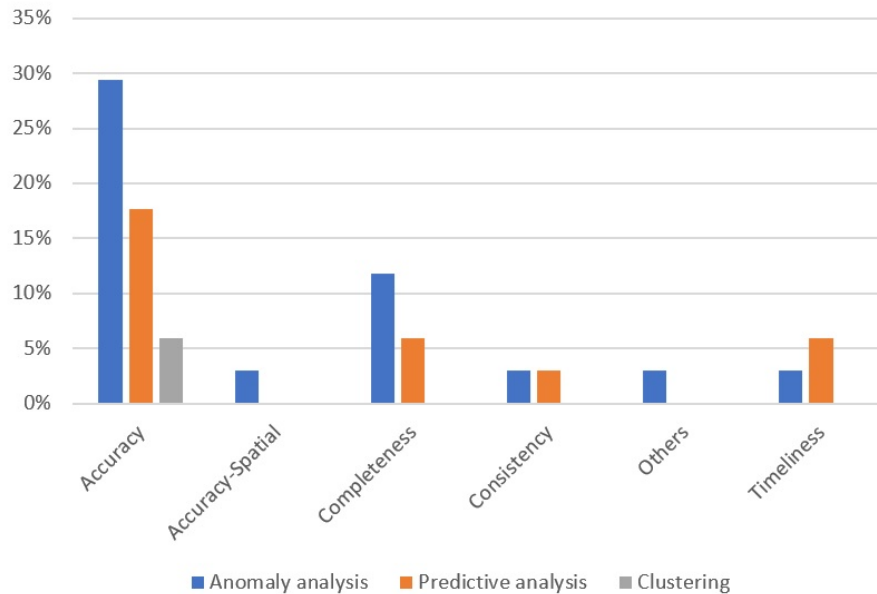


Figure 2.8: The key data mining techniques used to assess the main data quality dimensions in large-scale CPSs.

Figure 2.9 shows a holistic diagram of the main data quality management/assessment methods and techniques and their associated data quality dimensions that these techniques are addressing, based on the SLR results.

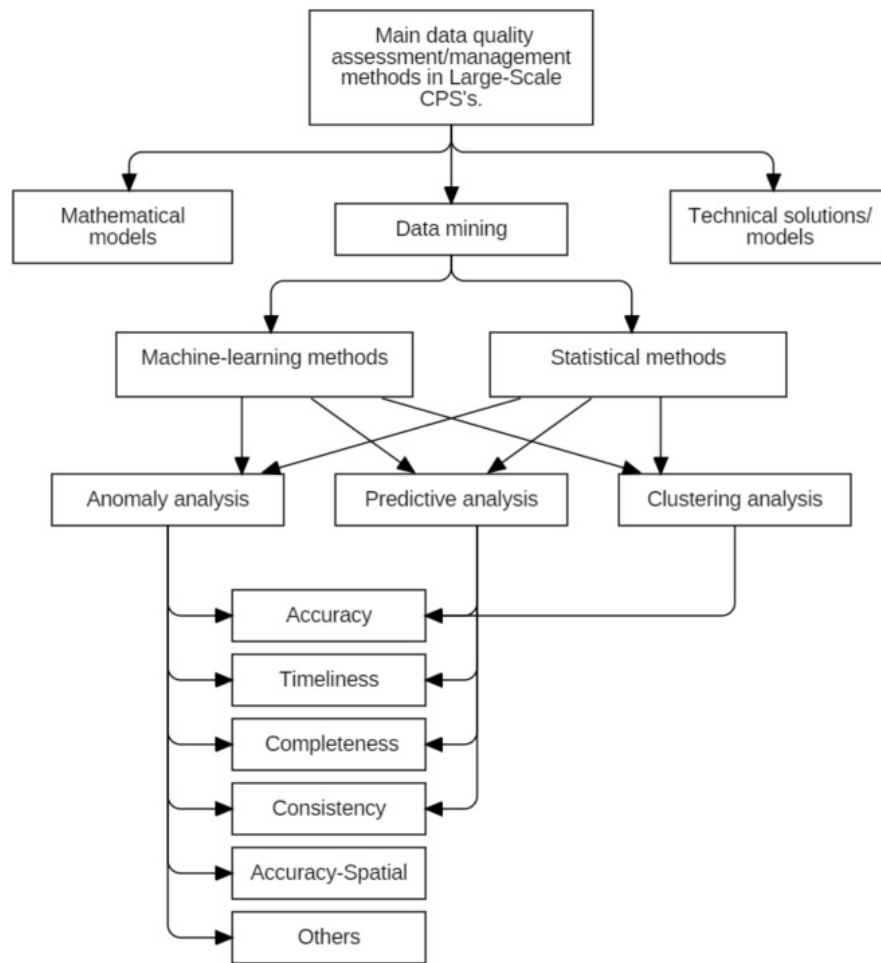


Figure 2.9: A holistic diagram of the main data quality management/assessment methods and techniques and their associated data quality dimensions that these techniques are addressing, based on the SLR results.

2.2.5.1 Anomaly Analysis for Data Quality Management

Anomaly analysis also called outlier detection, is the process of identifying unusual patterns in datasets which do not comply with well-established normal behaviour (Appice et al., 2014, p. 3). If the absolute value of deviation of a sensor node’s observation is higher than a pre-calculated threshold value, then this observation is an outlier (Chen et al., 2018).

As shown in Figures 2.7 and 2.8, anomaly analysis is a significant research field in the context of data quality assessment in large-scale CPSs, which mainly investigated using statistical and machine-learning based outlier detection techniques.

e.g., Deep Neural Networks (DNN) (Hanrong Lu et al., 2016), K-Nearest Neighbours algorithm (KNN) (Hanrong Lu et al., 2016), K-means clustering algorithm (Liu et al., 2019) as machine-learning based outlier detection methods, and standard deviation, correlation coefficient (Xinrui et al., 2019) and DBSCAN (Jayswal & Shukla, 2016; Abid et al., 2017; Nesa et al., 2018) as statistical outlier detection methods.

Outlier detection relies on the assumption that the values of sensor nodes' observations are correlated spatially, temporally or both spatially-and-temporally. However, this assumptions is not necessarily always valid, especially in large-scale CPSs where the correlations between sensor nodes may be affected by many parameters such as the size of the deployment environment and the geographical distribution of sensor nodes (Laso et al., 2017). For example, the approach of spatial continuity cannot be applied directly to the real-world temperature observations collected from the temperature sensor nodes distributed around London due to a phenomenon known as the Urban Heat Islands (UHI) ⁹. According to the Met Office, the phenomenon of urban heat islands is caused by many associated factors, such as the heat released from industrial, domestic facilities, concrete and other building material which observe sun heat during the day and release it back during the night. The phenomenon of urban heat islands may cause up to 5 degrees (unexpected) deviation among sensor nodes observations at the same point in time, which violates the spatial continuity constrains (MetOffice, 2019) ¹⁰ among sensor nodes observations. Moreover, the heat distribution in an urban area depends on many environmental parameters and geographical terrains such as wind speed, humidity, sunshine density, the existence of rivers and the density and height of urban structures. The heat profile map of London is shown in Figure 3.25, where the temperature in central London may reach 11 degree C⁰ and

⁹https://www.metoffice.gov.uk/binaries/content/assets/metofficegovuk/pdf/research/library-and-archive/library/publications/factsheets/factsheet_14-microclimates.pdf

¹⁰More details will be provided in chapter-3.

dropped by over 6 degrees C° in the suburbs at the same point in time (MetOffice, 2019; Chandler, 1965), as shown in Figure 3.25.

2.2.5.2 Predictive Analysis for Data Quality Management

Predictive analysis is the process of mining current and historical data to identify patterns and to forecast the future values of time series (Adhikari et al., 2015; Rawat et al., 2015, p. 507). Predictive analysis might be conducted using statistical or machine learning based techniques (Ratner, 2017, p. 9-12).

For example, machine learning model based on the Random Forest Prediction (Random Forest Regression) method was adopted by (Farooqi et al., 2018) for developing an automated data quality control mechanism for weather data. Another example based on statistical predictive analysis using the one step-forward approach, autoregressive–moving-average (ARMA) model for tackling the inevitable challenge of sensors and sensor networks failure in power terminals, (Li et al., 2019). Furthermore, some applications required a mixed-methods approach, where both machine-learning and statistical methods were adopted to tackle a particular data quality challenge. For example, Okafor et al. (2020) investigated the use of artificial neural network and linear regression for calibrating low-cost environmental monitoring sensors to improve the accuracy of their observations. Predictive analysis methods rely on predictive models developed using historical data as a training data set. Therefore using predictive analysis in real-time (online mode) applications raises performance concerns due to the complexity and volume of the required training data set, (Sta, 2019; Rager et al., 2018). Using predictive analysis is a challenge in real-time large-scale CPSs; thus it may require analysing hundreds of sensor nodes data streams in a relatively short time, (Mylavarapu et al., 2019; Auger et al., 2016). Furthermore, the training process for predictive analysis models requires relatively long and valid (anomaly-free) time series, which cannot be guaranteed in real-world scenarios (Chen et al., 2018, p.

560).

2.2.6 RQ3: Unaddressed Data Quality Management Challenges in Large-Scale CPSs and The Research Gap.

This section is to answer the third SLR review question (RQ3) listed in Table 2.1.

Data is the bridge between the real physical and the digital worlds where data are used to make intelligent decisions in CPS applications (Karkouch et al., 2015; Farooqi et al., 2018). Large-scale CPSs rely on the data gathered by sensors and other devices to make intelligent decisions, low-quality data may impact these decisions' reliability, and compromise these systems' quality of services. In general, ensuring the quality of data in large-scale CPSs is a challenge due to the following:

The heterogeneous nature of their data structures, the scale of data that these systems exchange and due to their real-time requirements (Farooqi et al., 2018; Liu et al., 2014).

CPSs are vulnerable to several external and internal factors such as communication networks errors, sensors failures which interrupt data transferring process, compromise the integrity of data and reduce the performance and reliability of these applications (Al-Milli & Almobaideen, 2019). Failures in wireless sensors and sensor networks are inevitable events in large-scale CPSs, and unusually such failures are unpredictable (Li et al., 2019; Larburu et al., 2014). Furthermore, there is a high possibility of getting erroneous data from sensor node networks due to the limitation in their computing power, storage capacity and communication capabilities (Liao et al., 2019; Abid et al., 2015; Ghosh et al., 2019).

There are no standard criteria to define high-quality data which typically diverse in measure attributes and requirements from an application to another

(Karkouch et al., 2016). Data quality is a subjective concept that varies by the purpose or the intended use of the data. Therefore data quality standards have not been fully identified or applied successfully in large-scale CPSs (Perez-Castillo et al., 2018).

The SLR data extraction Table 2.10 illustrates the attempts to tackle data quality issues associated with large-scale CPSs while revealing further emerging data quality challenges in which very little or no work has been done. Addressing data quality issues in large-scale CPSs is still an open challenge that is not fully enclosed yet (Peng et al., 2019; Perez-Castillo et al., 2018; Farooqi et al., 2018; Prathiba et al., 2016; Shih et al., 2016), which offers new research opportunities and higher possibilities for having more attention in the future. The data quality management issues that the literature did not resolve based on the SLR results are detailed in the following subsections:

- **Sensor nodes' Measurement Errors Detection:** The SLR primary studies which adopted prediction analysis models as data accuracy assessment techniques are sharing the following limitations:
 1. All of the proposed prediction analysis models were based on an assumption that data accuracy issues occur for a short interval of time (point outliers). None of the SLR primary studies proposed a solution to address data accuracy issues associated with long outliers. Long outliers change the time-series' pattern, so the inaccurate observations appear as the standard. In case, a time-series with long outliers is used as the predictive model training dataset. It will compromise the modes' ability to detect data accuracy issues correctly.
 2. No systematic method or approach was demonstrated by any of the SLR empirical primary studies on how it was possible to ensure the

quality of real-world dataset used to train or calibrate the predictive analysis model.

3. None of the SLR primary studies provided a comparison or a justification for why a particular predictive analysis technique was chosen over another. For example, it is not clear when to apply deep learning neural networks as a predictive technique (Krishna, 2018) instead of linear regression (Okafor et al., 2020).
 4. SLR primary studies that investigated anomaly analysis as a solution to evaluate the accuracy of sensor nodes measurements by comparing their observations with different sensor nodes or to a pre-calculated threshold value were based on the assumption that these sensor nodes are spatially correlated. However, this assumption is not necessarily always valid in large-scale CPSs. The spatial continuity among sensor nodes in large-scale CPS applications might be compromised because of the vast distance separating these devices or other factors that disrupt the spatial continuity constraints, as detailed in Section 2.2.5.1 and Section 3.6.
- **Sensor nodes' and Sensors Networks' Failures Detection:** The SLR primary studies provided no systematic method or a generic approach for detecting sensor nodes and sensor node networks hardware failures in large-scale CPSs. All proposed failure detection mechanisms were mainly domain-specific solutions. For example, signal processing techniques were utilised for monitoring the hardware status of a Chinese network of weather radars by Togneri et al. (2019) which can not be applied as a generic solution for hardware failures detection in sensor node networks of large-scale CPSs.
 - **Ensures the Quality of Observations' Spatial and Temporal Contextual Attributes:** The SLR primary studies revealed that further research is re-

quired to address the challenge of ensuring the quality of sensor nodes contextual information of both spatial and temporal attributes. Spatial data quality issues (sensor nodes location) may affect the validity of any related spatial analysis. Furthermore, very limited or no research have practically investigated the possibility of using observations timestamp analysis techniques as a potential solution to improve the quality of sensor nodes spatial contextual information.

2.3 The Research Questions

The systematic literature review has provided evidence to support the research questions, which aim to approach some of the emerging data quality challenges in large-scale CPSs. Accordingly, the broad research questions of this research are as follows:

1. Is it feasible to develop a proof of concept data quality management system for large-scale CPSs that can deliver the following:
 - (a) Detects sensor nodes measurements errors associated with the four main data quality dimensions; accuracy, timeliness, completeness, and consistency.
 - (b) Detects hardware failures in sensor nodes and sensors' communication networks.
 - (c) Ensures the quality of both spatial and temporal contextual attributes of sensor nodes observations.
2. Is it possible to empirically evaluate the effectiveness of the proposed data quality management system using a real-world, large-scale sensor node network as a case-study?

3. How to address bias concerns related to the evaluation process of the data quality management system, which emerges due to the presence of data quality issues in the testing / evaluating real-world dataset?

2.4 Summary

This chapter is an introduction to data quality and the main challenges associated with its management in large-scale CPSs. It incorporated a systematic literature review which indicated that data quality management in large-scale CPSs is still an open challenge. The SLR concluded that not much had been done to provide a practical, comprehensive data quality management solution to detect sensor nodes measurements errors associated with the main data quality dimensions of accuracy, timeliness, completeness, and consistency in large-scale CPSs. No systematic or generic approach was demonstrated for sensor nodes, and sensor node networks failures detection and further research is required to address the challenge of ensuring the quality of the spatial and temporal contextual attributes of sensor nodes observations.

To address these challenges, a proof of concept data quality management system is proposed for large-scale CPSs to detect errors in sensor nodes measurements, identify hardware failures in sensor nodes or sensor node networks and detect mismatches in temporal and spatial conceptual attributes of sensor nodes observations. The proposed system will be evaluated using real-world, sensor node networks. More details about the structure of the proposed system, the evaluation and validation methods will be provided in the next chapter (Chapter 3).

Chapter 3

System Design (Methodology)

“... there are three steps in a quality control process: the specification of what is wanted, the production of things to satisfy the specification, and the inspection of the things produced to see if they satisfy the specification.”

— Shewhart & Deming (1986)

The purpose of this chapter is to describe the research context. It presents the research design, the structure of the data quality management system, data analysis methods, the techniques used to address the research objectives, and explains the adopted logical sequence to conduct the research activities.

3.1 Overview of Research Paradigms

This research is an empirical study that uses software engineering and data science techniques to tackle the research questions and to deliver its objectives. Like any other applied engineering branches, software engineering disciplines can be utilised to develop new methods (solutions) via verifying a particular theory using

empirical observations, utilising these new methods in industrial projects, and continuously improving these methods during the projects life-cycle (Staron, 2020, p. 2-8).

According to Basili (1993), there are various forms of software engineering research methodologies which mainly categorised into experimental or analytical paradigms based on the conducted research method:

- The Experimental Paradigm

The experimental paradigm includes the following methods:

- Scientific method: proposing a theory or a model to measure, analyse a real-world behaviour and if possible, validate the hypotheses of that theory or model using a simulation model.
- Engineering method: observe an existing solution, and propose a better one, build measure and analysis until achieving tangible improvements.
- Empirical method: proposes a model and measure, analyses and validates that model using case studies or experiments (Wohlin et al., 2012, p. 3-8)

- The Analytical Paradigm

- Mathematical method: proposes a formal theory, obtains results and validates it if possible, with experimental measurements.

Empirical based studies use real-world systems as case studies to validate a given theory or hypothesis (Aggarwal et al., 2009). Empirical studies help evaluate, monitor, predict, and enhance a targeted system's performance quickly and at a relatively low cost. Empirical studies are especially beneficial for large-scale systems which compose various activities and resources that need to be managed with additional attention (Malhotra, 2015, P. 1).

The empirical approach was adopted in this research because it is the most suitable

method to achieve the research aim of developing a data quality management system for large-scale CPSs and evaluate its performance through empirical tests based on real-world case studies. More details about the research strategy and methods are detailed in the following sections.

3.2 Empirical Research Methods

Empirical research methods are mainly categorised into qualitative and quantitative approaches. However, it is almost always better to use a combination of both approaches to investigate a software engineering research hypothesis or to validate its outcome, which is known as the mixed-methods research methodology (Seaman, 2008, p. 60).

3.2.1 Quantitative

Quantitative methodology is an empirical research method, mainly concerned with numerical data-driven results where the collected observations are analysed using mathematical means to interpret a process or a project under study (Guéhéneuc & Khomh, 2019, p. 289). Quantitative research tends to process large-scale and representative sets of data collected from a controlled experimental environment (Blaxter, 2010, p. 65). In general, results generated by quantitative methods are reproducible and unbiased since they are based on statistical or mathematical techniques to validate a hypothesis or investigate a phenomenon (Malhotra, 2015, p. 3-4).

3.2.2 Qualitative

Qualitative research investigates in-depth a new process or technique based on explanation or textual descriptions related to human beliefs or behaviour (Malhotra,

2015, p. 4). Qualitative research attempts to understand a phenomenon using methods such as observations, maps, interviews, and focus group's discussions without having precise measurements or quantify analysis in their results (Creswell & Creswell, 2017, p. 49-51). Moreover, qualitative research can be used as a source of hypotheses for further investigation using quantitative research (Carvalho & White, 1997, p. 20).

3.2.3 Mixed-Method (Triangulation)

An empirical research approach involves data collection and analysis using qualitative and quantitative techniques to answer more complicated research questions (the case of this research) or verify the research hypothesis. Thus, a mixed-method research approach involves collecting a more robust array of evidence than those collected from the qualitative or the quantitative research method alone (Yin, 2017, p. 100-101).

Mixed-method is a research methodology as well as a research method of inquiry, where both qualitative and quantitative strands must be fully integrated (Mixed) through multiple data points about the same phenomenon in the study. If both qualitative and quantitative approaches, were used in the same study but without any links between them, then it is known as Quasi-mixed methods (Creamer, 2017, p. 41). Mixed-method is an empirical research method that often involves using software engineering methods to apply theories from other disciplines in order to be able to interpret quantitative and qualitative data (Guéhéneuc & Khomh, 2019, p. 289-290).

3.3 Empirical Research Strategy

Empirical studies can further be categorized according to the type of the implemented research strategy as an experimental, case study and survey approaches. The relation among empirical research methods and strategies is shown in Table 3.1, where experiments are mainly quantitative since experiments, typically, focus on collecting numerical measurements of a monitored phenomenon and provide results using statistical methods. In contrast, case-study can be conducted using any or both qualitative or quantitative approaches (Wohlin et al., 2012, p. 10-12).

Table 3.1: Qualitative and quantitative methods in empirical research strategies.

Strategy	Qualitative methods	Quantitative methods
Experiment	-	✓
Case study	✓	✓
Survey	✓	✓

The next sections focus on the experimental, and the case study research strategies as this research mainly rely on both of these approaches.

3.3.1 Experimental

Typically, an experimental study is an empirical test conducted under a controlled scope or laboratory environment to prove an established hypothesis or test a supposed correlation between system input and output variables (Münch et al., 2012, p. 178-180). It is difficult to define the experiments controls in software engineering experiments because software engineering tasks are mainly dependent on current technology and individuals' experiences. Thus, no standard method or procedure is adopted in the past to perform such tasks (Kitchenham et al.,

2002). Typically, the outcome of an experiment is either accepting or rejecting the experiment hypothesis. The term "null hypothesis" represents the opposite state of the suggested hypothesis, consequently proving an experiment hypothesis can be expressed as rejecting its null hypothesis (Smalheiser, 2017, p. 128). The main processes of the experimental research approach are shown in Figure 3.1.

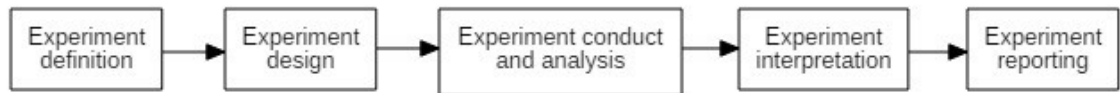


Figure 3.1: The main processes of the experimental research approach.

3.3.2 Case Study

A case study is an empirical research method aimed at investigating or tackling a real project or case within its real-world context. Typically, case studies rely on prior theoretical propositions or analysis methods to cope with the complex and dynamic characteristics of real-world projects (Yin, 2017, 46). Typically, the case study method presents scientific evidence collected under a lower level of control comparing with the experimental method (Zelkowitz & Wallace, 1998). The main processes of the case study approach are shown in Figure 3.2.

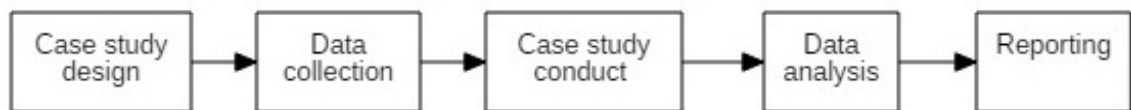


Figure 3.2: The main processes of the case study research approach.

3.3.3 Choosing the Research Methodology

Research methodology is a framework that links supporting methods, approaches and guidelines to formulate models and theories about the studied phenomenon, as well as to validate these models and theories to address the research objectives

in a systematic way (Blessing & Chakrabarti, 2009, p. 9-11). The choice of an appropriate research methodology approach involves deciding whether a qualitative, quantitative, or mixed-method approach should be adopted to study the research topic and which research strategy will be employed to conduct that approach. The choice of the research approach can be made based on the interconnection of three components; research methods, design, and paradigm which interpret the approach into practice, as shown in Figure 3.3. Other parameters may influence the choice of the research approach, such as the existing literature, the nature of the research problem, the researcher's personal experience, and well-established research practices (Creswell & Creswell, 2017, p. 43).

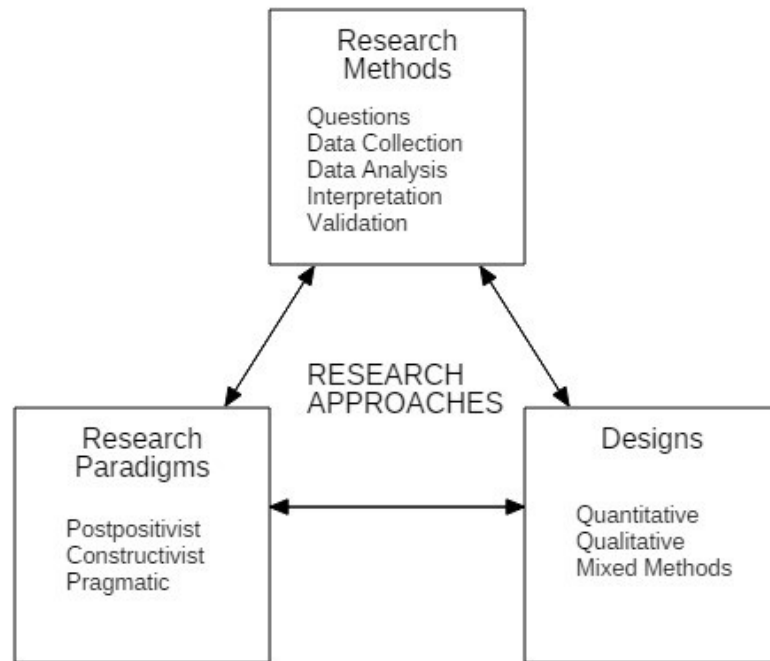


Figure 3.3: Research approaches based on the interconnection of Research Methods, Design, and paradigms.

- **Research paradigm** is a set of beliefs that attach a researcher to a particular worldview on how the research problem should be addressed (Kuhn, 1962, p. 264). In general, research is an interactive process driven by the researcher and affected by his or her scientific background, gender, ethnicity, social class, and even by other people involved in the research (Denzin & Lincoln, 2017,

p. 195). The research narratives or paradigms can be broadly categorised as postpositivist, constructivist and pragmatic (Creswell & Creswell, 2017, p. 43-47). Postpositivist paradigm is also known as the objectivist, empirical science and the scientific method. It is the traditional form of research based on the deterministic philosophy in which the researcher tries to identify and estimate the causes of the research problem by implementing a set of small tests or experiments to verify, fulfil, and refine the research questions or hypothesis. Postpositivist is an evidence-based paradigm where data, observations, numeric measurements and information collected by instruments or by the researcher himself are used to determine the effects or outcomes of the studied phenomena (Creswell & Creswell, 2017, p. 44).

- **Research design**¹ (methodology) is the early stages of research development, which, includes the development of the research questions, hypothesis and data analysis methods (Denzin & Lincoln, 2017, p. 549-450). The research design must be formulated depending on the type of research and the data collection strategy (Bairagi & Munot, 2019, p. 75). The research design emphasizes the research approach as qualitative, quantitative, or mixed methods and known as inquiry strategy of inquiry (Creswell & Creswell, 2017, p. 49).
- **Research methods** involve all the researcher's basic techniques to perform data processes in the study, such as data collection, analysis, features extraction and validation techniques. The research method's choice depends on the type of collected data and the intended type of information required by the researcher to fulfil the research objectives. It also reflects the choice of the research approach (Creswell & Creswell, 2017, p. 53-55).

A wide range of possible choices can drive the researcher's logical chain of actions

¹*within the context of the postpositivist paradigm

to address his research objectives. However, it is possible to drive the research through the appropriate direction by defining the most abstract level of action, the research paradigm in this case, and move on to elaborate the research methodology, methods and techniques, this approach is known as the Research Pyramid, as shown in Figure 3.4, (Jonker & Pennink, 2010, 23).

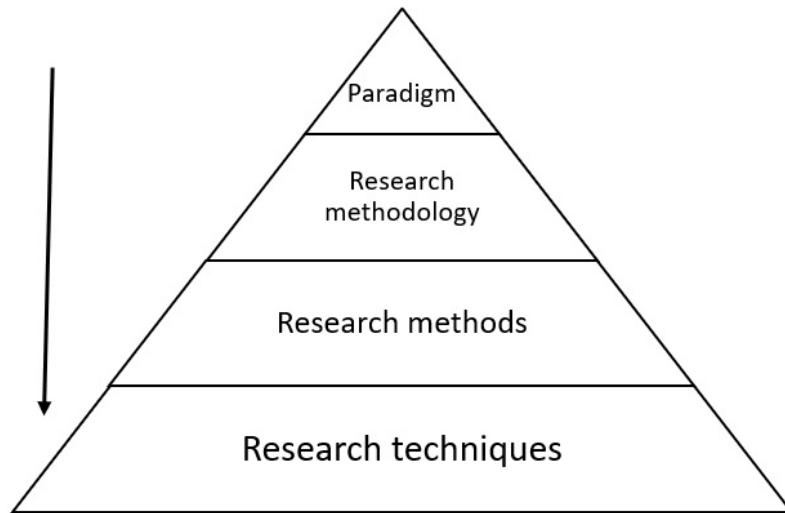


Figure 3.4: The logical sequence to decide the research methodology, methods and techniques.

The quantitative methodology was mainly adopted in this research to enable the possibility of using software engineering and data science literature to support the theoretical aspects of the research and empirically fulfil its objectives. This research mainly relies on primary data (real-world observations) to address its objectives and to verify its proposed solution's validity. These observations are collected from two different data sources as follows:

- **Large-scale, real-world sensor node network:** Observations from a large-scale environmental sensor node network, a temperature network in particular distributed around London were collected in real-time forming continuous time-series for each sensor node in the network.
- **Local sensor node network:** Observations were collected from a fully controlled, high-quality temperature sensor node network deployed at the Uni-

iversity of East London / Dockland campus. This network aims to provide benchmark observations to validate the quality of the data collected from the large-scale network. Thus, this network's collected data have no missing values or outliers and can be used to calibrate the system models before applying them using a real-world dataset. Furthermore, it provides a controlled environment to identify some data quality issues that were difficult to identify using the data collected from the real-world sensor node network.

Figure 3.5 shows a holistic overview of the overall research phases, which broadly identified as (i) Research definition, (ii) Literature review, (iii) Research design, (iv) Research conduct and (v) The research results interpretation. The details of the related phases are discussed in the following sections.

3.4 System Design and Development Phases

Systems analysis and systems design are key components of the process of information systems development². Systems analysis is a set of activities required to understand and specify the purpose of the system, its functionality and describe its details. In comparison, systems design describes the implementation details of the information system as well as describes how it will work and how its components are going to engage together. The outcome is a detailed technical implementation description of the system development process (Satzinger et al., 2015, p. 5-6). A management framework usually guides the analysis, design, and other development phases of an information system. This framework is known as the System Development Life Cycle (SDLC). SDLC of an information system includes all the planning, analysis, design, programming and testing stages and covers other development, deployment, and even maintenance details (Langer,

²Information systems are combinations of hardware, software, data and processes that interact coherently to provide a particular output. Some information systems may involve human actions in their process (Tilley & Rosenblatt, 2016, p. 4).

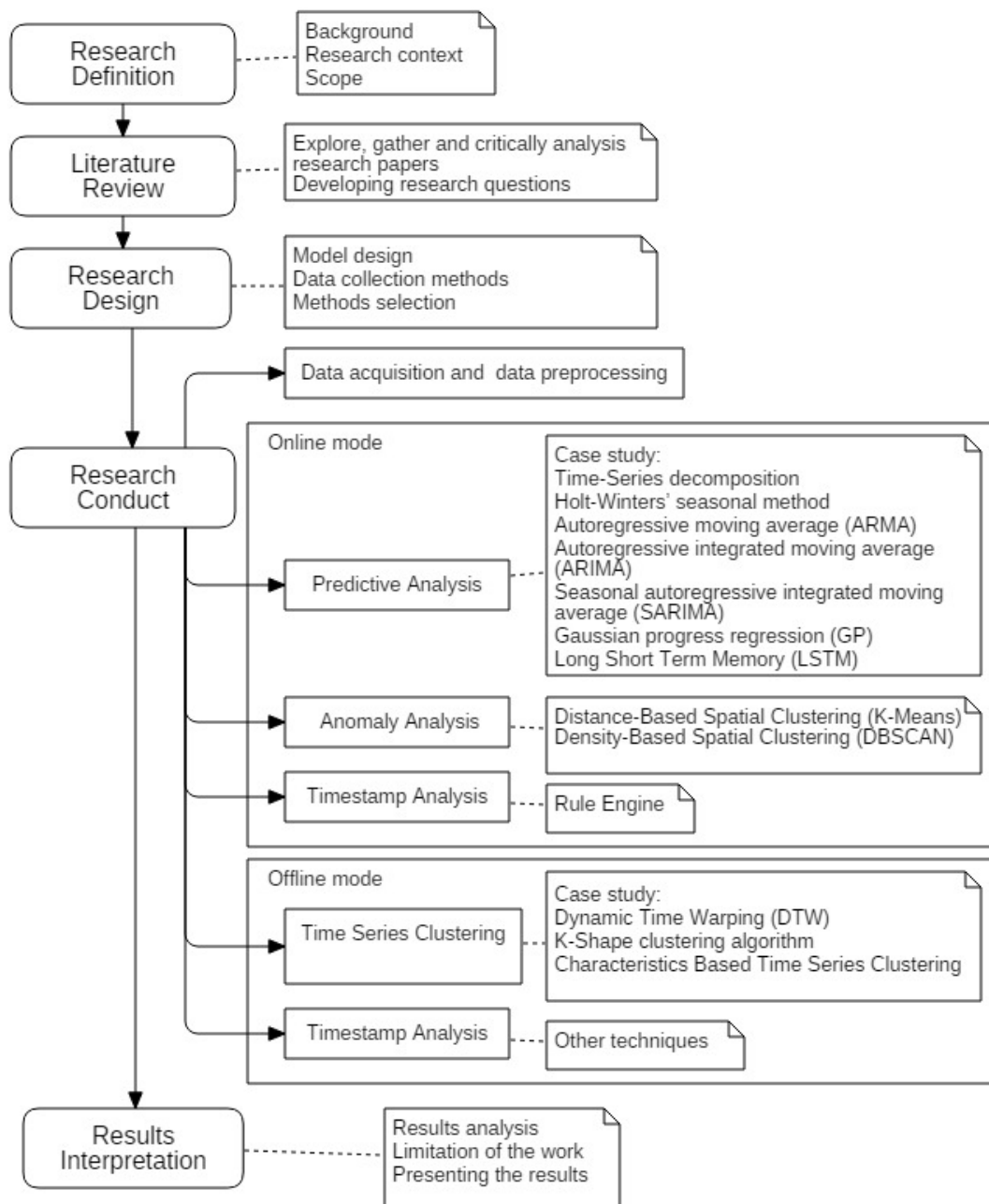


Figure 3.5: A holistic overview of the research overall phases.

2007, p. 10-15). Many different SDLC approaches can be adopted to develop information systems, but they are categorized into predictive and adaptive in general.

The predictive approach is usually adopted for developing information systems that all their requirements are well defined, and their development does not in-

involve introducing new processes. This approach requires extensive planning and coordinating in advance, to develop the information systems as specified in their designs.

The adaptive approach is suitable for developing information systems that cannot be thoroughly planned because it is not possible to provide a complete solution or determine all system requirements at the early stages of the system development (Satzinger et al., 2015, p. 297-298).

The data quality management system was developed as a series of models that have been modified many times or iterated during the research progresses. Iteration means that the phases of the system development life cycle were implemented sequentially with some overlapping, as shown in Figure 3.6. These models have gone through the same analysis steps, design, development and test and transferred the results as feedback to the next iteration until they satisfied a particular accuracy or performance criterion. Thus, this research methodology is mainly based on the adaptive SDLC approach. Since all of its development activities, designs and models have been modified often or iterated during the research progresses.

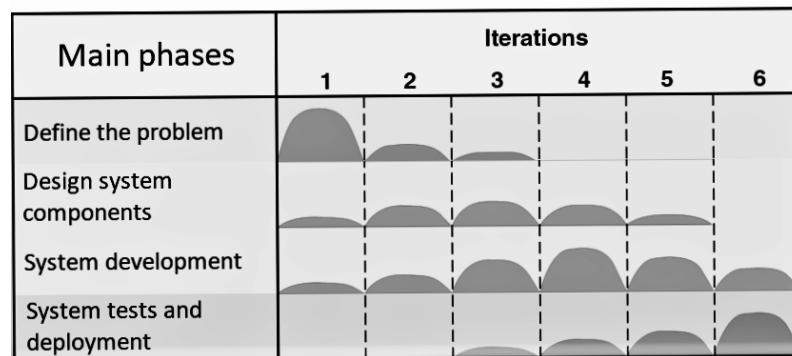


Figure 3.6: Adoptive SDLC iterations and the key development phases of the data quality management system (Satzinger et al., 2015, p. 300).

This research aims to develop a proof of concept data quality management system for large-scale CPSs. It relies on different statistical and machine learning models for detecting sensor nodes measurement errors, sensor nodes failers

and mismatches in sensor nodes contextual attributes. Each of these models has gone through the same high-level design and development activities shown in Figure 3.7, as adopted from (Brimicombe, 2009, p. 89).

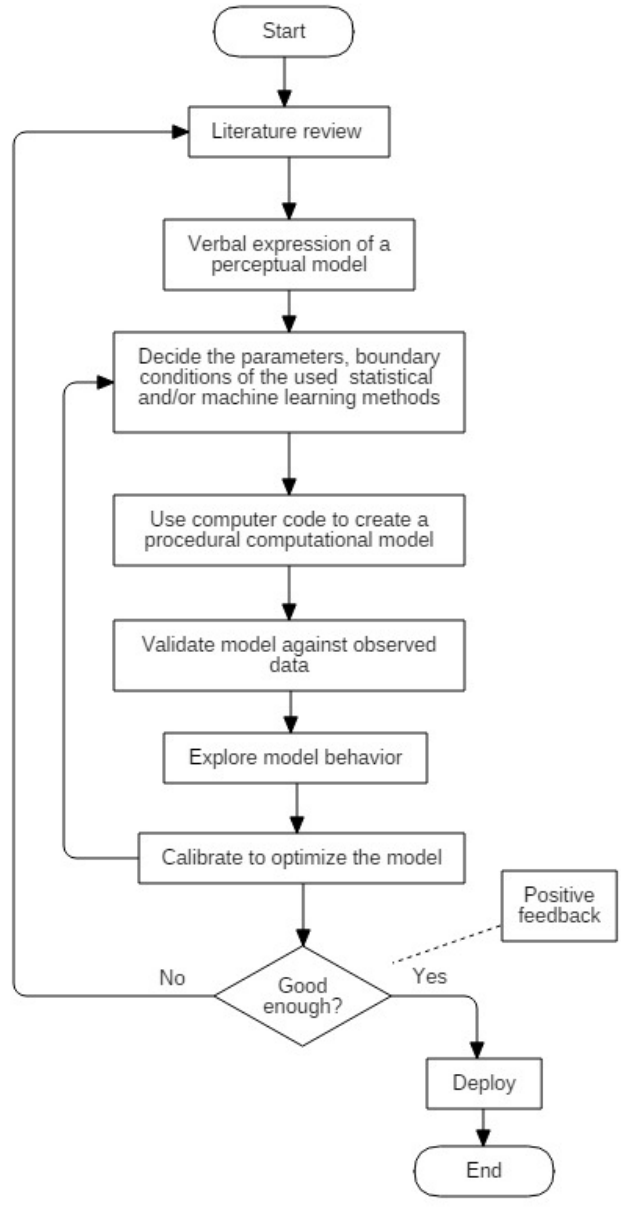


Figure 3.7: A holistic view of the data quality assessment models' development activities (Brimicombe, 2009, p. 89).

3.4.1 System Analysis and Design

The data quality management system was designed to address data quality challenges associated with detecting: sensor nodes measurement errors, sensor nodes hardware failures, and mismatches in sensor nodes spatial and temporal contextual attributes. Detecting sensor nodes measurement errors associated with the primary data quality dimensions of accuracy, timeliness, completeness, and consistency in large-scale CPSs were investigated using predictive and anomaly analysis models via utilising statistical and machine-learning techniques. Time-series clustering techniques were investigated as a feasible mean for detecting long-segmental outliers as an indicator of sensor nodes' continuous halting and incipient hardware failures. Furthermore, the quality of the spatial and temporal contextual attributes of sensor nodes observations was investigated using timestamp analysis techniques. More details about each component of the data quality management system will be provided in the coming sections, while this section will provide a general overview of the system design. The main components of the proof of concept, data quality management system are shown in Figure 3.8. It consists of three layers characterised according to their functionality, as follows:

- **Layer -1: Sensor Node Networks Layer:** Observations from two sensor node networks were collected and provided as input data to the system's second layer (the data warehousing layer). The first network is an outsourced large-scale sensor node network. It consists of hundreds of sensor nodes that collect observations of different environmental parameters such as temperature, humidity and air quality from around London. The second source of data is a temperature sensor node network that served as a benchmark data to test and calibrate the different components of the data quality management system, as shown in Figure 3.8, layer -1.

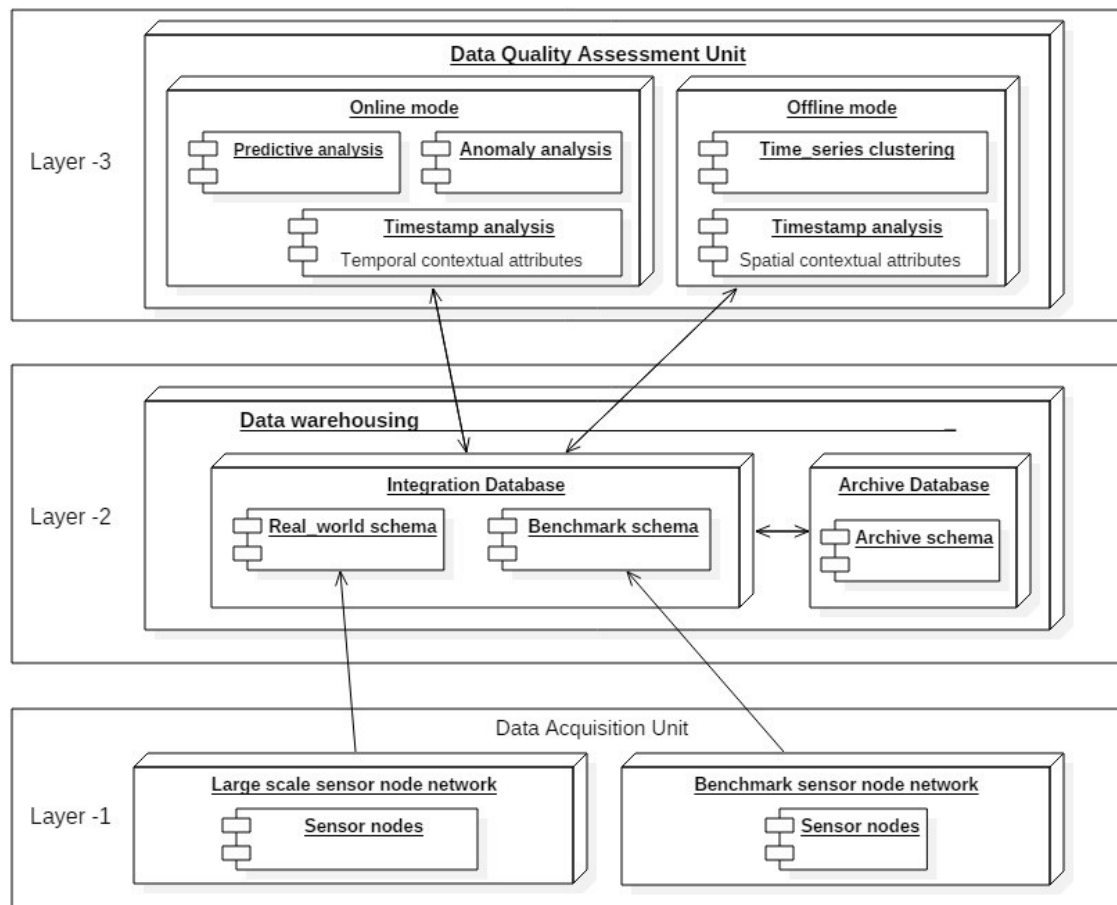


Figure 3.8: The main components of the proof of concept data quality management system. The mode-model structure of the data quality assessment unit is illustrated in Figure 3.11.

- Layer -2: Data Warehousing and Integration Layer:** The data quality management system is designed to perform data management and data integration in real-time. In this case, this is a challenge due to the large volume of the received data and the high diversity in data structure and attributes. Data warehousing was adopted as a solution for data collection and management. Data from different sources were linked together (integrated) based on various parameters, such as observations timestamps, geographic attributes and type. Technically, the data warehousing layer consists of two databases. The first database is the Integration Database which hosts two different data schemes the large-scale sensor node network observations schema and the benchmark local sensor node network observations schema. as shown in,

Figure 3.8, layer -2. The second database is the Archive Database. It has the same structure as the Integration Database, but it is dedicated to historical data storage which will be transferred automatically from the primary database (the data integration database).

The integration and the archive databases in the data warehousing Layer are linked using two direction arrows, thus the data from the archive database could be retrieved and used for training and testing the predictive analysis or time-series clustering models of the data quality assessment unit which require a long window of sensor-nodes time-series to enhance their accuracy.

The combination of both Layer-1 and Layer-2 of the data quality management system forms the **Data Acquisition Unit**.

- **Layer-3: Data Quality Assessment Layer:** consists of four main components; predictive analysis models, anomaly analysis models, time-series clustering models and timestamp analysis models. Many statistical, machine learning and time-series clustering technique were investigated as data quality assessment mechanisms in the data quality management system, as shown in Figure 3.8, layer-3. The role of each of these components is highlighted in Figure 3.11.

More details about the data quality management system are provided in the following sections.

3.4.2 Data Acquisition Unit

The first objective of the research is to investigate data quality challenges in large-scale cyber-physical systems based on the literature and based on empirical data analysis of observations collected from a real-world case-study. This objective was tackled in chapter-2, The Literature Review, and extended in Chapter-4,

Implementation and Results, using empirical analysis for observations' streams collected from the real-world, large-scale sensor node network.

In order to empirically investigate data quality issues in large-scale CPSs, the Data Acquisition Unit (DAU) was developed to collect data streams from sensor node networks in real-time. Its conceptual structure is shown in Figure 3.8 Layer-1 and Layer-2, which consist of the following components:

3.4.2.1 Sensor Node Networks

The proposed data quality management system was empirically validated using real-world data collected from environmental monitoring sensor node networks. Although many environmental parameters were collected at the early stages of conducting this research, temperature sensor nodes were selected as the primary data source to test the proposed data quality management system. Thus, temperature sensor nodes are the most available type of sensors, vastly distributed around London, and they managed by different providers, which offers an excellent opportunity to investigate data quality issues of such large-scale network of sensor nodes. Data Acquisition is the process of efficiently acquiring observations from sensor node networks. The notion of efficiency refers to the fact that sensor nodes are, typically, battery-powered and in most cases are distributed in remote and inaccessible terrains. Thus, data acquiring from sensor nodes must be achieved in the most energy-efficient method to prolong sensor nodes battery life. Another factor of efficiency is linked to the communication cost. Thus, reducing data transactions within the network will reduce the communication cost, which must be achieved without compromising the accuracy of observations (Sathe et al., 2013, p. 11). The topology of the two sensor node networks used in this research is a relatively complex mix of data pull-based, and data push-based observations triggered mechanisms. Both networks also utilise cloud computing as an essential component in their structure. In the pull-based method, the end-user defines the

frequency of triggering the data acquiring process. In contrast, in the push-based method, sensor nodes and their gateway (base-station or server) have their own interactive communication function which determines when sensor nodes push their observations to the gateway automatically (Sathe et al., 2013, p. 15,18). Wireless sensor nodes main hardware components and anatomy are briefly introduced in Appendix A, Section A.1.

3.4.2.2 Data Streams

Large-scale CPS applications such as environment monitoring systems typically involve many sensor nodes distributed over a vast geographical area, forming a large-scale sensor node network. If a sensor node network is denoted by S and the number of sensor nodes within the network S is between 1 and m , then it is possible to identify each sensor node in the network S by S_j where $j = \{1, \dots, m\}$ and $1 \leq j \leq m$ then:

$$S = \{S_j \mid 1 \leq j \leq m\} \quad (3.1)$$

Assuming that the value of an observation from the sensor node S_j at a point in time t_i is V_{ij} and if the sensor node S_j is configured to stream observations regularly based on a pre-set duty-cycle then the sampling rate or the duty-cycle of the sensor node S_j equals $t_{i+1} - t_i$. Thus, the timestamp attribute of the observations become irrelevant, and it is possible to use an index value i for indicating the time axis in the data stream (Sathe et al., 2013, p. 13-14). Since sensor nodes in large-scale environment monitoring systems are usually deployed in vast geographical terrains, each of these sensors, typically, has geographical coordinates attributes (X_j, Y_j) .

Based on this brief introduction, it is possible to specify the main attributes that the data acquisition unit must be able to process, as shown in Table 3.2, where, each

row represents a single observation and the included data are for demonstration purpose only.

Table 3.2: The main attributes of sensor nodes data streams in large-scale CPSs.

Time index	Timestamp	Sensor ID	Coordinates		Value
i	t_i	S_j	X_j	Y_j	V_{ij}
1	02/02/2020 01:00:00	1	-0.18	51.4	4.5
1	02/02/2020 01:00:00	2	-0.19	51.3	4.2
1	02/02/2020 01:00:00	3	-0.2	51.5	5.1
2	02/02/2020 01:15:00	1	-0.18	51.4	4.4
2	02/02/2020 01:15:00	2	-0.19	51.3	4.1
2	02/02/2020 01:15:00	3	-0.2	51.5	5.1

In an ideal situation, as shown in Table 3.2, observations from large-scale sensor node networks occur periodically in specific points in time and spatially labelled with two dimensional coordinates (Latitude and Longitude). Observations from active sensor nodes would be presented as discrete, time-stamped observations snapshots which form spatiotemporal time-series (Appice et al., 2014, p. 5-7), as shown in Figure 3.9.

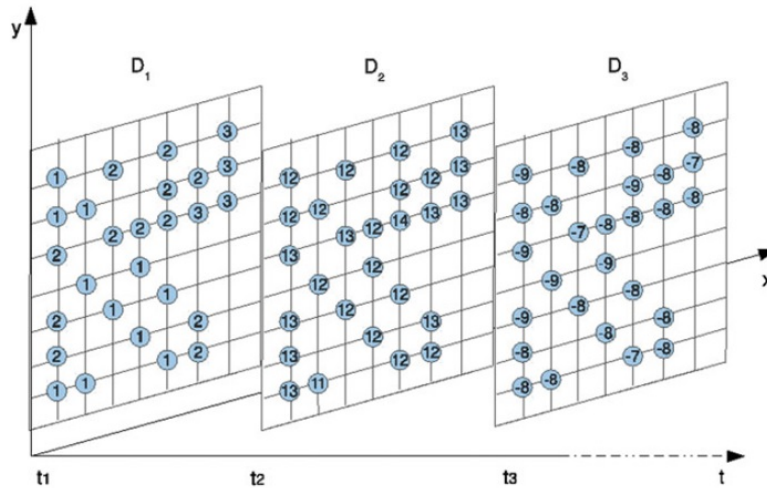


Figure 3.9: Sensor nodes data stream as discrete, time-stamped observations snapshots (Appice et al., 2014).

Each of the blue circles presents an observation from a particular sensor node with a value attribute V_{ij} shown as a number. Moreover, D_1, D_2, \dots, D_t are

data snapshots equally separated by time (t) and the (X, Y) coordinates are the geographical latitude and longitude location attributes of the sensor nodes.

3.4.2.3 Software Framework

The software framework is the combination of all software components and solutions used to facilitate observations transactions from remote sensor nodes to the local database warehouse of the data quality management system. The software framework includes cloud computing modules, databases, and objected-oriented programming tools used to develop the data acquisition software, particularly for this research. The high-level process diagram of the data acquisition unit is shown in Figure 3.10. The full description of the main components of the software framework is illustrated in details in Chapter-4.

The data acquisition unit is designed to collect sensor nodes observations effectively in real-time. It collects observations continuously based on a duty-cycle which triggers the data collection action every T minutes, where $T = t_{i+1} - t_i$ and, T is a dynamic parameter that may change based on the changes in the duty-cycles of sensor nodes. T is smaller than the shortest duty-cycle of any of the sensor nodes in the network. The data acquisition unit is designed to check actively and adjust its data collection duty-cycle T actively to find the shortest duty-cycle, as shown in the high-level process diagram of the data acquisition unit, Figure 3.10.

3.4.3 Data Quality Assessment Unit

The data quality assessment unit is the core component of the data quality management system designed to detect data quality challenges associated with errors in sensor nodes measurements, hardware failures in sensor nodes, and to detect mismatches in spatial and temporal contextual attributes of sensor nodes observations to ensure these observations fitness for use in large-scale CPS applications. It

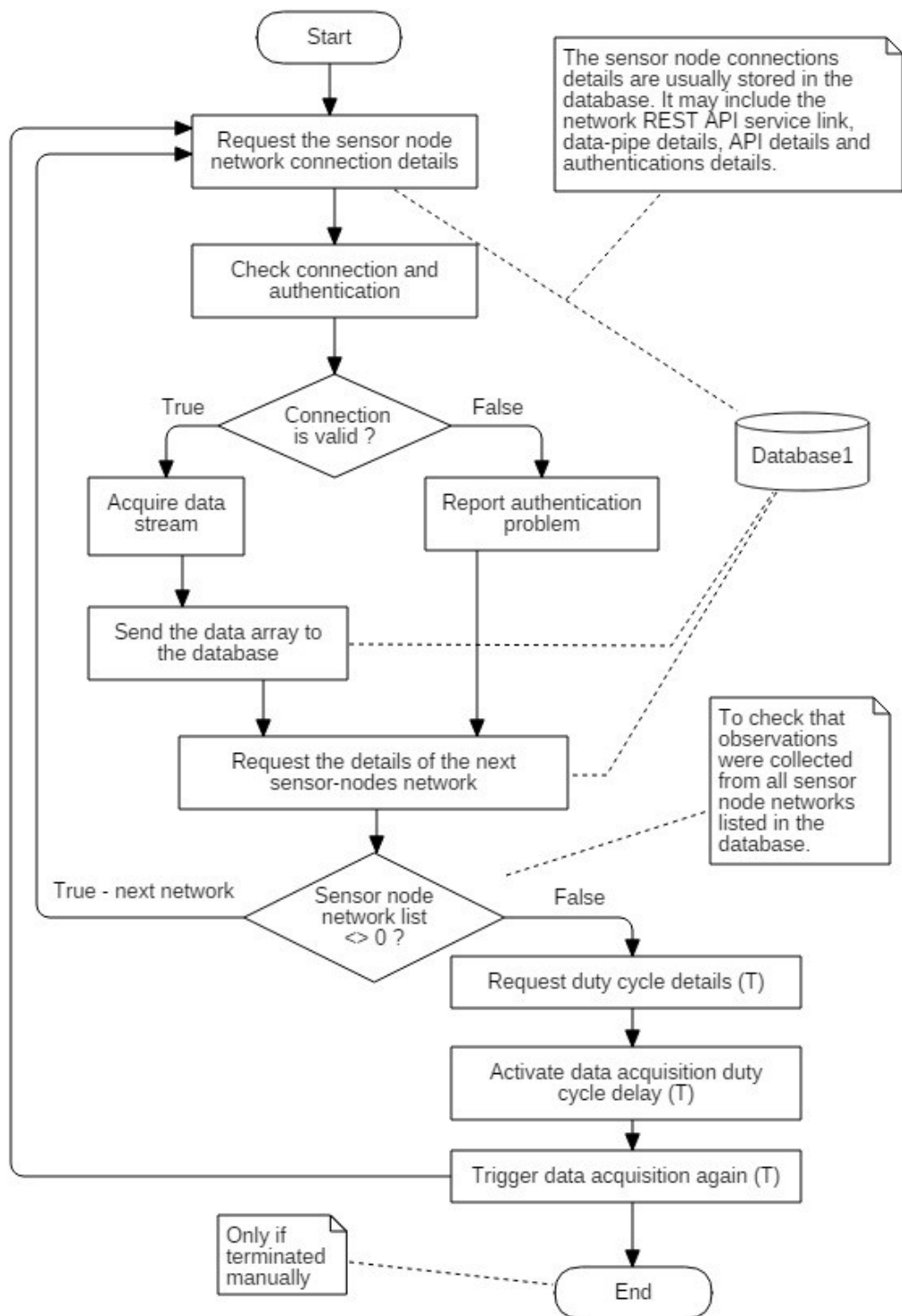


Figure 3.10: The high-level process diagram of the data acquisition unit.

utilises many data mining techniques, such as statistical and machine-learning predictive analysis, anomaly detection, time series clustering and timestamp analysis which can be categorised according to their operational mode into:

1. **Online mode**, the system checks data quality issues associated with errors in sensor nodes measurements in real-time. The data quality management system assesses the quality of the sensor nodes observations against the following four primary data quality dimensions accuracy, timeliness, completeness, and consistency in real-time.

In this context, the real-time notion means that the data quality evaluation of an observation must be completed before receiving the next observation from the same sensor node for all sensors in the network. In other words, if the data quality assessment unit completes the data quality checks in a shorter interval of the shortest duty-cycle of all sensor nodes in the network, then it is considered that the system had satisfied the real-time constraint.

Predictive analysis models and Anomaly analysis models were adopted, focusing mostly on evaluating the accuracy of observations based on their temporal correlation with older observations from the same sensors or their spatial correlation with other neighbouring sensor nodes observations, respectively.

2. **Offline mode:** the systems' components that typically analyse all sensor nodes time-series simultaneously to evaluate their compliance with data quality constraints. Unlike the online mode, which relies on short simple outliers' detection approach. The offline mode is mainly based on detecting long segmental outliers', which requires checking a window of time-series (set of observations). Therefore, in the offline model, the data quality management system performs much less routine data quality checks in comparison with the online mode. It may be triggered once every six hours or once a day.

The offline mode is useful for detecting systematic errors due to hardware failures in sensor nodes or communication networks and data quality issues related to sensor nodes observations consistency and geographical location accuracy.

Figure 3.11 shows the mode-model structure of the data quality assessment unit. Sensor nodes data stream is, typically, composed of a long sequence of observations forming long time-series. Theoretically, a sensor node data stream is infinite. Therefore, a time range of the data stream must be specified to be dealt with according to a model which can be specified to suit applications needs (Wang et al., 2016, p. 100-103). The most relevant data stream handling models (approaches) are:

- **The Snapshot Model:** a fixed-length data stream window specified between two pre-defined timestamps. The length of the snapshot window varies according to applications requirements.
- **The Landmark Model:** The data stream window ranges between the first fixed timestamp and the current time timestamp where the second timestamp shifts forward when new observations arrive.
- **The Sliding Window Model:** the length of the data stream window is specified without explicitly defining its starting the endpoints of the data stream window. When new observations arrive, the data stream window slides to cover the new range of observations without changing its interval.

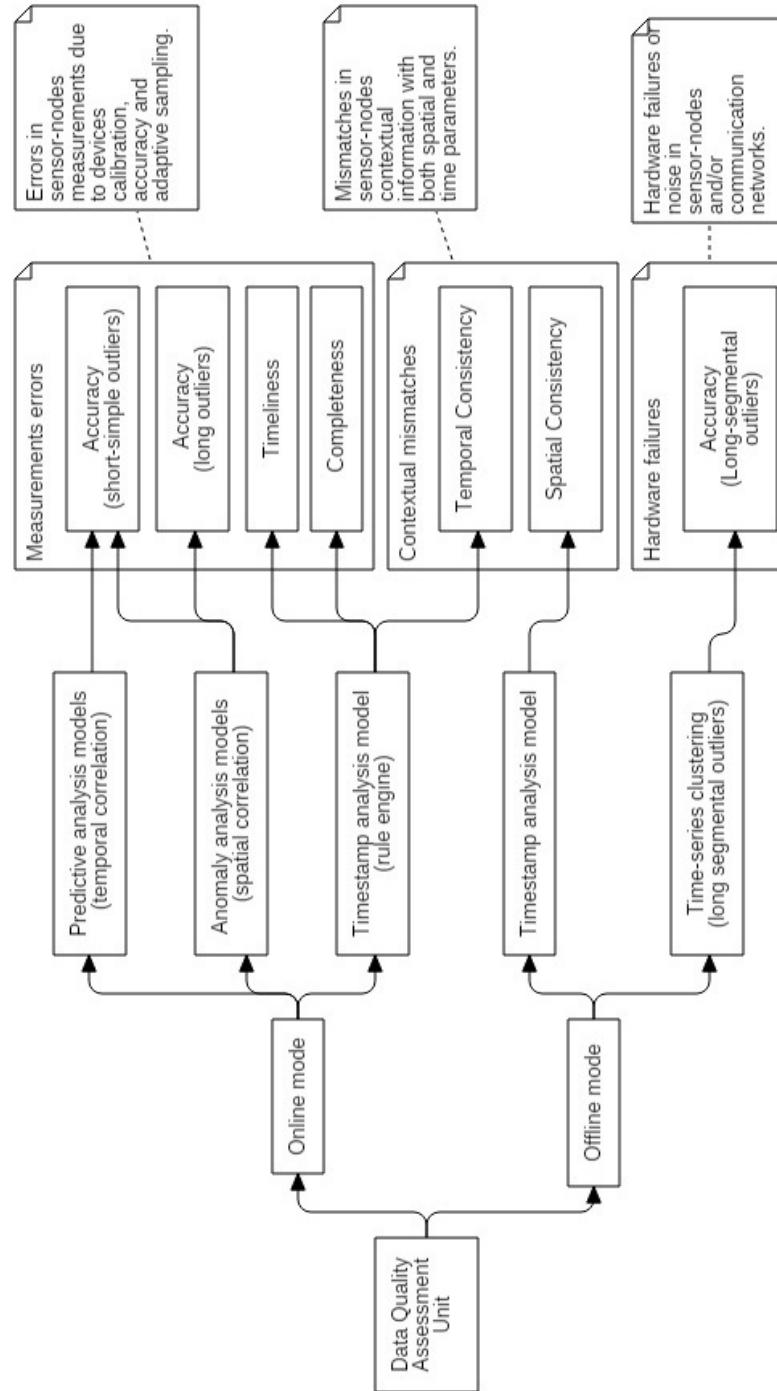


Figure 3.11: The mode-model structure of the data quality assessment unit and data quality dimensions.

3.4.4 Selecting Data Analysing Methods

There is a wide range of data analysis or data mining methods in the literature that can be applied to address the objectives of this research. The choice of the data analysis techniques is mainly based on the features or functions that these analysis techniques can provide, such as predictive analysis, clustering, and time-series clustering. Each of these techniques can be implemented using many different methods. In this research, the selection of the data analysis methods is based on comparing their features while focusing on the following aspects:

1. **Accuracy:** the precision or the correctness of the results produced by the applied method.
2. **Performance:** How fast the data analysis method is, what kind of computational power it requires, and how much time is required for training its model.
3. **Automation:** evaluating the possibility of fully automating the selected data analysis method for real-time applications.

Furthermore, choosing data analysis methods is also depends on the characteristics of the investigated data stream, mainly the existence of the trend and the seasonality which can be determined using Time-Series Decomposition (TSD) techniques.

3.4.5 Time-Series Decomposition

Time series decomposition is a graphical technique that provides a better understanding of the characteristics of time-series by splitting it into three pattern-categorised components (Montgomery et al., 2015, p. 15):

- **Trend** is a progressive increase or decrease in the value of the investigated attribute, which can be a slow long-term or a rapid short span change, Figure 3.12b.
- **Seasonality** is a constant repetitive behaviour within the time-series, such as, every day, week or year, Figure 3.12c.
- **The remainder** is the residue from a time-series after taken out both the trend and the seasonality components from it, Figure 3.12d.

Time series decomposition can be additive or multiplicative, in additive decomposition, a time-series Y_t can be defined as $Y_t = T_t + S_t + R_t$ where T_t , S_t and R_t are the Trend, Seasonality and the Remainder components, all at interval t . An example of the additive time-series decomposition is shown in Figure 3.12, using a time-series collected from a single real-world, temperature sensor node.

Additive decomposition is most suitable for time-series with seasonality that does not correlate with the trend. If the magnitude of the seasonality varies with the trend, as shown in Figure 3.13³, the multiplicative decomposition is more appropriate to be applied on the time-series. In multiplicative decomposition, a time-series Y_t can be defined as $Y_t = T_t \times S_t \times R_t$.

Figure 3.12b shows that temperature has a trend which gradually increases or decreases over days of slow-changing. Also, it has a clear daily seasonality (Figure 3.12c) where the temperature varies a few degrees from its lowest at the early hours of the morning and rises to a peak level in the afternoon. The time series decomposition showed that temperature time-series has a clear trend and seasonality, which indicates that it is possible to apply predictive analysis on this type of time-series.

³<https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/>

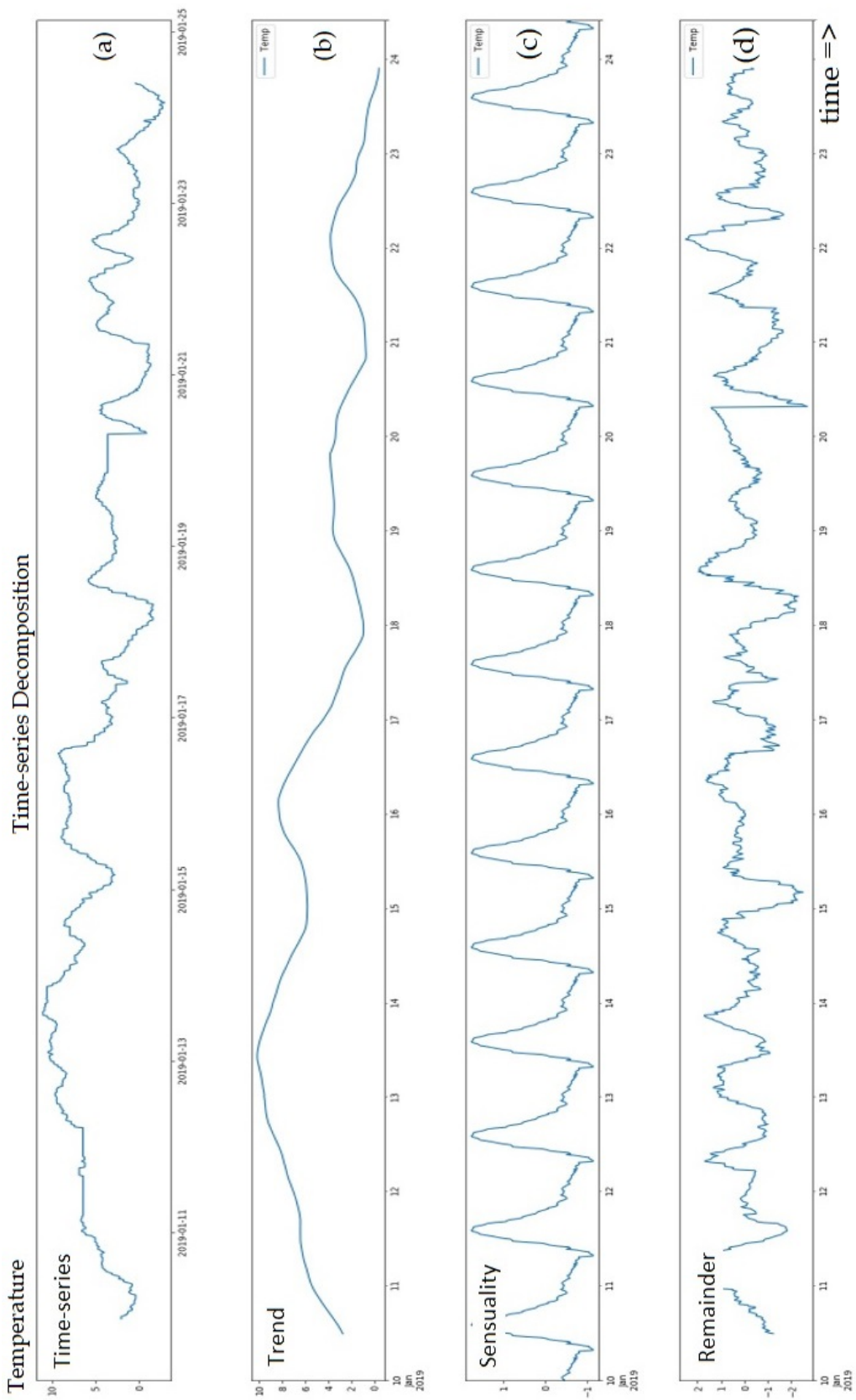


Figure 3.12: Additive time-series decomposition, a time-series of a single real-world, temperature sensor node.

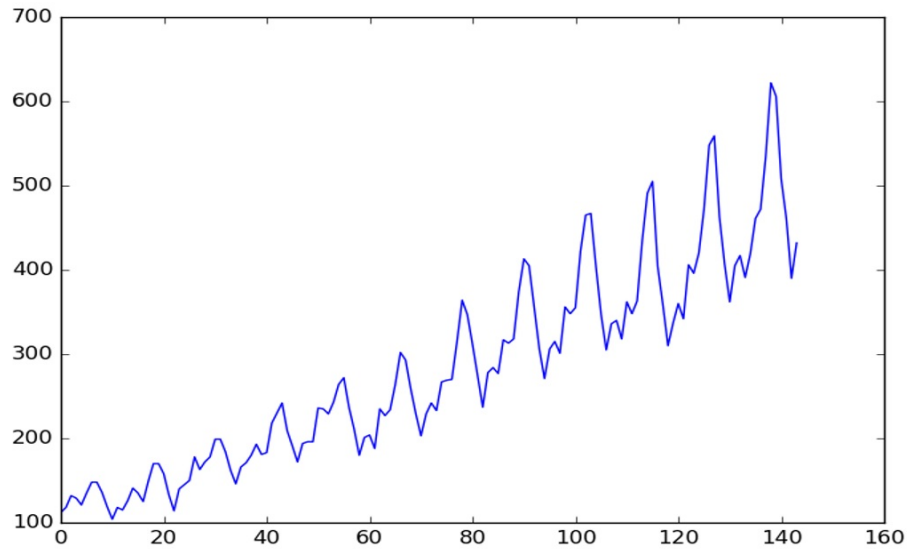


Figure 3.13: An example (demo data) of a time-series with a seasonality that its magnitude varies with the trend (Brownlee, 2017a).

3.5 Online Mode - Predictive Analysis Models

Accuracy assessment for the value attribute of observations was investigated based on the predictive analysis (regression) approach and using statistical and machine learning techniques within the online data quality assessment unit. This section outlines the design details of the predictive analysis models. The results interpretation and conclusions are detailed in Chapters-4. Predictive analysis models depend on the temporal correlation among sequential observations to predict forthcoming observations. Theoretically, predictive models can detect anomalies in observation's values based on comparing the predicted values with the actual observations. The high-level process diagram of the predictive data accuracy assessment model is shown in Figure 3.14.

All of the predictive analysis techniques used in the context of this research were evaluated using temperature observations from real-world sensor node networks. These observations form long time-series of each sensor node with only one independent variable (temperature). Therefore, sensor nodes time-series may

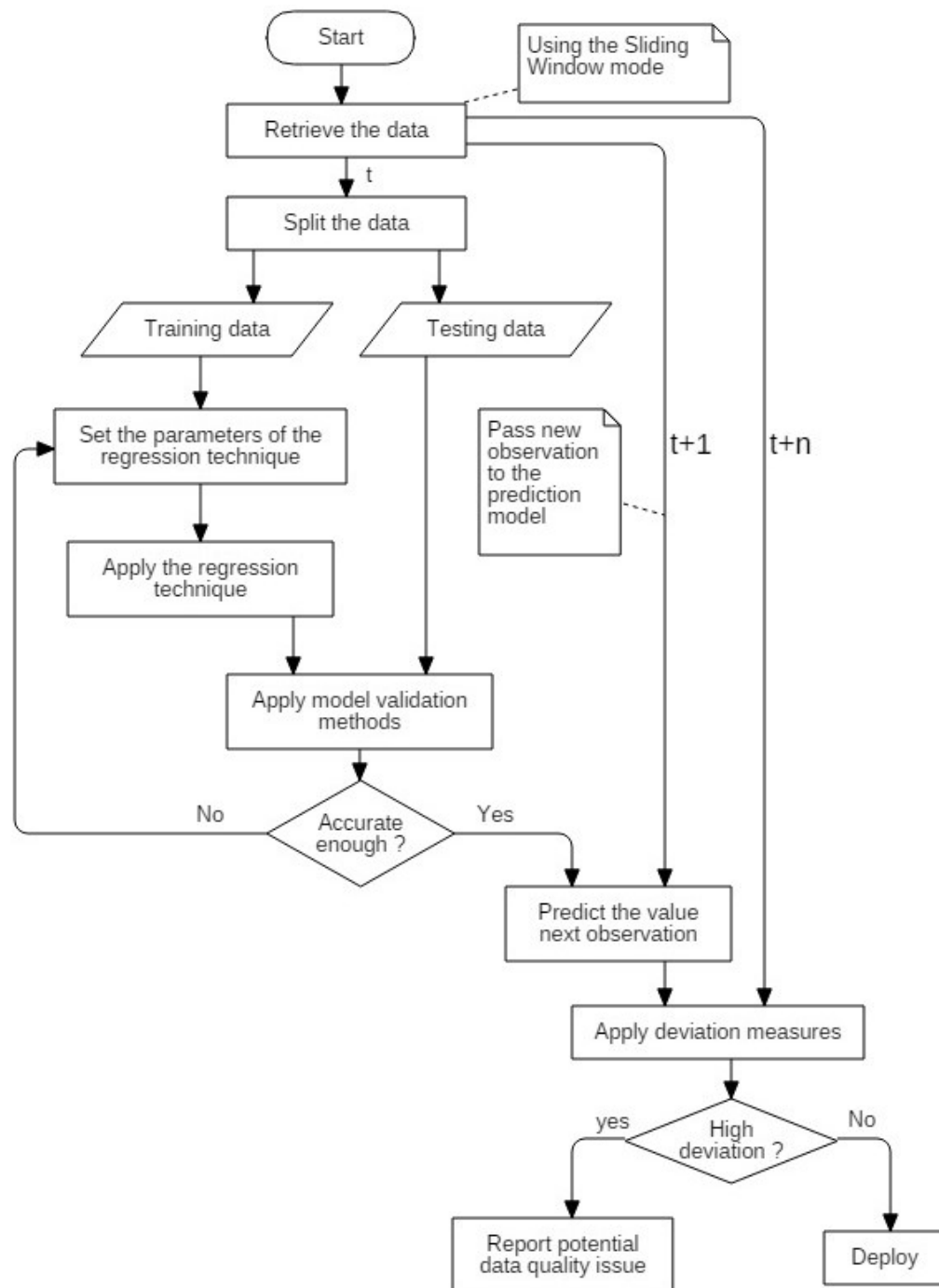


Figure 3.14: The high-level processes of a predictive data accuracy assessment model.

require to go through a sequence of preparing steps before being fitted into a predictive model, such as attribute reduction and aggregation. These steps vary from a model to another, depending on the used prediction analysis technique.

In general, predictive models must be tested using a dataset that is different from

the dataset used for training the model, in order to eliminate the possibility that the model may perform well if tested using the same training dataset but poorly with a different distinct data. Typically, the time-series of sensor node can be randomly divided into a training and testing datasets, as shown in Figure 3.14.

In this research, predictive models were tested and evaluated based on three parameters: accuracy, performance and automation feasibility. The main difference between these models was the algorithms or techniques used to construct them. The next step is to empirically test and verify these newly developed models using the ideal real-world observations collected from the high-quality sensor node network of the University of East London and ultimately using the real-world observations collected from the large-scale sensor node network distributed around London. The accuracy, performance and the feasibility of automation were compared and evaluated for each of the tested predictive analysis models.

The regression algorithms and techniques examined within the context of this research are: simple statistical forecasting methods, Holt-Winters Seasonal method, Autoregressive moving average (ARMA), Non-Seasonal Autoregressive integrated moving average (ARIMA) and Seasonal ARIMA Models. In addition, two advance machine learning techniques were also empirically investigated; Gaussian Process Regression and Long short-term memory recurrent neural network (LSTM-RNN), as shown in Figure 3.15.

3.5.1 Simple Forecasting Methods

Some of the most well-known simple forecasting methods are:

1. **Average method:** all predicted values are equal to the average (or “mean”) of the values of historical data.
2. **Naïve method:** the predicted value is equal to the latest observation, it is also known as the Random Walk because it works well with data that have

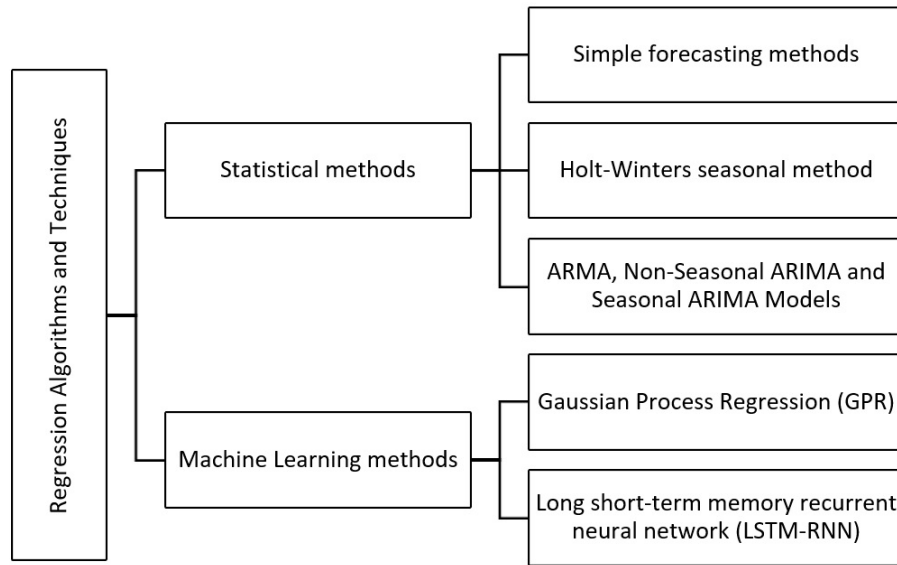


Figure 3.15: Predictive analysis techniques examined in the context of this research.

random patterns such as some economic and financial time series.

3. **Seasonal naïve method:** a prediction method that only can be applied to data with clear seasonality. The seasonal naïve method is based on setting each prediction to be equal to the last observed value from the same time as the previous season, such as previous year, months or quarters (Hyndman & Athanasopoulos, 2018, p. 57-58).

3.5.2 Holt-Winters Seasonal

Holt-Winters seasonal (H-Ws) method is a statistical forecasting technique that involves predicting observations of time-series that exhibit a trend and seasonality patterns. H-Ws prediction model is based on a forecasting and three smoothing equations for the level (L_t), trend (T_t) and seasonality (S_t). It also utilises three smoothing constants (α , β , γ) corresponding with each smoothing equation (Lee et al., 2013, p. 947-950). The forecasting equation of Holt-Winters seasonal method can be used with additive or multiplicative time-series (Islam & Watana-palachaikul, 2012, p. 44). Holt-Winters additive forecasting function (F) for the

time period (f) is⁴:

$$F_{t+1} = L_t + T_{t+f} + S_{t-s+f} \quad (3.2)$$

Where (L_t) is the estimated value of the level of a time-series at a time (t), the level is the simple exponential smoothing of a time-series, as shown in Equation 3.3.

$$L_t = \alpha(y_t - S_{(t-f)}) + (1 - \alpha)(L_{(t-1)} + T_{(t-1)}) \quad (3.3)$$

Where α is the smoothing constant of the level component, it adjusts the rate at which the weighted average of all observations y_t of time-series (y_1, y_2, \dots, y_T) (Hyndman & Athanasopoulos, 2018, p. 247-257).

The value of α ranges between 0 and 1, if α is close to (0), then the oldest observations in the time-series will have more effect (weight) on the predicted values, otherwise, if α is close to (1), then the most recent observations will have more weight. If $\alpha = 0$ then all predictions will be equal to the mean value of the time-series which is known as the Average forecasting method. If $\alpha = 1$ then, the value of the predicted observation will be equal to the last observation (latest) in the time-series which is known as the Naïve forecasting method (Random Walk). The estimated value of the trend (T_t) of the time-series (y_t) at time (t) is:

$$T_t = \beta(L_t - L_{t-1}) + (1 - \beta)T_{t-1} \quad (3.4)$$

Where β is the smoothing constant of the trend, $0 \leq \beta \leq 1$. The equation of the seasonality component (S_t) is denoted as:

$$S_t = \gamma(y_t - L_{t-1} - T_{t-1}) + (1 - \gamma)S_{t-f} \quad (3.5)$$

⁴Equations 2 to 10.3 are mainly referenced from (Hyndman & Athanasopoulos, 2018).

Where the value of the seasonal component smoothing constant of the seasonal component (γ) is within the range of:

$$0 \leq \gamma \leq (1 - \alpha) \quad (3.6)$$

Based on this brief introduction, the Holt-Winters seasonal forecasting method requires four parameters to be set before fitting its forecasting equation: the smoothing constants α, β, γ and f . The value of f can be determined from the time-series, and it varies from application to another, it can be, e.g., monthly, weekly, daily.

The smoothing constants α, β, γ can be set to an initial value, 0.4 for example, before fitting the training dataset (using the sliding window mode) to H-Ws prediction model, the next step is to compare the predicted values against the test dataset to evaluate the accuracy of the model. The feedback from the evaluation process can be used as a outcome to adjust the values of the smoothing constants. It is possible to repeat these steps to optimise the accuracy of the prediction model and practically determining the smoothing constants automatically, as shown in Figure 3.16.

3.5.3 ARMA, ARIMA and Seasonal ARIMA Models

Autoregressive Moving Average (ARMA) is a statistical analysis model that can perform time-series analysis and prediction based on previous observations. ARMA model, $ARMA(p,q)$, consists of a combination of both Autoregressive (AR) and Moving Average(MA) models and inherits their features (Islam & Watana-palachaikul, 2012, p. 49). In general, **Regression** is a linear correlation between two variables (Lind et al., 2018, p. 380), if a prediction model regresses a variable against itself by using the linear combination of its previous observation, such model is known as **Autoregressive model**, which can be described by the

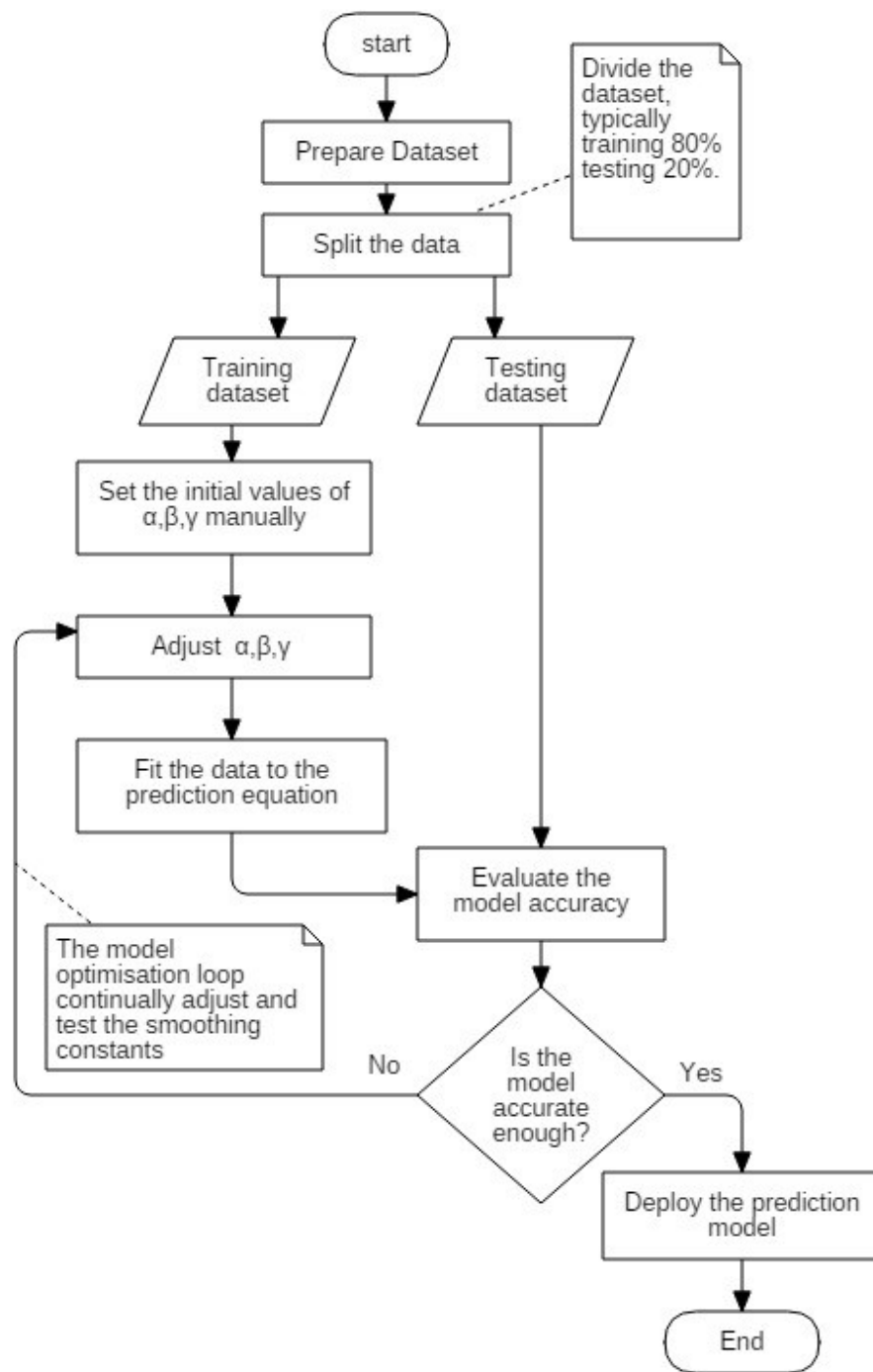


Figure 3.16: The high-level process diagram of Holt-Winters seasonal prediction model.

following equation:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \epsilon_t \quad (3.7)$$

Where y_t is the value of observation at time t . ϕ_p is the least-squares regression coefficient or the weight, which regulate the effect of the previous observations on the model. c is the average of the difference between sequential observations, and ϵ_t is the white noise where $y_t = c + y_{t-1} + \epsilon_t$. The parameter p is known as the order of the autoregressive model which indicates the number of previous observations (lagged) that the model will use in the prediction process.

Autoregressive models are restricted to process stationary data only, alongside other restrictions related to the different combinations of parameters that the model equation can process. For example, for an **AR(2)** model, the value of ϕ must be between -1 and 1, and $\phi_1 + \phi_2 < 1$, (Hyndman & Athanasopoulos, 2018, p. 295-305).

The second component of the ARMA is the Moving Average (MA) model. It is known as the moving average model because it continually changes and moves its predicted value with the timeline when new observations occur in the time-series and neglect the more distant ones (Anderson et al., 2016, p. 820). In the context of ARMA, the MA model uses the weighted average of a q number of the past forecast errors associated with the outcome of the autoregressive model as described in the following equation (Hyndman & Athanasopoulos, 2018, p. 307-308).

$$y_t = c + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \dots + \theta_q\epsilon_{t-q} \quad (3.8)$$

Or,

$$y_t = c + \epsilon_t + \sum_{i=1}^q \theta_i\epsilon_{t-i} \quad (3.9)$$

The parameter q is the order of the moving average model, θ is the coefficient, ϵ is the white noise, and c is the average of the difference between sequential observations. The **MA** model is also restricted to process stationary data only, alongside other restrictions related to the different combinations of parameters of the model equation.

Combining AR and MA reveals the **ARMA** model, it can effectively analysis any time-series based on its previous observations y_t and the ϵ_t errors associated with these observations (Fabozzi et al., 2014, p. 178). ARMA algorithm is described in the following equation:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \dots + \theta_q \epsilon_{t-q} \quad (3.10)$$

As shown in Equation 3.10, the ARMA model is based on fewer parameters comparing with the total number of parameters required by the AR are MA models separately. The high-level process diagram of the ARMA prediction model is shown in Figure 3.17. Using the ARMA model involves three main steps:

1. Checking whether the fitted dataset to the ARMA model is stationary or not if not, a series of differencing and stationary-status tests must be applied on the dataset till it meets the stationary status requirements.
2. Defining the (p, q) parameters that deliver the best prediction accuracy based on setting an initial value of p and q and fitting the model using a routine loop while testing different compensations of (p, q) with each iteration, as shown in Figure 3.17. Next, the prediction model's accuracy is checked against the testing dataset or using the Akaike Information Criterion (AIC), and Bayesian Information Criterion (BIC) tests. The combination of (p, q) that delivers the lowest AIC and BIC value is adopted in the prediction model to optimise its prediction accuracy. The empirical implementation aspects of this method are detailed in Section 4.2.1.3.
3. Verifying the ARMA model adequacy by inspecting the residuals, which should show the characteristics of white noise by not exhibiting any serial correlation. One method to check the residuals serial correlation is the **Q-statistic** test. The result of the test must show no reason to reject the null

hypothesis of no autocorrelation (Fabozzi et al., 2014, p. 178-179).

ARMA models are limited to process stationary datasets that do not have a trend nor a seasonality. Time-series exhibit non-stationary properties can be transferred into stationery using differencing or, and seasonal differencing.

Differencing can stabilise both the trend and seasonality components of a non-stationary time-series. In some cases, applying differencing on non-stationary time-series is not sufficient to stabilise them to the stationary status, in this case, it is possible to apply the differencing on the time-series again which is known as the second-order differencing.

If the time-series is showing high seasonality, **Seasonal differencing** can be applied. It is known as the lag-differencing because it subtracts an observation of the previous consecutive observation. It can stabilise time-series with high seasonality and can be used combined with the ordinary differencing to obtain the stationary status.

The process of testing and differencing time-series before fitting it to the ARMA model may involve a level of complexity and redundancy if applied manually, to tackle this issue, the Non-seasonal Autoregressive Integrated Moving Average (ARIMA) and, the Seasonal ARIMA (SARIMA) models were introduced. The non-seasonal ARIMA includes the integration parameter (d), which presents the differencing order of the model. (d) can be set to one in most cases or two in some cases in order to the dataset to reach stationary status.

Seasonal ARIMA prediction model is capable of processing seasonal time-series. It introduced additional seasonal parameters (P, Q, D) m to the ARIMA model. (P, Q, D) are for the seasonal part of the ARIMA model and m is equal to the number of observations in a single seasonal period (Hyndman & Athanasopoulos, 2018, p. 331-333).

ARMA models (ARMA, ARIMA and SARIMA) were examined within the context

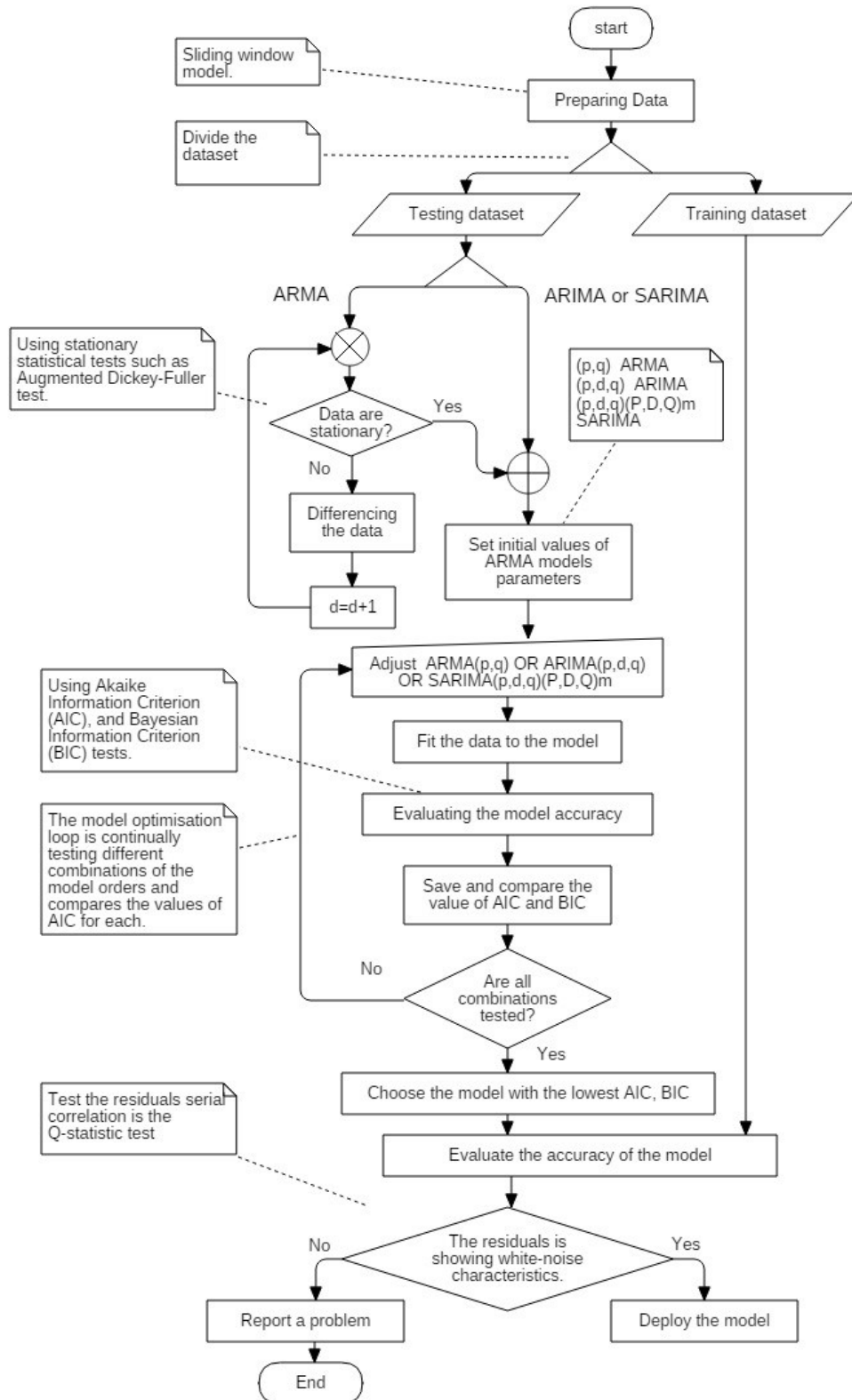


Figure 3.17: The high-level process diagram of the ARMA, ARIMA and SARIMA prediction models.

of this research. The aim was to investigate the possibility of using ARMA models as a predictive analysis mechanism to evaluate data accuracy in large-scale CPS. The tests were designed to compare ARMA models' performance with other regression techniques and compare the different ARMA models. It is expected that ARMA, ARIMA and SARIMA will have the same prediction accuracy; since all share the same mathematical foundation. However, the comparison among them was to evaluate differences in performance and complexity. ARMA models main testing and optimising processes are shown in Figure 3.17.

3.5.4 Gaussian Process Regression

Gaussian Process Regression (GPR) is a supervised machine learning technique utilises Gaussian Processes (GP) algorithm in its forecasting models. GP based regression models are nonlinear and non-parametric models based on the assumption that data regression models can be interpreted by an infinite number of parameters which can be reduced to the necessary number of parameters that can represent the data (Martin, 2018, p. 233).

GPRs model produces a continuous output composed of the mean and variance values of the predicted set of values, where the mean is the value of the prediction with the highest probability, and the variance are the ranges of confidence of the other possible values, as shown in Figure 3.18⁵.

The margins of variance present a continuous measure of confidence with significant importance for identifying outliers or data-accuracy issues of observations detected outside these confidence margins where the accuracy of these margins depends on the number and the accuracy of the observations fitted to the GPR model.

GPR is a machine-learning model that can "learn" from a training dataset to pre-

⁵https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpr_noisy_targets.html.

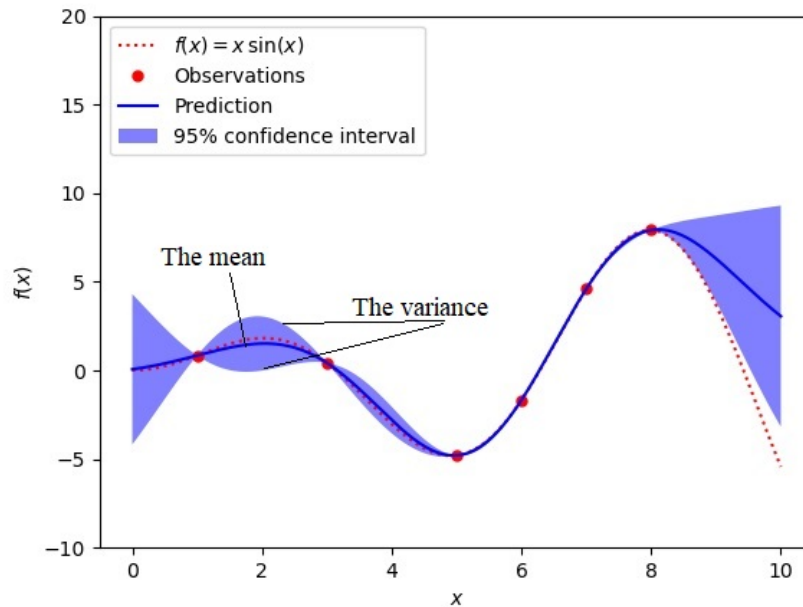


Figure 3.18: A introductory example to Gaussian Processes Regression (Pedregosa et al., 2011).

dict future estimated observations. The mathematical representation between the observations and predictions depends on a set of modelling relations algorithms known as the kernel. The kernel modelling approach is based on measuring the similarity between any two data vectors using similarity functions which self-optimised during the learning phase using the training dataset. The kernel makes predictions based on prior measures acquired from the training dataset.

GPR models can deliver constant predictions even when the input regressor consists of a relatively small number of observations (short time-series), or it may include noisy data (Kocijan, 2016, p. 2-5).

The main processes of the GPR predictive model are similar to the generic regression model shown in Figure 3.14.

3.5.5 Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) networks are the innovation and extension of the Deep Learning-based Recurrent Neural Networks (RNN). LSTM is an improvement to the RNN architecture to become able to process long sequences of data with temporal dependencies or simply, to perform time-series prediction. LSTM models consist of linear-memory cells controlled by three types of gate units which control the flow of data in and out of the memory cell and also control when to clear the cell of any stored information (Swamynathan, 2017, p. 333).

RNN, like many other conventional deep learning techniques, cannot retain or retrieve a long sequence of observations into its prediction model. This limitation is known as the vanishing gradient problem, and it was solved in LSTM version of RNN by adding the gates to the structure of the RNN model.

LSTM prediction model cannot process time-series directly. Time-series must be prepared to become compatible with the data structure requirements of the LSTM input layer, which includes, data normalisation, standardisation and data transforming into fixed, discrete, time-stepped datasets, a process known as "vectorisation" (Brownlee, 2017b, p. 27-35). The following steps outline the main data preparation process:

- Dividing the time-series into a training and testing datasets. The dividing ratio can be 80% training to 20% testing.
- Scaling the values of observations using data normalisation and standardisation processes into a range between -1 and 1 using a scaling coefficient (min and max) to become suitable to be fit to the LSTM model and to eliminate any possible influence of extreme values on the model. This process must be inverted later to return the values of observations to their original order.
- Transforming the dimensions of the training dataset to become compatible

with the data format of the LSTM's input layer, which expects to receive a three-dimensional data array; Samples, Time steps and Features. Thus, it is common to divide long time-series into relatively short sequences (time-windows) of data points, transferring the time series into a multi-dimensional data form (Wang et al., 2021, p. 262). Sensor nodes time-series consist of a sequence of an infinite number of observations. Since the contextual features of time-series, such as timestamps are considered as indices, the time-series can be viewed as a $(1 \times n)$ one-dimensional array, as shown in Figure 3.19. Where (n) is the total number of samples $n \rightarrow \infty$, and (y) is the value of the observation at any time (t) .

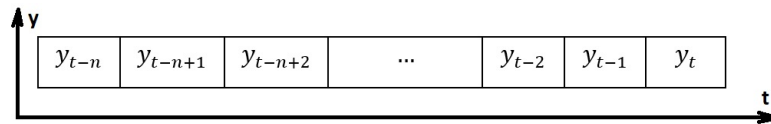


Figure 3.19: Sensor node's time-series as a $(n \times 1)$ array of observations.

Therefore, it is required to transform the dimensions of the time-series to become compatible with the data structure required by the LSTM input layer by dividing the time-series into fixed discrete datasets according to a Time-Steps model, as shown in Figure 3.20.

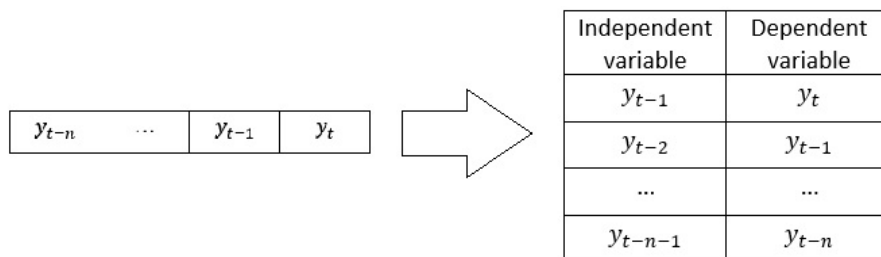


Figure 3.20: Time-series transformation (vectorisation) concept.

Time-steps model divides sensor nodes time-series into discrete rows of datasets where each dataset consists of (N) time-steps of sequential observations. The difference among these rows is a one-timestamp shift of (t) . These rows form a new two dimensional array, as shown in Figure 3.21, where

$N=7$ in this example. The value of the third dimension, "features", will be set to 1 since all the tests of this research were applied using a single feature (ambient temperature) only.

Independent							Dependent
y_{t-7}	y_{t-6}	y_{t-5}	y_{t-4}	y_{t-3}	y_{t-2}	y_{t-1}	y_t
y_{t-8}	y_{t-7}	y_{t-6}	y_{t-5}	y_{t-4}	y_{t-3}	y_{t-2}	y_{t-1}
y_{t-9}	y_{t-8}	y_{t-7}	y_{t-6}	y_{t-5}	y_{t-4}	y_{t-3}	y_{t-2}
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
y_{t-n-7}	y_{t-n-6}	y_{t-n-5}	y_{t-n-4}	y_{t-n-3}	y_{t-n-2}	y_{t-n-1}	y_{t-n}

Figure 3.21: LSTM time-series transforming process, time-steps = 7 in this example.

- Fitting the transformed training dataset to the LSTM model.
- Evaluating the performance of the LSTM model, as shown in Figure 3.22.

The high-level process diagram of the LSTM predictive model is shown in Figure 3.22.

3.6 Online Mode – Anomaly Analysis Models

Anomaly analysis techniques were proposed as a possible solution to detect data quality issues associated with observations accuracy in large-scale cyber-physical systems (CPSs) (Jayswal & Shukla, 2016; Abid et al., 2015; Ayadi et al., 2017). An outlier is an observation with a value that significantly deviates from the values of other reference observations or threshold value. It also referred to as abnormality or anomaly in the literature, and it represents an irregular change in the patterns of observations of real-world sensor node networks (Aggarwal, 2016, p. 1-2).

Predictive analysis models are most suitable for evaluating the accuracy of sensors

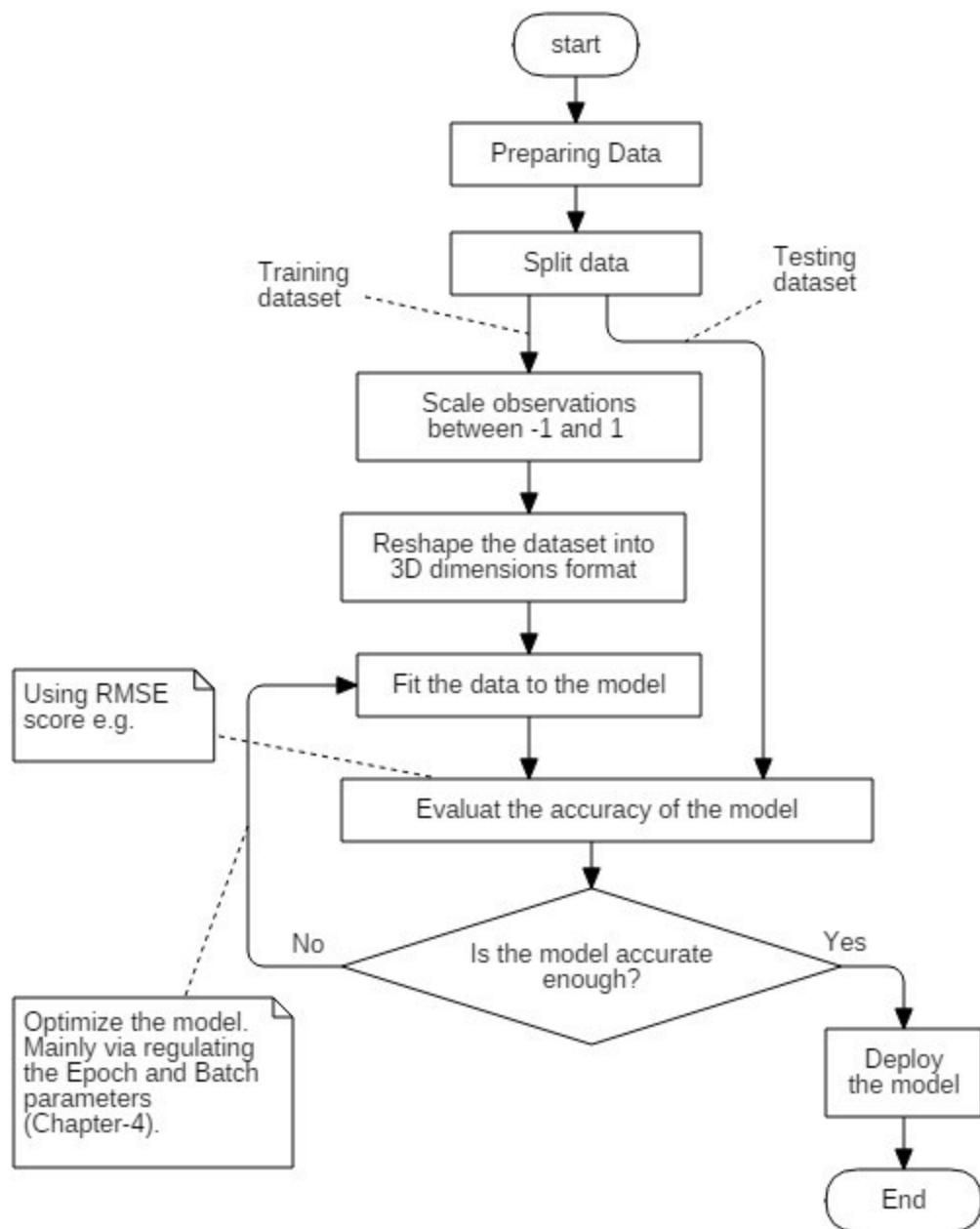


Figure 3.22: The high-level process diagram of the LSTM predictive model.

measurements errors that appear for a short interval (short outliers). Measurement via detecting outliers that occur for a relatively long time effect predictive models ability to render an accurate forecast. The pattern of time-series with long outliers will be distorted to a certain extent reflecting the wrong measurement as the standard pattern, which leads to higher forecast errors and limits the ability of the predictive analysis modes to detect data accuracy issues correctly (Berk, 2015,

p. 25). Therefore, anomaly analysis was investigated to tackle predictive analysis models' limitation.

Large-scale sensor node networks stream data continuously, forming spatiotemporal time-series of observations which occur at a specific time in a particular place (Appice et al., 2014, p. 3), where spatiotemporal time-series have spatial and temporal attributes.

The spatial attributes of sensor nodes observations in large-scale CPSs can be labelled according to its characteristics into contextual or behavioural. Spatial attributes are **behavioural** when they change dynamically with other attributes of the time-series. e.g. data collected from mobile sensor nodes, where sensor's location attributes change with other temporal attributes. In contrast, sensor measurements collected from different sensor nodes that have static locations, such as ambient temperature monitoring stations, are observations with **contextual** spatial attributes. An outlier in contextual time-series is an observation with value attribute is significantly diverts from the value attribute of other spatially correlated observations collected from nearby sensor nodes (Aggarwal, 2016, p. 346). This approach is known as the spatial continuity or spatial autocorrelations which is justified by **Tobler's** law of geography, which states that "everything is related to everything else, but near things are more related than distant things" (Tobler, 1970, p. 234). Spatial autocorrelations can be recognised among the time-series collected from the benchmark, sensor node network deployed at the University of East London which consists of four high-quality wireless temperature sensors, three of which were deployed outdoors, and the fourth was deployed indoors. The distance between the outdoor sensor nodes is relatively small (70 meters). A seven days time-window of the time-series from these sensor nodes is shown in Figure 3.23. As illustrated in Figure 3.23, the time-series of outdoors sensor nodes are showing a high correlation in value attribute, and pattern since all were distributed in a relatively small geographical area and governed by the spatial

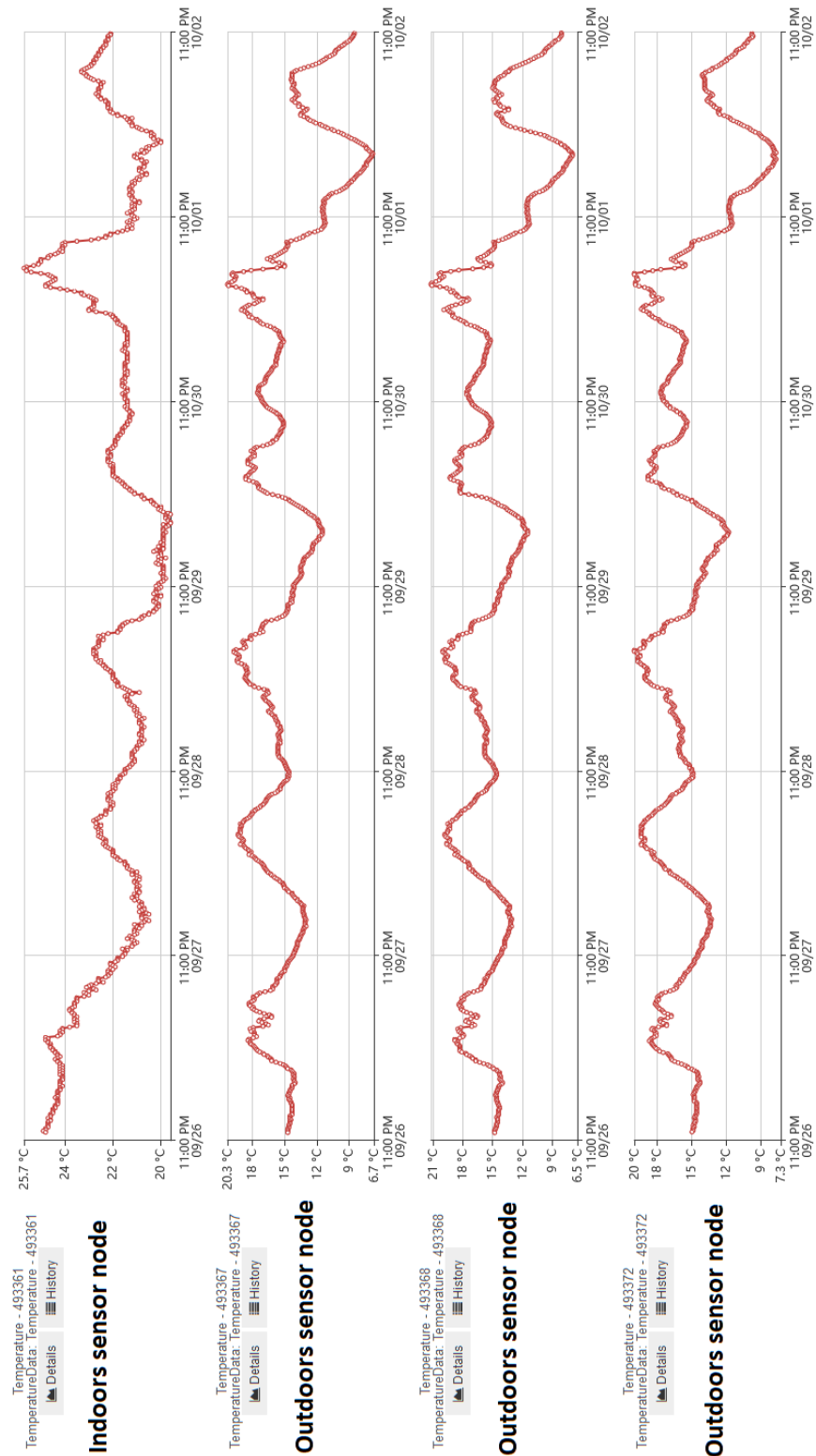


Figure 3.23: Spatial autocorrelations among time-series of ideal nearby sensor nodes.

continuity concept of Tobler's law. The time-series of the indoor sensor node (first) showed a significant correlation in pattern with the time-series of the rest of the sensor nodes associated with a systematic, consistent difference in the value attribute (temperature) of roughly +15 degrees.

However, Tobler's law is not necessarily always valid in large-scale CPSs applications. The same approach of spatial continuity **can not** be applied directly to the real-world observations collected from the large-scale, temperature sensor nodes distributed around London, because of the relatively high distance separating among them and due to a phenomenon known as the **Urban Heat Islands (UHI)**. According to the Met Office⁶, the phenomenon of heat islands is caused by many associated factors (MetOffice, 2019), including:

- Heat released from industrial and domestic facilities, concrete and other building material which observe sun heat during the day and release it back during the night.
- Solar radiation reflected by buildings glass and windows, manufacturing and cars emission which create a cloud of smog trapping the solar radiation inside and building up a pollution dome.
- The absence of strong wind which is significantly blocked and disturbed by tall buildings which increase the surface roughness and reduce the wind speed causing less heat dissipation and preventing cooler air exchange from rural areas.

The irregularity in temperature readings around London due to the impact of Urban Heat Islands is shown in Figure 3.24⁷.

In general, the heat distribution in an urban area depends on many environmental

⁶https://www.metoffice.gov.uk/binaries/content/assets/metofficegovuk/pdf/research/library-and-archive/library/publications/factsheets/factsheet_14-microclimates.pdf

⁷minimum temperatures in C, clear skies and light winds.

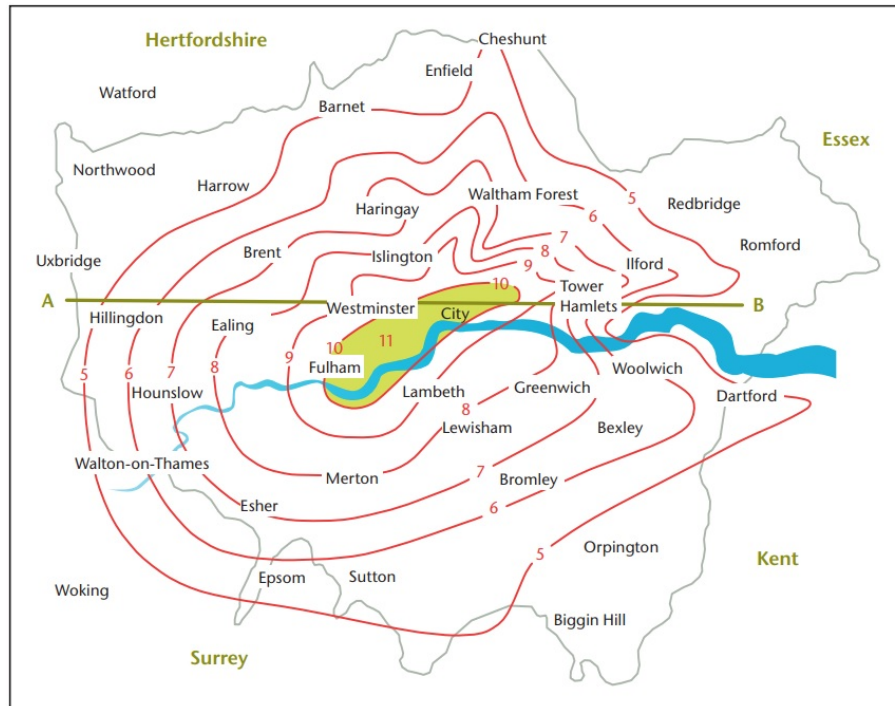


Figure 3.24: The irregularity in temperature levels around London due to the impact of Urban Heat Islands. The line A to B is presented in Figure 3.25, (MetOffice, 2019).

parameters and geographical terrains such as wind speed, humidity, sunshine density, the existence of rivers and the density and height of urban structures. The heat profile map of London is shown in Figure 3.25, where the temperatures in central London may reach 11 degree C⁰ while dropped by 6 degrees C⁰ in the suburbs (MetOffice, 2019; Chandler, 1965).

The phenomenon of urban heat islands may cause unexpected variations in the value attribute of the real-world, temperature sensor nodes observations which violate their spatial continuity (Aggarwal, 2016, p. 346-348). To tackle this issue, Spatial Partition Modelling was adopted and empirically tested in this research. Spatial data partitioning models facilitate correlation between data points which change over the space of interest by breaking up that space into more representative regions for local data points, which do not overlap and do not necessarily correlate with the centres of nearby regions. It is possible to describe spatially partitioned two-dimensional data using **Voronoi** tessellation method, as shown in

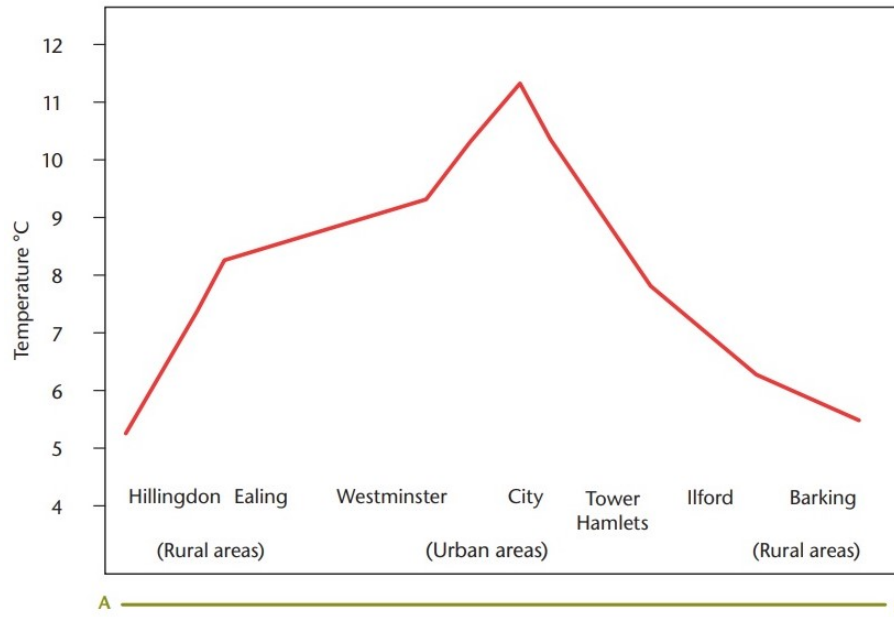


Figure 3.25: The heat profile map of London highlighting the impact of urban heat islands, (MetOffice, 2019).

Fig 3-25, where the dots are the centres of clusters or regions, also known as tiles (Lawson & Denison, 2002, p. 125-126).

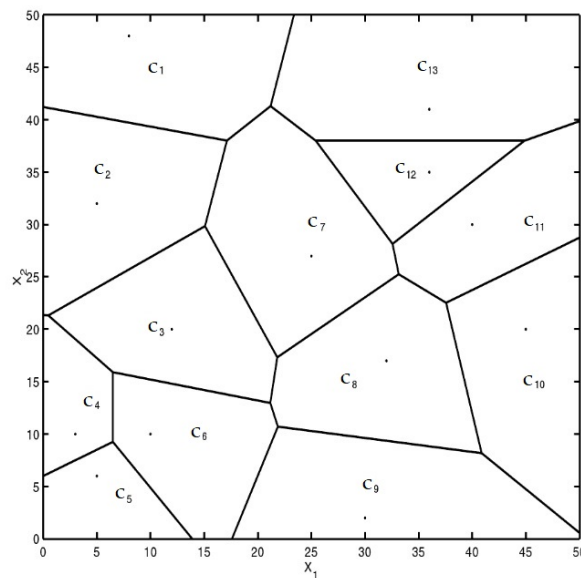


Figure 3.26: Voronoi tessellation method to describe spatially partitioned two-dimensional data, (Guo et al., 2003, P. 126).

Partitional clustering can be categorised into three main types: distance-based, model-based (distribution-based) and density-based (Guo et al., 2003, P. 232).

Model-based clustering is a probability distribution model based on the assumption that data-points in each cluster have a specific pattern of distribution, and the entire region of interest may have several distribution models (Bouveyron et al., 2019, p. 2-4).

Model-based clustering does not utilise similarity measures to partition data into groups (Bouveyron et al., 2019, p. 15). Data points grouped by model-based clustering are not necessarily spatially correlated, which does not support the approach of spatial continuity, and thus it was not investigated further in this research.

Distance-based and density-based clustering models were empirically investigated as spatial partitioning techniques for outlier detection based on K-means and DBSCAN clustering techniques, respectively. Both clustering methods were tested using the real-world observations collected from the large-scale temperature sensor node network distributed around London. K-means and DBSCAN were utilised as spatial partitioning techniques to label sensor nodes (cluster) according to their relative distance and density similarity measures.

Spatial partitioning clustering models were not applied to the ideal dataset collected from the benchmark sensor node network because its sensors were located in a relatively small geographic area (inside the University of East London / Docklands campus, within a 70 meters distance separating the wireless sensor nodes), and they are roughly at the same geographical location.

3.6.1 Distance-Based Spatial Clustering (K-means)

K-means is a distance-based clustering algorithm divides unlabelled dataset into k number of non-overlapping subsets (clusters) each of which is represented by the mean of the distance between its data points (Hartigan & Wong, 1979). The number of clusters (k) must be provided as an input parameter to the K-means algorithm which randomly allocates k number of data-point as the initial clusters

representatives (centroids) and assigns the remaining data points to their nearest centroid point. K-means then, re-allocates the clusters' centroids by calculating the mean of the data points of the same cluster and keeps repeating this process until the change becomes minimal or reaching a threshold criterion (Bhattacharyya & Kalita, 2013, p. 73).

K-means partitions unlabelled dataset observations into k clusters where each observation is assigned (labelled) into a cluster, the result is partitioning the dataset into Voronoi tiles (cells) (Brunton & Kutz, 2019, p. 164), as shown in Figure 3.26. A significant challenge of using K-means is how to estimate k , the number of clusters, the most commonly solutions used to tackle this issue are; a graphical technique known as the Elbow method and a metric technique known as the Silhouette analysis:

- **Elbow method** is based on applying K-means on the dataset using a range of K values and calculating the value of the sum of squared error (SSE) for each K. The next step is plotting the value of SSE for each K and locate the elbow point on the graph line. The best value of K is at the elbow point, which depicts the smallest number of clusters (k) with the lowest SSE (Swamynathan, 2019, p. 199-201).
- **Silhouette analysis method** evaluates the consistency of a cluster by measuring how closely are the data-point within the cluster and how the clusters are separated from each other. The value of the Silhouette coefficient (score) ranges from -1 to 1, where the following equation defines the Silhouette coefficient score:

$$\text{Silhouette Coefficient Score} = (x - y) / \max(x, y)^8 \quad (3.11)$$

x is the average distance between data-points in the same cluster and y is the

⁸(Kumar, 2016, p. 267)

average distance between data-points in nearby clusters. High silhouette coefficient indicates that all data points within the cluster are closely correlated and all clusters are very separated apart (Kumar, 2016, p. 267-268).

Since the Silhouette analysis method can provide a quantitative (metric) measurement to evaluate the accuracy of the clustering algorithm, it will be used in this research to estimate the optimum number of clusters in K-means. The main processes used to determine the optimum number of K-means clusters based on the Silhouette analysis method are shown in Figure 3.27.

K-means was applied, as detailed in Chapter 4, Section 4.2.2.2, using the snapshot data model on the spatial attributes of all available sensor nodes in the large-scale network. K-means was initiated with $K = 2$ and applied for n times where n is the maximum number of iterations, and it is equal to the total number of available sensor nodes -1. With each iteration, K gets increased by one ($k = n$ and $n = n + 1$), and the Silhouette coefficient was registered. K-means model with the highest Silhouette coefficient was selected, and its associated K parameter was considered as the optimum number of clusters.

3.6.2 Density-Based Spatial Clustering (DBSCAN)

Density-Based Spatial Clustering of Applications with Noise (DBSCAN), a partitioning algorithm, identifies clusters based on the density of points within a specified radius. It separates regions that are more crowded with data-points (sensor nodes in this case) from relatively less crowded regions. Unlike the K-means, DBSCAN does not need the number of clusters to be pre-set as an input parameter. Alternatively, it requires two parameters, "MinPoints" and Eps where "MinPoints" is the minimum number of data points within the radius Eps (epsilon). DBSCAN categorises data points into core, border and noise points, as illustrated in Figure 3.28, (Raschka & Mirjalili, 2017, p. 372-377).

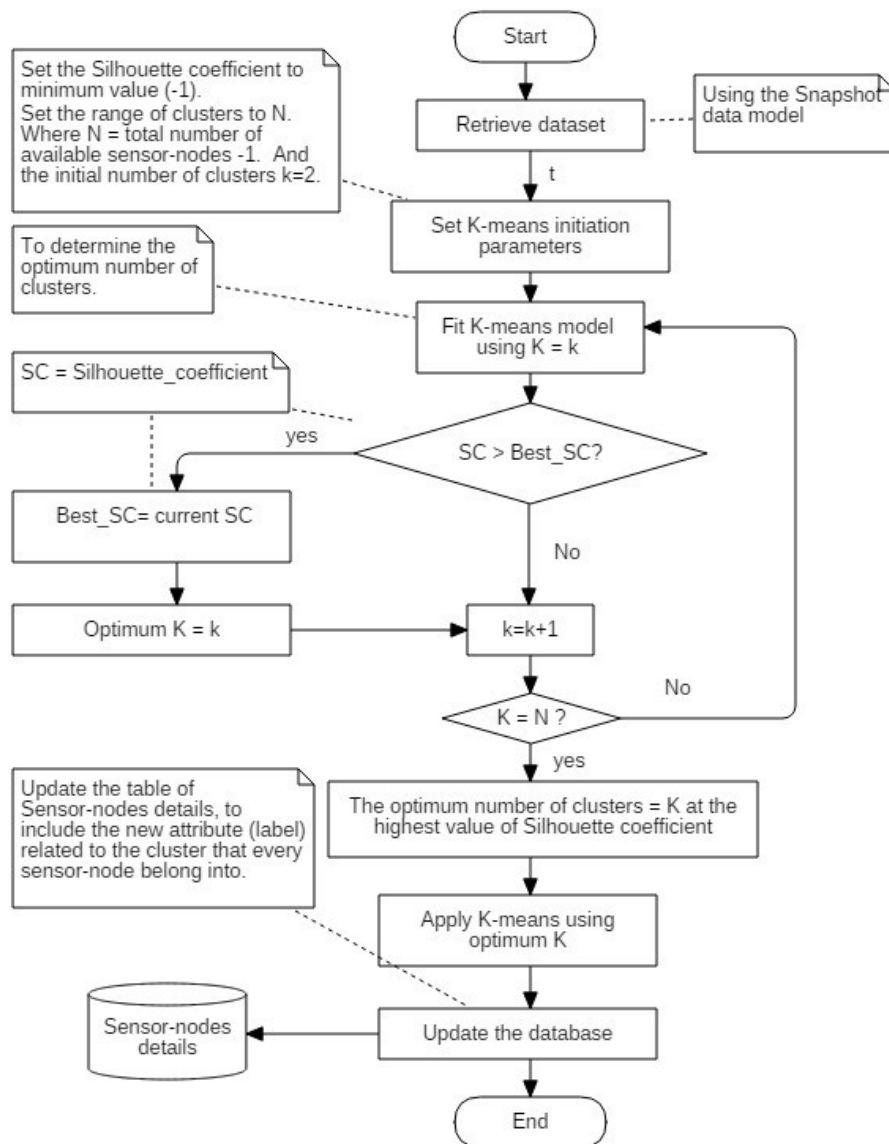


Figure 3.27: K-means partitional clustering model flowchart diagram illustrating K estimating process using the Silhouette coefficient analysis method.

DBSCAN labels a data point as a core point if it has at least the number of "Min-Points" neighbouring data points within the radius Eps. Data points which lay within the radius of the core point but may have less than "MinPoints" neighbouring points are considered as border points, and any other data points which are not core nor border data points are considered noise, Figure 3.28.

DBSCAN assigns a cluster for each core point, except if there are more than one core point within a range of Eps, in this case, DBSCAN connects these core

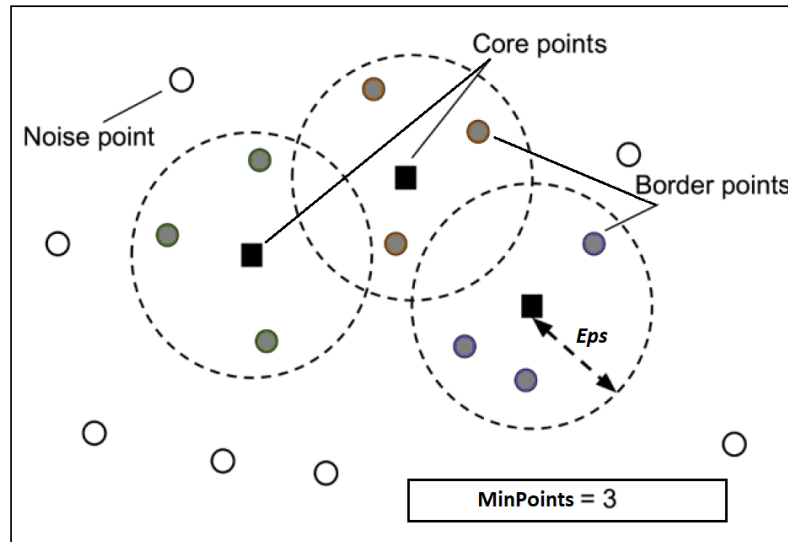


Figure 3.28: DBSCAN parameters and data points categorisation (Raschka & Mirjalili, 2017, p. 373).

points into one cluster and assign border points according to their associated core points (Bhattacharyya & Kalita, 2013, p. 77-79). DBSCAN can capture complicated nonlinear shaped clusters, and it is able to identify points which are not included in any cluster as noise. DBSCAN has a slower performance comparing to k-means, but still, it can deal with relatively large datasets (Müller et al., 2016, p. 189-190). DBSCAN determines the number of clusters based on the pre-set radius parameter Eps and the minimum number of data points (sensor nodes) within that radius. The main challenge associated with applying DBSCAN is how to determine the best Epsilon Eps parameter since the “MinPoints” parameter can be assigned to two (“MinPoints” = 2) to reflect the minimum number of sensor nodes required to establish a deviation comparison for outlier detection. To tackle the issue of determining the Epsilon parameter of DBSCAN, the Silhouette analysis method was adopted to estimate the optimum value of Eps based on the best clustering performance. DBSCAN was initiated with $Eps = \text{minimum possible value}$ and applied for n times where n is the maximum number of iterations.

With each iteration, Eps gets increased by a small fraction f , $Eps = Eps + n * f$, and the Silhouette coefficient was registered. DBSCAN model with the highest Sil-

houette coefficient was selected, and its associated Eps parameter was considered as the optimum Epsilon value, as shown in Figure 3.29. The empirical test of the DBSCAN model is detailed in Chapter 4, Section 4.2.2.3.

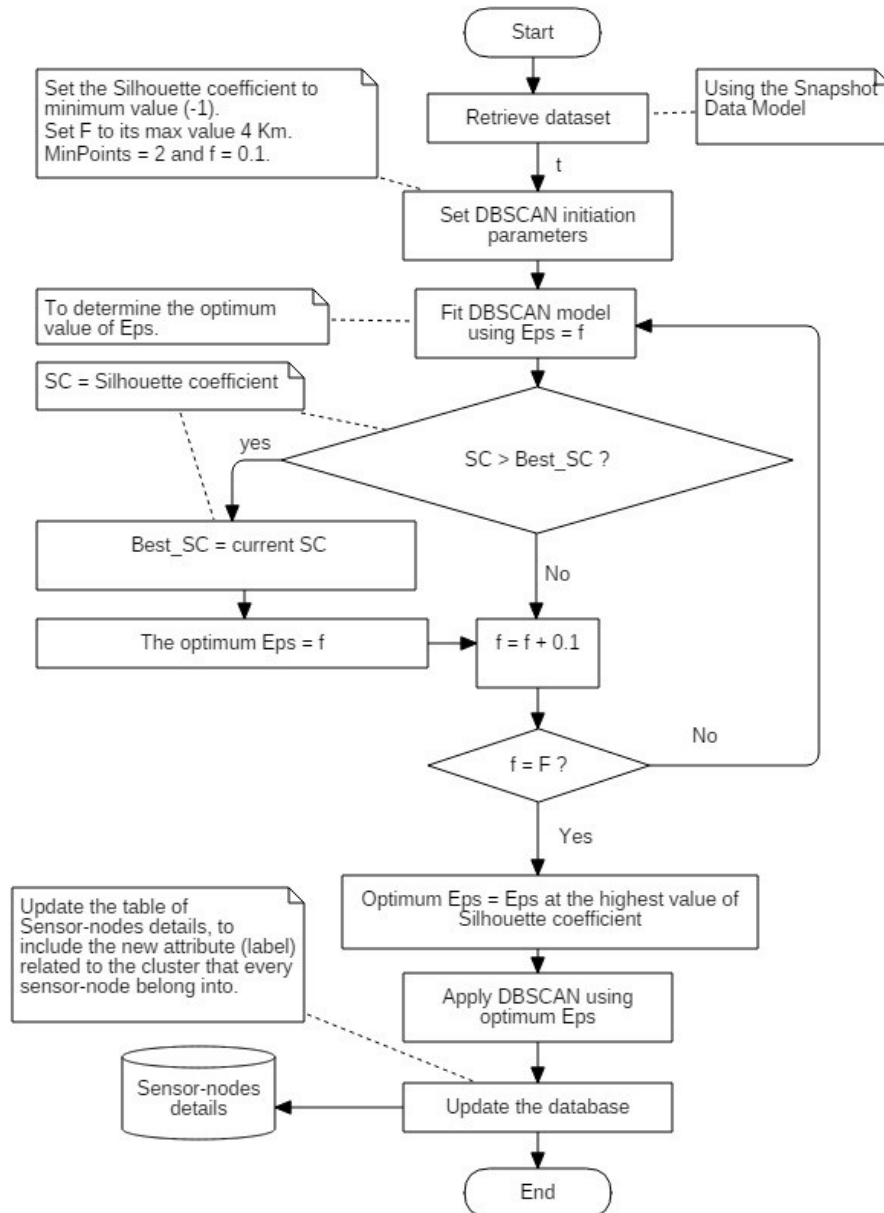


Figure 3.29: DBSCAN partitional clustering model flowchart diagram illustrating Eps estimating process using the Silhouette coefficient analysis method.

3.7 Online Mode - Timestamp Analysis (Temporal Consistency)

Time-series are a structured form of data consisting of a series of observations measured at different time points. Typically, time-series consist of data-points that occur according to a regular interval, e.g., every minute, hour, day or week. A dataset is a time-series if it exhibits timestamps with each data-point, regularity or time intervals, indicating the intervals among different timestamps (McKinney, 2017, p. 323). Active sensor nodes in large-scale CPS applications stream a constant georeferenced and timestamped time-series of numeric observations which are equally spaced in time. However, sensor nodes may become inactive and do not stream observations for an interval of time (Appice et al., 2014, p. 4). Sensor nodes in large-scale CPSs are vulnerable to many external and internal effects, as detailed in Chapter 2, that may impact the integrity of their measurements by compromising the **timeliness** (Rager et al., 2018; Auger et al., 2016) **completeness**, (Li et al., 2019; Togneri et al., 2019) and temporal or spatial **consistency**, (Togneri et al., 2019; Krishna, 2018; Liu et al., 2019) of their data stream. An example of a typical real-world time-series with data inconsistency issues is shown in Figure 3.30. Therefore, the ideal-case assumption that sensor nodes observations are evenly distributed in time, and every observation will be delivered in time with its associated timestamp is unrealistic in real-world scenarios. Thus data inconsistency issues may occur due to sensor nodes internal issues or due to limitations in data collection methods (Chu, 2014, p. 60-61).

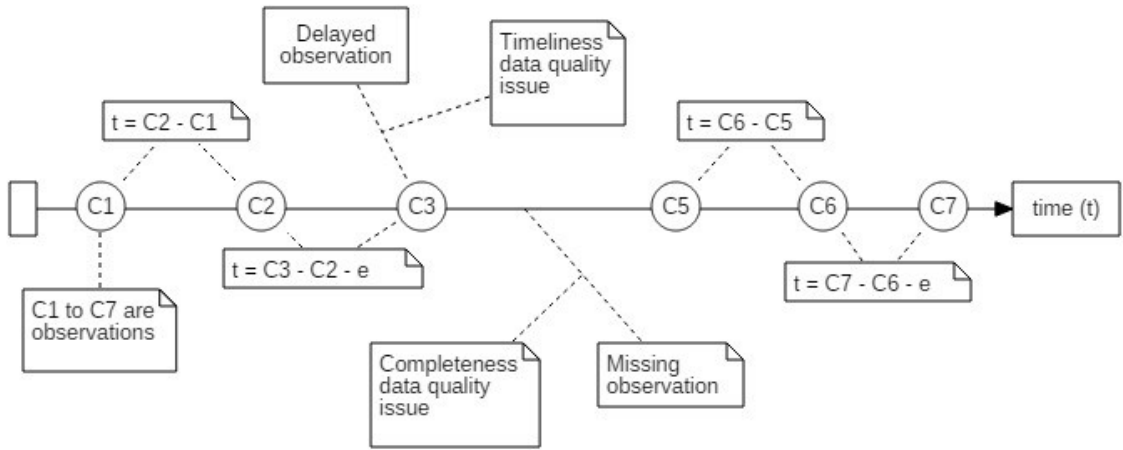


Figure 3.30: An example of a typical, real-world time-series with data inconsistency issues.

In order to detect timeliness, completeness and temporal-consistency data quality issues in sensor nodes' time-series, time-series periodicity mining ("periodic pattern mining") technique was investigated in this research.

In general, periodicity in time-series is the tendency of observations to recur at regular intervals (Otunba et al., 2014, p. 793-804). Periodicity analysis techniques consider time-series as a sequence of symbols where each symbol is associated with a timestamp and presents an event (observation) which do not necessarily occur precisely at the same point of time in each cycle due to inherited offset or noise (Chu, 2014, p. 45) as shown in Figure 3.30.

Assuming that the duty-cycle of an active sensor node S is t_{dc} then, t_{dc} is equivalent to the interval between any two timestamps in an ideal case, $t_{dc} = C_t - C_{t-1}$, where C_t is the timestamp of the current observation and C_{t-1} is the timestamp of the previous observation of the same sensor node. However, in real-world scenarios, observations may be delayed or even missed, which can be reflected by the time offset coefficient (e), where:

$$t_{dc} = C_t - C_{t-1} - e \quad (3.12)$$

Where $e = 0$ in the ideal-case in which sensor nodes observations arrive in a timely manner. Therefore, it is possible to estimate active sensor nodes duty-cycle from their data stream using the principle of periodicity analysis as follows:

1. Applying the one time-step data transformation model to transform the data stream into a two-dimensional array of C_t, C_{t-1} sets of observations to calculate the interval between every two sequential observations in the time-series, as illustrated in Figure 3.20. The shortest interval calculated by $C_t - C_{t-1}$ will be considered as the duty-cycle of the related sensor node (t_{dc}) at the server-side associated with the shortest offset coefficient error e .
2. Aggregating these intervals with their associated offset error coefficient. The aggregation of observations intervals will render a definite number of possible intervals between observations with a score (a rank) for the most recurrence intervals. A proximity function, such as rounding, may be applied in this step to eliminate the effect of the intervals decimal fractions (millisecond fractions associated with the minutes duty-cycle) on the aggregation outcome.
3. Identifying the interval with the highest recurrence score as the Threshold Interval t_{Ths} to use it as a reference to define the temporal consistency status of all observations from the related sensor node.

A rule-engine was developed inside the database of the data quality management system using SQL (Query Structured Language) to provide real-time insights about the temporal consistency of each observation. The SQL rule-engine is built-in inside the database of the system. Thus it can evaluate the timeliness, completeness of observations at the instant of their arrival to the database at record's level. The rule engine imposes the following policies:

- If $(t_{dc}/t_{Ths}) > 0$ and $(t_{dc}/t_{Ths}) \leq 1$, then no temporal consistency issue is

detected.

- If $(t_{dc}/t_{Ths}) > 1$ and $(t_{dc}/t_{Ths}) \leq 2$, then that observation is delayed beyond the regular offset coefficient error interval, indicating a timeliness data quality issue.
- If $(t_{dc}/t_{Ths}) > 2$ and $(t_{dc}/t_{Ths}) \leq 3$ then a single observation is missing, which indicates a data completeness issue.
- If $(t_{dc}/t_{Ths}) > 3$, that indicates two missing sequential observations or a long-outlier, which typically associated with hardware or communication failure and will be covered by the offline-mode of the data quality assessment unit.
- If $(t_{dc}/t_{Ths}) = 0$, that observation with its associated timestamp is duplicated, which may also (but not necessarily) indicates a hardware or communication failure.

The empirical evaluation of this approach is demonstrated in Chapter 4, Section 4.2.3.

3.8 Offline Mode - Time-series Clustering

The data quality management system composes of many different data quality assessment models. Each one of these models is designed to address a particular data quality challenge. The predictive analysis model, Section 3.5, is designed to detect accuracy data quality issues associated with sensor nodes' measurement errors based on their observations temporal correlation with earlier observations. Predictive analysis can distinguish "point outliers" or observations that significantly deviate from earlier observations for a relatively short interval of time.

Point outliers, also known as the contextual or short simple outliers, are sudden and abnormal errors that occur and disappear in a short interval of time (Zhuang

& Chen, 2006). A point outlier is an extreme observation that significantly deviates from their expected value and from the context of their immediate history (Aggarwal, 2015, p. 481-482). Therefore, the predictive analysis approach was adopted to detect point outliers, as detailed in Section 3.5. Examples of point (short) outliers are shown in Figure 3.31, as follows:

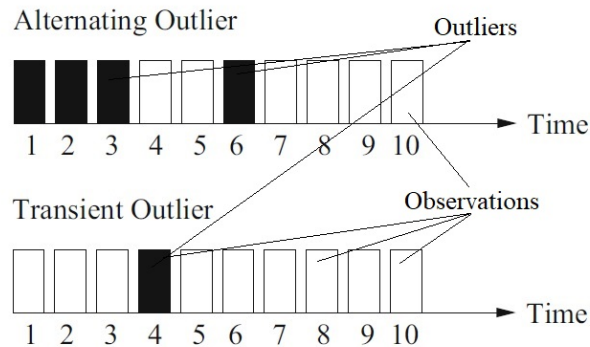


Figure 3.31: The different categories of the point (short) outliers, (SURI et al., 2019, p. 180).

- Alternating outliers are temporal outliers that occur in intermittent (occasional) points in time.
- Transient outlier, which occurs only once in the time-series (SURI et al., 2019, p. 179-180).

The second type of outliers is the **long segmental outliers**, also known as the shape outliers, are irregular observations that emerge for a relatively long time (Zhuang & Chen, 2006; Aggarwal, 2013, p. 189). Long segmental outliers occur in particular cases where a phenomenon has a long-term impact, such as forest fires or oil spills or due to sensor nodes failure (Ghorbel et al., 2015). According to Sailhan et al. (2010)(p. 6), long segmental outliers associated with sensor nodes failures are mainly categorised into:

Continuous halting faults: long outliers that show no or minimal variation in the

value attributes of their data stream for a relatively long interval of time, as shown in Figure 3.32.

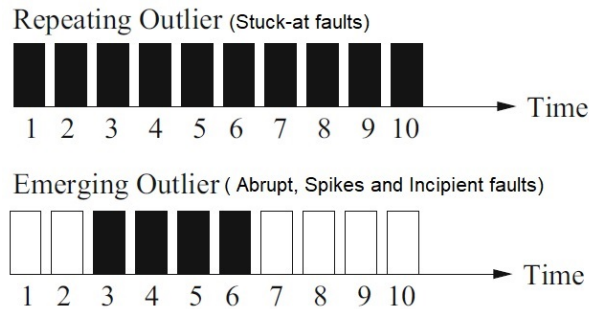


Figure 3.32: Different categories of temporal long outliers, (SURI et al., 2019, p. 180).

Abrupt (emerging) and incipient faults: a constant or linear increase offset to the measurement value that occurs over a longer interval than expected. Long segmental outliers last for a relatively long time and change the pattern of sensor nodes time-series (set of observations) (Aggarwal, 2015, p. 481-482). Thus, long segmental outliers break the temporal correlation of observations after and before their emergence and violate the possibility of using predictive analysis detection techniques to detect this type of anomalies (Berk, 2015, p. 25-27). To tackle this challenge, time-series similarity measures techniques were investigated in this research as a sensor nodes fault detection mechanism based on detecting long segmental outliers in the examined sensor nodes' time-series. Time-series similarity measures define outliers on time series's windows based on comparing them with other non-overlapping windows using a measurement metric such as Euclidean distance to measures the distance between different time series (Aggarwal, 2015; Aghabozorgi et al., 2015; Dean, 2014, p. 153). Therefore, time-series similarity measures were utilised in time-series clustering methods to compare the pattern of an entire or a substantial window of a time series with another based on their long-term temporal correlation (Dean, 2014; Aggarwal, 2017, p. 293). All of the time-series clustering techniques used in this research were experimented to detect

continuous (halting), abrupt (emerging) and incipient sensor nodes faults using real-world datasets, as follows:

The large-scale dataset consists of more than 200 time-series collected from real-world sensors distributed around London. This dataset will be utilised to test the ability of time-series clustering techniques to detect continuous (halting), and abrupt (emerging) long-outliers. Thus these types of long-outliers were detected in some time-series of the large-scale datasets, as detailed in Chapter 4, Section 4.1.2.2.

The local-network dataset consists of four time-series collected from real-world, high-quality sensor nodes deployed at the university of east London. One of the sensor nodes was installed indoor and the other three outdoor. This dataset was used to test the ability of the time-series clustering techniques to detect incipient faults with consistent offset long-outlier. Thus the indoor sensor node, in this case, represented a sensor with incipient fault. The indoor sensor streamed a time-series that is identical in its pattern with other three-time-series from the outdoor sensors but with a consistent offset of 10-15 C^o, as shown in Figure 3.23.

The purpose of time-series clustering is to identify faulty sensor nodes by comparing the shape or features of their time-series with time-series of other properly functioning sensor nodes. Furthermore, if the time-series clustering model detects multi-sensor nodes failures in the same network simultaneously, in that case, that usually indicates a power failure or network breakdown or other technical issues that have a mass impact on the sensor node network. In this research, Dynamic-Time Wrapping (DTW) time-series clustering technique was tested as an anomaly detection mechanism. DTW test was extended to include K-Shape and Characteristic-Based Clustering techniques in an attempt to find a higher performance clustering technique that can render accurate results while examining shorter time-series. The empirical implementation details of this approach are detailed in Chapter 4, Section 4.3.1.

3.8.1 Dynamic Time Warping

Dynamic time warping (DTW) is a time series clustering algorithm utilised to find corresponding regions of similarity between time-series. DTW can stretch or shrink (warp) time series non-linearly along its time axis to find the optimal correlation between different time series (Salvador & Chan, 2007), as shown in Figure 3.33.

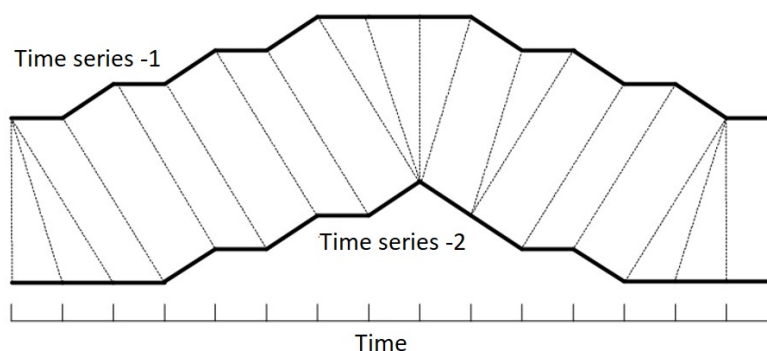


Figure 3.33: An illustration of how DTW warps one time series to another (Salvador & Chan, 2007).

DTW has many implementations in different disciplines, such as gesture recognition, robotics, manufacturing. However, it was mainly used for data mining as a distance measure between data points of time-series (Salvador & Chan, 2007). DTW is a shape-based time series clustering technique. Its algorithm computes the warping path distance between time series. DTW is not sensitive to time-shifting, and it does not require the time series to be on the same length as a condition to compare among them. For example, to compare time series T_1 , T_2 of length n and m , then DTW is going to measure the distance (T_1, T_2) with time complexity of $(n * m)$. Thus, DTW is a computationally expensive method for clustering long time-series or a large number of time-series simultaneously (Aggarwal & Reddy, 2014, p. 367-369).

3.8.2 K-Shape

K-Shape is a time-series clustering algorithm that uses cross-correlation measures to determine both the distance measure and the centroids for time-series clusters. K-Shape analyses the shape of the time series while clustering them. The theory behind K-Shape is similar to the one used by the K-means clustering algorithm. Both rely on the iterative refinement procedure, which scales linearly and produce equivalent and sufficiently separated clusters.

Comparing K-Shape to Dynamic Time Warping (DTW), both are shape-based time series clustering methods which considers the shape similarity between time series regardless of differences in amplitude and phase. Unlike the DTW method, K-Shape is a highly efficient and more domain-independent time series clustering method. K-Shape relies on time series cross-correlation measures, which are significantly faster than the time series distance measures method adopted by DTW (Paparrizos & Gravano, 2016).

3.8.3 Characteristic-Based Time-Series Clustering

Characteristic-based time-series clustering, also known as features extraction-based or statistical characteristics-based time series clustering. Unlike the shape-based time series clustering methods such as DTW or K-Shape, the characteristic-based clustering does not use the distance measure or the cross-correlation measures methods. Alternatively, it clusters time-series based on their captured global characteristics using classical statistical methods.

The set of features extracted from each time series can be fitted into any arbitrary clustering algorithm. The extracted features describe the statistical characteristics (global measures) of the time series, which can be extended to extract more than 100 different features, such as the absolute sum of changes, autocorrelation,

standard deviation and partial autocorrelation. The characteristic-based clustering reduces the dimensions of time series which makes it much less sensitive to the effect of missing values or noisy data. The advantage of the characteristic-based clustering is its high performance even if used to perform similarity searches or clustering amongst very long time series (Wang et al., 2006), as empirically shown in Chapter 4, Section 4.3.1.3.

3.9 Offline Mode – Timestamp Analysis (Spatial Attributes Consistency)

Timestamp analysis was investigated to test the possibility of utilising timestamps of sensor node observations in order to detect mismatches in the spatial contextual attributes of sensor nodes through the following scenarios:

- Detecting spatial mismatches in the contextual attributes of sensor nodes' observations. If a sensor node shows a significant deviation in its geographical location compared to the location of the other sensor nodes connected to the same local network (gateway module) than the coordinates of that sensor are potentially inaccurate.
- Detecting gateways or network failures. If all sensor nodes which are connected to the same gateway stop streaming observations at the same point in time that indicates a gateway or network failure.

This model utilises spatial and temporal analysis mechanism to identify mismatches in sensor nodes, geographical contextual attributes. This approach can also be utilised to identify network blackouts or gateway module failures. The assumption behind this approach is that; it is possible to identify sensor nodes which are connected to the same gateway module based on the spatial and temporal

attributes associated with their observations.

Considering a sensor node network topology like the one shown in Figure 3.34, as a case study. It consists of a group of sensor nodes ($S_{1_t}, S_{2_t}, S_{3_t}, S_{4_t}$) which stream observations via an analogue means using wired or wireless networks to be delivered to the gateway module g_{1_t} . The distance between the sensor nodes and the gateway device cannot be significant, because sensor nodes usually have a limited transmission range, often about 50 to 200 meters maximum.

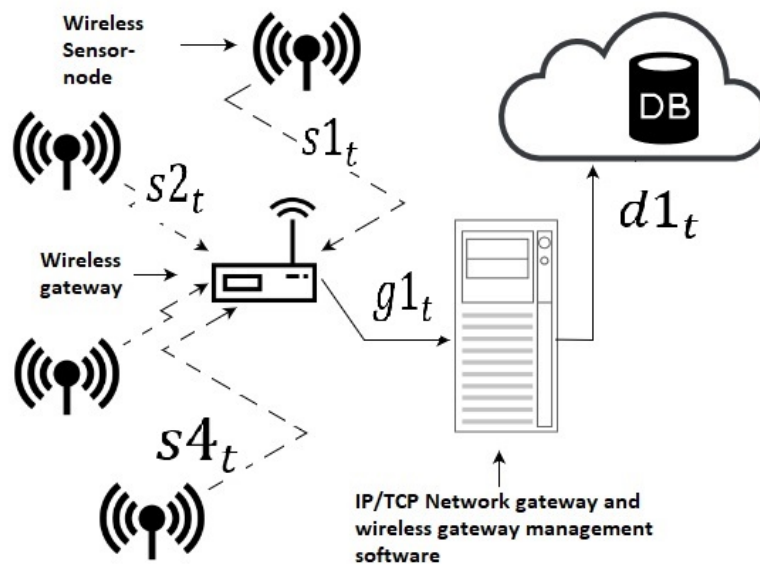


Figure 3.34: An example of a basic sensor node network topology.

Depending on its duty-cycle, each sensor node, typically, sends an observation every S_t minutes to its local network via the gateway module. Sensor nodes duty-cycles are not necessarily even in all sensor in the network, and they are not synchronised so that sensor nodes do not stream their observations at the same time. Sensor nodes utilise a routing protocol to: control when each sensor sends its observation, to optimise the use of the network bandwidth, and to enhance their power efficiency. As listed in Table 3.2, sensor nodes in large-scale CPSs stream observations with essential attributes including their identification ID, observation value and their associated timestamp and location coordinates. The gateway module receives sensors' observations, accumulates them, adds its identification

number and timestamp, and sends them later to the database according to a pre-set duty-cycle, as shown in Figure 3.35. The database receives the data stream from the gateway via the network and stores the observations sequentially, each observation in a new record. Typically, each record gets a unique ID ("row id") number and a new database timestamp which presents the database time at the observation arrival in the database (acquisition time), as shown in Figure 3.35.

However, this is the ideal situation as typically, real-world observations do not show any details related to their gateways or local networks. To determine whether a group of sensor nodes are connected to the same gateway module without the availability of the gateway details, timestamp analysis was investigated in this research. The assumption behind the timestamp analysis approach is that: *if a group of sensor nodes repeatedly exhibit the same database timestamps at a time t and exhibits the same gateway duty-cycle or its greatest common divisor, then these sensor nodes are probably connected to the same gateway.* This assumption was tested using the timestamp analysis model based on the following procedure:

1. To calculate sensor nodes' duty-cycle based on the following equation:

$$S_{dc} = x(y_{s_t} - y_{s_{t-1}}) \quad (3.13)$$

Where S_{dc} is the duty-cycle of sensor node S measured in seconds. x is the integer multiplication of the sensor node duty-cycle, y_{s_t} is the timestamp of the observation y at time t , and $y_{s_{t-1}}$ is the timestamp of the previous observation y at time $t - 1$. The values of observations are not relevant in this context. Sensor nodes' duty-cycles at the database's end are not constant. They may vary according to the gateway duty-cycle or other latency parameters, as shown in Figure 3.36.

2. To calculate the gateway duty-cycle, based on subtracting the observations

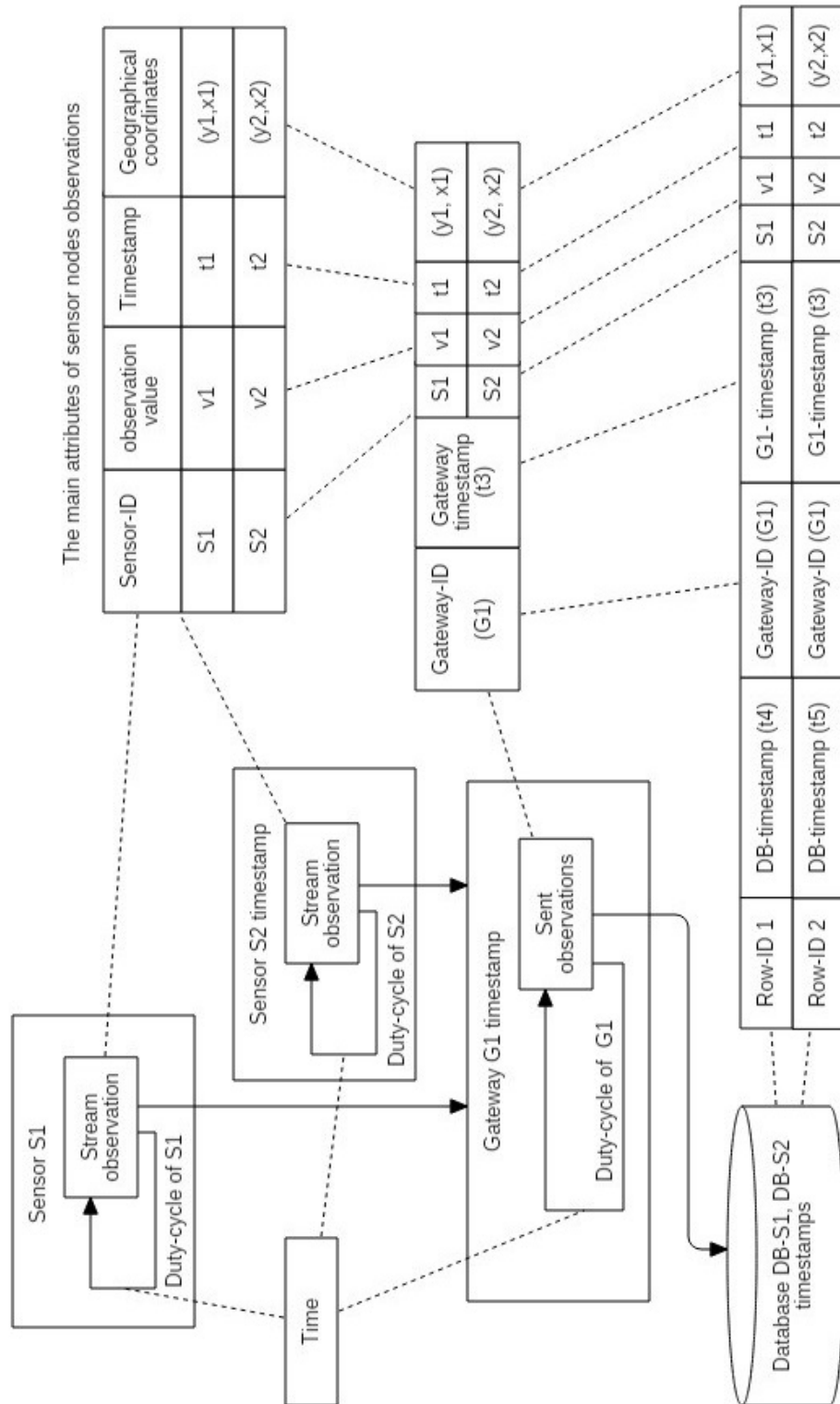


Figure 3.35: Each sensor node observation may hold up to three different timestamps added from the network components.

timestamp $y d_t$ in the database from the sensor nodes observation timestamp

y_{st} .

$$G_{dc} = \kappa(yd_t - y_{st}) \quad (3.14)$$

The gateway duty-cycle G_{dc} at the database's end is not constant, and this may vary according to the sensor nodes duty-cycles and the observations delays, as shown in Figure 3.36.

3. To calculate the exact gateway duty-cycle by using the Greatest Common Divisor (GSD) factor to rank the gateway duty-cycles for each sensor node and choose the one with the highest probability. The same procedure is used to determine the sensor node duty-cycle, as shown in Figure 3.36.
4. Sensor nodes which have coordinated database timestamps and have the same gateway duty-cycle are highly likely to be connected to the same gateway module or local network.

The proposed timestamp analysis model is empirically evaluated using time-series collected from the local sensor node network of the University of East London as a benchmark dataset and using the large-scale time-series collected from the real-world sensor node network distributed around London. More details about the implementation of this timestamp analysis model are provided in chapter-4, Section 4.3.2.

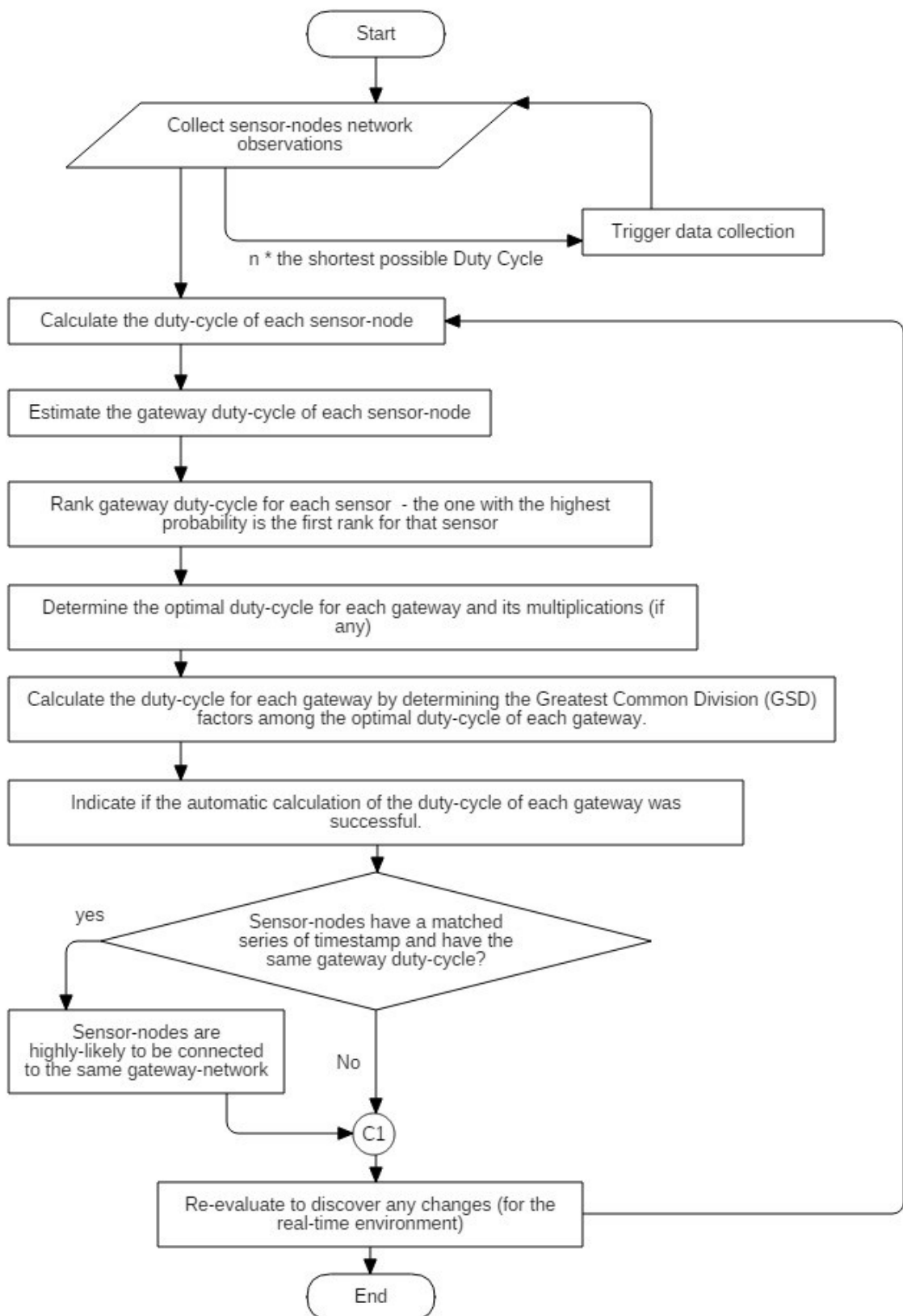


Figure 3.36: The main processes of the timestamp analysis model.

3.10 Summary

In this chapter, the research context was outlined including the structure and design of the data quality management system, data analysis methods, the techniques used to address the research objectives and the adopted logical sequence to conduct the research activities. The next chapter presents the implementation details and results of testing and evaluating the different components and models of the data quality management systems.

Chapter 4

Implementation and Results

“..there is often great virtue in bringing into the open the kind of assumptions that lead to useful methods. The need for robust methods seems to be intimately mixed up with the need for simple models.”

— (Box, 1979, p. 15)

This chapter shows the empirical findings and results from testing and evaluating the different components of the proposed data quality management system which were facilitated in Chapter-3.

This chapter fulfils the third and fourth objectives of the research related to the construction of the proof of concept, data quality management system, and evaluate its validity and performance using real-world large-scale sensor node network as a case study. This chapter comprises three sections; the first section is to show the practical aspects of the data acquisition unit. The second section is to present the empirical results of evaluating the different models of the online-mode data quality assessment unit, which consists of predictive analysis, anomaly analysis, and timestamp analysis models. These models are responsible for detecting data

quality issues associated with errors in sensor nodes measurements in real-time. The third section presents the results of evaluating the components of the offline unit which detects sensor nodes' hardware failures and detects spatial contextual attributes mismatches of sensor nodes' observations using time-series clustering and timestamp analysis respectively. Figure 3.11 shows the structure of the data quality assessment unit and the role of each of its components.

4.1 Data Acquisition and Data Process

This section presents the technical details related to the data acquisition unit focusing on the characteristics of the two sensor node networks used as data sources to investigate real-world data quality issues in large-scale CPSs and to validate the different models of the proposed data quality management system. The structure and design of the data acquisition unit were discussed in Chapter-3 the methodology, section 3.4.2. The data acquisition unit collected observations from sensor node networks in real-time. It consists of the three components: sensor node networks, data streams, and the software framework.

4.1.1 Sensor Node Networks

As mentioned in the previous section, real-world observations were used in all the tests conducted in this research. Two data sources were utilised for this purpose: a local sensor node network which was deployed at the University of East London and, a large-scale sensor node network distributed around London. Both networks are environmental monitoring sensor node networks, which collect ambient temperature observations, as follows:

4.1.1.1 The Large-Scale Sensor Node Network

The large-scale sensor node network is the primary data source of this research. It consists of over 200 temperature sensor nodes distributed around London and managed by different providers such as the Met Office¹, Open Weather Map² and Smart Citizen³. The geographical distribution of these sensors is shown in Figure 4.1.

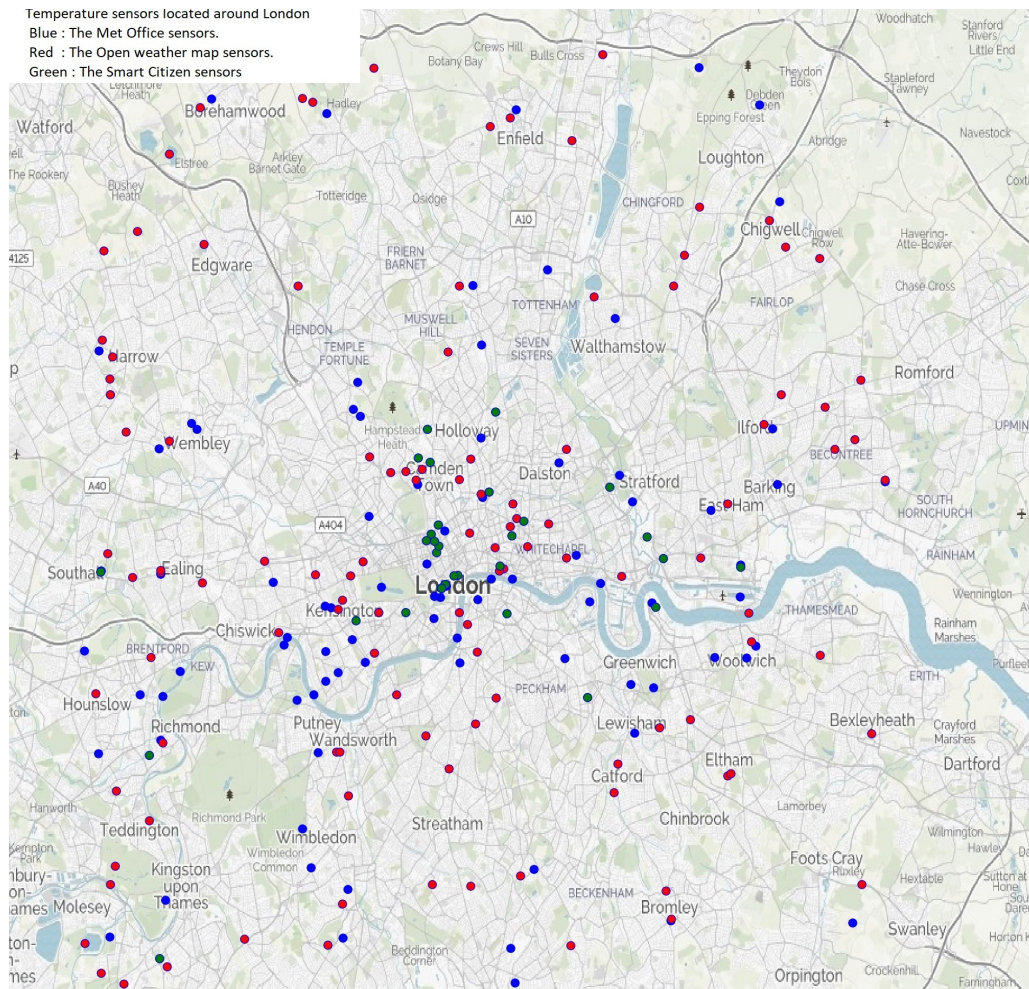


Figure 4.1: The geographical distribution of the real-world sensor nodes network used as a case study in this research, the Met Office (blue), Open Weather Map (red) and Smart Citizen (green).

Data streams from the large-scale sensor node network were coordinated by an

¹<https://www.metoffice.gov.uk>

²<https://openweathermap.org>

³<https://smartcitizen.me>

Internet of Things (IoT) search engine known as Thingful⁴. Thingful is owned by a U.K based company named Umbrellium.Ltd⁵ which is specialised in IoT projects associated with smart cities, connected vehicles, machine learning and big data analytics. Umbrellium has granted special access to its networks of sensor nodes data stream as a kind of sponsorship to this research and cooperation with the University of East London. Observations from sensor nodes that monitor different environmental phenomena such as weather conditions, air quality, noise and tide levels were collected using the data acquisition unit of the data quality management system. The technical details related to sensor nodes, network modules, gateways were not available. No information was available related to sensor nodes' age, type, calibration or maintenance history. Figure 4.2 shows the estimated topology of the large-scale sensor node network.

Although many environmental parameters were collected at the early stages of conducting this research, temperature sensor nodes were selected as the primary data source to test and evaluate the proposed data quality management system. Thus, temperature sensor nodes were the most available type of sensor, vastly distributed around London, and they were managed by different providers, which is an excellent opportunity to investigate data quality issues of such diverse and large-scale sensor node networks. The data acquisition unit is also responsible for the authentication processes and data management as detailed in the following sections.

4.1.1.2 The Local Sensor Node Network

The data quality assessment unit is based on many statistical and machine learning models. Most of these models utilise advance data mining techniques that involve data models training and testing phases, such as with the LSTM and the GPR

⁴<https://www.thingful.net/>

⁵<https://umbrellium.co.uk/products/thingful/>

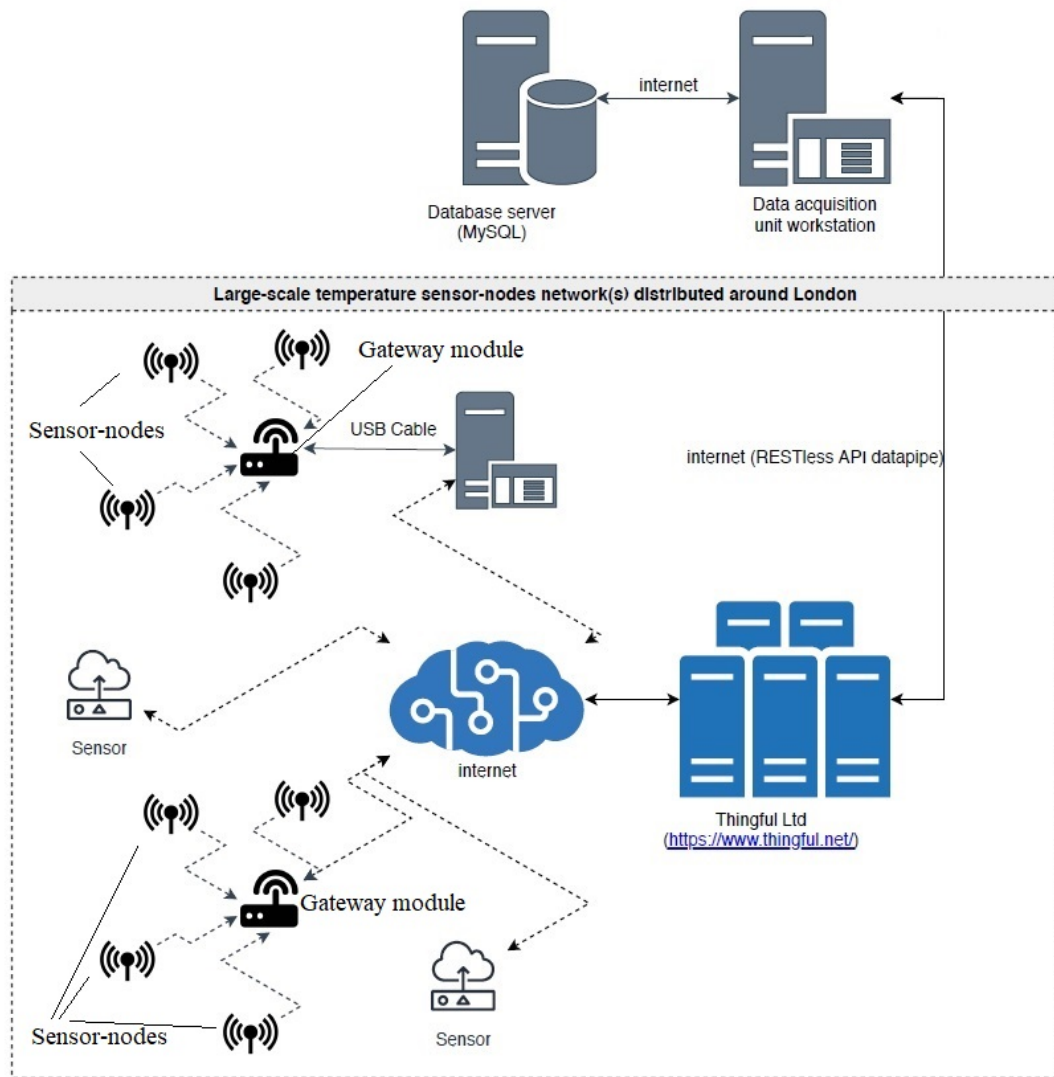


Figure 4.2: The estimated network topology of the large-scale sensor node network.

predictive models. The development process of most of the data quality assessment models involves an iteration phases of testing and calibration processes until these models satisfy a particular level of accuracy or performance criteria, as viewed in Section 3.4. These training and calibration processes require a relatively long sensor nodes' observations time-series with minimal outliers, missing values or noise to avoid any biased or misleading performance by the developed data quality assessment model which may occur because of the influence of the data quality issues on the accuracy or the performance of trained models. Since there was no practical means to ensure that the observations collected from the real-

world, large-scale sensor node network are free from outliers, noise or missing observations, a high-quality temperature sensor node network was deployed at the University of East London in order to provide high-quality time-series to train and calibrate the data quality assessment models. The local sensor node network works as a benchmark to the large-scale sensor node network which was needed to produce long, consistent, high-quality data streams of observations to train and adjust the different data quality assessment models and to evaluate the performance and accuracy of these models before utilising them in real-world scenarios⁶.

Figure 4.3 shows the deployment map of the local sensor node network at the University of East London.

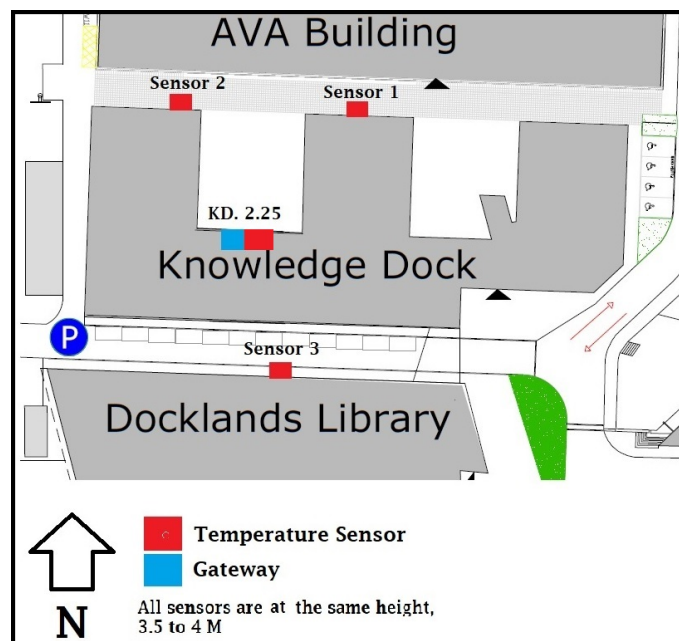


Figure 4.3: The deployment map of the local sensor node network at the University of East London.

The network topology of the local sensor node network was explicitly chosen to match the main structure of the large-scale sensor node network, where both networks collect observations from sensor nodes or gateway models to a remote

⁶It is possible to consider each test based on the observations from the local sensor node network as an experiment, not a case study because it was conducted using observations collected from a fully controlled environment.

cloud-based solution owned by Umbrellium in the large-scale network and Imonnit in the local network. Both networks stream observations to a pre-defined destination (UEL MySQL server in this case) based on API requests by the data acquisition unit and using JSON format. Both networks utilise the same security protocols and authentication processes. The topology of the local sensor node network is shown in Figure 4.4, and the full network topology is shown in Appendix A, Figure A.3.

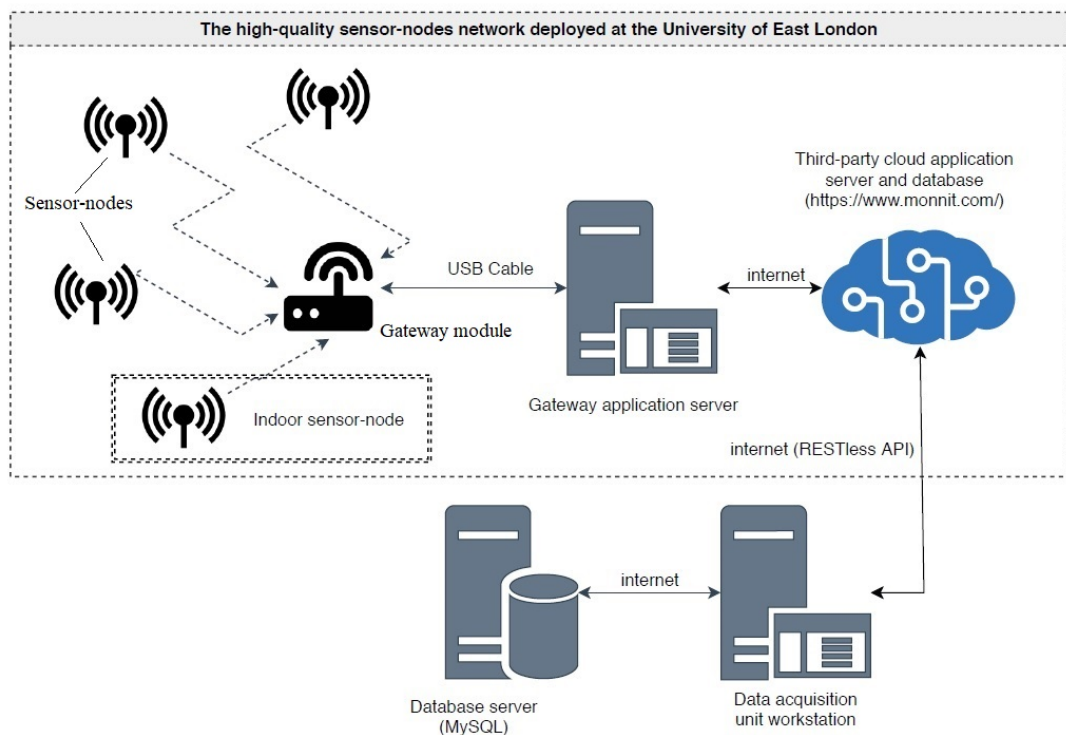


Figure 4.4: The topology of the local sensor node network.

The aim was to involve the same type of modules and processes in the local network in order to experience the same type of latency and possibly the same data quality issues that may occur in the large-scale network. The sensor node network consists of four high-quality wireless temperature sensor nodes, all connected to the same wireless gateway module. Three sensor nodes were deployed outdoors on the same distance from the gateway (70 m) and at the same height (4m), the fourth sensor node was deployed indoors to simulate a faulty sensor which

streams off-set observations. The local sensor nodes and the wireless gateway⁷ were outsourced with a complete software solution which covers hardware management and facilitates streaming the observations to the cloud of the provider company, Monnit Corporation⁸. Sensor nodes' observations were collected from Monnit cloud using the data acquisition unit applying the same steps used to collect the observations from the large-scale sensor node network through Thingful.Ltd. The technical specifications of the sensor nodes and gateway module of the local network are detailed in Appendix A, Sections A.2.

4.1.2 Datasets

All components of the proposed data quality management system were tested using real-world datasets collected from two different ambient temperature sensor node networks. The ideal (baseline) dataset collected from the local sensor node network deployed at the University of East London and the real-world (standard) dataset collected from the large-scale sensor node network distributed around London.

4.1.2.1 The Ideal Dataset

The ideal dataset was collected from the local sensor node network. The local sensor node network consists of four high-quality wireless temperature sensor nodes and a wireless Gateway. Each sensor node streams a single observation every 10 minutes. The Gateway receives and holds the observations and sends them to the database server in one push every 10 minutes. Three of these sensor nodes were deployed outdoors around the campus of the University of East London, while the fourth sensor node was deployed indoors inside the Knowledge

⁷<https://wireless-sensors.co.uk/>

⁸<https://www.monnit.com/>

Dock building, Figure 4.3. The ideal dataset consists of four time-series, as shown in the snapshot dataset example starting from 26/09/2019 8:00 a.m to 02/10/2019 23:50:00, in Table 4.1.

Table 4.1: The four time-series of the ideal dataset (a snapshot).

Time series (Sensor ID)	No. of Observations	location
493361	1008	Indoor
493367	1008	Outdoor
493368	1011	Outdoor
493372	1006	Outdoor
Total No. of Observations:	4033	

The main attributes of the ideal dataset are shown in Table 4.2. The ideal dataset is a high-quality, consistent time-series of four ambient temperature observations with no missing values or outliers. It was used as a benchmark (or as a baseline) dataset to evaluate and calibrate the different statistical, machine learning and time-series clustering techniques before applying them to the real-world observations of the large-scale sensor node network.

Table 4.2: The main attributes of the ideal dataset.

SensorID	MessageDate	Values (C°)	Battery (Volt)	Gateway ID	Signal Strength
493361	11/10/2019 22:13	24.8	100	936486	100
493361	11/10/2019 22:23	24.9	100	936486	100
493361	11/10/2019 22:33	24.8	100	936486	100
493361	11/10/2019 22:43	24.7	100	936486	100

All sensor nodes of the local network were deployed in a relatively small geographical area. Therefore, these time-series have high similarity in the trend's shape but may have some differences in the value attribute. This feature is of significant importance in testing shape-based time-series clustering methods, as detailed in Section 4.3.1.

4.1.2.2 The Real-World Dataset

The real-world dataset was collected from over 200 different sensor nodes, distributed around London and owned by many providers, all coordinated by Umbrellium.Ltd. These sensor nodes are geographical distributed as shown in Figure 4.1. The key attributes of the real-world dataset are shown in Figure 4.5.

SensorID	Latitude	Longitude	MessageDate	Ambient	License_name	Provider_uri	Provider_name
47qwbfbfa	51.65736	-0.21423	2019-01-20 14:27:28	4.49	Attribution-Sh...	http://openwe...	OpenWeathe...
47qwbfbfa	51.65736	-0.21423	2019-01-20 13:39:47	4.27	Attribution-Sh...	http://openwe...	OpenWeathe...
47qwbfbfa	51.65736	-0.21423	2019-01-20 13:14:55	4.4	Attribution-Sh...	http://openwe...	OpenWeathe...
47qwbfbfa	51.65736	-0.21423	2019-01-20 12:51:52	3.81	Attribution-Sh...	http://openwe...	OpenWeathe...
47qwbfbfa	51.65736	-0.21423	2019-01-20 12:14:31	3.79	Attribution-Sh...	http://openwe...	OpenWeathe...
47qwbfbfa	51.65736	-0.21423	2019-01-20 12:02:17	3.09	Attribution-Sh...	http://openwe...	OpenWeathe...
47qwbfbfa	51.65736	-0.21423	2019-01-20 11:14:37	2.07	Attribution-Sh...	http://openwe...	OpenWeathe...

Figure 4.5: The key attributes of the real-world, large-scale sensor nodes dataset.

The real-world dataset comprises observations collected from sensor nodes deployed in different geographical locations to monitor the ambient temperature from around London. In such systems, data quality issues may occur because of many reasons such as sensor nodes hardware failures, unreliable communication, external inference and noise. Thus, the real-world dataset typically involves many data quality issues. The accuracy and performance of all the data analysing models used in this research were evaluated by investigating their ability to detect data-quality issues associated with errors in sensor nodes measurements, hardware failures in sensor nodes, and mismatches in sensor nodes' spatial and temporal contextual attributes. Inspecting the real-world dataset using data visualisation and aggregation methods broadly revealed the presence of the following data quality issues:

4.1.2.2.1 Inaccurate Observations and Long-Outliers

Temperature time-series typically exhibits a daily seasonality and a trend, as shown in Figure 3.12. Time series that show a constant value attribute or a very

low seasonality over a relatively long-time are highly likely to encompass data quality issues related to the accuracy of the observations forming long-outliers. For example, applying aggregation functions⁹ on a one-week interval of sensor nodes' time-series revealed that all the Smart Citizen and six of the Mat Office sensors did not show any variations in their observations' value (trend), as shown in Figure 4.6.

Provider_name	Range_of_change_in_temperature	No_of_sensors
Met Office	30 - 4	45
OpenWeatherMap	30 - 4	24
Met Office	4 - 0	31
OpenWeatherMap	4 - 0	92
Met Office	No change	6
SmartCitizen	No change	76

Figure 4.6: Time-series of seven days window of sensor nodes observations aggregated by the variations range in their value attribute.

In this case, the sensor nodes did not stop streaming observations but kept repeating the same value attributes, forming long-outliers. This behaviour is highly likely to be related to sensor nodes hardware failure that affects their detection ability. Alternatively, it may indicate that these sensor nodes are down (power failure), and the system is compensating for their missing observations by repeating the last observation it received from these faulty sensors. Figure 4.7 shows an example of long segmental outliers. Two sensor nodes time-series are showing fixed value attribute for a relatively long time, compared to another time-series generated by a functional sensor node managed by the Met Office during the same time-window.

4.1.2.2.2 Missing or Inconsistent Observations

Examining sensor nodes time-series revealed that real-world time-series exhibits temporal inconsistencies which occur for a relatively short time and deform the seasonality component, as illustrated in the time-series decomposition diagram,

⁹Group by, Max and Min functions using SQL.

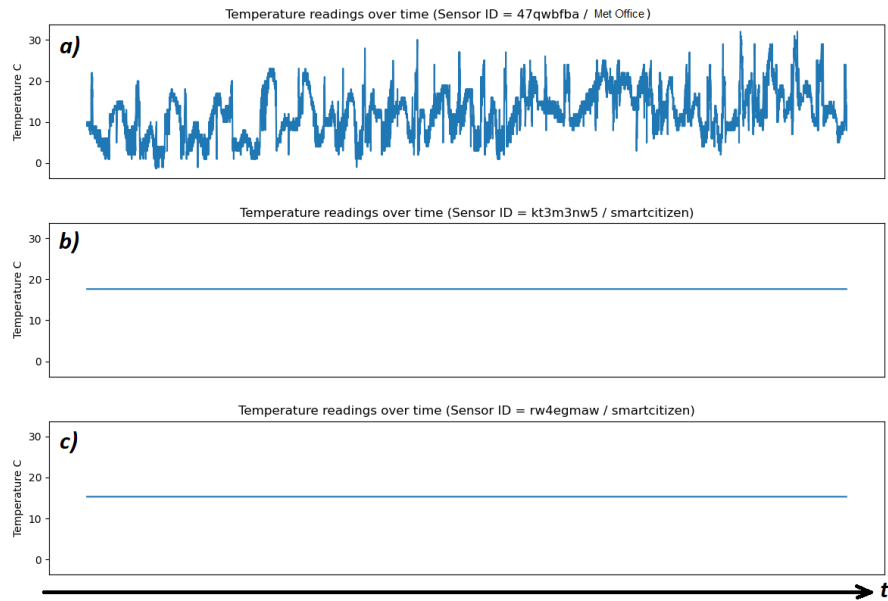


Figure 4.7: An example of two temperature time-series with long segmental outliers (b and c) comparing with a typical time-series collected from a functional sensor node (a).

Figure 4.8. Thus, the pattern of a test part of a time-series is anomalous if its occurrence frequency differs significantly from its expected standard frequency (Suri et al., 2019, p. 45).

Such inconsistency issues may occur due to noise, temporary hardware failures, or short network breakdowns. It may also occur because of hardware misconfiguration, e.g., if a gateway module's duty-cycle is equal or slightly longer than the duty-cycle of a sensor node in the same network. In that case, the gateway will regularly escape sending up to two of that sensor nodes' observations to their destination. Although the required frequency of observations may differ from an application to another, there is a certain threshold after which the data become non-beneficial. For example, in the case of temperature sensor nodes, and assuming that the minimum number of observations per day for each sensor node is four (one every 6 hours), a seven-day time-series must at least consist of 28 observations. Otherwise, it is considered as time-series with consistency data quality issue. For example, Figure 4.9 shows time-series of seven days' time window

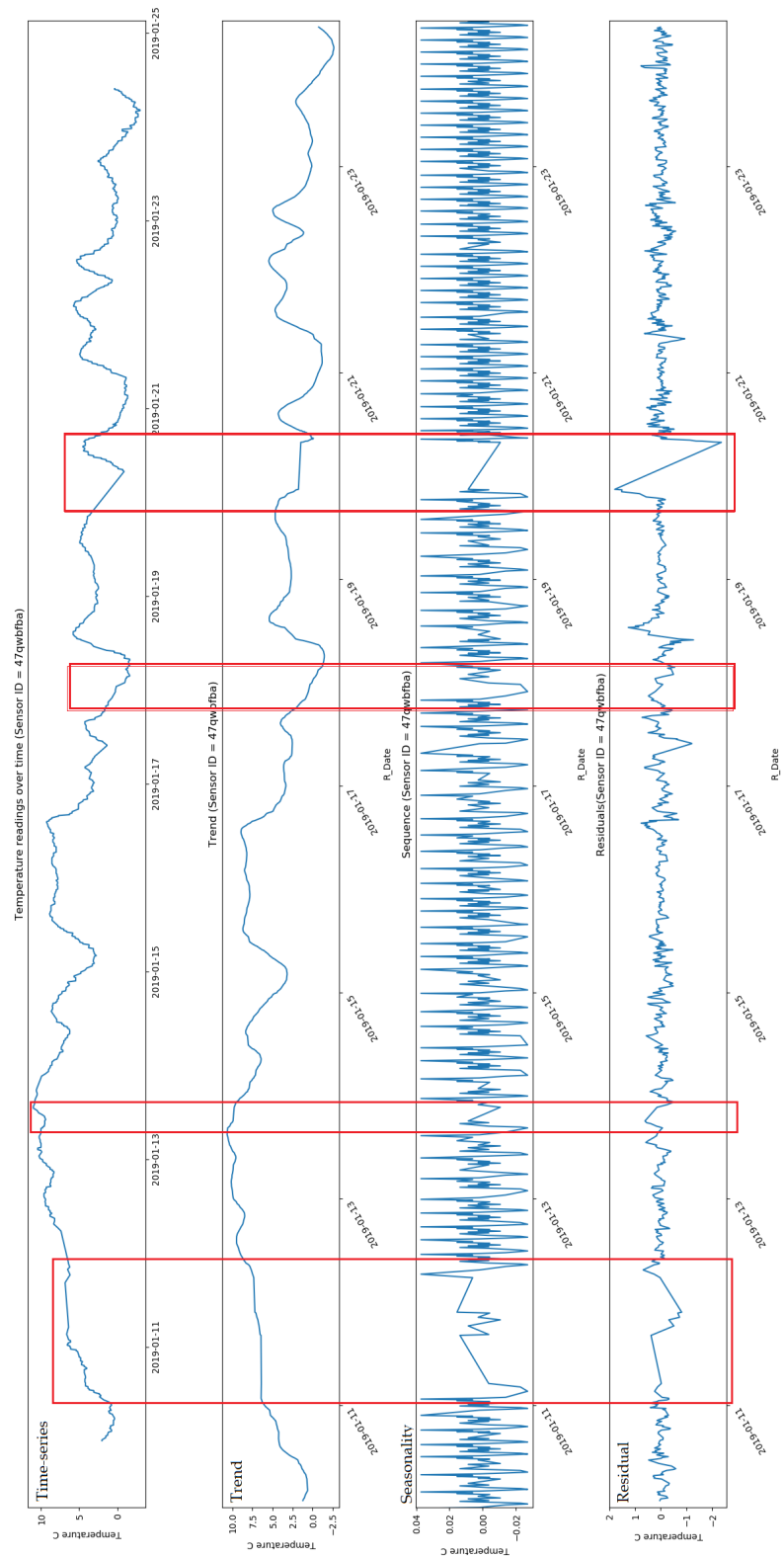


Figure 4.8: Examples of the temporal inconsistencies in time-series of real-world sensor nodes using time-series decomposition, (sensor ID=47qwbfba, 11-23/01/2019, London).

of 274 of the real-world sensor nodes categorised according to the range of the number of observations. Which demonstrates that the time-series of 8 of the Met office and 26 of the Open Weather Map sensor nodes have less than 28 observation between '2018-09-02 00:00:00' to '2018-09-08 32:59:00' revealing a potential data consistency issues.

Provider_name	Range_of_time_series_length	No_of_sensors
Met Office	168-28	74
Met Office	Less 28	8
OpenWeatherMap	168 - 672	82
OpenWeatherMap	168-28	8
OpenWeatherMap	Less 28	26
SmartCitizen	168-28	76

Figure 4.9: Time-series of 274 sensor nodes categorised according to the range of number of observations (seven days window).

4.1.2.2.3 Inaccurate Spatial Attributes

The geographical location of sensor nodes indicated by the spatial contextual attributes associated with their observations does not necessarily reflect the actual location of where these devices are deployed. Furthermore, there is no mechanism to verify that these attributes reflect the actual geographical location of the sensors. For example, the spatial distribution of some of the Smart Citizen sensor nodes (community sensors) ' showed that these devices are deployed somewhere over or very close to the German Embassy, the Embassy of Austria and the Egyptian Consulate in London, which is highly unlikely, as shown in Figure 4.10. These sensor nodes were probably used as test or as educational projects, and their spatial attributes were set randomly to central London as their estimated location.

4.1.3 Software Framework

The software framework is the third component of the data acquisition unit. It was developed to collect sensor nodes observations in real-time. It consists of software

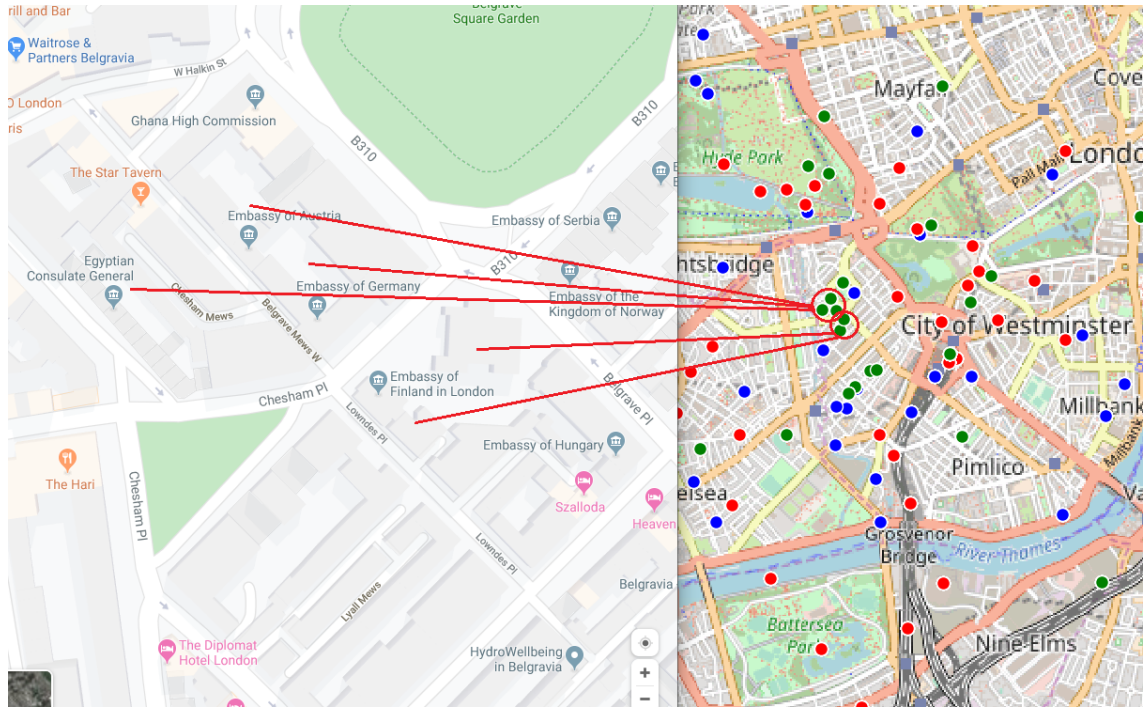


Figure 4.10: An example of sensor nodes that their coordinates do not reflect their correct location.

components and solutions applied to facilitate transactions of observations from remote sensor nodes to the local database at the University of East London. The software framework includes cloud computing modules, My-SQL database, and Java-based data acquisition software specifically developed for this research. The key component of the software framework are:

4.1.3.1 Cloud Computing Modules

As mentioned in the previous section, the data acquisition unit collects sensor nodes' observations from two different sensor node networks, the ideal and the large-scale networks. The access to the observations stream of these networks was provided and managed by two companies: Umbrellium and Monnit through their cloud solutions Thingful¹⁰ and iMonnit¹¹, respectively. In both cases, sensor nodes observations were not streamed directly to the required destination, which

¹⁰<https://www.thingful.net/>

¹¹<https://www.imonnit.com/Account/LogonOV?ReturnUrl=/>

is in this case, the local database at the University of East London. Instead, these observations were streamed from the sensor node networks to the cloud of the solution provider (Umbrellium, Monnit) and then were streamed back, on request, to the local database. The data acquisition unit facilitated the process of requesting the observations from the computing clouds of these companies via continuous recurrent RESTful requests.

As shown in Figure 4.2 and Figure 4.4, both data providers, “Umbrellium” and “Monnit”, were using a communication structure known as RESTful API to stream sensor nodes observations. REST (Representational state transfer) is a simple, high-performance and scalable data transaction architecture which can establish an efficient and secure connection between online data streaming services and any remote application or database. RESTful considered as a data structure, not a communication protocol because it is merely a set of design rules, pre-configured by the data provider to determine what responses their online API will afford (Dong et al., 2009, p. 243). RESTful settings define the structure of the full Uniform Resource Locator (URL) address of the endpoint API of the data provider, which known as the Request URL. The Request URL also includes the security authentications parameters which should be provided in order to approve data transactions to the requesting destination. The structure of the Request URL typically differs from data provider to another, depending on applications requirements and their security authentication procedures.

4.1.3.2 Java-Based Data Acquisition Software

The data acquisition software is the core component of the data acquisition unit. It coordinates all necessary processes including facilitating security authentications, requesting observations from Thingful and iMonnit, processing and saving observations in the local database at the University of East London. The data acquisition software was developed using Java and consists of six main components (classes),

as shown in the class diagram Figure 4.11.

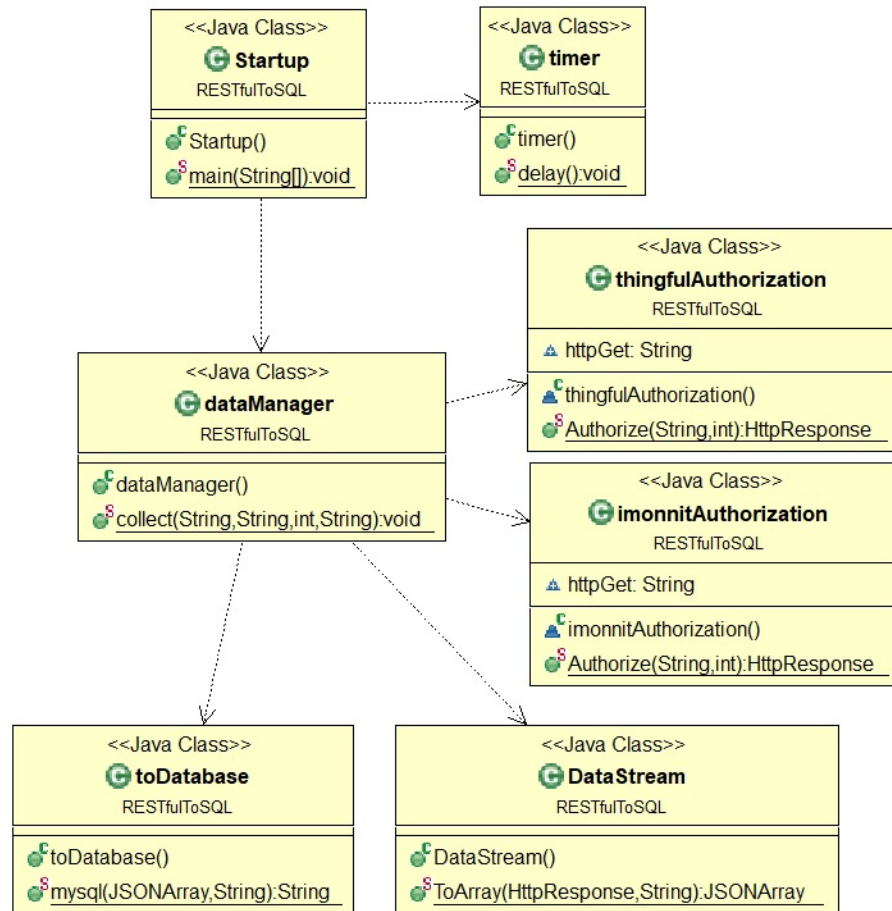


Figure 4.11: The classes and main attributes of the data acquisition software.

Each one of these classes has a specific role and combined these classes describe how the data acquisition software operates, as follows:

"Startup" class: starts the data acquisition process by inquiring the Requests URLs details from the local database and forward the request results to the “dataManager” class, as shown in the sequence diagram, Figure 4.12, steps 2 to 4.

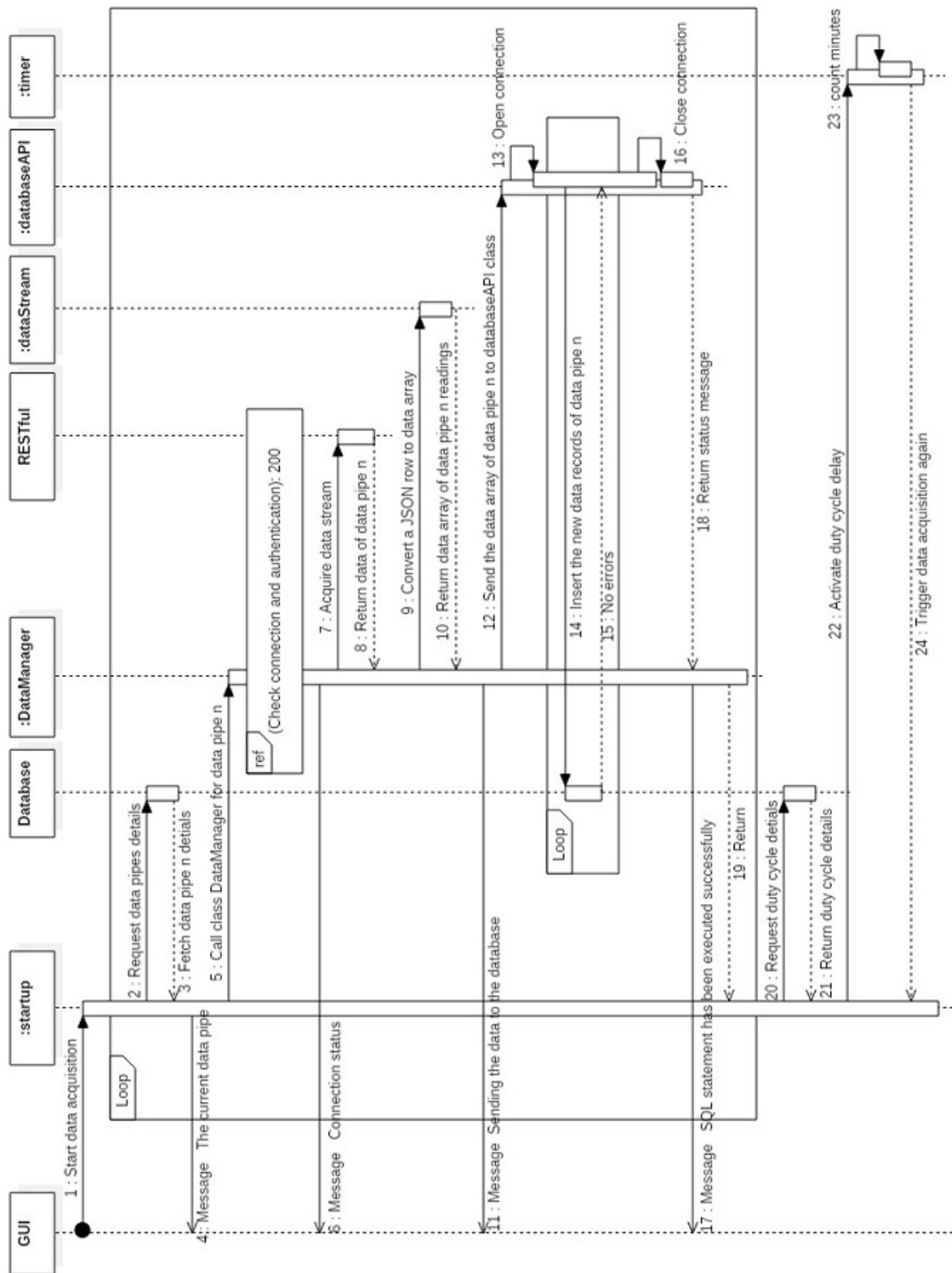


Figure 4.12: The sequence diagram of the data acquisition unit.

Figure 4.13 shows the requested details include the data providers details "Dp_details", authentication keys "Dp_API_token", the number of the observations or the requested time interval "Dp_sample_size". The requested details also include the

titles of the destination tables where the acquired observation will be stored (corresponding tables) "Dp_Table" according to the type of the environmental parameter.

Dp_id	Dp_Table	Dp_API_token	Dp_sample_size	Dp_details	Dp_type
1	th_environment_json	https://datapipes.thi...	331	Environment of London	thingful
2	th_energy_json	https://datapipes.thi...	30	Energy in London	thingful
4	th_Noise_json	https://datapipes.thi...	76	Noise in London	thingful
5	th_Air_quality_json	https://datapipes.thi...	689	Air quality in London	thingful
6	th_Weather_json	https://datapipes.thi...	421	Weather in London	thingful
7	th_imonnit_json	https://www.imonnit...	10	10 = time interval	imonnit

Figure 4.13: Data-pipes details used by the data acquisition software to construct the RESTful Requests URLs.

"dataManager" class: manages the data acquisition process, it sends requests to the authentication classes **"thingfulAuthorization"** and **"imonnitAuthorization"** to create the authentication script based on the details retrieved by the "Startup" class. Both are responsible for creating the RESTful API Request URL. After receiving the Request URL script, the "dataManager" class send the authentications token keys to the RESTful APIs of Thingful and iMonnit and open the connection in order to start requesting data. It connects with the selected RESTful API to make sure that the connection is open, and the security authentications were approved, as illustrated in Figure 4.12, step 5.

The next step is acquiring the observations list, which arrives as a one-piece JavaScript Object Notation (JSON) object. The "dataManager" class sends the JSON object to the **"dataStream"** class which verify that the received object is a valid JSON data object and dissolve it back as a JSON array. Each row in the array is a sensor node observation with all its associated details such as the sensor id, timestamp, geographical coordinates, provider details and license type, but in JSON format. The class returns the JSON array to the "dataManager" class, as shown in Figure 4.12, steps 7 to 10. The "DataManager" class forward the JSON array to the "toDatabase" class which establish a connection with the MySQL database and send the JSON array to it. Each row in the JSON array becomes a new record in the JSON destination table, as initially showed by the details table

in Figure 4.13. At this stage, the **"toDatabase"** class closes its connections with the current RESTful API and with the My-SQL database server and send a request to the "Startup" class to start acquiring data from the next data-pipe, as shown in Figure 4.12 steps 11-19.

After acquiring observations from all available "Requests URLs", the "Startup" class activates the **"timer"** class. The "timer" class halts the open-loop executing procedure for T minutes, where T is one minute less than the shortest sensor nodes duty-cycle in the system, Figure 4.12 steps 20-23. The data acquisition software runs continuously as a built-in service impeded in the operating system. It was deployed on a Linux Fedora 31 workstation. The deployment environment diagram of the data acquisition software and the deployment code are illustrated in Appendix A, Section A.3.

4.1.3.3 MySQL Database and Data Process

After receiving the JSON objects from the data acquisition software, the first data process that takes place inside the database is JSON decoding (parsing). All of the database JSON destination tables are configured to decode the JSON strings into a readable data format and save it in the corresponding observations tables. Each JSON destination table is configured to trigger a JSON parse procedure after receiving new records and send them as new records to the corresponding table, as shown in Figure 4.14.

The second data process is related to preventing duplication's in the observation's tables. This issue must be solved without losing or rejecting any observation from being saved in the database. Duplication's occur systematically in the observations tables due to miss-matching between the duty-cycle of the data acquisition requests and the duty-cycles of some sensor nodes. For example, if the frequency of the data acquisition requests issued by the data acquisition software is one every ten

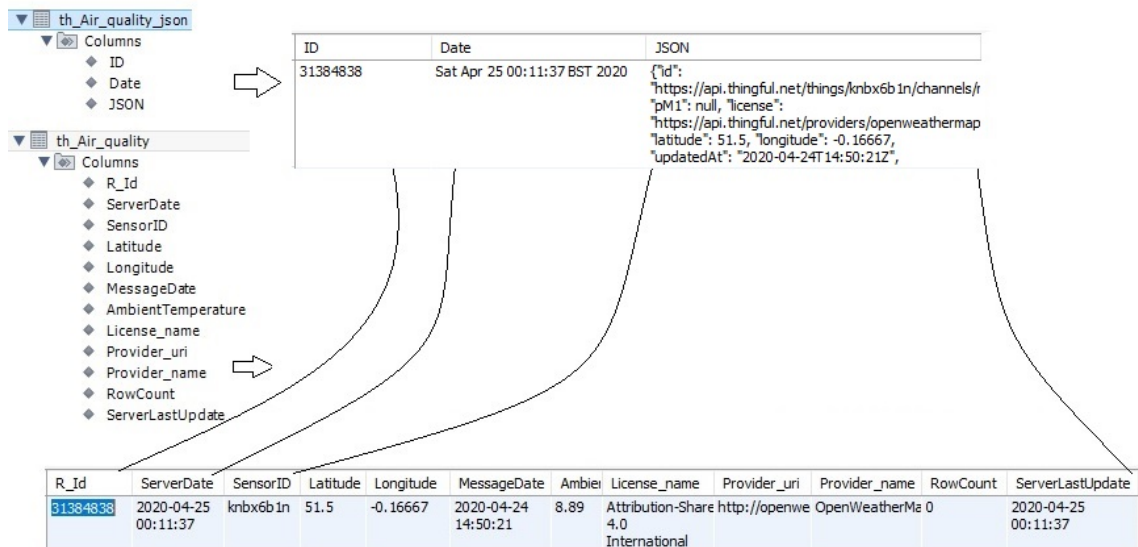


Figure 4.14: The JSON parsing process from the destination table into the corresponding observations table, air quality table as an example.

minutes, in that case, the JSON destination table will receive the same observation six times from the sensor node that has a one-hour duty-cycle.

In order to tackle this issue while keeping all records, no duplication prevention constraints were set in the JSON destination tables. Thus, all JSON tables will always accept all arrived records sent by the data acquisition software, forcing no constraints that may expel some of these observations. However, duplication prevention countermeasures are active in the corresponding tables. Whenever a record arrives, the corresponding table checks if the observation is already exists based on the observation sensor id, timestamp and value. If the corresponding table detects a duplicated observation, it automatically increases the duplication counter of that record by one and updates the server's last observation timestamp. Figure 4.15 shows the process diagram of the duplication prevention mechanism of the data acquisition unit, and Figure 4.14 shows both the **RowCount** and **ServerLastUpdate** attributes. This procedure has provided important indicators which were used later in the timestamp analysis model.

More technical details related to the data acquisition unit are presented in Appendix A, Section A.3.

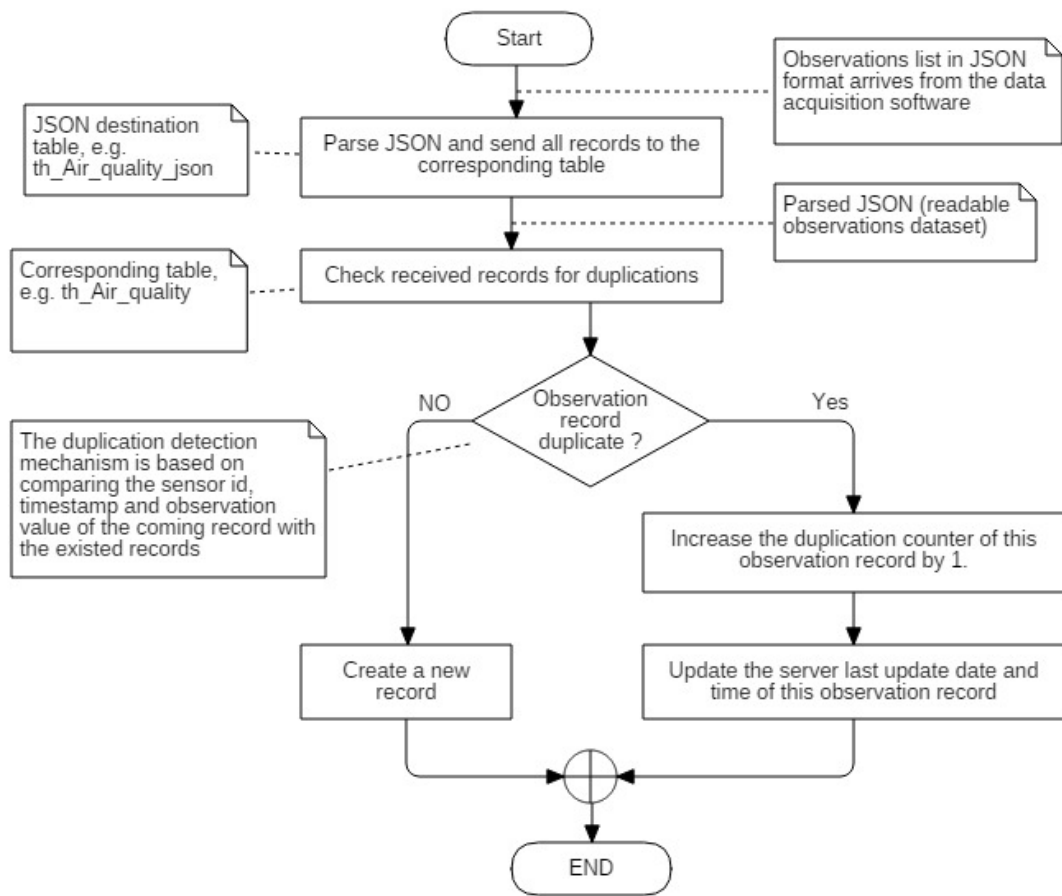


Figure 4.15: The process diagram of the duplication prevention mechanism of the data acquisition unit.

4.2 Online-Mode Data Quality Assessment

As outlined in section 3.4.3, the data quality assessment unit is the core component of the data quality management system. It consists of four main components: predictive analysis, anomaly analysis, time-series clustering and timestamp analysis models, which operate in online and offline modes. Each of these components has a specified role in detecting a particular type of data quality issues in data streams of large-scale CPSs. The focus of this section is on data quality assessment techniques that detect data quality issues associated with errors in sensor nodes measurements in real-time. The real-time notion in this context means that the data

quality assessment must be completed in a shorter interval than the shortest sensor node duty-cycle in the system. In other words, the data quality assessment of a set of observations must finish before receiving the next set of observations from the same sensor nodes for all sensors in the network. Thus, the online mode of the data quality assessment unit represents all components that satisfy the real-time constraints. Predictive and anomaly analysis models were utilised for evaluating the accuracy of observations by detecting short-simple and long-outliers based on their temporal correlation with earlier observations or based on their spatial correlation with other observations from neighbour sensor nodes, respectively. The detection of data quality issues of sensor nodes measurements associated with timeliness, consistency and completeness dimensions was facilitated using timestamp analysis by utilising periodicity analysis and a rule engine to detect temporal irregularity in sensor nodes time-series. The full structure of the online data quality assessment unit is shown in Figure 3.11. Many statistical and machine learning techniques were empirically experimented¹² within the context of real-time predictive and anomaly analysis models, as shown in Figure 4.16.

4.2.1 Predictive Analysis Models

This section outlines the empirical details of testing different predictive analysis techniques for evaluating the accuracy of sensor node measurements via detecting irregularities (short-outliers) in these sensors' observations. Chapter 3, Section 3.5 describes the structure and design of the predictive analysis models tested in this section.

¹²Note: All tests of this case study were conducted using Python 3.7 64 bits installed over a Linux (Fedora 31 64 bits) workstation. The processor of the workstation is an Intel(R) Core (TM) i7-7920HQ CPU @ 3.10GHz (8 CPUs), with 32 GB of RAM and A NVIDIA Quadro M1200 dedicated video card with 4GB DDR5 RAM.

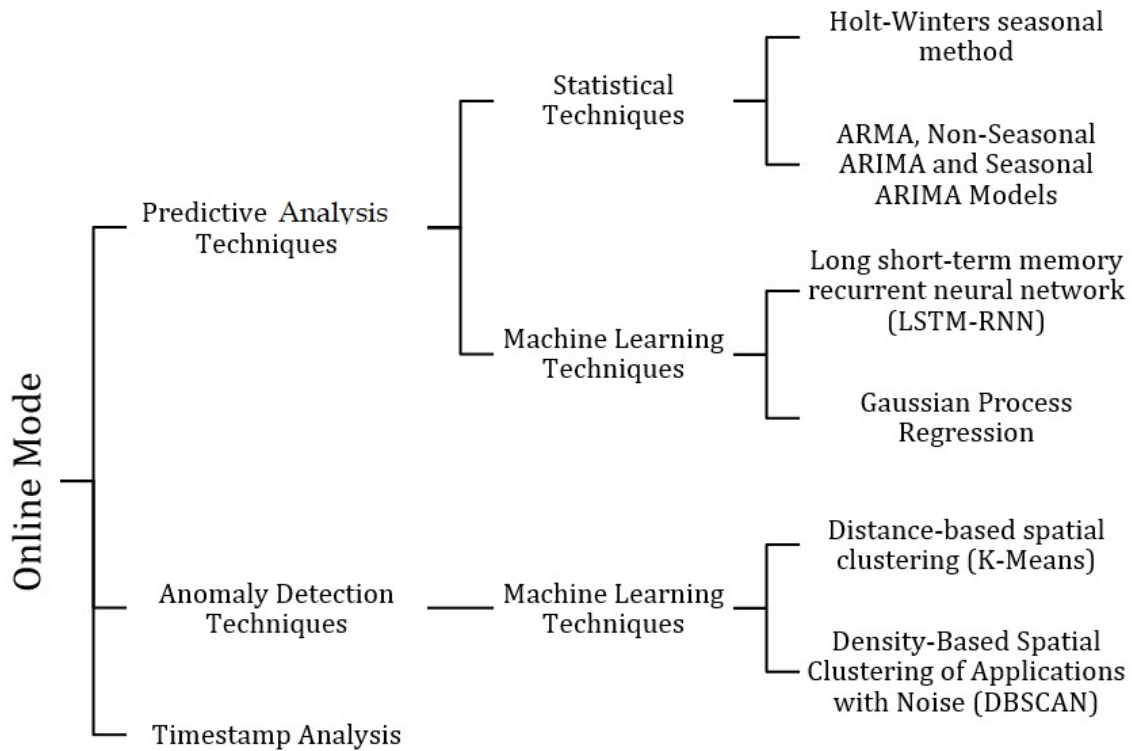


Figure 4.16: The algorithms and techniques empirically tested to validate the accuracy of sensor nodes' observations in real-time.

4.2.1.1 Datasets Modes and Details

As outlined in Chapter-3, Section 3.5, it is possible to examine the accuracy of a sensor node observation by comparing its value attribute to its predicted value using an autoregressive predictive model (Farooqi et al., 2018; Rager et al., 2018). An autoregressive model is a prediction model that regresses a variable using a combination of its previous observations (Lind et al., 2018, p. 380). Here, the autoregressive models were developed using sensor nodes' previous observations (time-series) to evaluate the accuracy of their current observations. To ensure the accuracy of the autoregressive (prediction) models, it is recommended to develop and train these models using a consistent time-series with a minimum level of outliers or missing values. Since there was no practical method to ensure that the collected data from the real-world large-scale sensor node network are outliers, noise free and complete, a benchmark time-series was required to train

and verify the accuracy of the predictions models and to verify the data quality of the observations of the selected (examined) real-world sensor.

A benchmark sensor node was selected randomly from the three outdoors wireless temperature sensor nodes of high-quality sensor node network deployed at the University of East London. The benchmark sensor node was selected randomly from the local network since the time-series of all sensors in the local network showed significant similarity in their pattern (trend) and consistency, as shown in Figure 3.23. Thus, all sensor nodes of the local network were sourced from the same manufacture, have the same configuration, all new with new batteries and deployed over a relatively small geographical location. The time-window of the selected time-series is between 2020/03/01 00:00:00 and 2020/03/15 23:59:59, with a total number of (2157) observations. The time-series decomposition graph of the benchmark time-series is shown in Figure 4.17.

The time-series decomposition graph of the benchmark sensor node (493372) shows a typical temperature trend, which slightly changes depending on the season. It also shows a daily temperature seasonality of $\pm 2C^{\circ}$ degrees. The residual is revealing a relatively high alternation, which is related to the characteristics of the ambient temperature as a natural phenomenon affected by many parameters, such as wind speed, humidity, sunshine density and the location of the sensor nodes (under a tree, beside a river or on a top of a building).

A real-world time-series was selected from the large-scale sensor node network based on two factors: first, it must have the same or the nearest possible sampling rate as the benchmark time-series, six observations per hour, 144 observations per day. Second, it must have minimal data quality issues related to completeness and consistency of observations. The time-window of the time-series of sensor node (jcw5m701) was selected to match the benchmark time-window with a total number of (937) observations. Figure 4.18 shows the time series decomposition

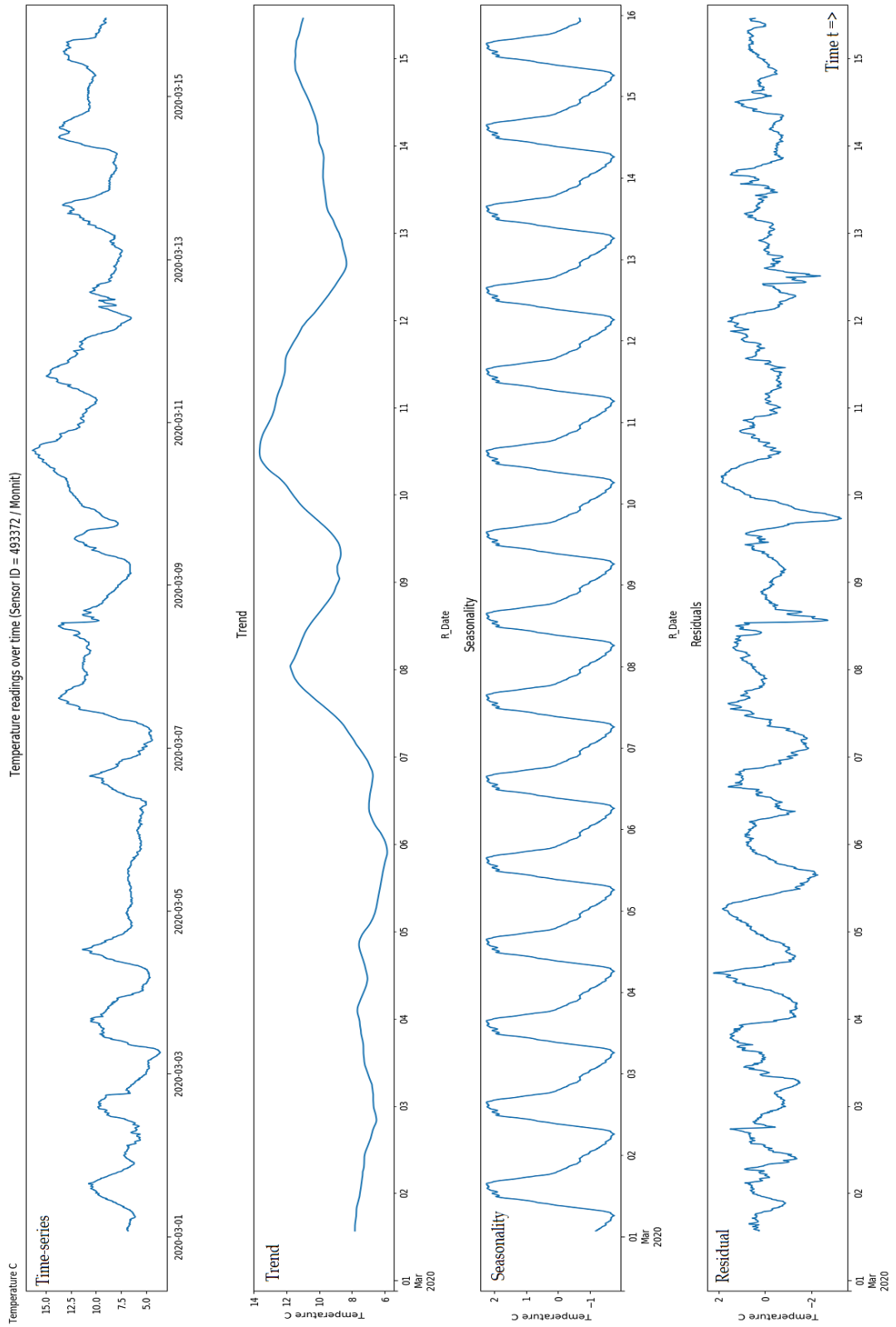


Figure 4.17: Time series decomposition of the benchmark time-series (Sensor ID: Monnit/493372).

graph of the real-world time-series. The time-series decomposition graph revealed a trend relatively similar to the trend of the benchmark time-series.

Both the real-world and ideal time-series were re-sampled, the re-sampling process is necessary to reconstruct the dataset into a time-series in which observations are precisely spaced in time. The re-sampling process does not change the values of observations. It equalises the time spaces between observations. The re-sampling process was configured to compensate missing values of the real-world time-series using the mean value of observations before and after the missing observations, if any. The re-sampling process makes it possible to merge the two datasets into one with the timestamp attribute as a common index, as shown in Figure 4.19¹³.

4.2.1.2 Holt-Winters

Holt-Winters' seasonal method is a statistical forecasting technique that involves predicting future observations of time-series that exhibit trend and seasonality patterns. The design of Holt-Winters model was illustrated in Section-3.5.2. This section is to present the empirical results of applying Holt-Winters (H-W) using both datasets described in Section- 4.2.1.1 while evaluating its accuracy, performance and the feasibility of automation. Practically, the Holt-Winters Python package provided by Statsmodels.org (Seabold & Perktold, 2010) was used to implement this test. Holt-Winters package does not need any of the three smoothing constants (α, β, γ) required by Holt-winters algorithm to be pre-configured in advance. In contrast, it has an optimisation mechanism to determine these smoothing constants from the training dataset and based on getting more basic parameters as input, as shown in Figure 4.20.

Both the *trend* and *seasonality* parameters were set to additive. The "*seasonal periods*" parameter was set to 144 (6 observations per hour * 24 hours), the configuration

¹³https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html

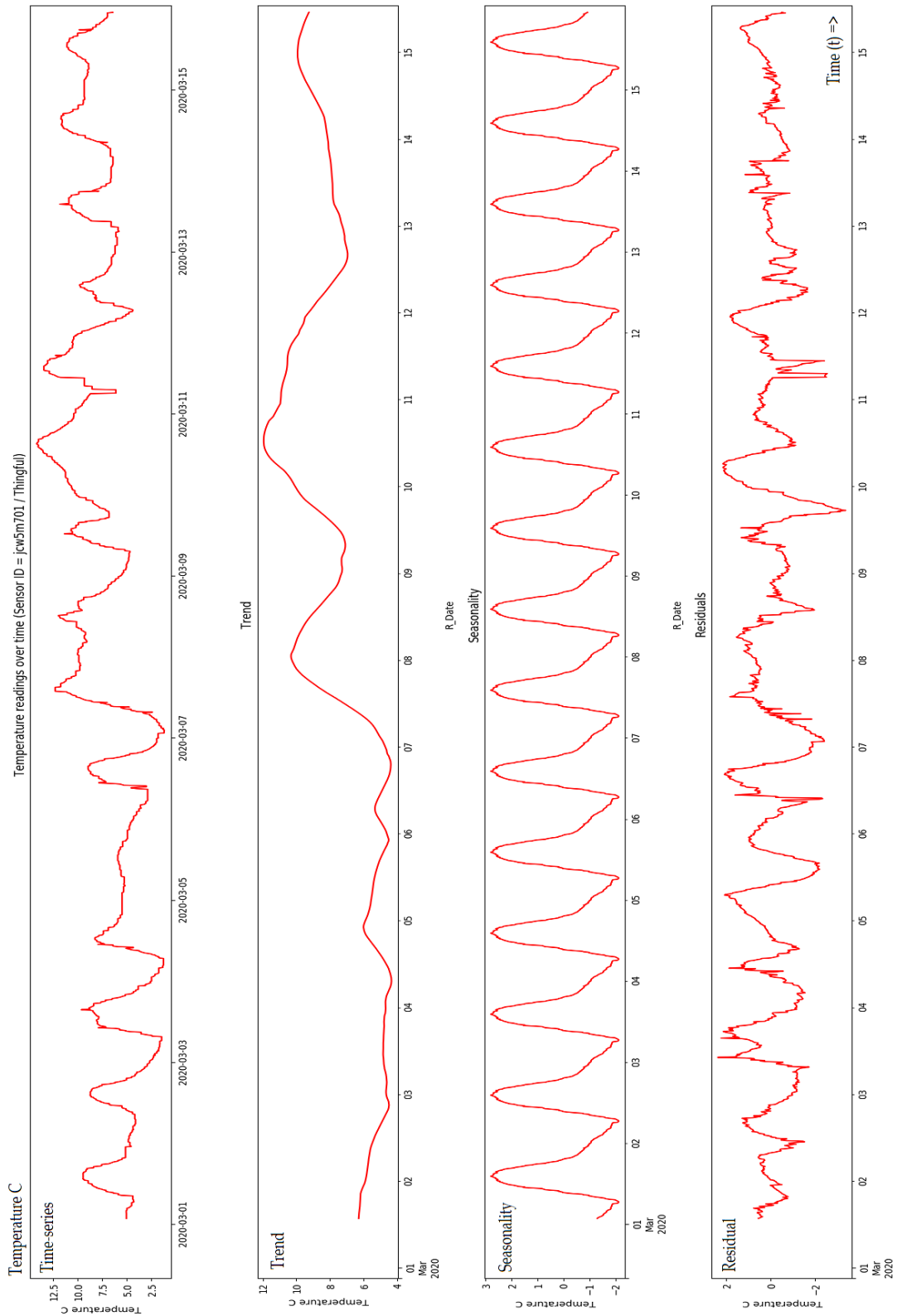


Figure 4.18: Time series decomposition of the real-world sensor node time-series (ID: Thingful/jcw5m701).

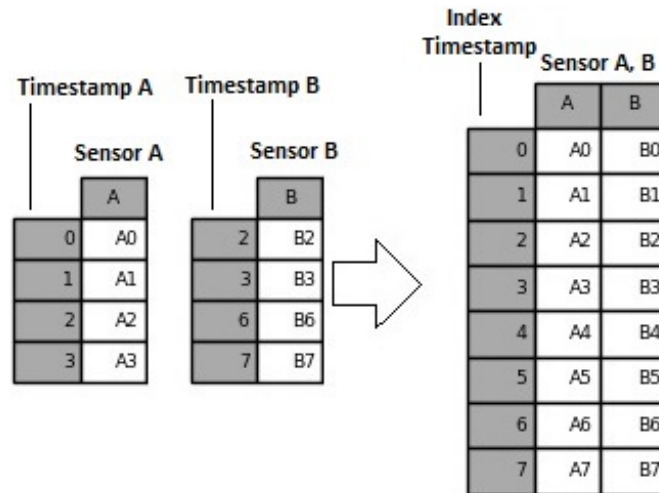


Figure 4.19: The merge and re-sampling processes applied to the selected dataset to become a time-series with the timestamp attribute as a common index (Pandas, 2020).

```
statsmodels.tsa.holtwinters.ExponentialSmoothing

class statsmodels.tsa.holtwinters.ExponentialSmoothing(endog, trend=None, damped=False, seasonal=None,
seasonal_periods=None, dates=None, freq=None, missing='none')[source]
```

Figure 4.20: The structure of Holt-Winters Python package and its input parameters (Seabold & Perktold, 2010).

and programming details are illustrated in Appendix A, Section A.4.1. The results of testing the Holt-Winters autoregressive model are as follows:

4.2.1.2.1 Model Accuracy

The H-W predictive model accuracy was evaluated based on a range of 25 tests using different dataset combinations with one step forward shift in each dataset. These sequential tests were conducted to evaluate the H-W models' prediction accuracy using a wide range of testing instances. These tests were conducted using a four days' time-series window of both the ideal and the real-world datasets. The test time-series consists of over 570 observations, with a shift of one observation (one step forward) in each test, for a 25 step, as shown in Figure 4.21.

Although Holt-Winters prediction model showed its capability to produce rel-

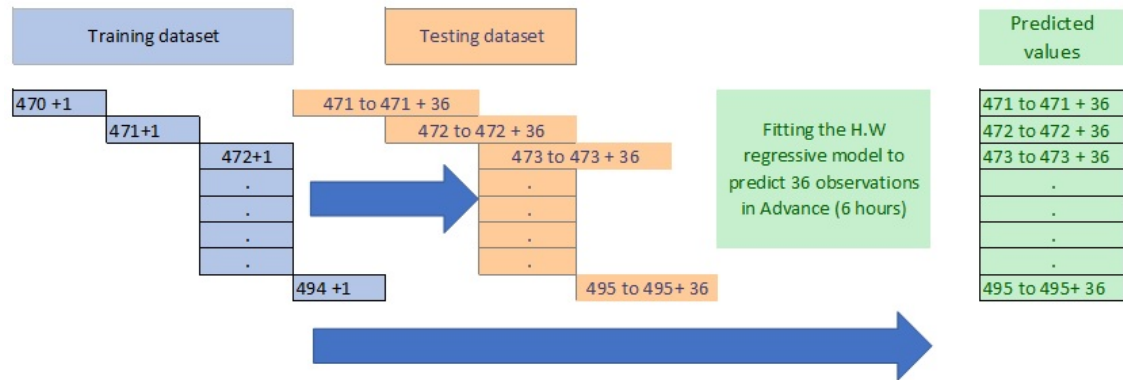


Figure 4.21: Holt-Winters was tested with twenty-five different dataset combinations with one step forward observation shift in each test.

actively accurate predictions, its predictions accuracy was not sustainable. The accuracy of H-W predictions was alternating randomly with each new observation fitted to the model using the sequential testing approach. For example, the difference between the predicted and the actual observation is 0.3°C at step 554, while this difference jumps to 5°C after a one-step forward shift in the time-series, as shown in Figure 4.22.

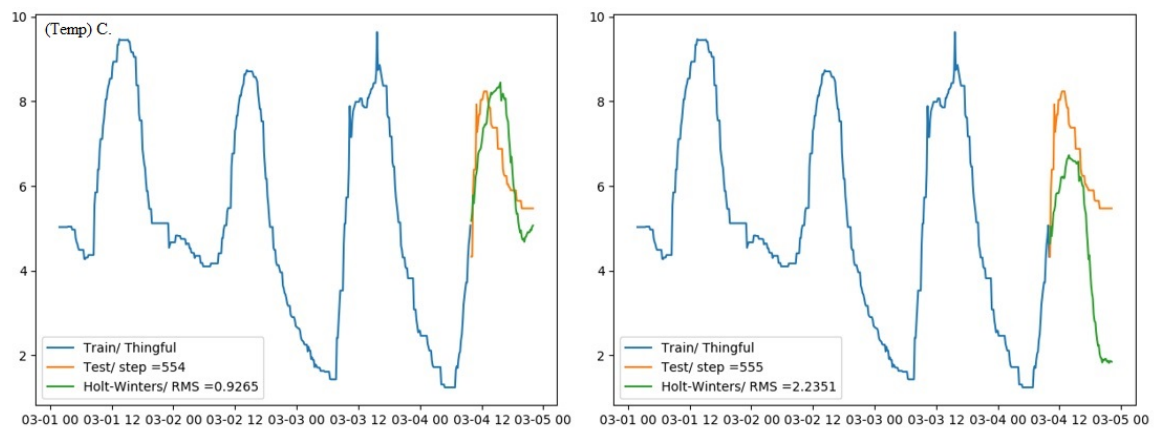


Figure 4.22: The predictions accuracy of H-W were alternating randomly with each new observation fitted to the model.

The ideal dataset was utilised to verify whether the accuracy fluctuating of H-W prediction model resulted from the presence of data quality issues, such as outliers, in the real-world dataset. This behaviour of accuracy fluctuating was identified in W-H prediction model with both the ideal and the real-world datasets.

Prediction model's accuracy can be measured using a scale-dependent unit based on the mean absolute or squared forecast error. The forecast error does not describe a mistake; it is the distance between an observation and its forecast value, and it is on the same scale of the evaluated observations (Hyndman & Athanasopoulos, 2018), as shown in Equation 4.1 (Hyndman & Athanasopoulos, 2018)¹⁴.

$$\text{Absolute Forecast Error } (e_t) = |\text{Predicted observation} - \text{Actual observation}| \quad (4.1)$$

One of the most common scale-dependent forecasting accuracy measurement methods is the Root Mean Squared Error (RMSE), which is based on the forecast error, as shown in Equation 4.2.

$$\text{Root mean squared error (RMSE)} = \sqrt{\text{mean}(e_t^2)} \quad (4.2)$$

The prediction accuracy of the H-W model was rapidly changing with each newly fitted dataset using the one-step forward shift approach, as shown in Figure 4.23.

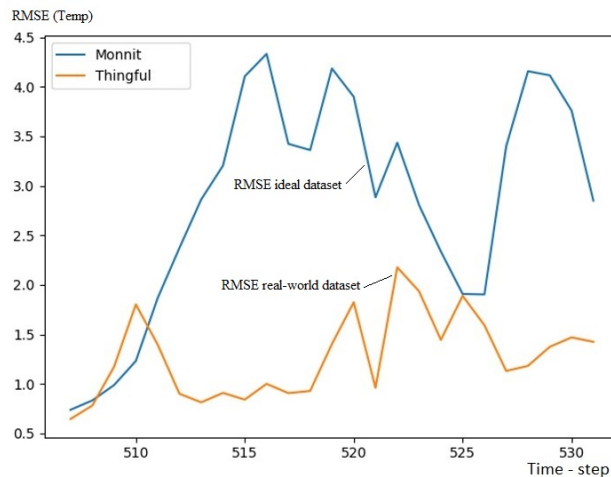


Figure 4.23: The RMSE of the H-W prediction models is rapidly and randomly changing with each newly fitted observation.

The inconsistency in the accuracy of predictions of the H-W regressive model is shown in Figure 4.24. It compares the accuracy of predictions between the ideal

¹⁴<https://otexts.com/fpp2/accuracy.html>

(left) and the real-world (right) time-series at the same point in time.

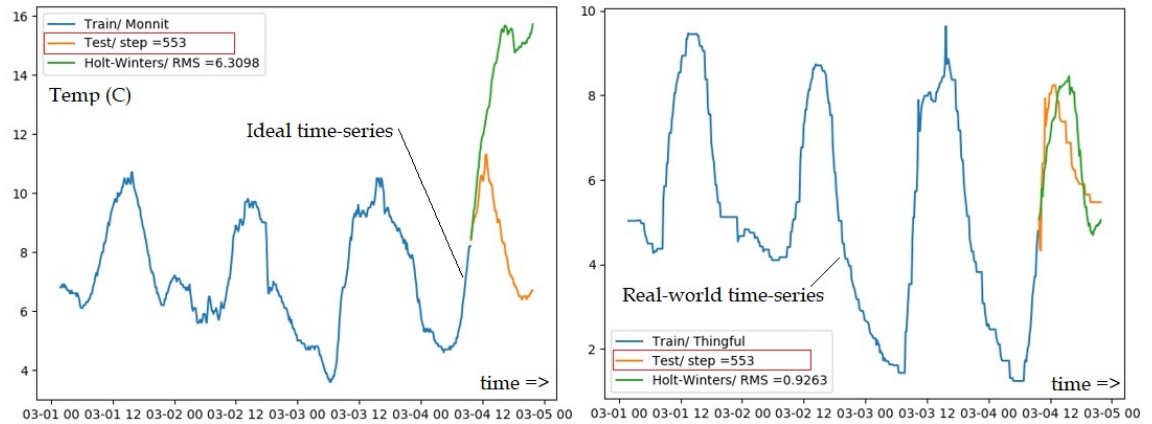


Figure 4.24: H.W regressive model is showing inconsistency in the accuracy of predictions between the ideal (left) and the real-world (right) time-series at the same point in time.

The H-W model was tested to predict the values of the following 323 observations (3 days). This test revealed an essential behaviour of H-W regressive model, as shown in Figure 4.25.

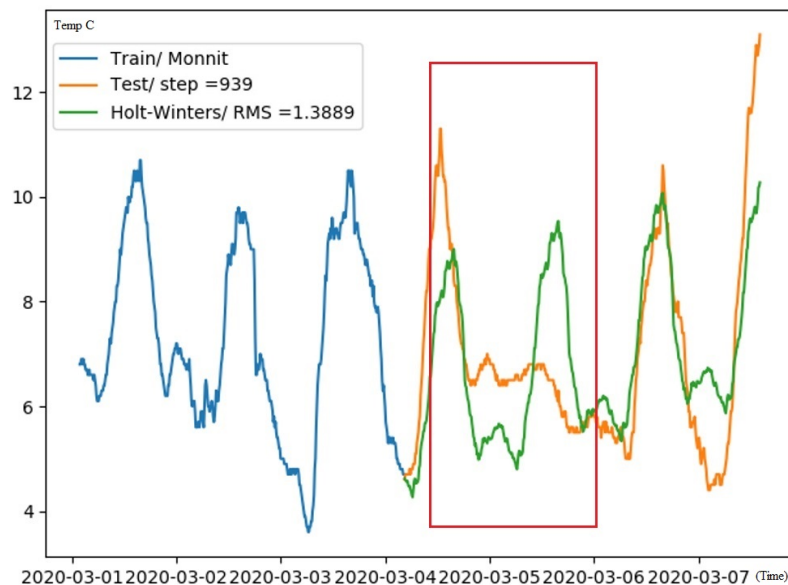


Figure 4.25: H-W prediction model can not adapt to rapid changes in the trend of the time-series.

This test revealed that H-W prediction model requires a time-series with a relatively consistent trend and regular seasonality to sustain accurate predictions. H-W prediction model determines the smoothing parameters of a time-series and

repeats them to estimate the values of future observations. Thus, H-W prediction model can not adapt to rapid changes in the trend of the time-series. Therefore, H-W prediction model can not be adopted as a reliable data accuracy assessment method because of its inconsistency in predictions accuracy and because of its limited ability to adapt with time-series that may exhibits some rapid trend changes.

4.2.1.2.2 Performance

As expected from a statistical-based prediction model, Holt-Winters rendered the prediction results in a brief interval in all the conducted tests. The most extended processing interval was 0.3 Sec with the ideal dataset and 0.24 Sec with the real-world dataset. The time needed for each of the sequential tests to render the prediction results for both the ideal benchmark and the real-world datasets is shown in Figure 4.26.

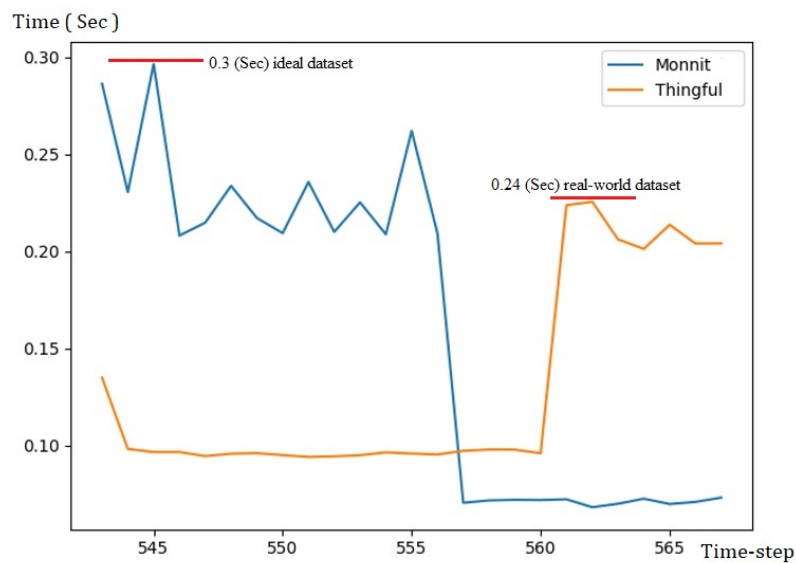


Figure 4.26: The time required by the Holt-Winters seasonal model to render the prediction result of the 25 sequential tests.

4.2.1.2.3 Feasibility of Automation

Holt-Winters is a fully automated prediction analysis technique. Initially, no calculations, configurations or tests on the dataset were required before fitting it

to the H-W prediction model. All the parameters expected by the H-W prediction model, e.g. the *seasonal periods*, were extracted from the time-series and all are constants where no regular changes are required. Therefore, technically, H-W seasonal method could be fitted into fully automated systems or applications easily.

4.2.1.3 ARMA, Non-Seasonal ARIMA and Seasonal ARIMA

This section presents the empirical results of testing ARMA, ARIMA and SARIMA prediction models using both datasets described in Section 4.2.1.1. The structure and the designs of ARMA models' as sensor nodes' measurement accuracy assessment mechanism are illustrated in Section 3.5.3. The configuration and programming details of ARMA predictive models are facilitated in Appendix A, Section A.4.2 and Section A.4.3.

ARMA, Non-Seasonal ARIMA and Seasonal ARIMA are statistical data analysis methods which combine both Autoregressive (AR) and Moving Average (MA) models and inherits their features. The ARMA model is limited to process stationary time-series, which do not exhibit a trend or seasonality. It also requires two parameters to be pre-set: p and q , where p is the order of the autoregressive model and q is the order of the moving average model. Time-series which exhibit non-stationary properties can be transformed into stationary using differencing and, if needed, seasonal differencing. ARIMA model has a built-in feature which can process the differencing process of non-stationary data, while Seasonal ARIMA model can process non-stationary seasonal time-series. Thus it introduced the seasonal parameters $(P,D,Q)m$ to the ARIMA model, where (P,D,Q) are orders of the seasonal part of the SARIMA model and m is the number of observations in a single seasonal period. Table 4.3 shows the different parameters of the ARMA models and their requirements.

Table 4.3: ARMA models parameters and requirements.

Model	Parameters		Requirements
ARMA	p	Autoregressive order (AR)	No trend
	q	Moving average order(MA)	No seasonality
ARIMA	p	Autoregressive order (AR)	No seasonality
	q	Moving average order(MA)	
	d	Difference order	
SARIMA	p	Autoregressive order (AR)	/
	q	Moving average order(MA)	
	d	Difference order	
	P	Seasonal autoregressive order	
	Q	Seasonal difference order	
	D	Seasonal moving average order	
	m	Time steps in a single seasonal period	

Before using the ARMA models, it is required to test the stationary status of the time-series (time-window) used for training the model, as illustrated in Figure 3.17. It is possible to use Time Series Decomposition (TSD) in order to inspect the time-series visually. However, this approach is not practical since a quantitative indicator is required to evaluate the stationary status of the time-series to support a fully automated system without human intervention. In general, stationarity¹⁵ (unit root) tests can be used to identify stationary time-series (Kirchgässner & Wolters, 2007, p. 178), the stationarity status of the ideal and the real-world datasets time-window were examined using the following stationarity tests:

- **Augmented Dickey-Fuller (ADF)** is a popular time-series stationarity test. It can mainly detect level and trend stationarity, and it lacks the ability to distinguish between non-stationary and near stationary (near unit root) time series correctly. The null hypothesis of the ADF test is that the time-series is not-stationary (has a unit root) the alternative hypothesis is that the time series is stationary (Kočenda & Černý, 2015, p. 70-72).

¹⁵<https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc442.htm>

- **Kwiatkowski-Phillips-Schmidt-Shin (KPSS)** is the response to the weakness of the ADF test. Near stationary time-series that incorrectly specified by the ADF test as non-stationary can be correctly identified as stationary with the KPSS test. However, the KPSS test is sensitive to the size of the test sample. It becomes less likely to reject the null hypothesis of stationarity with relatively large samples. The theoretical approach behind the KPSS is entirely different from the Dickey-Fuller based stationary tests. The null hypothesis of KPSS is that the time-series is stationary against the alternative hypothesis that the time series is not-stationary (Kočenda & Černý, 2015, p. 73-75).

In this case study, both ADF and KPSS tests were adopted. The tested time-series would be treated as a non-stationary if ADF, and KPSS tests provided conflicting results. Practically, the training dataset was divided into two datasets, the variance and the mean values of each sub-dataset were calculated and compared, as shown in Figure 4.27.

Shape of the DataFrame: (865, 1)	Shape of the DataFrame: (865, 1)
The size of the training dataset :692	The size of the training dataset :692
The size of the testing dataset :173	The size of the testing dataset :173
mean1=6.859491, mean2=7.726538	mean1=4.931586, mean2=5.573224
variance1=3.063529, variance2=7.900799	variance1=4.332560, variance2=11.797593

Figure 4.27: Testing the stationarity of ARMA training datasets by dividing each dataset and comparing the mean and the variance values of each sub-set. (left) ideal dataset, (right) real-world dataset.

The mean and the variance values of the two halves of the ideal dataset did not match with a significant deviation, which indicates that the dataset is not stationary. The same results were obtained from the real-world dataset. These outcomes were confirmed by the results of applying the ADF and the KPSS stationary tests on the ideal training dataset, Figure 4.28 and the real-world dataset, Figure 4.29.

Applying the ADF and KPSS tests on the ideal dataset produced contradictory

```

=====The ADF (Augmented Dickey-Fuller) Test =====
Checking the time series
<<<The null hypothesis: the time series is non-stationary>>>
Reject the Null hypothesis, the data is Stationary
The suggested order of Differencing is: 0
=====
=====The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test =====
Checking the time series
<<<The null hypothesis: the time series is stationary>>>
Reject the Null hypothesis, the data is Non-Stationary
The suggested order of Differencing is: 1

```

Figure 4.28: The results of applying the ADF and KPSS stationarity tests on the ideal dataset.

results, as shown in Figure 4.28, while applying the same tests on the real-world dataset, showed that the data is not stationary in both tests. as shown in Figure 4.29.

```

=====The ADF (Augmented Dickey-Fuller) Test =====
Checking the time series
<<<The null hypothesis: the time series is non-stationary>>>
Failed to reject the null hypothesis, the data is Non-Stationary
The suggested order of Differencing is: 1
=====
=====The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test =====
Checking the time series
<<<The null hypothesis: the time series is stationary>>>
Reject the Null hypothesis, the data is Non-Stationary
The suggested order of Differencing is: 1

```

Figure 4.29: The results of applying the ADF and KPSS stationarity tests on the real-world dataset.

The first-order differencing function was applied to stabilise the datasets into stationary status. Applying the ADF and KPSS tests on the new datasets showed that both datasets were stationary.

The typical way to estimate the values of ARMA models parameters (p,d,q) is by using the Autocorrelation Factor (ACF) and the Partial Autocorrelation Factor (PACF)¹⁶ diagrams. The ACF and the PACF diagrams can be visually inspected

¹⁶Autocorrelation measures the linear relationship between lagged values of a time series.

to determine the required ARMA parameters based on the distribution and the relationship among the lagged values presented by these diagrams (Hyndman & Athanasopoulos, 2018, p. 47). The ACF and the PACF diagrams of the real-world dataset are shown in Figure 4.30.

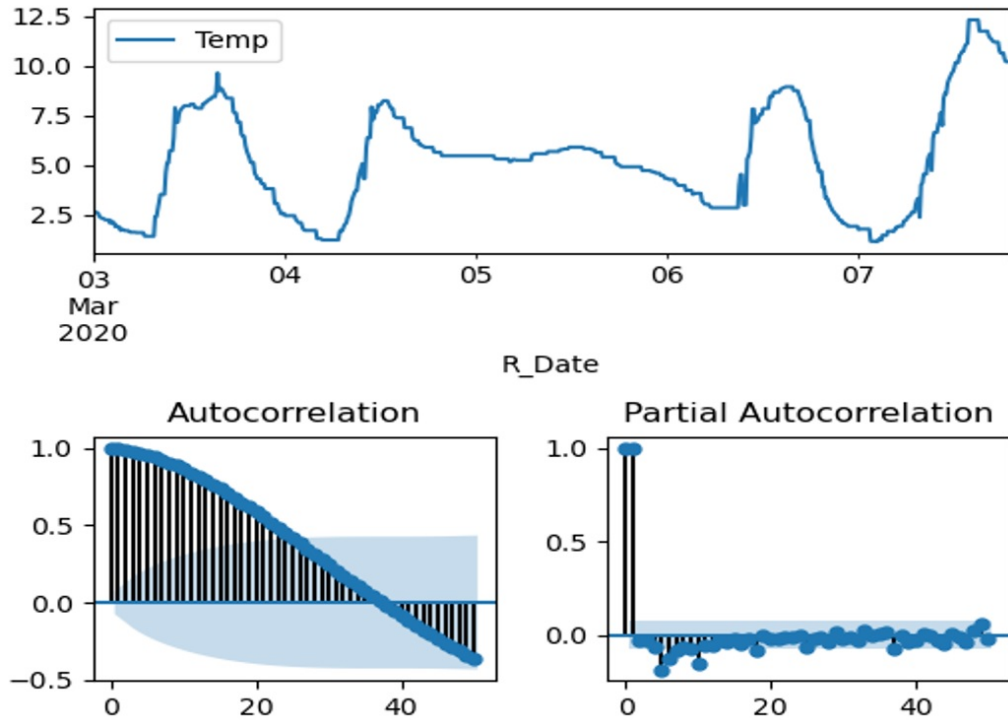


Figure 4.30: The ACF and PACF diagrams of the real-world (stationary) dataset.

The PACF diagram is showing two extreme lagged¹⁷ values (out of the light blue area) which indicates that the order of the autoregressive parameter (p) equals two. The ACF diagram is showing that the first two lags are entirely out of the threshold '95%' confidence area (the light blue area), which refers to the value of the moving average parameter (q), in this case, equals two¹⁸.

In order to get a quantitative means to determine the values of the ARMA models parameters, "the Grid Search, and AIC" method was utilised. This method is essential for automating the ARMA models, since it does not require human intervention to estimate the ARMA model parameters via the visual inspection

¹⁷A variable which its value depends on an earlier point in time.

¹⁸Using the ACF and PACF within the context of this research is for demonstration only, Grid Search, and AIC was used to determine the parameters of the ARMA models.

of the ACF and PACF diagrams. This method is based on testing ARMA models using different combinations of parameters and apply the Akaike Information Criteria (AIC) test on the outcome of each test. The combination of parameters which delivers the lowest AIC, is the most suitable to optimise the prediction accuracy of the ARMA model.

The disadvantage of the Grid Search and AIC method is that it becomes more time-consuming process with testing higher order models or when used to determine the parameters of a complicated ARMA model such as the SARIMA model. The results of applying the Grid Search and AIC method on the ideal dataset and the real-world dataset, using the ARMA model, are shown in Figure 4.31.

```

a)
Parameters with the lowest Akaike information criterion (AIC) :
p      2.000000
q      1.000000
aic   -934.716005

b)
Parameters with the lowest Akaike information criterion (AIC) :
p      2.000000
q      2.000000
aic    73.794283

```

Figure 4.31: The values of ARMA parameters of the ideal dataset (a) and the real-world dataset(b) determined using the Grid Search and AIC method.

Figure 4.31 indicates that the best parameters combination to optimise the ARMA model of the ideal dataset is (2,1), and (2,2) is the best parameters combination to optimise the ARMA model of the real-world dataset. The results of applying the Grid Search and AIC method on the ARIMA model using the ideal and the real-world datasets are shown in Figure 4.32.

The results of applying the Grid Search and AIC method on the ideal dataset and the real-world dataset, using SARIMA model, are shown in Figure 4.33.

The residual of the ARMA, ARIMA and SARIMA models with the lowest AIC

```

a)
Parameters with the lowest Akaike information criterion (AIC) :
p      2.000000
d      1.000000
q      1.000000
aic   -934.716005

b)
Parameters with the lowest Akaike information criterion (AIC) :
p      2.000000
d      1.000000
q      2.000000
aic    73.794283

```

Figure 4.32: The values of ARIMA parameters for the ideal dataset (a) and the real-world dataset(b) determined using the Grid Search and AIC method.

```

a)
The smallest AIC is -768.6331817858872 for model SARIMA(2, 0, 1)x(0, 0, 0, 4)

b)
The smallest AIC is -6.051259167300817 for model SARIMA(3, 1, 1)x(0, 0, 1, 4)

```

Figure 4.33: The values of SARIMA parameters for the ideal dataset (a) and the real-world dataset(b) determined using the Grid Search and AIC method.

were checked to make sure that they are normally distributed around zero and stationary. For example, the residual ACF and PACF of the ARMA (2,2) model of the real-world dataset reside almost entirely under the threshold value of the 95% confidence, as shown in Figure 4.34.

The residuals of the ARMA (2,2) models with the estimated parameters using the Grid Search and AIC method for the real-world dataset is shown in Figure 4.35. The residuals of the optimised ARMA models did not show any notable autocorrelation, no trend or seasonality, which means it is acceptable to fit these models using the estimated parameters.

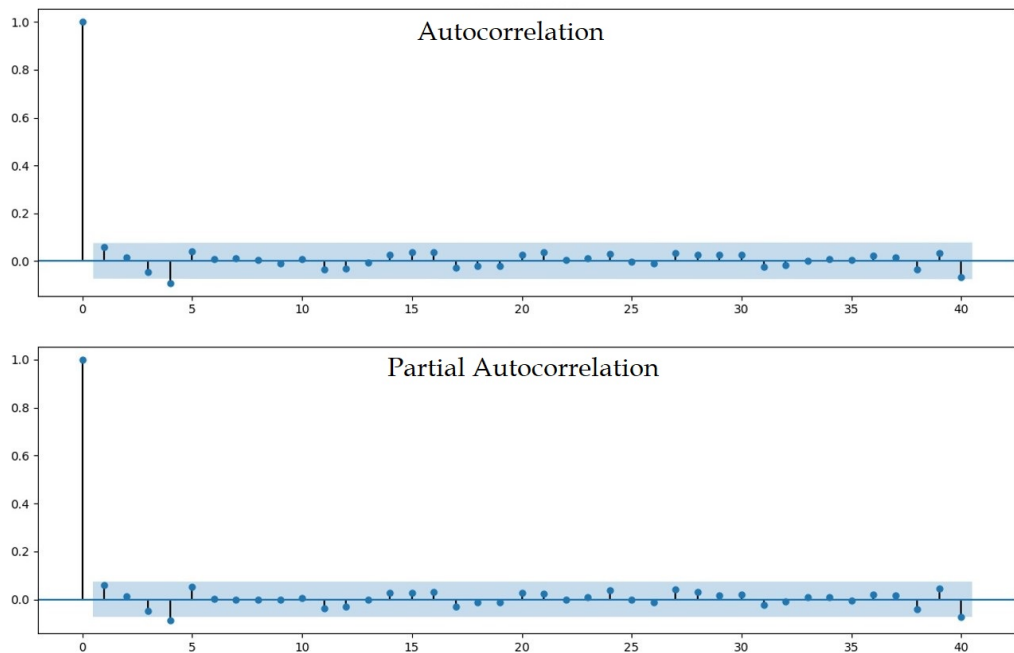


Figure 4.34: The residua ACF and PACF diagrams of the ARMA (2,2) model applied to the real-world dataset.

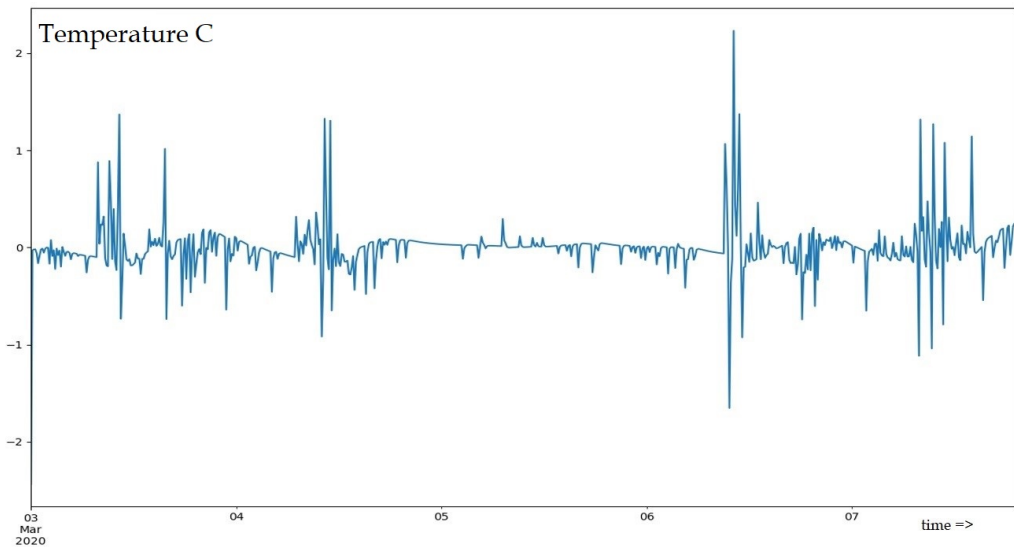


Figure 4.35: The residuals of the ARMA (2,2) models applied to the real-world dataset.

The result of fitting the ARMA (2,1) model with the testing part of the ideal dataset is shown in Figure 4.36.

While the results of fitting the ARMA (2,2) model with the testing part of the

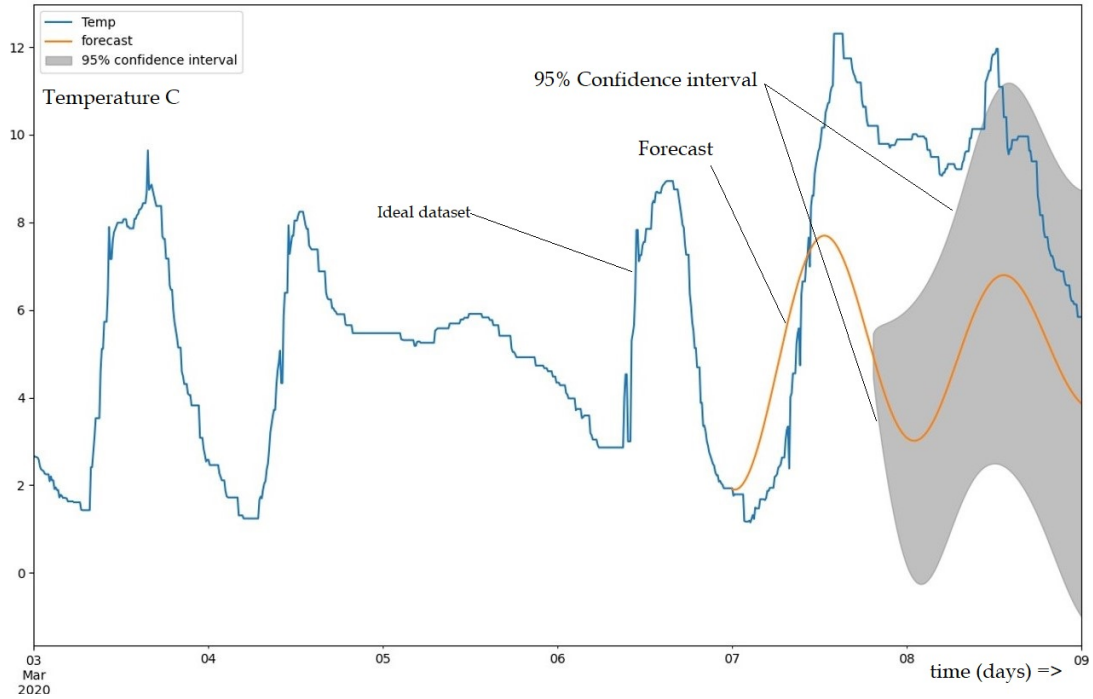


Figure 4.36: The result of fitting the ARMA (2,1) model with the testing part of ideal dataset.

real-world dataset is shown in Figure 4.37.

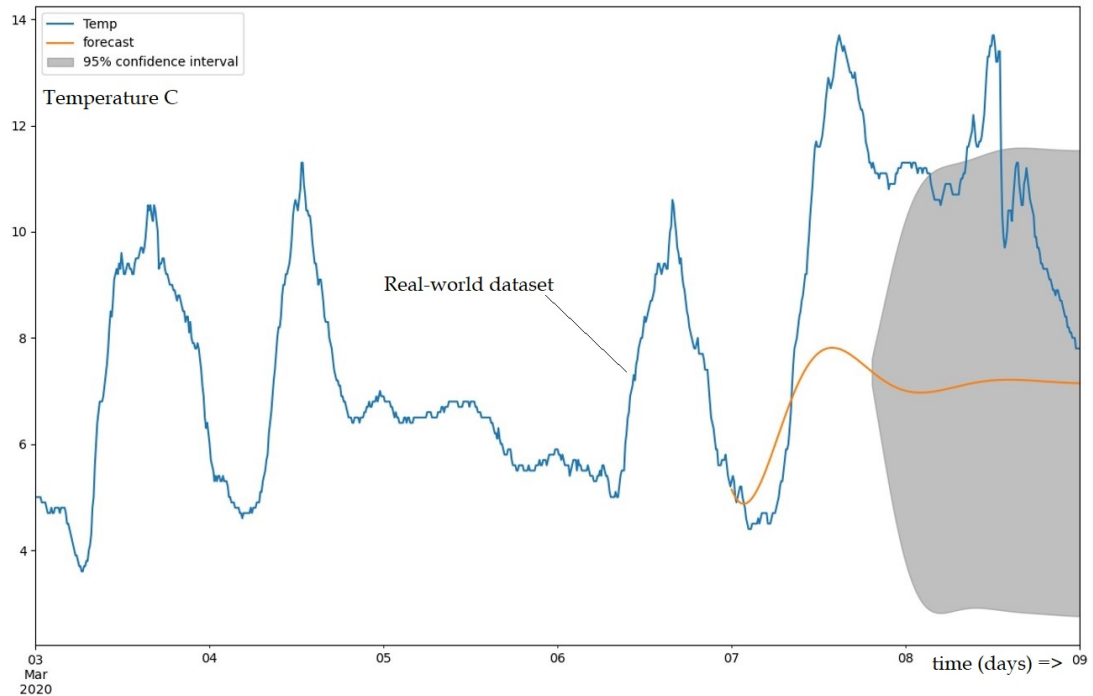


Figure 4.37: The results of fitting the ARMA (2,2) model with the testing part of the real-world dataset.

Both results are showing a relatively similar prediction pattern. The results are relatively accurate at the earliest steps of predictions, which rapidly decrease until they reach the grey area of the 95% confidence, where the predictions become significantly unreliable.

Since the duty-cycle of the sensor nodes, in both datasets, is very short (10 minutes) comparing with the prediction interval of the ARMA models. It is possible to select only the first predicted value and use it in the training set to predict the next observation, which is known as **The Rolling One-step Forecasts** (Hyndman & Athanasopoulos, 2018, p. 84).

The results of adopting the one-step forecasts approach with ARMA, ARIMA, and SARIMA models were significantly similar. Figure 4.38 shows the result of applying the one-step forecast approach to the ARMA model using the real-world dataset.

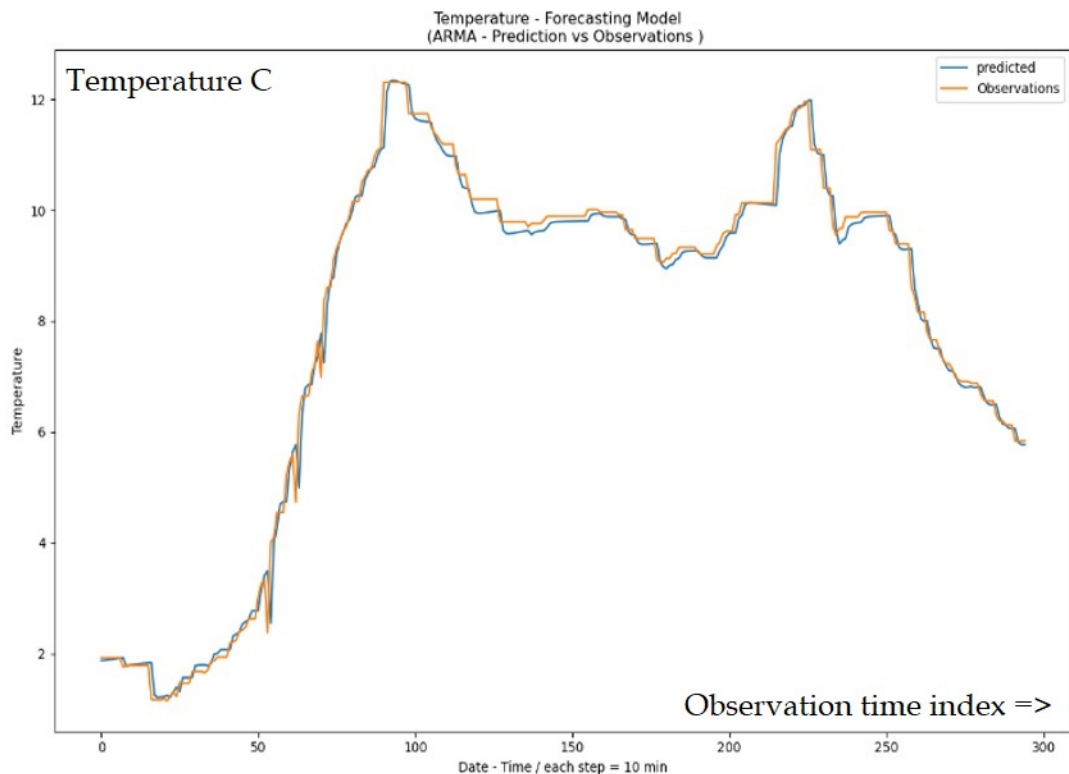


Figure 4.38: The one-step forecast approach applied to the ARMA models using the real-world dataset.

An overview of the full dataset, SARIMA prediction graph (green) and SARIMA one-step prediction approach using the real-world dataset are shown in Figure 4.39.

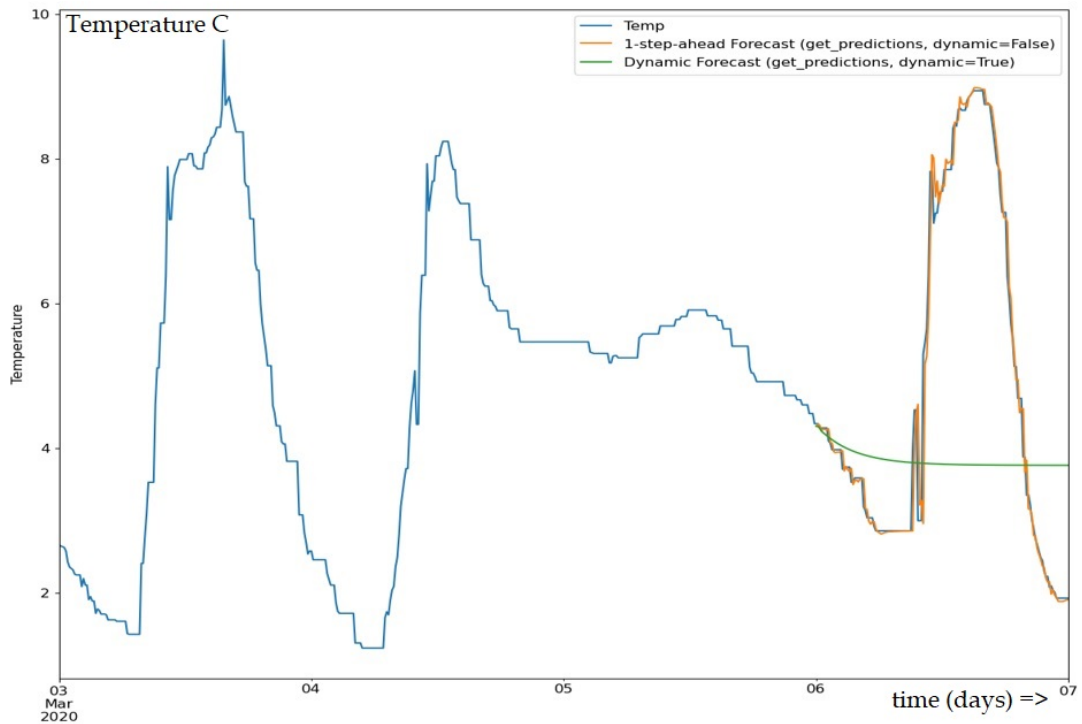


Figure 4.39: An overview of the full dataset, SARIMA prediction graph (green) and SARIMA one-step prediction approach using the real-world dataset.

ARMA, ARIMA and SARIMA prediction models were evaluated based on their predictions accuracy, performance and the feasibility of automation, as follows:

4.2.1.3.1 Accuracy

ARMA models share the same mathematical foundation combining autoregressive and moving average models into one comprehensive model. It is expected to get no significant variations in accuracy among the prediction results of the ARMA models (ARMA, ARIMA and SARIMA). Eight different tests were conducted on the ARMA models to ensure that the prediction accuracy and performance of these models are consistent. ARMA models' accuracy was measured by determining the Root Mean Square Error (RMSE) between the predicted and the actual observations of the test dataset. The one-step forecasting method was evaluated for each of the

ARMA models using the ideal and the real-world datasets. The results are listed in Table 4.4.

Table 4.4: The RMSE of the ARMA prediction models for the ideal and the real-world datasets.

Dataset	Model	RMSE
Ideal (Local Network)	ARMA	0.1823
	ARIMA	0.1772
	SARIMA	0.1364
Real-world (Large-scale Network)	ARMA	0.2497
	ARIMA	0.2528
	SARIMA	0.3229

In general, the values of the RMSE were relatively low. The values of RMSE of the ideal dataset were systematically lower than the RMSE of the real-world dataset, which is mainly related to the high quality of data of the ideal dataset which enhances the accuracy of predictions of the ARMA models.

The RMSE of the ARMA's models indicated that these models rendered relatively accurate predictions, but only when applied using the one-step-forward predictions approach (rolling forecasting). The tests showed that any extended range of ARMA, ARIMA and SARIMA forward predictions were always combined with high uncertainty margins. Therefore, the predictions of ARMA models' were only suitable for a relatively short interval accuracy assessment of sensor nodes measurements.

4.2.1.3.2 Performance

The performance of ARMA models was measured based on the time required to optimise (to determine the best combination of parameters) the models and the time required by the models to render the prediction results. The time required to optimise the ARMA and ARIMA models is relatively similar, between 3 to 7 Sec, as shown in Figure 4.40 (a) and (b). In contrast, the SARIMA model optimisation time is significantly higher, between 110 to 125 Sec, as shown in Figure 4.40 (c).

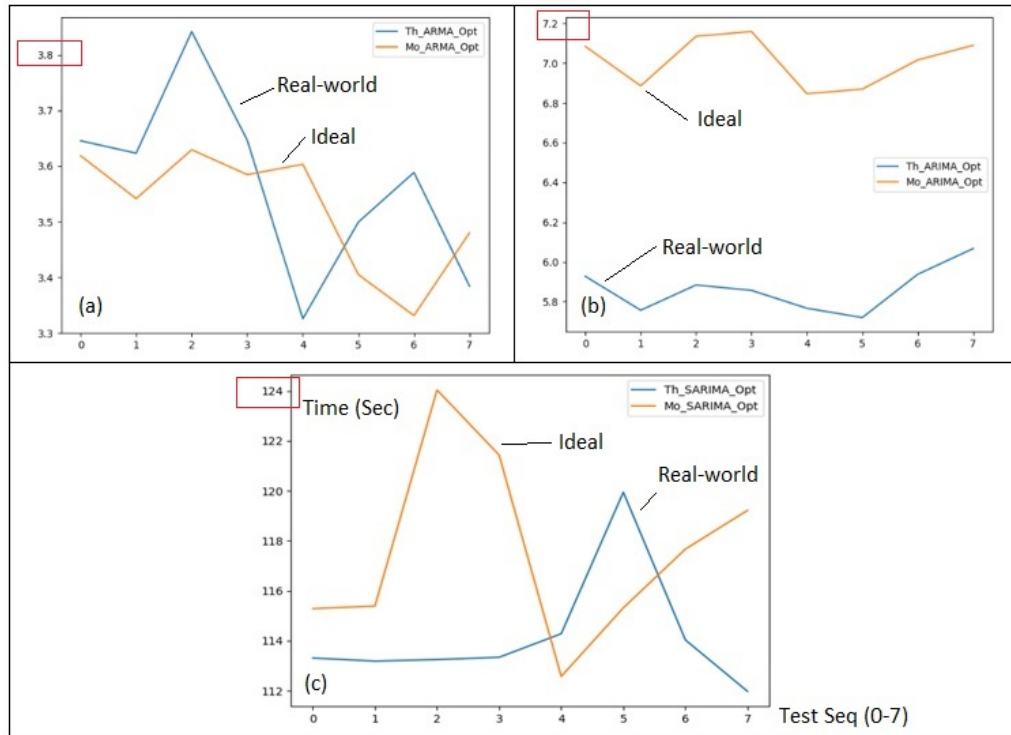


Figure 4.40: The time required to optimise the ARMA (a), ARIMA (b) and SARIMA (c) models for both the ideal and real-world datasets.

The optimisation time of the SARIMA model was significantly higher than other ARMA models. However, the performance of SARIMA was slightly better. The time required for the SARIMA model to render predictions was shorter (0.32 Sec) compared to the time required by ARMA (1.4 Sec) and ARIMA (1.3 Sec) models, as shown in Figure 4.41.

4.2.1.3.3 Feasibility of Automation

SARIMA model has less data preparation requirements comparing to the ARMA and ARIMA models. A sequence of stationarity tests and differencing processes are typically required to be applied to the training dataset before fitting it to the ARMA model. In contrast, SARIMA model can process non-stationary datasets directly, which makes it more likely to be utilised in a fully automated environment comparing to the other ARMA model.

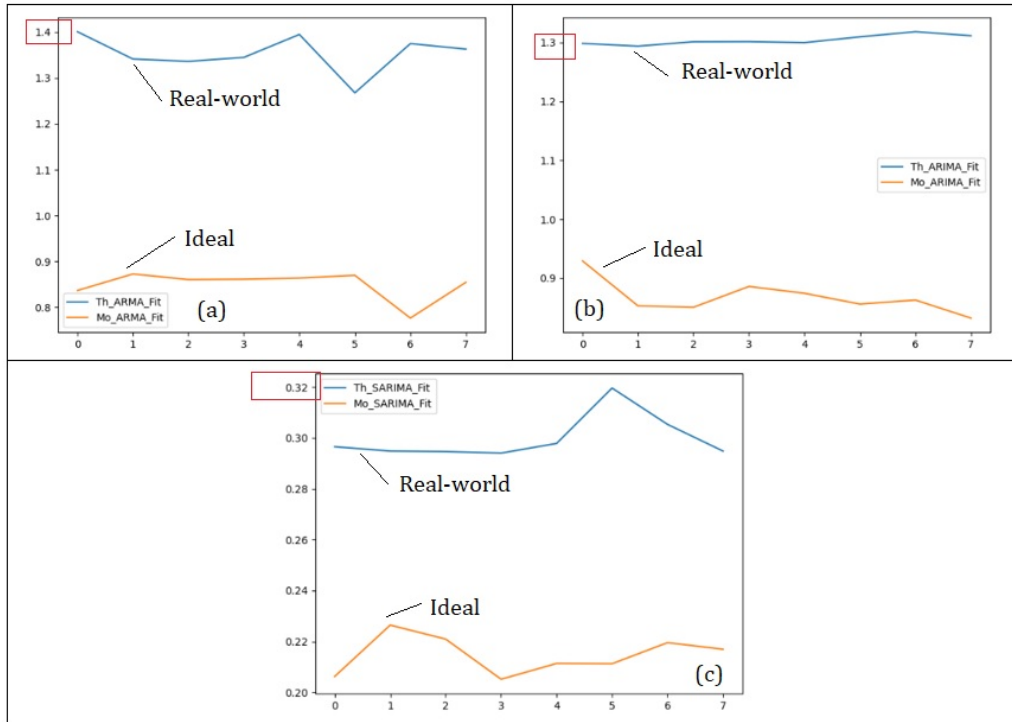


Figure 4.41: The time required by ARMA (a), ARIMA (b) and SARIMA (c) models to render predictions results.

4.2.1.4 Gaussian Process Regression

Since ARMA prediction models rendered relatively accurate predictions, but only when applied using the one-step-forward predictions approach (rolling forecasting) and any extended range of ARMA models predictions always combined with high uncertainty margins. Therefore, the predictions of ARMA models' were only suitable for a relatively short interval accuracy assessment. As an alternative to ARMA models, Gaussian Process Regression (GPR) prediction model was empirically tested in this section. This section outlines the technical details and results of applying the Gaussian Process Regression (GPR) model to the datasets defined in Section 4.2.1.1. The structure details and the design of the GPR model are illustrated in Section 3.5.4.

GPR is a supervised machine learning method utilises Gaussian Processes (GP) in its prediction model. The approach of using GPR as a prediction analysis

model relies on gaussian processes for machine learning approach advanced by Rasmussen and Williams (Williams & Rasmussen, 2006). The GPR model was developed based on the “GaussianProcessRegressor” Python package provided by Scikit-learn (Pedregosa et al., 2011). The configuration and programming details of using “GaussianProcessRegressor” Python package provided by Scikit-learn (Pedregosa et al., 2011) are illustrated in Appendix A, Section A.4.4. The structure of the “GaussianProcessRegressor” model is shown in Figure 4.42.

```
class
sklearn.gaussian_process.GaussianProcessRegressor(kernel=None,
*, alpha=1e-10, optimizer='fmin_l_bfgs_b', n_restarts_optimizer=0,
normalize_y=False, copy_X_train=True, random_state=None)
```

Figure 4.42: The structure and parameters of Gaussian Process Regressor package provided by Scikit-learn (Pedregosa et al., 2011).

Where:

- **Kernel** defines the covariance function of the GPR model. The kernel parameter is set to none by default, which means that the radial-basis function (RBF) kernel will be used. Many off shelf kernels are available which can be adopted directly. Furthermore, it is also possible to develop custom-built kernels.

The "Standard Flexible"¹⁹ kernel was adopted in this case-study and evaluated through many empirical tests. The standard flexible kernel is an off-shelf GPR kernel that combines the "Constant Basic" Kernel (the White kernel) with the "Radial-Basis Function" (RBF) kernel. The purpose of using the Constant kernel is to scale the magnitude of the main kernel (the RBF kernel) since it adjusts the mean value of the GPR and it resembles the noise level of observations. The Radial-basis function (RBF) kernel is a non-linear kernel, most suitable for controlling the smoothness of periodic signals with

¹⁹https://scikit-learn.org/stable/modules/gaussian_process.html#gaussian-process

noise. RBF kernel ratifies the signal noise level using the additional Constant kernel component and also by adjusting the alpha parameter.

- **alpha** is the noise parameter as a positive definite array. Its default value is (1e-10). It defines the noise value associated with the observations of the training dataset and limits any numerical issue during the fitting process, where a higher value of alpha is corresponding to a higher level of noise in observations.
- **optimiser** is the kernel parameters optimiser. Its default value is “fmin_l_bfgs_b” which refers to the internal (built-in) optimiser, and it is also possible to call an external optimiser. If the optimiser parameter is set to “none” the parameters of the kernel will have fixed values, and no optimisation processes will take place.
- **n_restarts_optimizer** is the number of iterations to optimise the GPR kernel. Its default value is 0 which implies that one run will take place based on the initial kernel parameters, practically, “n_restarts_optimizer” should be set between 10 and 20 to optimise the kernel.
- **normalize_y** is the normalisation status indicator of the training dataset. It can be set to true if the mean value is not zero and the variance is not equal to one (Avila & Hauck, 2017, p. 71).

The GPR model was evaluated initially using a basic kernel configuration. The internal optimiser was disabled, and alpha the noise parameter was set to zero. The basic GPR model was evaluated using 3 hours time-window from both ideal and the real-world datasets. The result of fitting the basic GPR model using the ideal dataset is shown in Figure 4.43.

The continues red line after the last observation (the last red dot) is the value of the prediction with the highest probability known as the mean. At the same time,

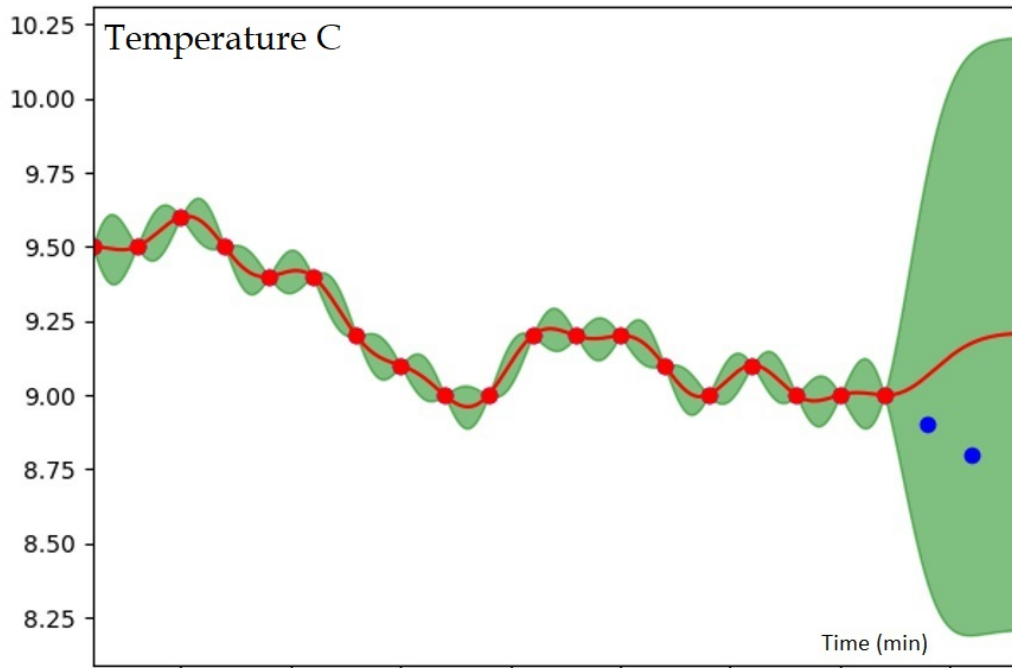


Figure 4.43: The result of fitting the ideal dataset to the basic GPR model.

the green margins are the ranges of confidence of the other possible values, which is known as the variance.

The confidence margins between the observations in Figure 4.43 are relatively low, about 0.125°C , however, these confidence margins get extremely large in the prediction interval, and it gets more significant with time. The blue dots are the actual future observations (19 and 20) which were not included in the fitted dataset. Both are within the confidence margins, which is a good indicator, but, still, the mean of the predicted values is significantly diverting from the actual observations.

The result of fitting the corresponding time-window of the real-world observations to the same GPR model is shown in Figure 4.44.

The confidence margins of the real-world dataset got relatively low, which is expected because of the inaccuracy associated with real-world observations comparing with the ideal dataset which was collected from a high-quality and fully controlled sensor node network. The mean value in the prediction interval is

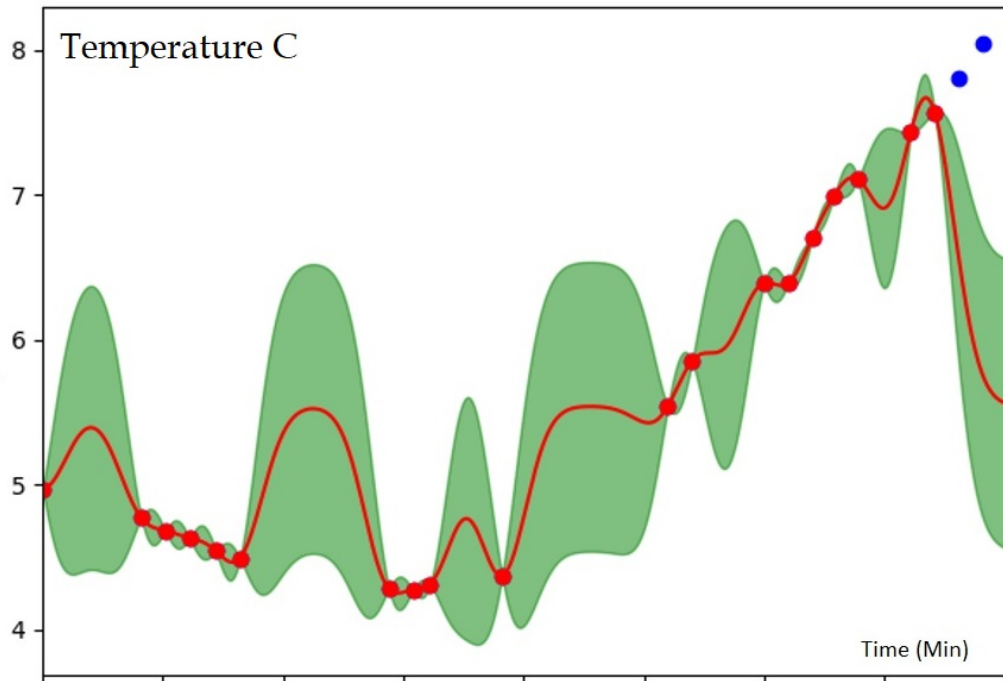


Figure 4.44: The result of fitting the real-world dataset to the basic GPR model.

highly diverted away from the actual observations. As illustrated in Figure 4.44, the real observations (blue dots) are outside the confidence margins, which indicates that GPR model with the basic configuration may produce inaccurate predictions and out of the acceptable confidence margins.

The GPR tests were reconstructed using the same kernel, but with the internal optimiser of the GPR model set to activate, the number of iterations was set to 20, and the noise parameter alpha was activated by setting it to 5. Since the optimiser parameter was activated, the GPR model will learn (adjusts its kernels parameters) from the training dataset. To achieve that, the dimensions of the dataset had to be transformed from a single dimension time-series into a two dimensions array of dependent and independent variables, as shown in Figure 3.20. Since the confidence intervals of the GPR model tend to be lower with time, the rolling one-step forecasts approach was adopted to select only the first predicted value and use it in the training set to predict the next observation (Hyndman & Athanasopoulos, 2018, p. 84). The result of fitting the ideal datasets to the new GPR model is shown

in Figure 4.45.

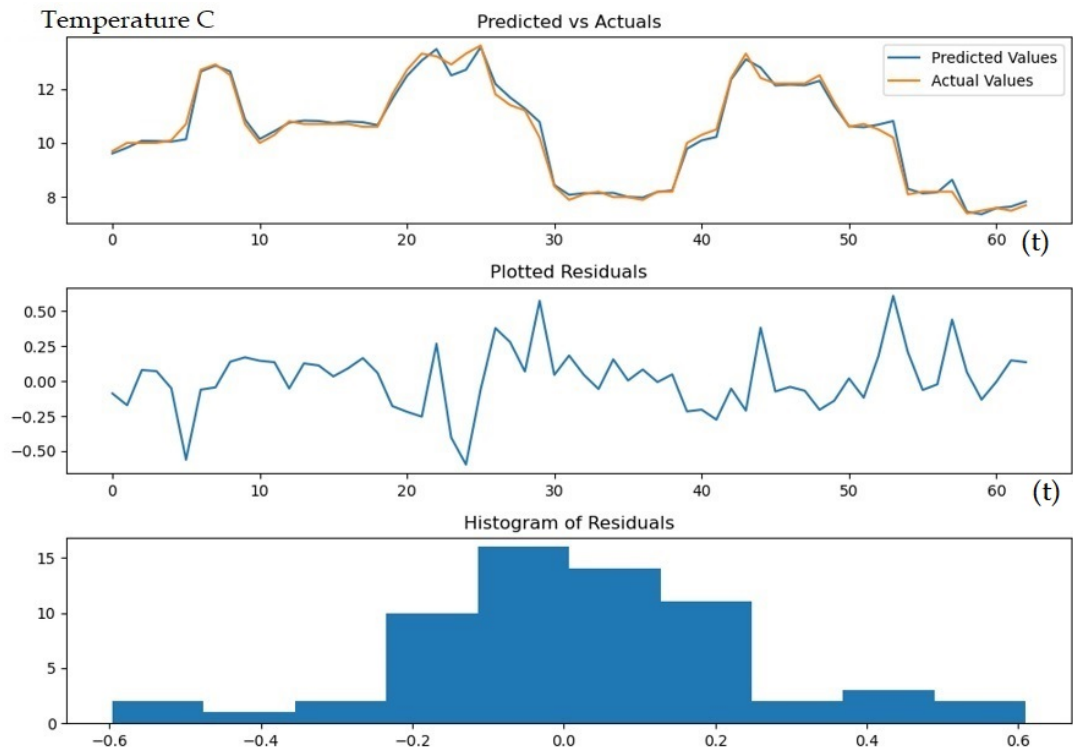


Figure 4.45: The result of fitting the ideal datasets to the GPR model with the optimiser iterations set to 20 and alpha to 5.

Repeating the same test produced roughly the same outcome with some differences. These differences are expected since the optimisation process of the GPR model randomly selects the GPR model parameters in each iteration within the allowed range of values (Pedregosa et al., 2011)²⁰.

The outcome of fitting the real-world dataset to the same GPR model used with the ideal dataset is shown in Figure 4.46.

²⁰https://scikit-learn.org/stable/modules/gaussian_process.html

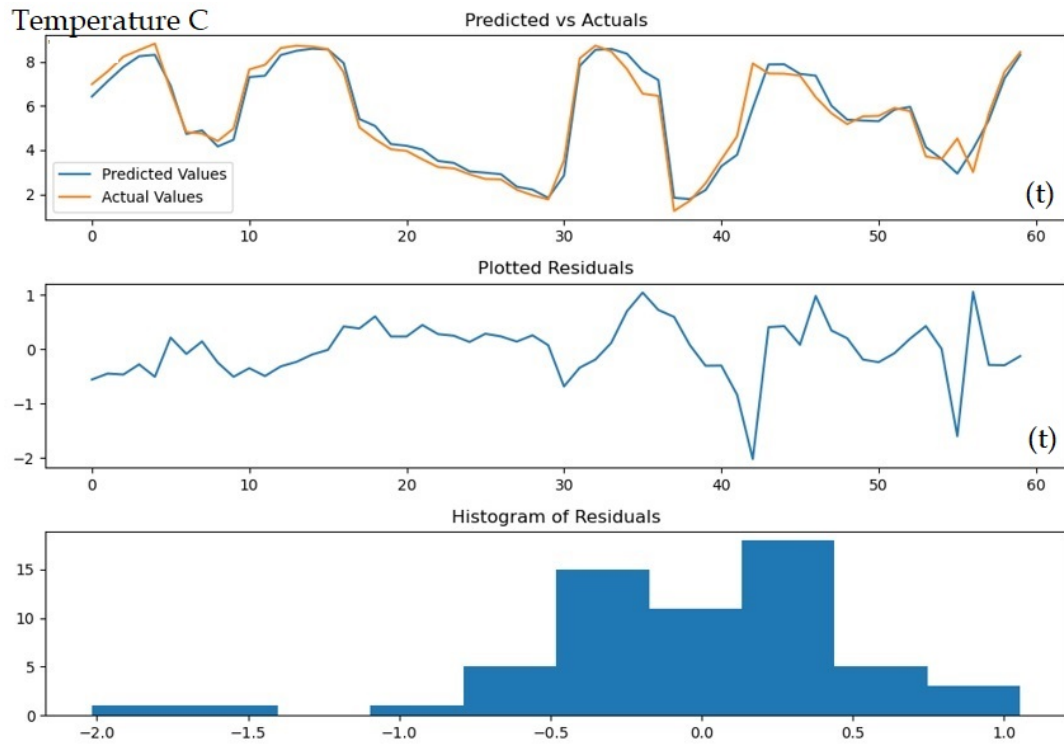


Figure 4.46: The result of fitting the real-world datasets to the GPR model with the optimiser iterations set to 20 and alpha to 5.

The GPR model was evaluated based on the accuracy of its prediction results, performance and the feasibility of automation, as follows:

4.2.1.4.1 Accuracy

The predictions accuracy of the GPR model was evaluated using the Root Mean Square Error (RMSE) between the predicted and the actual observations for the ideal and real-world datasets. Based on a the results of 25 sequential tests, the accuracy of predictions of the GPR model showed consistent results at $RMSE = 0.245$ for the ideal dataset and 0.475 for the real-world dataset. The significant difference between the of the RMSE of the GPR prediction model when applied to the ideal dataset 0.245 comparing to the real-world dataset 0.475 is justified because of the RMSE measure sensitivity to outliers in the dataset. The value of RMSE measurements may significantly affected by vertical outliers (observations' value attribute outliers) in the evaluated dataset (Jeyaraman et al., 2019; Khotimah

et al., 2019; Pachepsky & Rawls, 2004, p. 397). To overcome the measurement limitation of RMSE when applied to datasets with outliers, the prediction accuracy of the GPR model was evaluated again using the Coefficient of Determination. The coefficient of determination measures the amount of variation between two parameters (Singh, 2003, p. 585). The value of the coefficient of determination varies between 0 no correlation to 1 fully identical parameters (Keller, 2015, p. 135). Considering a predictive (regression) model, the coefficient of determination R^2 equals the square of the correlation between the predicted and the actual observations which can be calculated as shown in Equation 4.3 (Hyndman & Athanasopoulos, 2018)²¹.

$$R^2 = \frac{\sum (\hat{y}_t - \bar{y})^2}{\sum (y_t - \bar{y})^2} \quad (4.3)$$

Where y is a sensor node time-series, y_t is an observation at time t , \hat{y}_t is the predicted observation at time t and \bar{y} is the mean value on n observations of y (Hyndman & Athanasopoulos, 2018; Davis, 1995, p. 85-86).

The results of applying the coefficient of determination measure to the outcome of the GPR predictive model for both the ideal and real-world datasets are 0.970 and 0.946 respectively, which shows that the GPR model could render significantly accurate estimations of observations value attribute for both the ideal and the real-world datasets. However, the GPR predictive model rendered relatively accurate predictions, but only when applied using the one-step-forward predictions approach (rolling forecasting). The tests showed that any extended range of GPR forward predictions were always combined with high uncertainty margins. Therefore, the predictions of GPR model were only suitable for a relatively short interval accuracy assessment of sensor nodes measurements.

4.2.1.4.2 Performance

²¹<https://otexts.com/fpp2/least-squares.html>

The performance of the GPR model was evaluated based on the time required to optimise the model and the time needed for the model to render the prediction results. The performance tests were based on 25 sequential tests for the GPR model using both the ideal and the real-world datasets. GPR model showed a relatively stable performance of an average of 5.5 ± 1.5 Sec for the selected time-window for both the ideal and real-world datasets, as shown in Figure 4.47.

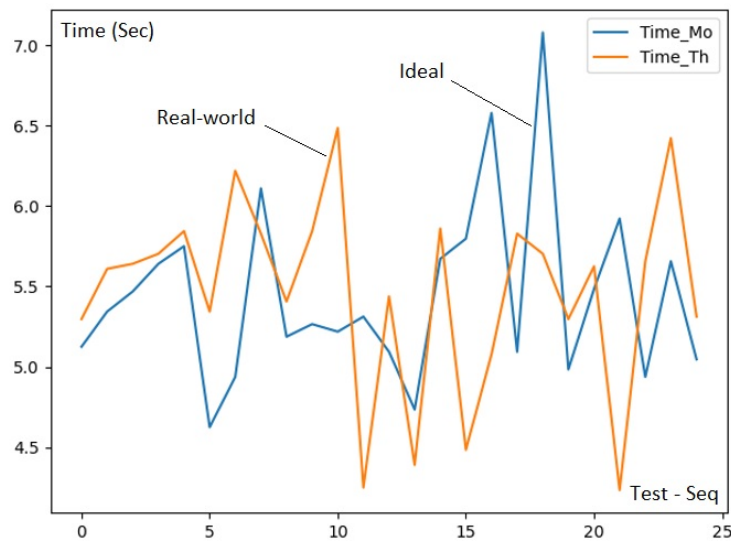


Figure 4.47: The time required by the GPR models to render prediction results of 25 sequential test for the ideal (Blue) and the real-world (Orange) datasets.

4.2.1.4.3 Feasibility of Automation

The data preparation requirements of the GPR prediction model were less compared to the ARMA or H-W models. However, the GPR model itself requires many configuration parameters to be set, some of which are difficult to be selected or optimised automatically. For example, the kernel of the GPR model has a noise parameter which must be estimated separately from the main noise parameter (α). Moreover, testing the validity of GPR models based on testing the residual stationary status may add more complexity to the automation process of the model which requires adding stationarity tests and differencing steps to its development process.

4.2.1.5 Long Short-Term Memory Networks

ARMA and GPR predictive models rendered relatively accurate predictions, but only when applied using the one-step-forward predictions approach. The tests showed that any extended range of ARMA or GPR forward predictions were always combined with high uncertainty margins. Therefore, the predictions of these models were only suitable for a relatively short interval accuracy assessment of sensor nodes measurements. Therefore, in this section, LSTM was tested as a prediction analysis mechanism for accuracy assessment for sensor nodes measurement as an alternative to ARMA and GPR models. LSTM is the extension to the deep learning-based Recurrent Neural Networks (RNN) utilised in this research as a prediction model. The design of the LSTM model is illustrated in Section 3.5.5. This section is to outline the technical details and results of applying the LSTM model on the datasets defined in Section 4.2.1.1. The first step before training the LSTM model is to prepare the dataset to be compatible with the data format requirements of the LSTM input layer. The data preparation process involves the following steps:

- **Resampling**, is a necessary process to reconstruct the dataset into a time-series in which observations are precisely spaced in time. The re-sampling process equalises the time spaces between observations without changing their values and compensates missing values by using the mean value of observations before and after the missing ones.
- **Scaling** is the process of changing the value of time-series observations to become within a given range, in this case between 1 and -1. Scaling is a recommended process when using neural networks or deep learning models to eliminate the effect of extreme values and to optimise the model performance (Raschka & Mirjalili, 2017, p. 42-54). Scaling was achieved

using the “MinMaxScaler” function developed by Scikit-learn (Pedregosa et al., 2011)²².

- **Creating a regressor array** (independent variables) and the target array (dependent variable) for training and validating the LSTM model. The same approach in Figure 3.21 was used where the independent variable array consists of 7 observations for each row. The final shape of the training and validating datasets is a 3D array of [samples, time steps, features], where the number of features is set to one (Temperature).

The next step is to create the LSTM model by using a constructor to define the model parameters and calling the fit function to train the model using the training dataset. Keras²³ functional API was used to construct the LSTM model. Keras is a deep learning framework with an industrial scale of features that covers a wide range of machine learning workflow aspects from data management to solution deployment. The main steps to build the LSTM model are:

- Defining LSTM model input layer by setting its shape to (7, 1) and the type of the variable to float32.

```
Input_layer = Input(shape = (7,1), dtype = "float32")
```

- Defining the LSTM layer by calling the LSTM library from Keras recurrent package and defines its main parameters.

```
lstm_layer = LSTM(64, input_shape = (7,1), return_seq = False)(input_layer)
```

The first parameter (64) is the Units indicator which defines the number of the LSTM cells (dimensions) that will be initialized.

²²<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html>

²³<https://keras.io/>

- Adding the dropout layer and defining its rate. Dropout is a common and very effective regularisation technique that can reduce over-fitting in neural networks. It works by randomly setting to zero some of the layer output features during the training phase of the model in order to introduce some arbitrary noise to the learning process. The dropout rate parameter controls the dropping ratio, and it can be set between 0.2 to 0.5 (Chollet, 2017, p. 109).

```
dropout_layer = Dropout(0.2)(lstm_layer)
```

- Adding the output layer based on the "Dense Layer", which is a neural network layer that connects every neuron in the previous layer to every neuron in the next layer. Since it establishes every possible connection between layers, it called the dense layer (Moolayil et al., 2019, p. 29). The activation parameter is to select the mathematical equation that defines the layer output, in this case, a linear function, which determines the output by multiplying the input by the weights for each neuron.

```
output_layer = Dense(1, activation = 'linear')(dropout_layer)
```

- Training the model by fitting the training datasets into the model object defined by the following:

```
ts_model.fit(x=X_train,y=y_train,batch_size=16,epochs=20,  
verbose=1,callbacks=[save_best],validation_data  
=(X_val,y_val),shuffle=True)
```

The "Batch" is a training sample from the training dataset defined by the "batch_size" parameter. The LSTM network updates its weight after processing each batch. An "epoch" is a complete cycle of processing the training dataset (all batches) and successfully updating the weight parameters of the

model. The epoch parameter determines the iteration number of processing all available batches. With each epoch, the selection of batches initialises randomly, which optimises the model weight parameters and enhance the accuracy of its predictions.

The output of the LSTM model fitted with the ideal dataset is shown in Figure 4.48.

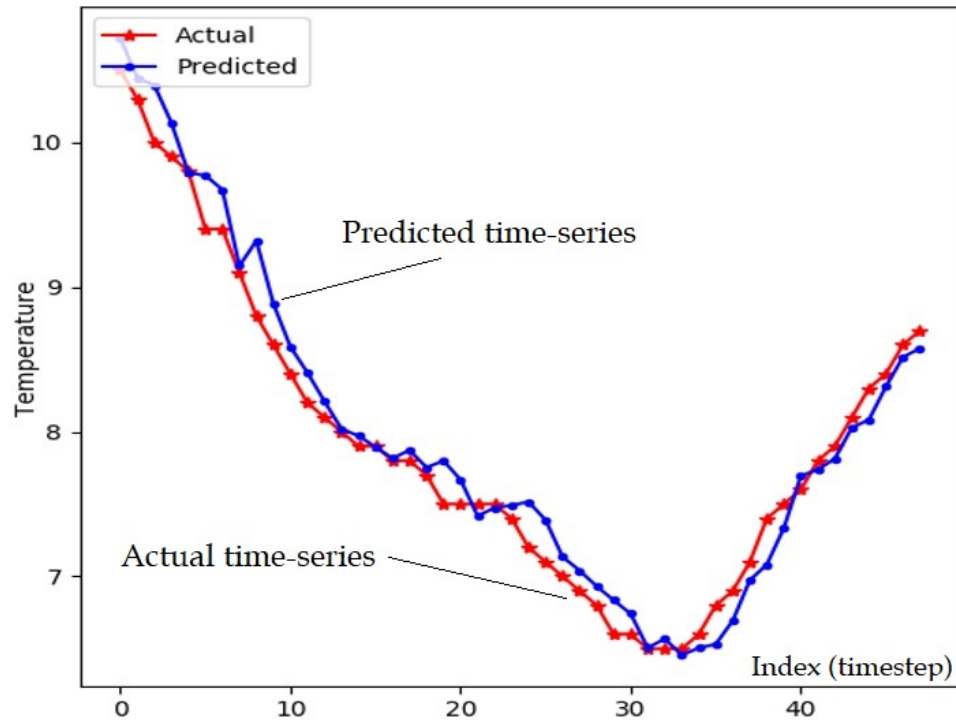


Figure 4.48: The output of the LSTM model fitted with the ideal dataset.

The output of the LSTM model fitted with the real-world dataset is shown in Figure 4.49.

The configuration and programming details of using Keras Python package to construct the LSTM prediction model are illustrated in Appendix A, Section A.4.5. The LSTM model was evaluated based on the accuracy of its predictions, performance and the feasibility of automation, as follows:

4.2.1.5.1 Accuracy

The accuracy of the LSTM model was evaluated using the Coefficient of Determination and the Root Mean Square Error (RMSE) measures between the predicted

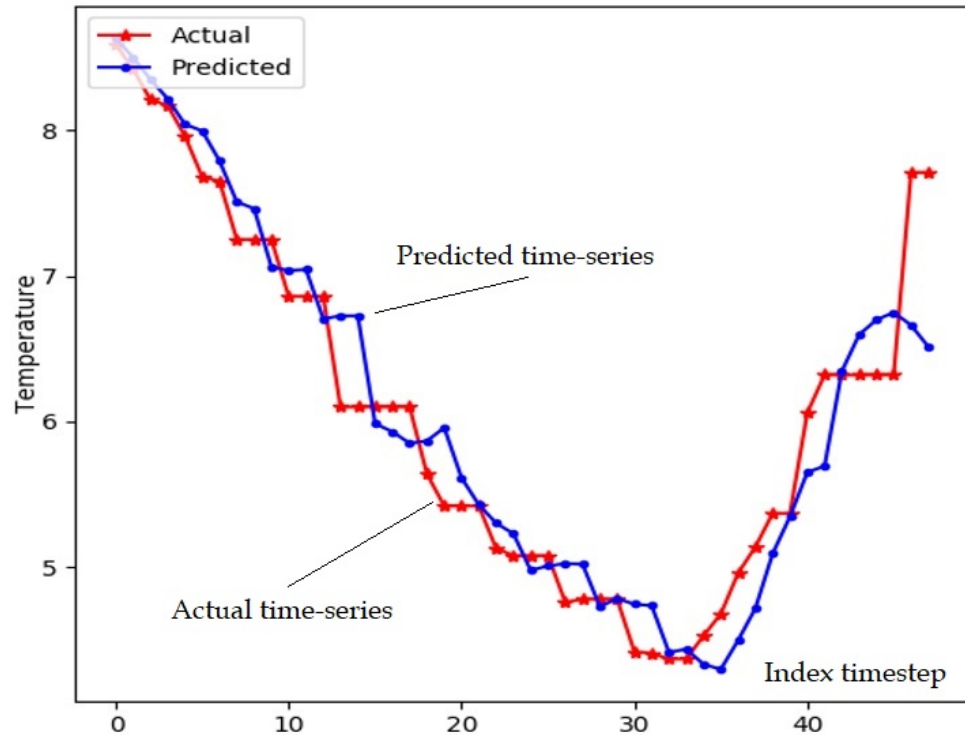


Figure 4.49: The output of the LSTM model fitted with the real-world dataset.

and the actual observations. A set of 25 sequential tests were conducted on the ideal and real-world datasets. The accuracy of the LSTM prediction model based on the coefficient of determination measure was consistent around 0.969 for the ideal dataset and 0.889 for the real-world dataset. And the results of applying the RMSE were 0.183 for the ideal dataset and 0.408 for the real-world dataset.

4.2.1.5.2 Performance

The performance of the LSTM models was measured based on the time required to train the model and the time needed for the model to render the prediction results. The performance evaluation was based on 25 sequential tests for the LSTM model using both the ideal and the real-world datasets. LSTM model showed a steady performance of rendering the results of an average of 20.2 +/- 0.4 Sec for the selected time-window for both the ideal and real-world datasets, as shown in Figure 4.50.

4.2.1.5.3 Feasibility of Automation

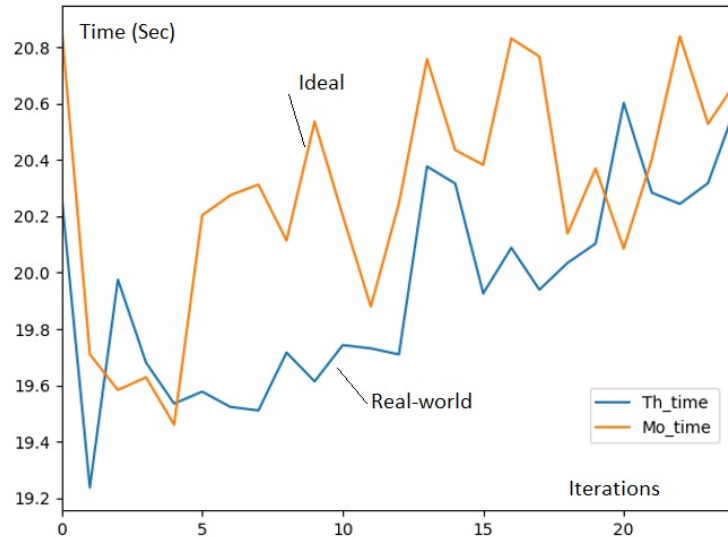


Figure 4.50: The time required by the LSTM model to render prediction results of 25 sequential test for the ideal(Orange) and the real-world(Blue) datasets.

Using the LSTM prediction model involves many data preparation steps, which are relatively generic comparing to the GPR model. LSTM model introduces a random noise factor to the dataset and trains its cells to deal with such kind of observations based on a built-in multi iteration training process. In general, LSTM model is more suitable to work in an automated environment, especially considering that it requires fewer adjustment steps and fewer optimisations processes comparing to the GPR model since most of these processes are built-in the LSTM model.

4.2.1.6 Utilising LSTM for Anomaly Detection

This section aims to find a valid predictive analysis model to be utilised as an accuracy assessment mechanism for sensor nodes measurements in large-scale CPS. As discussed earlier, since predictive analysis models rely on autoregression, these models can be utilised only to detect accuracy issues that occur for a relatively short intervals only (point outliers). Holt-Winters' seasonal was investigated first as a prediction model. However, it could not adopt with rapid changes in time-series, and it was sensitive to short-term alterations in the value attribute of observations in the training dataset. Therefore, the accuracy of the Holt-Winters'

prediction model was iterating unpredictably and thus rendering unreliable results. ARMA's and GPR were investigated as alternatives to Holt-Winters'. ARMA's and GPR predictive models were able to render relatively accurate predictions, but only when applied using the one-step-forward predictions approach (rolling forecasting). Any extended range of forwarding predictions was associated with a high level of uncertainty margins. Thus, ARMA's and GPR models' predictions were only suitable for short interval accuracy assessment. LSTM was investigated to overcome the limitations of ARMA's and GPR predictive models. LSTM model rendered accurate predictions with extended forward range competing with ARMA's and GPR models. The LSTM benefits from long time-series in its "learning" process. It utilises an advance mechanism to deal with noise with minimal configuration requirements. LSTM model is more suitable for working in an automated environment, especially considering that it requires fewer adjustment steps and fewer optimisations processes compared to the ARMA's or GPR models, since most of these processes are built-in the LSTM model.

Therefore, LSTM was selected as the most suitable predictive-based anomaly detection model to evaluate the accuracy of sensor nodes measurement in large-scale CPSs. LSTM model can be used to detect irregularity in observation's values based on comparing the predicted values with the actual observations, as detailed in Section 3.5, and as shown in Figure 3.14.

The first step is to determine the Deviation Threshold, which defines the allowed tolerance between the predicted and the actual observation. Thus measurements that exceed the deviation threshold are considered as observations with potential data accuracy issues, as shown in Figure 3.14. The deviation threshold is the acceptable level of variation in sensor nodes observations' value, which may occur for many reasons not necessarily related to a measurement error or hardware failure. Defining the deviation threshold value depends mainly on the type of application. For example, in critical applications, the deviation threshold could be

fixed to up to 5% or 10% of observation value. In this research, the ideal dataset was used to determine the deviation threshold by calculating the maximum difference between the mean value of observations collected from all local sensor nodes for the same time-window of the tested real-world dataset. All sensor nodes of the local sensors network are high-quality devices which streamed consistent and anomaly free time-series for the same time-window of the real-world dataset. The value differences among concurrent observations collected from these sensors can be used as a reference of the acceptable range of tolerance for this type of sensors, as shown in Equation 4.4.

$$|\text{Deviation threshold}| = \max \begin{cases} \bar{y}_{S1} - \bar{y}_{S2} \\ \bar{y}_{S2} - \bar{y}_{S3} \\ \bar{y}_{S1} - \bar{y}_{S3} \end{cases} \quad (4.4)$$

Where \bar{y}_{S1} , \bar{y}_{S2} and \bar{y}_{S3} are the mean value of the time-series of the three outdoor local sensor nodes S1, S2 and S3 respectively for the selected time window. Equation 4.4 was applied to detected anomalies as shown in Equation 4.5.

$$\begin{aligned} &\text{If } |\text{Actual} - \text{Predicted}| > \text{Deviation threshold then: 'Anomaly'} \\ &\text{Else: 'Pass'}; \end{aligned} \quad (4.5)$$

Applying equation 4.4 on the ideal dataset revealed that the deviation threshold for the selected time-window is 0.34 C°. The LSTM prediction model combined with equation 4.5 and using 0.34 C° as the estimated deviation threshold were utilised successfully to detect anomalies associated with accuracy data quality issues in the real-world dataset, as shown in Figure 4.51.

These detected anomalies can be seen on the real-world dataset as highlighted in Figure 4.52.

Observation_Seq	Actual	Predicted	Prediction_Error	Status
0	8.59	8.625703	0.035703	Pass
12	6.86	6.611035	0.248965	Pass
13	6.10	6.794491	0.694491	Anomaly
14	6.10	6.793972	0.693972	Anomaly
15	6.10	5.675078	0.424922	Anomaly
16	6.10	5.970956	0.129044	Pass
17	6.10	5.975647	0.124353	Pass
42	6.32	6.549469	0.229469	Pass
43	6.32	6.660898	0.340898	Anomaly
44	6.32	6.544711	0.224711	Pass
45	6.32	6.468438	0.148438	Pass
46	7.71	6.399046	1.310954	Anomaly
47	7.71	6.357684	1.352316	Anomaly

Figure 4.51: The LSTM prediction model using 0.34 C° as the deviation threshold was utilised successfully to detect value attribute anomalies in the real-world dataset.

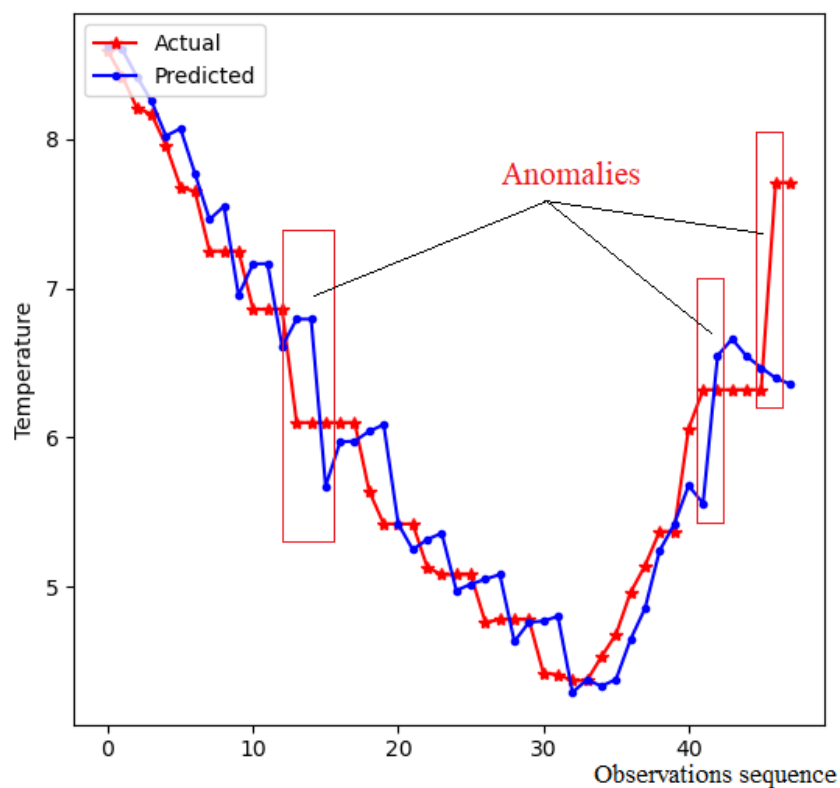


Figure 4.52: The accuracy data quality issues (anomalies) detected by the LSTM prediction model when applied on the real-world dataset.

The full implementation code and other programming details are illustrated in Appendix A, Section A.4.5.

4.2.2 Anomaly Analysis Models

Predictive analysis models are most suitable for detecting irregular sensor nodes measurement events that appear for a short interval (short outliers). The pattern of time-series with long outliers will be distorted to a certain extent reflecting the wrong measurement as the standard pattern which leads to higher prediction errors and limits the ability of the predictive analysis modes (autoregression) to detect the irregular data accuracy events correctly (Berk, 2015, p. 25-27). Therefore, Anomaly analysis models were investigated as data accuracy assessment mechanisms for sensor nodes measurements with long outliers or systematic measurement errors. Anomaly analysis models rely on the spatial correlation among neighbouring sensor nodes at the same point in time to assess the accuracy of their observations based on the concept of spatial continuity.

As detailed in Section 3.6, outliers in contextual time-series are observations with value attributes that differ significantly from the value attribute of nearby, spatially correlated observations which is known as the spatial continuity and justified by Tobler's law of geography (Tobler, 1970). However, the spatial continuity concept is not necessarily applicable to the real-world observations collected from large-scale, sensor node networks. For example, temperature observations collected from the sensor node network distributed around London are not necessarily spatially correlated due to a phenomenon known as the Urban Heat Islands (UHI) which violates their spatial continuity as shown in Figure 3.24. To tackle this issue, spatial partitioning (clustering) techniques were adopted. In general, spatial clustering is the process of grouping (labelling) objects based on similarity measures of their relative distance or density (Wang et al., 2017, p. 157-160). Spatial

clustering utilises spatial data partitioning models to facilitate correlation between data points by breaking up the space of interest into more representative regions for local data points. This section outlines the empirical details and outcomes of testing distance-based spatial clustering (K-Means) and density-based spatial clustering (DBSCAN) as spatial data partitioning techniques for anomaly analysis (outliers detection) in large-scale CPSs.

4.2.2.1 Dataset Modes and Details

The purpose of this section is to examine distance-based spatial clustering (K-Means) and density-based spatial clustering (DBSCAN) as spatial data partitioning techniques for outliers detection, especially long-outliers detection. Both techniques were used to break up the space of interest into more representative regions for local data points (sensor nodes). The aim is to create local groups of spatially correlated sensor nodes, then compare their observations at the same point in time to identify observations that significantly different from their neighbours as potential outliers. This approach of outlier detection does not require prior knowledge of the temporal sequence of sensor nodes observations. It compares the current or most recent observations from different nearby sensor nodes. Thus, both K-Means and DBSCAN models were tested using the Snapshot Data Model, shown in Figure 3.9, where only the most recent set of observations arrived from all available sensor nodes is considered and fitted to the models.

The number of sensor nodes and their geographical coordinates are the parameters that may change the outcome of the spatial partitioning (clustering) algorithms. The geographic distribution of all available temperature sensor nodes used in this case study based on their coordinates attributes over a real-scale map is shown in Figure 4.53.

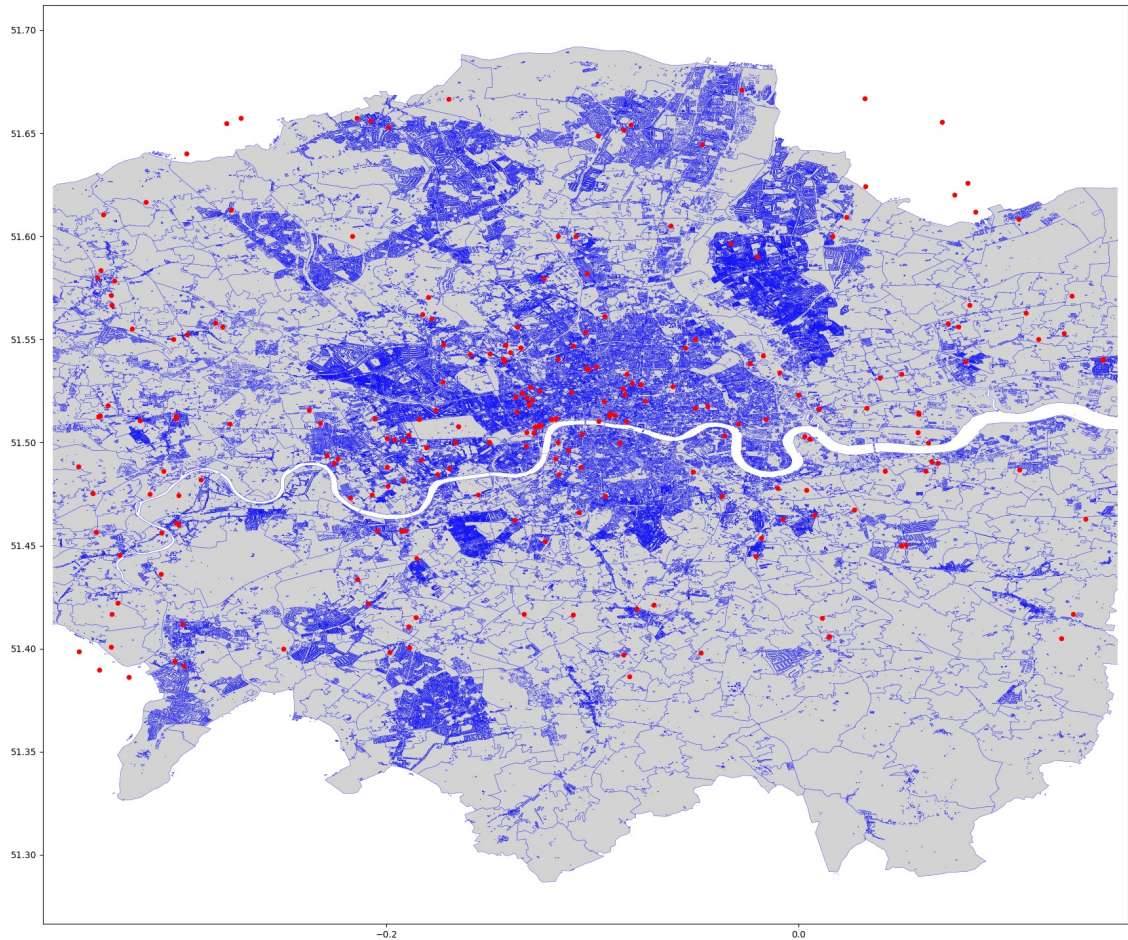


Figure 4.53: The geographic distribution of all available temperature sensor nodes over a real-scale map (London 04/2020).

4.2.2.2 Distance-Based Spatial Clustering (K-Means)

K-means clustering algorithm was applied using the "cluster.KMeans" Python package provided by Scikit-learn²⁴ (Pedregosa et al., 2011), the algorithm's main parameters are shown in Figure 4.54. The full version of the code and programming aspects of the K-means test are detailed in Appendix A, Section A.4.6.

```
KMeans(init='k-means++', n_clusters=i, n_init=10)
```

Figure 4.54: K-means main parameters by scikit-learn (Pedregosa et al., 2011).

Where "K-means++" is the augmented version of K-means with an optimised

²⁴<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

seeding technique, enhanced speed and accuracy (Arthur & Vassilvitskii, 2006), and "n_clusters" is the number of clusters K which must be provided as an input parameter to K-means model.

The optimum number of clusters k was determined using the Silhouette analysis method, which evaluates the accuracy of the clustering model and can be used to estimate the optimum number of clusters K for K-means as detailed in Section 3.6.1. Silhouette analysis was conducted and recorded against the number of clusters K for a range of tests from $K = 2$ to $K = \text{the number of sensors} - 1$. The result of the Silhouette analysis on K-means is shown in Figure 4.55, which indicates that the highest silhouette score was achieved at $k = 115$.

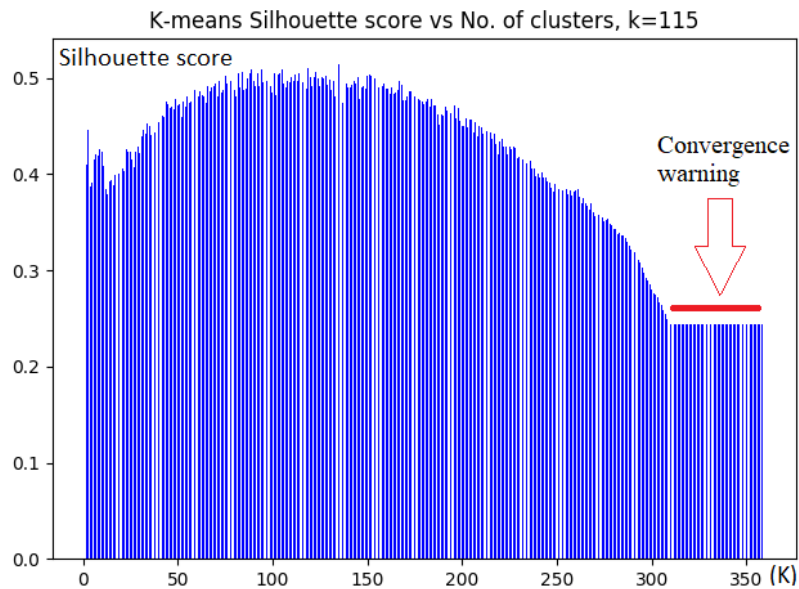


Figure 4.55: Silhouette analysis to determine K-means k, the highest score was reached at $K=115$.

The outcome of applying K-means while setting $K=115$ is visually illustration by Voronoi tessellations' approach to present spatially partitioned clusters plotted over the landscape of London, as shown in Figure 4.56. K-means was able to identify a centroid point for each cluster, where each coloured shapes is a cluster, and the blue dots are their centroids.

Applying k-means model for multiple times to the same dataset revealed that

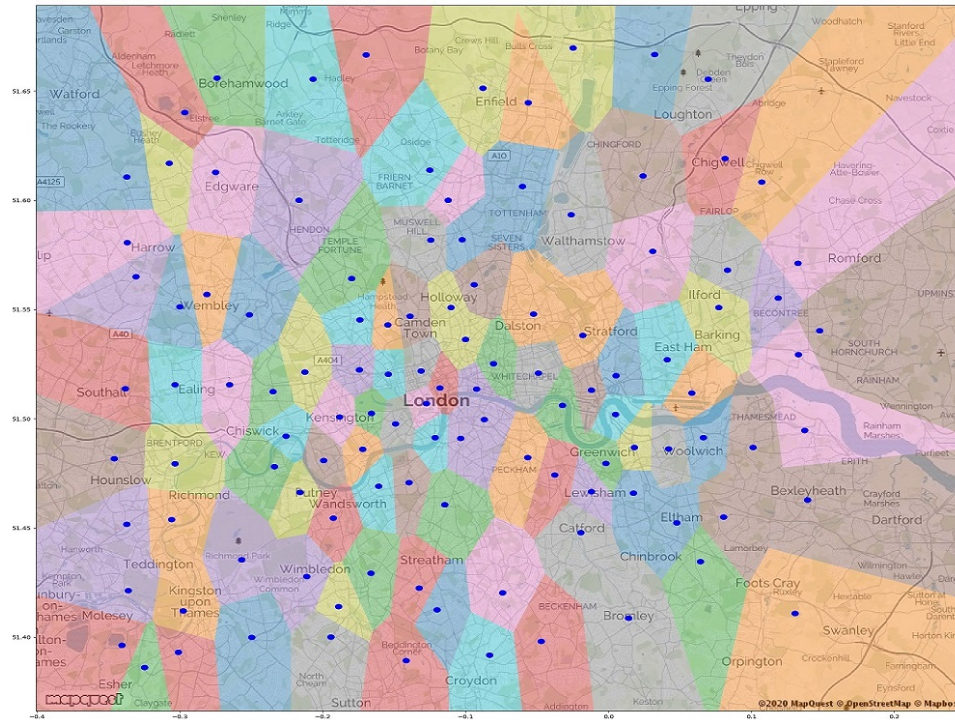


Figure 4.56: The outcome of applying K-means (K=115) represented by Voronoi tessellation approach plotted over the landscape of London, the blue dots are the centroids of the clusters.

K-means re-allocates the positions of the centroid points and creates a different set of clusters with different distribution after each test, as shown in Figure 4.57.

This behaviour of K-means model is justified by how its clustering algorithm works. It starts with an initial estimation of the location of K number of centroid points selected randomly from the dataset, where K is the pre-defined number of clusters. From this stage, K-means starts a routine of two steps:

1. Data points assignment, assigning data points to their nearest centroid point based on the squared Euclidean distance.
2. Centroids update, by recalculating and reassigning the centroids according to the mean value of the Euclidean distance of all data points assigned to that centroid's cluster.

K-means keeps iterating between these two steps until no significant changes in

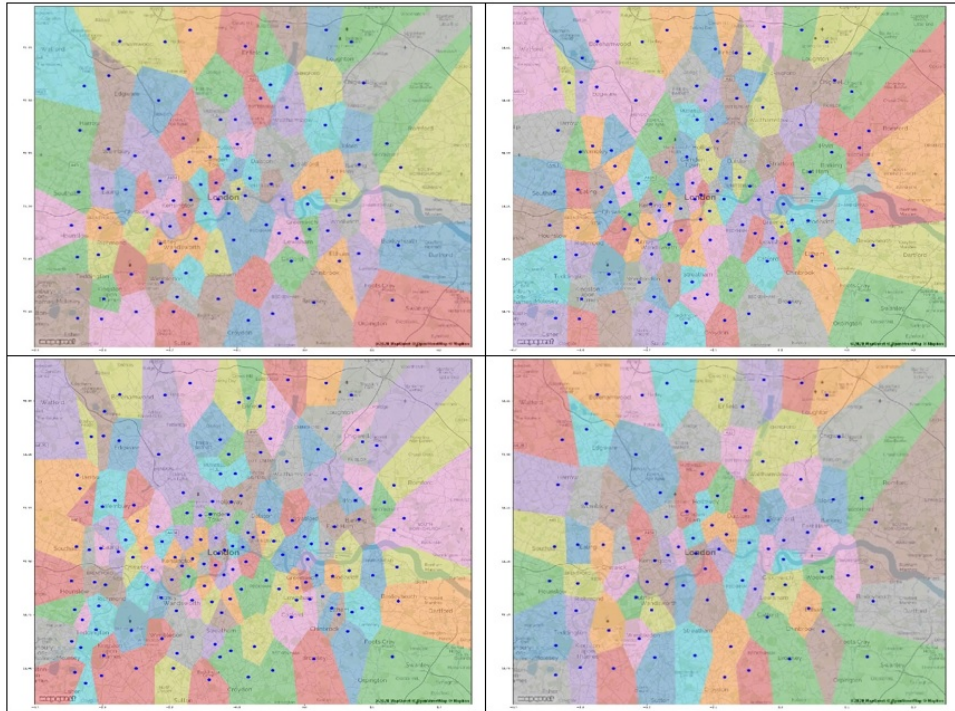


Figure 4.57: K-means model renders different outcomes after applying it many times on the same dataset.

centroid points location occur anymore, or a maximum number of tries is reached. K-means randomly select the centroid points whenever it initialises. Thus, it partitions the dataset differently each time, which shows that any calculations associated with the k-means cluster's such as the centroid threshold values must be recalculated again for all clusters after each time K-means applied.

The major drawback of using K-means as spatial data partitioning technique for outlier detection is that K-means partitions the dataset evenly without considering the minimum number of sensor nodes or the maximum relative distance allowed between sensor nodes in the same cluster. Thus, K-means may create a cluster of one sensor node or may include relatively distant sensor nodes in the same cluster which in both cases violate the spatial continuity constraints and may compromise the accuracy and validity of the outlier detection results.

Although K-means was utilised successfully in many large-scale CPSs geo-spatial partitioning applications, such as for optimising the operation of district energy

systems (Fazlollahi et al., 2014), enhancing the lifetime of wireless sensor networks (Kumar & Chaturvedi, 2014), optimising real-time traffic networks (Yang et al., 2020), street networks analysis (Goss et al., 2014) and even for bicycle sharing system analysis (Ma et al., 2019). In all these applications, K-means was used as a geospatial partitioning technique to assign data points to the nearest centroid based on the relative “Euclidean distance”. Thus, it does not consider the spherical shape of the earth and the actual distance between data points on the sphere which is known as the “Haversine distance” (Sinnott, 1984).

The performance of K-means partitioning model was evaluated using all available data points (360 sensor nodes). K-means was applied to the same dataset while using different values of K, as shown in Figure 4.58.

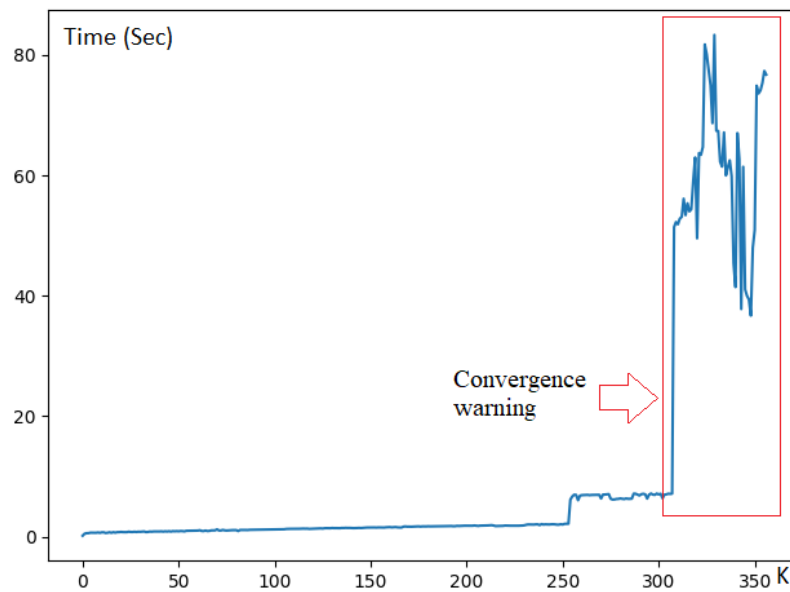


Figure 4.58: The time (Sec) required by K-means to render the clustering results applied to the same dataset while testing a range of K (1 to 360).

K-means model rendered its clustering results within 0.5 Sec in all cases, except when k reached 309, which is an unrealistic number of clusters in comparison with the total number of available data points (sensor nodes). K-means algorithm started to show a "Convergence Warning" at $K = 309$ while calculating the Silhouette analysis score, which was not possible after that point, as shown in

Figure 4.55.

4.2.2.3 Density-Based Spatial Clustering (DBSCAN)

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a clustering algorithm that identifies clusters based on the density of data points within a specified radius. Unlike K-means, DBSCAN does not require the number of clusters as an input parameter. Alternatively, MinPoints and Eps are required where MinPoints is the minimum number of data points within the radius Eps (Epsilon). DBSCAN clustering algorithm was applied using the “cluster.DBSCAN” Python package provided by Scikit-learn (Pedregosa et al., 2011), the algorithm’s main parameters are shown in Figure 4.59. The test design details are illustrated in Section 3.6.2. The full code and programming aspects of the test are detailed in Appendix A, Section A.4.6.

```
DBSCAN(eps=eps / kms_per_radian, min_samples=2, metric='haversine')
```

Figure 4.59: DBSCAN algorithm’s main parameters applied using "cluster.DBSCAN" Python package provided by Scikit-learn (Pedregosa et al., 2011).

The main challenge of applying DBSCAN is how to determine the optimum value of Epsilon (Eps). The MinPoints (min_samples) parameter was assigned to (2) to reflect the minimum number of sensor nodes required to establish a deviation comparison for outlier detection. The Silhouette analysis method was adopted to determine the optimum value of Eps based on the best clustering performance. The result of applying the Silhouette analysis to DBSCAN is shown in Figure 4.60, where the highest silhouette score was obtained at Eps = 1.66 Km.

The Haversine formula was adopted in the DBSCAN model to calculate the great-circle distance between data-points (sensor nodes) using their longitudes and latitudes attributes, as shown in Figure 4.59, where Eps was divided by the radius of the sphere (Earth) to become compatible with the used metric (Haver-

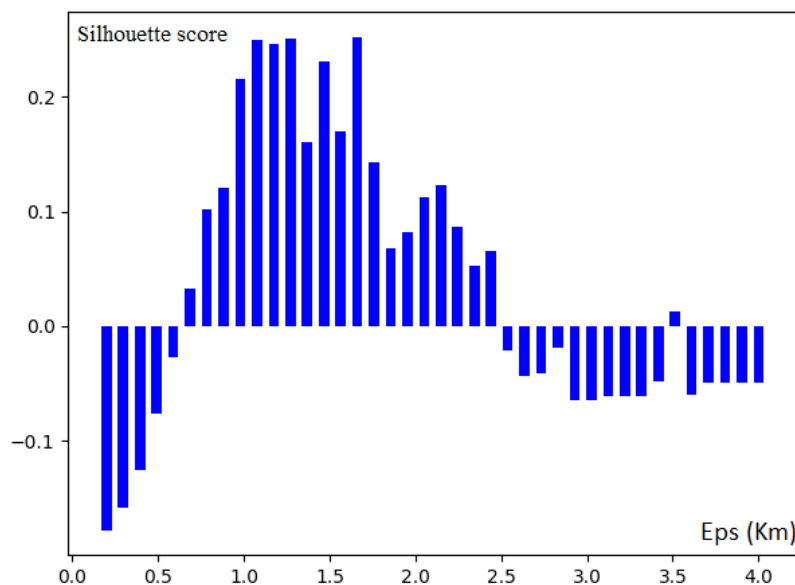


Figure 4.60: DBSCAN highest Silhouette score was obtained at Eps = 1.66 Km.

sine) (Ebrahimi et al., 2017).

DBSCAN clustering results are shown in Figure 4.61. Unlike K-means, DBSCAN clustering results were consistent, rendering fixed clusters after being applied for multi-times using the same dataset.

The results of DBSCAN clustering algorithm showed that temperature sensor nodes are not evenly distributed over the area of interest (London). DBSCAN categorised the region of interest into a “high-density” area and “low-density” area according to the available number of sensor node within the radius Eps, as shown in Figure 4.61. DBSCAN model spatially partitioned the data-points (sensor nodes) into 50 clusters, as shown in Figure 4.62.

DBSCAN model indicated that sensor nodes located in a low-density area might not fit in any cluster, as shown in Figure 4.61 (the red arrows which were added manually). Each sensor node in the low-density area may form a cluster by itself or with a distant sensor node(s), which in both cases violates the concept of spatial continuity and compromises the accuracy of the outlier detection model. Most of the distant sensor nodes were eliminated by DBSCAN, and considered as noise. Thus, the number of DBSCAN clusters is significantly lower compared with K-

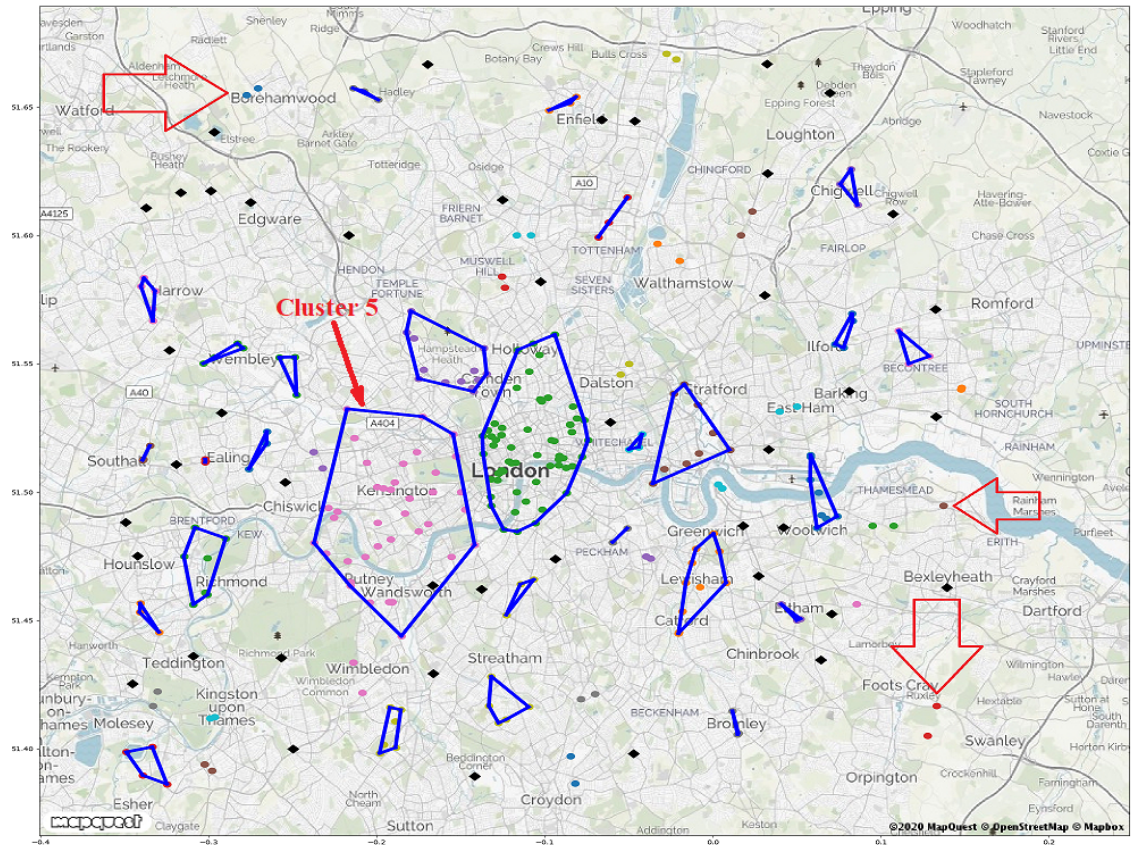


Figure 4.61: DBSCAN clustering result, the blue lines are highlighting high-density regions ($S > 3$) of sensor nodes distribution, the red arrows (added manually) are showing examples of sensor node(s) in low-density areas.

```

Epsilon: 3.707692307692308 --> silhouette score: -0.049738
Epsilon: 3.8051282051282054 --> silhouette score: -0.049738
Epsilon: 3.902564102564103 --> silhouette score: -0.049738
Epsilon: 4.0 --> silhouette score: -0.049738

Best epsilon = 1.6615384615384614

Estimated number of clusters = 50

```

Figure 4.62: DBSCAN spatially partitioned the available data-points (sensor nodes) into 50 clusters at $Eps = 1.66$.

means, which partitioned the region of interest into K number of clusters without considering the density of distribution of sensor nodes. Just like K -means, the performance of the DBSCAN model was evaluated using all available data points (360 sensor nodes). DBSCAN was able to render the clustering results in less than *0.5 Second*, in all tests, as shown in Figure 4.63.

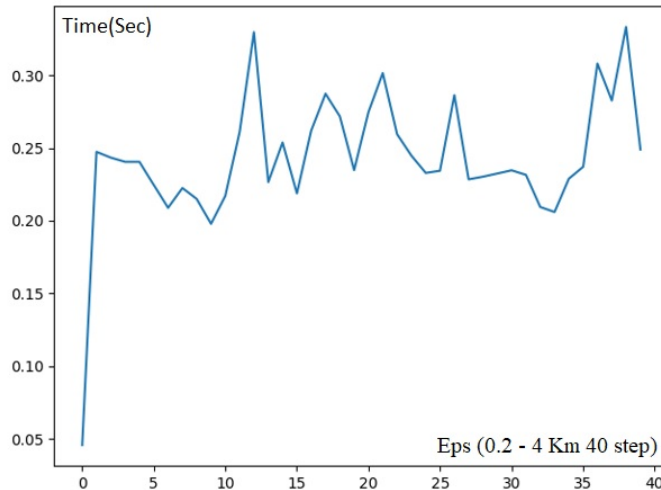


Figure 4.63: The time (Sec) required by DBSCAN to render the clustering results applied to the same dataset while testing a range of Eps (0.2 - 4 Km 40 step).

4.2.2.4 Utilising DBSCAN for Outliers Detection

Anomaly analysis models were investigated as accuracy assessment models via detecting outliers, especially long-outliers, or systematic errors in sensor nodes measurements. Anomaly analysis models rely on the spatial correlation among neighbouring sensor nodes to assess the accuracy of observations based on the concept of spatial continuity. Applying anomaly analysis directly to the dataset of real-world ambient temperature sensor nodes was not possible because of the effect of the phenomena of the Urban Heat Islands, which causes unexpected changes in the value attribute among sensor nodes and violates their spatial continuity. Therefore, spatial partitioning techniques were used to divide the region of interest into smaller and more representative domains for sensor nodes using clustering algorithms.

The evaluation of K-means as spatial data partitioning technique revealed that it partitions the region of interest without considering the minimum number of sensor nodes or the maximum relative distance allowed between sensor nodes in the same cluster. Thus, K-means may create a cluster of one sensor node or may include relatively distant sensor nodes in the same cluster which in both cases

violate the spatial continuity constraints and may compromise the accuracy and validity of the outlier detection results.

Alternatively, DBSCAN identifies clusters based on the density of data points within a specified radius. Unlike K-means, DBSCAN clustering results were consistent, rendering fixed clusters after being applied multiple times on the same dataset. It categorised the region of interest into “high-density” areas and “low-density” areas according to the number of sensor node within the radius Eps. DBSCAN model showed that sensor nodes located in a low-density area might not fit in any cluster and considered most of the distant sensor nodes as noise and eliminated them. Therefore, in this case study, DBSCAN was selected as the most suitable spatial portioning model for accuracy outliers detection in sensor nodes’ measurements in large-scale CPSs.

Technically, DBSCAN labels (defines) each of its clusters with a numeric value starting from 0 up to the total number of clusters. Thus, each sensor node would receive a new attribute that defines its cluster. The result of applying DBSCAN on the real-world dataset is shown in Figure 4.61, and the number of sensor nodes in each cluster is shown in Figure 4.64.

	Cluster Label	No. of Sensors
▶	-1	43
	0	10
	1	89
	2	2
	3	3
	4	5
	5	47
	6	2
	⋮	⋮
	48	2
	49	2

Figure 4.64: DBSCAN clusters labels and the number of sensor nodes in each cluster.

The first row in Figure 4.64 with cluster label (-1) indicates that 43 sensor nodes were not fitted in any cluster and denoted by DBSCAN as noise and eliminated

from the rest of the clusters²⁵. Using cluster-5 as a case study, it consists of 47 sensor nodes, as shown in Figure 4.64, and the spatial distribution of these sensors is shown in Figure 4.61. Since these sensors belong to the same cluster, it is safe to assume that these sensors' value attributes are correlated. Therefore, it is possible to compare these sensor nodes observations and specify observations with value attributes that exceed the deviation threshold as outliers. The first step is to determine the cluster's threshold value at a particular point in time t , in this case, the most recent set of observations represented by the last duty-cycle in the real-world dataset. Selecting the most recent duty-cycle of the real-world dataset which starts at '2020-03-15 23:49:59' and ends by '2020-03-15 23:59:59' revealed that only eight sensor nodes of the 47 had streamed observations during that duty-cycle, as shown in Figure 4.65.

	SensorID	ServerDate	Latitude	Longitude	AmbientTemperature
▶	0g8s4q56	2020-03-15 23:49:22	51.493839	-0.22882	5.97
	cmr50pbz	2020-03-15 23:49:22	51.50378	-0.18928	5.98
	d60c89mp	2020-03-15 23:49:24	51.457218	-0.19278	5.99
	d8srmzbn	2020-03-15 23:49:23	51.500938	-0.19175	5.92
	eprhpmw4	2020-03-15 23:49:22	51.511162	-0.18426	5.98
	g0vs2y25	2020-03-15 23:49:23	51.5	-0.16667	5.92
	j5p57q4s	2020-03-15 23:49:20	51.515579	-0.1763	5.96
	yrvwx4kc	2020-03-15 23:49:20	51.457191	-0.19089	3.33

Figure 4.65: The eight of the 47 sensor nodes in cluster-5 that streamed observations between '2020-03-15 23:49:59' and '2020-03-15 23:59:59' of the real-world dataset.

The next step is to calculate the cluster threshold (correlation threshold) by determining which is the most common value of observations within the clusters' sensor nodes (the eight sensor nodes) using a proximity function. The proximity function rounds the value attribute of all observations in the cluster to the first digit and group (aggregate) these attributes to define the most common observation value. Applying the proximity function showed that the threshold value of clusters-5 was 6C° at that particular duty-cycle, as shown in Figure 4.66.

The next step is to calculate the Correlation Error, which is the absolute difference

²⁵<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html>

	SensorID	ServerDate	Latitude	Longitude	Ambient_Temperature	Approximately_Function	Correlation_error	Status
▶	Og8s4q56	2020-03-15 23:49:22	51.493839	-0.22882	5.97	6.0	0.03	Pass
	cmr50pbz	2020-03-15 23:49:22	51.50378	-0.18928	5.98	6.0	0.02	Pass
	d60c89mp	2020-03-15 23:49:24	51.457218	-0.19278	5.99	6.0	0.01	Pass
	d8srmzbc	2020-03-15 23:49:23	51.500938	-0.19175	5.92	6.0	0.08	Pass
	eprhpmw4	2020-03-15 23:49:22	51.511162	-0.18426	5.98	6.0	0.02	Pass
	g0vs2y25	2020-03-15 23:49:23	51.5	-0.16667	5.92	6.0	0.08	Pass
	j5p57q4s	2020-03-15 23:49:20	51.515579	-0.1763	5.96	6.0	0.04	Pass
	yrvwx4kc	2020-03-15 23:49:20	51.457191	-0.19089	3.33	6.0	2.67	Outlier detected

Figure 4.66: Outlier detection in sensor nodes observations using anomaly analysis and spatial partitioning techniques.

between sensor nodes observations' values and the cluster-threshold value. Finally, to compare the correlation error with the deviation threshold determined in Section 4.2.1.6, observations with a correlation error that is larger than the deviation threshold are categorised as outliers, as shown in Figure 4.66.

4.2.3 Timestamp Analysis (Temporal Consistency)

In order to detect timeliness, completeness and temporal-consistency data quality issues in sensor nodes' data stream, time-series periodicity analysis was utilised to estimate the duty-cycle and the threshold-interval of each sensor node in the system. Sensor nodes' duty-cycles were estimated since there was no practical way to determine the exact duty-cycle separately from the offset coefficient error e associated with each observation and caused by many effects that delay observations deliver to their destinations. The duty-cycles and the threshold-intervals of each sensor node were used as the periodicity indicators to determine whether an observation is missing or delayed using an assessment rule engine. The rule engine was developed using SQL and embedded at the second layer of the data quality management system, the MySQL database, as shown in Figure 3.8, to check the temporal consistency of observations when they arrive at the database instantly.

This section is to describe the technical details and results of applying the periodicity analysis approach proposed in Section 3.7 to evaluate the temporal consistency

of sensor nodes observations using both the ideal and the real-world datasets described in Section 4.2.1.1, as follows:

- Estimating the duty-cycle for each sensor node by calculating the shortest interval among all sequential observations in the time-series of the associated sensor. Technically, this was implemented using a data structure that combines the timestamp of the previous observations with the data row of the new observation for each record, as shown in Figure 4.67 using a database trigger embedded in the observations collection tables, as illustrated in Appendix A, Section A.3.1.

	ID	ServerDate	SensorID	Data	MessageDate	MessageDate_N
▶	66545	2020-03-01 00:11:35	493372	6.9	2020-03-01 00:09:26	2020-03-01 00:19:26
	66551	2020-03-01 00:21:33	493372	6.8	2020-03-01 00:19:26	2020-03-01 00:39:25
	66557	2020-03-01 00:42:27	493372	6.9	2020-03-01 00:39:25	2020-03-01 00:59:23
	66559	2020-03-01 01:02:05	493372	6.9	2020-03-01 00:59:23	2020-03-01 01:29:22
	66565	2020-03-01 01:31:44	493372	6.9	2020-03-01 01:29:22	2020-03-01 01:49:22
	66571	2020-03-01 01:51:21	493372	6.8	2020-03-01 01:49:22	2020-03-01 02:09:20
	66576	2020-03-01 02:11:41	493372	6.8	2020-03-01 02:09:20	2020-03-01 02:39:18
	66586	2020-03-01 02:41:37	493372	6.8	2020-03-01 02:39:18	2020-03-01 02:59:17
	66589	2020-03-01 03:00:51	493372	6.8	2020-03-01 02:59:17	2020-03-01 03:19:16

Figure 4.67: The data stream as a two-dimensional array of C_t, C_{t-1} observations to calculate the interval between every two sequential observations.

The shortest interval calculated by:

$t_{dc} = \min(C_t(\text{MessageDate}_n) - C_{t-1}(\text{MessageDate}))$ was considered as the duty-cycle of the related sensor node (t_{dc}) at the server-side associated with the offset coefficient error e and $t_{dc} > 0$.

- Aggregating and ranking all intervals according to their recurrence for each sensor node. The interval with the highest recurrence score is the Threshold Interval t_{Ths} for the associated sensor node and used as a reference to define the temporal consistency status of all observations streamed from that sensor, as shown in Figure 4.68.
- The duty-cycle (t_{dc}) and the threshold-interval t_{Ths} of each sensor node were

	SensorID	Interval_threshold	The_proximit_Interval_threshold	Recurrence_score
▶	493372	9.9833	10	445
	493372	10.0000	10	371
	493372	19.9833	20	132
	493372	19.9667	20	73
	493372	20.0000	20	45
	493372	29.9833	30	16
	493372	29.9667	30	14

Figure 4.68: Aggregating observations intervals to determine the Threshold Interval t_{Ths} for each sensor node.

incorporated by the rule engine to check the temporal consistency of their observations. The rule engine calculates the (t_{dc}) and t_{Ths} for each sensor node and checks their observations upon their arrival to the database for temporal consistency constraints based on a pre-set policies. The rule engine consists of a sequence of logical expressions which combined with (t_{dc}) and t_{Ths} determine if a particular observation is delayed, missed or temporally fit for the purpose of use. The policies of the rule engine are listed in Section 3.7, while Figure 4.69 provides some insights on the logic behind the rule engine. The full code is listed in Appendix A, Section A.3.2.

```

WHEN
  (((TIMESTAMPDIFF(SECOND,
    `HW_Dataset_Thingful`.`ServerDate`,
    `HW_Dataset_Thingful`.`ServerLastUpdate`) / 60) / `HW_Dataset_Thingful_th`.`Interval_threshold`) > 1)
  AND (((TIMESTAMPDIFF(SECOND,
    `HW_Dataset_Thingful`.`ServerDate`,
    `HW_Dataset_Thingful`.`ServerLastUpdate`) / 60) / `HW_Dataset_Thingful_th`.`Interval_threshold`) <= 2))
THEN
  'Timeliness'
WHEN
  (((TIMESTAMPDIFF(SECOND,
    `HW_Dataset_Thingful`.`ServerDate`,
    `HW_Dataset_Thingful`.`ServerLastUpdate`) / 60) / `HW_Dataset_Thingful_th`.`Interval_threshold`) >= 2)

```

Figure 4.69: An example of the logical expressions (policies) used inside the rule engine.

The results of applying the temporal timestamp analysis approach using the periodicity analysis and the SQL rule engine on the ideal dataset are shown in Figure 4.70, and on the real-world data set are shown in Figure 4.71.

SensorID	MessageDate	MessageDate_N	interval	Interval_threshold	Consistency_status
493372	2020-03-01 06:19:11	2020-03-01 06:29:10	9.9833	10	Good
493372	2020-03-01 06:29:10	2020-03-01 06:39:10	10.0000	10	Good
493372	2020-03-01 06:39:10	2020-03-01 06:49:10	10.0000	10	Good
493372	2020-03-01 06:49:10	2020-03-01 07:09:08	19.9667	10	Timeliness
493372	2020-03-01 07:09:08	2020-03-01 07:29:06	19.9667	10	Timeliness
493372	2020-03-01 07:29:06	2020-03-01 07:59:04	29.9667	10	Missing observation
493372	2020-03-01 07:59:04	2020-03-01 08:19:03	19.9833	10	Timeliness
493372	2020-03-01 08:19:03	2020-03-01 08:39:02	19.9833	10	Timeliness

Figure 4.70: The result of applying the timestamp analysis approach for temporal consistency assessment on the ideal dataset.

SensorID	Latitude	Longitude	MessageDate	MessageDate_N	Interval	I_threshold	Consistency_status
jcw5m701	51.414749	0.01167	2020-03-13 13:51:07	2020-03-13 14:01:13	10.1000	10	Timeliness
jcw5m701	51.414749	0.01167	2020-03-13 13:31:40	2020-03-13 13:41:18	9.6333	10	Good
jcw5m701	51.414749	0.01167	2020-03-13 13:12:07	2020-03-13 13:22:01	9.9000	10	Good
jcw5m701	51.414749	0.01167	2020-03-13 13:01:31	2020-03-13 13:01:31	0.0000	10	Duplicated observation
jcw5m701	51.414749	0.01167	2020-03-13 13:01:31	2020-03-13 16:31:55	210.4000	10	Long outlier
jcw5m701	51.414749	0.01167	2020-03-13 12:51:08	2020-03-13 12:51:08	0.0000	10	Duplicated observation

Figure 4.71: The result of applying the timestamp analysis approach for temporal consistency assessment on the real-world dataset.

The timestamp analysis and time-series periodicity analysis methods were nearly 100% accurate in identifying timeliness, completeness and consistency data quality issues in sensor nodes data stream for both the ideal and the real-world datasets. Due to the limited number of the policies applied to check the temporal consistency of the sensor nodes observations, the rule engine rendered its assessment results instantly at the arrival of the observation to the database. The timestamp analysis was implemented using a rule engine to provide a level of automation to the process since rule engines can adapt with data stream changes, and can be configured by adding or revoking policies according to applications' requirements.

4.3 Offline-Mode Data Quality Assessment

The data quality assessment unit is the core component of the data quality management system proposed in this research. The data quality assessment unit consists of the:

Online unit covered in Section 4.2 which describes the components of the data

quality management system responsible for detecting data quality issues associated with errors in sensor nodes' measurements using predictive analysis, outlier analysis techniques and timestamp analysis for temporal consistency in real-time. **Offline unit** describes the components of the data quality management system responsible for detecting sensor nodes' hardware failures associated with long segmental outliers which last for a relatively long time and change the pattern of time-series using time-series clustering and timestamp analysis for spatial consistency. This section presents the results of the empirical assessment of the offline components of the data quality assessment unit. The full structure of the offline data quality assessment unit is shown in Figure 3.11. All of the time-series algorithms and techniques used within the context of this case study are shown in Figure 4.72.

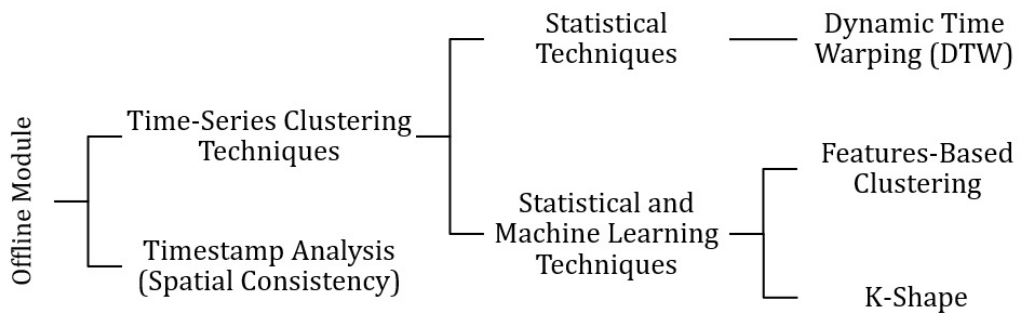


Figure 4.72: The algorithms and techniques adopted and evaluated within the context of the offline unit (module) of the data quality assessment.

4.3.1 Time-Series Clustering Models

As detailed in section 3.8, sensor nodes' hardware failure detection model is designed to detect long segmental outliers in sensor nodes' time-series as indicator of sensor nodes hardware failure. Long segmental outliers associated with sensor nodes' failures are categorised according to their faulty behaviour into continuous, abrupt and incipient faults (Sailhan et al., 2010). Sensor nodes' observations collected from the real-world, large-scale sensor node network were utilised to test

the ability of time-series clustering techniques to detect continuous (halting), and abrupt (emerging) long-outliers. Thus these types of long-outliers were detected in some time-series of the large-scale datasets. The ideal dataset collected from the local sensor node network was used to test the ability of the time-series clustering techniques to detect incipient faults with consistent offset long-outliers, as follows:

4.3.1.1 Datasets

The effectiveness of the proposed time-series clustering methods introduced in Section 3.8 is empirically evaluated using the time-series collected from both the ideal and the real-world sensor node networks, as follows:

4.3.1.1.1 The Ideal Dataset

The local dataset, described in Section 4.1.2.1, consists of four time-series collected from real-world, high-quality sensor nodes deployed at the University of East London. One of the sensor nodes was deployed indoors and the other three outdoors. This dataset was used to test the ability of the time-series clustering techniques to detect incipient faults with consistent offset long-outlier. Thus the indoor sensor node, in this case, represented a sensor with incipient fault. Since all the local sensor nodes were deployed in a relatively small geographical area, their time-series show significant similarity in the shape of their trend. However, they show some differences in the value attribute, especially with the indoor sensor which streamed a time-series with a consistent offset of 10-15 C^o from the other outdoors sensors, as shown in Figure 3.23. Furthermore, since this dataset is a well-known, high-quality dataset that has no missing values or outliers, it was used to test the time series clustering techniques before applying them to the real-world dataset.

4.3.1.1.2 The Real-World Dataset

The large-scale dataset consists of more than 200 time-series collected from real-

world sensors distributed around London. This dataset will be utilised to test the ability of time-series clustering techniques to detect continuous (halting), and abrupt (emerging) long-outliers. Thus these types of long-outliers were detected in some time-series of the large-scale datasets. The details of the real-world dataset are illustrated in Section 4.1.2.2. The accuracy and performance of the time-series clustering methods examined in this research were evaluated based on their ability to identify time-series with long-segmental outliers and the time required to render the clustering results. The sliding window data model described in Section 3.4.3 was applied to the real-world dataset using two different lengths of time-windows: The first: is a seven days' time window. The second is a two-days' time window to evaluate the accuracy and performance of time-series clustering techniques in comparison with the seven days' time window.

4.3.1.2 Dynamic Time Warping (DTW) and K-Shape

Dynamic Time Warping (DTW) and K-Shape time series clustering techniques were implemented using the Python package *tslearn.clustering* provided by Scikit-learn (Pedregosa et al., 2011). The main technical steps required to fit all available time-series from all sensor nodes as a three-dimensional data array to the DTW and K-Shape models are illustrated in the process flowchart diagrams shown in Figure 4.73. The programming aspects of both tests are detailed in Appendix A, Section A.4.7.

The outcome of applying the Dynamic Time Warping (DTW) and K-Shape time series clustering techniques to the ideal dataset is shown in Figure 4.74.

Both DTW and K-Shape techniques were successfully able to identify the time-series of the indoors sensor node (incipient faults pattern) from other time-series of the outdoors sensor nodes. This result is significant because both DTW and K-Shape are shape-based time series clustering techniques and all time-series used

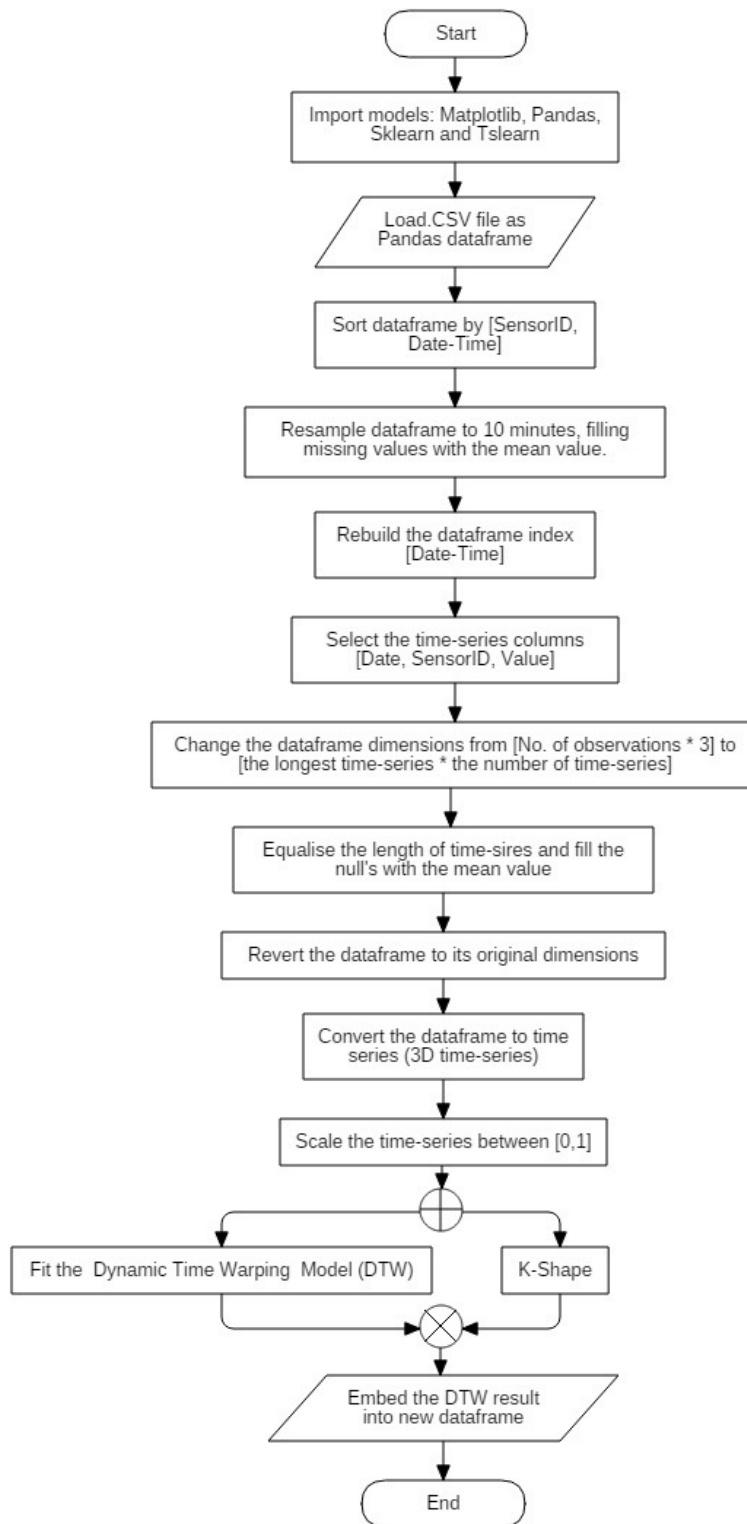


Figure 4.73: The process diagram of the technical steps implemented to fit all available time-series as a 3D array into the Dynamic Time Warping (DTW) and K-Shape time-series clustering models.

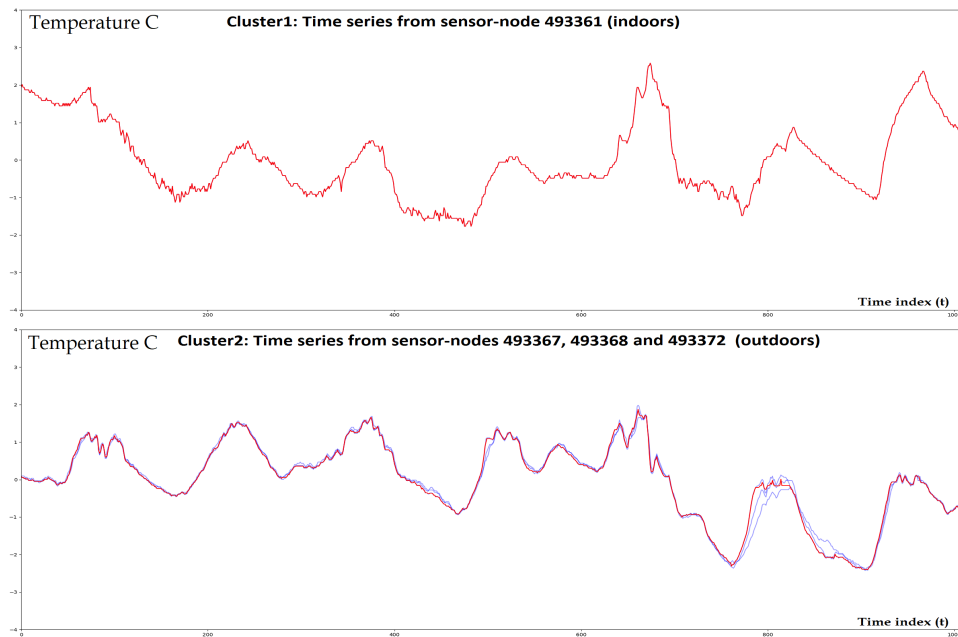


Figure 4.74: DTW and K-Shape were able to successfully differentiate the indoors time series (incipient faults pattern) from other outdoors time-series.

in this test have a significantly similar pattern, as shown in Figure 3.23. The red graph line in Figure 4.74 is the centroid time-series of the cluster, while the blue graph lines are the other time-series in the cluster.

The time-series used in the second test were collected from the large-scale, 274 sensor node network. The dataset of this test is much larger than the dataset of the local sensor node network. Both DTW and K-Shape rendered nearly identical clustering results when applied to the seven days' time-series, as shown in Figure 4.75 and in Figure 4.76.

Both DTW and K-Shape were able to separate the time-series with long continuous (halting) and abrupt (emerging) segmental outliers from the other time-series that exhibit typical variation in the trend and seasonality when applied to the seven days' time window. The y-axes in Figure 4.75 and Figure 4.76 do not reflect the actual value attribute of observations since all time-series were normalised before being fitted to the time-series clustering models.

Applying DTW and K-Shape to the two-day dataset showed that DTW is more

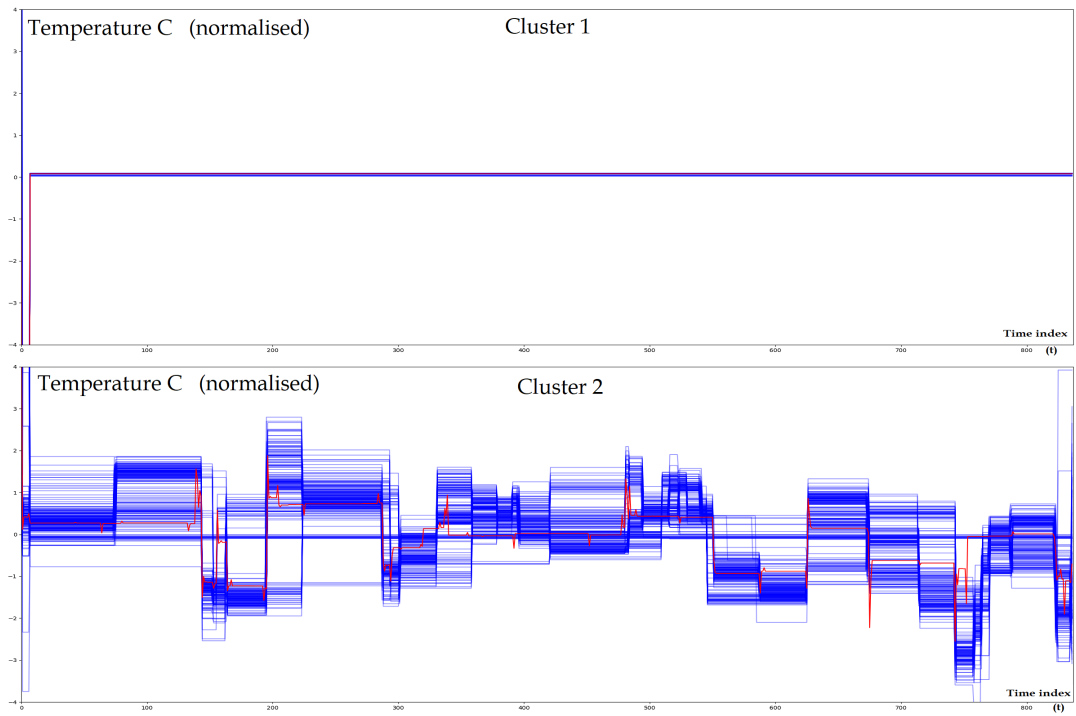


Figure 4.75: DTW successfully separated time-series with the long segmental outliers from other (typical) time-series when applied to 7-days window real-world dataset.

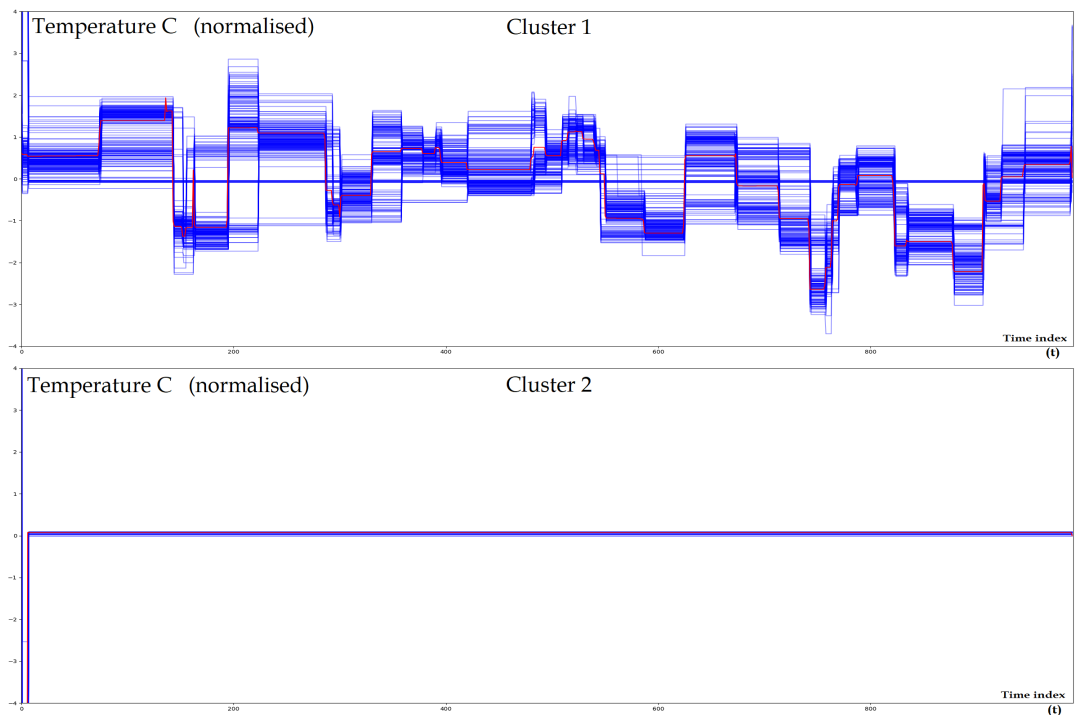


Figure 4.76: K-Shape successfully separated time-series with the long segmental outliers from other (typical) time-series when applied to 7-days window real-world dataset.

sensitive to the length of the time-window of the clustered time-series compared to K-Shape. The ability of DTW to differentiate the faulty from other (typical) time-series was more significantly affected comparing to K-Shape, as shown in Figure 4.77 and Figure 4.78.

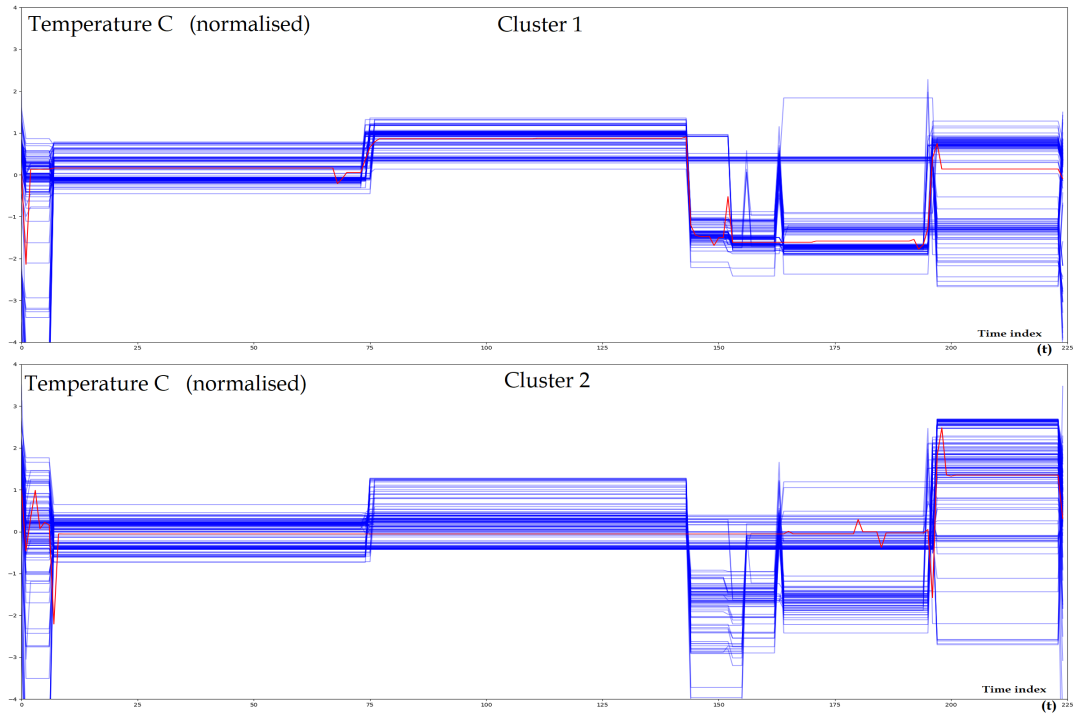


Figure 4.77: DTW is not able to differentiate time-series with long segmental outliers from other typical time-series after it was applied to a shorter two days' time-window of real-world time-series.

Figures 4.77 and 4.78 illustrate that K-Shape is more able to maintain its clustering accuracy when applied to a relatively shorter time-series comparing to DTW. In general, both shape-based time-series clustering techniques require relatively long time-series to enhance their clustering results, especially the DTW. Both techniques were able to differentiate time-series that showed the patterns of the continuous and abrupt sensor node long-segmental outliers with 100% accurate detection ratio when applied to seven days, or longer time-series, as shown in Figure 4.75 and Figure 4.76.

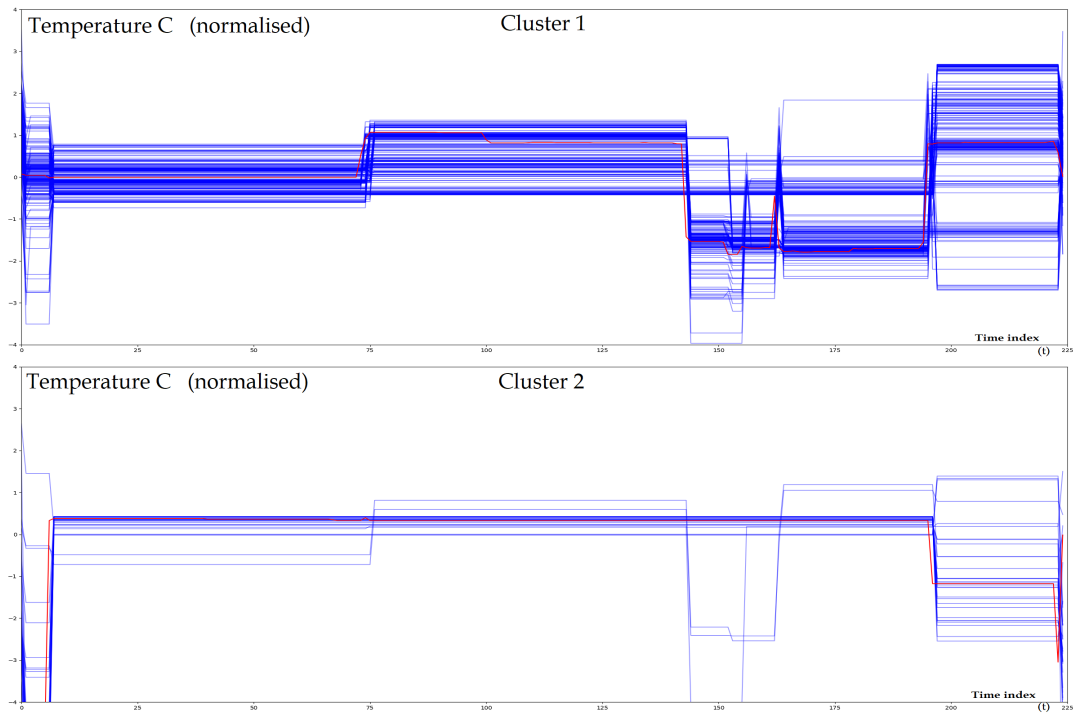


Figure 4.78: K-Shape is less able to differentiate time-series with long segmental outliers from other typical time-series after it was applied to a shorter, two days' time-window, of real-world time-series.

4.3.1.3 Characteristics-Based Time-Series Clustering

The characteristics (features)-based time-series clustering technique was investigated as an alternative to DTW and K-Shape. The aim was to find a higher performance time series clustering technique that can achieve accurate clustering results even when applied to a relatively short time-series. Using the dataset collected from the local sensor node network as a test benchmark, the features-based time-series clustering technique is implemented using the Python *tsfresh*²⁶ package provided by (Christ et al., 2018) which successfully separated the time-series (incipient faults pattern) of the indoors sensor node from the rest of time-series, as shown in Figure 4.79.

The colours of the graph lines in Figure 4.79 were set automatically to indicate that different coloured time-series belong to different clusters. The features-based

²⁶To download: <https://tsfresh.readthedocs.io/en/latest/>

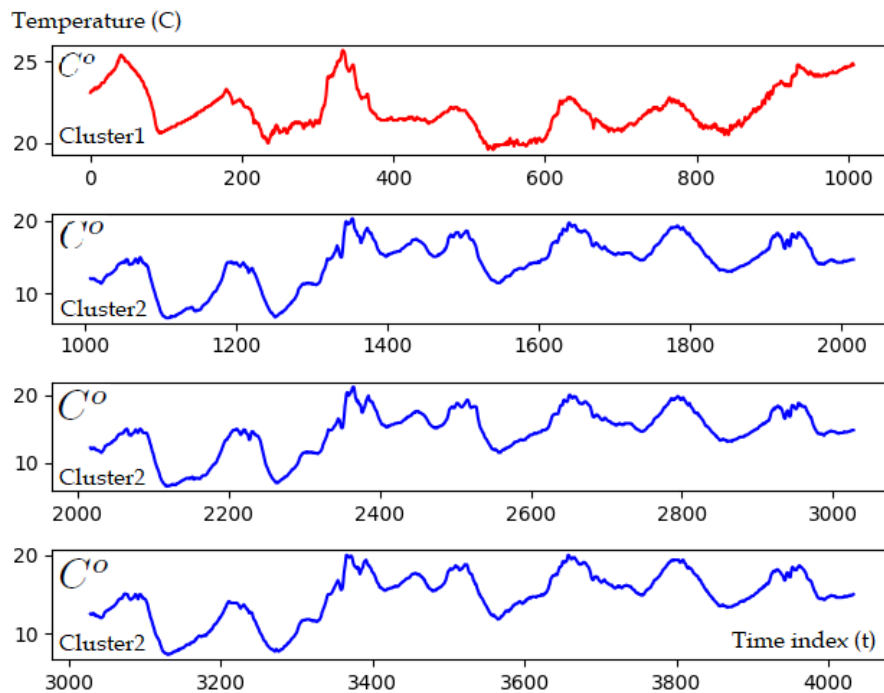


Figure 4.79: The feature-based time-series clustering method was able to differentiate the indoors, incipient faults time series from the other time-series of the ideal dataset.

time-series clustering model relies on using arbitrary clustering algorithms such as K-means to cluster the set of features extracted from the examined time-series. The selected features may vary from application to another based on the characteristics of the time-series chosen to be used as clustering reference. The *Tsfresh* package supports more than 200 different features to be extracted from time-series. In this case study, the "absolute sum of changes" was the main parameter used and fitted to the k-means clustering model, to detect continuous (stuck at) faults of sensors time-series that show no or minimal variation in their observations value attributes.

The technical aspects required to fit all available time-series to the features-based time-series clustering models are illustrated in the UML flowchart diagrams shown in Figure 4.80. The programming aspects of this case-study are detailed in Appendix A, Section A.4.8.

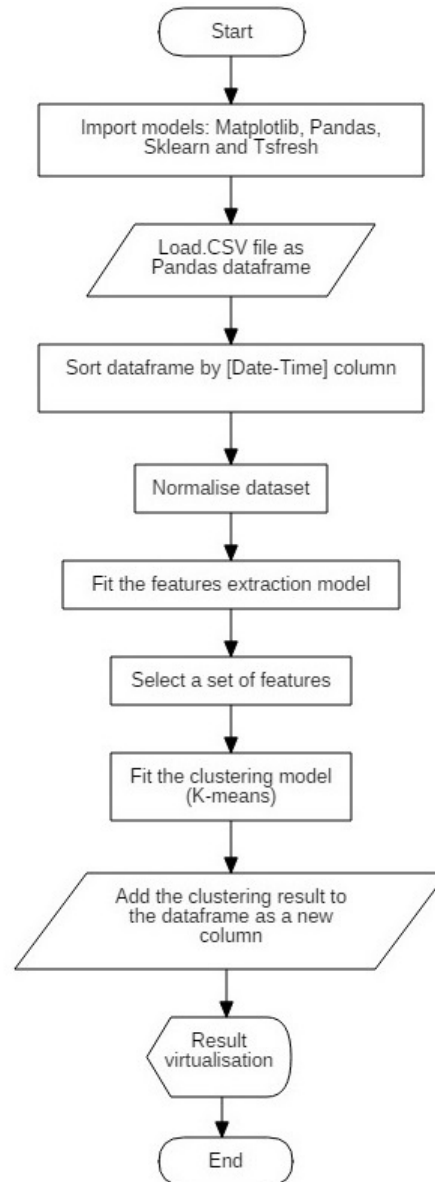


Figure 4.80: The process diagram of the technical steps implemented to fit all available time-series to the features-based time-series clustering model.

The outcome of applying the feature-based time-series clustering technique to the time-series of the large-scale sensor node network is shown in Figure 4.81 and in Figure 4.82. The colours of the graph lines were automatically set to indicate to which cluster each time-series belong.

The feature-based time-series clustering technique successfully categorised time-series with long segmental outliers even when it was applied to a relatively short time-series (two days' time window), as shown in Figure 4.82.

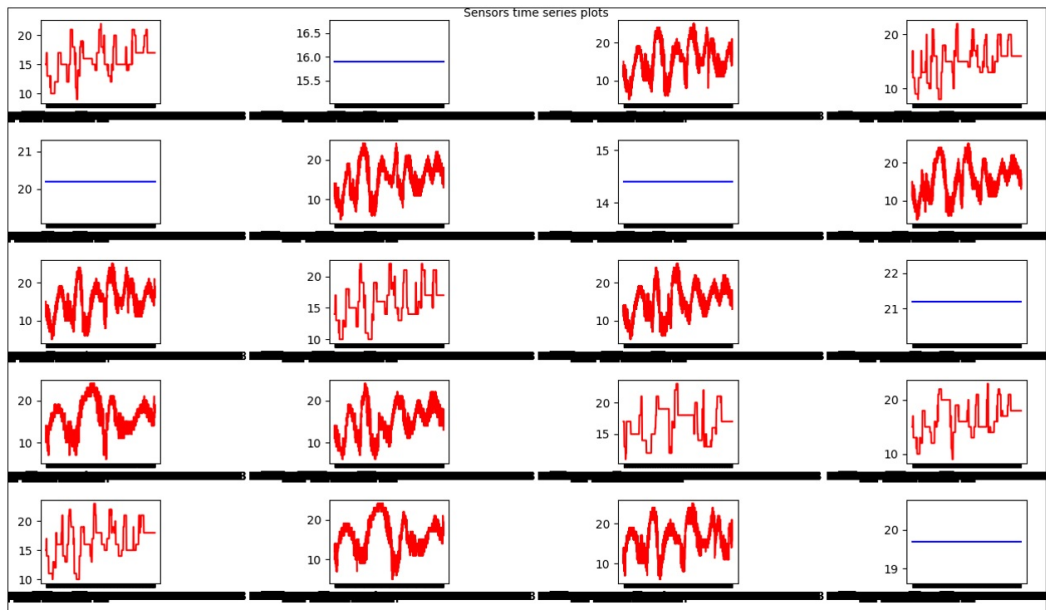


Figure 4.81: The feature-based time-series clustering technique successfully differentiated time-series with long segmental outliers when applied to the real-world (seven days' time window). The Graph lines with the same colour belong to the same cluster.

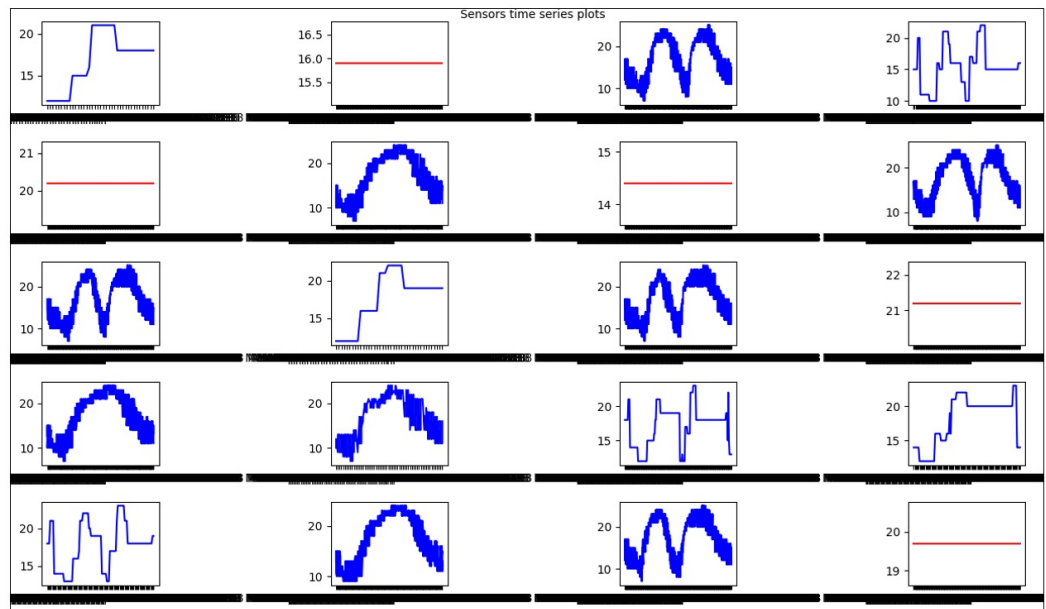


Figure 4.82: The feature-based time-series clustering technique successfully differentiated time-series with long segmental outliers even when applied to relatively short two days' time-series window of the real-world dataset. The Graph lines with the same colour belong to the same cluster.

Since all the used time-series clustering techniques were applied to the same dataset, it was possible to evaluate each of these methods' performance based on the time spent to render the clustering results. DTW required a significant amount of time to render the results at 360 Seconds compared to the feature-based and K-shape time series clustering techniques at around 30 Seconds when applied to seven day's time-window. It is essential to highlight that these results may vary according to the number and the type of the extracted features and the selected clustering algorithm. Although, DTW demanded more time than K-Shape to render the clustering results. It seems that the K-Shape Python package was able to manage the processing resources of the CPU cores more efficiently comparing to DTW, as shown in Figure 4.83.



Figure 4.83: The performance of the CPU four cores and the time required to perform the same task by DTW comparing to K-Shape, each graph line represents the performance of a single CPU core.

4.3.2 Timestamp Analysis Model (Spatial Attributes Consistency)

Spatial and temporal analysis mechanisms were investigated to identify mismatches in sensor nodes' spatial contextual attributes. This approach can also be utilised to identify networks blackouts or gateway modules failures. The assumption behind this approach is that it is possible to identify sensor nodes which are connected to the same gateway module based on the spatial and temporal attributes associated with their observations. If a sensor node shows a significant deviation in its geographical location comparing to other sensor nodes connected

to the same gateway than the coordinates of that sensor node are potentially inaccurate. Moreover, if all sensor nodes connected to the same gateway stop streaming data simultaneously, that indicates a gateway or network failure, as detailed in Section 3.9. The spatial consistency timestamp analysis model was developed using SQL as a built-in component inside the database. The timestamp analysis model was applied to the ideal dataset collected from the high-quality sensor nodes network of the University of East London, and to the large-scale dataset collected from the real-world sensor node networks distributed around London. Both datasets have the same data structure. A time-window from the ideal dataset presenting observation from all available (four) sensor nodes is shown in Figure 4.84.

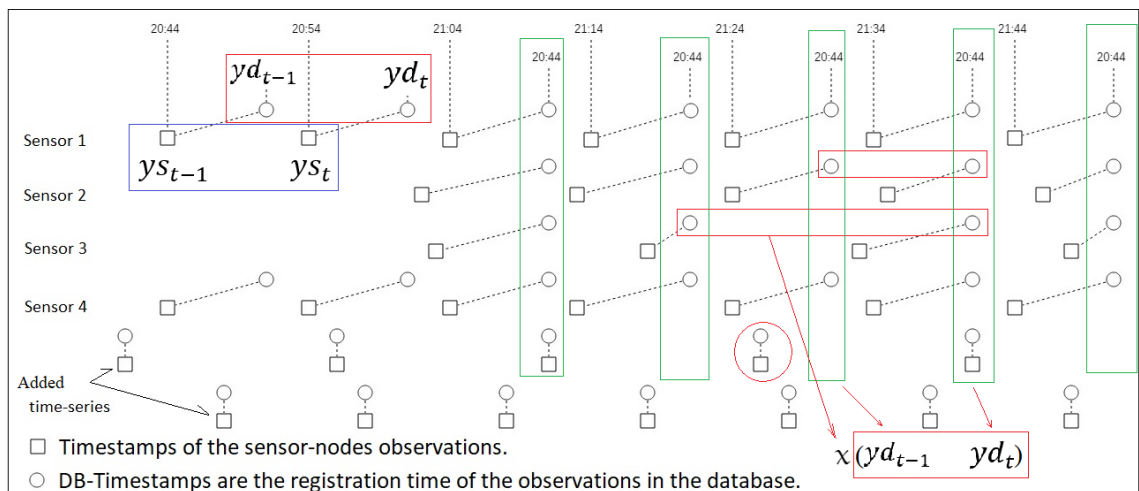


Figure 4.84: A time-window from the ideal dataset presenting observation from all available (four) sensor nodes.

The timestamp analysis model is based on the assumption that sensor nodes are highly likely to be connected to the same gateway module if their observations regularly exhibit correlated database timestamps, as shown in Figure 4.84 (the Green frames) and they retain the same gateway duty-cycle or their GCD as shown in Figure 4.84 (the Red frames).

Sensor nodes duty-cycles S_{dc} were calculated using the shortest interval of $y_{s_t} - y_{s_{t-1}}$ for each sensor using an aggregation function, where $S_{dc} = \min(y_{s_t} - y_{s_{t-1}})$,

as shown in Figure 4.84, (the Blue frame).

The gateways duty-cycles were estimated as the greatest common divider (GCD) of all database intervals $yd_t - yd_{t-1}$ for each sensor node, where $G_{dc} = x(yd_t - yd_{t-1})$ and $S_{dc} > 0$, as shown in Figure 4.84. Applying $yd_t - yd_{t-1}$ rendered multi-values of G_{dc} for each sensor node reflecting any missing or delayed observations, as shown in Figure 4.85.

$G_{dc} = x(yd_t - yd_{t-1})$				$S_{dc} = x(ys_t - ys_{t-1})$				
SensorID	GatewayHB	ServerDate_N	ServerDate	SensorHB	MessageDate_N	MessageDate	Data	RowCount
493372	305	2019-12-14 02:29:43	2019-12-14 02:24:38	599	2019-12-14 02:28:28	2019-12-14 02:18:29	6.6	3
493361	731	2019-12-14 02:36:49	2019-12-14 02:24:38	600	2019-12-14 02:30:56	2019-12-14 02:20:56	26.7	5
493368	731	2019-12-14 02:36:49	2019-12-14 02:24:38	898	2019-12-14 02:36:20	2019-12-14 02:21:22	5.6	6
493367	852	2019-12-14 02:43:55	2019-12-14 02:29:43	659	2019-12-14 02:38:27	2019-12-14 02:27:28	5.8	7
493372	852	2019-12-14 02:43:55	2019-12-14 02:29:43	601	2019-12-14 02:38:29	2019-12-14 02:28:28	6.4	8
493361	426	2019-12-14 02:43:55	2019-12-14 02:36:49	599	2019-12-14 02:40:55	2019-12-14 02:30:56	26.8	3
493368	1278	2019-12-14 02:58:07	2019-12-14 02:36:49	898	2019-12-14 02:51:18	2019-12-14 02:36:20	5.5	8
493367	426	2019-12-14 02:51:01	2019-12-14 02:43:55	660	2019-12-14 02:49:27	2019-12-14 02:38:27	5.6	4

Figure 4.85: Applying $yd_t - yd_{t-1}$ rendered multiple values of G_{dc} and its multiplications for each sensor node.

Therefore, G_{dc} intervals (the gateway duty-cycles) of each sensor node were aggregated and ranked based on their frequency of occurrence. The gateway module may skip some observations regularly due to misconfiguration problems which create conflicts between the duty-cycles of the sensor nodes and the gateway module and causes observations inconsistency in the time-series. This type of observations inconsistency is a special case of consistency data quality issues because it occurs regularly in the time series, unlike the common missing observations problem, which usually has no specific occurrence pattern. This type of observation inconsistency cannot skip more than two observations sequentially. For this reason, the Greatest Common Division (GSD) factor was applied only to the top three ranked gateway duty-cycles of each sensor node to determine the actual duty-cycle of the gateway and its multiplications, as shown in Figure 4.86. This approach successfully determined that all the sensor nodes in the local network are connected to the same gateway module since all exhibited the same

	SensorID	GatewayHB	SensorHB	Rowcount	optimal_GatewayHB
▶	493361	6	600	2	7
	493361	7	600	29	7
	493361	13	600	4	14
	493361	14	600	18	14
	493367	7	660	20	7
	493367	13	660	5	14
	493367	14	660	23	14
	493368	13	900	6	14
	493368	14	900	24	14
	493368	21	900	5	21
	493372	7	600	30	7
	493372	13	600	5	14
	493372	14	600	18	14

Figure 4.86: G_{dc} intervals were approximated, aggregated and aggregated again according to their greatest common divisor to determine the duty-cycle of gateway modules for each sensor nodes.

gateway duty-cycle and they all continuously shares the same observations timestamp at their occurrence in the database.

Applying the same approach to the real-world dataset indicated that none of the sensor nodes is connected to the same gateway module and indicated that all sensor nodes have the same gateway duty-cycle of one minute.

This approach of timestamp analysis mainly relies on the accuracy of observations' timestamps at their occurrence in the database. Therefore, the duty-cycle of the data acquisition unit was reduced from 9 minutes to 0.5 minutes while the duplication detection function was activated in the receiving table. Thus, this method ensured that gateways duty-cycles would be captured within $|0.5|$ minute accuracy confidence. This approach was applied to both the local (iMonnit) and the real-world (Thingful) sensor node networks, as shown in the sample of observations from the ideal network in Figure 4.85, the "RowCount" column for some records is up to 8 duplication's within a 14 minutes intervals (G_{dc}) between $y_{d_t} - y_{d_{t-1}}$ of observations. In contrast, the (G_{dc}) intervals of the real-world sensor nodes were echoing the duty-cycle of the data acquisition unit whenever it triggers the data collection procedure, as shown in Figure 4.87.

SensorID	RowCount	GatewayHB
024d0mnq	144	1
024d0mnq	98	3
024d0mnq	77	2
09nyrg07	237	1
09nyrg07	134	2
09nyrg07	72	3
0g8s4q56	326	1
0g8s4q56	266	2
0g8s4q56	107	3
0mypemb1	340	1
0mypemb1	236	2
0mypemb1	181	3

(G_dc) = 1 minute

Figure 4.87: G_{dc} , the gateway duty-cycles (if any) of the real-world sensor nodes were replicating the duty-cycle of the data acquisition unit.

The gateway duty-cycle, (G_{dc}), of all sensor nodes were equal to the duty cycle of the data collection duty-cycle, that indicates that these sensor nodes are consistently responding to the data acquisition request without any gateway delay. Therefore, it is highly likely that these sensor nodes are not connected to any gateway module, and they are operating as standalone stations which stream observations directly to the network.

This conclusion can be supported by the fact that gateway modules can only connect sensor nodes via analogue wire or wireless means within a limited range, up to 200 meters. Since all of the real-world temperature sensor nodes used in this case-study were relatively distinct from each other, it is technically not possible to connect these sensors to gateway modules. Alternatively, they function as individual stations that stream observations directly to the networks of data provides.

Taking into account the geographical distribution of all sensor nodes used in this case study, as shown in Figure 4.1, it seems that some of these sensors are not significantly distant from each other. To verify whether these sensor nodes are distant from each other or not, it is required to measure the shortest distance among all sensors and use that distance as a reference to compare it to the typical range of the gateway modules.

Calculating the shortest distance among all sensor nodes must be implemented for each sensor node network according to its owner and separately from the other networks. Thus, each sensors node network is mostly an independent body of infrastructure that is owned and managed by a different provider. Nearby sensor nodes from different providers do not stream data through each other network modules, at the analogue networking level at least. Therefore, the distance separating sensor nodes must be calculated only among sensors which belong to the same provider.

A list of sensor nodes providers of all sensors used in this case-study, for the selected time-window, is shown in Table 4.5.

Table 4.5: A list of sensor nodes providers of all sensors used in this case-study.

Provider	No. of sensor nodes	Total No. of Observations
Met Office	84	104007
Open Weather Map	116	530875
Open AQ	6	12877

Using the Met office sensor node network as a case study, the geographical distributions of the Met office sensor nodes is shown in Figure 4.88.

As shown in Figure 4.88, the distance among most of the Met office sensor nodes are extremely high comparing to the typical range of gateway modules. To measure the shortest distance between any two sensors in the network, the two sensor nodes of Kensington were chosen since both sensors seem (visually) to be deployed very close to each other, as shown in Figure 4.89.

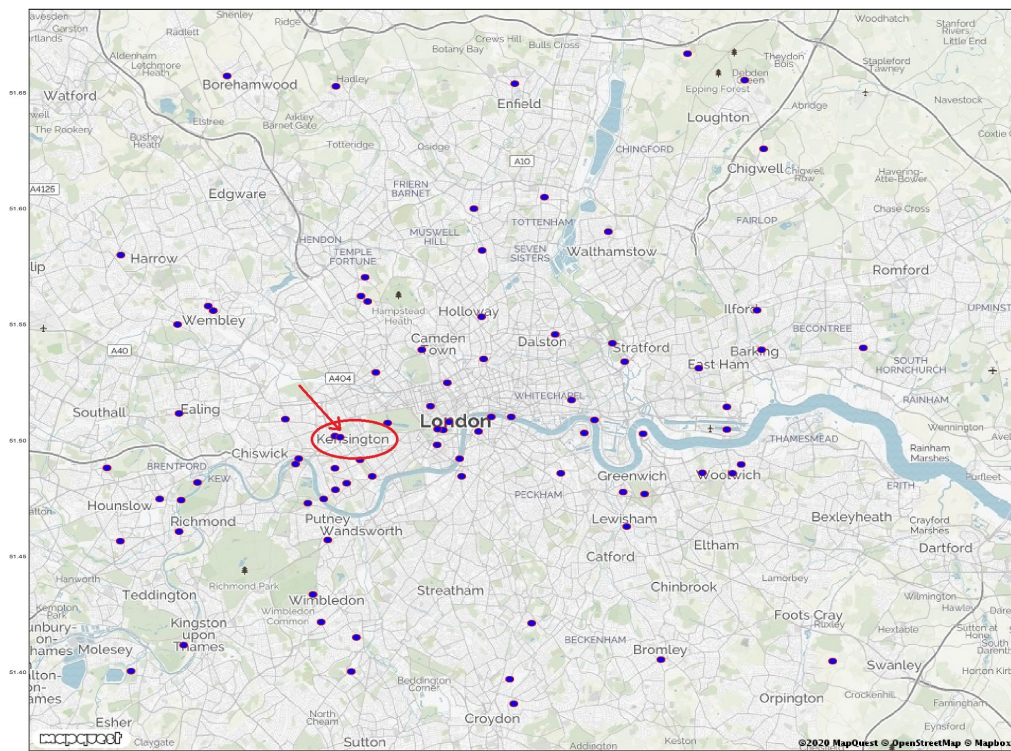


Figure 4.88: The geographical distribution of the Met office sensor nodes, highlighting the two sensor nodes of Kensington.

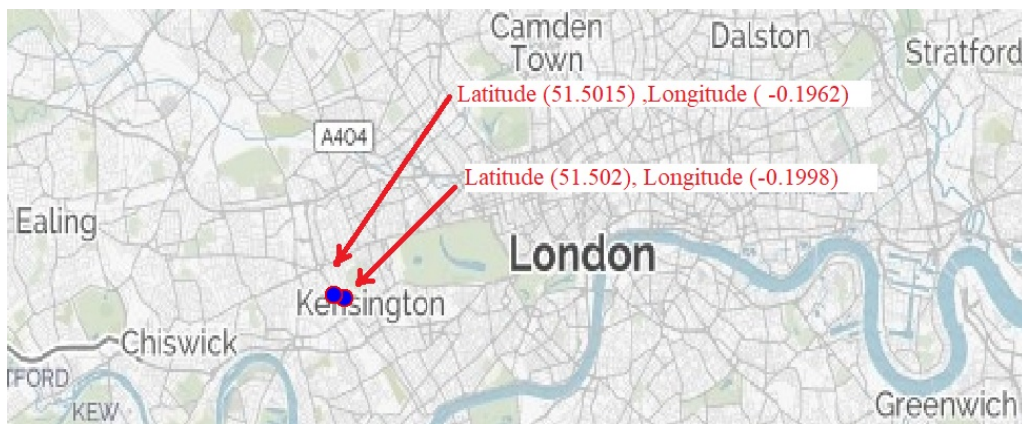


Figure 4.89: The two sensor nodes of Kensington as a reference to measure the shortest distance between any two sensors in the Met Office sensor node network used in this case-study.

Using the "GPS-distance between coordinates calculator" provided by (GPS-Coordinates, 2020) to determine the distance between the two sensors of Kensington based on their geographical coordinates (51.5015, -0.1962 and 51.502, -0.1998), revealed that these sensors are separated by 255 meters (0.26 KM), as shown in

Figure 4.90.

GPS Coordinates

Home My Location Coordinate Converter Where am I Distance Calculator My IP Address States Countries GPS Coordinates App

Distance Between Coordinates

Distance Calculator is use to calculate the **distance between coordinates** and distance between cities. If you are not sure what the [gps coordinates](#) are, you can use the [coordinates converter](#) to convert an address into latlong format or vice versa. If you don't know your location, use the [where am I right now](#) to find out.

Distance Calculator

Coordinate Distance Calculator calculates the distance between two gps coordinates. Enter the two gps coordinates in latitude and longitude format below, and our **distance calculator** will show you the distances between coordinates.

GPS Coordinates 1

Latitude

Longitude

GPS Coordinates 2

Latitude

Longitude

The distance is between the two gps coordinates is
0.26 KM or
0.16 Miles or
0.14 Nautical miles or
255.31 meters

Figure 4.90: Using the GPS-distance between coordinates calculator (GPS-Coordinates, 2020), revealed that Kensington sensors are separated by 255 meters (0.26 KM).

The result from Figure 4.90 indicates that the distance separating these real-world sensor nodes is relatively high compared to the maximum range of the gateway modules in general. This result confirms the outcome from the timestamp analysis model, which indicated that the none of the real-world sensor nodes used in the time-window used in this case study is connected to a common local gateway, if any. Alternatively, it is highly likely that these sensor nodes are operating as standalone stations which stream observations directly to the provider network. The timestamp analysis method was able to link sensor nodes to their local gateways successfully for detecting spatial mismatches in the contextual attributes of sensor nodes observations, but within a limited range where using gateway-modules is possible. Thus, if a sensor node shows a significant deviation in its geographical location compared to the location of the other sensor nodes connected to the same gateway module than the coordinates of that sensor node are potentially inaccurate. However, this approach is not valid in large-scale CPS applications which rely on standalone sensor nodes (stations) distributed over vast geographic area where no gateway modules can be used to connect these sensors

using an analogue medium. Therefore, these sensors stream their observations to the network directly.

4.4 Discussion and Summary

This chapter demonstrates the empirical findings and results from testing and evaluating the different models and components of the data quality management system, which were described in Chapter-3. This chapter fulfils the third and fourth objectives of the research by constructing a proof of concept, data quality management system and evaluating its validity and performance using a real-world, large-scale sensor node network as a case study. The data quality management system consists of three layers based on the functionality of each component, as follows:

4.4.1 The Data Acquisition Unit (Layers 1 and 2)

The Data Acquisition Unit (DAU) was developed to collect data streams from real-world sensor node networks in real-time. The collected data were used to empirically validate the different components of the proposed data quality management system. The real-world data were also used as a source to empirically identify the most common data quality issues in large-scale CPSs. The conceptual structure of the data acquisition unit is shown in Figure 3.8 Layer-1 and Layer-2, and its main processes are shown in Figure 3.10.

Inspecting the real-world dataset collected from over 200 temperature sensors revealed inaccuracy, inconsistency and mismatches in the temporal and spatial contextual attributes of these sensor nodes' observations which confirmed the literature review's outcomes and further validated the purpose of the proposed data quality management system.

Since there were no practical means to ensure that the collected observations from the real-world, large-scale network are noise free or complete, a benchmark high-quality, sensor node network was deployed at the University of East London. Both networks are real-world environmental monitoring sensor node networks which collect ambient temperature observations. The topology of the local sensor node network was explicitly chosen to match the main structure of the large-scale sensor node network, as shown in Figure 4.4. The aim was to involve the same type of modules and processes in the local network to experience the same type of latency and possibly the same data quality issues that may occur in the large-scale network. The local sensor node network streamed a long, noise-free and consistent sequence of observations utilised for training the system models and testing (calibrating) their accuracy before adopting them using real-world observations.

4.4.2 The Data Quality Assessment Unit

The data quality assessment unit is designed to detect data quality issues associated with errors in sensor nodes measurements, sensor nodes hardware failures, and mismatches in the spatial and temporal contextual attributes of sensor nodes observations to ensure these observations fitness for use in large-scale CPS applications. It utilises many data analysis techniques which categorised according to their operational mode into:

4.4.2.1 Online-Mode

The online-mode describes the functionality mode of the data quality management systems' components' responsible for detecting measurement errors in sensor nodes observations and render the data quality assessment results in real-time. In this context, the real-time notion indicates that the data quality assessment of a set of observations will be completed before receiving the next set of observations

from the same sensor nodes for all sensors in the network.

This unit is designed to detect errors in sensor nodes measurements associated with the primary data quality dimensions of accuracy, timeliness, completeness and consistency in large-scale CPS applications, as follows:

1. **Accuracy:** the accuracy of sensor nodes observations was evaluated based on their temporal correlation using predictive analysis models for short outliers detection, and based on sensor nodes' spatial correlation using anomaly analysis for long outliers detection, as follows:

- (a) **Predictive Analysis Models:** utilise autoregressive prediction techniques to use previous observations to predict future ones and compare the value of the actual (current) observations to the predicted ones to evaluate their accuracy:

- **Holt-Winters seasonal** could not adopt with rapid changes in the trend of sensor nodes' time-series. Furthermore, Holt-Winters is sensitive to short-term alterations in the value attribute of observations. Consequently, the accuracy of its predictions was iterating unpredictably. Thus Holt-Winters rendered unreliable predictions. Therefore, ARMA's models were investigated as alternative, more advanced predictive analysis techniques, as detailed in Section 4.2.1.2.
- **ARMA's and GPR** predictive models rendered relatively accurate and sustainable predictions, but only when applied using the one-step-forward predictions approach (rolling forecasting). Any extended range of forward predictions was associated with higher uncertainty margins. Thus, ARMA's and GPR models' predictions were only suitable for short interval accuracy evaluation, as detailed in Sections 4.2.1.3 and Section 4.2.1.4.

- **LSTM** predictive model rendered accurate predictions with extended forward range comparing with ARMA's and GPR models. LSTM predictive model benefits from long time-series in its learning process; therefore, it showed a higher ability to process rapid changes in time-series. LSTM utilises an advance mechanism to deal with noise with minimal configuration requirements. Furthermore, in general, the LSTM model is more suitable for an automated environment, especially considering that it requires fewer adjustment measures and fewer optimisations processes than the ARMA's or GPR models since most of the processes are integrated in the LSTM model, as detailed in Section 4.2.1.5.

Table 4.6 summarises the prediction analysis models' evaluation results based on their prediction accuracy, performance, and feasibility of automation applied to the ideal and real-world datasets.

Table 4.6: The evaluation results of the prediction analysis models applied to the ideal and real-world datasets.

Predictive Analysis Models	Dataset	Accuracy		Performance (Sec)		Feasibility of Automation
		Average (RMSE)/ Coefficient of Determination	Rolling Forecasting	Rendering interval	Optimisation interval	
H-W	Ideal	4.8699 / -		0.1960	-	High
	Real	1.5930 / -		0.2432	-	
ARMA	Ideal	0.1823 / -		0.8496	3.5246	Medium
	Real	0.2497 / -		1.3526	3.5695	
ARIMA	Ideal	0.1772 / -	✓	0.8681	7.0104	
	Real	0.2528 / -		1.3043	5.8651	
SARIMA	Ideal	0.1364 / -		0.2147	117.6147	
	Real	0.3229 / -		0.2998	114.1660	
GPR	Ideal	0.245 / 0.970	✓	5.9594	-	Low
	Real	0.475 / 0.946		5.02	-	
LSTM	Ideal	0.183 / 0.969		20.289	-	High
	Real	0.408 / 0.889		9.9459	-	

The limitation of the temporal correlation-based predictive analysis models is that they only detect sensors' measurement accuracy errors that occur for a short interval (point outliers). Thus, long-outliers change the time-series pattern to reflect the wrong measure-

ment as the standard time-series pattern, reducing the predictive analysis modes' ability to detect data accuracy issues. Therefore, spatial correlation-based anomaly analysis was investigated as a mechanism to detect accuracy errors that occur for a relatively long time (long-outliers).

(b) **Anomaly Analysis Models** are based on comparing observations' value attributes of nearby, spatially correlated, sensor nodes at a particular point in time. Applying anomaly analysis directly to the real-world temperature sensor node network was not possible because of the effect of the phenomena of Urban Heat Islands' in London. Urban heat islands cause sudden changes in observations' value attribute among temperature sensor nodes and violates their spatial continuity, as detailed in Section 3.6. Therefore, space partitioning techniques were utilised to divide the region of interest into smaller and more representative local areas using clustering algorithms, as follows:

- **K-means** was applied to partition the real-world network's sensor nodes into smaller and more representative distance-based groups of sensors (clusters) according to their spatial attributes. The Silhouette analysis method, which evaluates the clustering model's accuracy, was used to estimate the optimum number of clusters k for K-means. Applying the K-means model for multiple times to the same dataset revealed that K-means re-allocate the position of the centroid points' and created a fresh set of clusters with different distribution after each test. The major drawback of using K-means as spatial data partitioning technique for outlier detection is that K-means partitions the region of interest without considering the minimum number of sensor nodes or the maximum relative distance allowed between sensor nodes in the same cluster. Thus,

K-means may create a cluster of one sensor node or may include relatively distant sensor nodes in the same cluster which in both cases violate the spatial continuity constraints and may compromise the accuracy and validity of the outlier detection results, as detailed in Section 4.2.2.2.

- **DBSCAN** was investigated as an alternative to K-means. It identifies clusters based on the density of sensor nodes spatial distribution within a specified radius. The Silhouette analysis method was used to determine the optimum value of the DBSCAN radius Eps based on the best clustering performance. Unlike K-means, DBSCAN clustering results were consistent, rendering fixed clusters after being applied multiple times using the same dataset. The DBSCAN clustering algorithm results showed that temperature sensor nodes are not evenly distributed over the area of interest (London), as shown in Figure 4.61. DBSCAN categorised the region of interest into “high-density” areas and “low-density” areas according to the available number of sensor node within the radius Eps. DBSCAN model showed that sensor nodes located in the low-density area might not fit in any cluster and considered most of the distant sensor nodes as noise and eliminated them. In contrast, it identified geographical areas with high-density of sensor nodes, which are highly likely to be spatially correlated, and the accuracy of their observations can be verified using anomaly detection techniques. Applying anomaly analysis via direct comparison of sensor nodes’ observations values within DBSCAN clusters with a deviation threshold as a reference, successfully identified anomalies, as detailed in Section 4.2.2.3.

2. **Timeliness, Completeness and Temporal - Consistency** were evaluated using timestamp analysis which was nearly 100% accurate detecting and

identifying timeliness, completeness and consistency data quality issues in sensor nodes' data stream of both the ideal and the real-world datasets. This approach was implemented using a rule engine that determines the duty-cycle and the threshold interval for each sensor node to evaluate their temporal consistency. The rule engine was developed as an embedded component in the data quality management systems' database to ensure efficiency and minimal latency. The rule engine rendered its assessment results at the observation's arrival to the database instantly. The timestamp analysis was implemented using a rule engine to provide a level of automation to the process. Thus rule engines can adapt with data stream changes, and it can be configured by adding or revoking policies according to applications requirements, as detailed in Section 4.2.3.

4.4.2.2 Offline-Mode

The offline-mode describes the data quality management systems' components designed to perform much less routine data quality checks than the online components. These components would be triggered once every six hours or once a day. The offline components analysis all sensor nodes' time-series simultaneously using time-series clustering and timestamp analysis techniques. Time-series clustering was investigated as long-segmental outliers detection technique to identify sensor nodes hardware failures. Furthermore, timestamp analysis was investigated to detect mismatches in sensor nodes' spatial contextual attributes, as follows:

1. **Sensor Nodes Failure Detection** based on detecting long outliers in sensor nodes' data stream, where the occurrence of long-outliers in sensor nodes time-series indicates sensors malfunction (under standard conditions). Furthermore, detecting simultaneous long-segmental outliers in the data stream of multiple (nearby) sensor nodes indicates technical issues that have

a mass impact on the sensor node network, such as a power failure or a network breakdown. Time-series clustering techniques were applied as long-segmental outliers detection mechanisms, as follows:

- (a) **Dynamic Time Warping (DTW) and K-Shape** were applied to detect long segmental outliers in sensor nodes' data stream as an indication of sensor nodes hardware failure. Both techniques successfully identified sensor nodes' time-series with typical variations in the trend and seasonality from other time-series that showed continuous (stuck at), emerging and incipient faults behaviour up to 100% detection accuracy when applied to seven days time-series window. Applying DTW and K-Shape to the two-day interval dataset showed that DTW is more sensitive to the length of the time-window of the clustered time-series compared to K-Shape. The ability of DTW to differentiate the faulty from other (typical) time-series was more significantly affected compared to K-Shape, as detailed in Section 4.3.1.2. Thus, K-Shape was more able to maintain its clustering accuracy when applied to relatively shorter time-series than DTW. Both of the shape-based time-series clustering techniques required relatively long time-series to enhance the accuracy of their clustering results, especially the DTW. Therefore, the characteristics-based time-series clustering was investigated to find a more efficient way to detect long-segmental outliers even when applied to shorter time-series windows.
- (b) **Characteristics-Based Time-Series Clustering** (feature-based) relies on using arbitrary clustering algorithms such as K-means to cluster a set of features extracted from the examined sensor nodes data stream. The selected set of features may vary from application to another based on the time-series' characteristics chosen to be used as clustering reference. The feature-based time-series clustering technique identified the real-

world time-series with long segmental outliers successfully, even when it was applied to a relatively short interval of two days' time window. Since all the used time-series clustering techniques were applied to the same real-world dataset, it was possible to evaluate each of these methods' performance based on the time spent to render the clustering results. The feature-based time-series clustering technique rendered the clustering results in a short time interval, then K-shape and DTW. It is essential to highlight that these results may vary according to the examined time-series, the number and the type of the extracted features and the selected clustering algorithm, as detailed in Section 4.3.1.3.

2. Timestamp Analysis Model (Spatial Attributes Consistency): spatial and temporal analysis mechanisms were investigated to identify mismatches in sensor nodes' spatial contextual attributes. This approach can also be utilised to identify networks blackouts or gateway modules failures. The assumption behind this approach is that it is possible to identify sensor nodes which are connected to the same gateway module based on the spatial and temporal attributes associated with their observations. If a sensor node shows a significant deviation in its geographical location comparing to other sensor nodes connected to the same gateway than the coordinates of that sensor node are potentially inaccurate. Moreover, if all sensor nodes connected to the same gateway stop streaming data simultaneously, that indicates a gateway or network failure, as detailed in Section 3.9. The timestamp analysis model is based on the assumption that sensor nodes are highly likely to be connected to the same gateway module if their observations regularly exhibit correlated database timestamps and retain a similar gateway duty-cycle or any of its GCD. This approach successfully determined that all the local network's sensor nodes are connected to the same gateway module. Applying the same approach to the real-world dataset indicated that none

of the real-world sensor nodes utilised in this case study is connected to a common gateway module device. This conclusion is supported by the fact that gateway modules can only connect sensor nodes via analogue wire or wireless means within a limited range. Since all the real-world temperature sensor nodes in this case-study were relatively distinct from each other, it is technically infeasible to connect these sensors to common gateway modules. Alternatively, they function as standalone stations that stream observations directly to the networks of the data providers, as detailed in Section 4.3.2. The timestamp analysis method was able to link sensor nodes to their local gateways successfully for detecting spatial mismatches in the contextual attributes of sensor nodes observations, but within a limited range where using gateway modules is possible. Therefore, this approach is not valid in large-scale CPS applications which rely on standalone sensor nodes (stations) distributed over vast geographic area where no gateway modules can be used to connect these sensors using an analogue medium.

The next chapter revisits the research questions and illustrates the extent to which these questions were satisfied. It provides more insights into the key components of the data quality management systems.

Chapter 5

Conclusions and Future Work

“..erroneous assumption is that quality is an intangible and therefore not measurable. In fact, quality is precisely measurable by the oldest and most respected of measurements—cold hard cash.”
— (Crosby, 1979, p. 15)

This chapter revisits the research questions and illustrates the extent to which these questions were satisfied. It provides more insights into the key components of the data quality management systems. Finally, it outlines the conclusions of the research and future work.

5.1 Revisiting the Research Questions and Objectives

In this section, the research questions and objectives from Chapter-1 are discussed to explain the extent to which these questions were satisfied, as follows:

5.1.1 Review Question-1:

Is it feasible to develop a proof-of-concept data quality management system for large-scale CPSs that can; (1) detects sensor nodes measurements errors associated with the four main data quality dimensions of accuracy, timeliness, completeness, and consistency, (2) detects hardware failures in sensor nodes and sensors' communication networks and (3) ensures the quality of both spatial and temporal contextual attributes of sensor nodes observations?

To address this question the following objectives were set:

1. Objective 1: To investigate data quality challenges in large-scale cyber-physical systems based on the literature and based on empirical data analysis of observations collected from a real-world, large-scale sensor node network.

This objective is fulfilled by conducting a systematic literature review (SLR) to specify unaddressed data quality management challenges in large-scale CPSs based on the literature, as detailed in Chapter-2. The result of SLR revealed data quality challenges concerning sensor nodes' measurement errors detection, sensor nodes' failures detection and how to ensure the quality of observations' spatial and temporal contextual attributes in large-scale CPSs. Furthermore, inspecting the real-world dataset collected from the large-scale temperature sensor node network distributed around London revealed inaccuracy, inconsistency and mismatches in the temporal and spatial contextual attributes of these sensor nodes' observations which confirmed the SLR's outcomes and provided further evidence to support the research questions, as detailed in Section 4.1.2.2.

2. Objective 2: To investigate data mining techniques that may support the research aim, such as predictive and anomaly analysis techniques, time-series and timestamp analysis techniques.

This objective is fulfilled by viewing the literature systematically, as in chapter-2, and narratively as in chapter-3 to specify what data mining techniques were adopted to address data quality management challenges in large-scale CPSs. The result revealed that the most popular data mining techniques used for addressing data quality management challenges in large-scale CPSs are mainly based on anomaly analysis and predictive analysis techniques. These techniques were applied to address various data quality issues associated with the main data quality dimensions of accuracy, timeliness, completeness and consistency, as shown in Figure 2.8. However, it further revealed knowledge gaps that this research is bridging, e.g., all of the proposed prediction analysis models were based on an assumption that data accuracy issues occur for a short interval of time (point outliers). The reviewed primary studies did not provide a solution to address data accuracy issues associated with long outliers. Furthermore, the literature revealed that studies that investigated anomaly analysis as a solution to evaluate the accuracy of sensor nodes measurements by comparing their observations with different sensor nodes or to a pre-calculated threshold value were based on the assumption that these sensor nodes are spatially correlated. However, this assumption is not necessarily always valid in large-scale CPSs. The spatial continuity among sensor nodes in large-scale CPS applications might be compromised because of the vast distance separating these devices or other factors that disrupt the spatial continuity constraints, as detailed in Section 2.2.5.1 and Section 3.6. Therefore, predictive and anomaly analysis techniques were further investigated in this research as possible data accuracy assessment mechanisms for detecting short (point) and long outliers besides other data mining techniques such as time-series clustering and timestamp analysis to cover other aspects of data quality assessment performed

by the proposed data quality management system within the context of large-scale CPSs.

3. Objective 3: To construct, test and evaluate all the required models, components and tools of the proof-of-concept data quality management system to address the research aim. The data quality management system is expected to detect errors in sensors measurements, sensor nodes hardware failures, and mismatches in sensor nodes' spatial and temporal contextual attributes.

This objective is fulfilled by constructing, testing and evaluating the different components and models of the proposed proof-of-concept data quality management system. Each of the components or models serves a distinct purpose to ensure data fitness for use, as follows:

1. **Accuracy:** the accuracy of sensor nodes observations was evaluated based on their temporal correlation using predictive analysis models for short-interval outliers detection, and based on sensor nodes' spatial correlation using anomaly analysis for long outliers detection, as follows:

- (a) **Predictive Analysis Models:** utilises autoregressive prediction techniques to use previous observations to predict future ones and compare the value of the actual (current) observations to the predicted ones to evaluate their accuracy. The empirical tests showed that predictive analysis models that benefit from long time-series in its learning process, such as LSTM, showed higher ability to adopt with rapid changes in time-series, achieving a higher prediction accuracy. The key limitation of the predictive analysis approach as data accuracy assessment models is that these models can only detect sensors' measurement accuracy errors that occur for a short interval (point outliers). Thus, long-outliers change the time-series' pattern to reflect the wrong measurement as the standard time-series pattern, reducing the predictive analysis modes'

ability to detect data accuracy issues. Therefore, anomaly analysis was investigated as a mechanism to detect outliers that occur for a relatively long time (long-outliers).

(b) **Anomaly Analysis Models:** are based on comparing observations' value attributes of nearby, spatially correlated, sensor nodes at a particular point in time. Applying anomaly analysis directly to the real-world temperature sensor node network was not possible due to the effect of the phenomena of Urban Heat Islands' in London. Urban heat islands cause sudden changes in observations' value attribute among temperature sensor nodes and violates their spatial continuity, as detailed in Section 3.6. Therefore, space partitioning techniques were utilised to divide the region of interest into smaller and more representative local areas using clustering algorithms. The empirical tests showed that density-based spatial partitioning techniques, such as DBSCAN, are the most suitable for identifying geographical areas where sensor nodes are highly likely to be spatially correlated, and the accuracy of their observations can be verified using anomaly detection techniques.

2. **Timeliness, Completeness and Temporal - Mismatches:** were detected using timestamp analysis techniques which were nearly 100% accurate detecting timeliness, completeness and consistency data quality issues in sensor nodes' data stream of both the ideal and the real-world datasets. This approach was implemented using a rule engine that determines the duty-cycle and the threshold interval for each sensor node to evaluate their temporal consistency. The rule engine was developed as an embedded component in the data quality management systems' database to ensure efficiency and minimal latency. The timestamp analysis was implemented using a rule engine to provide a level of automation to the process. Thus rule engines can adapt with data stream changes, and it can be configured by adding

or revoking policies according to applications requirements, as detailed in Section 4.2.3.

3. **Sensor Nodes Failure Detection:** based on detecting long outliers in sensor nodes' data stream, where the occurrence of long-outliers in sensor nodes time-series indicates sensors malfunction (under standard conditions). Furthermore, detecting simultaneous long-segmental outliers in the data stream of multiple sensor nodes indicates technical issues that have a mass impact on the sensor node network, such as a power failure or a network breakdown. Time-series clustering (TSC) techniques were applied as long-segmental outliers detection mechanisms. The empirical tests showed that time-series clustering technique can be utilised successfully for ensuring observations fitness for use by evaluating the status of their streaming sensor nodes. Furthermore, the empirical tests showed that shape based time-series clustering techniques, such as DTW and K-shape require relatively long time-series to enhance the accuracy of their clustering results. In contrast, feature-based time-series clustering, such as the characteristics-based time-series clustering technique, was able to identify time-series with long segmental outliers successfully even when it was applied to a relatively short interval of time-series window and was able to render these results in a short time interval comparing to DTW and K-shape.
4. **Timestamp Analysis Model (Spatial Attributes Consistency):** spatial and temporal analysis mechanisms were investigated to identify mismatches in sensor nodes' spatial contextual attributes. The assumption behind this approach is that it is possible to identify sensor nodes which are connected to the same gateway module based on the spatial and temporal attributes associated with their observations. If a sensor node shows a significant deviation in its geographical location comparing to other sensor nodes connected

to the same gateway than the coordinates of that sensor node are potentially inaccurate. Moreover, if all sensor nodes connected to the same gateway stop streaming data simultaneously, that indicates a gateway or network failure, as detailed in Section 3.9. The timestamp analysis model is based on the assumption that sensor nodes are highly likely to be connected to the same gateway module if their observations regularly exhibit correlated database timestamps and retain a similar gateway duty-cycle or any of its GCD. This approach successfully determined that all the local network's sensor nodes are connected to the same gateway module. Applying the same approach to the real-world dataset indicated that none of the real-world sensor nodes utilised in this case study is connected to a common gateway module device. This conclusion is supported by the fact that gateway modules can only connect sensor nodes via analogue wire or wireless means within a limited range. Since all the real-world temperature sensor nodes in this case-study were relatively distinct from each other, it is technically infeasible to connect these sensors to common gateway modules. Alternatively, they function as standalone stations that stream observations directly to the networks of the data provider, as detailed in Section 4.3.2. The timestamp analysis method was able to link sensor nodes to their local gateways successfully for detecting spatial mismatches in the contextual attributes of sensor nodes observations, but within a limited range where using gateway modules is possible. Therefore, this approach is not valid in large-scale CPS applications which rely on standalone sensor nodes (stations) distributed over vast geographic area where no gateway modules can be used to connect these sensors using an analogue medium.

5.1.2 Review Question-2:

Is it possible to empirically evaluate the effectiveness of the proposed data quality management system using a real-world, large-scale sensor node network as a case-study?

To address this question the following objective was set:

Objective 4: To evaluate the effectiveness and performance of the proposed data quality management system utilising a real-world large-scale sensor node network as a case-study.

This objective is fulfilled by utilising a large-scale environmental sensor node network consists of over 200 temperature sensor nodes distributed around London and managed by different providers as a real-world case-study. Observations from the large-scale environmental sensor node network were collected in real-time forming continuous time-series for each sensor node in the network as detailed in section-4.1.1.1. The collected time-series were used for testing and evaluating the accuracy and performance of all the data quality assessment models adopted in the proposed data quality management system. For example, observations collected from the real-world, large-scale sensor node network were used to test the ability of the time-series clustering techniques to detect continuous (halting), and abrupt (emerging) long-outliers. Thus these types of long-outliers were detected in some time-series of the large-scale datasets, as detailed in section-4.3.1.

5.1.3 Review Question-3:

How to address bias concerns related to the evaluation process of the data quality management system, which emerges due to the presence of data quality issues in the testing or evaluating real-world dataset?

To address this question the following objective was set:

Objective 5: To validate the functionality and performance of the proposed data quality management system using a real-world, high-quality benchmark sensor node network. The benchmark sensor network must comprise high-quality sensors that stream consistent and error-free observations forming long time series of the same parameters collected from the large-scale sensor network.

To fulfil this objective a high-quality temperature sensor node network was deployed at the University of East London / Dockland campus. The local network purpose is to provide benchmark observations to validate the quality of the data collected from the large-scale network. It consists of four high-quality wireless temperature sensors, three of which were deployed outdoors, and the fourth was deployed indoors. The distance between the outdoor sensor nodes is relatively small (70 meters). One of the sensor nodes was installed indoors and the other three outdoor. The observations collected from the local sensor node network were utilised to empirically test and verify all of the developed data quality assessment models of the data quality management system. For example, the local network dataset was used to test the ability of the time-series clustering techniques to detect incipient faults with consistent offset long-outlier. Thus the indoor sensor node, in this case, represented a sensor with incipient fault. The indoor sensor streamed a time-series that is identical in its pattern with other three-time-series from the outdoor sensors but with a consistent offset of 10-15 C°, as shown in Figure 3.23. The topology of the local sensor node network was explicitly chosen to match the main structure of the large-scale sensor node network, as shown in Figure 4.4. The aim was to involve the same type of modules and processes in the local network to experience the same type of latency and possibly the same data quality issues that may occur in the large-scale network.

5.2 Contribution to Knowledge

The main contributions of this research are:

1. This research delivered a novel, proof of concept, data quality management system which is capable of evaluating sensor nodes' measurements based on the four dominant data quality dimensions, it detects sensor nodes' hardware failures and ensures the quality of observations' spatial and temporal contextual attributes in large-scale CPSs. Such a system can be utilised as a data quality assessment mechanism with compatible industrial or smart-cities scale CPS or IoT applications.
2. This research is an empirical study that utilises a real-world large-scale environment monitoring sensor node network consists of over 200 ambient temperature sensors distributed around London to support its outcomes and conclusions and further validate the robustness of the proposed data quality management system. Furthermore, it brings together advance predictive analysis, spatial partitioning, time-series clustering and timestamp analysis techniques which were successfully utilised to support the aim of this research. Therefore, this research can be used as an academic reference for future research conducted to address emerging data quality management challenges in the context of large-scale CPS applications.
3. This research is one of the very few studies that deliver an empirical data quality assessment solution based on advanced data science and machine-learning models while systematically addressing the bias concerns related to the evaluation process of the used models, which emerges because of the presence of data quality issues in the testing or evaluating real-world dataset. Thus in this research, a high-quality sensor node network was deployed at the University of East London and utilised to produce long, consistent,

high-quality data streams of benchmark observations to train and adjust the different data quality assessment models and to evaluate the performance and accuracy of these models before using them in real-world scenarios.

4. This research is an initiative empirical study that practically addresses data quality challenges associated with the contextual spatial and temporal attributes of sensor nodes observations in large-scale CPSs. It is one of the very few studies that provides and empirically validates a systematic approach for detecting inconsistency in temporal attributes and mismatches in the spatial attributes of sensor nodes' observations in the context of large-scale CPSs.
5. This research provides more insights and empirically exploits many of the large-scale CPSs features, e.g. this research has shown empirically that sensor nodes which are located near to each other (high-density areas) are highly likely to be spatially correlated. In contrast, distant sensor nodes (low-density area) might not fit in any spatial cluster or may form a cluster with other distant sensor nodes which violate their spatial continuity and thus verifying the accuracy of their observations using spatial correlation-based anomaly detection techniques may not render a reliable assessment result.

5.3 Conclusions

Data quality management is a set of procedures and activities that aim to fulfil data quality requirements by continuously monitoring, measuring, and ensuring data fitness for use. This research aims to develop a comprehensive proof-of-concept data quality management system for large-scale CPSs and empirically evaluate its validity. The data quality management system is designed to evaluate sensor nodes observations fitness for use via (1) detecting sensor nodes measurements

errors associated with accuracy, timeliness completeness and consistency, (2) detecting sensor nodes hardware failures and (3) detecting mismatches in the spatial conceptual attributes of sensor nodes' observations in large-scale CPSs.

Sensor nodes measurement errors detection:

For accuracy assessment, the empirical tests showed that predictive analysis based data quality assessment models are effective in detecting point (short) outliers. Predictive models that benefit from long time-series in their learning process, such as LSTM, are more able to render relatively accurate predictions with extended forward range comparing with other traditional statistical predictive models, such as the ARMA. The empirical tests also showed that anomaly analysis combined with spatial partitioning can be successfully utilised for detecting accuracy issues associated with both short and long-outlier in large-scale CPSs but only when applied to geographical areas with high-density of sensor nodes clusters.

Timestamp analysis techniques combined with a rule engine were utilised successfully and empirically proven in this research as an effective mean for detecting sensor nodes measurement errors associated with timeliness, completeness and consistency with a nearly 100% accuracy based on determining the duty-cycle and the threshold interval for each sensor node to evaluate its observations temporal consistency.

Sensor nodes hardware failure detection:

Time-series clustering is empirically proven in this research as an effective long-segmental outliers detection mechanism and utilised for detecting sensor nodes hardware failures successfully using shape-based and characteristics-based time series clustering techniques. Time-series clustering techniques successfully identified sensor nodes' time-series with typical variations in the trend and seasonality from other time-series that showed

continuous (stuck at) and incipient faults behaviour (long- segmental outliers) with up to 100% detection accuracy. The empirical tests also showed that the characteristics-based time-series clustering technique could maintain its detection accuracy even when it applied to a relatively short interval of time-series window comparing with the other shape-based (DTW and K-shape) time-series clustering techniques.

Observations spatial conceptual attributes mismatches detection:

Timestamp analysis techniques were utilised to identify mismatches in sensor nodes' spatial contextual attributes successfully, but only when applied to sensor nodes that rely on gateway module devices to stream their observations. The empirical tests in this research showed that timestamp analysis for detecting spatial mismatches in the contextual attributes of sensor nodes observations is not a valid approach in large-scale CPS applications that rely on standalone sensor nodes (stations) distributed over vast geographic area where no gateway modules can connect these sensors using an analogue medium.

The functionality and performance of the data quality management models were evaluated using observations collected from a real-world large-scale network comprises over 200 temperature sensor nodes distributed around London as a case-study and further validated using observations collected from a high-quality benchmark sensor node network to eliminate any bias concerns related to the evaluation process of the data quality management system, which emerges because of the presence of data quality issues in the testing or evaluating real-world dataset.

5.4 Future Work

As a short term plan, it is possible to investigate time-series injection techniques to enhance the accuracy of the time-series clustering techniques. The hypothesis behind the time-series injection approach is that: it is possible to inject time series with different combinations of outliers into the time-series stream of the system to enable the time-series clustering models to be more sensitive to identify similar outliers on their occurrence.

As a long term plan, this work can be extended to investigate data quality management of CPSs with higher frequency. Such system needs more advanced technical and parallel computing solutions to process sensor nodes observations and render the required data quality assessment results in real-time.

References

- Abid, A., Kachouri, A., Ben Fradj Guiloufi, A., Mahfoudhi, A., Nasri, N., & Abid, M. (2015). Centralized knn anomaly detector for wsn. In *2015 IEEE 12th International Multi-Conference on Systems, Signals Devices (SSD15)*, (pp. 1–4).
- Abid, A., Kachouri, A., & Mahfoudhi, A. (2017). Outlier detection for wireless sensor networks using density-based clustering approach. *IET Wireless Sensor Systems*, 7(4), 83–90.
- Addabbo, T., Fort, A., Mugnaini, M., Panzardi, E., Pozzebon, A., & Vignoli, V. (2019). A city-scale iot architecture for monumental structures monitoring. *Measurement*, 131, 349–357.
- Adhikari, M., Kar, S., Banerjee, S., & Biswas, U. (2015). Big Data Analysis for Cyber-Physical Systems. *undefined*.
URL <https://www.semanticscholar.org/paper/Big-Data-Analysis-for-Cyber-Physical-Systems-Adhikari-Kar/2f8e376b34c56ef8a3aa12797fd111c1aa58ae4b>
- Aggarwal, C. C. (2013). *Managing and Mining Sensor Data* | Charu C. Aggarwal | Springer. Springer US.
- Aggarwal, C. C. (2015). *Data mining: the textbook*. Springer.
- Aggarwal, C. C. (2016). An Introduction to Outlier Analysis. In *Outlier Analysis*, (pp. 1–34). Cham, Switzerland: Springer.
- Aggarwal, C. C. (2017). Time series and multidimensional streaming outlier detection. In *Outlier Analysis*, (pp. 273–310). Springer.
- Aggarwal, C. C., & Reddy, C. K. (2014). Data clustering. *Algorithms and applications*. Chapman&Hall/CRC Data mining and Knowledge Discovery series, Londra.
- Aggarwal, K. K., Singh, Y., Kaur, A., & Malhotra, R. (2009). Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: a replicated case study. *Softw. Process*, 14(1), 39–62.
- Aghabozorgi, S., Seyed Shirkhorshidi, A., & Ying Wah, T. (2015). Time-series clustering – a decade review. *Information Systems*, 53, 16–38.
URL <https://www.sciencedirect.com/science/article/pii/S0306437915000733>

- Ahmed, A., Pasha, M. A., Ahmad, Z., Masud, S., & Sikora, A. (2017). Energy efficient sensor network routing (eesnr) protocol for large distributed environmental monitoring applications. In *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 2, (pp. 740–745).
- Al-Milli, N., & Almobaideen, W. (2019). Hybrid neural network to impute missing data for iot applications. In *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, (pp. 121–125).
- Alhmiedat, T. (2015). *Wireless Sensor Networks toward Cyber-Physical Systems. undefined.*
 URL <https://www.semanticscholar.org/paper/Wireless-Sensor-Networks-toward-Cyber-Physical-Alhmiedat/554e16aae0da4f0cf1db403b75d7405d5f77a816>
- Amjad, M., Sharif, M., Afzal, M. K., & Kim, S. W. (2016). Tinyos-new trends, comparative views, and supported sensing applications: A review. *IEEE Sensors Journal*, 16(9), 2865–2889.
- Anderson, D. R., Sweeney, D. J., Williams, T. A., Camm, J. D., & Cochran, J. J. (2016). *Statistics for Business & Economics*. Boston, MA, USA: Cengage Learning.
- Andrés, G. R. C. (2016). Cleanwifi: The wireless network for air quality monitoring, community internet access and environmental education in smart cities. In *2016 ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT)*, (pp. 1–6).
- Antoo, A., & Mohammed, A. R. (2014). Eem-leach: Energy efficient multi-hop leach routing protocol for clustered wsns. In *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, (pp. 812–818).
- Appice, A., Ciampi, A., Fumarola, F., & Malerba, D. (2014). Sensor networks and data streams: Basics. In *Data Mining Techniques in Sensor Networks*, (pp. 1–8). Springer.
- Arthur, D., & Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding. Tech. rep., Stanford.
- Ashibani, Y., & Mahmoud, Q. H. (2017). Cyber physical systems security: Analysis, challenges and solutions. *Computers and Security*, 68, 81–97.
 URL <https://www.sciencedirect.com/science/article/pii/S0167404817300809>
- Auger, A., Exposito, E., & Lochin, E. (2016). iqas: An integration platform for qoi assessment as a service for smart cities. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, (pp. 88–93).
- Auger, A., Exposito, E., & Lochin, E. (2017). Sensor observation streams within cloud-based iot platforms: Challenges and directions. In *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, (pp. 177–184).

- Auger, A., Exposito, E., & Lochin, E. (2017). Survey on Quality of Observation within Sensor Web systems. *IET Wireless Sens. Syst.*, 7(6), 163–177.
- Avila, J., & Hauck, T. (2017). *scikit-learn Cookbook: Over 80 recipes for machine learning in Python with scikit-learn*. Packt Publishing Ltd.
- Ayadi, A., Ghorbel, O., Obeid, A. M., & Abid, M. (2017). Outlier detection approaches for wireless sensor networks: A survey. *Comput. Networks*, 129, 319–333.
- Badidi, E., Neyadi, N. E., Al Saeedi, M., Al Kaabi, F., & Maheswaran, M. (2018). *Building a Data Pipeline for the Management and Processing of Urban Data Streams*. Springer, Cham.
- Bahl, B. (2015). Inconsistency quality concerns for spatial database. In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIA-Com)*, (pp. 1328–1334).
- Bairagi, V., & Munot, M. V. (2019). *Research methodology: A practical and scientific approach*. CRC Press.
- Ballou, D., Wang, R., Pazer, H., & Kumar. Tayi, G. (1998). Modeling Information Manufacturing Systems to Determine Information Product Quality. *Manage. Sci.*, 44(4), 462–484.
URL <https://www.jstor.org/stable/2634609>
- Ballou, D. P., & Pazer, H. L. (1985). Modeling Data and Process Quality in Multi-Input, Multi-Output Information Systems. *Manage. Sci.*
URL <https://pubsonline.informs.org/doi/pdf/10.1287/mnsc.31.2.150>
- Barcelona Ciutat Digital, B. C. D. (2021). Barcelona ciutat digital.
URL <https://ajuntament.barcelona.cat/digital/ca>
- Barnaghi, P., Bermudez-Edo, M., & Tönjes, R. (2015). Challenges for Quality of Data in Smart Cities. *Data and Information Quality*, 6(2-3), 1–4.
- Basili, V. R. (1993). The experimental paradigm in software engineering. In *Experimental Software Engineering Issues: Critical Assessment and Future Directions*, (pp. 1–12). Springer.
- Batini, C., & Scannapieco, M. (2016). *Data Quality Dimensions*, (pp. 21–51). Cham: Springer International Publishing.
URL https://doi.org/10.1007/978-3-319-24106-7_2
- Benyuan Liu, & Towsley, D. (2004). A study of the coverage of large-scale sensor networks. In *2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE Cat. No.04EX975)*, (pp. 475–483).
- Berk, K. (2015). *Time series analysis*, (pp. 25–52). Wiesbaden: Springer Fachmedien Wiesbaden.
URL https://doi.org/10.1007/978-3-658-08669-5_3

- Berti-Équille, L. (2007). Measuring and Modelling Data Quality for Quality-Awareness in Data Mining. In *Quality Measures in Data Mining*, (pp. 101–126). Springer, Berlin, Heidelberg.
- Bhajantri, L. B., & Pundalik, R. (2017). Data processing in semantic sensor web: A survey. In *2017 3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, (pp. 166–170).
- Bhargavi, R. (2016). Complex event processing framework for big data applications. In *Data Science and Big Data Computing*, (pp. 41–56). Springer.
- Bhattacharyya, D. K., & Kalita, J. K. (2013). *Network anomaly detection: A machine learning perspective*. Crc Press.
- Bhuiyan, M. Z. A., Wu, J., Wang, G., Chen, Z., Chen, J., & Wang, T. (2017). Quality-guaranteed event-sensitive data collection and monitoring in vibration sensor networks. *IEEE Transactions on Industrial Informatics*, 13(2), 572–583.
- Bibri, S. E. (2018). *Introduction: The Rise of Sustainability, ICT, and Urbanization and the Materialization of Smart Sustainable Cities*. Springer, Cham.
- Bisadi, M., Akrami, A., Teimourzadeh, S., Aminifar, F., Kargahi, M., & Shahidehpour, M. (2018). IoT-Enabled Humans in the Loop for Energy Management Systems: Promoting Building Occupants' Participation in Optimizing Energy Consumption. *IEEE Electrif. Mag.*, 6(2), 64–72.
- Black, K. (2019). *Business statistics: for contemporary decision making*. John Wiley & Sons.
- Blaxter, L. (2010). *How to research*. McGraw-Hill Education (UK).
- Blessing, L. T. M., & Chakrabarti, A. (2009). *DRM, a Design Research Methodology*. London, England, UK: Springer-Verlag.
- Bonafini, F., Carvalho, D. F., Depari, A., Ferrari, P., Flammini, A., Pasetti, M., Rinaldi, S., & Sisinni, E. (2019). Evaluating indoor and outdoor localization services for lorawan in smart city applications. In *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4. 0&IoT)*, (pp. 300–305). IEEE.
- Bordel, B., Alcarria, R., Robles, T., & Martín, D. (2017). Cyber-physical systems: Extending pervasive sensing from control theory to the internet of things. *Pervasive and mobile computing*, 40, 156–184.
- Bose, S., Mukherjee, N., & Mistry, S. (2016). Environment monitoring in smart cities using virtual sensors. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, (pp. 399–404). IEEE.
- Bouveyron, C., Celeux, G., Murphy, T. B., & Raftery, A. E. (2019). *Model-based clustering and classification for data science: with applications in R*, vol. 50. Cambridge University Press.

- Box, G. (1979). Robustness in the strategy of scientific model building. In R. L. LAUNER, & G. N. WILKINSON (Eds.) *Robustness in Statistics*, (pp. 201 – 236). Academic Press.
 URL <http://www.sciencedirect.com/science/article/pii/B9780124381506500182>
- Brimicombe, A. (2009). *GIS, environmental modeling and engineering*. CRC Press.
- Brincat, A. A., Pacifici, F., Martinaglia, S., & Mazzola, F. (2019). The internet of things for intelligent transportation systems in real smart cities scenarios. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, (pp. 128–132). IEEE.
- Brownlee, J. (2017a). How to Decompose Time Series Data into Trend and Seasonality. [Online; accessed 4. Jan. 2021].
 URL <https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality>
- Brownlee, J. (2017b). *Long Short-term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning*. Machine Learning Mastery.
- Brunton, S. L., & Kutz, J. N. (2019). *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press.
- Burns, A., & Wellings, A. J. (2001). *Real-Time Systems and Programming Languages*. 3rd. Pearson Education.
- Bühmann, A., Härder, T., & Merker, C. (2006). A Middleware-Based Approach to Database Caching. In *Advances in Databases and Information Systems*, (pp. 184–199). Springer, Berlin, Heidelberg.
- Cai, L., & Zhu, Y. (2015). The Challenges of Data Quality and Data Quality Assessment in the Big Data Era. *Data Sci. J.*, 14(0).
- Carvalho, S., & White, H. (1997). *Combining the quantitative and qualitative approaches to poverty measurement and analysis: the practice and the potential*. The World Bank.
- Cassandras, C. G. (2016). Smart cities as cyber-physical social systems. *Engineering*, 2(2), 156–158.
 URL <https://www.sciencedirect.com/science/article/pii/S2095809916309420>
- Chandler, T. J. (1965). *The climate of London*. Hutchinson.
- Chauhan, S., Patel, P., Delicato, F. C., & Chaudhary, S. (2016). A development framework for programming cyber-physical systems. In *2016 IEEE/ACM 2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS)*, (pp. 47–53).
- Chen, L., Ho, Y., Hsieh, H., Huang, S., Lee, H., & Mahajan, S. (2018). Adf: An anomaly detection framework for large-scale pm2.5 sensing systems. *IEEE Internet of Things Journal*, 5(2), 559–570.

- Chen, M., Yang, J., Hu, L., Hossain, M. S., & Muhammad, G. (2018). Urban Healthcare Big Data System Based on Crowdsourced and Cloud-Based Air Quality Indicators. *IEEE Commun. Mag.*, 56(11), 14–20.
- Chidean, M. I., Morgado, E., Sanromán-Junquera, M., Ramiro-Bargueño, J., Ramos, J., & Caamaño, A. J. (2016). Energy efficiency and quality of data reconstruction through data-coupled clustering for self-organized large-scale wsns. *IEEE Sensors Journal*, 16(12), 5010–5020.
- Chollet, F. (2017). *Deep Learning with Python*. Manning Publications Company.
 URL <https://books.google.co.uk/books?id=Yo3CAQAACAAJ&dq=Deep+Learning+with+Python,+Manning&hl=en&sa=X&ved=2ahUKEwjygmt36ntAhUUesAKHXplBE8Q6AEwAHoECAIQAg>
- Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh—a python package). *Neurocomputing*, 307, 72–77.
- Chu, W. W. (2014). Data mining and knowledge discovery for big data. *Studies in Big Data*, 1, 153–192.
- Cisco (2020). What is a smart city?
 URL <https://www.cisco.com/c/en/us/solutions/industries/smart-connected-communities/what-is-a-smart-city.html>
- Creamer, E. G. (2017). *An introduction to fully integrated mixed methods research*. Sage Publications.
- Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.
- Crosby, P. B. (1979). *Quality is free: The art of making quality certain*, vol. 94. McGraw-hill New York.
- Davis, H. C. (1995). *Demographic projection techniques for regions and smaller areas: a primer*. UBC Press.
- De, S., Zhou, Y., Larizgoitia Abad, I., & Moessner, K. (2017). Cyber-physical-social frameworks for urban big data systems: A survey. *Applied Sciences*, 7(10).
 URL <https://www.mdpi.com/2076-3417/7/10/1017>
- de Aquino, G. R. C., de Farias, C. M., & Pirmez, L. (2019). *Hygieia: data quality assessment for smart sensor network*. New York, NY, USA: Association for Computing Machinery.
- Dean, J. (2014). *Big data, data mining, and machine learning: value creation for business leaders and practitioners*. John Wiley & Sons.
- Denzin, N. K., & Lincoln, Y. S. (2017). *The SAGE Handbook of Qualitative Research*. London, England, UK: SAGE Publications.
 URL <https://us.sagepub.com/en-us/nam/the-sage-handbook-of-qualitative-research/book242504>

- Dong, J., Paul, R., & Zhang, L.-J. (2009). *High assurance services computing*. Springer.
- Dong, W., Chen, C., Liu, X., & Bu, J. (2010). Providing os support for wireless sensor networks: Challenges and approaches. *IEEE communications surveys & tutorials*, 12(4), 519–530.
- Drăgoicea, M., Léonard, M., Ciolofan, S. N., & Militaru, G. (2019). Managing data, information, and technology in cyber physical systems: Public safety as a service and its systems. *IEEE Access*, 7, 92672–92692.
- Du, P., Yang, Q., He, Q., & Kwak, K. S. (2016). Energy-aware quality of information maximisation for wireless sensor networks. *IET communications*, 10(17), 2281–2289.
- Du, P., Yang, Q., Shen, Z., & Kwak, K. S. (2016). Quality of information maximization in lifetime-constrained wireless sensor networks. *IEEE Sensors Journal*, 16(19), 7278–7286.
- Ebrahimi, M., ShafieiBavani, E., Wong, R., & Chen, F. (2017). Exploring Celebrities on Inferring User Geolocation in Twitter. *undefined*.
URL <https://www.semanticscholar.org/paper/Exploring-Celebrities-on-Inferring-User-Geolocation-Ebrahimi-ShafieiBavani/de8b54f9f9db568da8ced3575e393e03ff1975cb>
- Efron, S. E., & Ravid, R. (2018). *Writing the literature review: A Practical Guide*. Guilford Publications.
- Ehrlinger, L., & Wöß, W. (2018). Automated schema quality measurement in large-scale information systems. In *International Workshop on Data Quality and Trust in Big Data*, (pp. 16–31). Springer.
- Environmental Research Group, I. C. L. (2021). London Air Quality Network Guide. [Online; accessed 6. May 2021].
URL <https://www.londonair.org.uk/londonair/guide/monitoring.aspx>
- European Commission, E. (2021). Smart cities.
URL https://ec.europa.eu/info/eu-regional-and-urban-development/topics/cities-and-urban-development/city-initiatives/smart-cities_en
- Fabozzi, F. J., Focardi, S. M., Rachev, S. T., & Arshanapalli, B. G. (2014). *The basics of financial econometrics: Tools, concepts, and asset management applications*. John Wiley & Sons.
- Fang, X. (2018). *Improving data quality for low-cost environmental sensors*. Ph.D. thesis, University of York.
URL <http://etheses.whiterose.ac.uk/21259>
- Farooqi, M. M., Ali Khattak, H., & Imran, M. (2018). Data quality techniques in the internet of things: Random forest regression. In *2018 14th International Conference on Emerging Technologies (ICET)*, (pp. 1–4).

- Fazlollahi, S., Girardin, L., & Maréchal, F. (2014). Clustering urban areas for optimizing the design and the operation of district energy systems. In *Computer Aided Chemical Engineering*, vol. 33, (pp. 1291–1296). Elsevier.
- Fink, G., Edgar, T., Rice, T., MacDonald, D., & Crawford, C. (2017). Security and privacy in cyber-physical systems. *Cyber-Physical Systems*, (pp. 129–141).
URL <https://www.sciencedirect.com/science/article/pii/B9780128038017000092>
- Fitriawan, H., Susanto, M., Arifin, A. S., Mause, D., & Trisanto, A. (2017). Zig-
bee based wireless sensor networks and performance analysis in various environments. In *2017 15th International Conference on Quality in Research (QiR) : International Symposium on Electrical and Computer Engineering*, (pp. 272–275).
- Foehr, M., Vollmar, J., Calà, A., Leitão, P., Karnouskos, S., & Colombo, A. W. (2017). Engineering of next generation cyber-physical automation system architectures. *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*, (pp. 185–206).
URL https://link.springer.com/chapter/10.1007/978-3-319-56345-9_8
- Forster, A. (2016). *Introduction to Wireless Sensor Networks*. John Wiley and Sons.
- Fürber, C., & Hepp, M. (2010). Using semantic web resources for data quality management. In P. Cimiano, & H. S. Pinto (Eds.) *Knowledge Engineering and Management by the Masses*, (pp. 211–225). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Geisler, S., Quix, C., Weber, S., & Jarke, M. (2016). Ontology-Based Data Quality Management for Data Streams. *J. Data and Information Quality*, 7(4), 1–34.
- Ghorbel, O., Jmal, M. W., Abid, M., & Snoussi, H. (2015). Distributed and efficient one-class outliers detection classifier in wireless sensors networks. In *International Conference on Wired/Wireless Internet Communication*, (pp. 259–273). Springer.
- Ghosh, N., Maity, K., Paul, R., & Maity, S. (2019). Outlier detection in sensor data using machine learning techniques for iot framework and wireless sensor networks: A brief study. In *2019 International Conference on Applied Machine Learning (ICAML)*, (pp. 187–190).
- Giacobbe, M., Di Pietro, R., Longo Minnolo, A., & Puliafito, A. (2018). Evaluating information quality in delivering iot-as-a-service. In *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, (pp. 405–410).
- Giffinger, R., & Pichler-Milanović, N. (2007). *Smart cities: Ranking of European medium-sized cities*. Centre of Regional Science, Vienna University of Technology.
- Gill, H. (2008a). A continuing vision: Cyber-physical systems. In *Fourth annual Carnegie Mellon conference on the electricity industry*.
URL <https://research.ece.cmu.edu/electricconf/2008/PDFs/Gill%20>

CMU%20Electrical%20Power%202008%20-%20%20Cyber-Physical%20Systems%20-%20A%20Progress%20Report.pdf

- Gill, H. (2008b). From vision to reality: cyber-physical systems. In *HCSS national workshop on new research directions for high confidence transportation CPS: automotive, aviation, and rail*, (pp. 18–20). Austin USA.
URL https://labs.ece.uw.edu/nsl/aar-cps/Gill_HCSS_Transportation_Cyber-Physical_Systems_2008.pdf
- Glowalla, P., & Sunyaev, A. (2014). Process-driven data quality management: A critical review on the application of process modeling languages. *J. Data and Information Quality*, 5(1-2), 1–30.
- Goldberg, M., & Zhang, Z. (2018). A cyber-physical system framework towards smart city and urban computing to aid people with disabilities. In *2018 27th Wireless and Optical Communication Conference (WOCC)*, (pp. 1–5). IEEE.
- Goss, Q., Akbaş, M. İ., Jaimes, L. G., & Sanchez-Arias, R. (2014). Street Network Generation with Adjustable Complexity Using k-Means Clustering. *2019 SoutheastCon*, (pp. 1–6).
- GPS-Coordinates (2020). Distance Calculator | Distance Between Coordinates | Distance Between Cities. [Online; accessed 12. Dec. 2020].
URL <https://gps-coordinates.org/distance-between-coordinates.php>
- Green, B. N., Johnson, C. D., & Adams, A. (2006). Writing narrative literature reviews for peer-reviewed journals: secrets of the trade. *Journal of chiropractic medicine*, 5(3), 101–117.
- Greer, C. (2014). NIST Cyber-Physical Systems Public Working Group Kickoff Webinar. [Online; accessed 1. May 2021].
URL <https://www.nist.gov/system/files/documents/el/CPS-PWG-Kickoff-Webinar-Presentation-FINAL.PDF>
- Greer, C., Burns, M., Wollman, D., & Griffor, E. (2019). Cyber-physical systems and internet of things.
URL <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1900-202.pdf>
- Grega, W., & Kornecki, A. J. (2015). Real-time cyber-physical systems transatlantic engineering curricula framework. In *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, (pp. 755–762). IEEE.
- Guéhéneuc, Y.-G., & Khomh, F. (2019). Empirical Software Engineering. In *Handbook of Software Engineering*, (pp. 285–320). Cham, Switzerland: Springer.
- Guillet, F., & Hamilton, H. J. (2007). *Quality measures in data mining*, vol. 43. Springer.
- Gumzej, R. (2018). *Engineering Safe and Secure Cyber-Physical Systems the Specification PEARL Approach*. Springer International Publishing Springer.

- Gunes, V., Peter, S., Givargis, T., & Vahid, F. (2014). A survey on concepts, applications, and challenges in cyber-physical systems. *KSII Transactions on Internet & Information Systems*, 8(12).
 URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.717.3807&rep=rep1&type=pdf>
- Guo, A., Liu, X., & Sun, T. (2018). *Research on Key Problems of Data Quality in Large Industrial Data Environment*. New York, NY, USA: Association for Computing Machinery.
- Guo, D., Peuquet, D. J., & Gahegan, M. (2003). ICEAGE: Interactive Clustering and Exploration of Large and High-Dimensional Geodata. *GeoInformatica*, 7(3), 229–253.
- Hakiri, A., & Gokhale, A. (2014). Work-in-Progress: Towards Real-Time Smart City Communications using Software Defined Wireless Mesh Networking. *2018 IEEE Real-Time Systems Symposium (RTSS)*, (pp. 177–180).
- Hanrong Lu, Xin Chen, Xuhui Lan, & Feng Zheng (2016). Duplicate data detection using gnn. In *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, (pp. 167–170).
- Hartigan, J. A., & Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1), 100–108.
- Haseeb, M., Hussain, H. I., Ślusarczyk, B., & Jermsittiparsert, K. (2019). Industry 4.0: A solution towards technology challenges of sustainable business performance. *Social Sciences*, 8, 154.
 URL <https://www.mdpi.com/2076-0760/8/5/154/htm>
- Herrera-Quintero, L. F., Vega-Alfonso, J. C., Banse, K. B. A., & Zambrano, E. C. (2018). Smart its sensor for the transportation planning based on iot approaches using serverless and microservices architecture. *IEEE Intelligent Transportation Systems Magazine*, 10(2), 17–27.
- Horn, P. (2001). *Autonomic Computing: IBM's Perspective on the State of Information Technology*. [Online; accessed 17. Nov. 2020].
 URL https://homeostasis.scs.carleton.ca/~soma/biosecc/readings/autonomic_computing.pdf
- Hu, F. (2014). *Cyber-physical systems : integrated computing and engineering design*. Crc Press.
- Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.
- Islam, S. M., & Watanapalachaikul, S. (2012). *Empirical finance: modelling and analysis of emerging financial and stock markets*. Springer Science & Business Media.

- ISO 5725-1, I. (1994). 5725-1: 1994, accuracy (trueness and precision) of measurement methods and results-part 1: General principles and definitions. *International Organization for Standardization, Geneva*.
- ISO 8402, I. (1994). Iso 8402: Quality management and quality assurance — vocabulary.
URL <https://www.iso.org/standard/20115.html>
- Jahromi, A. A., & Kundur, D. (2020). Fundamentals of Cyber-Physical Systems. In *Cyber-Physical Systems in the Built Environment*, (pp. 1–13). Springer, Cham.
- Jain, A., Patel, H., Nagalapatti, L., Gupta, N., Mehta, S., Guttula, S., Mujumdar, S., Afzal, S., Sharma Mittal, R., & Munigala, V. (2020). *Overview and Importance of Data Quality for Machine Learning Tasks*. New York, NY, USA: Association for Computing Machinery.
- Januzaj, E., Januzaj, V., & Mandl, P. (2019). *An Application of Distributed Data Mining to Identify Data Quality Problems*. New York, NY, USA: Association for Computing Machinery.
- Jayswal, M., & Shukla, M. (2016). Consolidated study analysis of different clustering techniques for data streams. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, (pp. 3541–3547).
- Jeyaraman, B. P., Olsen, L. R., & Wambugu, M. (2019). *Practical Machine Learning with R: Define, build, and evaluate machine learning models for real-world applications*. Packt Publishing.
URL <https://www.packtpub.com/product/practical-machine-learning-with-r/9781838550134>
- Jonker, J., & Pennink, B. W. (2010). The essence of methodology. In *The Essence of Research Methodology*, (pp. 21–41). Springer.
- Jugulum, R. (2014). *Competing with High Quality Data: Concepts, Tools, and Techniques for Building a Successful Approach to Data Quality*. Wiley.
URL <https://www.wiley.com/en-gb/exportProduct/pdf/9781118416495>
- Juran, J. M., Gryna, F. M., et al. (1988). *Juran's quality control handbook*, vol. 4. McGraw-Hill New York.
- Kale, S., Tamakuwala, H., Vijayakumar, V., Yang, L., & Kshatriya, B. S. R. (2019). *Big Data in Healthcare: Challenges and Promise*. Springer, Singapore.
- Karkouch, A., Al Moatassime, H., Mousannif, H., & Noel, T. (2015). Data quality enhancement in internet of things environment. In *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*, (pp. 1–8).
- Karkouch, A., Mousannif, H., Moatassime, H. A., & Noel, T. (2016). A model-driven architecture-based data quality management framework for the internet of things. In *2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech)*, (pp. 252–259).

- Karmakar, G. C., Das, R., & Kamruzzaman, J. (2020). Iot sensor numerical data trust model using temporal correlation. *IEEE Internet of Things Journal*, 7(4), 2573–2581.
- Keller, G. (2015). *Statistics for Management and Economics, Abbreviated*. Cengage Learning.
- Kenett, R. S., & Shmueli, G. (2016). *Information Quality: The Potential of Data and Analytics to Generate Knowledge*. Wiley.
URL <https://www.wiley.com/en-gb/exportProduct/pdf/9781118890653>
- Khotimah, K., Sadik, K., & Rizki, A. (2019). Study of robust regression modeling using mm-estimator and least median squares. In *ICSA 2019: Proceedings of the 1st International Conference on Statistics and Analytics, ICSA 2019, 2-3 August 2019, Bogor, Indonesia*, (p. 100). European Alliance for Innovation.
- Kim, E. (2017). Smart city service platform associated with smart home. In *2017 International Conference on Information Networking (ICOIN)*, (pp. 608–610). IEEE.
- Kim, J., Abdelzaher, T., Sha, L., Bar-Noy, A., Hobbs, R., & Dron, W. (2016). On maximizing quality of information for the internet of things: A real-time scheduling perspective (invited paper). In *2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, (pp. 202–211).
- Kim, S., Castillo, R. P. D., Caballero, I., Lee, J., Lee, C., Lee, D., Lee, S., & Mate, A. (2019). Extending data quality management for smart connected product operations. *IEEE Access*, 7, 144663–144678.
- Kirchgässner, G., & Wolters, J. (2007). *Introduction to modern time series analysis*. Springer Science & Business Media.
- Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. *EBSE Technical Report EBSE-2007-01 / Keele University - UK*.
- Kitchenham, B. A., Budgen, D., & Brereton, P. (2015). *Evidence-based software engineering and systematic reviews*, vol. 4. CRC press.
- Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., Emam, K. E., & Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Trans. Software Eng.*, 28(8), 721–734.
- Kočenda, E., & Černý, A. (2015). *Elements of time series econometrics: An applied approach*. Charles University in Prague, Karolinum Press.
- Kocijan, J. (2016). *Modelling and control of dynamic systems using Gaussian process models*. Springer.
- Kounev, S., Kephart, J. O., Milenkoski, A., Zhu, X., & Ag, S. I. P. (2018). *Self-Aware Computing Systems*. Cham Springer International Publishing Springer.

- Krishna, M. B. (2018). Group-based incentive and penalizing schemes for proactive participatory data sensing in iot networks. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, (pp. 796–801).
- Kuhn, T. S. (1962). *The structure of scientific revolutions*: University of chicago press. *Original edition*.
- Kumar, A. (2016). *Learning predictive analytics with Python*. Packt Publishing Ltd.
- Kumar, P., & Chaturvedi, A. (2014). Life time enhancement of wireless sensor network using fuzzy c-means clustering algorithm. In *2014 International Conference on Electronics and Communication Systems (ICECS)*, (pp. 1–5).
- Labouseur, A. G., & Matheus, C. C. (2017). An Introduction to Dynamic Data Quality Challenges. *J. Data and Information Quality*, *8*(2), 1–3.
- Lamnabhi-Lagarrigue, F., Annaswamy, A., Engell, S., Isaksson, A., Khargonekar, P., Murray, R. M., Nijmeijer, H., Samad, T., Tilbury, D., & Van den Hof, P. (2017). Systems & control for the future of humanity, research agenda: Current and future roles, impact and grand challenges. *Annual Reviews in Control*, *43*, 1–64.
URL <https://www.sciencedirect.com/science/article/pii/S1367578817300573>
- Langer, A. M. (2007). *Analysis and design of information systems*. Springer Science & Business Media.
- Larburu, N., Bults, R., van Sinderen, M., & Hermens, H. (2015). Quality-of-Data Management for Telemedicine Systems. *Procedia Comput. Sci.*, *63*, 451–458.
- Larburu, N., Bults, R. G. A., Widya, I., & Hermens, H. J. (2014). Quality of data computational models and telemedicine treatment effects. In *2014 IEEE 16th International Conference on e-Health Networking, Applications and Services (Healthcom)*, (pp. 364–369).
- Laso, P. M., Brosset, D., & Puentes, J. (2017). Analysis of quality measurements to categorize anomalies in sensor systems. In *2017 Computing Conference*, (pp. 1330–1338). IEEE.
- LAURA, A. (2016). How Smart City Barcelona Brought the Internet of Things to Life. [Online; accessed 6. May 2021].
URL <https://datasmart.ash.harvard.edu/news/article/how-smart-city-barcelona-brought-the-internet-of-things-to-life-789>
- Lawson, A. B., & Denison, D. G. (2002). *Spatial cluster modelling*. CRC press.
- Lawson, V. J., & Ramaswamy, L. (2016). Tau-five: a multi-tiered architecture for data quality and energy-sustainability in sensor networks. In *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, (pp. 169–176). IEEE.

- Lee, C.-F., Lee, J. C., & Lee, A. C. (2013). *Statistics for Business and Financial Economics*. New York, NY, USA: Springer-Verlag.
- Lee, E. A. (2006). Cyber-physical systems-are computing foundations adequate. In *Position paper for NSF workshop on cyber-physical systems: research motivation, techniques and roadmap*, vol. 2, (pp. 1–9). Citeseer.
URL <https://cps-vo.org/node/179>
- Lee, R., Jang, R., Park, M., Jeon, G., Kim, J., & Lee, S. (2020). Making iot data ready for smart city applications. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, (pp. 605–608).
- Lemahieu, W., Broucke, S. v., & Baesens, B. (2018). *Principles of Database Management*. Cambridge University Press.
URL <https://www.cambridge.org/gb/academic/subjects/computer-science/knowledge-management-databases-and-data-mining/principles-database-management-practical-guide-storing-managing-and-analyzing-big-and-small-data?format=HB&isbn=9781107186125>
- Li, D., Yan, L., Liu, Y., Yin, Q., Guo, S., & Zheng, H. (2019). Data quality improvement method based on data correlation for power internet of things. In *2019 12th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 2, (pp. 259–263).
- Li, W., Sha, K., & Zeadally, S. (2016). Routing in wireless sensor networks for cyber-physical systems. In *Cyber-Physical system design with sensor networking technologies*, (pp. 149–176). IET Digital Library.
- Liao, W., Kuai, S., & Chang, C. (2019). Energy harvesting path planning strategy on the quality of information for wireless sensor networks. In *2019 IEEE International Conference of Intelligent Applied Systems on Engineering (ICIASE)*, (pp. 82–85).
- Lin, C., Han, G., Du, J., Xu, T., Shu, L., & Lv, Z. (2020). Spatio-temporal congestion-aware path planning towards intelligent transportation systems in software-defined smart city iot. *IEEE Internet of Things Journal*.
- Lind, D. A., Marchal, W. G., & Wathen, S. A. (2018). *Basic Statistics for Business and Economics*. New York, NY, USA: McGraw-Hill Education.
URL <https://books.google.co.uk/books?id=QhyduQEACAAJ&dq=Basic+statistics+for+business+and+economics.+9th+Edition&hl=en&sa=X&ved=2ahUKEwj5Yv5wJLtAhXdTBUIHXKRKBN0Q6AEwAHoECAAQAg>
- Liu, C. H., Fan, J., Branch, J. W., & Leung, K. K. (2014). Toward qoi and energy-efficiency in internet-of-things sensory environments. *IEEE Transactions on Emerging Topics in Computing*, 2(4), 473–487.
- Liu, H., Wang, X., Lei, S., Zhang, X., Liu, W., & Qin, M. (2019). *A rule based data quality assessment architecture and application for electrical data*. New York, NY, USA: Association for Computing Machinery.

- Liu, L., Chen, W., Solanas, A., & He, A. (2017). Knowledge, attitude, and practice about internet of things for healthcare. In *2017 International Smart Cities Conference (ISC2)*, (pp. 1–4). IEEE.
- Liu, X., Yu, X. L., & Fei, T. (2012). Research on Building Data Acquisition Methods in Smart City. *2020 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, (pp. 144–147).
- Liu, Y., Weng, X., Wan, J., Yue, X., Song, H., & Vasilakos, A. V. (2017). Exploring data validity in transportation systems for smart cities. *IEEE Communications Magazine*, *55*(5), 26–33.
- Lohstroh, M., Derler, P., & Sirjani, M. (2018). *Principles of Modeling - Essays Dedicated to Edward A. Lee on the Occasion of His 60th Birthday* | Marten Lohstroh | Springer. Springer International Publishing.
- Loshin, D. (2011). *The Practitioner's Guide to Data Quality Improvement*. Morgan Kaufmann.
- Luo, T., Huang, J., Kanhere, S. S., Zhang, J., & Das, S. K. (2019). Improving IoT Data Quality in Mobile Crowd Sensing: A Cross Validation Approach. *IEEE IoT J.*, *6*(3), 5651–5664.
URL <https://ieeexplore.ieee.org/document/8666717>
- Ma, X., Cao, R., & Jin, Y. (2019). Spatiotemporal Clustering Analysis of Bicycle Sharing System with Data Mining Approach. *Information*, *10*(5), 163.
- Mahanti, R. (2019). *Data Quality: Dimensions, Measurement, Strategy, Management, and Governance*. Quality Press.
- Mahmood, I., & Zubairi, J. A. (2019). Efficient waste transportation and recycling: Enabling technologies for smart cities using the internet of things. *IEEE Electrification Magazine*, *7*(3), 33–43.
- Malhotra, R. (2015). *Empirical research in software engineering: concepts, analysis, and applications*. CRC Press.
- Martin, O. (2018). *Bayesian Analysis with Python: Introduction to statistical modeling and probabilistic programming using PyMC3 and ArviZ*. Packt Publishing Ltd.
- Maydanchik, A. (2007). *Data quality assessment*. Technics publications.
- McKinney, W. (2017). *Python for Data Analysis, 2nd Edition*. Sebastopol, CA, USA: O'Reilly Media, Inc.
URL <https://www.oreilly.com/library/view/python-for-data/9781491957653>
- MetOffice (2019). National meteorological library and archive fact sheet 14 - microclimates.
URL https://www.metoffice.gov.uk/binaries/content/assets/metofficegovuk/pdf/research/library-and-archive/library/publications/factsheets/factsheet_14-microclimates.pdf

- Micic, N., Neagu, D., Campean, F., & Zadeh, E. H. (2017). Towards a data quality framework for heterogeneous data. In *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, (pp. 155–162).
- Minerva, R., Biru, A., & Rotondi, D. (2015). Towards a definition of the internet of things (iot). *IEEE Internet Initiative*, 1(1), 1–86.
- Minoli, D., Sohraby, K., & Occhiogrosso, B. (2017). IoT Considerations, Requirements, and Architectures for Smart Buildings—Energy Optimization and Next-Generation Building Management Systems. *IEEE IoT J.*, 4(1), 269–283.
- Mois, G., Sanislav, T., & Folea, S. C. (2016). A cyber-physical system for environmental monitoring. *IEEE Transactions on Instrumentation and Measurement*, 65(6), 1463–1471.
- Montgomery, D. C., Jennings, C. L., & Kulahci, M. (2015). *Introduction to time series analysis and forecasting*. John Wiley & Sons.
- Moolayil, J., Moolayil, J., & John, S. (2019). *Learn Keras for Deep Neural Networks*. Springer.
- Möller, D. P. F. (2016). Systems and Software Engineering. In *Guide to Computing Fundamentals in Cyber-Physical Systems: Concepts, Design Methods, and Applications*, (pp. 235–305). Springer, Cham.
- Mosley, M., International, D., Brackett, M., & Earley, S. (2009). *The DAMA Guide to the Data Management Body of Knowledge*. Technics Publications, LLC.
URL https://books.google.co.uk/books?id=_0uMtAEACAAJ
- Müller, A. C., Guido, S., et al. (2016). *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc."
- Münch, J., Armbrust, O., Kowalczyk, M., & Soto, M. (2012). Empirical studies. In *Software Process Definition and Management*, (pp. 177–186). Springer.
- Mylavarapu, G., Thomas, J. P., & Viswanathan, K. A. (2019). An automated big data accuracy assessment tool. In *2019 IEEE 4th International Conference on Big Data Analytics (ICBDA)*, (pp. 193–197).
- Naik, D. R., Das, L. B., & Bindiya, T. (2018). Wireless sensor networks with zigbee and wifi for environment monitoring, traffic management and vehicle monitoring in smart cities. In *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, (pp. 46–50). IEEE.
- Nesa, N., Ghosh, T., & Banerjee, I. (2018). Outlier detection in sensed data using statistical learning models for iot. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, (pp. 1–6).

- Nguyen, T. H., Nunavath, V., & Prinz, A. (2014). Big data metadata management in smart grids. In *Big data and internet of things: A Roadmap for Smart Environments*, (pp. 189–214). Springer.
- Okafor, N. U., Alghorani, Y., & Delaney, D. T. (2020). Improving Data Quality of Low-cost IoT Sensors in Environmental Monitoring Networks Using Data Fusion and Machine Learning Approach. *ICT Express*, 6(3), 220–228.
- Otunba, R., Lin, J., & Senin, P. (2014). Mbpd: Motif-based period detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, (pp. 793–804). Springer.
- Pachepsky, Y., & Rawls, W. J. (2004). *Development of pedotransfer functions in soil hydrology*, vol. 30. Elsevier.
- Pan, M., Wang, J., Errapotu, S. M., Zhang, X., Ding, J., & Han, Z. (2019). *Big Data Privacy Preservation for Cyber-Physical Systems*. Springer.
- Pandas (2020). Merge, join, concatenate and compare — pandas 1.2.0 documentation. [Online; accessed 8. Jan. 2021].
URL https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html
- Paparrizos, J., & Gravano, L. (2016). k-Shape: Efficient and Accurate Clustering of Time Series. *SIGMOD Rec.*, 45(1), 69–76.
- Patel, A. R., Azadi, S., Babaei, M. H., Mollaei, N., Patel, K. L., & Mehta, D. R. (2018). Significance of robotics in manufacturing, energy, goods and transport sector in internet of things (iot) paradigm. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCCUBEA)*, (pp. 1–4).
- Pattanavijit, N., Vateekul, P., & Sarinnapakorn, K. (2015). A linear-clustering algorithm for controlling quality of large scale water-level data in thailand. In *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, (pp. 269–274).
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Peng, B., Shang, F., Wang, Y., Chen, G., Zhou, Z., & He, L. (2019). Research on data quality detection technology based on ubiquitous state grid internet of things platform. In *2019 IEEE 3rd International Electrical and Energy Conference (CIEEC)*, (pp. 1018–1023).
- Perez-Castillo, R., Carretero, A. G., Rodriguez, M., Caballero, I., Piattini, M., Mate, A., Kim, S., & Lee, D. (2018). Data quality best practices in iot environments. In *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, (pp. 272–275).

- Pernici, B., & Scannapieco, M. (2003). *Data Quality in Web Information Systems*, (pp. 48–68). Berlin, Heidelberg: Springer Berlin Heidelberg.
URL https://doi.org/10.1007/978-3-540-39733-5_3
- Pelech-Pilichowski, T. (2018). On adaptive prediction of nonstationary and inconsistent large time series data. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, (pp. 1260–1265).
- Pipino, L. L., Lee, Y. W., & Wang, R. Y. (2002). Data quality assessment. *Commun. ACM*, *45*(4), 211–218.
- Platzer, A. (2019). *Logical Foundations Of Cyber-Physical Systems..* Springer.
- Prathiba, B., Sankar, K. J., & Sumalatha, V. (2016). Enhancing the data quality in wireless sensor networks — a review. In *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, (pp. 448–454).
- Puiu, D., Barnaghi, P., Tönjes, R., Kümper, D., Ali, M. I., Mileo, A., Xavier Parreira, J., Fischer, M., Kolozali, S., Farajidavar, N., Gao, F., Iggena, T., Pham, T., Nechifor, C., Puschmann, D., & Fernandes, J. (2016). Citypulse: Large scale data analytics framework for smart cities. *IEEE Access*, *4*, 1086–1108.
- R., G., R., B., & P., K. (2020). Faulty-data detection and data quality measure in cyber–physical systems through Weibull distribution. *Comput. Commun.*, *150*, 262–268.
- Rager, S. T., Ciftcioglu, E. N., Ramanathan, R., La Porta, T. F., & Govindan, R. (2018). Scalability and satisfiability of quality-of-information in wireless networks. *IEEE/ACM Transactions on Networking*, *26*(1), 398–411.
- Raschka, S., & Mirjalili, V. (2017). *Python machine learning*. Packt Publishing Ltd.
- Rathore, M. M., Ahmad, A., Paul, A., & Jeon, G. (2015). Efficient graph-oriented smart transportation using internet of things generated big data. In *2015 11th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, (pp. 512–519).
- Ratner, B. (2017). *Statistical and Machine-Learning Data Mining, Third Edition: Techniques for Better Predictive Modeling and Analysis of Big Data, Third Edition*. Chapman & Hall/CRC: Chapman & Hall/CRC.
- Rawat, D. B., Rodrigues, J. J. P. C., & Stojmenovic, I. (2015). *Cyber-Physical Systems: From Theory to Practice*. USA: CRC Press, Inc.
- Rhee, S. (2019). Cyber-physical systems/internet of things for smart cities.
URL <https://www.nist.gov/programs-projects/cyber-physical-systemsinternet-things-smart-cities>

- Robbins, D. E., & Tanik, M. M. (2013). Cyber-Physical Ecosystems: App-Centric Software Ecosystems in Cyber-Physical Environments. In *Applied Cyber-Physical Systems*, (pp. 141–147). Springer, New York, NY.
- Rula, A., Maurino, A., & Batini, C. (2016). *Data Quality Issues in Linked Open Data*, (pp. 87–112). Cham: Springer International Publishing.
URL https://doi.org/10.1007/978-3-319-24106-7_4
- Sailhan, F., Delot, T., Pathak, A., Puech, A., & Roy, M. (2010). Dependable sensor networks. In *Atelier sur la Gestion des Données dans les Systèmes d'Information Pervasifs (GEDSIP) au sein de la conférence Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID)*, (pp. 1–15).
URL <https://hal.archives-ouvertes.fr/hal-01125818/>
- Salvador, S., & Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.*, 11(5), 561–580.
- Sanislav, T., & Miclea, L. (2012). Cyber-Physical Systems - Concept, Challenges and Research Areas. *Journal of Control Engineering and Applied Informatics*, 14(2), 28–33.
URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.472.8858&rep=rep1&type=pdf>
- Sanislav, T., Mois, G., Folea, S., Miclea, L., Gambardella, G., & Prinetto, P. (2014). A cloud-based cyber-physical system for environmental monitoring. In *2014 3rd Mediterranean Conference on Embedded Computing (MECO)*, (pp. 6–9).
- Santos, J., Wauters, T., Volckaert, B., & De Turck, F. (2017). Resource provisioning for iot application services in smart cities. In *2017 13th International Conference on Network and Service Management (CNSM)*, (pp. 1–9). IEEE.
- Sathe, S., Papaioannou, T. G., Jeung, H., & Aberer, K. (2013). A survey of model-based sensor data acquisition and management. In *Managing and mining sensor data*, (pp. 9–50). Springer.
- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2015). *Systems Analysis and Design in a Changing World*. Boston, MA, USA: Cengage Learning.
- Scannapieco, M., Missier, P., & Batini, C. (2005). Data quality at a glance. *Datenbank-Spektrum*, 14(January), 6–14.
- Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., & Grafberger, A. (2018). Automating large-scale data quality verification. *Proc. VLDB Endow.*, 11(12), 1781–1794.
URL <https://doi.org/10.14778/3229863.3229867>
- Seabold, S., & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.
- Seaman, C. B. (2008). Qualitative Methods. In *Guide to Advanced Empirical Software Engineering*, (pp. 35–62). London, England, UK: Springer.

- Sebastian-Coleman, L. (2013). *Measuring Data Quality for Ongoing Improvement*. Morgan Kaufmann.
- Sha, K., & Shi, W. (2008). Consistency-driven data quality management of networked sensor systems. *J. Parallel Distrib. Comput.*, 68(9), 1207–1221.
- Sha, K., & Zeadally, S. (2015). Data Quality Challenges in Cyber-Physical Systems. *J. Data and Information Quality*, 6(2-3), 1–4.
- Shewhart, W. A., & Deming, W. E. (1986). *Statistical method from the viewpoint of quality control*. Courier Corporation.
- Shih, C.-S., Chou, J.-J., Reijers, N., & Kuo, T.-W. (2016). Designing CPS/IoT applications for smart buildings and cities. *IET Cyber-Phys. Syst.: Theor. Appl.*, 1(1), 3–12.
- Shrivastava, S., Patel, D., Bhamidipaty, A., Gifford, W. M., Siegel, S. A., Ganapavarapu, V. S., & Kalagnanam, J. R. (2019). Dqa: Scalable, automated and interactive data quality advisor. In *2019 IEEE International Conference on Big Data (Big Data)*, (pp. 2913–2922).
- Shukla, S., Balachandran K, & Sumitha V S (2016). A framework for smart transportation using big data. In *2016 International Conference on ICT in Business Industry Government (ICTBIG)*, (pp. 1–3).
- Siddesh, G. M., Deka, G. C., Srinivasa, K. G., & Patnaik, L. M. (2015). *Cyber-physical systems: a computational perspective*. CRC Press.
- Singh, D. (2003). *Practical Statistics 2 Vols. Set*. Atlantic Publishers & Dist.
- Sinnott, R. (1984). Virtues of the Haversine. *undefined*.
URL <https://www.semanticscholar.org/paper/Virtues-of-the-Haversine-Sinnott/d1761591716859275573d4d315c973f2dbc26eae>
- Smalheiser, N. (2017). *Data Literacy*. Cambridge, MA, USA: Academic Press.
URL <https://www.elsevier.com/books/data-literacy/smalheiser/978-0-12-811306-6>
- Smarsly, K., Theiler, M., & Dragos, K. (2017). Ifc-based modeling of cyber-physical systems in civil engineering. In *Proceedings of the 24th International Workshop on Intelligent Computing in Engineering (EG-ICE)*. Nottingham, UK, vol. 7.
- Smartsantander (2021). Santander facility.
URL <https://www.smartsantander.eu/index.php/testbeds/item/132-santander-summary>
- Sohraby, K., Minoli, D., & Znati, T. (2007). *Wireless Sensor Networks: Technology, Protocols, and Applications*. Wiley.
URL <https://www.wiley.com/en-gb/exportProduct/pdf/9780470112755>

- Song, Z., Sun, Y., Wan, J., & Liang, P. (2017). Data quality management for service-oriented manufacturing cyber-physical systems. *Comput. Electr. Eng.*, *64*, 34–44.
- Sta, H. B. (2019). Strategy for evaluation the data in the context of smart cities: Case study of transport system. In *2019 IEEE International Smart Cities Conference (ISC2)*, (pp. 611–618).
- Staron, M. (2020). *Action Research in Software Engineering*. Cham, Switzerland: Springer International Publishing.
- SURI, N. M. R., Murty, M. N., & Athithan, G. (2019). *Outlier detection: techniques and applications*. Springer.
- Suri, N. R., Athithan, G., et al. (2019). Research issues in outlier detection. In *Outlier Detection: Techniques and Applications*, (pp. 29–51). Springer.
- Swamynathan, M. (2017). *Mastering Machine Learning with Python in Six Steps*. New York, NY, USA: Apress.
- Swamynathan, M. (2019). *Mastering machine learning with python in six steps: A practical implementation guide to predictive data analytics using python*. Apress.
- Tao, F., Qi, Q., Liu, A., & Kusiak, A. (2018). Data-driven smart manufacturing. *J. Manuf. Syst.*, *48*, 157–169.
- Tilley, S., & Rosenblatt, H. J. (2016). *Systems analysis and design*. Nelson Education.
- Tobler, W. R. (1970). A computer movie simulating urban growth in the detroit region. *Economic geography*, *46*(sup1), 234–240.
 URL <https://web.archive.org/web/20190308014451/http://pdfs.semanticscholar.org/ea5/eefedd4fa34b7de7448c0c8e0822e9fdf956.pdf>
- Togneri, R., Camponogara, G., Soininen, J., & Kamienski, C. (2019). Foundations of data quality assurance for iot-based smart applications. In *2019 IEEE Latin-American Conference on Communications (LATINCOM)*, (pp. 1–6).
- Tomescu, D., Heiman, A., & Badescu, A. (2019). An automatic remote monitoring system for large networks. In *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, (pp. 71–73).
- Törngren, M., Asplund, F., Bensalem, S., McDermid, J., Passerone, R., Pfeifer, H., Sangiovanni-Vincentelli, A., & Schätz, B. (2017). Characterization, analysis, and recommendations for exploiting the opportunities of cyber-physical systems. *Cyber-Physical Systems Foundations, Principles and Applications*, (pp. 3–14).
 URL <https://www.sciencedirect.com/science/article/pii/B9780128038017000018>
- Vacca, J. R. (2015). *Handbook of sensor networking: advanced technologies and applications*. CRC Press.

- Vaidya, S., Ambad, P., & Bhosle, S. (2018). Industry 4.0 – A Glimpse. *Procedia Manuf.*, 20, 233–238.
- Včelák, J., Vodička, A., Maška, M., & Mrňa, J. (2017). Smart building monitoring from structure to indoor environment. In *2017 Smart City Symposium Prague (SCSP)*, (pp. 1–5). IEEE.
- Walia, J., Walia, A., Lund, C., & Arefi, A. (2019). The characteristics of smart energy information management systems for built environments. In *2019 IEEE 10th International Workshop on Applied Measurements for Power Systems (AMPS)*, (pp. 1–6).
- Wand, Y., & Wang, R. Y. (1996). Anchoring data quality dimensions in ontological foundations. *Commun. ACM*, 39(11), 86–95.
- Wang, J., Cao, Z., Mao, X., Li, X.-Y., & Liu, Y. (2015). Towards energy efficient duty-cycled networks: analysis, implications and improvement. *IEEE Transactions on Computers*, 65(1), 270–280.
- Wang, J., Wu, Y., Hsu, H.-H., & Cheng, Z. (2017). Spatial big data analytics for cellular communication systems. In *Big Data Analytics for Sensor-Network Collected Intelligence*, (pp. 153–166). Elsevier.
- Wang, Q., & Bu, S. (2020). Deep learning enhanced situation awareness for high renewable-penetrated power systems with multiple data corruptions. *IET Renewable Power Gener.*, 14(7), 1134–1142.
- Wang, R. Y., Kon, H. B., & Madnick, S. E. (1993). Data quality requirements analysis and modeling. In *Proceedings of IEEE 9th International Conference on Data Engineering*, (pp. 670–677). IEEE.
- Wang, R. Y., & Strong, D. M. (1996). Beyond Accuracy: What Data Quality Means to Data Consumers. *Journal of Management Information Systems*, 12(4), 5–33.
- Wang, X., Smith, K., & Hyndman, R. (2006). Characteristic-Based Clustering for Time Series Data. *Data Min. Knowl. Disc.*, 13(3), 335–364.
- Wang, X., Wang, X., & Wilkes, M. (2021). Unsupervised fraud detection in environmental time series data. In *New Developments in Unsupervised Outlier Detection*, (pp. 257–277). Springer.
- Williams, C. K., & Rasmussen, C. E. (2006). *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA.
URL <http://www.gaussianprocess.org/gpml>
- Williams, D., & Tang, H. (2020). Data Quality Management for Industry 4.0: A Survey - ProQuest. [Online; accessed 13. Sep. 2020].
URL <https://search.proquest.com/docview/2386939130?pq-origsite=gscholar&fromopenview=true>

- Wing, J. (2008). Cyber-physical systems. *From Computing Research News*, 20.
URL <http://lazowska.cs.washington.edu/initiatives/WingCRN.pdf>
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). Introduction. In *Experimentation in Software Engineering*, (pp. 3–8). Berlin, Germany: Springer.
- Wu, F.-J., Luo, T., & Tan, H. P. (2016). Case studies of wsn-cps applications. *Cyber-Physical System Design with Sensor Networking Technologies*, 2, 269.
- Xie, L., Shi, Y., Hou, Y. T., Lou, W., Sherali, H. D., & Zhou, H. (2014). Rechargeable sensor networks with magnetic resonant coupling. In *Rechargeable Sensor Networks: Technology, Theory, and Application: Introducing Energy Harvesting to Sensor Networks*, (pp. 31–68). Citeseer.
- Xinrui, Y., Lei, W., & Ruiyi, L. (2019). Data quality evaluation of chinese wind profile radar network in 2018. In *2019 International Conference on Meteorology Observations (ICMO)*, (pp. 1–4).
- Yang, J., Han, Y., Wang, Y., Jiang, B., Lv, Z., & Song, H. (2020). Optimization of real-time traffic network assignment based on IoT data using DBN and clustering model in smart city. *Future Gener. Comput. Syst.*, 108, 976–986.
- Yin, R. K. (2017). *Case study research and applications: Design and methods*. Sage publications.
- Zanni, A. (2015). Cyber-physical systems and smart cities.
URL <https://developer.ibm.com/articles/ba-cyber-physical-systems-and-smart-cities-iot>
- Zelkowitz, M. V., & Wallace, D. R. (1998). Experimental models for validating technology. *Computer*, 31(5), 23–31.
- Zemicheal, T., & Dietterich, T. G. (2019). *Anomaly detection in the presence of missing values for weather data quality control*. New York, NY, USA: Association for Computing Machinery.
- Zhang, L. (2015). Multi-view, multi-domain, and multi-paradigm approaches for specification and modeling of big data driven cyber-physical systems.
URL <https://api.semanticscholar.org/CorpusID:63432037>
- Zhang, Q., Duan, R., Wang, J., & Cui, Y. (2019). Smart building environment monitoring based on gaussian process. In *2019 International Conference on Control, Automation and Information Sciences (ICCAIS)*, (pp. 1–6). IEEE.
- Zhang, Y., Duan, W., & Wang, F. (2011). Architecture and real-time characteristics analysis of the cyber-physical system. In *2011 IEEE 3rd International Conference on Communication Software and Networks*, (pp. 317–320).

- Zhiping, Z., Nannan, S., & Xuebo, Z. (2014). A novel authentication protocol for mobile nodes in multi-base-station wireless sensor network. In *ICINS 2014 - 2014 International Conference on Information and Network Security*, (pp. 52–59).
- Zhou, H., Yu, K., Lee, M., & Han, C. (2018). The application of last observation carried forward method for missing data estimation in the context of industrial wireless sensor networks. In *2018 IEEE Asia-Pacific Conference on Antennas and Propagation (APCAP)*, (pp. 1–2).
- Zhuang, Y., & Chen, L. (2006). In-network outlier cleaning for data collection in sensor networks. In *CleanDB*.

Appendices

Appendix A

Technical and Implementation Details

A.1 Sensor Nodes Anatomy and Data Quality

Sensor nodes network is a group of few to thousands of sensor nodes connected to an external server and each other via wired or wireless medium (Forster 2016). Applications such as environment monitoring typically involve numerous sensor nodes deployed in a vast geographical area to form a large-scale sensor nodes network (Benyuan Liu & Towsley 2004). Although sensor nodes' anatomy is not the focus of this research, it is crucial to understand how sensor nodes work to identify a significant source of data quality issues in large-scale CPSs. The anatomy of sensor nodes network will be explored very briefly in this section to understand some of the technical characteristics of sensor nodes and the impact of its design on its quality of service and performance in sensor networks and its different applications.

A.1.1 Hardware Components

This section explores some of the sensor nodes concepts and components. Although there is a wide variety in the sensor nodes types, sizes, and functionalities, some common characteristics and components are briefly explored in this section. Figure A.1 shows the main components of a typical sensor node which consist of:

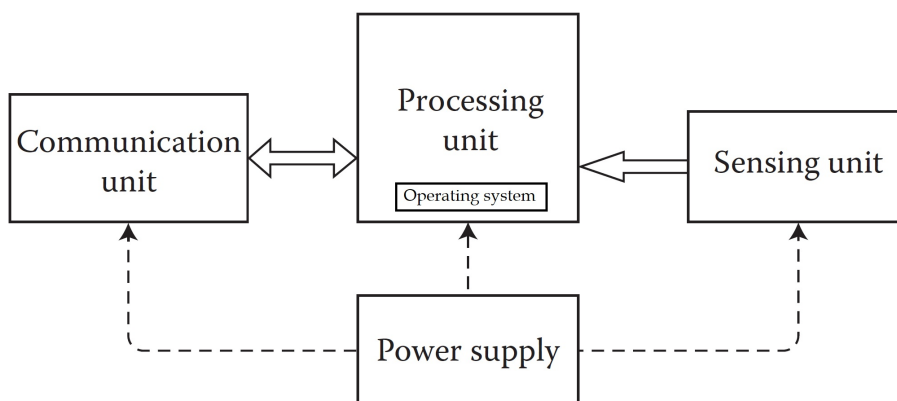


Figure A.1: The main components of a typical wireless sensor node (Vacca 2015 p. 2-1).

Micro-controller: is a small computer unit, with memory, processor and a general-purpose input/output ports. The Micro-controller is responsible for all of the computational processes on-board and also responsible for coordinating with external sensor nodes.

Sensing Unit (sensor): responsible for interacting with the environment and interpreting a physical reading into a signal that the system (the environmental monitoring system in this case) can read and understand.

Radio transceiver: is responsible for receiving and sending data. It encompasses a micro-controller responsible for controlling data packages, buffering, validating and implementing communication protocols. The radio transceiver is one of the most power-consuming components in the sensor node (Forster 2016).

Battery: the sensor node energy source. It can be a battery such as AAA or AA or other types of rechargeable batteries.

Reducing power consumption to prolong the battery life of sensor nodes is crucial to extend the sensor network lifetime mainly because sensor nodes usually deployed in hard-to-reach environments, and maintaining these devices might not be feasible (Xie et al. 2014). Sensor nodes power consumption relies on the amount of the provided electrical current. Assuming that the power supply voltage is constant, the sensor node has a sleeping mode that significantly reduces power consumption by decreasing or completely stopping some of its hardware components, known as the Duty-Cycle (Wang et al. 2015). The duty-cycle method increases sensor node energy efficiency via a binary counter to wake up the sensor node at specific time instants (Addabbo et al. 2019). Some sensor nodes rely on oscillators for tracking the time. Oscillators count the ticks after the sensor node reboot. Therefore, sensor nodes have no real global time (Forster 2016), causing one of the most common errors that affect the quality of data, especially in real-time systems (Liu et al. 2017).

A.1.2 Operating System

Sensor nodes' operating system is a programming code responsible for orchestrating and managing sensor nodes' hardware components such as microprocessor, memory, input-output devices and applications, as shown in Figure A.2 (Forster 2016).

Typically, sensor nodes are small-sized with limited battery power capacity, limited computing power and limited communication capability. Sensor node operating systems must efficiently manage all the available resources to bridge the gap between the limitation in resources and the required applications' complexity (Amjad et al. 2016), (Dong et al. 2010). TinyOS, Contiki and LiteOS are examples of sensor nodes operating systems responsible for resource management and perform the sensor node's required tasks (Amjad et al. 2016). For example, to reduce

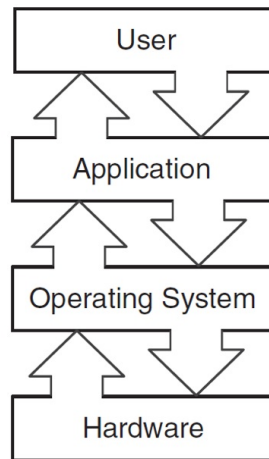


Figure A.2: A representation of sensor-nodes components interaction chain (Forster 2016).

power consumption in wireless sensor nodes, a light-weight adaptive duty-cycling (LAD) protocol was proposed and tested on a large-scale sensor network. The test results showed a significant reduction in energy consumption, while the practical implementation proved the proposed solution's effectiveness (Wang et al. 2015).

A.2 The Technical Details of the Local Sensor Node Network

Monnit Wireless Temperature Sensor



Technical Overview

General Description

The RF Wireless Temperature Sensor uses a type NTC thermistor to measure temperature.

Features

- Accurate to $\pm 1^\circ\text{C}$ ($\pm 1.8^\circ\text{F}$)
- Increased accuracy by user calibration to $\pm 0.25^\circ\text{C}$ ($\pm 0.45^\circ\text{F}$)
- Free iMonnit basic online wireless sensor monitoring and notification system to configure sensors, view data and set alerts via SMS text and email.

Principle of Operation

The Monnit Wireless Temperature Sensor outputs the ambient temperature in degrees Fahrenheit. It is programmed to sleep for a user-given time interval (heartbeat) and then wakeup, send power to the NTC Thermistor and wait for it to stabilize, and convert the analog data, mathematically compute the temperature and transmit the data to the gateway. To stay within the abilities of the processor, the temperature is computed off a data table provided by the manufacturer. To reduce error, a variable resistor configuration is implemented over specified temperature ranges.

Power Options

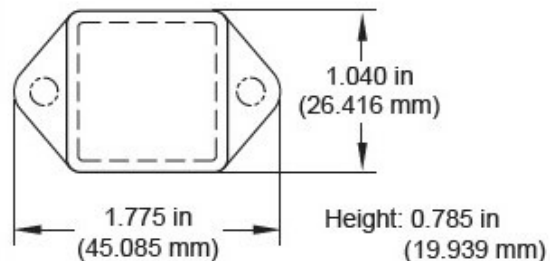
Sensors are powered by a replaceable 3.0 V coin cell battery. Optional AA battery powered sensors are available. The AA version of these sensors are larger in size (3" [L] x 2.1" [W] x 1.2" [H]) and include two long-life AA batteries.

It is recommended that unless you are using the AA battery solution, you set heartbeat to no faster than one hour to preserve battery life.

Monnit Sensor Core Specifications


- Power: Replaceable 3.0 V coin cell battery
- Communication: RF 900, 920, 868 and 433 MHz
- Dimensions: 1.775" x 1.040" x 0.785"
- Antenna: 4" wire antenna
- Operating Temperature: -7° to 60°C (20° to 140°F)
- Device Range: 250 - 300 ft. non-line-of-sight*
- Battery Life: At 1 hour heartbeat setting, coin cell battery will last ~ 1-2 years.**

* Actual range may vary depending on environment.
** Battery life is determined by sensor reporting frequency and other variables.



Applications

- Ambient Temperature Monitoring
- Environmental Monitoring
- Smart Machines & Smart Structures
- HVAC Operation & Testing
- Data Center Monitoring

Technical Specifications	
Supply Voltage	2.0 - 3.6 VDC *
Current Consumption	0.7 μ A (sleep mode) 2 mA (radio idle/off mode) 2 mA (measurement mode) 25 mA (radio RX mode) 35 mA (radio TX mode)
Operating Temperature Range (Board Circuitry and Coin Cell)	-7°C to +60°C (20°F to +140°F)**
Optimal Battery Temperature Range (Coin Cell)	+10°C to +50°C (+50°F to +122°F)
Thermistor Temperature Range (Thermistor Only)	-40°C to +125°C (-40°F to +257°F) (Limited to Main Unit Circuitry, -7°C to +60°C unless wire leads are being used.)
Accuracy @ 25°C	+/- 1% (1° C or 1.8° F)
User Calibrated Accuracy	+/- 0.25° C (± 0.45° F)
Time Constant @ 25°C	30 sec
Certifications	 900 MHz product; FCC ID: ZTL- RFSC1 and IC: 9794A-RFSC1. 920 MHz product; ARIB STD-T108 R210-103733. 868 and 433 MHz product tested and found to comply with: CISPR 22:2008-09 / EN 55022:2010 - Class B and ETSI EN 300 220-2 V2.4.1 (2012-05).

* Hardware can not withstand negative voltage. Please take care when connecting a power device.

** At temperatures above 100°C, it is possible for the board circuitry to lose programmed memory.

Caution/Notice:

This product is designed for application in an ordinary environment (normal room temperature, humidity and atmospheric pressure). Do not use this sensor under the following conditions as these factors can deteriorate the product characteristics and cause failures and burn-out.

- Corrosive gas or deoxidizing gas - chlorine gas, hydrogen sulfide gas, ammonia gas, sulfuric acid gas, nitric oxides gas, etc.)
- Volatile or flammable gas.
- Dusty conditions.
- Under low or high pressure.
- Wet or excessively humid locations.
- Places with salt water, oils chemical liquids or organic solvents.
- Where there are excessively strong vibrations.
- Other places where similar hazardous conditions exist.

Use this product within the specified temperature range. Higher temperature may cause deterioration of the characteristics or the material quality of this product.

MONNIT

For more information about our products or to place an order, please contact our sales department at 801-561-5555.

Visit us on the web at www.monnit.com.

Monnit Corporation
4403 South 500 West
Murray, UT 84123
801-561-5555
www.monnit.com



WIRELESS SENSOR ADAPTER

Part # MNG-8-WSA-USB

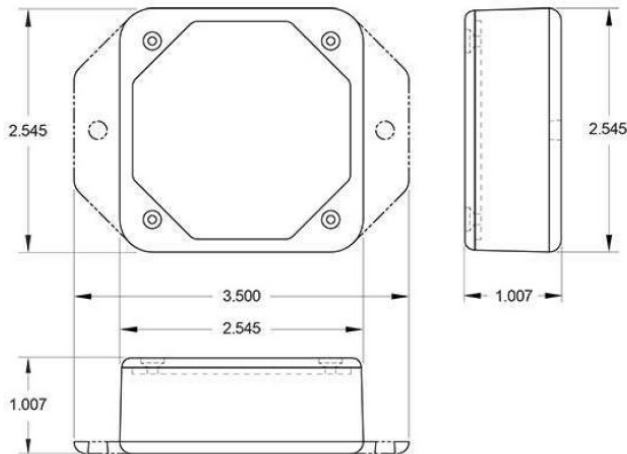
Region | Frequency Europe - 868 MHz ↗

Wireless Sensor Adapter Specifications

Device Range	250 - 300 ft. non-line-of-sight device range*
Enclosure Material	ABS
Compatibility	Windows 10, 8.1, 8, 7 and Vista
Device Memory	14,336 up messages (to gateway/server) 2,048 down messages (to sensors)
Power Supply	Powered through USB output of PC Optional - External Power with Power Supply Optional - RS232 Power (Call for more information)
Sensing Element	100 mA
Max Operating Temperature	-40°C to +85°C (-40°F to +185°F)
Connection Type	Universal Serial Bus (USB) Optional RS232 (115200 baudrate, 8 data bits, no parity, 1 stop bit, DB9 connector)

868 MHz Specifications

Operating Frequency	868 MHz Operating Frequency
Certifications	CE Certified and Complies with CISPR 22:2008-09 / EN 55022:2010 - Class B and ETSI EN 300 220-2 V2.4.1 (2012-05)



Wireless Sensor Adapter For Long Range Communication to PC

[Services \(https://www.monnit.com/services/\)](https://www.monnit.com/services/)
[Partnerships \(https://www.monnit.com/partner/\)](https://www.monnit.com/partner/)
[Support \(https://www.monnit.com/support/frequently-asked-questions/\)](https://www.monnit.com/support/frequently-asked-questions/)

About Us

[Monnit \(https://www.monnit.com/company/\)](https://www.monnit.com/company/)
[News \(https://www.monnit.com/blog/\)](https://www.monnit.com/blog/)
[Webinars \(https://www.monnit.com/webinars/\)](https://www.monnit.com/webinars/)
[Awards \(https://www.monnit.com/awards-and-recognition/\)](https://www.monnit.com/awards-and-recognition/)
[Careers \(https://www.monnit.com/company/careers/\)](https://www.monnit.com/company/careers/)

Legal

[Terms \(https://www.monnit.com/company/legal/terms/\)](https://www.monnit.com/company/legal/terms/)
[Privacy \(https://www.monnit.com/company/legal/privacy/\)](https://www.monnit.com/company/legal/privacy/)
[Returns \(https://www.monnit.com/company/legal/returns/\)](https://www.monnit.com/company/legal/returns/)
[Cookies \(https://www.monnit.com/company/legal/cookies/\)](https://www.monnit.com/company/legal/cookies/)
[Data Retention \(https://www.monnit.com/company/legal/data-retention/\)](https://www.monnit.com/company/legal/data-retention/)

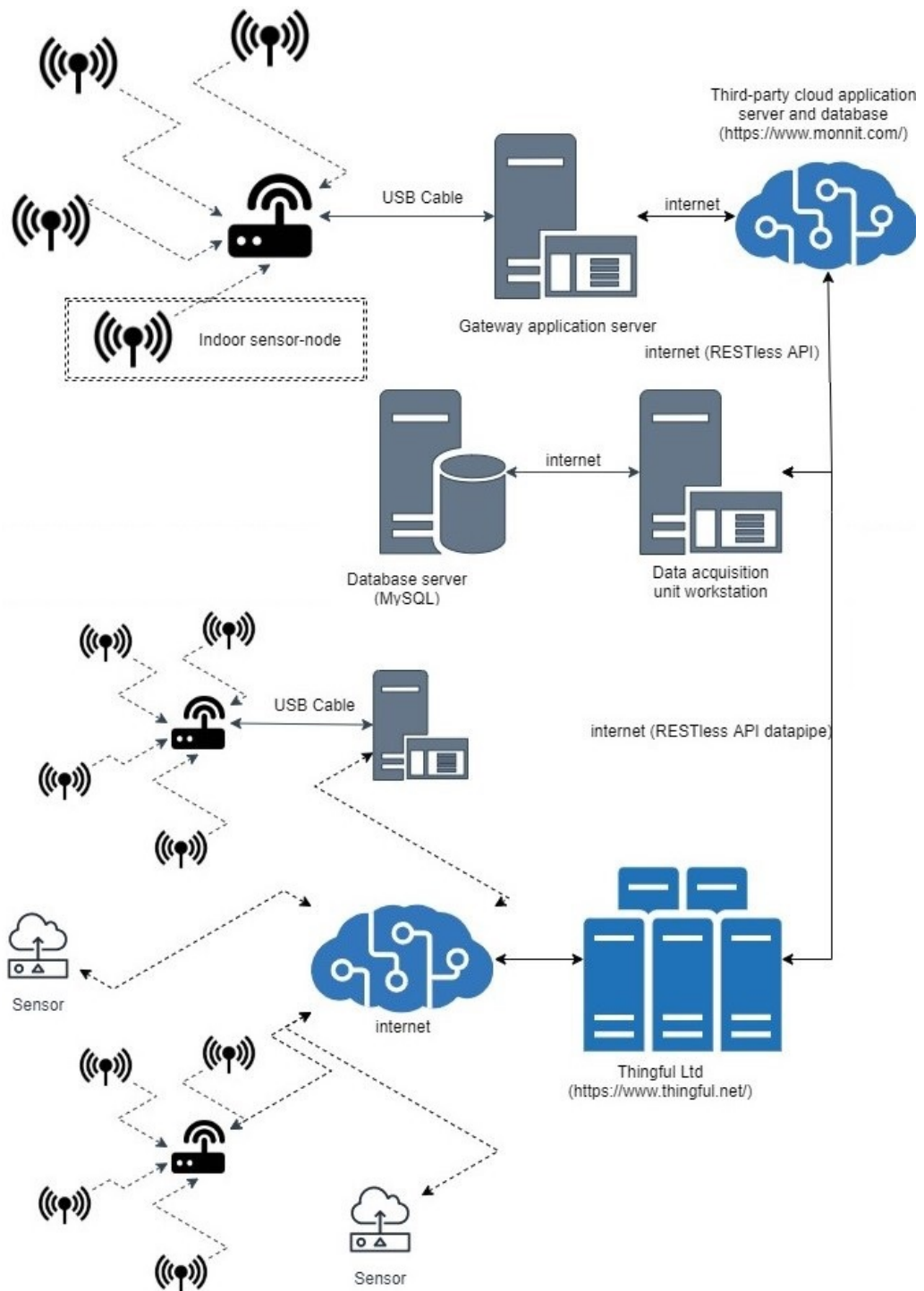


Figure A.3: The full network topology of the sensor node networks utilised in this research.

A.3 Technical Details of The Data Acquisition Unit

The data acquisition software is the core component of the data acquisition unit. It was developed using Java and comprised six components (classes), as shown in the class diagram Figure 4.11. Each one of these classes has a specific role and combined these classes describe how the data acquisition software operates. The Java code scripts of the main classes of the data acquisition unit are as follows:

Listing A.1: The Java code scripts of the main classes of the data acquisition unit are as follows:

```
package RESTfulToSQL;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import org.json.JSONException;

public class Startup {
public static void main(String[] args) throws JSONException {
    System.out.println("Starting up ^_^");
    while (true) { timer.delay();
        try { Connection Myconn;
            Myconn = DriverManager.getConnection( "jdbc:mysql://161.76.xxx.xxx:3306/thingdb?" +
                "verifyServerCertificate=false&useSSL=true", "user name", "p.w");
            System.out.println
                ("Trying to fetch data from Thingful.Ltd data pipes  _-_- --> " + Myconn.isValid(0));
            Statement st = Myconn.createStatement();
            ResultSet rs = st.executeQuery
                ("SELECT Dp_id,Dp_table,Dp_API_token,Dp_sample_size, "
                + "Dp_details FROM th_datapipes where Dp_type = 'thingful'");
            rs.beforeFirst();
            while (rs.next()) { dataManager.collect(rs.getString("Dp_API_token"),
                rs.getString("Dp_table"), rs.getInt("Dp_sample_size"), "thingful");}
        } catch (SQLException e) { e.printStackTrace();}
        try { Connection Myconn;
            Myconn = DriverManager.getConnection("jdbc:mysql://161.76.xxx.xxx:3306/thingdb?" +
                "verifyServerCertificate=false&useSSL=true", "user name", "p.w");
            System.out.println
                ("Trying to fetch data from imonnit.Ltd REST API  _-_- --> " + Myconn.isValid(0));
```

```

Statement st = Myconn.createStatement();
ResultSet rs = st.executeQuery
        ("SELECT Dp_id,Dp_table,Dp_API_token,Dp_sample_size,"
        + "Dp_details FROM th_datapipes where Dp_type = 'imonnit'");
rs.beforeFirst();
while (rs.next()) { dataManager.collect(rs.getString("Dp_API_token"),
        rs.getString("Dp_table"), rs.getInt("Dp_sample_size"), "imonnit");}
} catch (SQLException e) {e.printStackTrace();}
}
}
}

```

```

package RESTfulToSQL;

import org.apache.http.HttpResponse;
import org.json.JSONArray;
import org.json.JSONException;

public class dataManager {
    public static void collect(String token, String tableName, int sampleSize,String AuthorType)
        throws JSONException {
        HttpResponse response = null;
        if (AuthorType == "thingful") {response = thingfulAuthorization.Authorize(token,sampleSize);}
        else if (AuthorType == "imonnit") {response = imonnitAuthorization.Authorize(token,sampleSize);}
        System.out.println("The Servers response: " + response.getStatusLine());
        if ( response.getStatusLine().getStatusCode() == 200 ) {
            JSONArray JA = DataStream.ToArray(response,AuthorType);
            if (JA.length() > 0) {
                System.out.print("The number of the collected readings is : " + JA.length());
                System.out.println(" --> Sending the data to the database...");
                String DbStatus = toDatabase.mysql(JA,tableName);
                System.out.println(DbStatus);
                System.out.println("\n");
            }
            else {System.out.println("No data have been collected !");}
        }
        else {System.out.println("Authentication Problem - Data Collection is not possible");}
    }
}
}
}
}
}

```

```

package RESTfulToSQL;

import org.apache.http.HttpResponse;

```

```

import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.HttpClientBuilder;
import java.io.IOException;

class thingfulAuthorization {String httpGet;
    public static HttpResponse Authorize(String token, int sampleSize) {
        System.out.println("Connecting the Datapipe . . . .");
        // the Http-client, that will send the request to Thingful
        HttpClient httpclient = HttpClientBuilder.create().build();
        HttpGet httpGet = new HttpGet(token + "?limit=" + sampleSize); // My data-pipe details
        // The httpGET request
        httpGet.addHeader("Thingful-Authorization", "Bearer a5550912-00be-4a25-xxxxxxx");
        // The token from Thingful.
        // To add the authorisation header to the request.
        HttpResponse response = null;
        try {response = httpclient.execute(httpGet);
        } catch (IOException e) {e.printStackTrace();}
        return response;
    }
}

```

```

package RESTfulToSQL;
import org.apache.http.HttpResponse;
import org.apache.http.ParseException;
import org.apache.http.util.EntityUtils;
import org.json.JSONArray;
import org.json.JSONException;
import java.io.IOException;
public class DataStream {
    public static JSONArray ToArray(HttpResponse response, String AuthorType)
        throws JSONException {
        String stringResponse, st = null;
        JSONArray ja = null;
        try {stringResponse = EntityUtils.toString(response.getEntity(), "UTF-8");
            if (AuthorType == "imonnit") {st = stringResponse.replace
                ("{"Method\":\"AccountRecentDataMessages\",\"Result\":", "");
                st = st.replace("}]", "]}");
            } else if (AuthorType == "thingful") {
                st = stringResponse.replace("data: ", "").replace("]\n" + "\n" + "[", ",");
            }
            ja = new JSONArray(st);
        } catch (ParseException | IOException e) {e.printStackTrace();} // The response as String,

```

```

    which you could be converted to a JSONObject.
    return (ja);
  }
}

```

```

package RESTfulToSQL;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Date;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

public class toDatabase {
    public static String mysql(JSONArray ja, String tableName) throws JSONException {
        int n = 0;
        try { Connection Myconn = DriverManager.getConnection(
            "jdbc:mysql://161.76.xxx.xxx:3306/thingdb?" +
            "verifyServerCertificate=false&useSSL=true", "user name", "p.w");
            System.out.println("The connection to the database is valid: " + Myconn.isValid(0));
            Myconn.setAutoCommit(true);
            n = ja.length();
            for (int i = 0; i < n; i++) // To GET INDIVIDUAL JSON OBJECT FROM JSON ARRAY
                {JSONObject jo = ja.getJSONObject(i);
                    PreparedStatement pstmt = Myconn.prepareStatement("insert into " + tableName +
                        " (date, json) values (?,?)");
                    pstmt.setString(1, new Date().toString());
                    pstmt.setString(2, String.valueOf(jo));
                    pstmt.execute();}
            System.out.println
                ("An insert statement has been executed, not verified ..0_0");
            Myconn.close();
            System.out.println("Closing the connection... connection to the database is valid: "
                + Myconn.isValid(0));
        } catch (SQLException e) {e.printStackTrace();}
        return "It's a good idea to check your Database ^_* ";
    }
}

```

The deployment environment diagram of the data acquisition software is illustrated in Figure A.4.

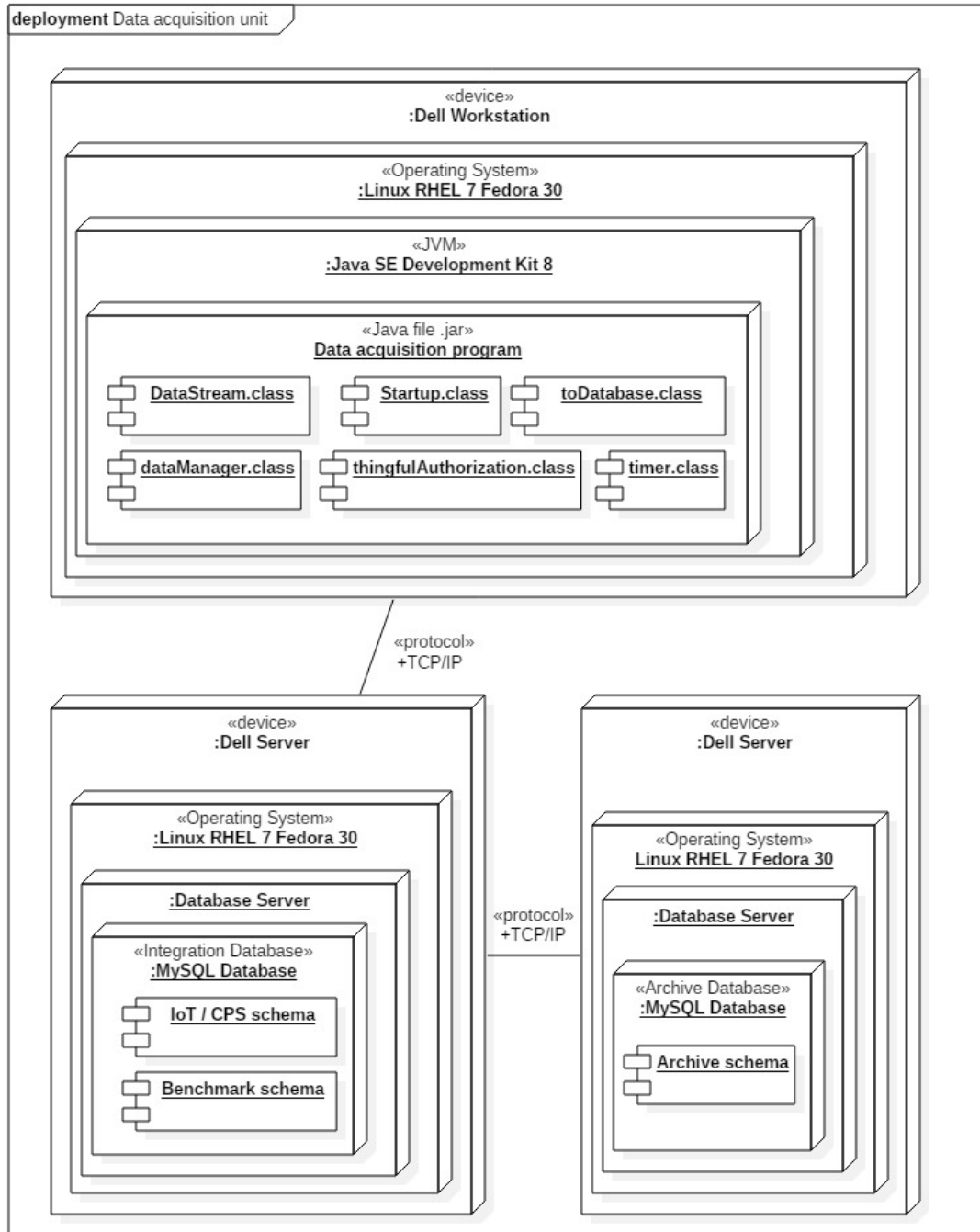


Figure A.4: The deployment diagram of the data acquisition software.

Listing A.2: The deployment code of the Java based data acquisitions unit over Linux.

```
\\ Source: 'https://unix.stackexchange.com/questions/143706/how-to-run-java-service-as-a-non-root-
user-on-centos'-6
\\ More details: "http://www.jcgonzalez.com/linux-java-service-wrapper-example"

sudo vi /etc/init.d/thingful
#!/bin/sh
SERVICE_NAME=thingful
PATH_TO_JAR=/home/.../thingful/Thingful.jar
PID_PATH_NAME=/tmp/Thingful-pid
case $1 in start)
    echo "Starting $SERVICE_NAME ..."
    if [ ! -f $PID_PATH_NAME ]; then
        nohup java -jar $PATH_TO_JAR /tmp 2>> /dev/null >> /dev/null &
        echo $! > $PID_PATH_NAME
        echo "$SERVICE_NAME started ..."
    else
        echo "$SERVICE_NAME is already running ..." fi ;; stop)
    if [ -f $PID_PATH_NAME ]; then
        PID=$(cat $PID_PATH_NAME);
        echo "$SERVICE_NAME stoping ..."
        kill $PID;
        echo "$SERVICE_NAME stopped ..."
        rm $PID_PATH_NAME
    else
        echo "$SERVICE_NAME is not running ..." fi ;; restart)
    if [ -f $PID_PATH_NAME ]; then
        PID=$(cat $PID_PATH_NAME);
        echo "$SERVICE_NAME stopping ...";
        kill $PID;
        echo "$SERVICE_NAME stopped ...";
        rm $PID_PATH_NAME
        echo "$SERVICE_NAME starting ..."
        nohup java -jar $PATH_TO_JAR /tmp 2>> /dev/null >> /dev/null &
        echo $! > $PID_PATH_NAME
        echo "$SERVICE_NAME started ..."
    else
        echo "$SERVICE_NAME is not running ..." fi ;; esac
```

The corresponding tables of the local database at the University of East London receive sensor nodes observations from the data acquisition unit as JSON objects.

The JSON parsing and the observations duplication prevention mechanisms are built-in inside these tables as triggers, as detailed in Section 4.1.3.3.

A.3.1 JSON Parsing and Duplication Prevention Trigger

Listing A.3: Following is an example of the JSON parsing and the duplication prevention trigger built-in as SQL script in the database JSON corresponding tables, using the "th_Air_quality_json" table as a case study.

```
CREATE DEFINER='user name'@'%' TRIGGER `th_Air_quality_json_AFTER_INSERT`  
  
AFTER INSERT ON `th_Air_quality_json`  
FOR EACH ROW BEGIN insert into th_Air_quality  
values (  
    new.ID,  
    STR_TO_DATE(concat(substr(SUBSTR(new.Date, 5),1,16),  
    substr(SUBSTR(new.Date, 5),21,4)),'%M %d %H:%i:%s %Y'),  
    SUBSTR(JSON_UNQUOTE(JSON_EXTRACT(new.JSON, '$.id')), 33, 8),  
    JSON_UNQUOTE(JSON_EXTRACT(new.json, '$.latitude')) ,  
    JSON_UNQUOTE(JSON_EXTRACT(new.json, '$.longitude')) ,  
    REPLACE(REPLACE(JSON_UNQUOTE(JSON_EXTRACT(new.json, '$.updatedAt')), 'T', ' '), 'Z', '') ,  
    CASE WHEN JSON_UNQUOTE(JSON_EXTRACT(new.json, '$.temperature')) = 'null' THEN  
    JSON_UNQUOTE(JSON_EXTRACT(new.json, '$."airTemperature,WeatherTemperature,AmbientTemperature  
"))  
    WHEN JSON_UNQUOTE(JSON_EXTRACT(new.json, '$.temperature')) = '' THEN  
    JSON_UNQUOTE(JSON_EXTRACT(new.json, '$."airTemperature,WeatherTemperature,AmbientTemperature  
"))  
    ELSE COALESCE(JSON_UNQUOTE(JSON_EXTRACT(new.json, '$.temperature')), 'null')  
    END ,  
    JSON_UNQUOTE(JSON_EXTRACT(new.json, '$."license.name"')) ,  
    JSON_UNQUOTE(JSON_EXTRACT(new.json, '$."provider.uri"')) ,  
    JSON_UNQUOTE(JSON_EXTRACT(new.json, '$."provider.name"')) ,  
    0 ,  
    STR_TO_DATE(concat(substr(SUBSTR(new.Date, 5),1,16),  
    substr(SUBSTR(new.Date, 5),21,4)),'%M %d %H:%i:%s %Y'))  
    ON DUPLICATE KEY UPDATE RowCount=RowCount+1,  
    ServerLastUpdate = STR_TO_DATE(concat(substr(SUBSTR(new.Date,5),1,16),  
    substr(SUBSTR(new.Date, 5),21,4)),'%M %d %H:%i:%s %Y') ;  
  
END
```


A.3.2 The Periodicity Analysis Rule Engine.

Listing A.4: The full SQL code and logic behind the periodicity analysis rule engine.

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = 'user name'@'% '
  SQL SECURITY DEFINER
VIEW 'HW_Dataset_Thingful_03' AS
  SELECT
    'HW_Dataset_Thingful'.'SensorID' AS 'SensorID',
    'HW_Dataset_Thingful'.'Latitude' AS 'Latitude',
    'HW_Dataset_Thingful'.'Longitude' AS 'Longitude',
    'HW_Dataset_Thingful'.'ServerDate' AS 'MessageDate',
    'HW_Dataset_Thingful'.'ServerLastUpdate' AS 'ServerLastUpdate',
    (TIMESTAMPDIFF(SECOND,
      'HW_Dataset_Thingful'.'ServerDate',
      'HW_Dataset_Thingful'.'ServerLastUpdate') / 60) AS 'Interval',
    'HW_Dataset_Thingful_th'.'Interval_threshold' AS 'I_threshold',
    (CASE
      WHEN
        (((TIMESTAMPDIFF(SECOND,
          'HW_Dataset_Thingful'.'ServerDate',
          'HW_Dataset_Thingful'.'ServerLastUpdate') / 60) / 'HW_Dataset_Thingful_th'.'
Interval_threshold') > 0)
        AND (((TIMESTAMPDIFF(SECOND,
          'HW_Dataset_Thingful'.'ServerDate',
          'HW_Dataset_Thingful'.'ServerLastUpdate') / 60) / 'HW_Dataset_Thingful_th'.'
Interval_threshold') <= 1))
      THEN
        'Good'
      WHEN
        (((TIMESTAMPDIFF(SECOND,
          'HW_Dataset_Thingful'.'ServerDate',
          'HW_Dataset_Thingful'.'ServerLastUpdate') / 60) / 'HW_Dataset_Thingful_th'.'
Interval_threshold') > 1)
        AND (((TIMESTAMPDIFF(SECOND,
          'HW_Dataset_Thingful'.'ServerDate',
          'HW_Dataset_Thingful'.'ServerLastUpdate') / 60) / 'HW_Dataset_Thingful_th'.'
Interval_threshold') <= 2))
      THEN
```

```

        'Timeliness'
    WHEN
        (((TIMESTAMPDIFF(SECOND,
            'HW_Dataset_Thingful'. 'ServerDate',
            'HW_Dataset_Thingful'. 'ServerLastUpdate') / 60) / 'HW_Dataset_Thingful_th'. '
Interval_threshold') >= 2)
        AND (((TIMESTAMPDIFF(SECOND,
            'HW_Dataset_Thingful'. 'ServerDate',
            'HW_Dataset_Thingful'. 'ServerLastUpdate') / 60) / 'HW_Dataset_Thingful_th'. '
Interval_threshold') < 3))
    THEN
        'Missing observation'
    WHEN
        (((TIMESTAMPDIFF(SECOND,
            'HW_Dataset_Thingful'. 'ServerDate',
            'HW_Dataset_Thingful'. 'ServerLastUpdate') / 60) / 'HW_Dataset_Thingful_th'. '
Interval_threshold') >= 3)
    THEN
        'Long outlier'
    WHEN
        (((TIMESTAMPDIFF(SECOND,
            'HW_Dataset_Thingful'. 'ServerDate',
            'HW_Dataset_Thingful'. 'ServerLastUpdate') / 60) / 'HW_Dataset_Thingful_th'. '
Interval_threshold') = 0)
    THEN
        'Duplicated observation'
    END) AS 'Consistency_status'
FROM
    ('HW_Dataset_Thingful'
    JOIN 'HW_Dataset_Thingful_th')

```

A.4 Configuration and programming details

A.4.1 Holt-Winters predictive model

Listing A.5: The configuration and programming details of Holt-Winters predictive model using Holt-Winters Python package provided by Statsmodels.org (Seabold & Perktold 2010).

```
import os
import time
from math import sqrt
import matplotlib.pyplot as plt
import pandas as pd
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.holtwinters import ExponentialSmoothing as HoltWinters

os.chdir('DataBase')
# The ideal dataset
df1 = pd.read_csv('HW-Dataset-Monnit.csv', skiprows=0)[['Date', 'Value']]
df1['Date'] = pd.to_datetime(df1['Date'], format='%d/%m/%Y %I:%M %p')
df1.columns = ['R_Date', 'Mo_Temp']
df1.set_index('R_Date', inplace=True)
df1.sort_index(inplace=True, ascending=True)
df1 = df1.resample('10T').mean().ffill()
# print(df1.head(5))
# The real-world dataset
df2 = pd.read_csv('HW-Dataset-Thingful.csv', skiprows=0)[['ServerDate', 'AmbientTemperature']]
df2['ServerDate'] = pd.to_datetime(df2['ServerDate'], format='%Y-%m-%d %H:%M:%S')
df2.columns = ['R_Date', 'Th_Temp']
df2.set_index('R_Date', inplace=True)
df2.sort_index(inplace=True, ascending=True)
df2 = df2.resample('10T').mean().ffill()
# print(df2.head(5))

hw = pd.concat([df1, df2], axis=1).reindex(df1.index)
hw = hw.dropna(how='any', axis=0)
res = sm.tsa.seasonal_decompose(hw.Mo_Temp.interpolate(), freq=144, model='additive') #
    Multiplicative additive
fig, axs = plt.subplots(4, 1, figsize=(24, 12), sharey=False)
axs[0].plot(hw.Mo_Temp)
```

```

axs[0].set_title('Temperature readings over time (Sensor ID = 493372 / Monnit)')
axs[0].set_ylabel('Temperature C')
res.trend.plot(ax=axs[1])
axs[1].set_title('Trend')
axs[1].set_ylabel('Temperature C')
res.seasonal.plot(ax=axs[2])
axs[2].set_title('Seasonality')
axs[2].set_ylabel('Temperature C')
res.resid.plot(ax=axs[3])
axs[3].set_title('Residuals')
axs[3].set_ylabel('Temperature C')
plt.show()

res = sm.tsa.seasonal_decompose(hw.Th_Temp.interpolate(), freq=144, model='additive') #
    Multiplicative additive
fig, axs = plt.subplots(4, 1, figsize=(24, 12), sharey=False)
axs[0].plot(hw.Th_Temp, c='r')
axs[0].set_title('Temperature readings over time (Sensor ID = jcw5m701 / Thingful)')
axs[0].set_ylabel('Temperature C')
res.trend.plot(ax=axs[1], c='r')
axs[1].set_title('Trend')
axs[1].set_ylabel('Temperature C')
res.seasonal.plot(ax=axs[2], c='r')
axs[2].set_title('Seasonality')
axs[2].set_ylabel('Temperature C')
res.resid.plot(ax=axs[3], c='r')
axs[3].set_title('Residuals')
axs[3].set_ylabel('Temperature C')
plt.show()

t_pred = 36
t_loop = 25
rms_list = []
for i in range(t_loop):
    t_start = 470 + i
    train0, test0 = hw.iloc[:t_start, 0], hw.iloc[t_start:t_start + t_pred + 1, 0] # +1 because it
        represent a real

    tm = time.process_time()
    model = HoltWinters(train0, seasonal_periods=144, trend='add', seasonal='add').fit()
    pred0 = model.predict(start=test0.index[0], end=test0.index[t_pred])
    elapsed_time_m = time.process_time() - tm
    rmsmo = round(sqrt(mean_squared_error(test0, pred0)), 4)
    # if t_start + t_pred + 1 > 560:
    plt.plot(train0.index, train0, label='Train/ Monnit')

```

```

plt.plot(test0.index, test0, label='Test/ step =' + str(t_start + t_pred + 1))
plt.plot(pred0.index, pred0, label='Holt-Winters/ RMS =' + str(rmsmo))
plt.legend(loc='best')
# sometimes it is required to stop the plots to get the last two plots
plt.show()
train1, test1 = hw.iloc[:t_start, 1], hw.iloc[t_start:t_start + t_pred + 1, 1]
th = time.process_time()
model = HoltWinters(train1, seasonal_periods=144, trend='add', seasonal='add').fit()
pred1 = model.predict(start=test1.index[0], end=test1.index[t_pred])
elapsed_time_th = time.process_time() - th
rmsth = round(sqrt(mean_squared_error(test1, pred1)), 4)
plt.plot(train1.index, train1, label='Train/ Thingful')
plt.plot(test1.index, test1, label='Test/ step =' + str(t_start + t_pred + 1))
plt.plot(pred1.index, pred1, label='Holt-Winters/ RMS =' + str(rmsth))
plt.legend(loc='best')
# sometimes it is required to stop the plots to get the last two plots
# if t_start + t_pred + 1 > 515:
plt.show()
# https://stackoverflow.com/questions/10715965/add-one-row-to-pandas-dataframe
rms_list.append((t_start + t_pred + 1, rmsmo, elapsed_time_m, rmsth, elapsed_time_th))

pdf = pd.DataFrame(rms_list, columns=['Steps', 'Monnit_rms', 'Monnit_t', 'Thingful_rms', 'Thingful_t
'])
pdf.set_index('Steps', inplace=True)
print(pdf)
rmsdf = pdf[['Monnit_rms', 'Thingful_rms']]
plt.figure()
rmsdf.plot()
plt.show()
tdf = pdf[['Monnit_t', 'Thingful_t']]
plt.figure()
tdf.plot()
plt.show()

```

A.4.2 ARMA and ARIMA predictive models

Listing A.6: The configuration and programming details of ARMA, ARIMA predictive models using ARMA and ARIMA Python packages provided by Statsmodels.org (Seabold & Perktold 2010).

```
import math
import os
import time
from math import sqrt
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import statsmodels.api as sm
from pmdarima.arima.utils import ndiffs
from scipy import stats
from sklearn.metrics import mean_squared_error
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.arima_model import ARMA

os.chdir('DataBase')
df = pd.read_csv('HW-Dataset-Monnit.csv', skiprows=0)[['Date', 'Value']]
df['Date'] = pd.to_datetime(df['Date'], format='%d/%m/%Y %I:%M %p') # monnit
# df = pd.read_csv('HW-Dataset-Thingful.csv', skiprows=0)[['ServerDate', 'AmbientTemperature']]
# df['ServerDate'] = pd.to_datetime(df['ServerDate'], format='%Y-%m-%d %H:%M:%S') # thingful
df.columns = ['R_Date', 'Temp']
df.set_index('R_Date', inplace=True)
df.sort_index(inplace=True, ascending=True)
df = df.resample('10T').mean().ffill()
# ts22 = df.resample('10T').mean().ffill()
ts = df['2020-03-03 00:00:00':'2020-03-09 00:00:00']
print('Shape of the DataFrame:', ts.shape)
no_of_observations = ts.shape[0]
train_ts, test_ts = np.split(ts, [int(.8 * len(ts))])
print('The size of the training dataset : ' + str(len(train_ts)))
print('The size of the testing dataset : ' + str(len(test_ts)))
##### #####      Function to plot signal, ACF and PACF
def plotds(xt, nlag=50, fig_size=(12, 10)):
    if not isinstance(xt, pd.Series):
        layout = (2, 2)
        ax_xt = plt.subplot2grid(layout, (0, 0), colspan=2)
        ax_acf = plt.subplot2grid(layout, (1, 0))
```

```

ax_pacf = plt.subplot2grid(layout, (1, 1))
xt.plot(ax=ax_xt)
ax_xt.set_title('Time-Series Autocorrelation(ACF) and Partial Autocorrelation Diagrams(PACF)
')
plot_acf(xt, lags=nlag, ax=ax_acf)
plot_pacf(xt, lags=nlag, ax=ax_pacf, method='ywm')
# https://www.statsmodels.org/0.8.0/generated/statsmodels.graphics.tsaplots.plot_pacf.html
plt.figure(figsize=fig_size)
plt.tight_layout()
plt.show()

# Function to plot Time series decomposition
def plottsd(xt, rfreg):
    res = sm.tsa.seasonal_decompose(xt, period=rfreg, model='additive') # Multiplicative additive
    fig, axs = plt.subplots(4, 1, figsize=(24, 12), sharey=False)
    axs[0].plot(xt)
    axs[0].set_title('Time Series Decomposition')
    res.trend.plot(ax=axs[1])
    axs[1].set_title('Trend')
    res.seasonal.plot(ax=axs[2])
    axs[2].set_title('Sequence')
    res.resid.plot(ax=axs[3])
    axs[3].set_title('Residuals')
    plt.show()

# Function of ADF test
def adft(xt, ttxt):
    print('====The ADF (Augmented Dickey-Fuller) Test =====')
    print(ttxt)
    print('<<<The null hypothesis: the time series is non-stationary>>>')
    ndiffs(xt, test='adf')
    if ndiffs(xt, test='adf') > 0:
        print('Failed to reject the null hypothesis, the data is Non-Stationary')
    else:
        print('Reject the Null hypothesis, the data is Stationary')
    print('The suggested order of Differencing is: ' + str(ndiffs(xt, test='adf')))
    print('=====')

# define KPSS
def kpss_test(timeseries, ttxt):
    print(
        '====The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test =====')
    print(ttxt)
    print('<<<The null hypothesis: the time series is stationary>>>')

    if ndiffs(timeseries, test='kpss') > 0:

```

```

        print('Reject the Null hypothesis, the data is Non-Stationary')
    else:
        print('Failed to reject the null hypothesis, the data is Stationary')
        print('The suggested order of Differencing is: ' + str(ndiffs(timeseries, test='kpss')))
        print('=====')
# define KPSS
def pp_test(timeseries, txt):
    print(
        '=====The Phillips-Perron (PP) Test
        =====')
    print(txt)
    print('<<<The null hypothesis: the time series is stationary>>>')

    if ndiffs(timeseries, test='pp') > 0:
        print('Reject the Null hypothesis, the data is Non-Stationary')
    else:
        print('Failed to reject the null hypothesis, the data is Stationary')
        print('The suggested order of Differencing is: ' + str(ndiffs(timeseries, test='kpss')))
        print('=====')
##### plot
plots(train_ts, nlag=50)
plottsd(train_ts, 144)
##### Evaluate mean and variance at mid values to test stationary status
R = math.trunc(no_of_observations / 2)
mean1, mean2 = train_ts.iloc[:R].mean(), train_ts.iloc[R:].mean()
var1, var2 = train_ts.iloc[:R].var(), train_ts.iloc[R:].var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
##### Augmented Dickey-Fuller(ADF) Test
adft(train_ts['Temp'], 'Checking the time series')
kpss_test(train_ts['Temp'], 'Checking the time series')
pp_test(train_ts['Temp'], 'Checking the time series')
# If Contradictory results of ADF and KPSS unit root tests
# https://stats.stackexchange.com/questions/30569/what-is-the-difference-between-a-stationary-test-
and-a-unit-root
test/235916#235916
# How to get the order virtually https://www.youtube.com/watch?v=ZE\_WGBe0\_VU
# ##### # QQ plot and probability plot #
sm.qqplot(train_ts['Temp'], line='s')
plt.show()
plt.boxplot(train_ts.Temp)
plt.show()
plt.hist(train_ts.Temp)
plt.show()
# ---Differencing 1

```



```

F_order_diff = train_ts.diff(1).dropna()
fig, ax = plt.subplots(2, sharex=True)
fig.set_size_inches(12, 10)
train_ts['Temp'].plot(ax=ax[0], color='b')
ax[0].set_title('Observations before differencing')
F_order_diff.plot(ax=ax[1], color='r')
ax[1].set_title('First-order differences ')
plt.show()
# plot the new thing
# plot signal
plotds(F_order_diff, nlag=50)
adft(F_order_diff.Temp, 'Checking the First-order differences')
kpss_test(F_order_diff.Temp, 'Checking the First-order differences')
pp_test(F_order_diff.Temp, 'Checking the First-order differences')
plt.hist(F_order_diff.Temp)
plt.show()
##Optimize ARMA parameters
tm = time.process_time()
aicVal = []
for ar in range(0, 3):
    for ma in range(0, 3):
        try:
            arma_obj = ARMA(F_order_diff['Temp'], order=(ar, ma))
            arma_obj_fit = arma_obj.fit(dispatch=False)
            aicVal.append([ar, ma, arma_obj_fit.aic])
            labels = ['p', 'q', 'aic']
            rdf = pd.DataFrame.from_records(aicVal, columns=labels)
        except ValueError:
            pass
ARMA_t_elapsed_time_m = round(time.process_time() - tm, 4)

# if AIC is negative
# https://stats.stackexchange.com/questions/486/negative-values-for-aicc-corrected-akaike-information-criterion/720
# get the row of minimum value 'The minimizing the Akaike information criterion (AIC) : ' +
aic = rdf.loc[rdf['aic'].idxmin()] #
print('Parameters with the lowest Akaike information criterion (AIC) : ')
print(aic)
p_AR = int(aic[0])
q_MA = int(aic[1])
##### https://www.statsmodels.org/stable/examples/notebooks/generated/tsa\_arma\_0.html
tm = time.process_time()
arma_mod = sm.tsa.ARMA(train_ts, (p_AR, q_MA)).fit(dispatch=False)
ARMA_p_elapsed_time_m = round(time.process_time() - tm, 4)
# print(arma_mod.summary())

```

```

print(arma_mod.aic, arma_mod.bic, arma_mod.hqic)
# Normality test
print(sm.stats.durbin_watson(arma_mod.resid.values))

fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111)
ax = arma_mod.resid.plot(ax=ax);
plt.show()
resid = arma_mod.resid
print('Normality test : ' + str(stats.normaltest(resid)))
# QQ plot and probability plot
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111)
fig = sm.qqplot(resid, line='q', ax=ax, fit=True)
plt.show()

fig = plt.figure(figsize=(12, 8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(resid.values.squeeze(), lags=40, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(resid, lags=40, ax=ax2)
plt.show()
predict_sunspots = arma_mod.predict('2020-03-07 00:00:00', '2020-03-09 00:00:00', dynamic=True)
# print(predict_sunspots)
fig, ax = plt.subplots(figsize=(12, 8))
ax = ts.loc['2020-03-03 00:00:00':'2020-03-09 00:00:00'].plot(ax=ax)
fig = arma_mod.plot_predict('2020-03-07 00:00:00', '2020-03-09 00:00:00', dynamic=True, ax=ax, plot_
    insample=False)
plt.show()

size = int(no_of_observations * 0.66)
# print(size)
train, test = ts.iloc[:size, 0], ts.iloc[size:, 0]
# train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
T_all = list()
for t in range(len(test)):
    # print('x')
    model = ARMA(history, order=(p_AR, q_MA))
    model_fit = model.fit(dispatch=False)
    output = model_fit.forecast()
    # print(output)
    yhat = output[0]
    predictions.append(yhat)

```

```

obs = test[t]
history.append(obs)
T_all.append([float(str(yhat).rstrip('[').rstrip(')'), obs])
labels = ['predicted', 'Observations']
rdf2 = pd.DataFrame.from_records(T_all, columns=labels)

error = mean_squared_error(test, predictions)
ARMA_rmsth = round(sqrt(mean_squared_error(test, predictions)), 4)
# plot
rdf2[['predicted', 'Observations']].plot(figsize=(12, 8))
plt.title('Temperature - Forecasting Model \n (ARMA - Prediction vs Observations )',
          fontsize=12, loc='center')
plt.xlabel('Date - Time / each step = 10 min')
plt.ylabel('Temperature')
plt.legend(loc='best')
plt.show()
#####ARIMA#####
tm = time.process_time()
aicVal = []
for d in range(1, 5):
    for ari in range(0, 3):
        for maj in range(0, 3):
            try:
                arima_obj = ARIMA(train_ts['Temp'], order=(ari, d, maj))
                arima_obj_fit = arima_obj.fit(dispatch=False)
                aicVal.append([ari, d, maj, arima_obj_fit.aic])
                labels = ['p', 'd', 'q', 'aic']
                rdf = pd.DataFrame.from_records(aicVal, columns=labels)
            except ValueError:
                pass
ARIMA_t_elapsed_time_m = round(time.process_time() - tm, 4)
# get the row of minimum value 'The minimizing the Akaike information criterion (AIC) : ' +
aic = rdf.loc[rdf['aic'].idxmin()] #
print('Parameters with the lowest Akaike information criterion (AIC) : ')
print(aic)
p_AR = int(aic[0])
d_DF = int(aic[1])
q_MA = int(aic[2])
# ARIMA model
tm = time.process_time()
arima_mod = sm.tsa.ARMA(train_ts, (p_AR, d_DF, q_MA)).fit(dispatch=False)
ARIMA_p_elapsed_time_m = round(time.process_time() - tm, 4)
predict_sunspots = arima_mod.predict('2020-03-07 00:00:00', '2020-03-09 00:00:00', dynamic=True)
# print(predict_sunspots)
fig, ax = plt.subplots(figsize=(12, 8))

```

```

ax = ts.loc['2020-03-03 00:00:00':'2020-03-09 00:00:00'].plot(ax=ax)
fig = arima_mod.plot_predict('2020-03-07 00:00:00', '2020-03-09 00:00:00', dynamic=True, ax=ax, plot
    _insample=False)
plt.show()
size = int(no_of_observations * 0.66)
train, test = ts.iloc[:size, 0], ts.iloc[size:, 0]
history = [x for x in train]
predictions = list()
T_all = list()

for t in range(len(test)):
    # print('x')
    model = ARIMA(history, order=(p_AR, d_DF, q_MA))
    model_fit = model.fit(dispatch=False)
    output = model_fit.forecast()
    # print(output)
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    T_all.append([float(str(yhat).rstrip('.')rstrip(' ')), obs])
    labels = ['predicted', 'Observations']
    rdf2 = pd.DataFrame.from_records(T_all, columns=labels)

error = mean_squared_error(test, predictions)
ARIMA_rmsth = round(sqrt(mean_squared_error(test, predictions)), 4)
# plot
rdf2[['predicted', 'Observations']].plot(figsize=(12, 8))
plt.title('Temperature - Forecasting Model \n (ARIMA - Prediction vs Observations)',
    fontsize=12, loc='center')
plt.xlabel('Date - Time / each step = 10 min')
plt.ylabel('Temperature')
plt.legend(loc='best')
plt.show()
print('Time required to optimize ARMA model : ' + str(ARMA_t_elapsed_time_m))
print('Time required to fit ARMA model : ' + str(ARMA_p_elapsed_time_m))
print('ARMA Root Mean Square Error is : ' + str(format(ARMA_rmsth)))
print('Time required to optimize ARIMA model : ' + str(ARIMA_t_elapsed_time_m))
print('Time required to fit ARIMA model : ' + str(ARIMA_p_elapsed_time_m))
print('ARIMA Root Mean Square Error is : ' + str(format(ARIMA_rmsth)))

```

A.4.3 SARIMA predictive model

Listing A.7: The configuration and programming details of SARIMA predictive models using SARIMAX Python packages provided by Statsmodels.org (Seabold & Perktold 2010).

```
# Import libraries
import itertools
import os
import time
import warnings
from math import sqrt
import matplotlib.pyplot as plt
import pandas as pd
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error

# Defaults
os.chdir('DataBase')
df = pd.read_csv('HW-Dataset-Monnit.csv', skiprows=0)[['Date', 'Value']]
df['Date'] = pd.to_datetime(df['Date'], format='%d/%m/%Y %I:%M %p') # monnit
#df = pd.read_csv('HW-Dataset-Thingful.csv', skiprows=0)[['ServerDate', 'AmbientTemperature']]
#df['ServerDate'] = pd.to_datetime(df['ServerDate'], format='%Y-%m-%d %H:%M:%S') # thingful
df.columns = ['R_Date', 'Temp']
df.set_index('R_Date', inplace=True)
df.sort_index(inplace=True, ascending=True)
df = df.resample('10T').mean().ffill()

ts = df['2020-03-03 00:00:00':'2020-03-07 00:00:00']
plt.rcParams['figure.figsize'] = (20.0, 10.0)
plt.rcParams.update({'font.size': 12})
#plt.style.use('ggplot')

# Define the d and q parameters to take any value between 0 and 1
q = d = range(0, 2)

# Define the p parameters to take any value between 0 and 3
p = range(0, 4)

# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 4) for x in list(itertools.product(p, d, q))]
print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMA: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMA: {} x {}'.format(pdq[1], seasonal_pdq[2]))
```

```

print('SARIMA: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMA: {} x {}'.format(pdq[2], seasonal_pdq[4]))

train_data = ts[:'2020-03-07 00:00:00']
test_data = ts['2020-03-06 00:00:00':'2020-03-07 00:00:00']
warnings.filterwarnings("ignore") # specify to ignore warning messages
AIC = []
SARIMAX_model = []
tm = time.process_time()
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(train_data,
                                             order=param,
                                             seasonal_order=param_seasonal,
                                             enforce_stationarity=False,
                                             enforce_invertibility=False)

            results = mod.fit(dispatch=False) # to stop getting a lot of details
            print('SARIMAX{}x{} - AIC:{}'.format(param, param_seasonal, results.aic), end='\r')
            AIC.append(results.aic)
            SARIMAX_model.append([param, param_seasonal])
        except:
            continue
SARIMA_t_elapsed_time_m = round(time.process_time() - tm, 4)
print('The smallest AIC is {} for model SARIMA{}x{}'.format(min(AIC), SARIMAX_model[AIC.index(min(
    AIC))][0],
                                                         SARIMAX_model[AIC.index(min(AIC))][1]))

# To fit the actual model using parameters set of with the lowest AIC
tm = time.process_time()
mod = sm.tsa.statespace.SARIMAX(train_data,
                                 order=SARIMAX_model[AIC.index(min(AIC))][0],
                                 seasonal_order=SARIMAX_model[AIC.index(min(AIC))][1],
                                 enforce_stationarity=False,
                                 enforce_invertibility=False)

results = mod.fit(dispatch=False)
SARIMA_p_elapsed_time_m = round(time.process_time() - tm, 4)
results.plot_diagnostics(figsize=(10, 7)) # model performance
plt.show()

pred0 = results.get_prediction(start='2020-03-06 00:00:00', dynamic=False)
pred0_ci = pred0.conf_int()
pred1 = results.get_prediction(start='2020-03-06 00:00:00', dynamic=True)
pred1_ci = pred1.conf_int()
ax = ts.plot(figsize=(12, 10))
pred0.predicted_mean.plot(ax=ax, label='1-step-ahead Forecast (get_predictions, dynamic=False)')

```

```

pred1.predicted_mean.plot(ax=ax, label='Dynamic Forecast (get_predictions, dynamic=True)')
plt.ylabel('Temperature')
plt.xlabel('Date')
plt.legend()
plt.show()

SARIMA_rmsth = round(sqrt(mean_squared_error(test_data, pred0.predicted_mean)), 4)
print('Time required to optimize SARIMA model :' + str(SARIMA_t_elapsed_time_m))
print('Time required to fit SARIMA model :' + str(SARIMA_p_elapsed_time_m))
print('SARIMA Root Mean Square Error is :' + str(format(SARIMA_rmsth)))
# ARMA, ARIMA, SARIMA performance and RMS
ARMA_R = pd.read_csv('ARMA_R.csv')
print(ARMA_R)
A_Fit = ARMA_R[['Th_ARMA_Fit', 'Mo_ARMA_Fit']]
AI_Fit = ARMA_R[['Th_ARIMA_Fit', 'Mo_ARIMA_Fit']]
SAI_Fit = ARMA_R[['Th_SARIMA_Fit', 'Mo_SARIMA_Fit']]
A_Opt = ARMA_R[['Th_ARMA_Opt', 'Mo_ARMA_Opt']]
AI_Opt = ARMA_R[['Th_ARIMA_Opt', 'Mo_ARIMA_Opt']]
SAI_Opt = ARMA_R[['Th_SARIMA_Opt', 'Mo_SARIMA_Opt']]
fig = plt.figure(figsize=(12, 8))
A_Fit.plot()
AI_Fit.plot()
SAI_Fit.plot()
A_Opt.plot()
AI_Opt.plot()
SAI_Opt.plot()
plt.show()

```

A.4.4 GPR predictive model

Listing A.8: The configuration and programming details of using “GaussianProcessRegressor” Python package in the GPR prediction model, Scikit-learn (Pedregosa et al. 2011).

```

import os
import time
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import ConstantKernel as CK
from sklearn.gaussian_process.kernels import RBF

```

```

from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from pmdarima.arima.utils import ndiffs

# set current working directory
os.chdir('DataBase')
rms_list = []

def load_data(df):
    # https://www.geeksforgeeks.org/python-working-with-date-and-time-using-pandas/
    months = []
    obs_sums = []
    counts = []
    # Convert date into minutes
    D = df['R_Date'].dt.day
    h = df['R_Date'].dt.hour
    m = df['R_Date'].dt.minute
    h_float = D * 24 * 60 + h * 60 + m
    obs_v = df.Temp
    for date_t, obs_t in zip(h_float, obs_v):
        if not months or date_t != months[-1]:
            months.append(date_t)
            obs_sums.append(obs_t)
            counts.append(1)
        else:
            # aggregate and produce average
            obs_sums[-1] += obs_t
            counts[-1] += 1

    months = np.asarray(months).reshape(-1, 1)
    avg_obs = np.asarray(obs_sums) / counts
    return months, avg_obs

def toList(ts, n_steps):
    # To convert a time series to a 2 dimensional list of n_steps in each row and one depended
    column
    x = []
    y = []
    for i in range(n_steps, ts.shape[0]):
        x.append(list(ts.iloc[i - n_steps:i - 1]))
        y.append(ts.iloc[i])
    x, y = np.array(x), np.array(y)
    return x, y

df_a = pd.read_csv('HW-Dataset-Monnit.csv', skiprows=0)[['Date', 'Value']]

```



```

df_a['Date'] = pd.to_datetime(df_a['Date'], format='%d/%m/%Y %I:%M %p') # monnit
#df_a = pd.read_csv('HW-Dataset-Thingful.csv', skiprows=0)[['ServerDate', 'AmbientTemperature']]
#df_a['ServerDate'] = pd.to_datetime(df_a['ServerDate'], format='%Y-%m-%d %H:%M:%S') # thingful
df_a.columns = ['R_Date', 'Temp']
ts_s = df_a.loc[2:20]
ts_p = df_a.loc[0:22]
ts_t = df_a.loc[2:434]
#ts_t = df_a.loc[2:434]
# load data
X, y = load_data(ts_s)
X1, y1 = load_data(ts_p)
# Read the dataset into a pandas.DataFrame
print('Shape of the data frame:', df_a.shape)
mix_k = kernel = CK(1.0, (1e-4, 1e4)) * RBF(10, (1e-4, 1e4))

gp = GaussianProcessRegressor(kernel=mix_k, alpha=0, normalize_y=True)

gp.fit(X, y)

print("\nLearned kernel: %s" % gp.kernel_)
print("Log-marginal-likelihood: %.3f"
      % gp.log_marginal_likelihood(gp.kernel_.theta))

X_ = np.linspace(X.min(), X.max() + 30, 1000)[:, np.newaxis]
y_pred, y_std = gp.predict(X_, return_std=True)

plt.plot(X_, y_pred, c='r')
plt.fill_between(X_[0], y_pred - y_std, y_pred + y_std, alpha=0.5, color='g')
plt.xlim(X_.min(), X_.max())
plt.scatter(X1, y1, c='b')
plt.scatter(X, y, c='r')
plt.xlabel("Time (Minute)")
plt.ylabel(r"Temperature C")
plt.title(r"Gaussian Process Regression")
plt.tight_layout()
plt.show()

indep_data, dep_data = toList(ts_t['Temp'], 2)
print('Shape of the dataset:', indep_data.shape, dep_data.shape)
# Divide the dataset into training and testing using a fancy way
train_set = np.random.choice([True, False], len(dep_data), p=[.8, .2])
# gpr
gpr = GaussianProcessRegressor()
t_loop = 25
for i in range(t_loop):

```

```

tm = time.process_time()
r2 =0
gpr = GaussianProcessRegressor(alpha=5,
                                n_restarts_optimizer=20,
                                kernel=mix_k)

gpr.fit(indep_data[train_set], dep_data[train_set]) #
# ~~ to indicate the false (20%) ratio
test_preds = gpr.predict(indep_data[~train_set]) # indep_data[~train_set]
elapsed_time_m = time.process_time() - tm
#r2 = r2_score(test_preds, dep_data[~train_set]) ~~Coefficient of determination
r2 = mean_squared_error(test_preds, dep_data[~train_set], squared=False)
rsid= test_preds - dep_data[~train_set]
rms_list.append((elapsed_time_m, r2, ndiffs(rsid, test='adf')))
print(elapsed_time_m,r2)
pd.DataFrame(rms_list,columns=['Time_Mo', 'RMS_Mo', 'Stationary_Mo']).to_csv("GPR.csv")
print(rms_list)
f, ax = plt.subplots(figsize=(10, 7), nrows=3)
f.tight_layout()
ax[0].plot(range(len(test_preds)), test_preds, label='Predicted Values')
ax[0].plot(range(len(test_preds)), dep_data[~train_set], label='Actual Values')
ax[0].set_title("Predicted vs Actuals")
ax[0].legend(loc='best')
ax[1].plot(range(len(test_preds)), test_preds - dep_data[~train_set])
ax[1].set_title("Plotted Residuals")
ax[2].hist(test_preds - dep_data[~train_set])
ax[2].set_title("Histogram of Residuals")
plt.show()
#r2 = r2_score(test_preds, dep_data[~train_set]) Coefficient of determination
r2 = mean_squared_error(test_preds, dep_data[~train_set], squared=False)
print('R-squared on validation set of the original Temperature:', r2)
GPR_R = pd.read_csv('GPR-th.csv',header = 0,encoding = 'unicode_escape')
A_Fit = GPR_R[['Time_Mo', 'Time_Th']]
fig = plt.figure(figsize=(12, 8))
A_Fit.plot()
plt.show()
r_Fit = GPR_R[['RMS_Mo', 'RMS_Th']]
fig = plt.figure(figsize=(12, 8))
r_Fit.plot()
plt.show()

```

A.4.5 LSTM predictive model

Listing A.9: The configuration and programming details of using Keras Python package to construct the LSTM prediction model (Pedregosa et al. 2011).

```
import os
import time
import numpy as np
import pandas as pd
import seaborn as sns

from keras.callbacks import ModelCheckpoint
from keras.layers import Dense, Input, Dropout
from keras.layers.recurrent import LSTM
from keras.models import Model
from keras.models import load_model
from matplotlib import pyplot as plt
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler

def toList(ts, n_steps):
    # To convert a time series to a 2 dimensional list of n_steps in each row and one depended
    # column
    x = []
    y = []
    for i in range(n_steps, ts.shape[0]):
        x.append(list(ts.iloc[i - n_steps:i - 1]))
        y.append(ts.iloc[i])
    x, y = np.array(x), np.array(y)
    return x, y

# set current working directory
os.chdir('DataBase')
#df = pd.read_csv('HW-Dataset-Monnit.csv', skiprows=0)[['Date', 'Value']]
#df['Date'] = pd.to_datetime(df['Date'], format='%d/%m/%Y %I:%M %p') # monnit
df = pd.read_csv('HW-Dataset-Thingful.csv', skiprows=0)[['ServerDate', 'AmbientTemperature']]
df['ServerDate'] = pd.to_datetime(df['ServerDate'], format='%Y-%m-%d %H:%M:%S') # thingful

df.columns = ['R_Date', 'Temp']
df.set_index('R_Date', inplace=True)
df.sort_index(inplace=True, ascending=True)
df = df.resample('10T').mean().ffill()
print('Shape of the data frame:', df.shape)
```

```

# Box plot
plt.figure()
g = sns.boxplot(df['Temp'])
g.set_title('Box plot of Temperature - Sensor')
plt.figure()
# sns.lineplot(x='Year', y='Fatalities', data=df['PRES'], hue='Twin_Cities')
g = sns.lineplot(data=df['Temp'])
g.set_title('Time series of Temperature - Sensor')
g.set_xlabel('Index')
g.set_ylabel('Temperature readings in C')
plt.show()

# mixmax to scale the dataset
# variable within [0,1].
scaler = MinMaxScaler(feature_range=(0, 1))
df['scaled_Temp'] = scaler.fit_transform(np.array(df['Temp']).reshape(-1, 1))
print(df.head())

# Splitting the dataset into train and validation.
df_train = df[:'2020-03-12 00:00:00']
df_val = df['2020-03-12 00:01:00':]
print('Shape of train:', df_train.shape)
print('Shape of test:', df_val.shape)

# Reset and rebuild the index of the validation set
df_val.reset_index(drop=True, inplace=True)

# Plot the scaled training dataset
plt.figure()
g = sns.lineplot(data=df_train['scaled_Temp'], color='b')
g.set_title('Time series of scaled temperature in train set')
g.set_xlabel('Index')
g.set_ylabel('Scaled temperature readings')

plt.figure()
g = sns.lineplot(data=df_val['scaled_Temp'], color='r')
g.set_title('Time series of scaled temperature in validation set')
g.set_xlabel('Index')
g.set_ylabel('Scaled temperature readings')
plt.show()

X_train, y_train = toList(df_train['scaled_Temp'], 8)
print('Shape of train arrays:', X_train.shape, y_train.shape)
X_val, y_val = toList(df_val['scaled_Temp'], 8)
print('Shape of validation arrays:', X_val.shape, y_val.shape)

# The input to RNN layers must be of shape (number of samples, number of timesteps, number of
  features per timestep).

```

```

# The number of features per timestep is one (Temperature).
X_train, X_val = X_train.reshape((X_train.shape[0], X_train.shape[1], 1)), X_val.reshape(
    (X_val.shape[0], X_val.shape[1], 1))
print('Shape of 3D arrays:', X_train.shape, X_val.shape)
# Defining the model parameters using the Keras Functional API.
# Define input layer which has shape (None, 7) and of type float32. None indicates the number of
    instances
input_layer = Input(shape=(7, 1), dtype='float32')
lstm_layer = LSTM(64, input_shape=(7, 1), return_sequences=False)(input_layer)
dropout_layer = Dropout(0.2)(lstm_layer)
# The output layer gives prediction for the next day.
output_layer = Dense(1, activation='linear')(dropout_layer)
# The model object
ts_model = Model(inputs=input_layer, outputs=output_layer)
ts_model.compile(loss='mse', optimizer='adam')
ts_model.summary()
save_weights_at = os.path.join('keras_models', 'LSTM-Temp.hdf5')
#print(save_weights_at)
save_best = ModelCheckpoint(save_weights_at, monitor='val_loss', verbose=0,
                            save_best_only=True, save_weights_only=False, mode='min',
                            period=1)
t_loop = 25
print(df_val['Temp'].iloc[8:])
rms_list = []
for i in range(t_loop):
    tm = time.process_time()
    # ts_model.fit(x=X_train, y=y_train, batch_size=16, epochs=20, verbose=1, callbacks=[save_best],
        validation_data=(X_val, y_val), shuffle=True)
    best_model = load_model(os.path.join('keras_models', 'LSTM-Temp.hdf5'))
    preds = best_model.predict(X_val)
    pred_Temp = scaler.inverse_transform(preds)
    pred_Temp = np.squeeze(pred_Temp)
    elapsed_time_m = round(time.process_time() - tm, 4)
    # R-squared is also calculated for the predictions on the original variable.
    ##r2 = r2_score(df_val['Temp'].iloc[8:], pred_Temp)
    #r2 = mean_squared_error(df_val['Temp'].iloc[8:], pred_Temp, squared=False)
    #print(r2)
    #rms_list.append((elapsed_time_m, r2))

# To plot the first 50 actual and predicted values of temperature.
#pd.DataFrame(rms_list, columns=['Time_Mo', 'RMS_Mo']).to_csv("LSTM_r.csv")
plt.figure(figsize=(5.5, 5.5))
plt.plot(range(48), df_val['Temp'].iloc[8:56], linestyle='-', marker='*', color='r')
#pd.DataFrame(range(48), df_val['Temp'].iloc[8:56]).to_csv("LSTM_Out.csv")
plt.plot(range(48), pred_Temp[:48], linestyle='-', marker='.', color='b')

```

```

#pd.DataFrame(range(48), pred_Temp[:48]).to_csv("LSTM_predicted.csv")
plt.legend(['Actual', 'Predicted'], loc=2)
plt.title('Actual vs Predicted Temperature')
plt.ylabel('Temperature')
plt.xlabel('Index')
plt.show()

# Define anomalies
LSTM_list = []
LSTM_list = list(zip(range(48), df_val['Temp'].iloc[8:56], pred_Temp[:48]))
LSTM_DF= pd.DataFrame(LSTM_list, columns=['Observation_Seq','Actual','Predicted'])
rmse=mean_squared_error(df_val['Temp'].iloc[8:56], pred_Temp[:48], squared=False)
print('RMSE on validation set of the original Temperature:', rmse)
r2 = r2_score(df_val['Temp'].iloc[8:56], pred_Temp[:48])
print('R-squared on validation set of the original Temperature:', r2)
LSTM_DF.to_csv("LSTM_r.csv")
LSTM_DF['Prediction_Error'] = abs(LSTM_DF['Predicted']-LSTM_DF['Actual'])
LSTM_DF['Status'] = np.where(LSTM_DF['Prediction_Error']> 0.34, 'Anomaly', 'Pass')
print(LSTM_DF)
# plt.savefig('B07887_05_11.png', format='png', dpi=300)
ARMA_R = pd.read_csv('LSTM_th.csv', header=0, encoding='unicode_escape')
#print(ARMA_R)
A_Fit = ARMA_R[['Th_rms', 'Mo_rms']]
AI_Fit = ARMA_R[['Th_time', 'Mo_time']]
fig = plt.figure(figsize=(12, 8))
A_Fit.plot()
AI_Fit.plot()
plt.show()

```

A.4.6 K-means and DBSCAN Partitioning Models

Listing A.10: The configuration and programming details of using the “cluster.KMeans” and the “cluster.DBSCAN” Python packages to construct the spatial partitioning models (Pedregosa et al. 2011).

```

import os
import time
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import scipy.spatial as spatial
from PIL import Image

```

```

from geopy.distance import great_circle
from shapely.geometry import MultiPoint
from sklearn import metrics
from sklearn.cluster import DBSCAN
from sklearn.cluster import KMeans

def get_map(x, y, z, size, filename):
    import urllib.request
    # static_map = "https://harrywood.dev.openstreetmap.org/staticmaplite/staticmap.php?center
    ={0},{1}&zoom={2}&size={3}x{3}&mptype=mapnik".format(
    # y, x, z, size)
    static_map = "https://www.mapquestapi.com/staticmap/v4/getmap?key=KK14fV0hLMfNuNiHPp9m86
    uAAKqKpJev&" \
                "size=1000,800&ttype=map&imagetype=jpg&" \
                "zoom=10&scalebar=false&traffic=false&" \
                "bestfit=51.3662,-0.4058,51.6892,0.2431" \
                "&xis=https://s.aolcdn.com/os/mapquest/yogi/icons/poi-active.png,1,c,0,0&ellipse=
    fill:0x70ff0000|color:0xff0000|width:2|40.00,-105.25,40.04,-105.30"
    static_map_filename, headers = urllib.request.urlretrieve(static_map, filename)
    return static_map_filename

def geomap(data, zoom=10, point_size=18, point_color='r', point_alpha=1):
    # corrections to match geo with static map
    z = zoom
    picsize = 1200
    wx = 1.0 * 360 * (picsize / 256) / (2 ** z)
    wy = 0.76 * 360 * (picsize / 256) / (2 ** z)
    # center of London
    y = 51.5277
    x = -0.0764
    x_min, x_max = x - wx / 2, x + wx / 2
    y_min, y_max = y - wy / 2, y + wy / 2
    static_map_filename = os.path.join('', 'nyc_staticmap_{}_{}.png'.format(z, picsize))
    if os.path.isfile(static_map_filename) == False:
        get_map(x, y, z, picsize, static_map_filename)
    img = Image.open(static_map_filename)
    # add the static map
    plt.imshow(img, zorder=0, extent=[-0.4008, 0.2481, 51.3662, 51.6892], interpolation='none',
    aspect='auto')
    # add the scatter plot of events
    plt.plot(
        data['Longitude'],
        data['Latitude'],
        '.',
        markerfacecolor=point_color,

```

```

        markeredgecolor='b',
        markersize=point_size,
        alpha=point_alpha)
# limit the plot to the given box
plt.xlim(-0.4008, 0.2481)
plt.ylim(51.3662, 51.6892)

def voronoi_polygons_2d(vor, radius=None):
    if vor.points.shape[1] != 2:
        raise ValueError("Requires 2D input")

    new_regions = []
    new_vertices = vor.vertices.tolist()
    center = vor.points.mean(axis=0)
    if radius is None:
        radius = vor.points.ptp().max() * 2

    # Construct a map containing all ridges for a given point
    all_ridges = {}
    for (p1, p2), (v1, v2) in zip(vor.ridge_points, vor.ridge_vertices):
        all_ridges.setdefault(p1, []).append((p2, v1, v2))
        all_ridges.setdefault(p2, []).append((p1, v1, v2))

    # Reconstruct infinite regions
    for p1, region in enumerate(vor.point_region):
        vertices = vor.regions[region]

        if all([v >= 0 for v in vertices]):
            # finite region
            new_regions.append(vertices)
            continue

        # reconstruct a non-finite region
        ridges = all_ridges[p1]
        new_region = [v for v in vertices if v >= 0]

        for p2, v1, v2 in ridges:
            if v2 < 0:
                v1, v2 = v2, v1
            if v1 >= 0:
                # finite ridge: already in the region
                continue

            # Compute the missing endpoint of an infinite ridge
            t = vor.points[p2] - vor.points[p1] # tangent
            t /= np.linalg.norm(t)
            n = np.array([-t[1], t[0]]) # normal
            midpoint = vor.points[[p1, p2]].mean(axis=0)

```



```

        direction = np.sign(np.dot(midpoint - center, n)) * n
        far_point = vor.vertices[v2] + direction * radius
        new_region.append(len(new_vertices))
        new_vertices.append(far_point.tolist())

    # sort region counterclockwise
    vs = np.asarray([new_vertices[v] for v in new_region])
    c = vs.mean(axis=0)
    angles = np.arctan2(vs[:, 1] - c[1], vs[:, 0] - c[0])
    new_region = np.array(new_region)[np.argsort(angles)]

    # finish
    new_regions.append(new_region.tolist())
return new_regions, np.asarray(new_vertices)

def get_centermost_point(cluster):
    centroid = (MultiPoint(cluster).centroid.x, MultiPoint(cluster).centroid.y)
    centermost_point = min(cluster, key=lambda point: great_circle(point, centroid).m)
    return tuple(centermost_point)

# Load data
os.chdir('DataBase')
df = pd.read_csv('th_temp_sensors.csv', encoding='utf-8')
data = df[['Longitude', 'Latitude']]

# Plotting sensor-nodes over the Map
fig = plt.figure()
fig.set_size_inches(20, 20)
# geomap is responsible for downloading the map
geomap(df)
plt.title('Sensor-nodes distribution')
plt.show()

# Determining the best K for K-means
scores = []
k_list = []
DB_list = []
range_K = np.arange(2, 20) # change to 359 ~~ Change back to 359
silhouette_score = -1
K_best = 0
K_model = None
K_clusters = None
for i in range_K:
    # Train the model
    tm = time.process_time()
    kmeans = KMeans(init='k-means++', n_clusters=i, n_init=10)
    kmeans.fit(df[['Longitude', 'Latitude']])
    score = metrics.silhouette_score(df[['Longitude', 'Latitude']], kmeans.labels_,
                                     metric='euclidean', sample_size=len(df[['Longitude', 'Latitude
    ]]))

```

```

K_means_elapsed_time_m = round(time.process_time() - tm, 4)
k_list.append((K_means_elapsed_time_m))
print("Number of clusters =", i)
print("Silhouette score =", score)
scores.append(score)

if score > silhouette_score:
    silhouette_score = score
    K_best = i
    K_model = kmeans
    K_clusters = K_model.cluster_centers_

# Plot scores
plt.figure()
plt.bar(range_K, scores, width=0.6, color='b', align='center')
print("The best Silhouette score is : ", silhouette_score, " k = ", K_best)
plt.title('K-means Silhouette score vs No. of clusters, k=' + str(K_best))
plt.show()
print('K-mean++ is Done ! Ok *_^')

# compute Voronoi tessellation
vor = spatial.Voronoi(K_clusters)

# compute regions
regions, vertices = voronoi_polygons_2d(vor)

# prepare figure
fig = plt.figure()
fig.set_size_inches(20, 20)

# geomap
geomap(df[['Longitude', 'Latitude']], 13, 2, 'k', 0.1)

# centroids
plt.plot(K_clusters[:, 0], K_clusters[:, 1], 'bo', markersize=10)

# colorize and plot the boundaries
for region in regions:
    polygon = vertices[region]
    plt.fill(*zip(*polygon), alpha=0.4)
plt.show()

# DBSCAN

# Define epsilon as 6371.0088 kilometers, converted to radians for use by Haversine
kms_per_radian = 6371.0088

# The maximum distance between two samples for them to be considered as in the same neighborhood.
epsilon = 1.8 / kms_per_radian # Initial value

# Find the best epsilon
eps_grid = np.linspace(0.2, 4, num=40)
s_scores = []
e_best = eps_grid[0]
silhouette_score_max = -1
DB_model_best = None

```

```

DB_labels_best = None

for eps in eps_grid:
    # Train DBSCAN clustering model
    tm = time.process_time()
    DBS_model = DBSCAN(eps=eps / kms_per_radian, min_samples=2, algorithm='ball_tree', metric='
    haversine').fit(
        np.radians(df[['Longitude', 'Latitude']]))
    # Extract labels
    DBS_labels = DBS_model.labels_
    # Extract performance metric
    s_score = round(metrics.silhouette_score(data, DBS_labels), 6)
    DB_elapsed_time_m = round(time.process_time() - tm, 4)
    DB_list.append((DB_elapsed_time_m))
    s_scores.append(s_score)
    print("Epsilon:", eps, " --> silhouette score:", s_score)
    if s_score > silhouette_score_max:
        silhouette_score_max = s_score
        eps_best = eps
        DB_model_best = DBS_model
        DB_labels_best = DBS_labels
#print(DB_labels_best)
df['cluster_label'] = DB_labels_best
df.to_csv('th_temp_sensors_labels.csv')
db_labels = sorted(set(DB_labels_best))
fig = plt.figure()
fig.set_size_inches(20, 20)
empty = pd.DataFrame(columns=['Longitude', 'Latitude'])
geomap(empty, 13, 2, 'k', 0.1)

# convex hulls for every cluster
for k in db_labels:
    xy = data[DB_model_best.labels_ == k]
    plt.plot(xy['Longitude'], xy['Latitude'], 'kD' if k < 0 else 'o', markersize=10)
    if k >= 0:
        xy = data[DB_model_best.labels_ == k][['Longitude', 'Latitude']].reset_index(drop=True)
        try:
            hull = spatial.ConvexHull(xy.values)
            for simplex in hull.simplices:
                plt.plot(xy.iloc[simplex]['Longitude'], xy.iloc[simplex]['Latitude'], 'b-', lw=5)
        except:
            pass
plt.show()
# Plot silhouette scores vs epsilon
plt.figure()

```

```

plt.bar(eps_grid, s_scores, width=0.06, color='b', align='center')
plt.title('Silhouette score vs epsilon, Best epsilon = ' + str(eps_best))
plt.show()
# Best params
print("\nBest epsilon =", eps_best)
# Associated model and labels for best epsilon
# Check for unassigned datapoints in the labels
offset = 0
if -1 in DB_labels_best:
    offset = 1
# Number of clusters in the data
num_clusters = len(set(DB_labels_best)) - offset
print("\nEstimated number of clusters =", num_clusters)
# Time plot plot
plt.plot(k_list)
plt.show()
plt.plot(DB_list)
plt.show()

```

A.4.7 DTW and K-Shape Models

Listing A.11: The programming aspects of using both DTW and K-Shape models based on the Python package *tslearn.clustering* provided by Scikit-learn (Pedregosa et al. 2011).

```

import os
import time
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pandas import read_csv
from tslearn.clustering import KShape, TimeSeriesScalerMeanVariance
from tslearn.clustering import TimeSeriesKMeans
from tslearn.utils import to_time_series_dataset

dtw_list = []
KS_list = []
TSC_range = np.arange(1, 21)
os.chdir('DataBase')
# this is the only line to be changed for a different dataset
#####
# df = read_csv('SensorHistory.csv', header=0, squeeze=True)[['Date', 'SensorID', 'Value']]

```

```

df = read_csv('th_temp_weather_sensors_2d.csv', header=0, squeeze=True)[['UpdatedAt', 'id', '
    Temperature']]
#####
# labeling the columns
df.columns = ['Date', 'SensorID', 'Value']
# Fixing the date column to the correct datetime format
#####
# df['Date'] = pd.to_datetime(df['Date'], format='%d/%m/%Y %I:%M %p') # for the indoor sensors
df['Date'] = pd.to_datetime(df['Date'],
                            format='%Y-%m-%d %H:%M:%S') # //stackoverflow.com/questions/1759455/how
                            -can-i-account-for-period-am-pm-with-datetime-strptime
#####
# sort the data properly
df.sort_values(by=['SensorID', 'Date'], inplace=True, ascending=False)
df['Date'] = pd.to_datetime(df['Date'], unit='s')
df = df.set_index(['Date'])
res = df.groupby('SensorID').resample('10min').mean().reset_index(
    'Date').ffill() # //stackoverflow.com/questions/51705583/pandas-resample-timeseries-data-to-15-
    mins-and-45-mins-using-multi-index-or-co
res['SensorID'] = res.index.astype(str)
df = res[['Date', 'SensorID', 'Value']]
df.reset_index(drop=True, inplace=True)
df = df.set_index(['Date'])
#print(df)
print('Data processing --> pivot')
# chech why df is their https://stackoverflow.com/questions/42134486/typeerror-pivot-table-got-
    multiple-values-for-keyword-argument-values
pivot_df = pd.pivot_table(df, index=["Date"], columns=["SensorID"], values=["Value"], aggfunc=np.
    mean).ffill(
    axis=1).bfill(axis=1)
#pivot_df.to_csv('PythonExport_3.csv', sep=',')
# Drop the null value
pivot_df = pivot_df.dropna(how='any',
                            axis=0) # //stackoverflow.com/questions/44548721/remove-row-with-null-
                            value-from-pandas-data-frame/44548976
stacked_df = pivot_df.stack()
#print(stacked_df)
print('Data processing --> To time series')
# source: https://stackoverflow.com/questions/55662705/how-to-transfer-a-data-frame-column-into-the-
    format-that-tslearn-needs
Sensor_data = stacked_df.groupby('SensorID').agg(list)['Value'] # .to_numpy()
X_train = to_time_series_dataset(Sensor_data)
# print(np.argwhere(np.isnan(X_train)))
# print(X_train.shape)
Y_train = TimeSeriesScalerMeanVariance(mu=0., std=1.).fit_transform(X_train)

```

```

# Y_train = TimeSeriesScalerMeanVariance().fit_transform(X_train)
#####
ks = TimeSeriesKMeans(n_clusters=2, metric="dtw", random_state=0)
print('Fitting DTW Model... it usually takes a while')
for i in TSC_range:
    tm = time.process_time()
    y_pred = ks.fit_predict(Y_train)
    DTW_elapsed_time_m = round(time.process_time() - tm, 4)
    dtw_list.append((DTW_elapsed_time_m))
pd.DataFrame(dtw_list).to_csv("TSC_DTW.csv")
#dfs = pd.DataFrame(y_pred)
#dfs.to_csv('DTW_X.csv', sep=',')
#####
# print(km.cluster_centers_)
# print(km.init)
sz = Y_train.shape[1]
# Euclidean k-means
#####
# ks = KShape(n_clusters=2, verbose=False, random_state=0)
# print('Fitting the Model... it usually takes a while')
# y_pred = ks.fit_predict(Y_train)
#####
# print(y_pred[1])
print('Plotting... it usually takes a while also!!')
plt.figure(figsize=(24, 20))
for yi in range(2):
    plt.subplot(2, 1, 1 + yi)
    for xx in Y_train[y_pred == yi]:
        plt.plot(xx.ravel(), "b-", alpha=.4)
    plt.plot(ks.cluster_centers_[yi].ravel(), "r-")
    plt.xlim(0, sz)
    plt.ylim(-4, 4)
    plt.title("Cluster %d" % (yi + 1))
plt.tight_layout()
print('It should show something very soon...')
plt.show()
#####
ks = KShape(n_clusters=2, verbose=False, random_state=0)
for i in TSC_range:
    print('Fitting KS Model... it usually takes a while')
    tm = time.process_time()
    y_pred = ks.fit_predict(Y_train)
    KS_elapsed_time_m = round(time.process_time() - tm, 4)
    KS_list.append((KS_elapsed_time_m))
pd.DataFrame(KS_list).to_csv("TSC_KS.csv")

```

```
#####
# print(y_pred[1])
print('Plotting... it usually takes a while also!!')
plt.figure(figsize=(24, 20))
for yi in range(2):
    plt.subplot(2, 1, 1 + yi)
    for xx in Y_train[y_pred == yi]:
        plt.plot(xx.ravel(), "b-", alpha=.4)
    plt.plot(ks.cluster_centers_[yi].ravel(), "r-")
    plt.xlim(0, sz)
    plt.ylim(-4, 4)
    plt.title("Cluster %d" % (yi + 1))
plt.tight_layout()
print('It should show something very soon...')
plt.show()
```

A.4.8 Characteristics (features)-based Model

Listing A.12: The programming aspects of using the Python *tsfresh* package provided by (Christ et al. 2018).

```
import os
import time
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pandas import read_csv
from sklearn.cluster import KMeans
from tsfresh.feature_extraction import extract_features
from tsfresh.feature_extraction.settings import EfficientFCParameters

FB_list = []
TSC_range = np.arange(1, 2)
os.chdir('C:\\Users\\DellPC\\PycharmProjects\\PhDProject\\DataBase')
df = read_csv('th_temp_weather_sensors_2d.csv', header=0, index_col='R_date', squeeze=True)
df = df.sort_index()
df.rename(columns={'Temperature': 'Value', 'id': 'SensorID'}, inplace=True)

settings_time = EfficientFCParameters() # ComprehensiveFCParameters()# EfficientFCParameters()#
    MinimalFCParameters() #ComprehensiveFCParameters()# # TimeBasedFCParameters()#
    MinimalFCParameters() #EfficientFCParameters() #
for i in TSC_range:
```

```

X_tsfresh = extract_features(df, column_id="SensorID", column_value='Value', # column_kind='
kind'

                                default_fc_parameters=settings_time
                                ) # .settings.from_columns('Value__absolute_sum_of_changes',
columns_to_ignore=None)
# X_tsfresh.drop(columns, inplace=True, axis=1)
X_tsfresh.to_csv('PythonExport_testt.csv', sep=',')
kmeans = KMeans(n_clusters=2)
X_training = X_tsfresh[
    ['Value__absolute_sum_of_changes'
     # , 'Value__maximum', 'Value__mean', 'Value__median', 'Value__minimum', 'Value__standard_
     deviation',
     # 'Value__variance'
    ]]
# df.Gene1 = df.Gene1 + 1
X_training['Value__absolute_sum_of_changes'] = X_training['Value__absolute_sum_of_changes'].
astype('float64')
X_training['Value__absolute_sum_of_changes'] = X_training['Value__absolute_sum_of_changes'] / (
    X_training['Value__absolute_sum_of_changes'] + 1)
# print (X_training)

print('Fitting F-B Model.... it usually takes a while')
tm = time.process_time()
y = kmeans.fit_predict(X_training)
FB_elapsed_time_m = round(time.process_time() - tm, 4)
FB_list.append((FB_elapsed_time_m))
pd.DataFrame(FB_list).to_csv("TSC_FB.csv")

X_tsfresh['Cluster'] = y
X_tsfresh.to_csv('PythonExport_X_tsfresh.csv', sep=',')

df_indoor = X_tsfresh[X_tsfresh['Cluster'] < 1]
df_outdoor = X_tsfresh[X_tsfresh['Cluster'] >= 1]

fig, ax = plt.subplots(figsize=(10, 7), nrows=2, ncols=2)
plt.suptitle("The main extracted features from the time-series of one indoor and three "
            "outdoor temperature sensors", y=-1)
#ideal dataset only
# fig.tight_layout()
# ax[0, 0].scatter(df_indoor.iloc[:, 2], df_indoor.iloc[:, 2], label='Cluster 1', marker="*", s=80)
# ax[0, 0].scatter(df_outdoor.iloc[:, 2], df_outdoor.iloc[:, 2], label='Cluster 2', marker="*", s
=80)
# ax[0, 0].set_title("Mean value")
# ax[0, 0].legend(loc='best')
# ax[0, 1].scatter(df_indoor.iloc[:, 3], df_indoor.iloc[:, 3], label='Cluster 1', marker=">", s=80)

```



```

# ax[0, 1].scatter(df_outdoor.iloc[:, 3], df_outdoor.iloc[:, 3], label='Cluster 2', marker="<", s
=80)
# ax[0, 1].set_title("Median")
# ax[0, 1].legend(loc='best')
# ax[1, 0].scatter(df_indoor.iloc[:, 5], df_indoor.iloc[:, 5], label='Cluster 1', marker=">", s=80)
# ax[1, 0].scatter(df_outdoor.iloc[:, 5], df_outdoor.iloc[:, 5], label='Cluster 2', marker=">", s
=80)
# ax[1, 0].set_title("Standard Deviation")
# ax[1, 0].legend(loc='best')
# ax[1, 1].scatter(df_indoor.iloc[:, 1], df_indoor.iloc[:, 7], label='Cluster 1', marker=".", s=80)
# ax[1, 1].scatter(df_outdoor.iloc[:, 1], df_outdoor.iloc[:, 7], label='Cluster 2', marker=".", s
=80)
# ax[1, 1].set_title("Variance")
# ax[1, 1].legend(loc='best')
# plt.show()

i = 0
fig, axs = plt.subplots(len(X_tsfresh) // 4 + 1, 4, figsize=(14, 98), subplot_kw=dict(projection='
polar'))
fig.tight_layout()
fig.suptitle('Sensors time series polar plots', y=1, fontsize=10)

for index, row in X_tsfresh.iterrows():
    if row['Cluster'] == 1:
        C = 'r'
    else:
        C = 'b'
    SData = df[df.SensorID == index]
    axs[i // 4, i % 4].plot(SData['R_id'], SData['Value'], c=C)
    i += 1
plt.show()

i = 0
fig, axs = plt.subplots(len(X_tsfresh) // 4 + 1, 4, figsize=(14, 98))
fig.tight_layout()
fig.suptitle('Sensors time series plots', y=1, fontsize=10)
for index, row in X_tsfresh.iterrows():
    if row['Cluster'] == 1:
        C = 'r'
    else:
        C = 'b'
    SData = df[df.SensorID == index]
    axs[i // 4, i % 4].plot(SData['Value'], c=C)
    i += 1

```

```

plt.show()

i = 0
fig, axs = plt.subplots(len(X_tsfresh) // 4 + 1, 4, figsize=(14, 98))
fig.tight_layout()
fig.suptitle('Sensors time series Boxplots', y=1, fontsize=10)
for index, row in X_tsfresh.iterrows():
    SData = df[df.SensorID == index]
    axs[i // 4, i % 4].boxplot(SData['Value'])
    i += 1
plt.show()

# To plot everything in one place
for index, row in X_tsfresh.iterrows():
    if row['Cluster'] == 1:
        C = 'r'
    else:
        C = 'b'
    PData = df[df.SensorID == index]
    PData.reset_index(inplace=True)
    plt.plot(PData.index, PData['Value'], c=C)

plt.show()

TSC_L = pd.read_csv('TSC_list_2.csv')
A_Fit = TSC_L[['DTW', 'Feature-based']]
B_Fit = TSC_L[['K-Shape', 'Feature-based']]
fig = plt.figure(figsize=(12, 8))
A_Fit.plot()
B_Fit.plot()
plt.show()

TSC_L = pd.read_csv('TSC_list_7.csv')
A_Fit = TSC_L[['DTW', 'Feature-based']]
B_Fit = TSC_L[['K-Shape', 'Feature-based']]
fig = plt.figure(figsize=(12, 8))
A_Fit.plot()
B_Fit.plot()
plt.show()

```

Appendix B

Research Integrity and Technology RI Completion Certificate

CERTIFICATE of ACHIEVEMENT

This is to certify that

AHMED ALWAN

has completed the course

Research Integrity Modules

24 February 2018

End of course quiz - Engineering and Technology Grade: 95.00 %

