



Dissertation

Master in Computer Engineering - Mobile Computing

***Load forecasting: a cross-field study on server  
and energy load forecasting***  
***Impact of temporal factors on generalization ability and  
performance of regression models***

Cláudio Alexandre Duarte Silva

Leiria, September 2019

*This page has been intentionally left blank.*



Dissertation

Master in Computer Engineering - Mobile Computing

***Load forecasting: a cross-field study on server  
and energy load forecasting  
Impact of temporal factors on generalization ability and  
performance of regression models***

Cláudio Alexandre Duarte Silva

Dissertation developed under the guidance and supervision of Professor Carlos Fernando Almeida Grilo and Professor Catarina Helena Branco Simões da Silva, from the School of Technology and Management of the Polytechnic Institute of Leiria

Leiria, September 2019

*This page has been intentionally left blank.*

# Acknowledgements

This page is dedicated to the ones whom supported me during my journey. My thanks go firstly to the Professors Catarina Helena Branco Simões da Silva and Carlos Fernando Almeida Grilo for accompanying, supporting and encouraging me throughout this whole project. I would like also to thank them for their availability, solid patience and guidance in my times of uncertainty. By extent, I would also like to thank my lab colleagues at the Computer Science and Communication Research Center at IPLeia for the continuous support and companionship throughout the many long nights passed there working on this dissertation. A special thanks goes also to Eng. José Manuel Terras who provided the necessary dataset for the energy load forecasting case study.

A special thanks goes out also to my family and childhood friends for letting me work after hours and always supporting and encouraging me to see the end of this latest academic step.

Finally, I would like to thank the Escola Superior de Tecnologia e Gestão of the Instituto Politécnico de Leiria for providing the environment and working conditions that ultimately enabled this work.

*This page has been intentionally left blank.*

# Abstract

The server load prediction and energy load forecasting have available a wide range of approaches and applications, with their general goal being the prediction of future load for a specific period of time on a given system. Depending on the specific goal, different methodologies can be applied.

In this dissertation, the integration of additional temporal information to datasets, as a mean to create a more generalized model is studied. The main steps involve a deep literature review in order to find the most suited methodologies and/or learning methods. A novel dataset enrichment process through the integration of extra temporal information and lastly, a cross-model testing stage, where trained models for server load prediction and energy load forecast are applied to the opposite field. This last stage, tests and analyses the generalization level of the created models through the temporal information integration procedure.

The created models were both oriented to short-term load forecasting problems, with the use of data from single and combined months, regarding real data from Wikipedia servers of the year 2016 in the case of server load prediction and real data regarding the consumption levels in April 2016 of the city of Leiria/Portugal for the energy load forecasting case study. The learning methods used for creating the different models were linear regression, artificial neural networks and support vector machines oriented to regression problems, more precisely the Smoreg implementation. Results prove that it is possible to tune the dataset features, e.g., granularity and time window to improve prediction results and generalization. Results from this work, as well as an optimization approach through the use of genetic algorithms, normalization effects, split ratio vs cross-validation influence and different granularities and time windows were peer-reviewed published.

**Keywords:***(Load Prediction, Server load prediction, Energy load forecasting, Regression problems, Linear regression, Artificial neural networks, Support vector machines, Model generalization)*

*This page has been intentionally left blank.*



# Resumo

As áreas de estudo de previsão de carga em servidores e a previsão de carga elétrica em áreas urbanas, possuem inúmeras abordagens e aplicativos. O objetivo de uma forma geral, é a previsão de carga futura por um específico período no tempo num determinado sistema. Dependendo do objetivo, diferentes metodologias e/ou técnicas podem ser aplicadas.

Nesta dissertação, estuda-se a integração de informação temporal adicional em *datasets*, como forma de criar modelos mais generalizados/genéricos. As principais etapas envolvem uma profunda e detalhada revisão da literatura existente, com o intuito de encontrar metodologias e /ou métodos de aprendizagem mais adequados a este tipo de problema (previsão de carga), o enriquecimento de *datasets* através do um novo método de indução de informações temporais e por fim, uma última etapa de testes envolvendo a troca de modelos. Modelos treinados para previsão de carga de servidor e a previsão de carga elétrica são aplicados à área/espectro que lhe é oposto. Nesta última etapa é testado e avaliado o nível de generalização que os modelos criados por meio dos procedimentos de indução de informações temporais atingem/possuem.

Todos os modelos criados foram construídos para problemas de previsão de carga a curto prazo (*short term load forecasting*). Estes modelos foram desenvolvidos/treinados com recurso a dados dos servidores da Wikipédia de vários meses do ano de 2016 e a combinação destes meses. Os dados são provenientes do fluxo de dados/pedidos reais feitos a estes servidores da Wikipédia. No caso dos dados referentes a previsão de carga eléctrica, os dados também são reais e contêm o consumo real no mês de abril de 2016 registado pela EDP na cidade de Leiria - Portugal. Os métodos de aprendizagem que foram selecionados para criar os diferentes modelos foram a regressão linear, as redes neurais e as support vector machines orientadas para problemas de regressão, mais precisamente a implementação *Smoreg*. Os resultados confirmam que é possível ajustar a configuração dos *datasets*, por exemplo, granularidade e janela de tempo (*time window*) de forma a otimizar as previsões e a capacidade generalização do modelo. Os resultados deste trabalho, bem como uma otimização de configuração de *datasets* através do uso de algoritmos genéticos, efeitos de normalização, influência da aplicação do *split ratio vs cross-validation* e a aplicação de diferentes granularidades e janelas de tempo foram publicados em conferências internacionais e nacionais e em revista.

**Palavras-chave:** (*Previsão de carga, Previsão de carga em servidores, Previsão de carga de eléctrica, Problemas de regressão, Regressão linear, Redes neuronais, Support vector machines, Capacidade de generalização*)

*This page has been intentionally left blank.*

# Acronyms and abbreviations

- Amazon EC2** Amazon Elastic Compute Cloud. 23
- AML** Amazon Machine Learning. 49
- ANN** Artificial Neural Networks. 8, 9, 16, 17, 20, 30, 34, 36–39, 46, 52, 57, 74, 80, 81, 83–87, 91, 93–98, 100–102, 105, 106, 112, 135, 137–144, 147, 148, 150–154
- ANNSTLF** Neural Network Based Electric Load Forecasting System. 16
- API** Application Programming Interface. 3, 4, 48, 50, 60–62, 68, 70, 71
- ARIMA** Autoregressive Integrated Moving Average. 8, 16, 17
- ARIMAX** Autoregressive Integrated Moving Average with Exogenous Variables. 8
- ARMA** Autoregressive Moving Average. 8
- ARMAX** Autoregressive Moving Average with Exogenous Variables. 8
- AWS** Amazon Web Services. 23
- CNN** Convolutional Neural Networks. 49
- DVFS** Dynamic Voltage and Frequency Scaling. 20
- EDP** Energias de Portugal. 53, 56, 64, 66, 99, 113
- EES** Electrical Energy Storage. 64
- ETL** Extract, Transform, Load. 50
- GLP** Grid Load Profile. 64, 65
- GUI** Graphical User Interface. 53
- LSSVM** Least Squares Support Vector Machines. 17
- LTLF** Long-Term Load Forecasting. 6, 7, 17
- MAD** Mean Absolute Deviation. 11, 17

**MAE** Mean Absolute Error. 11, 75, 76, 79, 81, 101, 134, 136–138, 146, 149

**MAPE** Mean Absolute Percentage Error. 12, 17, 52, 76, 79–91, 93, 94, 96–98, 101–106, 108, 109, 111–113, 133–154

**ME** Mean Error. 11, 12, 75, 76

**MLlib** Machine Learning in Apache Spark. 49

**MLP** Multi-layer Perceptron. 46

**MPE** Mean Percentage Error. 12, 76

**MSE** Mean Squared Error. 11, 75

**MTLF** Medium-Term Load Forecasting. 6, 7

**NIST** National Institute of Standards and Technology. 21

**NLP** Natural Language Processing. 50

**NN** Neural Networks. 7, 21, 36

**PE** Percentage Error. 12

**PU** Processing Units. 36

**R<sup>2</sup>** R-Squared. 12, 13, 76, 80, 83–87, 90, 91, 97, 101, 102, 105, 106, 134–138, 143, 144, 146–150, 154

**$\bar{R}^2$**  Adjusted R-Squared. 12, 13, 76, 79–81, 83, 86, 93, 98, 101, 104, 105, 109, 111, 133–135, 137–139, 142, 146–148, 150–152

**RBF** Radial Basis Function. 46

**RKHS** Reproducing Kernel Hilbert Spaces. 44

**RMSE** Root Mean Squared Error. 11, 75, 76, 79, 81, 101, 134–138, 149

**RNN** Recurrent Neural Networks. 17

**RSE** Residual Standard Error. 11, 75

**SE** Support Vectors. 41, 46, 48

**SLA** Service Level Agreements. 2, 20, 57

**STLF** Short-Term Load Forecasting. 6–8, 15–17, 39

**SVC** Support Vector Classifier. 30, 41, 42, 46

**SVM** Support Vector Machines. 8, 9, 17, 21, 29, 30, 34, 39–41, 44–46, 48, 51, 52, 57, 74, 81–87, 95–98, 100, 101, 103–106, 112, 137–144, 148, 149, 151–154

**SVR** Support Vector Regression. 17, 46, 48

**URL** Uniform Resource Locator. 62

**UTC** Coordinated Universal Time. 58

**VM** Virtual Machines. 20, 22

**VSTLF** Very-Short Term Load Forecasting. 6

**WEKA** Waikato Environment for Knowledge Analysis. 25, 48, 50–53, 78

**WLM** Work Load Management. 23

*This page has been intentionally left blank.*

# List of Figures

3.1	Machine learning process . . . . .	26
3.2	Data integration example . . . . .	27
3.3	Data transformation using normalization . . . . .	28
3.4	Data reduction exemplification . . . . .	28
3.5	Classification example on linearly separable data . . . . .	30
3.6	Classification with non-linearly separable data . . . . .	31
3.7	Regression problem example . . . . .	31
3.8	Linear regression example . . . . .	35
3.9	Biological neuron vs Artificial neuron . . . . .	37
3.10	Multilayer feedforward implementation of 2-3-2 topology . . . . .	38
3.11	Backpropagation example . . . . .	38
3.12	Different types of transfer functions . . . . .	39
3.13	Convex function with a local/global minimum . . . . .	40
3.14	Hyperplane with respective SE . . . . .	41
3.15	Hyperplanes with initial and new support vectors . . . . .	42
3.16	Datapoint changed without direct impact in the hyperplane . . . . .	43
3.17	Two-dimensional data points that cannot be separated by a linear hyperplane . . . . .	44
3.18	Dimensional space before and after application of kernel function . . . . .	45
3.19	Examples of loss functions . . . . .	47
3.20	WEKA Explorer . . . . .	51
3.21	WEKA Classify example with Wikipedia dataset . . . . .	51
3.22	Pentaho time series framework incorporate in the WEKA GUI . . . . .	52
4.1	Generic solution architecture diagram . . . . .	53
4.2	Steps involved in the data transformation module . . . . .	54
4.3	Data example where granularity of 3 was applied . . . . .	55
4.4	Data example where time window of 3 was applied . . . . .	55
4.5	Final dataset composition diagram . . . . .	56
4.6	Wikipedia problem overview . . . . .	57
4.7	Wikipedia solution architecture diagram . . . . .	58
4.8	Wikimedia dataset storing naming nomenclature adopted . . . . .	58
4.9	Record structure example . . . . .	59
4.10	WikiGather API: single hour API functions . . . . .	61
4.11	WikiGather API: four-hour time-window API functions . . . . .	62
4.12	Collecting and cleaning module task overview with parallelism . . . . .	62
4.13	Data treatment: original row transformed into a dataset record . . . . .	63
4.14	Energy storage in a Grid Load Profile . . . . .	64

4.15	GRID during a day: baseload, renewable, intermediate load and peak load	65
4.16	EDP problem overview with respective implemented solution	66
4.17	EDP solution architecture diagram	66
4.18	EDP dataset record structure example without treatment	67
4.19	Data transformation API: grid search functions overview	69
4.20	Data transformation API: functions that integrate the configuration file	69
4.21	Data transformation API: create single dataset file	70
5.1	Example of Wikipedia’s dataset variations	73
5.2	Normalization example: before and after data	77
5.3	Influence of granularity and time window in MAPE and R-Squared for linear regression with 10 folds cross-validation	79
5.4	Influence of granularity and time window in MAPE and R-Squared for ANN with 10 folds cross-validation	81
5.5	Influence of granularity and time window in MAPE and R-Squared for SVM with 10 folds cross-validation	82
5.6	Time window distribution across all best models for single month with 10 folds cross-validation	85
5.7	Time window distribution across all best models for single month with 10 folds cross-validation on basic dataset	85
5.8	Time window distribution across all best models for single year with 10 folds cross-validation	87
5.9	Time window distribution across all best models for single year with 10 folds cross-validation	88
5.10	Correlation matrix for Wikipedia’s enhanced dataset	89
5.11	Influence of granularity and time window in MAPE and R-Squared for linear regression with 10 folds cross-validation on enhanced dataset	90
5.12	Influence of granularity and time window in MAPE and R-Squared for ANN with 10 folds cross-validation on enhanced dataset	92
5.13	Influence of granularity and time window in MAPE and R-Squared for SVM with 10 folds cross-validation on enhanced dataset	93
5.14	Time window distribution across all best models for single month with 10 folds cross-validation on enhanced dataset	95
5.15	Time window distribution across all best models for single month with 10 folds cross-validation on enhanced dataset	96
5.16	Time window distribution across all best models for single month with 10 folds cross-validation on enhanced dataset	97
5.17	Time window distribution across all best models for single year with 10 folds cross-validation on enhanced dataset	98
6.1	Example of EDP’s dataset variations	100
6.2	Influence of granularity and time window in MAPE and R-Squared for EDP basic dataset	102
6.3	Time window distribution for EDP basic dataset	103
6.4	Correlation matrix for EDP’s enhanced dataset	104
6.5	Influence of granularity and time window in MAPE and R-Squared for EDP enhanced dataset	105



6.6	Time window distribution for EDP enhanced dataset . . . . .	106
6.7	Model exchange architecture . . . . .	107
6.8	Wikipedia prediction results overview for basic dataset . . . . .	108
6.9	EDP prediction results overview for basic dataset . . . . .	110
6.10	Wikipedia prediction results overview for enhanced dataset . . . . .	111
6.11	EDP prediction results overview for enhanced dataset . . . . .	112
A.1	Influence of granularity and time window in MAPE and R-Squared for linear regression . . . . .	134
A.2	Influence of granularity and time window in MAPE and R-Squared for ANN	136
A.3	Influence of granularity and time window in MAPE and R-Squared for SVM	137
A.4	Time window distribution across all best models for single month with split ratio . . . . .	140
A.5	Time window distribution across all best models for single month with split ratio . . . . .	141
A.6	Linear regression learning method with actual and predicted values for baseline and best models. . . . .	142
A.7	Time window distribution across all best models for single month with split ratio . . . . .	143
A.8	Time window distribution across all best models for single year with split ratio . . . . .	144
A.9	Correlation matrix for Wikipedia’s enhanced dataset . . . . .	145
A.10	Influence of granularity and time window in MAPE and R-Squared for linear regression on enhanced dataset with split ratio . . . . .	147
A.11	Influence of granularity and time window in MAPE and R-Squared for ANN on enhanced dataset with split ratio . . . . .	148
A.12	Influence of granularity and time window in MAPE and R-Squared for SVM on enhanced dataset with split ratio . . . . .	149
A.13	Time window distribution across all best models for single month with split ratio and enhanced dataset . . . . .	151
A.14	Time window distribution across all best models for single month with split ratio and enhanced dataset . . . . .	152
A.15	Time window distribution across all best models for yearly dataset with split ratio and enhanced dataset . . . . .	153
A.16	Time window distribution across all best models for single year with split ratio and enhanced dataset . . . . .	154

*This page has been intentionally left blank.*

# List of Tables

4.1	Different page types from Wikipedia . . . . .	59
4.2	EDP dataset showing consumption values for 3 consecutive hours(time window 3 . . . . .	68
5.1	Dataset configuration attributes with respective values . . . . .	74
5.2	Different type of data normalization . . . . .	77
5.3	Result synthesis for all months using linear regression, basic dataset and 10 folds cross-validation . . . . .	79
5.4	Result synthesis for all months using ANN, basic dataset and 10 folds cross-validation . . . . .	80
5.5	Result synthesis for all months using SVM, basic dataset and cross-validation 10 folds . . . . .	82
5.6	Best models with respective time window and granularity configuration for Linear regression, ANN and SVM, using 10 folds cross-validation with enhanced dataset . . . . .	84
5.7	Average and best results for single year dataset using 10 folds cross-validation	86
5.8	Result synthesis for all months using linear regression, enhanced dataset and 10 folds cross-validation . . . . .	90
5.9	Result synthesis for all months using ANN, enhanced dataset and 10 folds cross-validation . . . . .	91
5.10	Result synthesis for all months using SVM, enhanced dataset and cross-validation 10 folds . . . . .	92
5.11	Best models with respective time window and granularity configuration for Linear regression, ANN and SVM, using 10 folds cross-validation on enhanced dataset . . . . .	94
5.12	Average and best results for single year dataset using 10 folds cross-validation	96
6.1	Average and best results for EDP basic dataset using 10 folds cross-validation	101
6.2	Average and best results for EDP enhanced dataset using 10 folds cross-validation . . . . .	104
6.3	Model exchange results overview for Wikipedia prediction with linear regression, ANN and SVM on basic dataset . . . . .	108
6.4	Model exchange results overview for EDP prediction with linear regression, ANN and SVM on basic dataset . . . . .	109
6.5	Model exchange results overview for Wikipedia prediction with linear regression, ANN and SVM on enhanced dataset . . . . .	110

6.6	Model exchange results overview for EDP prediction with linear regression, ANN and SVM on enhanced dataset . . . . .	111
A.1	Result synthesis for all months using linear regression, basic dataset and split ratio 70/30 . . . . .	134
A.2	Result synthesis for all months using ANN, basic dataset and split ratio 70/30 . . . . .	135
A.3	Result synthesis for all months using SVM, basic dataset and split ratio 70/30 . . . . .	137
A.4	Best models with configuration for Linear regression, ANN and SVM with basic dataset and split ratio . . . . .	139
A.5	Average and best results for single year dataset using split ratio . . . . .	142
A.6	Result synthesis on enhanced dataset for all months using linear regression with split ratio . . . . .	146
A.7	Result synthesis on enhanced dataset for all months using ANN with split ratio . . . . .	147
A.8	Result synthesis on enhanced dataset for all months using SVM with split ratio . . . . .	149
A.9	Best models with configuration for Linear regression, ANN and SVM on enhanced dataset with split ratio . . . . .	150
A.10	Average and best results for single year enhanced dataset using split ratio . . . . .	153

# Contents

<b>Acknowledgements</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>Resumo</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Objectives and Contributions . . . . .	3
1.2.1 Publications . . . . .	3
1.2.2 Software contributions . . . . .	4
1.3 Document structure . . . . .	4
<b>2 Load forecasting</b>	<b>5</b>
2.1 Load forecasting . . . . .	5
2.1.1 Types of load forecasting . . . . .	6
2.1.2 Load forecasting methods . . . . .	7
2.1.3 Forecasting performance . . . . .	10
2.1.4 Crucial factors for load forecasting . . . . .	13
2.2 Energy load forecasting . . . . .	14
2.2.1 Related work . . . . .	14
2.2.2 Areas of application . . . . .	18
2.2.3 Energy load forecasting challenges . . . . .	19
2.3 Server load prediction . . . . .	20
2.3.1 Related work . . . . .	20
2.3.2 Areas of application . . . . .	21
2.3.3 Server load prediction challenges . . . . .	23
2.4 Conclusion . . . . .	24
<b>3 Learning approaches</b>	<b>25</b>
3.1 Machine learning overview . . . . .	25
3.2 Supervised learning vs. Unsupervised learning . . . . .	29
3.3 Classification vs Regression problems . . . . .	30
3.4 Challenges and applicability . . . . .	32
3.5 Learning methods . . . . .	33
3.5.1 Linear Regression . . . . .	34
3.5.2 Artificial Neural Networks . . . . .	36

3.5.3	Support vector machines . . . . .	39
3.6	Software tools . . . . .	48
3.7	Conclusion . . . . .	52
<b>4</b>	<b>Proposed approach</b>	<b>53</b>
4.1	Solution architecture overview . . . . .	53
4.1.1	Data gathering, analysis and cleaning . . . . .	53
4.1.2	Data subset configuration and transformation module . . . . .	54
4.1.3	Tests, evaluations and model deployment . . . . .	56
4.2	The Wikipedia problem . . . . .	56
4.2.1	Solution approach in the Wikipedia problem . . . . .	57
4.3	The EDP problem . . . . .	64
4.3.1	Solution approach in the EDP problem . . . . .	66
4.4	Datasets transformation module . . . . .	68
4.5	Conclusion . . . . .	71
<b>5</b>	<b>Server load prediction: experimental setup and tests</b>	<b>73</b>
5.1	Test environment . . . . .	73
5.1.1	Evaluation metric definition . . . . .	75
5.1.2	Normalization vs Non-normalization . . . . .	76
5.2	Server Load Prediction on basic dataset with cross-validation . . . . .	77
5.2.1	Test runs and analysis for multiple months dataset . . . . .	78
5.2.2	Test runs and analysis for single year dataset . . . . .	86
5.3	Server Load Prediction on enhanced dataset with cross-validation . . . . .	88
5.3.1	Feature analysis . . . . .	88
5.3.2	Test runs and analysis for multiple months dataset . . . . .	89
5.3.3	Test runs and analysis for single year dataset . . . . .	96
5.4	Conclusion . . . . .	98
<b>6</b>	<b>Energy load forecast and model exchange: experimental setup and tests</b>	<b>99</b>
6.1	Test environment . . . . .	99
6.2	Energy load forecast on basic dataset with cross-validation . . . . .	100
6.2.1	Results analysis . . . . .	101
6.3	Energy load forecast on enhanced dataset with cross-validation . . . . .	103
6.3.1	Feature analysis . . . . .	103
6.3.2	Results analysis . . . . .	104
6.4	Model exchange . . . . .	106
6.4.1	Basic dataset result analysis . . . . .	107
6.4.2	Enhanced dataset result analysis . . . . .	110
6.5	Conclusion . . . . .	112
<b>7</b>	<b>Conclusions and Future work</b>	<b>113</b>
<b>Appendix A</b>	<b>Server Load Prediction with split ratio</b>	<b>133</b>
A.1	Server Load Prediction on basic dataset for split ratio . . . . .	133
A.1.1	Test runs and analysis for multiple months dataset . . . . .	133
A.1.2	Test runs and analysis for single year dataset . . . . .	141

A.2 Server Load Prediction on enhanced dataset for split ratio . . . . .	144
A.2.1 Feature analysis . . . . .	144
A.2.2 Test runs and analysis for multiple months dataset . . . . .	146
A.2.3 Test runs and analysis for single year dataset . . . . .	152

*This page has been intentionally left blank.*



# Chapter 1

## Introduction

“ *An unsophisticated forecaster uses statistics as a drunken man uses lamp-posts - for support rather than for illumination.* ”

---

Andrew Lang, (1844 – 1912)

Since the earliest of times, humanity has used forecasting in daily life decisions. In those early days, forecasting was often associated with fortune-telling. However, nowadays the difference between fortune-telling and forecasting is clear and stands on the transparency of the methods used to reach the predictions. In forecasting, it is possible to replicate the forecasting process by using the same model and data and thus obtaining the same or equivalent results. However, with fortune-telling that is not the case, the crystal ball and the fortune-teller’s words are the only things to rely on.

Forecasting brings humanity the ability to anticipate events, to better adapt to changes, and to possibly alter outcomes. Be it a farmer in ancient Rome who predicted a storm because of his back pain, or an XXI century meteorologist whose job is to predict the weather. Forecasting goes as far back as 650 B.C., when the Babylonians tried to predict weather changes based on optical phenomena and the outlook, quantity, and patterns of clouds in the sky [1].

Nowadays, forecasting appears in our society in many forms, sometimes implicitly without us even noticing it. For instance, when we decide to take an umbrella when leaving the house even though it is not raining. There can be a multitude of factors on why we took the umbrella, for example, it might have rained all week before. No matter the reason, we probably based our decision in past experiences and/or existing information.

The act of forecasting is also found and used in a variety of different areas, such as, commercial, industrial, scientific, and economic activities [2]. It thus, results in a variety of distinct types of forecasting. In this work, we focus on load forecasting, which can be defined as the science or art of predicting the future load in a specific period ahead on a given system [3]. Forecasts are obtained by analysing past and/or present data in search for patterns or relevant relations in the data. These patterns can be used to forecast a few

minutes ahead or as far as 50 years into the future. Usually, an accurate load model is created to mathematically represent the relationship between all influential variables and the load [3].

The forecasting of load derived from the need to find a breakeven between supply and demand in a specific area, which is usually hard to achieve. It emerges in a variety of areas, such as, electric load forecasting, load forecasting in computer servers, commonly known as server load prediction, sales forecasting, dam water levels forecasting or traffic forecasting. [4–8].

## 1.1 Motivation

In this work, we focus on problems associated to energy load forecasting and server load prediction. Energy load forecasting aims to determine the energy demand if the system and therefore determine the necessary supply levels. Energy in general, and electric power particularly, has an incredible importance in modern society. All electrical appliances, and a vast number of infrastructures, have it as one of their main supports to operate. Electricity is a given in our lives and it is almost impossible to imagine our society without it. This electric power system is a complex system that spreads from the end customer, to the distribution infrastructures that transport it through the electric network, and finally to the power plant, which is responsible for the production of electricity. At any given time, it is crucial to preserve the equilibrium between consumption and production, or supply and demand, otherwise, it could become difficult to guarantee the stability of the power system. In situations where this equilibrium is unsettled, the worst-case scenario of a blackout could occur, leading to a monetary loss of billions and uncountable damage to our society.

Server load prediction on the other hand sets its goal in determining the future load of servers/datacentres, either in terms of data volume or number of connections. The load in these types of environments tends to be more mutable than other fields such as supercomputing settings or common science environments [9], where the tasks to be supported are easier to predict ahead. Datacentres technicians challenge themselves every day to minimize waste in resource usage and maximize their profit range. Efficient resource management is key in the viability of a datacentre, ensuring that workloads have the necessary resources without breaching Service Level Agreements (SLA). To efficiently adjust resource allocation, prediction algorithms are used to forecast incoming load and/or resources needed to guarantee the performance of the datacentre for every kind of specific load [10].

However, since our society continuously develops a bigger and healthier conscience towards our planet, datacentres nowadays try also ensure that they fulfil their part to keep this world a healthy place for future generations to come. The use of load forecasting lets datacentres minimize resource consumption, reducing energy consumption and, as a result, reduce their carbon footprint [11].

## 1.2 Objectives and Contributions

In this work we have two fundamental areas of study of application of load forecasting systems. Hence, we can set two main goals, namely (i) to develop a load forecast architecture that allows for the prediction of load in a web server, using previous records of users' connections and transferred data; and (ii) to develop a load prediction system that can determine the energy demand in the city of Leiria/Portugal based on historical data on energy consumption.

With these goals we can outline the main contributions of this work:

1. Review of the state-of-the art in both scenarios, including the most relevant works and techniques to tackle energy load prediction and server load prediction;
2. Definition, characterization and analysis of two case studies that represent each individual area;
3. Definition of data pre-processing methods/guidelines applied in different areas of study;
4. Develop and publish two frameworks/Application Programming Interface (API). One that automates the data creation and manipulation on numerical datasets and another one that gathers, manipulates and validates the Wikimedia datasets;
5. Create a novel methodology that increases model performance and generalization by adding extra temporal information to datasets;
6. Cross train and test the two case studies for model generalization analysis;
7. Publish the work carried out in the scientific available means;

### 1.2.1 Publications

From this work resulted the publication and respective presentation of three scientific papers and one journal article:

1. **Conference Paper:** "*A Comparison Approach for Load Forecasting*": CENTERIS 2018 - Conference on ENTERprise Information Systems/ProjMAN 2018.
2. **Conference Poster:** "*Granularity and time window on forecasting regression problems*": RECPAD 2018 – Portuguese Association for Pattern Recognition.
3. **Conference Paper/Book publication:** "*Server load prediction on Wikipedia traffic: influence of granularity and time window*": SoCPaR 2018 - Proceedings of the Ninth International Conference of Soft Computing and Pattern Recognition.
4. **Journal:** "*Model Optimisation for Server Loading Forecasting with Genetic Algorithms*": International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM), ISSN: 2150-7988, Volume 11, pp. 178 - 191, 2019.

## 1.2.2 Software contributions

Sideways to the published work, contributions for future researchers were made in the form of two different APIs:

1. The **WikiGather API** was created from the need to download large raw data quantities from the Wikimedia foundation, with a simple and efficient tool. The API allows an easy integration in researchers python projects. It was published and publicly made available through GitHub at `WikiGather:URL`
2. The **Dataset transformation API** is a simple and efficient tool that adds extra temporal information to time-series datasets, although it can also be applied in any numeric only/non-time-series datasets. The API allows an easy integration in researchers python projects. It was published and publicly made available through GitHub at `Dataset transformation API:URL`

## 1.3 Document structure

The rest of the document is structured as follows. In Chapter 2, an explanation in a detailed manner regarding the load forecasting problem is presented. Error measures, forecasting accuracy and cyclic behaviour are thoroughly described. Subsequently, server load prediction and energy load forecasting are presented and explained, along with a literature review for each area.

Chapter 3 gives the reader the knowledge needed in machine learning for further chapters. An introduction to machine learning is made, covering supervised and unsupervised learning, benefits and disadvantages; regression and classification problems, time series problems and lastly the learning methods adopted in this work.

Chapter 4 focuses on the proposed approaches, which are presented and described in detail. The description and justification of the implementation of the above-mentioned APIs into the overall architecture is also presented in the chapter .

In Chapter 5 all the steps taken throughout server load prediction testing and evaluation process are presented. These steps involve, amongst others, dataset generation, evaluation metrics definition, and dataset optimization. Results regarding each test environment are presented and discussed.

In Chapter 6, the energy load forecasting case study is presented and analysed. Testing and evaluation procedures are identical to server load prediction. Subsequently, the model exchange scenario is presented, tested and analysed.

Finally, Chapter 7 presents the conclusions and future work.

# Chapter 2

## Load forecasting

In this chapter a literature review is conducted, based on the most recent published research studies on load forecasting. An introduction to the definition of load forecasting is made, followed by the description of the most common approaches used in load forecasting problems, including the main challenges for a successful forecast of load, description of several load forecasting methods, and the different types of load forecasting. Subsequently, an analysis of server load prediction and energy load forecast is made.

### 2.1 Load forecasting

Load forecasting emerged as a support to predict and prevent resource imbalance [12]. Previous methodologies, such as the pull system, which initiated a new task/work only when there was customer demand for it, presented chronic flaws, such as the way they handled unexpected events. For instance, new milk was only made available, when costumers had nearly emptied the shelf and demand for milk existed. This meant that the restocking process was not proactive, but merely reactive. The restocking process was only triggered (the pull needed in a pull system) when shortage and demand occurred. This method only worked if milk was available and reachable to the restocking staff. If this was not the case, for example, there was no milk in the warehouse, the process of ordering and shipping new stock could mean loss in revenue. To overcome these drawbacks, load forecasting methods can help predict consumption rates, preventing resource imbalance, along with several unforeseen events [13].

A common challenge in load forecasting is the seasonality in the data. Load series tend to be fairly intricate and present different levels of seasonality, e.g., the load in a specific hour can be correlated to the load of the previous hour, but also correlated to the load of the exact same hour from the same day on the previous week [14].

The implications for organizations that do not use load forecasting or have incorrect forecasts can be extremely negative. The inability to identify these problems may trigger a snowball effect on the long run. Overestimation might lead to overstock, which could be impossible to resell with profit. On the other hand, underestimation could result in revenue loss or in the incapability to fulfil the organization's goals. When forecasting load, it is of

the utmost importance to keep a balance between resources and demand, avoiding stock outs or over stocking.

The above mentioned factors led to load forecasting becoming one of the major research fields in various areas, such as, electric load forecasting, load forecasting in computer servers, also commonly known as server load prediction, sales forecasting, dam water levels forecasting, traffic forecasting, etc. [4–8].

In any category, a precise load model that depicts the mathematical relationship between the influential variables and the load is needed. This accurate relationship is commonly defined by the role each variable plays in the load model. With the construction of the mathematical model, model parameters are identified and calculated through different estimation techniques [3]. These parameter estimation techniques provide the necessary tools that enable the efficient use of data, in the aid of constant estimation, used in the mathematical model creation process [15]. In the last 50 years, parameter estimation techniques used in load forecasting have been mostly based on the least squares minimization criterion [15].

Different approaches and applications of load forecasting have appeared through time, with the general goal remaining the prediction of future load. By definition, load forecasting is exactly that, i.e., the science or art of anticipating the future load on a given system, for a specific period of time [3]. Depending on the more specific goal, different time ranges can be used. Industry usually requires forecasts ranging from short-term forecasts, just a few minutes, hours, or days ahead within one, two, or three years of available data [16], up to long-term forecast, that can predict up to 50 years ahead. However, due to the fact that most variables such as weather, traffic, or energy demand are progressively harder to accurately foresee as time advances, long-term predictions tend to be less accurate.

### **2.1.1 Types of load forecasting**

Load forecasting methods can be divided according to the forecast time horizon, which is usually problem dependent. It can be set by the specific goal of the prediction to achieve, or it can be constrained by the availability and accuracy of data. Therefore, each forecast horizon is used for different purposes and/or restrictions. According to Gross [17], these horizons can be divided as:

- Very-Short Term Load Forecasting (VSTLF);
- Short-Term Load Forecasting (STLF);
- Medium-Term Load Forecasting (MTLF);
- Long-Term Load Forecasting (LTLF).

VSTLF has a forecast horizon lower than an hour, STLF has a forecast horizon of up to 30 days, MTLF has a forecast horizon of up to one year, and LTLF has a forecast horizon higher than year. LTLF and MTLF can be usually found in transport and distribution capacities, schedule maintenance, plan production among others. STLF can be used in the planning of daily tasks. VSTLF on the other hand, with a maximum time horizon of one hour, can be used in cases where adjustments are needed every hour, such as, resource adjustment in mainframes.

Along with the distinct types of load forecasting, a large variety of load forecasting methods exist that aid in the forecast process. Some methods evolved with the goal of achieving better results in a certain type of load forecasting, such as the case of time series, which achieve better results in VSTLF and STLF forecasts.

## 2.1.2 Load forecasting methods

Over the last decades, numerous forecasting methods have been created. The econometric approach and end-use methods are the most commonly used for medium and long-term forecasting. The econometric model uses a system of relationships between variables such as exchange rates or inflation and tries to apply different methods, e.g., statistical techniques to find relevant data econometrics among the economic variables [18]. For the end-use approach, a list is created containing different users for a product/instance, which is the target of the forecast. The users are then asked about their individual preferences; from this information, a demand forecast is ascertained [19].

Short-term forecasting uses a selection of methods, such as, regression models, Neural Networks (NN), time series, similar day approach, fuzzy logic, statistical learning algorithms or expert systems [16]. The continuous development, investigation and improvement of tools, lead to more accurate load forecasting methodologies.

When taking a statistical approach on load forecasting, a mathematical model is needed to represent the different variables and the load. Two commonly mentioned categories when using this approach are multiplicative and additive models. The main difference between these two models is how the load is represented. In the case of additive models, the load is a sum of different instances of the load, while in multiplicative models, the load is a multiplication of the different variables.

An example of an additive model, was defined by Chen as a load constituted by the sum of four different sources [20]:

$$L = L_n + L_w + L_s + L_r, \quad (2.1)$$

where  $L$  represents the total load,  $L_n$  represents the "normal" load,  $L_w$  is the weather sensitive load, while  $L_s$  is the special load, which only occurs when there is a big deviation from the usual load pattern;  $L_r$  is the load that originated from noise.

On the other hand, an example of a multiplicative model is:

$$L = L_n * L_w * F_s * F_r \quad (2.2)$$

where  $L_n$  and  $L_w$ , are the normal load and weather sensitive load, respectively.  $F_s$  and  $F_r$  are correction factors, which have positive values and can decrease or increase the final load. These correction factors are based on special events and random fluctuation respectively.

Lastly, another approach that can be taken when analysing load forecasting methods is to divide them between STLF methods and MTLF and LTLF methods.

## Short-term load forecasting methods

STLF uses artificial intelligence and statistical techniques, such as, regression methods, similar day approach, time series, Artificial Neural Networks (ANN), expert systems or Support Vector Machines (SVM).

**Regression methods:** One of the most used statistical techniques for STLF is regression. Regression methods use statistical processes for estimating the relationships between variables. Through these methods, it is possible to comprehend how the value of a dependent variable varies with the value oscillation of an independent variable. Electric companies use widely these types of techniques. Their load forecasts commonly present the relationship between load consumption and the other variables, e.g., customer class, weather or day type. A large variety of regression models try to predict the next day peak consumption levels. These regression models often include deterministic influences such as holidays, exogenous influences, for instance, weather, and stochastic influence, for example, average load. Common methods include linear and non-linear regression or Bayesian methods [16].

**Similar day approach:** The similar day approach is based on the search of historical data for a certain day or days within a certain interval of time that contains similar features as the day to be forecasted. Such features could be the day of the week, the date or even the weather. In this approach, the total load of a day is considered as forecast. Forecasts can also be regression procedures or linear combinations composed of several similar days [16].

**Time series:** When using time series, it is crucial that the data used has an internal structure, i.e., seasonal variation, autocorrelation or trend. Time series forecasts detect and base their analysis in such structures. The most common time series methods are Autoregressive Integrated Moving Average (ARIMA) [21], Autoregressive Moving Average (ARMA) [22], Autoregressive Integrated Moving Average with Exogenous Variables (ARIMAX) [23] and Autoregressive Moving Average with Exogenous Variables (ARMAX) [24].

ARMA models are used for stationary processes, while ARIMA models, which are an extension of ARMA, are used for nonstationary processes. ARIMA and ARMA use, in most cases, only two input parameters, which are the load and time. ARIMAX is used in the more traditional time series models and depend mainly on the time of the day and the weather. ARMAX models are frequently used with evolutionary programming, which is a stochastic optimization algorithm [25]. Evolutionary programming is used to identify the ARMAX model parameters [16].

**Expert systems:** Expert systems started being used in the 1960's for geological prospecting and computer design. Expert systems are software created with human knowledge, usually, experts in the field of application, which integrate procedures and rules used and/or defined by them. The created software is often able to make forecasts without human aid. Expert systems are typically based on a set of rules, which are created using mainly heuristics [16].

**Artificial neural networks:** ANN are inspired by brain functions and the brain itself. ANN are algorithms which represent, and process information based on methods created from those brain analogies. Most ANN systems are composed by processing



units commonly referred as artificial neurons, analogous to biological neurons, which are interconnected to other computing units via connection strengths, analogous to synaptic junctions/axon terminal. Comparably in a biological neuron the dendrites receive signals, the cell body processes them, and the axon is used to send signals out to other neurons.

ANN make use of a variety of different architecture types, a common type used is feedforward. Feedforward architectures have for example the input units/layers connected to a set of output units/layers, or the input units/layers are connected to the hidden units/layers, which are connected to the output units/layers. Besides different architectures, ANN can also use different types of learning algorithms, which are used to train the ANN system. Error-correction learning or classical Hebbian learning rule are examples of learning algorithms [26]. In load forecasting the most common training algorithm is backpropagation [16].

**Support vector machines:** SVM are a technique for solving regression and classification problems and originated from Vapnik's statistical learning theory [27]. SVM use a high dimensional space in which they create a non-linear mapping of the data, through the aid of kernel functions. Kernel functions enable SVM to perform operations in an input space, instead of a high dimensional feature space, decreasing the area in which the mapping is done.

### **Medium and long-term load forecasting methods**

When performing a medium or long-term load forecast, the most common methods used are econometric modelling, end-use modelling and a combination of both methodologies.

**Econometric method:** The econometric method uses statistical techniques with economic theories to make load forecasts. Econometric methods estimate the relationship between the load and the dependent variables. The estimation process is done through time series or least-square methodologies. A common approach in econometric methods is to aggregate different econometric models from different areas that are related to the load into a single one [16].

**End-use method:** Detailed information regarding the end user or the main objective of the area in which the load was generated, are the main columns in which the end-use methodology builds its forecasts. For instance, information concerning the end user, e.g. gender, ethnicity or age, is used as basis for the forecasts. The causes or the end use in load generated in a web-server are also factors that the end-use methodology uses when making forecasts. Through end-use methodologies, it is possible to understand, which user category influenced the load and to what purpose [16]. This type of methodology is very sensitive regarding the quality and the amount of data. However, the forecasts tend to be very accurate.

**Econometric and end-use methodologies:** Econometric and end-use methodologies are dependent to a large amount of data regarding the load. Human interaction is needed when using these methodologies due to their high complexity and the unavailability of essential information. Experienced staff can usually fill the missing information with satisfactory results.

Another challenge when using a combination of these two methods is the different formats that each load might have [16]. The statistical model of Feinberg [28] is used in these situations. Using this model increases the forecast accuracy and combats some of the above-mentioned problems. This method uses mainly historical data to learn the most adequate parameters for the created model.

### 2.1.3 Forecasting performance

The process of evaluating a load forecasting method is a challenging task. When choosing a viable metric to measure forecast errors, the choice is in most cases difficult, since different measures can be useful in some problems and disadvantageous in others [29]. Gardner proposed that performance of results is directly connected to the type of data and error measures used [30].

Reducing the forecast error measures to just one is in most cases undesirable, since distinct types of forecast errors may give the results different dimensions. Each measurement has an individual method of calculation, which might produce complementary information. By just limiting the error measures to one, valuable information could be sacrificed/lost. When analysing each forecasting measure, in order to select which one is the most suitable to the problem, it is possible to get a more complete picture concerning the forecasting methods.

Nevertheless, error metrics should not be the only ones used for model evaluation; the opinion of experienced staff can sometimes debunk promising errors. For instance, researchers concluded that lower inventory cost and better customer service level, were not directly correlated to a better/lower error performance [31]. Thus, low forecasting errors should be considered as a tool for a goal and not a goal itself.

Nonetheless when analysing, forecasting performance from the statistical perspective, the focus goes mainly on error calculation. This error represents the difference between the actual value and the forecasted value for a certain period of time. The formula below represents the calculation of the forecast error.

$$E_t = A_t - F_t \quad (2.3)$$

where  $E$  is the error of the forecast or residual at the period  $t$ ,  $A$  is the actual value at the period  $t$  and  $F$  is the forecast value at period  $t$ .

In addition to the forecast error, there are other error metrics used to better evaluate the accuracy of a forecast. These metrics can be split between absolute errors and relative errors. Absolute errors are always in the same scale as the data used as input. This means that it is impossible to compare results from data with different scales. Relative errors, on the other hand, are commonly used to compare results between different datasets since they are scale independent. The disadvantage of using this type of errors, relies in the fact that they tend to be infinite or undefined the closer the actual value is to zero. This is due the formula used in calculating the percentage value:

$$P_t = 100 * (E_t/A_t) \quad (2.4)$$

#### Load forecasting metrics

As aforesaid, when evaluating the performance of a model, a statistical analysis is the most common approach. There are two types of metrics used: the first one is the absolute measurement of the errors, and the second is the measurement of the relative errors/percentage errors [32].

### Absolute errors

The most common absolute errors used in load forecasting are Mean Error (ME), Mean Squared Error (MSE), Mean Absolute Error (MAE) or Mean Absolute Deviation (MAD), Root Mean Squared Error (RMSE) and Residual Standard Error (RSE).

The ME is calculated through the equation:

$$ME = \sum_{t=1}^n \frac{e_t}{n}, \quad (2.5)$$

where  $n$  is the number of measurements/iterations,  $e_t$  is the forecast error at the period  $t$ , which was presented in Equation 2.3. Its value should be close to zero. This proximity of zero means the absence of any kind of trend in the prediction values, i.e., the absence of divergence of the results, or bias.

The MSE is calculated as follows:

$$MSE = \sum_{t=1}^n \frac{e_t^2}{n}. \quad (2.6)$$

MSE measures forecasting errors and its variance, it is related to standard deviation of forecast errors. The mean forecast error is only usable when it deviates from zero and is sensitive to errors and outliers smaller than one [13].

The equation of MAE is as follows:

$$MAE = \sum_{t=1}^n \frac{|e_t|}{n}. \quad (2.7)$$

It measures variance just like MSE. MAE is usually easier to interpret than MSE. This is mainly due to the fact that the error has the same dimension as the forecast. Contrary to MSE, MAE is not as sensitive to outliers as MSE. This is mostly due to the use of an absolute value instead of a quadratic value [13].

The RMSE, which is calculated as:

$$RMSE = \sqrt{\sum_{t=1}^n \frac{e_t^2}{n}}, \quad (2.8)$$

is the mean of the total error values squared. The evolution of the quadratic error contributes to emphasise the disparities between the target value and the result obtained by the forecast model.

Finally, the RSE, whose formula is:

$$RSE = \sqrt{\sum_{t=1}^n \frac{e_t^2}{n-1}}, \quad (2.9)$$

measures the dispersion of error values around zero. The RSE value is a measure of standard deviation and is used to generate forecast intervals on the mean error, thus, ensuring that

the distribution of error values approaches a normal distribution. The parameters used for the absolute analysis of the error should not be seen in an individualized way but evaluated as a whole [32].

It is frequent to adopt the minimization of error during the learning process as decision criteria of the forecast model. However, obtaining good results in terms of error, does not guarantee a good model performance. Sometimes, in the adjustment phase, error values can be reduced by increasing the complexity of the model, which does not always translate into an increase in forecast performance [32]. This type of occurrence is usually referred to as over-fitting. Other statistical metrics are directly linked to the analysis of the relative error measure. According to the comparative study developed by Hippert, the most widely used benchmark in load forecasting problems is the Mean Absolute Percentage Error (MAPE) [33].

### Relative errors/Percentage errors

The most common relative errors used in load forecasting are Percentage Error (PE), Mean Percentage Error (MPE), MAPE, R-Squared ( $R^2$ ) and Adjusted R-Squared ( $\bar{R}^2$ ).

The PE, which formula is as follows:

$$PE_t = \frac{(Y_t - \hat{Y}_t)}{\hat{Y}_t}(100), \quad (2.10)$$

where  $Y_t$  is the actual value and  $\hat{Y}_t$  is the predicted value. This metric represents the ratio between the target value and the result of the forecast at the period  $t$ .

MPE, which is calculated as:

$$MPE = \sum_{t=1}^n \frac{PE_t}{n}, \quad (2.11)$$

and just like ME, the value of MPE should be close to zero.

MAPE, which is very similar to MPE is calculated through the equation:

$$MAPE = \sum_{t=1}^n \frac{|PE_t|}{n}. \quad (2.12)$$

Contrary to MPE, the MAPE metric uses the absolute values/modulus of the errors to avoid error cancellation.

Lastly the metrics  $R^2$  and  $\bar{R}^2$  are presented. The metric  $R^2$ , also commonly referred to as coefficient of determination, is calculated through the equation below.

$$R^2 = 1 - \frac{\sum_{t=1}^n (Y_t - \hat{Y}_t)^2}{\sum_{t=1}^n (Y_t - \bar{Y})^2}, \quad (2.13)$$

where  $\bar{Y}$  is the sample mean of the predicted/observed values. This metrics depicts the percentage of the variance of the observed value which can be described by the input variables/predictors. The main problem with this metric is that adding input variables

increases the value of  $R^2$ , without differentiating between useful and useless input variables.  $\bar{R}^2$ , which is described in the equation below, tackles this exact problem.

$$\bar{R}^2 = 1 - (1 - R^2) \left[ \frac{n - 1}{n - (k + 1)} \right], \quad (2.14)$$

where  $n$  is the number of the sample size tested and  $k$  is the number of independent variables that are present in the regression equation, i.e., all the variables that are not the objective class/prediction.

Only an overall analysis of all the previous indicators is revealing of the performance of a model. In the chapter of tests and results each metric is analysed in a more detailed manner adjusted to the problems studied in this work.

### 2.1.4 Crucial factors for load forecasting

Throughout the evolution of load forecasting, numerous factors have been researched, in terms of influence in the load forecasting process [34–42]. Geng [43] divided these factors into three types, short-term influence factors, middle-term influence factors and long-term influence factors:

- The **short-term factors**, usually appear in a single timed situation, and tend to not possess a temporal dimension. An example of this are sudden weather changes.
- The **middle-term factors**, tend to be more continuous than short-term factors. These types of factors have a temporal dimension, as in the case of seasonal climate changes.
- The **long-term factors**, as the name suggests, occur throughout a long period of time. During this period, more than one forecast is usually made. The temporal dimension in these cases, is larger. For example, the population of a country or the GDP (gross domestic product) of a country.

According to Cavallaro [44], the main factors that can influence load, depending on the type of problem are:

- **Economic factors**, such as, newly created industrial sites, demand side management, economic trends, i.e., expansion or recession, and price changes in electricity. Economic factors influence may vary depending on the location, for instance, the electricity consumption, throughout the day in an underdeveloped country has its maximum between 6:00 pm to 9:00 pm. However, in developed countries the maximum occurs between 11:00 am to 4:00 am [45]. The cost of electricity also influences the load variations. With high electricity prices, the consumption is usually lower when compared to lower prices. Companies usually make use of different prices for different periods of the day. For instance, the night times are usually cheaper than day times.

The researchers in [45] defined the most impactful categories as GDP (Gross Domestic Product), GNP (Gross National Product), economic trends, such as, expansion and recession, petroleum price, the number of cold and hot days, electricity prices, industry development and population.

- **Temporal (calendar) factors**, for example, the yearly holy-days such as Christmas, the seasonal load differences between summer and winter, holydays, the increase or decrease of daylight hours, the weekly cycles, the difference between weekends and weekdays, the differences between day and night. Temporal factors can range from a daily spectrum, up to several years. These factors are for instance, working days, time factors, such as, daily routine activities, night activities versus day activities, the season of the year, weekend, weekdays, etc. Usually, through analysis of these patterns, it is possible to identify the working hours, resting hours or leisure hours.
- **Meteorological factors**, for instance, humidity, temperature, snow, cloud formation, and rain. Load forecasting might also be influenced by factors from meteorological origin, such as, humidity, snowfall, wind speed, rain, cloud formations or temperatures. In addition, to their individual influence, meteorological factors often correlated with each other. For example, rain is correlated to humidity and cloud formations are correlated to temperature.
- **Random events** such as, television shows, sporting events, natural disasters, tend to be factors with high impact. Yet, some of them can be hard to foresee, e.g., natural disasters.

## 2.2 Energy load forecasting

One of the first people studying load forecasting in the electric industry was Samuel Insull. He discovered that consumers exhibited a pattern when consuming energy, for instance, households and commercial buildings consumed more electricity during daytime, whilst industry and street lighting consumed more electricity during the night time [46].

Since those early times, the electric industry has come far. Nowadays, our modern society depends on electricity, and one of the challenges the industry faces is the difficulty and cost to store larger quantities of electricity, making it crucial to keep an equilibrium between consumption and production. The maintenance of this equilibrium is extremely important since it guarantees the stability of the power system and minimizes costs. Energy load forecasting helps in finding this equilibrium.

Energy companies create accurate models for electric power load forecasting, which are essential to their planning and operation. It aids them in making important decisions regarding the generation and or purchasing of electricity. In addition to keeping power systems stable, load forecasting also helps with electricity distribution, energy suppliers, security of supply, system planning, power market design and power market operation [16].

### 2.2.1 Related work

One of the earliest studies published regarding energy load forecasting, was made by Reyneau in the year 1918. His work consisted in predicting load on residence circuits [47]. Between 1918 and 1941 not many articles were published. The researchers from that period studied mainly load forecast of energy consumption and cost comparison resulting from the analysis of load forecast [48–52].

After the 1940's the use of weather factors, started to become a crucial component in energy load forecasting. Dryar's article was one of the first to use these factors [53]. He used successfully energy load forecasting together with diverse weather factors, such as, temperature, wind speed and degree of cloudiness, to predict energy peak consumption. One of his conclusions was that on days with open sky, the morning peak would occur at 8:30 am, and on hot summer days it would occur at 11:00 am. These forecasts were all related to data from the year 1944. In the same year, Davison, used the same approach to make his own energy load forecasting regarding electric utilities.

Between the 1940's and 1950's more and more researchers used external variables to increase their models [54–61]. Some researchers create models specifically for a season, for instance, in the article [56] a model was created using weather variables, such as, winter temperatures, wind speed or sunshine. Their forecast was also related to a season; in this case, it was the electricity consumption with the purpose of heating.

In four articles that were published in 1955, three of them, studied the influence of weather and atmospheric conditions on the load forecast [62–64]. The fourth article analysed the effect of temperature on the consumption of electrical energy [65].

From that period, Hooke's study is one that stands out amongst others. He published one of the first studies where critical factors for energy load forecasting were defined [66]. He considered that, firstly, the data used should come from as many different sources as possible and, secondly, the staff involved in the load forecasting process should also come from different areas. The forecast should be based on studies of the various components which together would determine the demand for electricity. Along with the demand, a forecast related long-term growth of industry should also be made. With these factors, it would be possible to create a load forecast regarding system-peak values.

In the 1960's, time series approaches started to be published, along with articles using different forecast horizons [67–69]. An article from that corresponding period, described the calculation of smoothing and prediction operators when applying time series to energy load forecast [70].

Chen and Winters tried to forecast the peak demand for an electric utility with a hybrid exponential model. They forecasted the peak load demand for an electric utility, with a STLF horizon, i.e., one day in advance. When using their method, they achieved better results with less data than other methodologies already in use by other energy companies [71].

In 1966, Heinemann and Nordman, proposed a regression approach for energy load forecasting. They concluded that during summer, air conditioning contributed significantly to an electric utility system peak load. They separated the total system load into two components, temperature-sensitive load and non-temperature-sensitive load. Through the examination of historical data, they found that temperature-sensitive loads depended not only upon coincident but also antecedent weather conditions [72].

In 1968, one of the first approaches of energy load forecast with an online component was published. The researchers, created a statistical method for online load prediction, based in spectral decomposition. In the spectral decomposition method, the load was represented by the sum of a long-term trend, a component fluctuating with the day of week, together with a component which varies from day to day and hour to hour. Their method

was implemented as part of an automatic load-dispatching program for the control of a power system, and even though their offline tests gave satisfactory results, their method was affected by measurement errors when applied online [73].

After the 1970's, the time-series approach started to resurface in energy load forecasting [74]. Before that, researchers based their studies mainly in statistical methods. Nevertheless, the recurrence of time series did not mean that the already existing techniques stopped being used.

In [75] the use of the Box-Jenkins time series analysis technique in STLF was demonstrated. The model used by them was ARIMA. One of their focal points in the model was the inclusion of weather variables in the model, in which the forecast was a linear combination of load and temperature values and forecasting errors. The load in their study was the hourly mean electricity generated by the energy company, as well as, imported electricity. The created forecast was used in the daily production planning. The researchers concluded that the use of Box Jenkins time series analysis in load forecasting led to a simple, fast and accurate forecasting algorithm, which achieved also good results in other forecasting problems arising from electrical energy systems.

The main research areas from that decade were energy load forecasting with the use of weather information [76–82], energy load forecasting with focus on the statistical methods used [83–93], being the most common approach used the stochastic models, energy load forecasting with an online perspective [94–98] and lastly, making energy load forecast using the time-series approach [99–104], the main methodology used was Box-Jenkins time series analysis.

In the decades of 1980 and 1990, ANN started to regain enthusiast in load forecasting [105–110], one reason being the use of backpropagation in ANN, as suggested by Werbos [111]. Back propagation is nowadays one of the most popular ANN learning algorithms for energy load forecasting [16].

One of the most renown articles from those decades [110], presented an improved ANN approach, in addition to a new method for selecting the training cases for the ANN. Their new method used the minimum distance measurement to identify the appropriate historical patterns of load and temperature readings. This was then used to estimate the network weights. Through their approach, they avoided the problem of holidays and sudden weather changes. This had a positive effect in the training of the network. The use of their methodology, improved accuracy over other methods when tested using two years of utility data.

Researchers developed a ANN for STLF used by a Greek power corporation in [112]. They used a fully connected three-layer feedforward network with back propagation. Their input variables were temperature, day of the week and historical hourly load data. The model they created could forecast load for up to seven days.

A Neural Network Based Electric Load Forecasting System (ANNSTLF), was proposed in [113]. This system is based on several ANN strategies that capture various trends in the data. The researchers used a multilayer perceptron trained with backpropagation. The model analyses the humidity and temperature effect on the load. This method had an improved approach in [114], where the researchers used two ANN instead of one. One



network predicted the changes in the load and the second predicted the base load. Their ANN considered the electricity price as one of the main characteristics of the system load.

Lastly, researchers created a multi layered feed forward neural network for STLF [115]. In the model, three different types of data were used as inputs to the ANN: weather related inputs, historical loads and season related inputs.

Many studies use ANN with other forecasting techniques, such as, time series [116], regression trees [117] or fuzzy logic [118].

With the evolution of machine learning, a new and simpler technique appeared, that gradually overtook ANN in terms of popularity, SVM [119], which were introduced by Vapnik [120]. Besides simplicity, another key factor for SVM popularity was its performance. SVM took first place in the EUNITE competition on load forecasting [121]. The researchers created a model for mid-term load forecasting that predicted the daily maximum load of the next 31 days. SVM became one of the main techniques used in load forecasting [122–127].

In [128], a method based on SVM for the electric power system STLF was presented. The SVM were trained with load and weather data to a STLF problem with a prediction time of 24 hours. The algorithm improved both training time and accuracy when compared to ANN.

The authors in [129], created a hybrid two-stage model for STLF. In the first stage, the daily load was forecasted using time-series methods. In the second stage, the deviation caused by time-series methods is forecasted using SVM. The results are then added to the result of the first stage. Their methodology made the forecasting procedure more adaptive and accurate.

Another variation of SVM is presented in [130], where the authors proposed a fixed size Least Squares Support Vector Machines (LSSVM) using a nonlinear model. Their study concluded that, in the case of STLF, using large time-series outperformed the LSSVM using a linear model.

Nowadays, Support Vector Regression (SVR) is the main technique used in the load forecasting community, a variation of SVM used to estimate the regression function [131]. In [126], researchers presented a SVR model to forecast the electric loads, using an immune algorithm to determine the parameters of the SVR model. They concluded that their model presented better results in forecasting performance than other methods, such as, the ANN model.

The researchers in [132] created a recurrent SVR model for LTLF based on Recurrent Neural Networks (RNN) [133]. They used genetic algorithms to determine the optimal parameters for their model. The conclusion was that their model outperformed other models, such as regression models or ANN. The same researchers proposed a hybrid model of SVR with simulated annealing to forecast long-term electric load [134]. Their study indicated that their model was superior to ARIMA models in terms of MAPE and MAD.

## 2.2.2 Areas of application

The applicability of energy load forecasting is quite diverse, energy companies, electricity pricing, wind power, structure planning, peak demand identification or datacenter energy consumptions are examples of areas in which energy load forecasting is involved. With the information from energy load forecasting, electricity pricing can be forecasted. By analysing consumption trends, it is possible to adapt electricity prices to upcoming variations.

Price forecasting importance to the energy industry is immense, since they generally cannot pass their costs on to the end consumers. The costs of under or over producing of power for an inadequate demand are extremely high. The results can be huge financial losses or even bankruptcy. Price forecasts from a few hours to a few months ahead are common practice in the energy industry. Forecasting loads and prices in electricity markets are mutually connected activities, and error in load forecasting will propagate to price forecasting. Companies who can forecast the volatile prices with a reasonable level of accuracy, with the support of load forecasting can adjust production or consumption schedule in order to reduce the risk and maximize profit [135].

In the energy industry, balance between electricity demand and consumption is indispensable in order to avoid supply or quality issues. However, the wind power generation, which is directly related to wind speed, availability of wind power is not constant. Wind power generation is subject to seasonal variations, daily cycles or temperature changes. Managing the inconsistency of wind power generation is a key aspect for its usability in the energy industry [136]. Energy load forecasting can help tackle that problem. Through load forecast, engineers can predict the consumption demand and as a result use wind power when available instead of traditional non-renewable resources.

Energy load forecasting has also its fair-share in the structure planning process. It aids companies in understanding better their future consumption [137]. For instance, during the planning process of a new supermarket, researchers can predict what the electricity consumption will be. Minimizing the risks for companies, by understanding the short-term or long-term electricity consumption helps them to plan and make economically viable decisions.

Load forecasting helps also in planning the future location, size and or type of power systems, by identifying areas or regions with high or growing demand [137]. It eases also the decision and planning for maintenance by knowing the future demand. Power plants can schedule better maintenance interventions and ensure that it has the minimum impact on consumers. An example of this is when energy companies decide to do the maintenance on residential areas during daytime, when most people are at work.

Power plants, responsible for producing power, can, through the forecast of future demand, quantify the required resources to operate. The identification of peaks or trends in the consumption helps energy companies maximize the utility of power plants. Forecasting avoids under or over power generation.

Another area of application for energy load forecasting is the real-time control of power systems. Short-term forecasts can support the on-line or offline tasks, like maintenance scheduling, security analysis, and on-line load flow solutions. It helps in coordinating forecasted load changes with the generation of power [138].

Energy companies also make use of energy load forecasting when making decisions on expansion and/or enhancement of their infrastructures, power generation augmentation and regional energy exchange. Frequently when sealing bilateral contracts between suppliers and consumer energy, load forecasting is necessary for successful negotiations [137]. Companies tend to also use the created forecast for promotional events/operations, for instance, cheaper prices on lower consumption periods.

Lastly, energy load forecasting can be applied on datacenters. Companies like Google, Amazon or Microsoft, who have enormous datacenters, use energy forecast to foresee their datacenter consumption and thus analyse options that guarantee the ecological and economic viability of their datacenters.

Nonetheless, with the many applicability's of energy load forecasting, come a similar number of challenges.

### **2.2.3 Energy load forecasting challenges**

One of the greatest challenges of energy load forecasting is the weather, mainly due to its unpredictability and inability to control. Forecasts can be based on expected weather conditions, however, the weather data used in the forecasts might differ from the real weather and thus the forecasted outcome might differ from the real weather. Sudden variations in the weather can affect the electricity demand. This can have a direct impact on revenues, especially if the power plant generates more than the demand.

Most experienced utility forecasters use manual methods that rely on a thorough understanding of a wide range of contributing factors, based on upcoming events or particular datasets. Relying on the manual forecasting is not sustainable due to the increasing number and complexity of forecasts. Utilities must therefore look for technologies that can accurately give results and eliminate problems that may occur if experienced forecasters retire or leave employment.

Different areas may also influence the electric consumption and as a result, individual forecast models, from these different areas, might be needed. Which in the end will be used for the final energy load forecast. This increases also the complexity in forecasts and increases the chances of errors.

Getting accurate and usable data for the forecast can be another challenge since the data might have missing values, wrong values, or outdated ones. Another problem with the data, can be wrong or outdated trends; these trends may vary depending on the seasons and the total consumption; for two similar seasons, it is very possible to have two different trends.

Creating a complete dataset is also difficult. Numerous complex factors that affect demand for electricity can be useful in one dataset, but inappropriate in another one. Even with one reliable dataset, the problem of too much dependency of said dataset might occur.

Ultimately, a wrong forecast might have devastating consequences and an acceptable margin of error for the forecast must always be analysed and readjusted when needed.

## 2.3 Server load prediction

In server load prediction, the most common goal is to determine the load that will be put on the servers by users, either in terms of data volume or number of connections. Efficient resource management is essential in the viability of the datacenter, ensuring that workloads have the necessary resources without breaching SLA. To efficiently adjust resource allocation, prediction algorithms are used to forecast incoming load and/or resources needed to guarantee the performance of the datacenter for every kind of specific load [139]. In addition, load forecasting lets datacenters minimize resource consumption, reducing energy consumption and, as a result, reducing carbon emission [140].

### 2.3.1 Related work

Literature review in server load prediction, contrary to energy load forecasting, does not present a chronological order. The review classifies articles by their main area of study.

The first area of analysis is automatic scaling, one of the most studied areas in server load prediction. By scaling the resources that an application has access to, depending on the workload it has and through a react approach, it is possible to reduce significantly the company's costs. However, one downside that comes with automatic scaling, is that, resources made available tend to lag after the demand, i.e., extra resources will get allocated after an overload situation already started. When downscaling, the reactive approach can release resources that will be needed in near future, e.g., the load goes over the upscaling threshold just after the release of resources. By using automatic scaling with load prediction these problems can be tackled.

In [141], researcher's created a hybrid cloud environment, with automatic scaling where traffic could change location when traffic spike occurred. With the use of load forecasting they redirect traffic from a heavy used server to a less used ones, where the future load was also predicted.

Automatic scaling is also very important when talking about green computing. A common approach to lower the systems consumption is the use of Dynamic Voltage and Frequency Scaling (DVFS), which adapts the required frequency and/or voltage of the CPU, according to its demands. In [142, 143], researchers used DVFS and ANN to predict the number of machines needed in the next 5 minutes. The researchers in [144], used DVFS and power actuation. They had a forecast horizon of 10 minutes where they also predicted the number of machines needed at that time.

In anomaly detection, another area of study, the objective is to use a prediction method with a small forecasting horizon and raise alert when the actual value is significantly different from the prediction. The researchers in [145], used self-organizing maps to detect when the system parameters were abnormal. Another method used by researchers was the PREPARE system [146]: when the system detected an anomaly, it increased the memory or CPU allocations of the Virtual Machines (VM) or migrated them.

Another area of study is the performance simulation, where the forecast is used to anticipate the performance a task would have in different systems. In [147], the researchers had a heterogeneous grid computing system and predicted the execution time of a task in

that environment. A benchmark from different machine types and workloads was analysed by k-means regression. The resulting model could estimate the run time and workload of a profiled task on different machines.

The last area of study considers the forecast method used. A vast number of studies present new forecasting methods and compare them with others. In [148], the researchers presented a summary of forecasting methods implemented in the R statistic language, along with their strong and weak points. They created an algorithm for automatic method selection based on prediction goals and, to a lesser degree, input data.

In [5], researchers worked on data from cloud servers of Amazon EC2. The solution estimated the server load average based on the website user's activities and assumed the number of transactions and activities in the server. They created different prediction models using different types of algorithms. Information was extracted from the collected website access and server uptime logs from a web application company's website, creating the datasets from this information. Predictions from this model had below average results in terms of accuracy, yet in terms of identification of trends the outcomes were relevant, despite some assumptions that did not reflect real life data.

The performance of different NN algorithms was evaluated to correctly predict the CPU load of a host in [149]. A multilayer NN, taking as input patterns collected from the load traces and predicting the future load statistics, was used. Subsequently, tests were carried out using different models with quick propagation, backpropagation, with and without momentum and the resilient propagation algorithms for the load traces. Resilient propagation algorithms had a better prediction accuracy compared to the remaining algorithms.

In [150], SVM were used to predict server load and a novel method of selecting free parameters was proposed to increase the prediction precision. Firstly, they reconstructed the phase space, where the embedded dimension was the key value to be decided. Secondly, they tested different parameter values and tried to achieve the best possible model. Lastly, they defined the prediction's steps with which they wanted to get the prediction values. They concluded that, by using SVM to predict server load in the future in addition with their novel approach to select free parameters for SVM, they could efficiently predict time series, including stationary and non-stationary.

### **2.3.2 Areas of application**

Server load prediction is a fairly new research area, when compared to energy load forecasting. New areas of applicability appear at a high rate. Examples of such areas are: cloud computing, grid computing, utility computing, virtualization, automated service provisioning or server load prediction.

The National Institute of Standards and Technology (NIST) defines cloud computing as a model that enables on-demand network access to a shared pool of configurable computing resources, such as, servers, networks, applications, services and storage, that can be rapidly supplied and/or released with minimal service provider interaction [151]. Cloud computing uses the predictions of load of different components to adjust its resources to an optimal state in a proactive approach. For example, the predictions of CPU load, enables the system

to adjust the number of cores to an optimal state, releasing or allocating further resources when needed.

Grid computing is another field that uses server load prediction, it coordinates network and computational resources to achieve a common computational objective. Grid computing can be often found in scientific applications, due to their usually computation-intensive tasks. The difference between cloud computing and grid computing relies in the absence of virtualization and in the goal [151]. In grid computing the resources might be allocated to just one goal, while on cloud computing, resources can be allocated taking in consideration different applications that run on the system. Load prediction lets researchers, analyse the resources needed to run, and adjust the resources in accordance to a minimal and optimal computational time, without compromising the whole system.

Server load prediction is also frequently found in utility computing. Utility computing provides resources on-demand and charges customers based on usage, instead of hardware. Server load prediction enables service providers to truly maximize resource utilization and minimize their operating costs.

Another field of applicability for server load prediction is virtualization. Virtualization is a technology that overlooks the physical hardware and provides virtualized resources for applications. A virtualized server, which is frequently referred as VM, provides the capability of merging computing resources from clusters of servers and dynamically allocate or re-allocate virtual resources to applications on-demand [151]. VM bring also the advantage of migration, which helps systems such as cloud computing to better balancing load across the datacenters. Through the use of load prediction, it is possible to test the optimal hardware configuration needed for a VM.

Autonomic computing aims to build computing systems capable of self-management. The system reacts to internal and external actions without human intervention. The goal of autonomic computing is to overcome the complexity of managing computer systems and avoid human interaction that may cause failures to the system. The IBM's mainframes are an example of autonomic computing: through the analyses of future load, the system independently readjusts the resources for upcoming load.

The process of load prediction in autonomic computing involves usually predicting the number of application instances required to handle demand at each particular level, periodically predict the future demand and determining resource requirements, and lastly automatically allocate resources using the predicted resource requirements [151].

Improving energy efficiency in datacenters is another major area in server load prediction. It has been estimated that the cost of powering and cooling accounts for 53% of the total operational expenses of datacenters [152]. Companies such as Google or Amazon are under enormous pressure to reduce energy consumption. The goal is not only to cut down energy cost in datacenters, but also to meet government regulations and environmental standards. Energy-aware job scheduling [153] is one way to do this, as well as reducing power consumption by turning off unused machines. For any of these solutions load prediction is indispensable since, by foreseeing future load, it is possible to adequate schedule energy-aware jobs or turn of machines with no load running on them.

Server load prediction became an indispensable tool for the commercial viability of products like Amazons web service, Google App Engine, Microsoft Windows Azure platform, or IBM's mainframe.

Amazon Web Services (AWS) [154] is a set of cloud services, providing cloud-based computation, storage and other functionality that enable organizations and individuals to deploy applications and services on an on-demand basis. One of AWS cloud services is Amazon Elastic Compute Cloud (Amazon EC2), which enables cloud users to launch and manage server instances through tools or APIs. EC2 instances are virtual machines running on top of the Xen virtualization engine [155]. Users have only control over the software part of the EC2 instances. Amazon is solely responsible for automatically scale the resources to the applications needs. Server load prediction supports Amazon's decisions, by trying to identify trends in the applications resource consumption.

Another use that server load prediction has in Amazon is the security and intrusion detection. This is done by predicting future load: if the real number differs highly from the predicted one, this might indicate a security flaw, triggering further investigations [156].

Google App Engine are Google-managed datacenters on which web applications run [157]. Applications can be developed using Django, Pylons, web2py or CherryPy, as well as a custom Google-written web application framework similar to JSP or ASP.NET. Google deploys the application to a cluster and monitors the application. Through load prediction, Google adjusts the resources needed for that application. In some occasions, if the application has problems, instead of allocating extra resources, Google shuts down the application [157].

Microsoft's Windows Azure platform has three different components, each one providing a specific set of services to cloud users [158]. Windows Azure, which runs applications and stores data on servers in datacenters, is one of that components. Users can create web applications using technologies such as ASP.NET that run on the platform. Through load prediction, the system automatically adjusts the resources to the necessary demand.

Lastly, IBM's mainframes make use of load prediction [159]. The operating system of IBM's mainframes, z/OS, has a dedicated component called Work Load Management (WLM), that predicts the future load on the system. WLM has two different objectives when it comes to load forecasting. Firstly, it verifies the business goals defined in the system and automatically assigns resources to the load based on their importance level and goal. Secondly it analyses load forecast to achieve an optimal use of resources from the system's perspective [160].

### **2.3.3 Server load prediction challenges**

Server load prediction evolved a lot since its earliest steps and several techniques have been developed as a support for it. Server load prediction, will certainly continue to evolve alongside society's needs, being society responsible to dictate the direction in which it will evolve.

A challenge that server load prediction faces, which is mainly due to its tender age in the scientific community, is the quantity of research done compare to other load forecasting areas such as energy load forecasting. Although server load prediction has been widely adopted by the industry, the research on server load prediction is still at an early stage. Many existing issues have not been fully addressed, while new challenges keep emerging from industry applications [161].

Numerous artificial intelligence and statistical techniques, created for short, medium and long-term load forecasting, operate on servers without a previous thorough planning. A thoughtful analysis on the created models must be done, in order to create a mathematical theory that explains the development and convergence of the used techniques [162].

Another field of research in server load prediction in need, is the categorization of relevant factors, *a priori*, that can identify a suitable algorithm for a specific type of load. Applicability and limitations of developed techniques must be tested, as well as, an investigation regarding the creation of a more generalized technique, that has a wider scope of application. Lastly, server load prediction and resource allocation systems must develop alongside, sharing contribution in each field [163].

## 2.4 Conclusion

In this chapter, background knowledge regarding load forecasting was presented, explaining in detail the different types of load forecasting, load forecasting methods, the performance of forecasts and crucial factors that must be taken in consideration when forecasting load. A literature review was conducted concerning server load prediction and energy load forecasting, each field was explained and related work in each field was presented and analysed. The applicability of server load prediction and energy load forecasting was also exhibited, along with their challenges. The obtained information is sufficient to give a good overview of the state of the art in load forecasting and aids in deciding which research topic to pursue.

In the following chapter, machine learning concepts and methods, some of which were already mentioned in the load forecasting methods, will be introduced and explained. The chapter builds on the knowledge acquired on this chapter.



# Chapter 3

## Learning approaches

In this chapter we introduce machine learning concepts along with necessary information for understanding the machine learning methodologies applied in this dissertation. We also describe the challenges and areas of application followed by the machine learning methods used in this work. Common software used in machine learning is introduced and explained, with focus on the Waikato Environment for Knowledge Analysis (WEKA) framework.

### 3.1 Machine learning overview

Researchers define machine learning as the ability of a computer program to develop and/or acquire new skills or knowledge from already existing information, with the goal of optimizing the performance criterion. The popularity of machine learning has significantly risen in recent years [164]. The fact that it may help to reduce costs and tedious and repetitive human tasks, are among the causes for its popularity. A definition that suits machine learning applied to load forecasting is given by [165].

The implementation of automation processes increased the already large amount of data produced every day. The analysis of these data is frequently a costly, slow and complex task. Machine learning techniques can tackle these problems, already proven in a vast area of applications, e.g., bio-surveillance [166–168], text and speech recognition [169–171], credit card fraud detection [172–174], medical diagnosis [175–177], computer vision [178–180], among many others.

The common approach when using machine learning is illustrated in Figure 3.1.

In the first step, data is selected and reviewed in an attempt to find quality issues and/or verify its usability for the final solution. If the available data does not present the necessary quality or quantity to proceed to further steps, the change to a completely new source of data might be a necessary action to be taken.

In step 2, data pre-processing, the data is transformed so that it can be easily used as input for the model. This step can be an iterative process when trying to create a satisfactory data input for the model. Although machine learning methods may work with raw input data, it is advisable and a good practice to prepare data prior to analysis. Data gathering methods are often created with little to no filters resulting in various data quality issues.

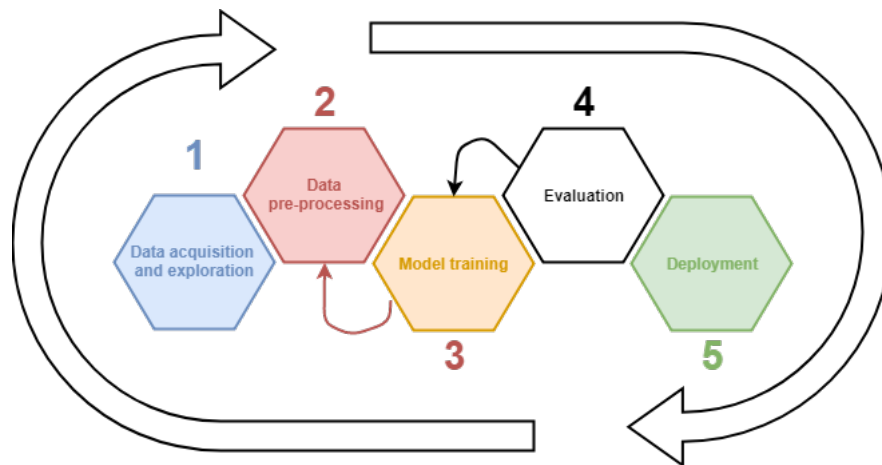


Figure 3.1: Machine learning process

For example, out of range value, e.g.,  $height = -1.71m$ , missing values, impossible data combinations, such as, *Gender: Male* and *Pregnant = Yes*, etc... If the analysis is not made correctly, the obtained results might be misleading. Along with incorrect values in data, redundant, irrelevant, unreadable and/or noisy data can increasingly difficult the learning process of the algorithm. Data pre-processing involve tasks such as data cleaning, normalization, feature selection and extraction, normalization and/or transformation. These tasks are non-trivial and hence, careful considerations must be made, in order to succeed in modelling. The product of the data pre-processing step is the final dataset. Data pre-processing methods can be divided into the following categories:

- **Data cleaning:** The collected data is often incomplete or does not meet the quality standards of the used machine learning algorithm. Attributes might miss values or even be completely left out of the data. Inconsistency in data might be another problem, which along with noise, is commonly found in data prior to cleaning. Data cleaning routines try to identify and solve these problems by filling missing values, smooth the noise in data, identify and resolve inconsistencies and remove outliers when needed.

Dirty data can cause problems to machine learning algorithms. Although most algorithms possess routines to deal with these problems, they tend to be very simple and limited when used. Therefore, most researchers use their own created routines based on frequently occurring problems. Common approaches are missing values routines, which are triggered when many records appear with no values in them. The routine can opt to ignore the records; this is done more frequently in classification problems that miss their label. They can also fill in the missing values, with average value for instance, or they can simply use a global constant that represents a missing value, e.g., the “*N.a*” value.

In case the data presents a high level of noise, frequent techniques used are clustering, which helps identifying outliers. Binning methods, which use the neighbourhood approach to smooth the data that stands out among its neighbours. Regression methods can also prove useful when smoothing noise out of data: by finding an equation that best fits the data, noise is frequently automatically smoothed out.

Lastly, human interaction is a valuable asset for smoothing. Trained staff can frequently identify and verify if some value is actually noise or not. Data that presents inconsistencies can sometimes be manually corrected through the use of external references, e.g., data with a partial inconsistency can be verified using a simple piece of paper. In these situations, the revision of code in routines is essential to rectify the errors. Functional dependencies between attributes can often be used to identify values that contradict the functional constraints.

- **Data integration:** Data integration involves combining data from different sources in a single dataset. These sources may include multiple datasets, different databases or single files, as seen in Figure 3.2 .

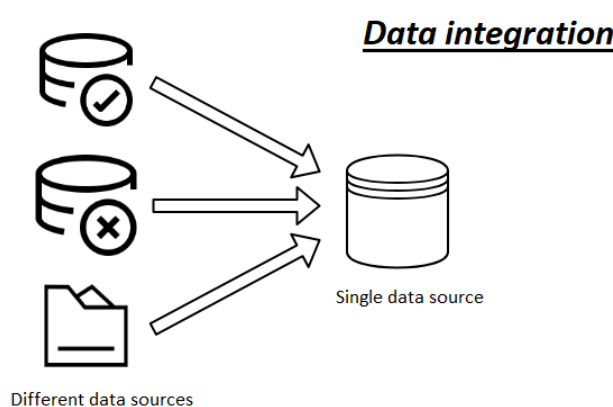


Figure 3.2: Data integration example

However, the process can involve a certain number of issues that must be taken into consideration. Identity identification is one such issue. For instance, the id field related to a customer in one database might not refer to the same in another database, dataset or file. Another issue that might occur is redundancy in data. This typically occurs because an attribute had different names in different sources, resulting in the import of two differently named attributes that represent the same data. The addition of metadata is a common solution for this problem, since it characterizes the data and removes ambiguity.

- **Data transformation:** Data is consolidated and transformed in the data transformation process. Normalization is one approach used. In normalization the attributes are scaled in order to fall within a certain range, for example, values between -100 and 100 can be scaled to range between -1 and 1, as seen in Figure 3.3.

Smoothing is another approach used in data transformation. It works in the same way as described in the data cleaning process. Generalization of data is also frequently used when transforming the data. For instance, in categorical attributes a street name can be rewritten to a city or country to better suit the overall syntax of the dataset. The same can be done with numerical values, where instead of the age as a number, the age is represented as a category such as, child, adult or senior. Lastly, aggregation can be used to transform the data. Aggregations or summary operations are used, to better suit the overall data. For example, the daily sales can be aggregated in order to create an attribute of monthly sales.

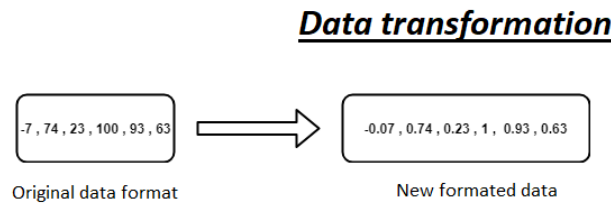


Figure 3.3: Data transformation using normalization

- Data reduction:** Large datasets can be arduous tasks to analyse. In some cases, a large dataset can prove to be infeasible or impractical to analyse. In such situations data reduction is used. Data reduction techniques reduce the size of the dataset without compromising the integrity of the data. It often involves reducing the number of records or the number of attributes in the data. Figure 3.4 depicts in a simple way such process.

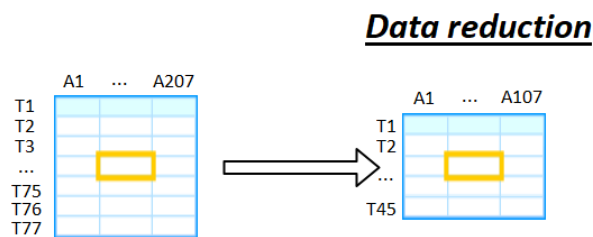


Figure 3.4: Data reduction exemplification

Frequent methods used in data reduction are dimension reduction, data compression, discretization and concept hierarchy generation or numerosity reduction. Dimension reduction involves removing redundant, weakly related or irrelevant attributes; data compression uses encoding mechanisms to reduce the size of the dataset, much like software such as *WinRAR* use. Numerosity reduction, replaces a large set of data with a representation of it, for instance, parametric models which save only the model parameters instead of the complete data.

Subsequently, to the data pre-processing step comes the model training step. This step involves selecting the machine learning algorithm/learning method better suited to the problem. After selecting the learning method, the modelling assumptions must be verified. This includes, for instance, confirming if missing values are allowed or if all attributes must have uniform distributions. Before creating a model, the learning process should be defined, describing in detail: training, testing and evaluating. The learning process should also have defined how to split the dataset into training, test and/or validation data sets. After defining the learning process, the model is created. The creation of a model encompasses the definition and selection of the algorithm parameters. If the created model is not satisfactory, step 2 can be re-done in order to create a better suited input for the model.

As for the evaluation step, the results from the created model are compared to the defined goals. At the end of this step, it is important to summarize the results in terms of business success criteria, stating if the model is capable to meet the company's objectives. When a model meets the success criteria it becomes an approved model. After having one, the reviewing of the evaluation process starts, reconsidering all previous steps and trying to identify important tasks or factors that may have been overlooked. The last task in the evaluation step is to decide if the model is deployed or if a new iteration is started.

The last step in the machine learning process is the deployment of the model. In this step a deployment plan is created, which summarizes the deployment strategy, including the necessary steps and how to perform them. After deploying a model, monitoring and maintenance of the model are essential to avoid any serious complications.

## 3.2 Supervised learning vs. Unsupervised learning

In supervised learning the objective is to automatically infer a model (hypothesis) from a set of labelled examples that is capable to make forecasts given new unlabelled data. A label  $y$  is the desired forecast of an instance  $x$ . Labels can be a set of finite values, such as, male and female, which are commonly called classes. These classes can sometimes be encoded by integer numbers, to better suit the used learning method, e.g., male = 1 and female = -1. This type of encoding is frequently used in binary problems, being one class called the positive class and the other the negative class. In general, an encoding does not imply a structure in the classes, i.e.,  $y = 1$  and  $y = 2$  are not necessarily closer than  $y = 1$  and  $y = 7$ . In supervised learning, the data consists in a set of pairs, containing an instance  $x$  and a label  $y$  and is denominated as labelled data [181].

Supervised learning includes regression problems, where the label is a continuous value, for example, the electric consumption in a city or the blood pressure level of a person based on the height and/or weight of that person [181].

In contrast, in unsupervised learning, the algorithm has no access to the labels of the data. The goal in unsupervised learning is to model the distribution or structure in the data in order to acquire useful information to better understand the supplied data. It tries to identify inherent similarities between the data and separate them into groups accordingly, assigning its own new label to each group.

One of the most common unsupervised learning tasks is clustering, where the goal is to identify potentially useful clusters/groups in the data. For instance, a server might gradually develop a concept of "huge traffic days" and "low traffic days".

To conclude, supervised learning is the most commonly learning task used. Supervised learning uses learning methods, such as, linear and logistic regression, multi-class classification, and SVM. Supervised learning requires that the output is already known and that the data used to train the learning method is correctly labelled.

On the other hand, unsupervised learning is more closely aligned with the task of identifying complex patterns and/or processes. Some examples of unsupervised learning algorithms are k-means clustering and association rules [182]. For instance, in load forecasting an example of unsupervised learning is the creation of clusters that categorize the data patterns into different categories. Taking our two case studies as examples, the created clusters could divide the data into workdays load and weekend load.

### 3.3 Classification vs Regression problems

The classification problem definition can be delimited to a two-class problem without the loss of generality. The goal in a classification problem is to separate the two classes through a function calculated from the available examples. The created function, commonly referred to as classifier, will try to correctly divide (classify) the data on unseen examples [182]. Figure 3.5 depicts such classification attempt.

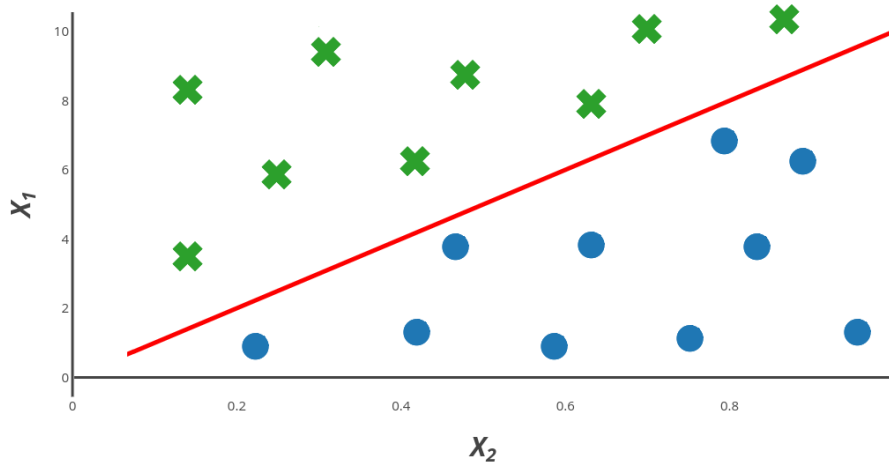


Figure 3.5: Classification example on linearly separable data

In the presented example, it is possible to use various linear classifiers to separate the data, however, there is only one that maximizes the margin, which is the distance between the linear classifier and the nearest data point of each class. This linear classifier is denominated as optimal separating hyperplane.

The classification in the example given has linearly separable data, i.e., it is possible to completely divide the data into two separate groups with just one separating plane. In general, this will not be the case. Figure 3.6 depicts a simple, yet more accurate classification problem.

In this example it is impossible to find a linearly separating hyperplane that successfully divides the data into two separate groups. In these situations, the introduction of a cost function associated with misclassification is the suitable approach. This approach helps the algorithm to define the width of the hyperplanes margin, since it sets a cost value for each point that is wrongly classified within the hyperplane. Learning methods, such as SVM use this type of approach. The final hyperplane divides the data into two separate groups, with some elements inside the hyperplane, and in some cases when appropriate with some elements on the wrong side of the classification. Common learning methods used in classification problems are decision trees, naive Bayes, linear Support Vector Classifier (SVC), K Nearest Neighbours, logistic regression, ANN, etc. [182].

In regression problems, the objective is to predict a continuous value, e.g., predicting the hourly load on a server for a given hour. There are several types of regression techniques.

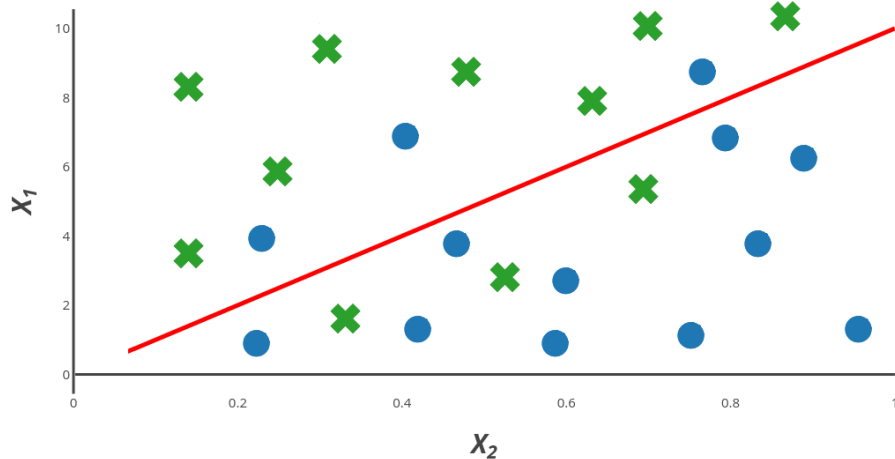


Figure 3.6: Classification with non-linearly separable data

The oldest and simplest technique of regression is linear regression. Linear regression estimates the relationship between two variables. The mathematical formula used is  $y = mx + b$  and creates a straight line, as seen in Figure 3.7. This means that in a graph with a  $X$  and  $Y$ -axis, the relationship between  $X$  and  $Y$  is a straight line with some outliers. For instance, the increase in population should have a direct effect in food consumption and might increase at the same rate. If this is the case, the relationship is referred as a strong linear relationship between the two variables.

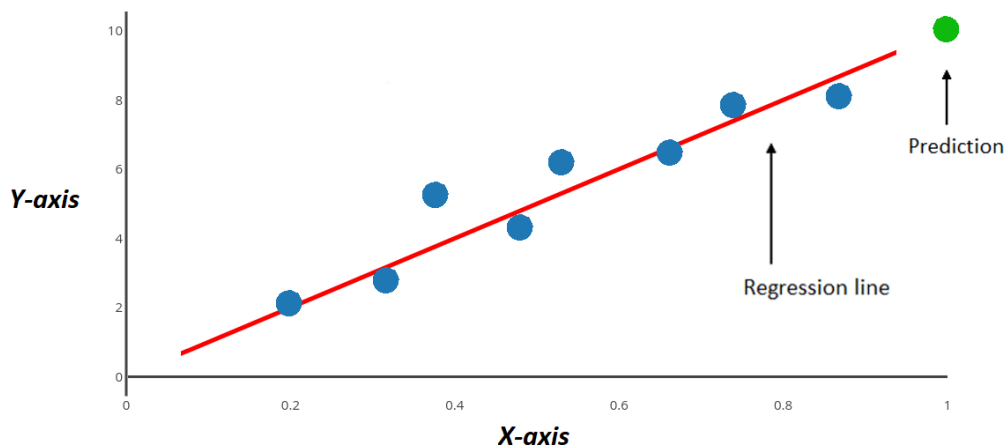


Figure 3.7: Regression problem example

More advanced techniques, such as multiple regression, forecast the relationship between multiple variables, e.g., the correlation between level of education, yearly income and residence. However, the addition of variables increases significantly the complexity of the forecast. There are several techniques used in multiple regression, such as standard, hierarchical, or stepwise regression [182].

**Standard multiple regression:** In standard multiple regression all variables are considered at the same time when forecasting. For example, the relationship between the

education and income are taken in consideration at the same time when forecasting the residence. Standard multiple regression analyses the degree of influence of each individual variable on the forecast.

**Hierarchical regression:** The hierarchical regression is a sequential process where the variables used for forecast are used as input for the model in a pre-specified order defined in advance. The algorithm does not determine the order in which the variables are used for forecast.

**Stepwise multiple regression:** An algorithm that uses stepwise multiple regression analyses which variables are better suited for the forecast. The stepwise model evaluates the order of importance each variable has and selects the most relevant subset.

**Stepwise regression:** Stepwise regression, which is different from stepwise multiple regression, does not analyse a single variable, it analyses sets of variables. The final solution results from the addition of all the relevant subset. This can result in situations where the final solution excludes completely a subset of variables.

When transcribing the types of regression into the load forecasting problem, there is no clear type that transcends the others. It is more a trial and error process that tries to find the best suitable regression type to the specific load problem and data.

### 3.4 Challenges and applicability

Machine learning problems have usually large datasets associated to them. The required amount of computer processing capacity and memory for these large datasets tends to be huge, with processors needing to have multiple cores with high speeds. Researchers continuously try to improve the efficiency in their machine learning approaches, since improving their existing hardware is usually a costly option. These hardware and code limitations are critical challenges researchers face; often it is simply impossible to wait for a solution for days [182].

Another challenge in machine learning is obtaining the first useful result/solution. Before the refining/tuning process starts, a first useful result/solution must be obtained, refining/tuning a result that does not satisfy the initial goals of the problem, can result in valuable loss of resources and time towards the deadline. However, even with a useful result which is already refined/tuned, another challenge occurs: it is utmost necessary to verify if the created result has not been over-refined/tuned, i.e., the current result is too fragile and will probably no work outside a specific dataset.

In some occasions the challenge might lie even before achieving and refining/tuning a result. Researchers might have defined their goals wrongly and keep refining/tuning learning methods that are not suited to answer correctly their goals. In order to use the hardware and machine learning approach efficiently, researchers need sometimes to step back and review their goals/objectives.

Researchers might also rely too much on their intuition. The first step of a machine learning problem is usually the creation of a hypothesis that will solve a problem. With regard to achieve this, researchers use their intuition to create a starting point, which will



aid in discovering a solution for the problem. However, the intuition might be sometimes wrong, and researchers may have to rethink/redefine their initial assumptions.

Lastly, machine learning can be found in many applications. Since the symbiosis between applications and machine learning often functions very well, the end user tends to completely ignore its existence. Common areas of applications are automation, fraud detection, resource management, complex analysis, customer service, machine efficiency, animal protection, predicting waiting times and access control [183].

### 3.5 Learning methods

Algorithms in machine learning play a vital part; almost every action taken in machine learning has a connection with the algorithms/learning methods. These algorithms are formulas or procedures that are used to solve a problem. The domain of the problem will directly affect the choice of algorithm used. The basic premise is however always the same: to solve the defined problem.

Algorithms can be compared to a box that stores a method that solves a specific problem. They process the data and try to create an output that solves said problem. The created output/solution tends to be specific to the type of problem being solved.

Frequently used techniques in machine learning are [183]:

- **Bayesian inference:** Bayesian inference basis its forecasts in various statistical methods. Since statistical methods can create more than one seemingly correct solution, the choice of a function that has the highest probability of succeeding becomes the core objective. For instance, a set of symptoms can be used as input and the output can be the probability of a disease that has these symptoms. Assuming that multiple diseases share a large set of symptoms, the probability of each disease will be different. However, the highest probability might not be the correct disease; it is important to present the different outputs with the corresponding probability so that in the end, a lower probability disease, which may be the correct one, is not discarded.

Ultimately when using Bayesian inference, the main idea is to never completely trust a hypothesis without analysing the steps used to create the said hypothesis. One of the most recognizable solutions that resulted from Bayesian inference is the spam filter [184].

- **Nature inspired meta-heuristics:** Evolutionary algorithms rely, as the name suggests, on the principles of evolution to solve problems. They often use methodologies such as survival of the fittest, which removes solutions that do not achieve the desired output. Another nature inspired algorithm commonly used, is the artificial bee colony algorithm [185], which is inspired on the bee foraging behaviour. It focuses on numerical optimization, extending combinatorial, constrained and multi-objective optimization problems. Ant colony, which is another example, is inspired on the ant communication behaviour through pheromones when creating paths [186]. Ant colony focuses in graph and combinatorial problems. Lastly, particle swarms also basis its approach on animal flocking behaviour. Particle swarms are mainly used in numerical optimization problems [187].

- **Learning by analogy systems:** Learning by analogy systems use kernel machines to identify and recognize patterns in data. The analogy system identifies a pattern in the input data and compares them with already known patterns of outputs to create a solution for a problem. The objective in these systems is to use the patterns/similarities in data to determine the best suited solution for a problem. The main thought is that since a particular solution already proved to be efficient on a given problem, using the same or similar solution might also prove to be efficient on similar problems [188]. The recommend function in Amazon for example, resulted from the application of such systems.
- **Time series analysis:** A time series has a series of data entries which are arranged in a chronological order. Time series analysis, studies the data with the intention of extracting important characteristics or statistics in the data [189]. A common method used in time-series analysis is Box-Jenkins.
- **Regression analysis:** In regression analysis a set of statistical procedures are used to estimate relationships between variables. Regression analysis helps identifying dependence in variables and uses these dependencies for forecasts [190].
- **Artificial neural networks:** ANN are a network of neurons, where each neuron solves a small part of the whole problem, the final solution is created by using the whole network. Through the use of backward propagation or backpropagation of errors, ANN try to determine the conditions under which errors are removed from the network. The ANN may be trained using the backpropagation algorithm until the actual output matches the desired output. The output created by a neuron is only part of the whole solution, neurons pass their output to the next neurons until a group of neurons creates a final output [191].
- **Support vector machines:** SVM are supervised learning models, that use associative learning algorithms to analyse data used as input. SVM can be used for regression or classification problems. Initially, the training set is divided into two categories, SVM try to assign each new record to one of the two categories. In case the input data is non-linear divisible, SVM use the kernel trick two transform the problem in a linear divisible problem and perform its usual task [192].

The focus in this dissertation will be on the learning methods linear regression, ANN and SVM.

### 3.5.1 Linear Regression

Linear regression uses the relationship between an independent variable or variables, the input, and the dependent variable, the output, to build its model. Its applicability is mainly in regression problems, however, sometimes it can be used in classification problems to identify the correlation between attributes. Linear regression estimates the coefficients for a hyperplane or line that best fits the training data, as seen in Figure 3.8. It is a very simple regression algorithm when compared to ANN or SVM. Due to its simplicity it is fast to train. Nonetheless, its simplicity does not imply bad performance. On the contrary,

linear regression has great performance in situations where the output variable is a linear combination of the inputs [193].

Linear regression can be classified as simple linear regression or multi linear regression. In simple linear regression only one independent variable exists, whereas in multi linear regression, as the name suggests, two or more independent variables exist.

Simple linear regression studies and summarizes the relationships between two continuous variables, as mentioned above. The independent variable is sometimes referred as explanatory or predictor amongst researchers and is commonly denoted as  $x$ . The dependent variable, is amongst other names, referred as outcome or response and is commonly denoted as  $y$ .

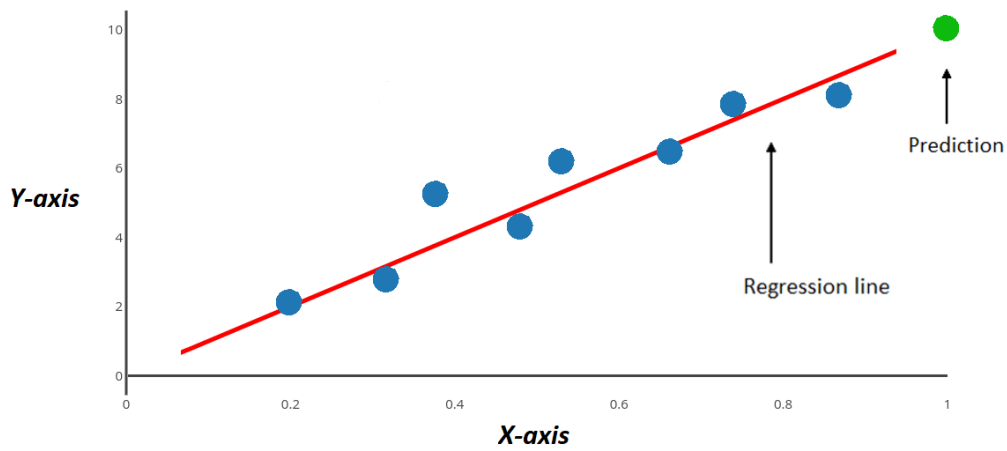


Figure 3.8: Linear regression example

In order to find which line better fits the training data the following equation is used:

$$\hat{y}_i = \alpha + \beta x_i, \quad (3.1)$$

where  $\hat{y}_i$  is the predicted response or fitted value for experimental unit  $i$ ,  $x_i$  is the predictor value for experimental unit  $i$ . When using Equation 3.1 to calculate  $\hat{y}_i$ , the forecast value usually has an error, commonly denoted as prediction error or residual error, associated to it, which can be defined as:

$$\xi_i = y_i - \hat{y}_i \quad (3.2)$$

From these two equations it is possible to conclude that the line that best fits the training data will be one for which the  $n$  prediction errors, one per observed data point, has the smallest overall value. This goal can be achieved through the use of the least squares criterion, which minimizes the sum of the squared prediction and can be calculated using:

$$Q = \sum_{i=1}^n (y - \hat{y})^2 \quad (3.3)$$

The squaring of the prediction error is done in order to avoid that the positive and negative prediction errors cancel each other out when summed and therefore yield 0. The

problem that arises when using Equation 3.3 is that it just gives one value per calculated regression line, i.e., in order to have the best possible result, the implementation of Equation 3.3 must be done in an infinite number of possible lines. This problem can be tackled by using Equation 3.1 in conjunction with Equation 3.3, which leads to the following equation:

$$Q = \sum_{i=1}^n (y_i - (\alpha + \beta x_i))^2 \quad (3.4)$$

However, in order to achieve this, the missing values  $\alpha$  and  $\beta$  of Equation 3.1 must be calculated. This can be done through the equations:

$$\begin{aligned} \beta &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \\ \alpha &= \bar{y} - \beta \bar{x}, \end{aligned} \quad (3.5)$$

where  $\bar{x}$  and  $\bar{y}$  are the average values of the observation in  $x$  and  $y$ , respectively. Since  $\beta$  and  $\alpha$  are derived using the least squares criterion, the resulting Equation 3.1 is often referred to as least squares line or least squares regression line.

### 3.5.2 Artificial Neural Networks

ANN, as a proposal applied to short-term prediction, have had a considerable development since the eighties. However, its application is not exempt from some constraints and repairs, either due to intrinsic limitations of the methodology itself or to constructive difficulties of the models [113]. The overwhelming majority of ANN-based models for short-term forecasting are used in the electrical energy production sector [17].

According to researchers in [194], ANN were developed based on the principle of "operational likelihood" of the cerebral NN. An ANN thus consists of an interconnection of several Processing Units (PU), with a configuration similar to that of the brain neuron. Due to this similarity of functioning, the name of neuron is also given to PU. Figure 3.9 depicts an artificial neuron and a biological one.

The entries and exits of the PU correspond, similarly, to the synapses and axons of the brain neurons. Each new input presented is weighted by a synaptic weight, which is indicative of the strength of the bond. Connection strengths are adjusted by the learning algorithms, in an iterative learning process resulting from experiments. Each PU performs very simple operations, starting with the weighted sum of the inputs by the weights of the respective connections, consequently resulting in the activation value of the PU.

The activation function(s) inserted in the PU are indispensable in the intermediary layers. The introduction of such functions is one of the reasons that makes ANN extremely popular amongst researchers [195]. An activation function from a neuron transfers values coming from the input layers. The information regarding an activation function becomes the input of the upcoming activation function and whose output is the value of the PU. Usually, the activation function is non-linear and differentiable everywhere and it is the same for the whole ANN. The values transported are often between 0 and 1 or -1 and 1.

PU whose source is outside the ANN are categorized as input units. Computing units whose function it is to generate the activation values, used for generation and interpretation

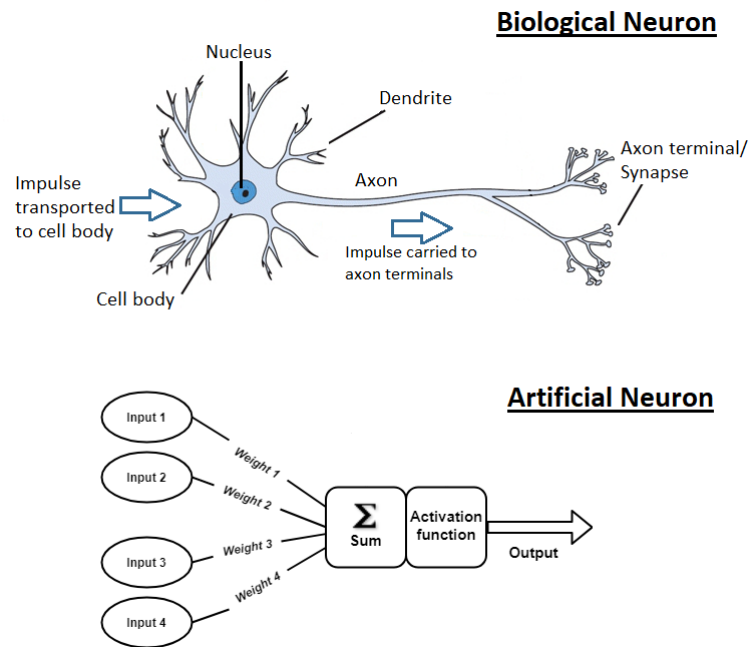


Figure 3.9: Biological neuron vs Artificial neuron

of actions in other system are categorized as output units. Computing units which have neither function are categorized as hidden units.

Although several types of ANN exist, the most commonly used are multilayer feedforward networks [196], which is the one used in this work.

Multilayer feedforward networks possess one or more hidden layers between their input layers and output layers. The most common constellation in research is a multilayer feedforward network with only one hidden layer. Hidden layers are constituted by hidden neurons sometimes also referred as hidden units.

A common way of describing a ANN topology is through a sequence of numbers, e.g., 3-4-3; this network would consist in 3 input neurons, 4 hidden neurons which would help in the calculation of the final solution, and 3 output neurons, Figure 3.10, shows the implementation of a multilayer feedforward network with a 2-3-2 topology.

Usually, fully connected networks are used. In these networks, units from one layer are all connected to all units of the next layer. In load forecasting the most common approach is to multilayer feedforward networks trained with the backpropagation algorithm, which is illustrated in Figure 3.11 [16].

Backpropagation can be divided into two distinct phases, a forward and a backward passing phase. In the forward phase, the weights are not altered and only the sum and the activation values are computed within the ANN. After an output is created, an associated error is calculated, and retro propagated successively to the previous layers. As seen in the figure, each neuron calculates an error with the retro propagated information. After calculating a new error, the respective weights that connect the next layer are updated. This process is repeated for all layers until reaching the input layer. Forward and backward

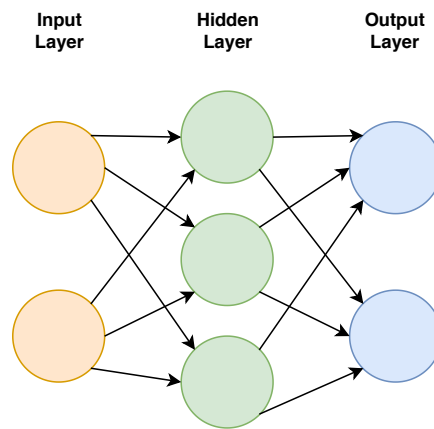


Figure 3.10: Multilayer feedforward implementation of 2-3-2 topology

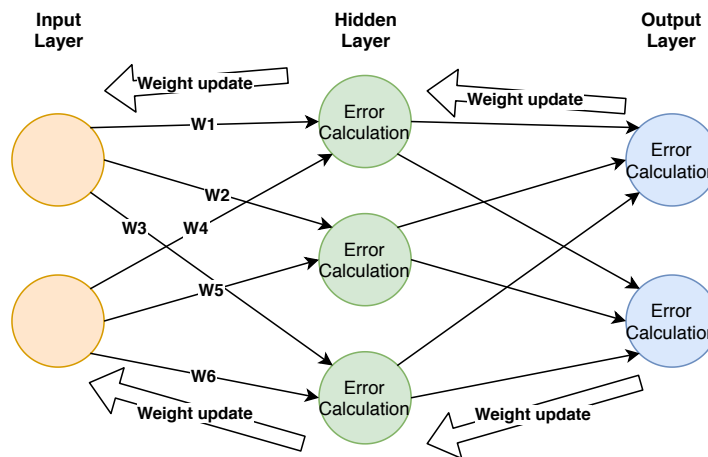


Figure 3.11: Backpropagation example

phases operate on the same data points and only change to the next data points after both phases are completed.

In backpropagation the activation functions should be differentiable. The most common are: linear, hyperbolic tangent and sigmoid, as seen in Figure 3.12 [197]. Usually, the sigmoid functions are preferable, since a small change in the weights may produce a change in the output, making it possible to analyse its evolution. The transfer function for the output layer is usually chosen as a function of the data. For continuous values, for which the value limits are not known, linear functions are commonly used [26].

ANN can also be trained focusing on supervised and unsupervised learning. In the first case, for each of the input vectors the respective outputs are known, the error value is calculated through the difference between the output value of ANN and the known value. The updating of the weights of the connections is done through the learning algorithm. With the presentation of the new input vectors, a new error value is calculated, ending this learning process when the error value becomes acceptable [26].

In unsupervised learning the weights of the connections are modified only in response to the inputs and the outputs are unknown. The Hebb rule [198], for example, and which was one of the first rules to be proposed, states that the weight of the connection between

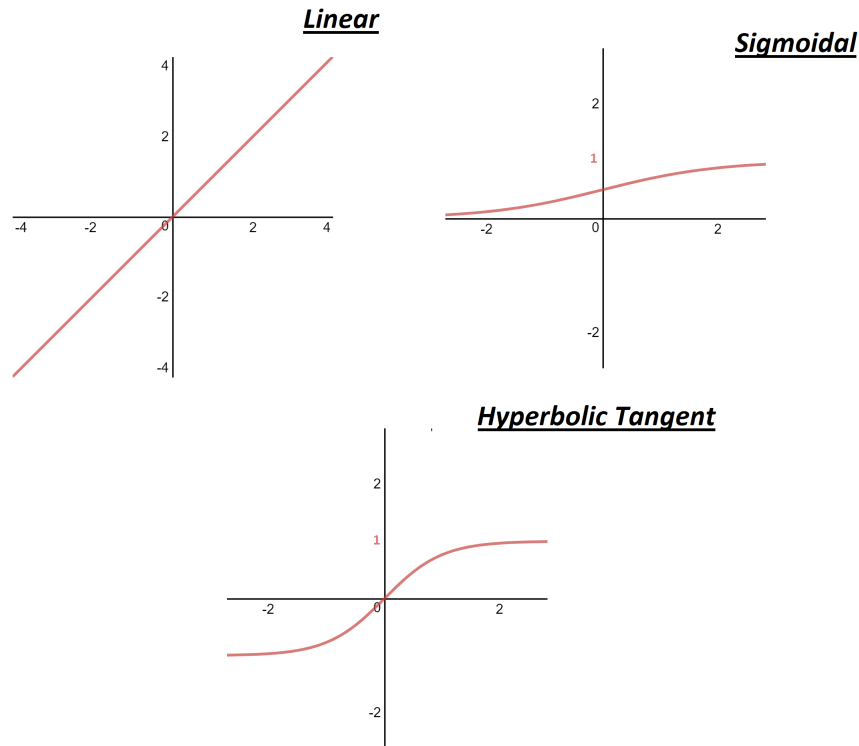


Figure 3.12: Different types of transfer functions

the neurons  $X_1$  and  $X_2$  should increase whenever an increase in the output of  $X_2$  results in an increase in the output of  $X_1$ .

Backpropagation is responsible for the wide application of ANN. About 80% of the networks use it [199] and it is predominantly used in STLF [26].

### 3.5.3 Support vector machines

SVM are supervised learning models that use learning algorithms for solving classification and regression problems. They were firstly introduced by Vapnik and its co-workers in 1990 [200]. SVM are learning algorithms focused on two class discriminant functions. Their application can be found in handwriting recognition, gene expression, data analysis, text categorization or load forecasting, amongst other. Researchers have developed different methods for SVM in order to increase their applicability in different areas, such as, factor analysis, clustering or regression.

Since SVM is a supervised learning model, its input data is constituted by a set of input attributes  $x_1, x_2, \dots, x_n$ , and a desired output value  $y$ . The output of the SVM are a set of weights  $w$ , one per feature, whose linear combination predicts  $y$ .

SVM use a combination of convex optimization [201] and statistical learning [27] to find a suitable plane/margin in a data space that can separate the data into two different classes.

- **Convex optimization:** In convex optimization the objective is to use convex functions to eliminate local minimums, a common problem in optimization problems. Since the function has a convexity, the local minimum is always a global minimum, as seen in Figure 3.13 [202].

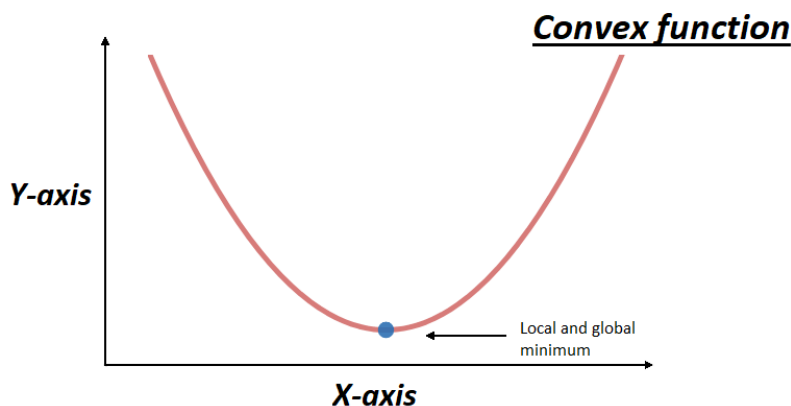


Figure 3.13: Convex function with a local/global minimum

- **Statistical learning:** In statistical learning the goal is to define a particular mathematical formation for the generic concept of learning. The starting point in statistical learning is to use a finite set of input values with known output values to learn the assumed but unknown function that represent the relationship between input and output values. The objective of the learning process is to find a prediction rule for the output that can efficiently predict new instances of data.

Statistical learning assumes that all input values  $x$  are contained in a known set  $X$ , describing the range and format of the input value and all output values  $y$  are contained in a known set  $Y$  that describe the range and format of the output values. The collected data sequence can be described as  $D = ((x_1, y_1), \dots, (x_n, y_n))$  and the learn function can be denoted as  $f : X \rightarrow Y$ .  $f(x)$  will try to find the best output value  $y$  for an arbitrary  $x$ . In statistical learning this is guaranteed by assuming that all the pairs of data  $(x, y)$  in  $D$  are independently generated by the same yet unknown probability distribution  $P$  on  $X \times Y$ . This is a two steps process: firstly, the input values  $x$  are marginally distributed by  $Px$ ; secondly, the output values  $y$  are generated by the conditional probability  $P(y/x)$  on  $Y$  given  $x$ . In order to test the quality of an estimated output  $f(x)$ , the common approach is the use of a loss function, commonly referred to as risk, which is given by the equation:

$$Risk(f) = \int_{X \times Y} L(x, y, f(x)) dP(x, y), \quad (3.6)$$

where  $L(x, y, f(x))$  is the loss function, which measures the quality of the learning function  $f$  for a particular choice of  $(x, y)$ .

SVM represent their data as points in a high or infinite dimensional space. A hyperplane is created in order to correctly divided the data into different categories. SVM can then



predict to which category new data belongs. A hyperplane should have the largest margin possible, which is the minimal distance between the hyperplane separating the two classes and the closest data points to the hyperplane. These closest data points are commonly referred to as Support Vectors (SE). Figure 3.14 depicts a hyperplane with its SE and respective margin.

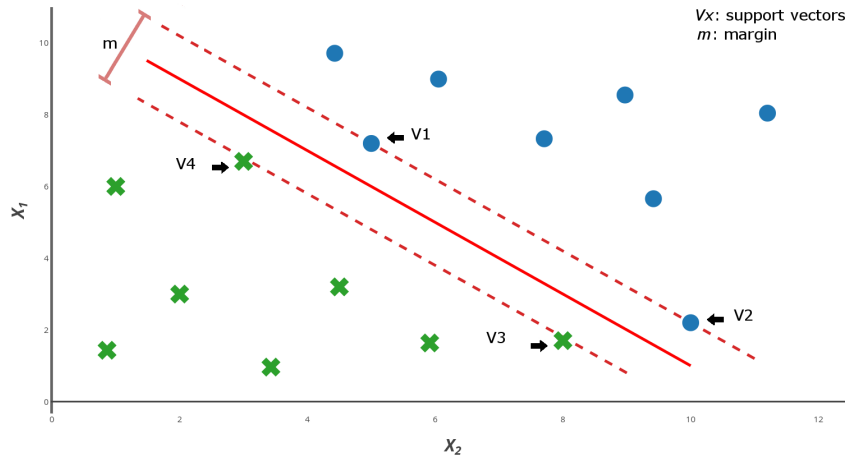


Figure 3.14: Hyperplane with respective SE

The SE are crucial elements and must be taken extra care when interacting with the training set since if deleted, the position of the dividing hyperplane would be changed. Figure 3.15 depicts a hyperplane with the initial SE and the new hyperplane after the SE were changed. If the changed datapoints are not SE, the hyperplane will not suffer any change, as seen in Figure 3.16.

The problem to find the optimal hyperplane is often tackled by optimization techniques. One of the most common used is the Lagrange multiplier. Lagrange multipliers, sometimes referred to as dual variables, are scalar variables that calculate function extremes in mathematical optimization with constraints [203]. The optimization algorithm for the hyperplane generates the weights of the output so that only the SE are taken into consideration when calculating the weights.

SVM were initially developed to solve classification problems, however, nowadays its applicability has been extended to the domain of regression problems. This led to an update in the literature terminology for SVM, since initially SVM referred to classifications with support vector methods and also regressions with support vector methods. At the present time, the term SVM refers more to the generic concept of the learning method without the definition of classification or regression. The terms Support Vector Classification (SVC) and Support Vector Regression (SVR) are the current nomenclature used by researchers to denominate SVM for classification and regression problems, respectively.

The simplest model in SVC, is called maximal margin classifier or hard margin and is more used as a theoretical example then in real-world use case since it requires the data to be linearly separable. The initial dataset in constituted by linearly separable data  $S = (x_1, y_1), \dots, (x_n, y_n)$ , where  $X \subset \mathbb{R}$  and denotes the input space;  $Y = \{-1, +1\}$

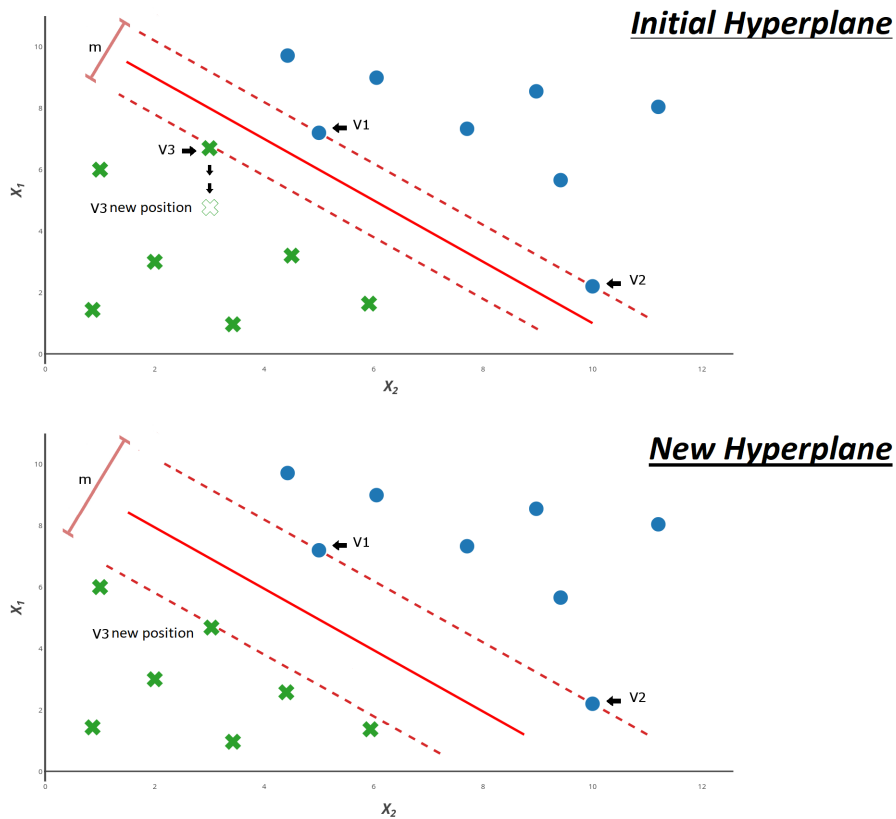


Figure 3.15: Hyperplanes with initial and new support vectors

represents the output domain for binary classification. The equation used when defining a hyperplane in SVC is:

$$H_0 : w \cdot x + b = 0, \quad (3.7)$$

where  $w$  is the weight of the vector,  $x$  is the input vector and  $b$  is the bias. Equations 3.8 and 3.9 refer to the lines that limit the hyperplane, being  $-1, 1$  the binary classification values.

$$H_1 : w \cdot x + b = -1 \quad (3.8)$$

$$H_2 : w \cdot x + b = 1. \quad (3.9)$$

When trying to maximize a margin, the objective is to increase the distance between  $H_1$  and  $H_2$ , therefore the distance between  $H_0$  and  $H_2$  can be written as:

$$|w \cdot x + b| / \|w\| = 1 / \|w\|, \quad (3.10)$$

Which makes the calculation of the whole distance, i.e.,  $H_1$  and  $H_2$ :

$$2 / \|w\| \quad (3.11)$$

From these equations, it is possible to conclude that, in order to maximize the margin,  $\|w\|$  must be minimized, with the condition that there are no datapoints between  $H_1$  and

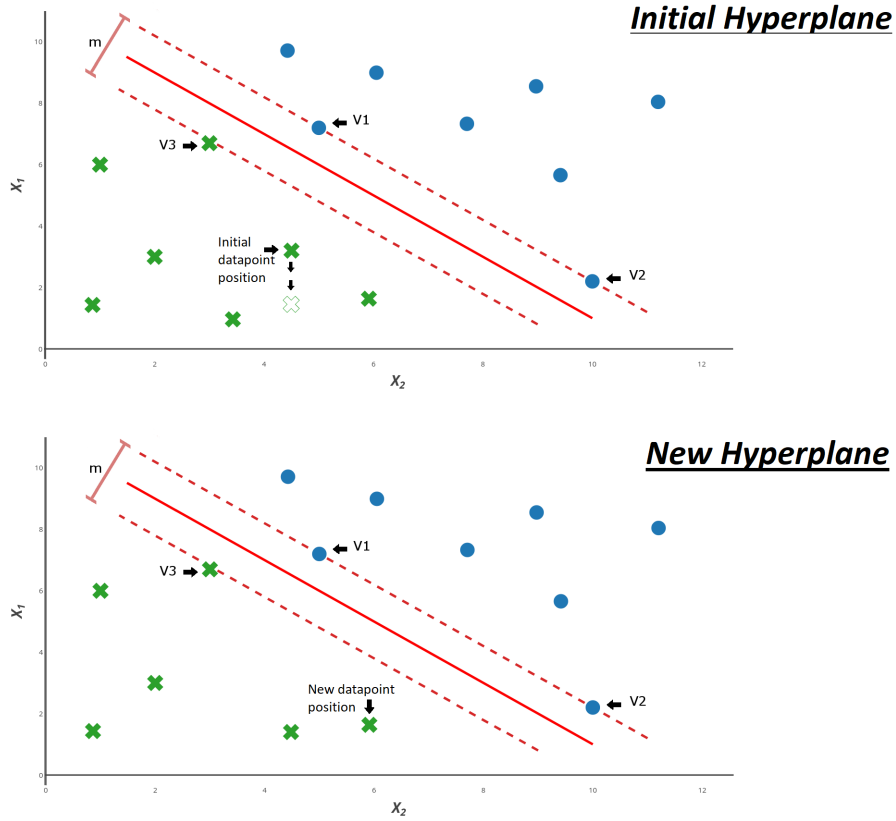


Figure 3.16: Datapoint changed without direct impact in the hyperplane

$H_2$ . In the case of linearly separable data, the support vector algorithm simply tries to create a separating hyperplane with largest margin, by reducing the  $\|w\|$ . Therefore, it is possible to formulate the following optimization problem:

$$y_n(x_n \cdot w + b) - 1 \geq 0. \quad (3.12)$$

As mentioned above, these types of problems can be solved using the Lagrange multiplier.

Although the hard margin method produces a hypothesis that is perfectly consistent with the training data, it has almost no applicability in the real world since most of the data is not linearly separable and possesses noise. Researchers developed a method based on the maximal margin algorithm that enables the use of it in nonlinearly separable data, the soft margin classifier [204]. This method tolerates outliers and noise in data without altering drastically the final solution, making the data linearly separable. In order to achieve this, the learning method allows misclassification on the training data when necessary, this is done by introducing a cost  $\xi_n$  to the Equation 3.12, resulting in the equation:

$$\begin{aligned} y_n(x_n \cdot w + b) - 1 &\geq 1 - \xi_n & n = 1, \dots, \delta \\ \xi_n &\geq 0 & n = 1, \dots, \delta, \end{aligned} \quad (3.13)$$

where  $\delta$  is the total number of data points and  $x_n$  and  $y_n$  are the values/labels at the given  $\delta$ .

From these equations, it is possible to conclude that the corresponding  $\xi_n$  must exceed unity and consequently,  $\sum_n \xi_n$  stipulates the upper bound for the number of training errors. With these conclusions, it is possible to update Equation 3.11 and define a new optimization problem:

$$\begin{aligned} \min \quad & \|w\| + C \sum_{\delta=1}^{\delta} \xi_n^k \\ \text{s.t.} \quad & y_n(x_n \cdot w + b) \geq 1 - \xi_n, \forall n, \end{aligned} \quad (3.14)$$

where  $C$  is a parameter defined by the users that influences how much misclassification errors suffer penalization, i.e., a larger  $C$  will penalize higher the errors.  $k$  stipulates the type of problem the function becomes. For instance, for a  $X \subset \mathbb{N}$ , the problem is a convex problem, except for  $k = 0$ , which eliminates completely  $\xi$ . For  $k = 1, 2$ , the problem, along with the convex optimization problem also becomes a quadratic programming problem [205].

Even though the creation of a soft margin method increased the usability of SVM, the most common problems in the real world remain of non-linearly separable, even with the appliance of the soft margin technique, as seen in Figure 3.17. Therefore, to simplify the problem and make it linear separable again, it is necessary to have a higher dimension space. The data points should be mapped into an alternative higher dimensional space, denominated as feature space [206].

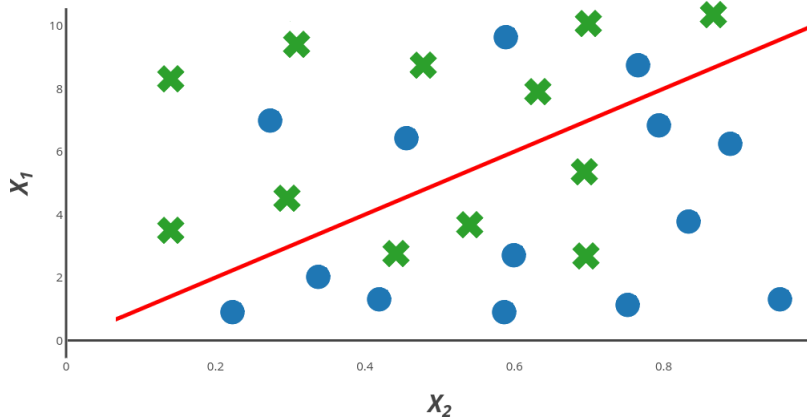


Figure 3.17: Two-dimensional data points that cannot be separated by a linear hyperplane

The use of a kernel functions enables the operations to be in the input space rather than the potentially high dimensional feature space, which would have increased the complexity of the problem. Nevertheless, the learning process is still dependent upon the number of training patterns and a large training set, to provide a good data distribution for a high dimensional problem.

The conclusion made in the kernel theory is based on the work Reproducing Kernel Hilbert Spaces (RKHS) [207]. This work concluded that an inner product in feature space had an equivalent kernel in input space:

$$K(x, x') = |(x), (x')|. \quad (3.15)$$

This means that the kernel can represent a legitimate inner product in feature space, coming from the input space. This core concept enables the transformation of a non-linearly separable problem into a linearly separable problem by just increasing its dimension. The resulting linearly separable problem can be solved by SVM. Figure 3.18 exemplifies a non-linearly separable problem transformed into a linearly separable problem through the use of the kernel function.

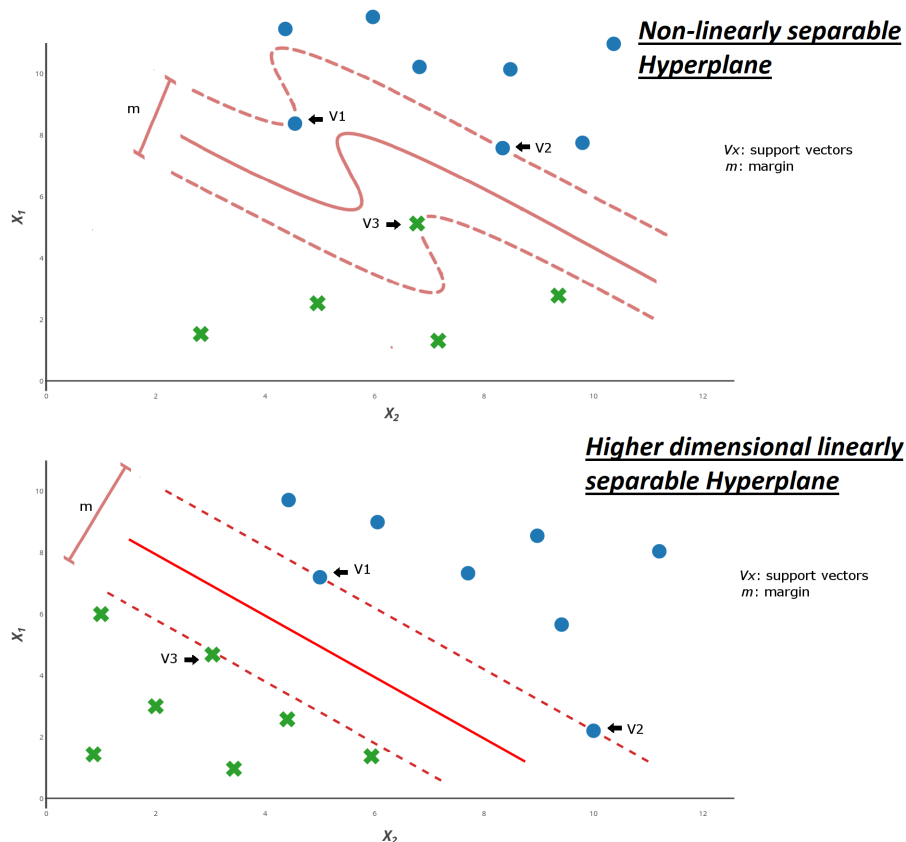


Figure 3.18: Dimensional space before and after application of kernel function

Data normalisation might be required for certain kernels; therefore, data normalisation must be taken in consideration depending on the used kernel function. Common kernel functions used by researchers are:

- **Polynomial:** Polynomial kernel function is frequently used in non-linear modelling. There are two formulas used for this, being the second one the preferable since it avoids problems with zero.

$$K(x, x') = |x, x'|^n, \quad (3.16)$$

$$K(x, x') = (|x, x'| + 1)^n, \quad (3.17)$$

Where  $n$  is the dimension.

- **Multi-layer perceptron kernel:** The multi-layer perceptron kernel is a variant of the Multi-layer Perceptron (MLP) learning algorithm presented in the ANN with a single hidden layer implementation:

$$K(x, x') = \tanh(\rho|x, x'| + \varrho), \quad (3.18)$$

where  $\tanh$  represents the hyperbolic tangent function,  $\rho$  is the scale of the values and  $\varrho$  is an offset value. In the MLP kernel implementation, the SE corresponds to the first layer while the Lagrange multipliers correspond to the weights.

- **Radial basis function kernel (RBF kernel):** Radial Basis Function (RBF) kernel are mostly used in SVC. They produce a piecewise linear solution, which are attractive when discontinuities are tolerated.

$$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right), \quad (3.19)$$

where  $\sigma$  is an input constant parameter.

- **Dirichlet kernel:** The Dirichlet kernel, sometimes referred as Fourier series kernel, is a less used kernel, since it has already been shown its poor performance [208]. It is calculated through the formula:

$$K(x, x') = \frac{\sin\left(n + \frac{1}{2}\right)(x - x')}{\sin\left(\frac{1}{2}(x - x')\right)}, \quad (3.20)$$

where the kernel interval is  $[-\frac{\pi}{2}, \frac{\pi}{2}]$ , and  $n \in \mathbb{N}$ .

- **Tensor product kernel:** Through the use of tensor product of kernels, which is the product of two vectors creating a vector space, it is possible to create multidimensional kernels [209]:

$$K(x, x') = \prod K_n(x_n, x'_n) \quad (3.21)$$

- **Additive kernel:** Another approach taken by researchers is the addition/sum of different kernels to create a more complex kernel. This is guaranteed since the sum of two positive definite functions is a positive definite:

$$K(x, x') = \sum_n K_n(x, x') \quad (3.22)$$

- **Splines and bsplines kernel:** Spline and bsplines kernels have a large appeal in the research community since they have a high level of flexibility. Their implementation, however, requires a higher level of statistical knowledge when compared to the previous mentioned examples.

SVM, when applied to regression problems, are denominated as mentioned above as SVR. SVM can change their area of application to regression by introducing an alternative loss function [210]. In order to accomplish this, the loss function must be modified to

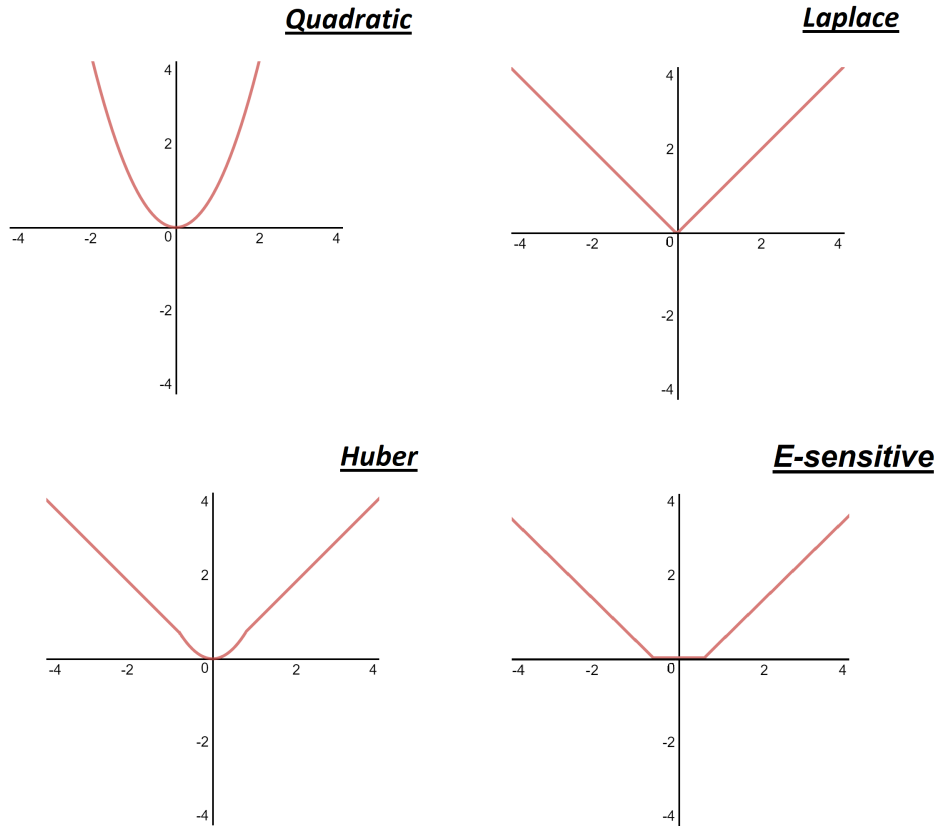


Figure 3.19: Examples of loss functions

include a distance measure. Possible loss functions can be quadratic, Laplace, Huber and  $\varepsilon$ -sensitive, as exemplified in Figure 3.19

The loss function in the quadratic example corresponds to the conventional least squares error criterion. It is the simplest to explain and use. Assuming that the problem of approximating the set of data,

$$D = \{(x^1, y^1), \dots, (x^l, y^l), \quad x \in \mathbb{R}^n, y \in \mathbb{R}, \}, \quad (3.23)$$

whose linear function is defined by:

$$f(x) = |w \cdot x| + b, \quad (3.24)$$

it is possible to outline the optimal regression function as:

$$\Phi(w, \xi) = \frac{1}{2} \|w\|^2 + C \sum_n (\xi_n^- + \xi_n^+), \quad (3.25)$$

where  $C$  is an input value, pre-specified and  $\xi_n^-$  and  $\xi_n^+$  are variables that represent the upper and lower constraints of the created output.

The Laplacian loss function is less sensitive to outliers than the quadratic loss function. Huber proposed the loss function in the example above as a robust loss function that has optimal properties when the underlying distribution of the data is unknown. These three loss functions produce no sparseness in the SE. Sparseness in SE reflect in fewer non-zero values and, since the prediction time for SVM, is proportional to the number of non-zero dual variables, it will create lower prediction times [210]. Vapnik proposed the loss function  $\varepsilon$ -sensitive, in order to tackle the sparsity problem. It is an approximation to Huber's loss function that enables a sparse set of SE to be obtained.

The above examples of loss functions work mainly with linearly separable data and, just like its classification counterpart, SVR can use kernel functions to transform non-linear data into linear data and thus making a prediction in regression problems with different types of data distribution.

### 3.6 Software tools

The importance of machine learning in many industries along with the interest that the research community has, originated a variety of frameworks to support machine learning solutions. The frameworks can be for generic general use in machine learning or they can be created for a specific type of machine learning problem. In this section, the frameworks presented will be of the generic type.

Load forecasting is just one sub-area of machine learning's applicability, therefore, the number of generic frameworks, and consequently its user base, highly outnumbers the specific load forecasting frameworks. The vast majority of researchers in the load forecasting area use these generic frameworks instead of specific ones. This is mainly due, as mentioned above, to the fact that specific frameworks are usually limited to specific data. There are many different frameworks in machine learning, nevertheless, there are a set of frameworks that tends to be more popular amongst researchers. Amid these set of frameworks Tensorflow [211], Mllib: Machine learning in apache spark [212], Caffe [213], Torch [214], Amazon Machine learning [215], Microsoft CNTK [216], Singa [217], IBM Watson [218], Rapidminer [219] and WEKA [220] standout, being WEKA the framework used in this work.

- **Tensorflow:** Tensorflow is an open source framework developed by Google Brain [221] for different language understanding and perceptual tasks. It has a broad area of applications in machine learning, being deep learning the one that stands out most. Tensor Flow is compatible with most new GPUs and CPUs, it incorporates many Google services such as Speech recognition, Google Photos, Gmail, or Google Search.

Google's framework provides a different set of APIs ranging from high-level APIs up to kernel APIs. The programming languages used are Python, C++, Java and Go. Tensorflow has also broad use cases in the optimization area. It uses data flow graphs to perform these tasks. The mathematical computations are calculated using a directed graph containing nodes and edges. These nodes can act as the endpoints where data is fed or used to implement the operations. The edges represent the input/output associations between different nodes.



- **MLlib: Machine learning in apache spark:** Machine Learning in Apache Spark (MLlib) is Spark's largest distributed machine learning open source library. MLlib is an implementation of standard learning methods for common learning settings, which includes classification, regression, clustering, dimensionality reduction and collaborative filtering. It was developed in Scala and includes Python, Scala and Java APIs.

MLlib's rapid growth and adoption, is mainly due to its open-source community, which contains contributions from over 140 people. MLlib tries to simplify the development of machine learning use cases by providing developers with a wide range of tools.

- **Caffe:** Caffe is a machine learning framework that works with C, C++, Python, and MATLAB. The Caffe machine learning framework has its focus points in speed and modularity. The main area of application is computer vision/image classification by leveraging Convolutional Neural Networks (CNN) [222]. One of the appeals that Caffe has, is the large collection of pre-trained models that does not require coding or implementation.

Caffe is often described as the better suited framework for application development, differing from other frameworks such as Tensorflow, who are suited better for research. However, Caffe is not a suitable framework when working with areas such as sound, text or time series. Its applicability is mainly centred in computer-vision.

- **Torch:** Torch is often referred to as the simplest machine learning framework in the research community. It is easy to setup and its learning curve is easier when comparing to other machine learning frameworks. It was developed in 2002 and is mainly used in an Ubuntu environment. It can be found in companies such as Facebook or Twitter. Its programming language is a scripting language, LuaJIT, although a python version exists called PyTorch. Its scripting language is one of the reasons for Torch's appeal, due to its simplicity and ease in reading and understanding.

The goal of Torch is to increase flexibility and speed in developing scientific algorithms while maintaining a low complexity in the process. Torch has a large ecosystem API's in areas such as, computer vision, parallel processing, signal processing, audio, image, video, etc.

- **Amazon Machine learning:** Amazon machine learning services, Amazon Machine Learning (AML), are a collection of tools used for developing learning models without the need to code. AML is included in Amazon web services. The technology behind AML is used by Amazon's internal data scientists to power their Amazon Cloud Services. AML can interact with Amazon's S3 data, RDS or Redshift and can perform binary classification, regression or multi-class categorization.
- **Microsoft CNTK:** Microsoft CNTK is an open-source machine learning framework, mainly used in speech recognition. Nonetheless, it can also be used for image and text. CNTK uses CPUs and GPUs for their calculations and is similar to Caffe, working with programming languages such as Python, C++, and even command line. CNTK provides higher performance and scalability when compared to TensorFlow; it also supports multiple simultaneous operating machines.

- **Apache Singa:** Apache Singa sets its focus on distributed deep learning [223]. Singa's main areas are image recognition and Natural Language Processing (NLP). Singa is divided into three components: IO, Model and Core. The IO component contains classes used for reading/writing data to the network and disk. The Model component focus on the algorithms and data structures used for machine learning models and the core component handle tensor operations and memory management functions.
- **IBM Watson:** IBM Watson is a machine learning framework that focuses on question answering, a sub-area of natural language analysis [224]. It applies advanced natural language processing, knowledge representation, automated reasoning, information retrieval and diverse machine learning algorithms when doing text analysis. Even though its focus relies on question answering, IBM Watson is largely used by the research community in diverse other machine learning areas such as, hardware optimization. IBM Watson API's are usable through IBM's webservice platform Bluemix [225]. IBM Watson can be used in applications that try to replicate human actions such as, hearing, talking, seeing, reading, tasting, interpreting or learning.
- **Rapidminer:** Rapidminer is a software platform that englobes data preparation processes, machine learning application, and predictive model deployment. It uses a drag and drop approach throughout most of its solution creation process, which reduces the need for pre-needed programming knowledge. Along with machine learning procedures, it also provides data mining procedures, such as, the Extract, Transform, Load (ETL) process commonly used in data loading and transformation. Through the use of python and R, users can extend already implemented learning schemes, models and algorithms.
- **Weka:** WEKA is a framework that allows users to apply different machine learning techniques. Users can use already implemented learning methods or implement their own learning methods that can be used without the need to re-implement data manipulation or scheme evaluation. The WEKA framework is released as an open source software, enabling users' free access to the source code, which leads to a facilitated creation of projects that incorporate or extend WEKA.

The main focus in WEKA relies on providing a comprehensive collection of machine learning algorithms and data pre-processing tools. The variety of already implemented machine learning methods in addition with a simple graphical user interface, allows users to quickly use and compare different machine learning methods on new data sets. WEKA provides a JAVA API, which users can use to automate the integration of new learning methods. The framework includes learning methods for classification, regression, association rule mining, attribute selection and clustering. WEKA's main graphical user interface is the Explorer, as seen in Figure 3.20.

It is built on a panel-based interface and is where most of the researcher work is done. Each panel corresponds to a different task. In the "*Preprocess*" panel, users can load and transform their datasets through the use of already implemented WEKA data pre-processing procedures. The WEKA framework calls these "*filters*".

In the "*Classify*" tab users can apply and configure different machine learning methods, define their training set or use cross-validation. Figure 3.21 depicts an

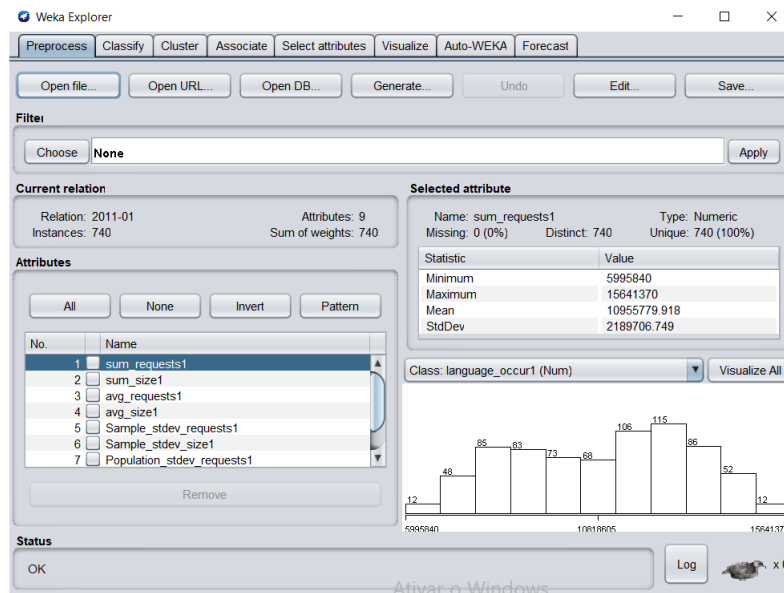


Figure 3.20: WEKA Explorer

example with one of the dissertations Wikipedia datasets, where an SVM variant was applied.

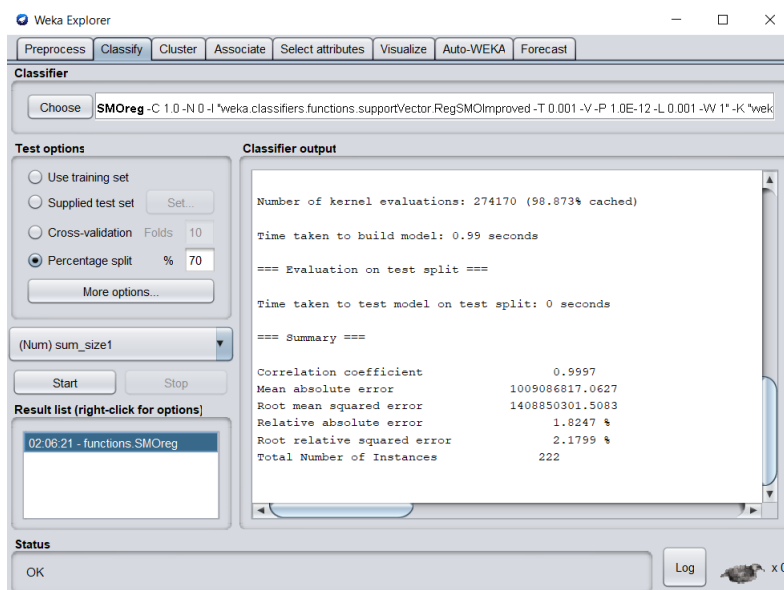


Figure 3.21: WEKA Classify example with Wikipedia dataset

In the “*Classify*” tab, users have an output panel that enables them to easily analyse their model performance, from a pre-defined metric selected by them. The “*Cluster*” panel enables users to run a clustering algorithm on their datasets. It provides simple statistics for evaluation of clustering performance. WEKA also has a dedicated panel for attribute selection, “*Select attributes*”, where the users can apply algorithms and evaluation criteria for identifying the most important attributes in their dataset. The last tab in the standard version of WEKA is the Visualize one, this panel has a

color-coded scatter plot matrix, regarding the distribution of datapoints between a set of variables.

Along with the standard tabs, WEKA has a dedicated time series analysis environment developed by Pentaho [226] that can be installed through WEKA packet manager, that updated adds additional tabs to WEKA, as seen in Figure 3.22. This time series framework uses WEKA regression algorithms in conjunction with a process of formatting data. The formatting of the data consists in removing the temporal ordering of individual input examples and encode instead their temporal dependencies via additional input fields.

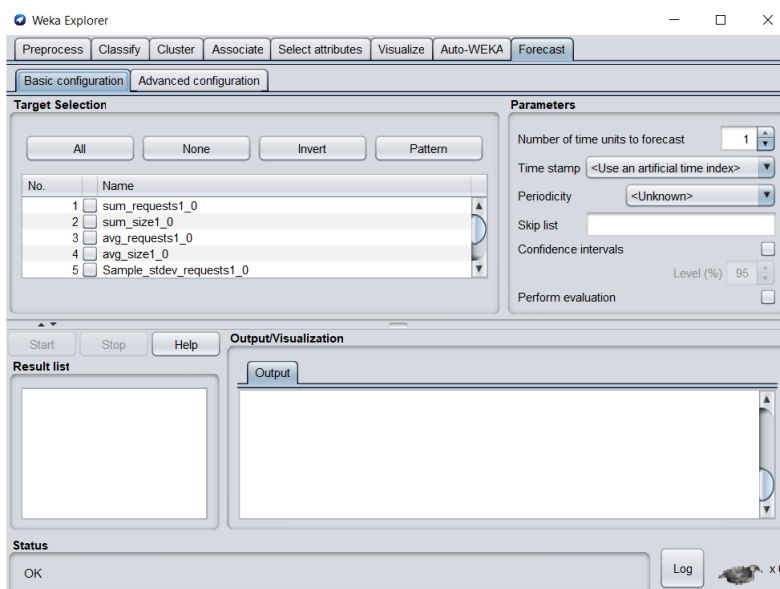


Figure 3.22: Pentaho time series framework incorporate in the WEKA GUI

The model construction approach is identical to the one taken with standard WEKA. However, in terms of output it differs substantially from standard WEKA. It is possible to define multiple objectives in a single run, predict for more than one instance upfront or choose additional evaluation metrics, such as MAPE, which is not available in standard WEKA.

### 3.7 Conclusion

This chapter presented generic concepts of machine learning along with the terms and approaches of supervised and unsupervised learning and regression and classification problems. A literature review was presented concerning the challenges and applicability of machine learning. Linear regression, ANN and SVM were explained in detail. The necessary information needed for understanding their application in the dissertation was also presented. Lastly, a review on the most used software tools in machine learning was conducted and presented. In the following chapter, the proposed approach adopted in this work is introduced and described in detail.

# Chapter 4

## Proposed approach

The proposed approach chapter introduces a generic solution architecture along with the description of all the steps involved in its process, disclosing the specific data and therefore inherent changes to each problem. Following the generic solution architecture presentation, its application is described in detail for both case studies.

### 4.1 Solution architecture overview

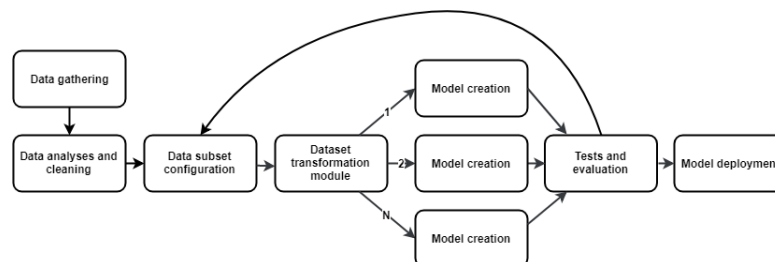


Figure 4.1: Generic solution architecture diagram

Figure 4.1 presents the generic architecture of the solution developed in this work. The approach is applied in the Wikipedia and Energias de Portugal (EDP) problems, with each step adjusted in accordance with the data and needs of each individual problem.

#### 4.1.1 Data gathering, analysis and cleaning

In the first step, the data gathering process, the data is treated and collected from single or multiple sources and stored locally. This is achieved through a manual approach and/or using scripts.

After the gathering process, the data is analysed and cleaned. Data analysis tools, such as, WEKA Graphical User Interface (GUI), text editors, Microsoft Excel and data source documentation are examples of tools that were/can be used. In the cleaning process, the methodologies used are correlation coefficient analyses, to eliminate irrelevant or repetitive

data, direct manipulation in the dataset through Microsoft Excel and/or text editors, e.g., wrongly formatted data, correctly re-formatted. The use of average values to fill in missing values and/or the application of a cleaning script, which incorporates all the above methods plus some extra ones. This last one, however involves a detailed analysis and understanding of the data and the problems contained in it.

### 4.1.2 Data subset configuration and transformation module

Subsequently to the cleaning step, the configuration values for different train and test sets are defined. This method enables the possibility of testing different dataset configurations, a partial grid search approach presumably [227], avoiding thus the use of a complete grid search methodology, which would be extremely resource consuming. The subset configuration values used in this dissertation were define in accordance with the human interaction/patterns. For the granularity the values 1,8 and 24 where chosen, while for the time window 1, 4, 8, 12, 24, 48 and 168 were used.

After defining the configuration values, the transformation module is pursued. This module involves a set of mandatory transformations to the train and test sets, selected in the *Data subset configuration* step. Each transformation cycle is applied, one at a time, to a train and test set pair. Figure 4.2 depicts the steps involved in a single transformation cycle.

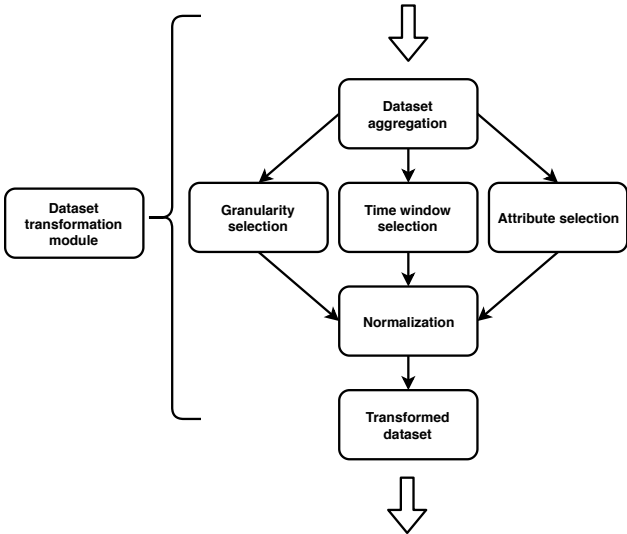


Figure 4.2: Steps involved in the data transformation module

The dataset aggregation step is only applied in cases where more than one train and test sets is available. The module takes multiple train and test sets and creates new train and test set, containing the different possible combinations of the aggregated pairs. For instance, taking this work as an example, a resulting train and test can contain the data from sets of the months of January, February and March. If required, it is possible to define a specific set of pairs as input for the train and test set. For example, a resulting train and test set from such input can be the combination of the months January, April, June and August.

Afterwards, the next module applies a granularity value to each record. The granularity defines the level of detail in the data, which in this work is always in an hourly scale. Figure 4.3 portrays a scenario before and after the granularity application process. The initial data consisted of three different records, with three attributes. The records had a time dimension inherent to the order of the records, i.e., the first record represented the data from the first hour, the second record the data from the second hour, and so forth until the end of the dataset. Through the application of a granularity value, the initial dataset was transformed into a dataset with records of  $G$  granularity, which in the example is equal to three. Each record from the new dataset is thus the sum of  $G$  records of that attribute.

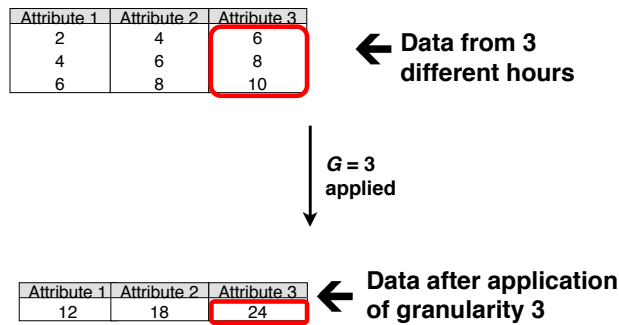


Figure 4.3: Data example where granularity of 3 was applied

Following the granularity step, the time window definition step can be applied or not. In this step the number of time window units back in time a record will contain is defined. Taking again this work as an example, each record would contain  $TW$  hours back in time. Since the granularity step is applied a priori, the created history/time-window shares the same scale. Figure 4.4 depicts an example of an application of  $TW$  three to a dataset.

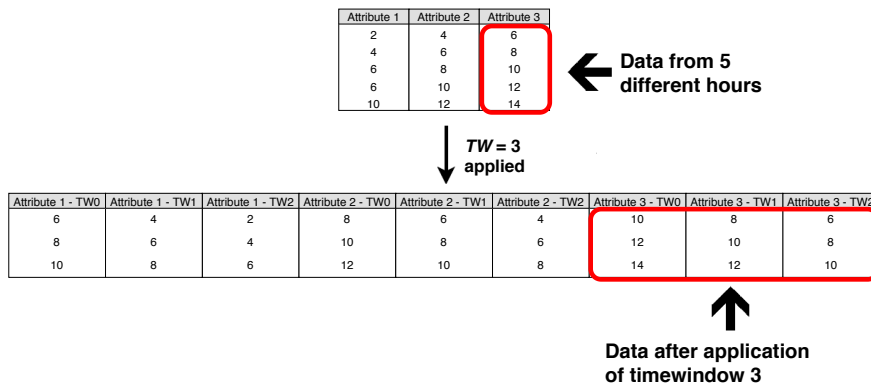


Figure 4.4: Data example where time window of 3 was applied

Subsequently, to the appliance of granularity and time window comes the attribute selection step. The different sets of possible/desired combinations of attributes are defined in this step, creating different combinations of training and test sets. Just like the time window step, this step can be applied or not.

The last step in this approach, consists in applying normalization on the resulting train and test set. For each train and test set that is normalized, there is another one that remains

unaltered. This is done in order to evaluate how normalization affects the performance of the created model. The transformed train and test set, thus results from a different set of transformations/configurations values, as depicted by Figure 4.5:

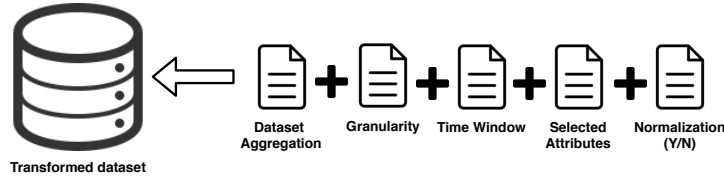


Figure 4.5: Final dataset composition diagram

### 4.1.3 Tests, evaluations and model deployment

For each newly created train and test set pair, a model is created and evaluated. Through this process, it is possible to select the most suitable sub-set of datasets. This step also functions as a validation point: if the subset of data presents unviable results for further testing, a new iteration of steps is initiated. When finally, a suitable model for each problem is found, the resulting model is deployed.

In this work, a grid search approach was initially taken in consideration. However, as mentioned above, this was impractical, which eventually led to the adaptation of a subset approach. The total number of created, and tested datasets can be calculated through the following formula:

$$TotalDatasets = M \times G \times TW \times A \times 2, \quad (4.1)$$

where  $M$  is the number of different months,  $G$  is the number of different values of granularity applied,  $TW$  is the highest value of Time Window a dataset can have,  $A$  is the number of attributes before the application of the transformation module and, lastly, the value 2, which represents the normalization and non-normalization of datasets.

In the case of the Wikipedia problem, a total of 169344 datasets were needed to be created and tested through the application of a grid search approach, which translates to approximately 4.65 Terabytes of initial raw data. Through the use of a subset approach only 1440 train and test sets were used. For the EDP problem, 8064 datasets were needed for a grid search approach. The lower number of datasets when compared to Wikipedia is due to the availability of only a single month of data and a lower number of attributes, provided by EDP. Nevertheless, by using the subset approach only 120 train and test sets were needed.

## 4.2 The Wikipedia problem

Server load prediction can be used in many different scenarios. However, the scenarios in which its use stands out are clearly the ones with high activity. Wikipedia falls into such category. Each day, an enormous quantity of load is generated in the servers, which



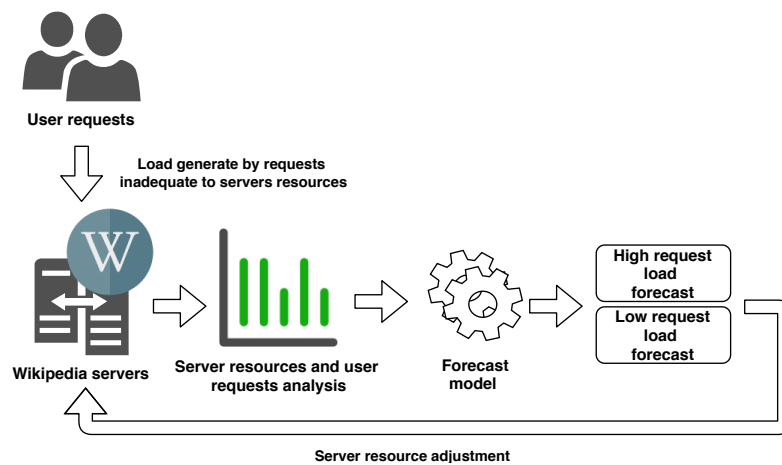


Figure 4.6: Wikipedia problem overview

of course takes a toll on Wikipedia’s servers. Correct adjustment of resources is utmost important when creating a reliable solution for Wikipedia users. Figure 4.6 depicts an overview of the problem Wikipedia servers face every day and how the approach developed in this dissertation can tackle does problems.

Each Wikipedia server has an initial resource configuration, that is set in accordance with the foreseen incoming load. However, this type of configuration is not the most suited throughout the whole day. Resources must be adapted continuously in order to guarantee SLA. Server resources must be optimized from an economical perspective and from the hardware’s lifespan perspective. Server load prediction can achieve this. By analysing the load patterns of previous data, a model can be constructed that predicts incoming load and even though the load is generated mainly by human interaction, past load patterns might not be enough to create a reliable model. The inclusion of external variables, such as calendar information and/or global events information, can increase the models accuracy.

With a created model, resources can be adjusted upfront for load peaks and low points, thus maintaining the adequate resource for the SLA, economical variability, hardware lifespan and/or green computing.

### 4.2.1 Solution approach in the Wikipedia problem

Figure 4.7 depicts the solution architecture that resulted from adapting the general architecture to the needs of the Wikipedia problem. The initial steps involve downloading, formatting, and cleaning the data available from the Wikimedia foundation and later on store said data locally. With the stored data, a transformation module is applied to each dataset, which creates new training and test sets. The original datasets are preserved for future modelling and testing and as functioning benchmark baseline. The newly created train and test sets are submitted to the model creation and testing module. This module creates models based on linear regression, ANN and SVM. After finishing the modelling and evaluation process, two distinct files are created, one file contains all the raw unselected testing results and a second file that contains the information from the runs of desired datasets and evaluation metrics.

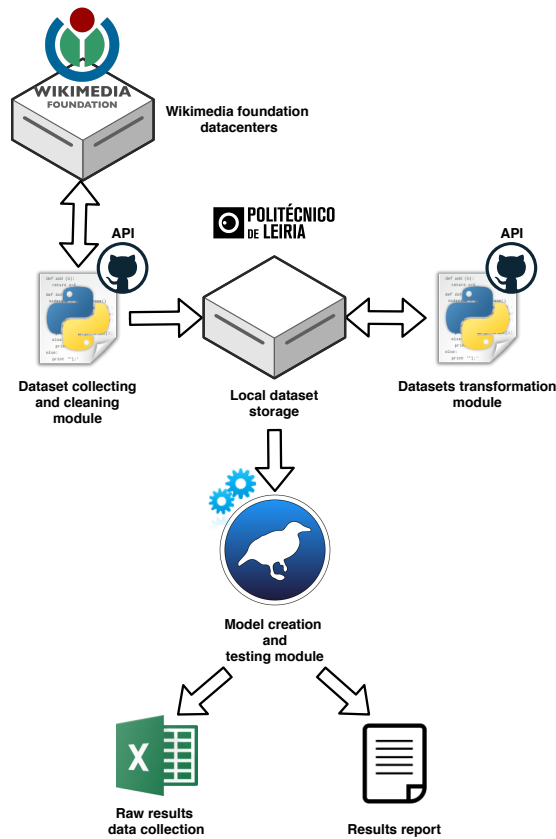


Figure 4.7: Wikipedia solution architecture diagram

## Data overview

The initial data acquired for the Wikipedia problem was collected from the Wikimedia foundation. Whenever a request of a page is made to Wikipedia, whether for reading or editing, the made request always passes through one of Wikipedia’s internal hosts. From this request, the project name, the size of the page, and the title of the page are collected. The resulting data is written into a page view raw dataset where one record is the hourly aggregate of the pageviews/requests and size of an individual page. The datasets are stored under the nomenclature showed in Figure 4.8.

```

${YEAR}/${YEAR}-${MONTH}/pagecounts-${YEAR}${MONTH}${DAY}-
${HOUR}0000.gz
${YEAR}/${YEAR}-${MONTH}/projectcounts-${YEAR}${MONTH}${DAY}-
${HOUR}0000|

```

Figure 4.8: Wikimedia dataset storing naming nomenclature adopted

The time used in this nomenclature is in Coordinated Universal Time (UTC) time zone, it refers to the end period of the hourly aggregate. This information is crucial when analysing data day-wise, since the nomenclature might suggest that a day worth of data is from hour 0 to 23 of the same day. However, the correct data regarding a full day is hour

1 to hour 0 of the following day. Figure 4.9 exemplifies five records of data of the initial dataset.

```
en.b 19th_Century_Literature 1 8422
en.b 360_Assembly/360_Instructions/BC 1 12217
en.b 360_Assembly/360_Instructions/LM 2 0
en.b 360_Assembly/Branch_Instructions 1 53068
en.b 360_Assembly/Pseudo_Instructions 1 8853
```

Figure 4.9: Record structure example

Each attribute is split by a space, having a record a total of four attributes. The first attribute refers to the language the page is written in. In the example given, all records start with *en* followed by a dot as second parameter, e.g., the first line has *en.b*. The *en* refers to the Wikipedia language, which in this example is English. The second parameter after the dot indicates what type of page that was requested. Table 4.1 enumerates the different page types. Attributes that do not contain any second parameter are Wikipedia projects.

Table 4.1: Different page types from Wikipedia

Type of page	Parameter
Wikisource	.s
Wikiversity	.v
Mediawiki	.w
Wikibooks	.b
Wiktionary	.d
Wikimedia	.m
Wikipedia mobile	.mw
Wikinews	.n
Wikiquote	.q

The second attribute is the name of the retrieved page. In the example given, for the first record this would be *19\_Century\_Literature*. The third attribute indicates the number of times a specific page was requested within that hour. These request numbers are not solely unique visits. The last attribute is the size of the returned content. In the given example, the first record had just one request which accounted in total for 8422 response bytes.

Each hourly dataset was composed by multiple of dozens of millions of records, with each record having a total of four attributes. Although the raw data presents an enormous quantity of data, the resulting dataset was significantly reduced by containing hourly aggregates of data, instead of unique page requests. On average, a final hourly dataset for a month has 720 records and is composed by nine different attributes, nevertheless, this will be explained in detailed manner in the next subsection.

An early data analysis was conducted on this data, which led to the following conclusions:

- Downloading and storing all data locally was not feasible since a day worth of data translates to between 20 Gb and 40 Gb of storage occupancy.

- Non-consistent and incomplete data was found in the raw dataset. For instance, the language attribute referring to the English language was found with the values of *EN*, *en* and *En*. Another common problem found were records containing just partial information, with attributes often left blank.
- Due to the nature of this project different workstations were used, which led to problems with the encoding of the data. The raw dataset contained data from a variety of languages, which led to problems with certain languages, such as, Chinese. The solution found was the re-encoding of the original data into a readable format for the sake of not discarding English related data wrongly interpreted.
- The data contained outliers, which in most cases was residual and easily corrected. However, the month of April of the year 2016 was an extreme case where data corresponding to around one fourth to a half of the month was an outlier.

### **Dataset collecting and cleaning module and concurrent system**

In the data gathering and cleaning process an initial python script was developed. The objective was to automate processes associated with all data gathering, formatting and cleaning tasks. Through the script it was possible to treat the large data quantities involved as small, temporary, data fragments and keep the process running non-stop.

Through a literature review, it was concluded that there were no current implemented solutions for the above-mentioned task. As a contribution for future researchers who would like to use the Wikipedia datasets locally and without data problems, an API based on the initial script was developed and published through GitHub at `WikiGather: URL`.

Figure 4.10 depicts the different functions available for single hour treatment, through the API.

The first API function presented in the figure is the *encoding()* function. This function was implemented to act as a counter measure to encoding problems that users with workstations from special regions might find. It is a mere support function, with no influence in the dataset.

The *get\_URL()* function creates a list of URLs containing all hourly datasets to be treated. Users can define what period in a month they would like to use and/or a period from various different months. The resulting list will be used by the dataset creation functions. *create\_ds()* allows users to create a dataset consisting of nine attributes. The created dataset contains information from all languages available in the raw files. The attributes of the created dataset are the number of requests made in an hour, generated load in an hour, average requests and load, standard deviation of requests and load, one using sample data and the other using the population and lastly, the number of unique page appearance differentiated by language. The *create\_ds\_lang()*, on the other hand, presents a similar approach to *create\_ds()*. The main difference relies in the fact that the information used in the dataset consists only of a single language, which is defined by the user. The attributes of created dataset are the same, with the exceptions of the number of unique pages per language. This attribute is deleted since its impact in a single language scope is almost none. The attribute would mostly increment with each record, rendering

## API functions

The API comes with a different set of functions that lets you adjust your Wikipedia datasets to the users needs:

```
# Encoding function to resolve encoding issues
def encoding():
```

```
#Generate list of the dataset names stored online
def get_URL():
```

```
# Create a Dataset with 9 attributes from the raw file,contains data from all languages
def create_ds():
```

```
# Create a Dataset with 8 attributes from the raw file, contains just data from the desired language
def create_ds_lang(desiredLang):
```

```
# Write a dataset out in CSV format
def write_CSV():
```

```
# Calculate the average,standard deviaton for request and webpage size
def calculate_all():
```

```
# Clear data from all variables, used when in need to restart a process
def clear_data():
```

Figure 4.10: WikiGather API: single hour API functions

its usability null. In any of the create functions, the hourly raw files are just temporarily decompressed for the information gathering process and deleted after that, thus avoiding storage problems.

In cases where users only desire to calculate the attributes without creating a dataset, users have at their disposal the *calculate\_all()* function. The keener users can update this function to use statistical metrics that better suit their problems needs. Changes made in this function will automatically be used when calling any of the creation functions.

For situations where users wish to erase all changes already taken, they can use *clear\_data()*. This function deletes all the steps already taken by other functions and completely cleans all information stored in the memory. Lastly, the *write\_CSV()* function enables the user to store locally all the datasets created, e.g., an user has the possibility to store a complete dataset and/or save single parts of a train and test set combination.

The functions mentioned thus far, were conceived with only a one-hour time-window in mind. However, if users need a different level of time-window they have at their disposal a set of functions which were conceived for a four-hour time-window range. Figure 4.11 presents the overview of the four-hour time-window function available through the API.

All the above functions were based on their one-hour granularity counterpart. Adjustments were made so that in the end each record contained the initial eight attributes of a specific language, and the eight attributes of time window 1,2 and 3.

#### 4 Hours functions

The API also provides functions for creating datasets with aggregates of data of more than a hour, in this template the 4 hours are used as default value:

```
# Create a dataset with 4 hours worth of data per record,contains data just from the defined language
def create_ds_4h(desiredLang):
```

```
# Write out a 4 hours record dataset in CSV format
def write_CSV_4h():
```

```
# Calculate the average,standard deviaton for request and webpage size
def calculate_all_4h(desiredLang):
```

Figure 4.11: WikiGather API: four-hour time-window API functions

Due to the fact, that the dataset creation process initially took between 5 and 6 days to accomplish, a concurrent system based on threads was implemented. This reduced the operation time to 1, 2 days and optimized the whole process. Figure 4.12 depicts an overview of the parallel system along with different tasks taken in the collecting and cleaning module (which gave origin to the above-mentioned API).

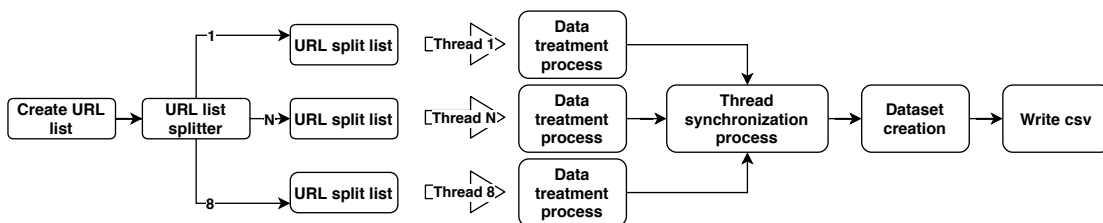


Figure 4.12: Collecting and cleaning module task overview with parallelism

As seen in the figure, the process starts by creating a list containing all the verified Uniform Resource Locator (URL) regarding a complete month. This step involves exploring and validating the data, since some of the raw datasets were corrupted and/or wrongly named, e.g., the attributes contained in the data were not the correct ones, instead being the attributes regarding a different pageview/raw file.

With the URL list successfully created, the next step is to forward this list to a splitter module which is responsible for dividing the list into equal  $X$  sub-lists, where  $X$  is the number of virtual CPU cores available in the workstation. In all our used workstations the number of cores were 8, hence why the example ranges from 1 to 8. Nevertheless, the module was written so that it can dynamically detect the number of available cores and adjust accordingly, which widens the horizon of applicability. Subsequently to the sublist creation, each list is fed to a thread and the data treatment process is initiated, which is exemplified in Figure 4.13.

The first task in the data treatment process is to download the files, which, as mentioned already, come in a compressed format. The files are then temporarily decompressed, treated and all relevant information is stored. When a complete treatment cycle is concluded, the

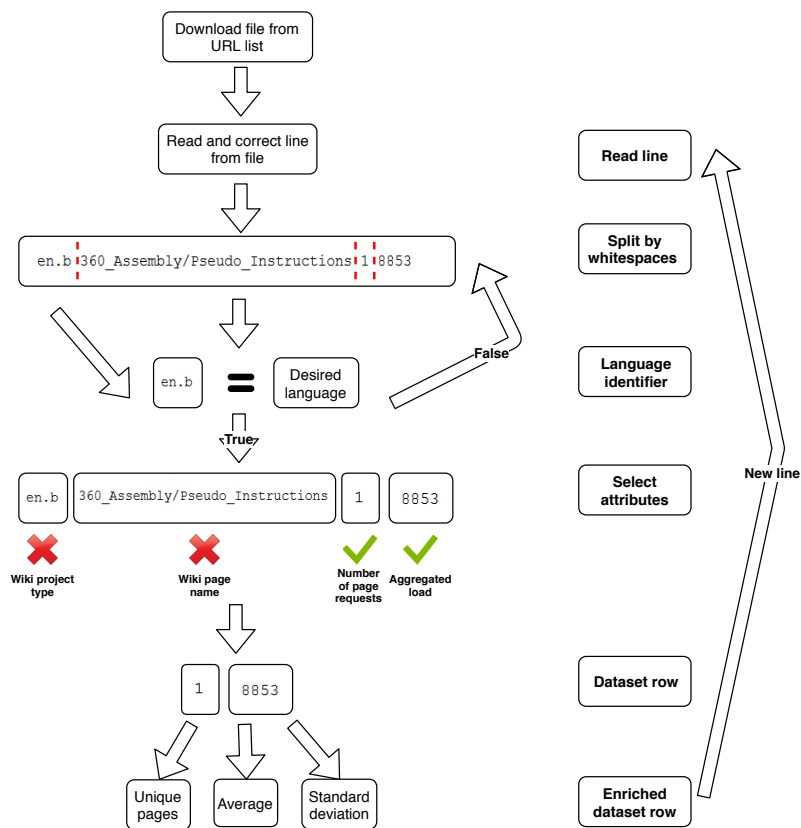


Figure 4.13: Data treatment: original row transformed into a dataset record

compressed raw file is deleted. With this methodology, it is possible to analyse the large data quantities without depleting the storage capacity. This process is repeated until all parts of data of a month are treated.

The succeeding step involves reading the data referring to an hour line by line. During this procedure, an algorithm tries to find and correct anomalies in a line, for instance, the number of requests and/or load may contain text information. In such situations the algorithm identifies and removes the incorrect data, re-using if possible the remaining information. Each line is then divided by white spaces, creating a total of four different attributes.

From these four attributes, the attribute project type is used to identify and analyse if the row contains data regarding our desired language or not. If this is not the case, the row is discarded without further treatment. In the next step, from the four resulting attributes, *project type* and *page name* are discarded from the final dataset. This is due to each row from the final dataset referring to the hourly aggregate of the desired language and these attributes refer to singular occasions.

When the process of creating a simple dataset row is finished, a data row enrichment process is initiated. This process is responsible for creating attributes that will be used in the dataset enrichment phase. The attributes originated in this process are the number of unique wiki pages that were requested in an hour, average values for aggregate load and number of requests and the standard deviation of load and number of requests. The standard deviation was calculated also using the population and a sample. The sample data

contained only information from weekends and the population information from all week. When all data of a single pageview/raw file is successfully treated, the algorithm passes to the next hour, repeating the process until all the datasets from a sublist entrusted to a thread are treated.

The thread synchronization process is then responsible for collecting across the different threads each hourly dataset and reorganize them correctly so that, in the end, a dataset is created with the correct temporal order in the records. With the collecting and treatment process successfully completed, the last step is to store the dataset in *.csv* format on the local storage partition. However, before starting the writing to file process, an algorithm checks if the format of each record and the number of attributes and records is correct.

### 4.3 The EDP problem

The EDP company, as an energy providing facility, faces a big challenge when trying to keep a balance between supply and demand. The Electrical Energy Storage (EES), depicted in Figure 4.14, refers to the process of transmuting electrical energy from a power network into a storable form [228]. It is a key factor when adjusting the production level. The storage of energy usually occurs in situations where the supply surpasses the demand and a peak in consumption is not foreseen.

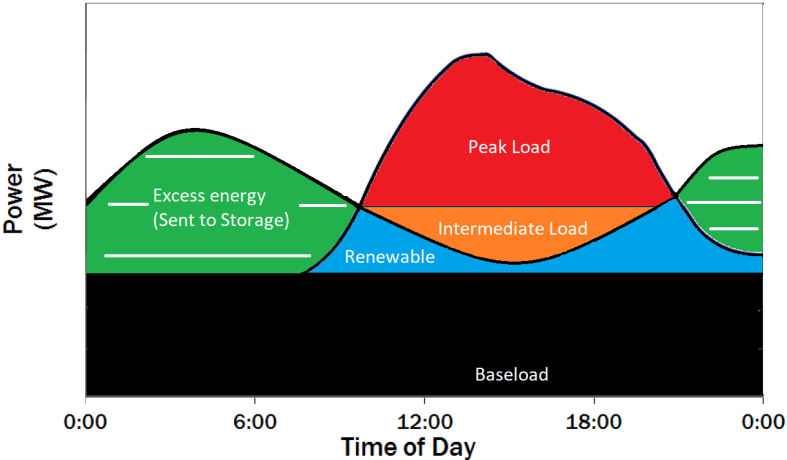


Figure 4.14: Energy storage in a Grid Load Profile

However, there are two problems in the process of storing energy: the quantity that can be stored and the time that energy can be stored before dissipating. Identifying consumption trends/patterns and adjusting production accordingly is thus indispensable. Figure 4.15, presents an example regarding the energy production levels throughout a day, commonly referred to as Grid Load Profile (GLP) [229].

Baseload generation is kept throughout the whole day. These generators are usually low on cost and/or face a penalty when turned off, for instance, modern coal units or nuclear power plants. The next instance in the grid is renewable energy, such as, solar and wind energy. This type of energy production has a downside in terms of its availability. Since, consumption peaks may not coincide with the renewable production time window, another



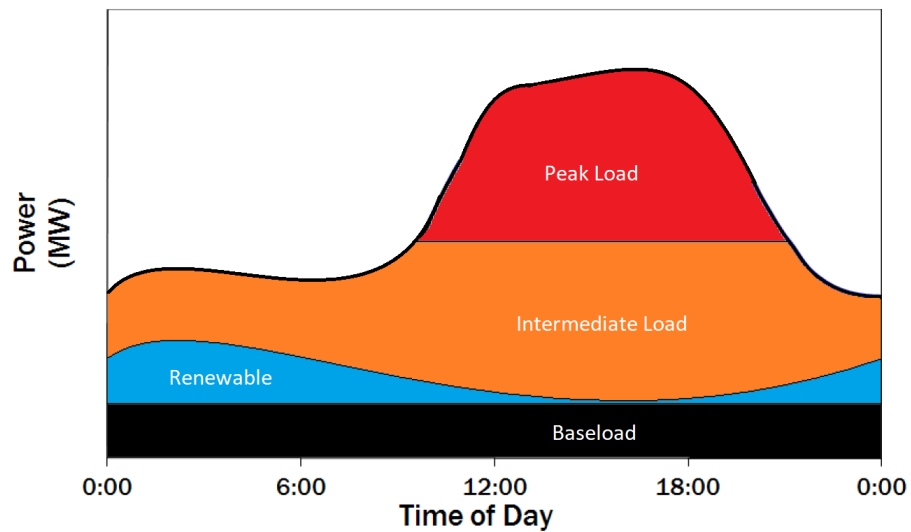


Figure 4.15: GRID during a day: baseload, renewable, intermediate load and peak load

source of supply may be required, the intermediate load. This type of generation tries to adapt to the consumption level, i.e., the production units, follow the ups and downs of the consumption load patterns. The units responsible for this type of energy tend to have higher prices and/or high operational flexibility, e.g., older coal plants and hydroelectric units. The last instance of the GLP is the peak load energy. The units involved in this type of energy have the highest cost, such as, diesel generator or open cycle natural gas combustion turbines. The use of this units occurs mostly in situations with exceptionally high consumption levels and/or unavailability of other energy sources.

Due to the volatility of renewable energy and the difficulties found in the storage of energy, the adequate use of stored energy is crucial for the correct functioning of the power system. For instance, the use of stored energy in situations where a high production level on renewable energy is foreseen or the risk of energy dissipation is high, increases the efficiency of the overall power system.

Nevertheless, the use of stored energy must be taken cautiously. If stored energy is frequently used, the power system might incur in a deficit in crucial situations with unusual high consumption levels. Accurate load forecasts in consumption and production levels can tackle such situation.

An accurate load forecast is an imperative tool for the correct operations of a power system, e.g., maintenance, contract evaluation, tariff rates adjustment, and/or scheduling [230]. Accurate load forecasts play also a vital part when defining new energy policies [231].

However, in order to create an accurate load forecast, there are various variables that must be taken into account. Weather variables such as, temperature, dew point, wind speed, cloud cover and humidity have a strong impact in the final forecast. Historical load data is another crucial variable. In order to achieve a better forecast result, an adequate number of input variables must be found. Lastly, the use of accurate load forecasts plays a key role when implementing new concepts of smart building and/or smart grids [232].

Figure 4.16 presents an overview of the overall problem along with solutions provided by the approach developed in this dissertation.

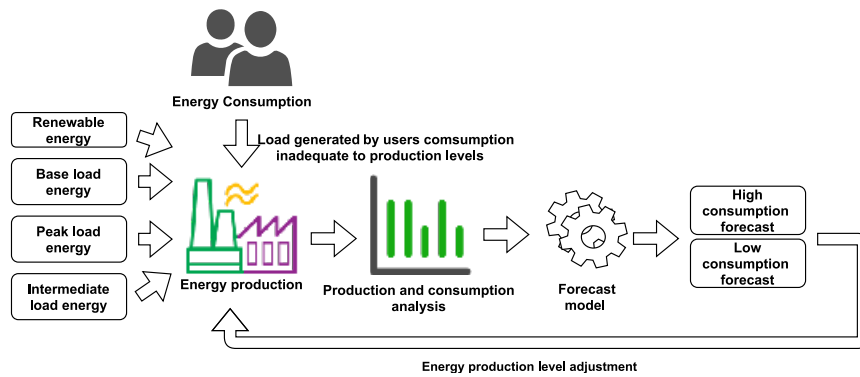


Figure 4.16: EDP problem overview with respective implemented solution

### 4.3.1 Solution approach in the EDP problem

The EDP solution architecture, as seen in Figure 4.17, is in its overall very similar to the Wikipedia one.

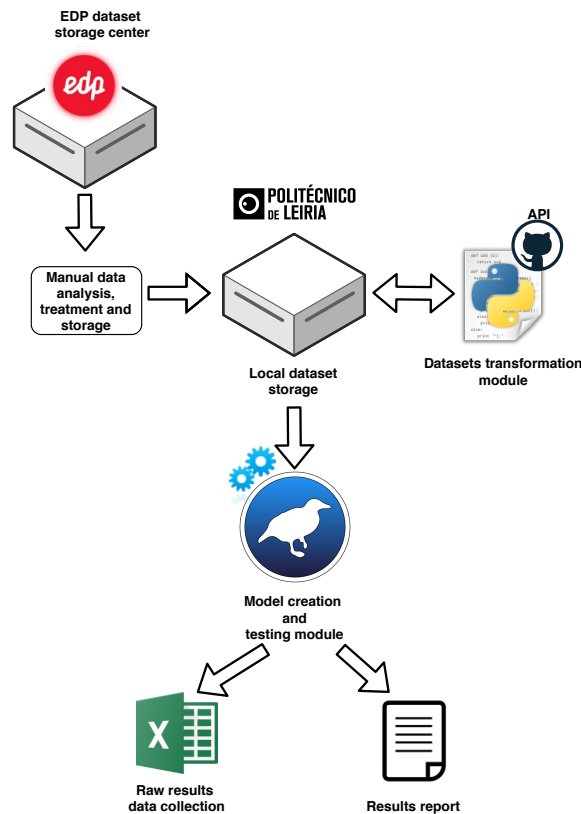


Figure 4.17: EDP solution architecture diagram

The main difference is in the gathering and treatment process. The whole process starts with the acquisition of the dataset. This step was facilitated by, Eng. José Manuel Terras, a staff member from EDP distribution, who provided the necessary dataset for further analysis. The dataset contained data from the month of April 2016 exclusively from the city of Leiria. The dataset was then manually analysed, treated and stored locally. The

treatment was fairly straightforward since the whole dataset presented a simple structure. The focus relied mainly on deleting unnecessary attributes, correct wrongly formatted data and/or value and convert the data to hourly records, i.e., the data came as fifteen minutes records. In order to have comparable results between, the two solutions, the granularity was adjusted to the same time interval/scale, which is an hourly scale. When this process was concluded, the following procedures were identical to the Wikipedia architecture. This was done in order to minimize the differences between the two architectures, so that the generalization level in each model can be compared.

## Data overview

The initial dataset had information regarding the energy consumption of the city of Leiria in the month of April 2016. The raw dataset came with five attributes, electrical tension, electrical current, two geographical labels denominated *Nível 1* and *Nível 2* and a combined attribute containing the date and hour. Figure 4.18 presents an example of the record structure of the raw dataset file.

Data+Hora	Nível 1	Nível 2	Corrente	Tensão
01/04/2017 00:00	LEIRIA	Zona URBANA	87	15000
01/04/2017 00:15	LEIRIA	Zona URBANA	73	15000
01/04/2017 00:30	LEIRIA	Zona URBANA	84	15000
01/04/2017 00:45	LEIRIA	Zona URBANA	74	15000
01/04/2017 01:00	LEIRIA	Zona URBANA	77	15000
01/04/2017 01:15	LEIRIA	Zona URBANA	88	15000
01/04/2017 01:30	LEIRIA	Zona URBANA	78	15000

Figure 4.18: EDP dataset record structure example without treatment

The granularity presented in this dataset was of 15 minutes intervals and, as already mentioned, this needed to be adjusted to the hourly range, reducing the initial number of records from 2880 to a total of 720.

Following the granularity adjustment, the quality of the attributes was evaluated. The attribute date-hour was the first to be excluded from the final dataset. Due to the granularity adjustment, each record represented the aggregate of an hour instead of the initial 15 minutes interval. Nonetheless, in virtue of the nature of the dataset, instead of updating the attribute *Data+Hora*, it was deleted since the dataset already gave this information, i.e., as a consequence of the datasets structure, the data already had an inherent temporal factor, which in this case was an hour per record, thus rendering the date-hour attribute obsolete.

The attributes containing geographical labels and electrical tension were also left out from the final dataset. These three attributes only presented one value, which in the case of the geographical labels was *Leiria* and in the case of the electrical tension was 15000. In order for an attribute to be relevant to the learning method, it needs to present some kind of variation in values. When this is not the case, the usefulness of the attributes is rendered null.

The final dataset was thus composed by the electrical current and a temporal dimension added to each record. This was done in the transformation module by applying the time window procedure. Table 4.2 shows an example of a dataset with a temporal dimension of two extra hours.

Table 4.2: EDP dataset showing consumption values for 3 consecutive hours(time window 3

Current_0	Current_1	Current_2
318	316	320
320	308	274
320	308	274
308	274	308
...	...	...

## 4.4 Datasets transformation module

With the conclusion of the dataset collecting and cleaning module, the datasets transformation module is initiated. This module is responsible for creating the different combinations of train and test sets. Like its data gathering counterpart, these group of manipulation tasks are accomplished through a dedicated script. The script was later converted to an API, so that other researchers can apply the transformation approach to their problems. The API enables researchers to enrich their numerical datasets with extra temporal factors. Although, being mainly developed for time-series, the API can also be used on datasets which contain only numeric attributes. The API is published and publicly available through GitHub at `Dataset transformation API:URL`.

The functions available to users were divided into several different categories, contributing to a simple and efficient work environment. These categories are: Grid search functions, file configuration functions, single dataset creation functions, configuration functions and specific task functions.

The grid search functions, as seen in Figure 4.19, are only two. These functions present the highest-level abstraction for users. The `ds_config()` function enables users to create a group of different datasets, by just defining the objective class and the highest value of time-window and granularity their datasets will possess. The function then creates combinations of datasets from 1 to the maximum value of granularity and time-window. The second function, `ds_creator_future()`, is very similar to `ds_config()`. In both cases, the objective class and the maximum values for granularity are defined. The main difference relies in the future scope of prediction, i.e., the users define a specific prediction scope instead of the next hour, as done by `ds_config()`.

The above described functions use most of the functions found in the other categories. This was conceived in order to enable users a simple and efficient start.

However, a user might not want to create all train and tests possible through grid search. For these situations, a set of functions was created in which a configuration file can be used to only create the train and test sets with parameters specified in the configuration file. Figure 4.20 depicts all the functions available to use in conjunction with the configuration file.

The `ds_config_file` and `ds_config_file_future_exactly` functions work identically to the grid search functions `ds_config()` and `ds_creator_future()`, respectively. The main difference

## API functions

The API comes with a different set of functions that lets the user models its dataset to its needs.

### Grid search functions

```
#Create multiple datasets with all possible configurations until max granularity and history
def ds_config(maxGranularity,maxHourHistory,objectiveID):
```

```
#Create multiple datasets with all possible configurations until max granularity and history with a range of specific
def ds_creator_future(granularity,hourHistory,usableAttributes,norm,originalDataset,monthNum,objectiveID,name,futureT
```

Figure 4.19: Data transformation API: grid search functions overview

### File configuration functions

```
#Use configuration to create sets of training and test sets
def ds_config_file(objectiveID,addMonthBool):
```

```
#Use configuration to create sets of training and test sets with objective class of a specific hour
def ds_config_file_future_exactly(objectiveID,addMon,non_hour):
```

```
##Use configuration to create sets of training and test sets with objective class consisting of a range of specific
def ds_config_file_futureCombinations(objectiveID,addMon,non_hour):
```

Figure 4.20: Data transformation API: functions that integrate the configuration file

is that these two functions only create datasets with the specific configuration defined in the configuration file, instead of creating all possible combinations until reaching the maximum value. The *ds\_config\_file\_futureCombinations* function adds an extra prediction layer in comparison to *ds\_creator\_future()*. It creates datasets with multiple prediction scopes, i.e., instead of just having on specific future hour in the prediction scope, it is possible to define a range of multiple hours ahead to predict.

In situations where a user wants to create just a single dataset file, a set of functions were created. Figure 4.21 presents an overview of the single dataset creation functions.

The first function available to the user is *ds\_creator()*. This function creates a single dataset according to the users specifications. The values available for configuration are granularity, time-window, the attributes to be used and discarded, normalization, and additional datasets, which will work as source to create a dataset containing the aggregate data of all datasets. The *future\_ds()* function works similarly to *ds\_creator()*. The only difference is in the prediction scope. While *ds\_creator()* has a prediction scope of the next hour, the *future\_ds()* function lets users specify which prediction scope they want.

The *add\_months()* function, provides users with the possibility to use multiple datasets to create a single aggregate dataset. This function does not alter the data from the datasets; it merely gathers, orders and aggregates the data into a single dataset.

The *granularity\_indexer()*, *history()* and *normalization()* functions work as dataset manipulators. In cases where users just need to apply a single transformation to their

### Create single dataset

```
#Create a single dataset with specific configuration
def ds_creator(granularity, hourHistory, usableAttributes, norm, originalDataset, monthNum, objectiveID, name):

#Create a single dataset composed by N months
def add_months(allDS, monthNumber):

#Create a single dataset composed by N months and a future hour for objective class
def future_ds(allDS, futureTimeWindow, objectiveID, history):

#Tranform dataset into N granularity
def granularity_indexer(tmpDS, granu):

#Normalize dataset using min max
def normalization(allDS):

#Delete specific attrirbutes
def delete_metrics(metrics, tempDS):

#Create history for records
def history(hourHistory, allDS):
```

Figure 4.21: Data transformation API: create single dataset file

dataset, they can use these functions. The functions *granularity\_indexer()* and *history()* allows users to define which granularity and history/time-window they would like to have in their original dataset. The integrity of the original information contained in the dataset is guaranteed in the process. The *normalization()* function applies a normalization process to the source dataset using min max.

Lastly, the *delete\_metrics()* function, as the name suggests, removes the attributes specified by the user from the dataset; the integrity of the data is also guaranteed.

In occasions where users only require using the grid search functions, but the current configuration does not suit their problems needs, a set of configuration functions are available to adjust the grid search approach. These functions do not interact in any kind with the dataset; their interaction is solely with the grid search functions.

The last category of functions are specific task functions. These functions, similarly to the configuration category, do not interact with the dataset. Their main objective is to support all the transformation functions. Their application scope ranges from reading files to write a *.CSV* file of the created dataset.

The overview of the data transformation API serves merely as an introduction to its capabilities. A more detailed presentation of the generic applicability of the API along with the description of the functionality of each function is found in the GitHub directories above mentioned. All the functions available in the dataset transformation API were created in order to automatize the transformation process presented in Figure 4.2. This process is shared by the server load prediction and energy load forecast case studies. With the creation of suitable train and test sets, the model creation and evaluation phase is initiated. This phase will be presented and explained in detail in the next chapter.

## 4.5 Conclusion

This chapter presented a general approach to the load forecasting problems, followed by a description and adaptation to the server load prediction and energy load forecast problem.

As a consequence of the literature review, regarding the existence and availability of Wikipedia dataset gathering and general dataset temporal manipulation utensils. It was concluded that such tools were not available, thus both approaches, developed in this work were converted and published as APIs.

This chapter presented all the steps taken from the initial step of data gathering and analyses all the way to creating a ready to use train and test set. Next chapter presents the tests and results, which include the model creation and evaluation processes.

*This page has been intentionally left blank.*



# Chapter 5

## Server load prediction: experimental setup and tests

In the server load prediction chapter, the experimental setup, test runs and the results of the experiments are described. On an initial phase, test environment along with the selected evaluation metrics, are introduced; the information presented is essential for understanding the remaining of the chapter. Following the definition of test environment and metrics, comes the experimental run on the server load prediction case study. In this section the test runs, and results are analysed. Subsequently, a data enhanced version of the dataset is tested and evaluated. Lastly, a conclusion is presented containing a synthesis of the most relevant information of this chapter.

### 5.1 Test environment

In the initial setup of the test environment, a set of data preparation and analysis steps were required to produce an initial usable dataset. These steps involved the gathering of data, exploration of the initial raw information, analysis and cleaning of the created datasets and, lastly, the final preparation of the dataset. All these steps were taken and automated by dedicated scripts, which were later converted to APIs and which are described in the previous chapter.

The resulting dataset had two different variations amongst which the transformation process was built on. These variations were a basic/standard one, containing the most basic attributes and an enriched/enhanced one, in which the attributes were added to increase the models performances. Figure 5.1 presents an example of Wikipedia’s datasets variations.

#### Baseline scenario dataset

sum_requests1	sum_size1
10987514	307499141981
11380047	322211589360
10904549	307850796262
10745933	304349241813
10906089	307135912627

#### Enriched dataset scenario

sum_requests1	sum_size1	avg_requests1	avg_size1	Weekend_stdev_requests1	Weekend_stdev_size1	Population_stdev_requests1	Population_stdev_size1	language_occur1
10987514	307499141981	5.560772652	155625.0867	778.8128	20101379	3894.062908	100506869.1	1975897
11380047	322211589360	5.242195817	148426.1221	767.1601	19957007	3835.799855	99785013.08	2170855
10904549	307850796262	5.357068675	151237.6034	778.9079	20359599	3894.538371	101797971.6	2035544
10745933	304349241813	5.420641038	153524.8722	785.9689	20498673	3929.843453	102493337.9	1982410
10906089	307135912627	5.466135895	153936.6346	803.3273	20924495	4016.635384	104622449.5	1995210

Figure 5.1: Example of Wikipedia’s dataset variations

Each dataset variation was transformed in accordance to a set of different test configurations, e.g., granularity and time window variations. The resulting different configurations were also created through the developed API. Table 5.1 shows all the different datasets configuration attributes that were used.

Table 5.1: Dataset configuration attributes with respective values

Attribute	Value	Description
Number of months	1 to 8	Defines the number of months to aggregate into a single dataset
Granularity	1, 8, 24	Defines the level of detail (in hours) each record will contain
Time window	1, 4, 8, 12, 24, 48, 168	Defines the number of hours back in time a record will contain
Normalization flag	0 or 1	Flag that triggers the application of normalization to the data
Add months flag	0 or 1	Flag that dictates whether or not to aggregate multiple months into a single dataset

The transformation process originated a total of 672 different datasets, per dataset variation, to be used in the evaluation process, i.e., from the initial broader scope of possible dataset configurations (partial-grid search approach described in Chapter 4), a smaller, more representative group, was selected to proceed to the modelling phase. The modelling phase was an investigative and experimental process in which the different datasets were used to create predictive models through linear regression, ANN and SVM. These models were designed to predict the next hour of the incoming load with a split ratio of 70/30, i.e., 70% of the data was used for training the model, while 30% was used for testing, or 10 folds cross-validation. For the split ratio method, the records order remained unaltered after the splitting.

For the parametrization of each learning method, the objective was to achieve the best results from an average point of view. After some preliminary experiments, the following parameterizations were chosen as standard model configurations.

The linear regression model uses the Akaike criterion [233] for model selection, with the ability to deal with weighted instances. The M5 selection method was also adopted. This method steps through the attributes removing the one with the smallest standardised coefficient until no improvement is observed in the estimate of the error given by the Akaike information criterion. For the ridge parameter a value of  $1.0e-8$  was chosen.

The ANN model was created with a Multilayer Perceptron architecture and uses the backpropagation learning algorithm. The number of neurons was defined dynamically since our dataset varies in the total number of attributes according to the used variant. The hidden layers were defined by the formula:

$$\text{Hidden layer neurons} = ((NA + NON))/2, \quad (5.1)$$

where  $NA$  is the number of input attributes and  $NON$  is the number of output neurons.

A linear output unit with no threshold was also used. For the learning rate, the value 0.3 was selected. This learning rate is a proportionality constant that defines the amount of change of the weights. The use of 0.3 prevented the network from diverging from the target output, as well as improved the general performance. A training time of 500 iterations was defined. To conclude, a momentum of 0.2 for weights updating was applied during the learning process.

Lastly, the SVM model implemented was based on the *SMOreg* variation. This SVM variation implements a support vector machine for regressions. The *RegOptimizer*, which is the learning algorithm, was defined as *RegSMOImproved* with an  $e$  of  $1.0e-12$  and tolerance of  $0.001$ . The  $C$  parameter, which defines a margin of tolerance where no penalty is given to errors, was set to  $1$ . With a high parameter value, SVM tries to choose a smaller-margin hyperplane with fewer support vectors. On the contrary, a very small value of  $C$  causes the SVM to look for a larger-margin separating hyperplane, even if that hyperplane classifies more points. The value of  $1$  offered the best compromise. For the SVM kernel, the Polynomial kernel was used, since it is mostly used in non-linear modelling, which fits our type of problem.

### 5.1.1 Evaluation metric definition

The metrics through which the models were evaluated, were selected from the metrics group presented in Section 2.1.3. The decision were based on the information, advantages and short-comings that these metrics would carry when adopted to our case studies. The selection of metrics is identical in server load prediction and energy load forecast.

The first metric analysed was ME, represented in Equation 2.5. This metric calculates the mean error in the same scale as the problem. Even though this metric seems initially promising, on a deeper analysis it is possible to verify that the output would probably be misleading when used in our problems. The metric does not square or uses the absolute value/modulus, which can lead to positive and negative errors cancelling each other out, producing a much lower and deceptive error.

The next metric is depicted in Equation 2.6, MSE. This metric does not present any problem with errors cancelling each other out, since the errors are squared. However, this produces a value that is not on the same scale of the problem. Additionally, the squaring of errors amplifies the impact of outliers.

RMSE, presented in Equation 2.8 is frequently mentioned alongside of MSE. The two equations work in a very similar way; the main difference between RMSE and MSE is that, with RMSE on the final squared error, the square root is taken. This brings the final value of RMSE back to the same scale as the problem. The impact of outliers is however preserved since the root is taken just in the final overall error.

Following RMSE, the metric MAE, described in Equation 2.7 is analysed. MAE works as MSE, but contrary to this one, it is not sensitive to outliers since it does not square its errors and uses absolute value/modulus instead. This also guarantees that the final value is on the same scale as the error.

The last metric of absolute errors analysed is RSE, presented in Equation 2.9. RSE works similarly to RMSE; the difference between these two metrics is that the first is used for known values and the other for observed values. In short, known values, are values that are absolute and do not have any error of measurement associated to them. Observed values, as the name suggests were taken through a measuring process, which can have an error of measurement associated. The "-1" value found in the RSE equation is conventionally adopted to even out these errors of measurement.

In the case of relative errors/percentage errors, MPE, MAPE and the R-Squared variations were selected for analysis. MPE defined in Equation 2.11, calculates the ratio between the target value and the result of the forecast at a specific period in time in percentual form. MPE presents the same core short-coming as ME, the calculation of the error does not use the modulus, which as with ME, creates a problem of positive and negative errors cancelling each other out.

The MAPE function presented in Equation 2.12, tackles this short-coming. Similar to MPE, MAPE calculates the ratio between the target value and the result of the forecast at a specific period in time in percentual form. However, the target and the forecasted values are in modulus which prevents the cancellation of errors.

The last metrics are the two variations of R-Squared:  $R^2$  and  $\bar{R}^2$ .  $R^2$  shows the percentage value that characterises the variations in the objective, which can be explained through the predictors/input variables. This metric has a big short-coming, hence why, the  $\bar{R}^2$  was created. The  $R^2$  can increase with the addition of predictors, i.e., although  $R^2$  increases with the addition of extra data, which induces in believing that the performance of the model is increasing. The truth might be that the model is simply just classifying noise,  $R^2$  cannot differentiate between useful information and noise.  $\bar{R}^2$  tackles exactly this problem. It is a modified version of  $R^2$  that takes into consideration the predictors/input variables.  $\bar{R}^2$  increases only if the predictors that were added to the model increase the overall solution, otherwise it decreases.

From the above-mentioned metric analysis, the metrics MAE, RMSE, MAPE,  $R^2$  and  $\bar{R}^2$  were chosen. MAE and RMSE were chosen in conjunction to analyse the error in the same scale/units as the problem, along with the impact of outliers in data. MAPE was chosen for its simple, yet efficient error representation, along with the possibility to compare models from different problems. The metrics  $R^2$  in addition with  $\bar{R}^2$  were used to identify the percentage of overfitting present in the models. This is done by subtracting  $\bar{R}^2$  to  $R^2$ , as  $R^2$  indicates the percentage of all input variables that are able to define the variations in the objective, while  $\bar{R}^2$  only indicates the percentage of variables that define the variations in the objective while at the same time increasing the performance of the model.

### 5.1.2 Normalization vs Non-normalization

The normalization of data is a frequent process used in machine learning. When normalizing data, the objective is to remove units and/or rescale all variables to the same scale. Usually the scale used is between 0 and 1. Nevertheless, when speaking of normalization there are various different types of normalization. The most frequent ones are presented in Table 5.2 [234].

From the aforementioned types, the one that was most suitable to the problems studied in this work was the Min-Max Feature scaling. The function which defines this type of normalization is presented in the equation below:

$$X' = a + \frac{(X - X_{min})(a - b)}{X_{max} - X_{min}}, \quad (5.2)$$

Table 5.2: Different type of data normalization

Name	Description
Standard core	Used for normalizing errors when the mean and standard deviation are known.
Studentized residual	Used for normalizing errors when the estimation/observation of values was used.
Standardized moment	Used for normalizing values in a specific range by using the standard deviation as a measure of scale.
Student's t-statistic	Used for normalizing errors when the mean and standard deviation are unknown.
Coefficient of variation	Used for normalizing dispersion in data, uses the mean as a measure of scale.
Min-Max Feature scaling	Feature scaling is used to reformat all values between a certain interval [a,b], the most common interval used is between [0,1].

where  $X_{min}$  and  $X_{max}$  are the minimum and maximum values of the specified dataset and  $a$  and  $b$  are the scale interval of the normalization. The minimum and maximum values used in this work were chosen from the complete dataset, i.e., occasionally, researchers might opt to choose the minimum and maximum just from a certain data spectrum such as, only the train set. We, however, opted for using the complete spectrum of data. For the scale interval the values 0 and 1 were chosen.

The decision to use the Min-Max Feature scaling was based on the desire to analyse the impact of bigger attributes vs smaller attributes in the performance of the models.

Figure 5.2 presents an example of data regarding the basic Wikipedia dataset before and after the application of normalization.


sum_requests	sum_size		sum_requests	sum_size
15663475	475790705958.00	<b>Normalization</b> 	1	1
14418496	436015729975.00		0.807668	0.827412
12852936	393967055292.00		0.56581	0.644959
12216833	371907750512.00		0.467541	0.549241
12302409	370147331980.00		0.480761	0.541603
12710822	386828749023.00		0.543855	0.613985

Figure 5.2: Normalization example: before and after data

The analysis of the impact of normalization was performed on all tests. It was possible to observe an insensitivity pattern towards normalization, i.e., the performance of the models was not affected by normalization, being the end result the same as when normalization was not applied. Nevertheless, this hypothesis was just confirmed at the end of all test runs, hence why all tests runs were performed with and without normalization. Due to the results not differing with and without normalization, it was opted to only present below the results without normalization. A deeper analysis of these results is available through the published papers.

## 5.2 Server Load Prediction on basic dataset with cross-validation

The first test scenario presented was conducted using the basic Wikipedia dataset with a split ratio of 70/30 and 10 folds cross-validation, as mentioned above. Since results

between the two approaches were very similar it was opted to only present the cross-validation results and analysis (and append the split ratio ones) as cross-validation presents statistically stronger results. The analysis of results was split between quantity of data (single month data or whole year data) and learning model.

The use of cross-validation is usually done to validate the results from a statistical point of view and/or to increase the amount of data available through redundancy and thus fortify the statistical validity of the results.

Similar to split ratio, when creating and evaluating a model, training and test sets are used. In the split approach a single dataset is divided in two, a percentage of the dataset is used for training and another is used for testing, in our case 70%/30%. This approach however can produce non-representative results when the data set is small, i.e., the quantity of data available for training and testing is too small for representing the real diversity of the problem. The resulting model may present low error values, these values however might be deceiving since the variations in the train and test data were possibly minimal; the model did not learn the pattern and merely memorized the small sample of data available.

In cross-validation, even though the amount of data might be reduced, through the use of folds this problem is smoothen out. In this process, the data is divided in  $k$  folds - the most common value used is 10 folds - each fold is then used one time for testing and  $k$  times for training.

For instance, a dataset with 300 instances that uses cross-validation with 3 folds, will be split into three parts. Three different models will be built, each model is trained on two parts and tested on a third. The first model is trained on part one and two and tested on part three. The second model is trained on part one and part three and tested on part two and so on. When all models are built and tested, the average results of these models will give an overall better understanding of the models performance. The application of cross-validation enables the model to fortify the validity of its results with lower quantity of data.

In WEKA the cross-validation approach is a little different. For example, in cross-validation of 10 folds, WEKA invokes the learning method 11 times, once for each fold and a final time on the entire dataset. The final time creates the final output model.

The cross-validation applied in the test cases was stratified, i.e., each fold had its data rearranged so that each fold contained enough information to be a good representative of the overall dataset/problem.

### **5.2.1 Test runs and analysis for multiple months dataset**

The months used for the multiple months scenario were from January 2016 to August 2016. At the time of the study these datasets were the most recent available through the Wikimedia foundation. As for the test setup, the months were tested individually not sharing any information between them. The configurations used were 1, 8 and 24 for granularity, 1, 4, 8, 12, 24, 48 and 168 for time window/history and normalization and non-normalization of data.

## Linear Regression

Table 5.3: Result synthesis for all months using linear regression, basic dataset and 10 folds cross-validation

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
January	0.9404	0.0296	1.84 ± 1.51	1.12	14.29	70.02 ± 55.79	3.79	1176.98	89.64 ± 59.16	5.13	1454.25
February	0.9464	0.0300	1.79 ± 1.61	1.19	14.28	64.39 ± 51.19	4.20	1189.01	82.42 ± 54.30	6.72	1474.54
March	0.9550	0.0139	2.02 ± 1.46	1.10	14.41	66.49 ± 48.02	3.50	1005.66	82.14 ± 50.52	4.54	1152.11
April	0.8862	0.0673	2.13 ± 1.86	1.21	14.64	74.64 ± 65.52	3.82	1134.41	99.75 ± 70.25	5.36	1383.89
May	0.9441	0.0261	1.96 ± 1.60	1.37	13.82	61.60 ± 48.59	4.30	1031.26	78.51 ± 51.46	5.80	1306.28
June	0.8880	0.0597	1.99 ± 1.72	1.13	13.83	69.74 ± 59.32	3.53	1059.06	91.69 ± 63.28	4.89	1341.73
July	0.8328	0.0355	2.83 ± 2.29	1.51	15.68	86.68 ± 66.54	4.47	788.25	109.49 ± 70.37	6.14	990.08
August	0.9360	0.0330	2.13 ± 1.53	1.51	13.47	18.93 ± 12.65	4.42	288.56	22.80 ± 13.24	5.44	284.31

As can be seen in Table 5.3,  $\bar{R}^2$  varied between 83.3% and 98.8%, while overfitting fluctuated between 1,4% and 6%. The best models had a MAPE between 1.11% and 1.51%, which translates to around 3.50 Gigabytes and 4.47 Gigabytes. Lastly, when analysing MAE and RMSE, it is possible to verify, through the small difference between these values, that outliers had minimal effect the models performances.

Having analysed the synthesis of the test run results, an average analysis with regards to granularity and time window is also performed, as seen in Figure 5.3.

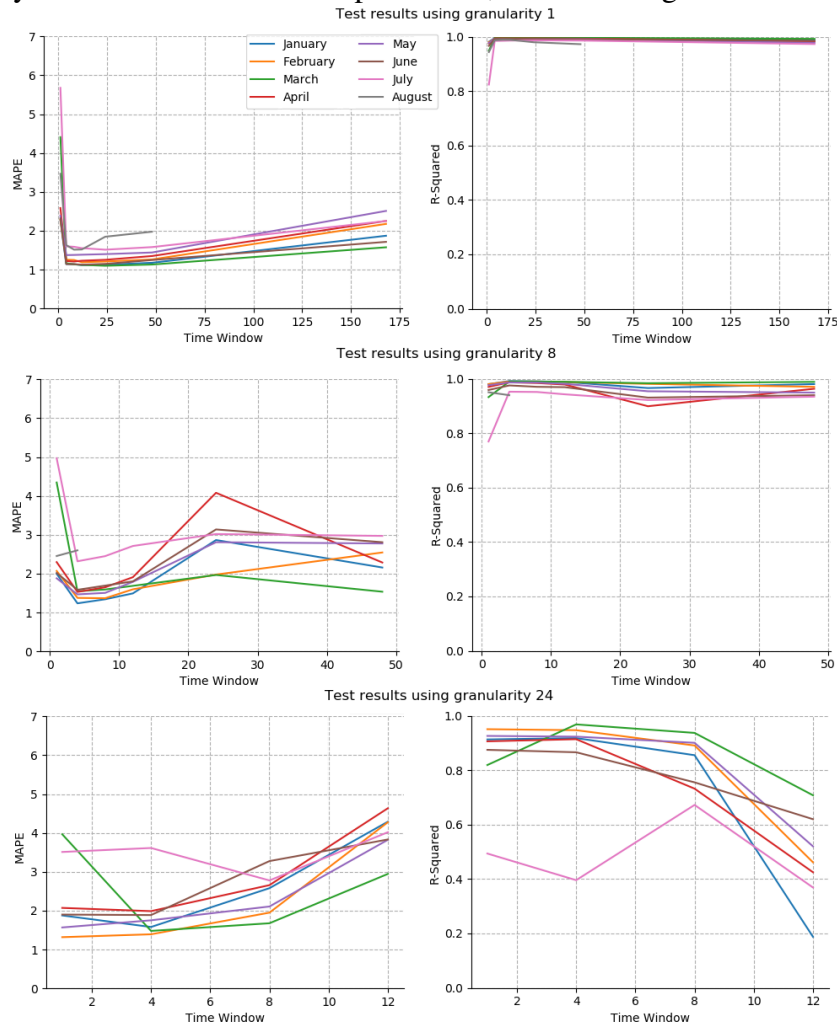


Figure 5.3: Influence of granularity and time window in MAPE and R-Squared for linear regression with 10 folds cross-validation

The use of granularity 1 achieved its lowest errors between 4 and 48 time window; after that, an increase in time window lead to worse models. The variations of  $R^2$  were accompanied by the MAPE variations. This trend was also verified up to granularity 8 and 24. When using a granularity of 8, the models performed better between the 4 and 12 mark in all months. The months of April and May which have better performance for a time window of 24 when using split ratio, had also their overall best results in the same range. Finally, for granularity of 24, the models performed well at time window 4. An increase in granularity seemed to increase the overall error, as well as to reduce the beneficial effects of the time window.

### Artificial neural networks

Table 5.4: Result synthesis for all months using ANN, basic dataset and 10 folds cross-validation

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
January	0.9051	0.0485	2.10 ± 1.65	1.22	12.93	66.97 ± 49.41	4.37	974.62	83.46 ± 52.12	5.72	1102.84
February	0.9017	0.0555	2.07 ± 1.76	1.29	13.42	66.83 ± 52.05	4.54	838.92	85.09 ± 55.22	6.72	1043.45
March	0.9191	0.0342	2.34 ± 1.72	1.24	15.39	71.35 ± 51.17	3.93	964.50	87.94 ± 53.81	4.94	1185.12
April	0.8134	0.1073	2.79 ± 2.38	1.38	18.90	77.66 ± 67.32	4.39	468.16	103.06 ± 72.00	6.05	606.37
May	0.8586	0.0789	2.40 ± 1.87	1.47	13.40	74.35 ± 56.35	4.57	998.57	93.38 ± 59.48	6.08	1230.55
June	0.8496	0.0668	2.47 ± 2.05	1.24	14.39	87.00 ± 71.86	3.90	1087.65	113.03 ± 76.46	5.38	1381.73
July	0.8631	0.0382	3.32 ± 2.76	1.88	16.40	100.28 ± 81.90	5.49	907.84	129.53 ± 86.98	7.20	1201.10
August	0.9463	0.0219	2.19 ± 1.62	1.62	13.82	18.11 ± 12.42	4.69	245.65	22.16 ± 13.09	5.98	273.70

As shown in the table, the  $\bar{R}^2$  fluctuated between 81.3% and 94.6%, a result slightly worse than the one achieved with linear regression. The percentage of overfitting increased, as well, ranging amid 3.4% and 10.7%. In terms of best models, results varied between 1.22% and 1.88% slightly worse than linear regression. When analysing the results in terms of homogeneous data, the impact of outliers was higher when compared to linear regression.

Subsequent to this analysis the variations of MAPE and  $R^2$  was also performed, Figure 5.4 illustrates the variations.

As demonstrated in Figure 5.4, with the use of granularity 1, a time window between 4 and 8 is the most suited. beyond that, the MAPE increases gradually with an increase in time window. In April, an increase in error can be verified beyond the 48 hour mark. The  $R^2$  value accompanied the MAPE variations in every instance, as can be verified in April, where the  $R^2$  peak occurs at the same time as the MAPE peak.

For granularity of 8, an increase in time window reflected an increase in MAPE. The best results occurred at time window 4. Beyond the 24 hour mark, however, MAPE values decreased with the increase in time window. For the month of March this trend appears at around time window 8. The  $R^2$  accompanied the MAPE variations although with a smaller magnitude, i.e.,  $R^2$  followed the MAPE trend, but the magnitude in which this happens is much smaller than for other granularities.

Finally, for granularity 24, the results are more chaotic. In this granularity and with ANN there was no clear optimal time window for the majority of the months, as can be verified by comparing January and May. For January, the lowest value was found with time window 8, while for May with a time window 8 was were the highest MAPE was achieved.  $R^2$ , similarly to the other granularity values, accompanied the MAPE variations.



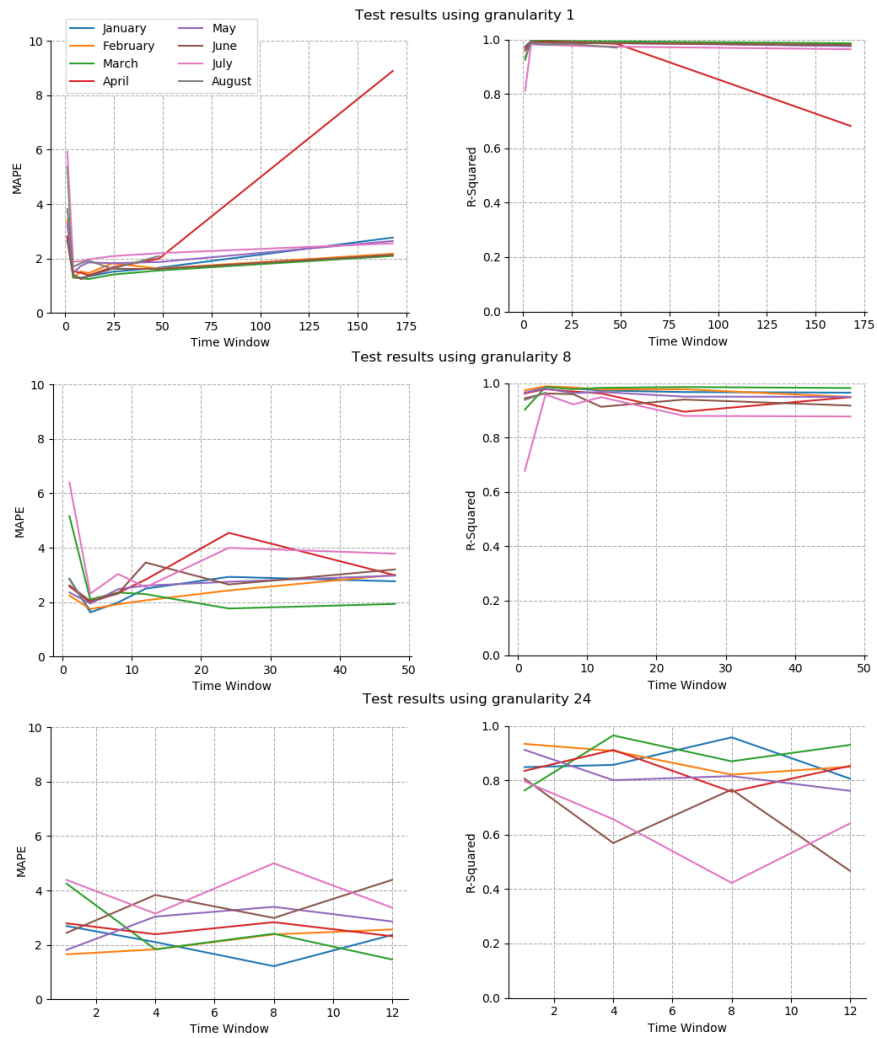


Figure 5.4: Influence of granularity and time window in MAPE and R-Squared for ANN with 10 folds cross-validation

To conclude, ANN presents the worst results among all learning methods. However, on months with outliers, through the use of cross-validation, ANN performed better than linear regression and SVM. In terms of granularity, an increase in granularity represents worse results from an average point of view. For time window, the beneficial effect on the model is achieved with lower time window, for a higher granularity value.

### Support vector machine

As seen in Table 5.5, when applying SVM, the majority of months decreased their minimum MAPE when compared to linear regression and ANN. In terms of  $\bar{R}^2$ , the results varied between 85.75% and 95.33%, while for overfitting, results varied between 1.53% and 7.74%. With regards to the best models, all months were able to achieve MAPE values under the 1.51% mark. The best models also presented a good performance in terms of outliers acceptance, i.e., the difference between MAE and RMSE was significantly low.

Table 5.5: Result synthesis for all months using SVM, basic dataset and cross-validation 10 folds

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
January	0.8744	0.0774	1.70 ± 1.42	1.11	12.93	66.97 ± 49.41	4.37	974.62	83.46 ± 52.12	5.72	1102.84
February	0.9355	0.0354	1.63 ± 1.44	1.11	13.42	66.83 ± 52.05	4.54	838.92	85.09 ± 55.22	6.72	1043.45
March	0.9448	0.0227	1.90 ± 1.37	1.10	15.39	71.35 ± 51.17	3.93	964.50	87.94 ± 53.81	4.94	1185.12
April	0.8957	0.0591	1.90 ± 1.60	1.22	18.90	77.66 ± 67.32	4.39	468.16	103.06 ± 72.00	6.05	606.37
May	0.9007	0.0587	1.83 ± 1.49	1.37	13.40	74.35 ± 56.35	4.57	998.57	93.38 ± 59.48	6.08	1230.55
June	0.9003	0.0479	1.89 ± 1.68	1.13	14.39	87.00 ± 71.86	3.90	1087.65	113.03 ± 76.46	5.38	1381.73
July	0.8575	0.0265	2.77 ± 2.22	1.51	16.40	100.28 ± 81.90	5.49	907.84	129.53 ± 86.98	7.20	1201.10
August	0.9533	0.0156	1.98 ± 1.56	1.24	13.82	18.11 ± 12.42	4.69	245.65	22.16 ± 13.09	5.98	273.70

When analysing the impact of granularity and time window, a similar yet improved big picture of linear regression can be verified. Figure 5.5 illustrates the results achieved.

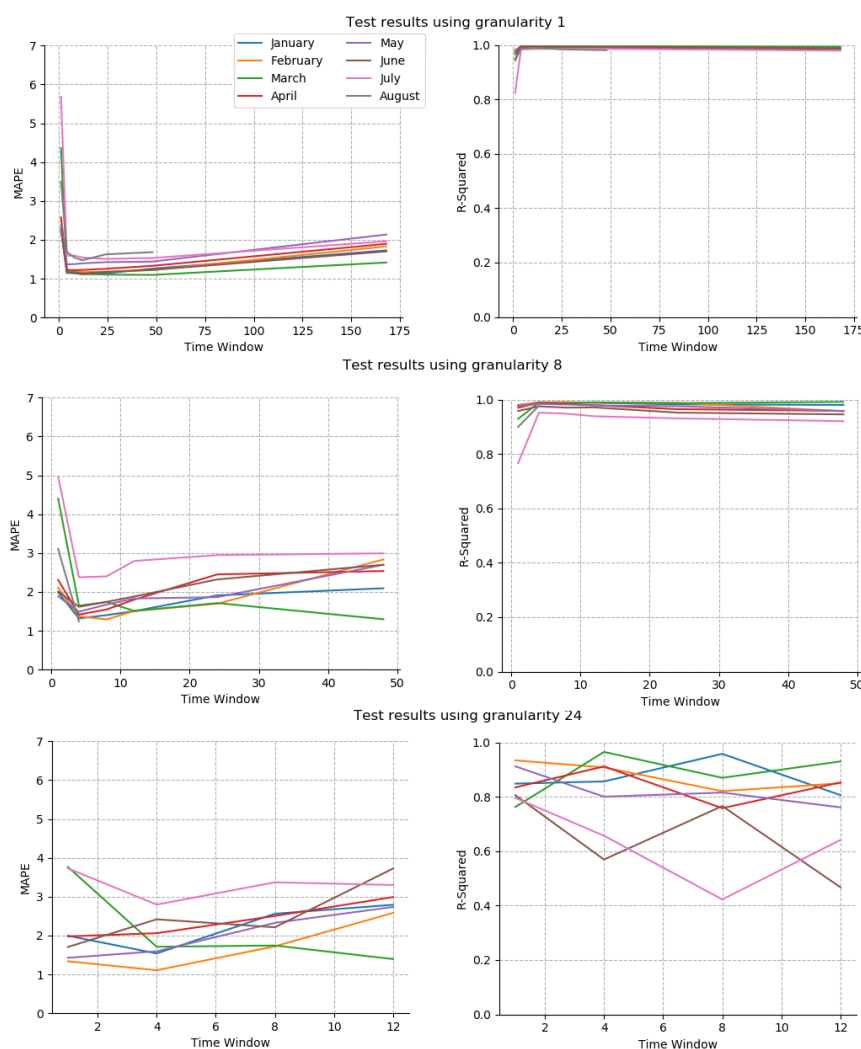


Figure 5.5: Influence of granularity and time window in MAPE and R-Squared for SVM with 10 folds cross-validation

The overall results show a uniform pattern between months. Although with the increase in granularity the fluctuation of MAPE values increase, its magnitude is still low. SVM

presented the overall best results in terms of MAPE and  $R^2$  variations. Peak values that occur with linear regression and ANN do not occur with SVM, for instance, the month of April with granularity 1 when using ANN or the month of April with granularity 8, when using linear regression. Across all months and with any configuration, there was no model that exceeded a MAPE of 4%.

For granularity 1, the created models achieved MAPE values between 1.10% and 2.20%. The best time windows lied between 4 and 24; beyond that, the error increases gradually with time window.  $R^2$  accompanied MAPE across all time windows, never falling below 0.96.

With a granularity of 8, the best time windows were between 4 and 8, increasing the MAPE value on most months beyond that. The  $R^2$  variations are also minor with different time windows in most months.

Last of all, for granularity 24, the results varied a lot more than with lower granularities: all months had higher MAPE variations, with the best time windows differing between months. Similar to split ratio, an increase in granularity, resulted in a shorter beneficial period for time window. The  $R^2$  values were the lowest, by a last margin, within this granularity. The MAPE values, however, did not seem to be affected by these results on the same scale, as a decrease in almost the double of  $R^2$  for July reflected only a minimal increase in the MAPE.

## **Model comparison**

The best models per learning method for each month are compiled in Table 5.6.

With cross-validation the best performing learning method was linear regression, all months had their best model with it. All metrics were taken into account when selecting the most adequate learning method.

For the month of January, linear regression was the best learning method, but only with a minimal advantage; it achieved a higher  $\bar{R}^2$  and handled better outliers, while only getting 0.01% MAPE above SVM.

In February the difference in values was more significant: SVM was an obvious exclusion since it achieved the worst results. SVM had higher model overfitting, lower  $\bar{R}^2$  and difficulties with outliers, these three shortcomings justified its exclusion, although having the lowest MAPE of all learning methods. As for ANN, the reason for its exclusion was its higher MAPE. ANN achieved very similar results to linear regression, even handling slightly better outliers; the higher MAPE value worked as tiebreaker.

Linear regression and SVM were the best learning methods for March, April, May, June and July in terms of MAPE. In these months, MAPE always tied, except for April, which had only a 0.01% difference. The decision for linear regression in detriment of SVM lied mainly in its ability to better handle outliers, a recurrent pattern in all months.

Lastly, for August, the best achieving learning model in terms of MAPE was SVM, however its poor performance in  $\bar{R}^2$ , overfitting and outliers handling led to the use of linear regression, even though it had MAPE 0.27% higher than SVM.

Table 5.6: Best models with respective time window and granularity configuration for Linear regression, ANN and SVM, using 10 folds cross-validation with enhanced dataset

Month	Learning Method	Granularity	Time Window	$\bar{R}^2$	Model Overfitting	MAPE	MAE	RMSE
January	<b>LR</b>	<b>1</b>	<b>12</b>	<b>0.9950</b>	<b>0.0002</b>	<b>1.12</b>	<b>3.79</b>	<b>5.15</b>
	ANN	24	8	0.8688	0.0895	1.22	100.86	131.41
	SVM	1	12	0.9949	0.0002	1.11	3.79	5.19
February	<b>LR</b>	<b>1</b>	<b>12</b>	<b>0.9926</b>	<b>0.0002</b>	<b>1.19</b>	<b>4.20</b>	<b>6.72</b>
	ANN	1	4	0.9928	0.0001	1.29	4.54	6.72
	SVM	24	4	0.9424	0.0168	1.11	94.70	128.18
March	<b>LR</b>	<b>1</b>	<b>24</b>	<b>0.9962</b>	<b>0.0002</b>	<b>1.10</b>	<b>3.50</b>	<b>4.54</b>
	ANN	1	12	0.9960	0.0001	1.24	3.93	4.95
	SVM	1	48	0.9957	0.0006	1.10	3.46	4.62
April	<b>LR</b>	<b>1</b>	<b>8</b>	<b>0.9942</b>	<b>0.0001</b>	<b>1.21</b>	<b>3.82</b>	<b>5.36</b>
	ANN	1	12	0.9925	0.0002	1.38	4.39	6.05
	SVM	1	8	0.9941	0.0001	1.22	3.86	5.42
May	<b>LR</b>	<b>1</b>	<b>4</b>	<b>0.9925</b>	<b>0.0001</b>	<b>1.37</b>	<b>4.30</b>	<b>5.80</b>
	ANN	1	4	0.9918	0.0001	1.47	4.57	6.08
	SVM	1	4	0.9925	0.0001	1.37	4.28	5.80
June	<b>LR</b>	<b>1</b>	<b>12</b>	<b>0.9925</b>	<b>0.0002</b>	<b>1.13</b>	<b>3.53</b>	<b>4.89</b>
	ANN	1	8	0.9911	0.0002	1.24	3.90	5.38
	SVM	1	12	0.9924	0.0002	1.13	3.57	4.94
July	<b>LR</b>	<b>1</b>	<b>24</b>	<b>0.9860</b>	<b>0.0009</b>	<b>1.51</b>	<b>4.47</b>	<b>6.14</b>
	ANN	1	4	0.9824	0.0002	1.88	5.49	7.20
	SVM	1	24	0.9858	0.0009	1.51	4.46	6.18
August	<b>LR</b>	<b>1</b>	<b>8</b>	<b>0.9870</b>	<b>0.0020</b>	<b>1.51</b>	<b>4.42</b>	<b>5.44</b>
	ANN	1	24	0.9619	0.0221	1.62	4.69	6.01
	SVM	8	4	0.9002	0.0776	1.24	28.66	42.22

Overall, in all months, except for February, linear regression and SVM achieved similar results. The decision for linear regression in detriment of SVM lied mostly in minor details. ANN was the worst performing learning method, with exception as mentioned above, in the month of February.

In terms of time window and granularity of the best models per learning method, a clear trend is visible. Figure 5.6 shows the time window and granularity distribution in these models.

With cross-validation the granularity values were vastly uniform, being predominately 1. Most models had either time window 4 or 12, occupying each time window 29.17% each of the total distribution.

In terms of results per learning model, a time window of 4 was the predominant value for ANN and SVM, while for linear regression time window 12 dominated. This outcome was somewhat unexpected, given the similarity of results between linear regression and SVM. Linear regression was the only learning method which only used granularity 1, which is an expected outcome taking into consideration the variations of MAPE and  $R^2$  that were shown.

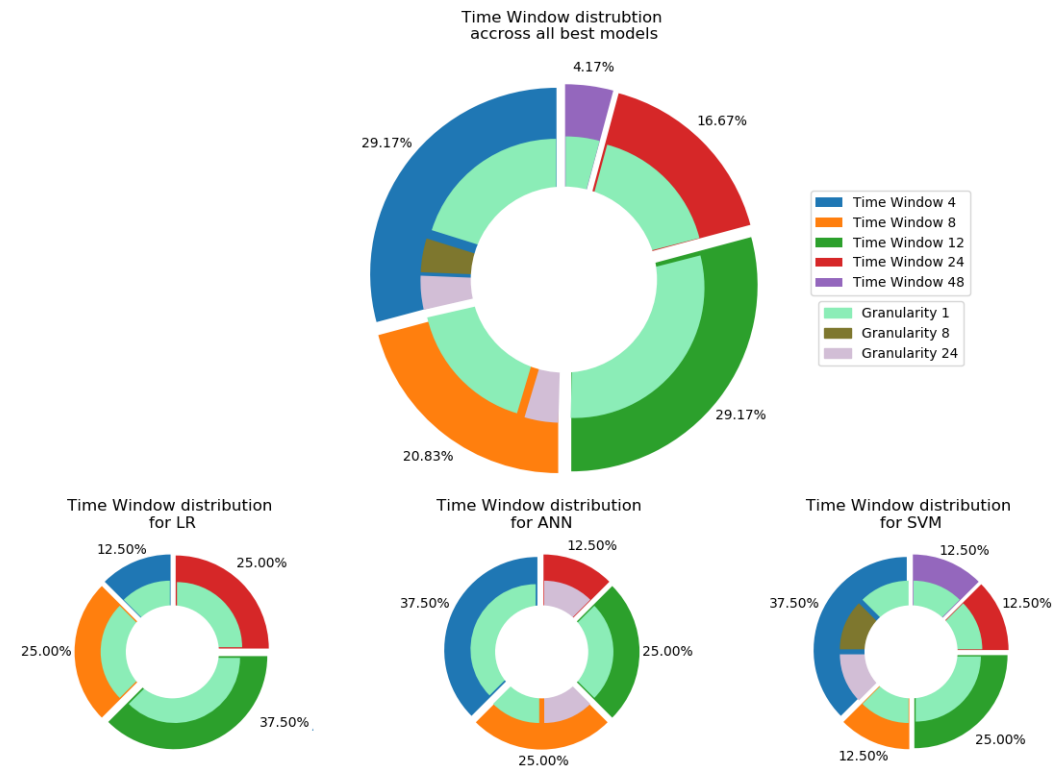


Figure 5.6: Time window distribution across all best models for single month with 10 folds cross-validation

All three learning models performed better with lower granularity and lower time window, a pattern that was visible through analysis of the MAPE and  $R^2$  distributions. A total of 75% of the best models were achieved with time windows smaller than 12 for linear regression and SVM, while for ANN this percentage rose to 87.5%. SVM is the only learning method that achieved good results with a time window of 48.

Lastly, a MAPE distribution analysis was made and presented in Figure 5.7.

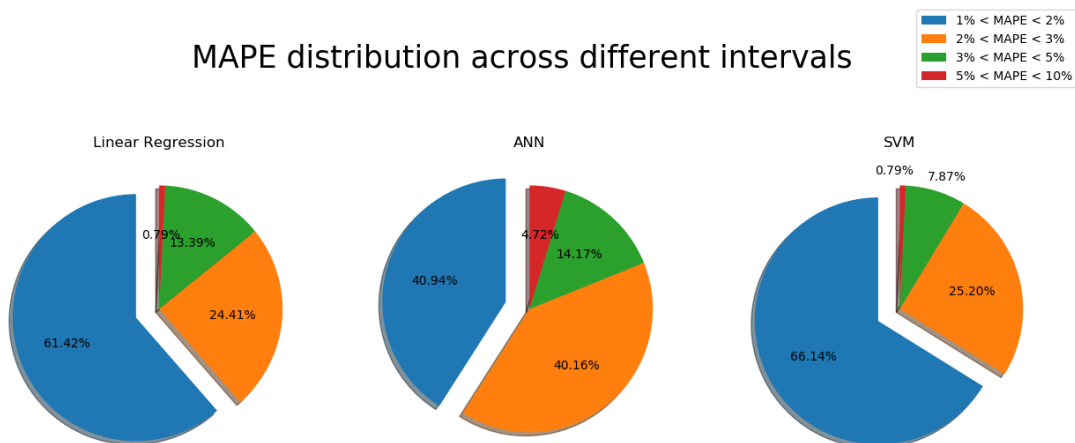


Figure 5.7: Time window distribution across all best models for single month with 10 folds cross-validation on basic dataset

Linear regression and SVM, as expected, presented very similar MAPE distributions. Their biggest difference lied between the MAPE range 3 and 5, which for linear regression occupied 13.39%, while for SVM occupied 7,87%. In terms of overall performance SVM was the best learning method, with 66.14% of the models achieving MAPE values between 1 and 2 percent, linear regression comes close second with 61.42%. ANN was the worst performing model, with 81.1% of the models having a MAPE between 1 and 3 percent, of which only 40,94% were between 1% and 2% MAPE.

## 5.2.2 Test runs and analysis for single year dataset

A synthesis of results of the single year dataset was created and shown in Table 5.7.

Table 5.7: Average and best results for single year dataset using 10 folds cross-validation

Result synthesis for single year dataset

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
LR	0.9250	0.0207	2.15 ± 2.12	1.27	12.85	90.49 ± 85.66	4.05	521.33	124.78 ± 92.30	5.88	717.62
ANN	0.8906	0.0288	2.71 ± 2.50	1.35	13.41	119.32 ± 108.14	4.30	560.67	161.30 ± 116.05	5.94	752.04
SVM	0.9258	0.0205	2.16 ± 2.15	1.27	12.83	91.52 ± 86.86	4.06	524.18	126.54 ± 93.72	5.87	741.85

Best models with respective configuration

Learning Method	Granularity	Time Window	$\bar{R}^2$	Model Overfitting	MAPE	MAE	RMSE
LR	1	48	0.99342	0.00012	1.27	4.05	5.88
ANN	1	12	0.99343	0.00003	1.35	4.30	5.94
SVM	1	24	0.99340	0.00006	1.27	4.08	5.92

In terms of average MAPE values, results varied between 2.15% and 2.71%. As for average  $\bar{R}^2$ , results of 92.5% and 92.6% for linear regression and SVM respectively and 89% for ANN were achieved. The overfitting in models on average never surpassed the 3% mark. The best models created by these learning methods had a MAPE of 1.27% for linear regression and SVM and 1.35% for ANN. The configurations used for these models were granularity 1 and time window 48, 12 and 24 for linear regression, ANN and SVM accordingly. With regards to the MAPE and  $R^2$  distribution, Figure 5.8 presents the overall variations of these metrics.

For granularity 1, an increase in MAPE for ANN was verified after time window 24, the best results were found between a time window of 8 or 24. As for linear regression and SVM, results were almost identical. The two graphs overlap each other, hence why only SVM is visible. The variations of values is minimal after a time window of 4. Nevertheless, the trend of these variations where in the direction of a smaller MAPE.  $R^2$  accompanied all variations of MAPE.

With a granularity of 8, linear regression and SVM again overlapped results. Their best performing time window were between 8 and 24, after that MAPE would gradually increase, never to surpass the 3% MAPE mark. For ANN the best performing time window was 4, after that MAPE gradually increased, stabilizing its value at time window 48 and

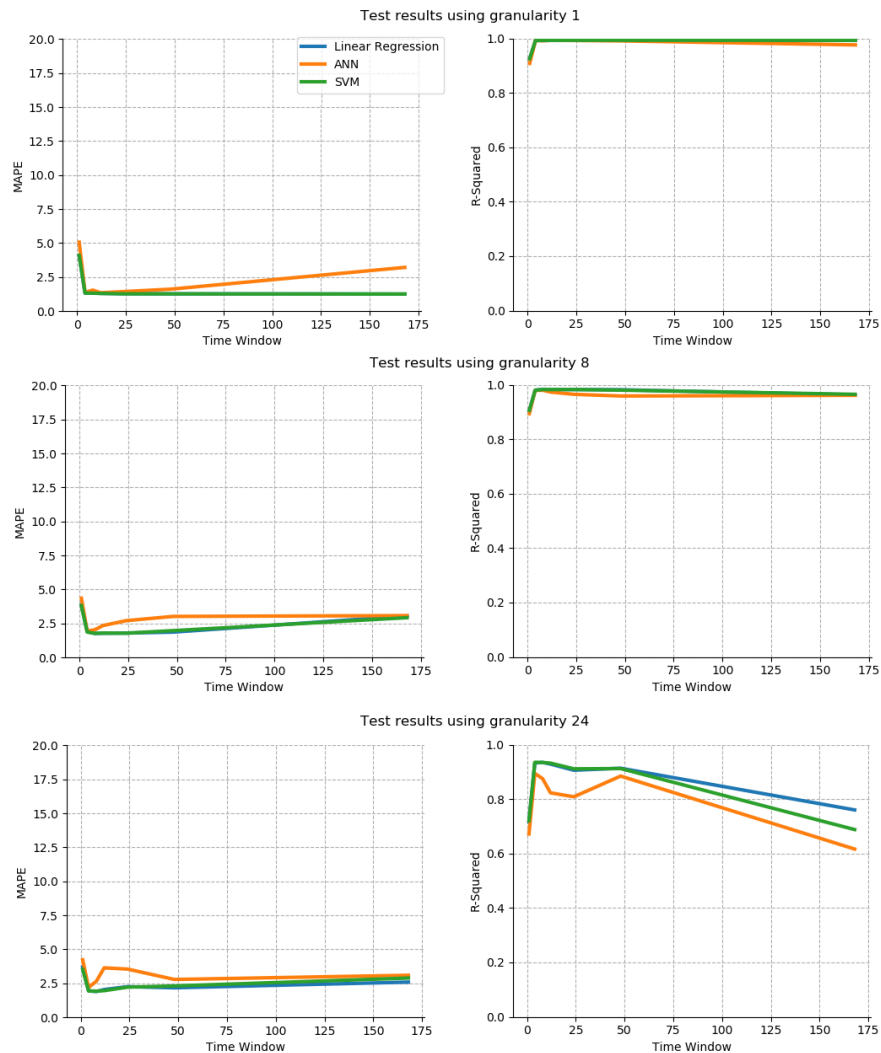


Figure 5.8: Time window distribution across all best models for single year with 10 folds cross-validation

maintaining its values in the vicinity of the MAPE achieved with time window 48.  $R^2$  again accompanied all variations of MAPE.

Finally, with a granularity of 24, linear regression and SVM again overlapped results but only until a time window 24, after that their results dispersed, with linear regression achieving slightly better results than SVM. Both learning methods achieved their best results with time window 4. For ANN the best performing time window was also 4, with results stabilizing after time window 48. All three learning methods achieved very similar outcomes after time window 48.

To conclude, the MAPE distribution was also analysed, as depicted in Figure 5.9.

From the presented results it is possible to verify that the best performing learning method from an overall perspective was SVM, while the worst performing method was ANN. ANN was also the only learning method that had models with errors between 5% and 10%.

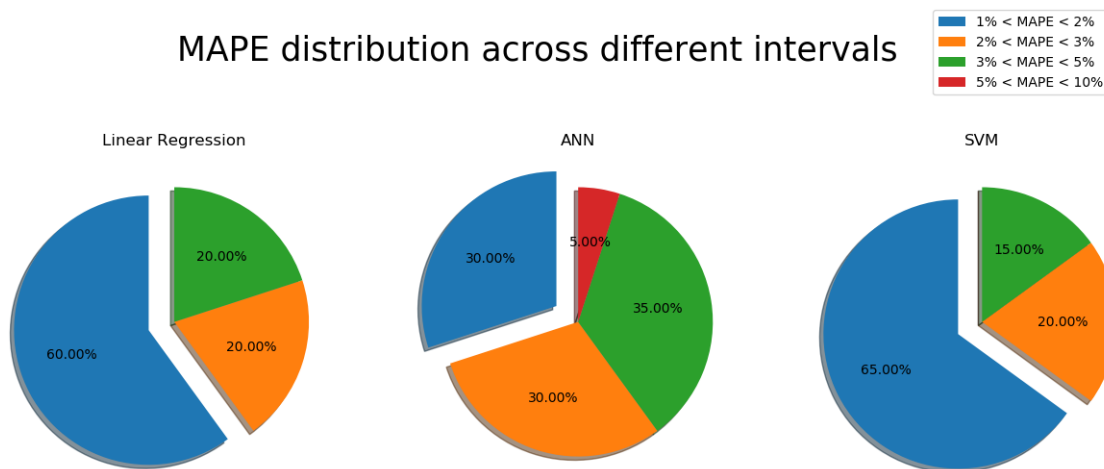


Figure 5.9: Time window distribution across all best models for single year with 10 folds cross-validation

## 5.3 Server Load Prediction on enhanced dataset with cross-validation

Equivalent to the basic dataset scenario, different configurations for time window and granularity were used. The testing and analysis procedures were overall similar to the basic scenario. The differences between these two test scenarios are the enhanced dataset used and an initial feature analysis of the dataset. This procedure was not compulsory for the basic scenario since the dataset only contained two attributes in its initial form.

### 5.3.1 Feature analysis

On an initial approach, different attributes along with different formats for the same attributes were tested. For instance, a date attribute was tested as string, numeric, time stamp, and split into four different attributes: hour, day, month and year. All these variations for date had a detrimental effect on the models performance (the focus was on MAPE values). Flag attributes such as indicators of week days or weekends were also tested, as well as, period of the day: morning, afternoon, evening, midday and midnight. All these attributes had a detrimental effect in the end result. These initial test results led to the exclusion of the said attributes. Tests runs showed that statistical attributes were the most efficient. A reason for this could lie in the creation process of the dataset. Since the dataset is an aggregate of English language pages, the non-statistical attributes may not perform so well in non-region locked data, i.e., non-statistical attributes such as time of the day focus on human activity patterns, since the records were aggregates of data from different parts of the world, human patterns would most likely not be transferred to the record, thus rendering these attributes inefficient to the learning methods.

From the testing phase emerged a final dataset, the enhanced dataset depicted in Figure 5.1. Nevertheless, in order to validate the attributes chosen from a statistical point of view,



a feature analysis was conducted. A correlation matrix between attributes was created, as seen in Figure 5.10.

	sum_requests1	avg_requests1	avg_size1	Weekend_stdev_requests1	Weekend_stdev_size1	Population_stdev_requests1	Population_stdev_size1	language_occur1
sum_requests1	1							
avg_requests1	0.76	1						
avg_size1	0.77	0.83	1					
Weekend_stdev_requests1	0.56	0.86	0.79	1				
Weekend_stdev_size1	0.58	0.85	0.82	0.78	1			
Population_stdev_requests1	0.56	0.86	0.79	0.81	0.78	1		
Population_stdev_size1	0.58	0.85	0.82	0.78	0.81	0.78	1	
language_occur1	0.78	0.18	0.24	0.03	0.07	0.03	0.07	1

Figure 5.10: Correlation matrix for Wikipedia’s enhanced dataset

All the features presented in the figure are positively correlated among themselves, this is, when a value of a variable increases or decrease, i.e., changes its direction, all correlated features change their values in the same direction. The impact of this on each attribute is usually influenced by the correlation coefficient between these features. When analysing features in terms of correlation, usually a high correlation, e.g., above 0.95, implies that the two attributes have more or less the same impact in the models performance, sharing the same information. In situations like these, the exclusion of one of the attributes or the fusion of the two attributes into one might be options. Nevertheless, the resulting model should always be tested before and after the changes, the information that these attributes do not share might be crucial for the models performance. On the contrary, a lower correlation coefficient indicates that the variables share very little information between them, which may indicate low redundancy of information for the learning method. Testing before and after is again a necessity, as the low correlation coefficient might just be a symptom from one of the attributes being noise to the learning method and thus having a detrimental effect in the models performance.

From the information presented in Figure 5.10 the correlation between attributes varies between 0.03 and 0.86, which indicates that low correlated attributes may be just noise to the learning method, while high correlated ones suggest an eventual redundancy in information. Test were conducted for each attribute in order two measure their overall impact in the models performance. The results allowed to conclude that removing any attribute would worsen the models performance.

### 5.3.2 Test runs and analysis for multiple months dataset

With the validation of the enhanced dataset, testing and analysis were carried out. The testing and evaluation methodologies, as well as the dataset configurations, are identical to the ones used with the basic dataset.

#### Linear Regression

From the results shown in Table 5.8, it is possible to confirm that the enhanced dataset increased the learning methods performances immensely. Minimum MAPE ranged mostly between 0.11% and 0.3%, while average MAPE was below 1% in the majority of months.

Table 5.8: Result synthesis for all months using linear regression, enhanced dataset and 10 folds cross-validation

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
January	0.9972	0.0018	0.63 ± 0.52	0.11	12.06	29.74 ± 22.67	0.41	876.90	37.55 ± 24.01	0.60	1062.23
February	0.9974	0.0015	0.75 ± 0.56	0.11	13.15	36.09 ± 25.08	0.50	818.57	44.27 ± 26.43	0.73	911.66
March	0.9855	0.0095	0.68 ± 0.60	0.14	11.96	29.47 ± 27.86	0.52	680.42	41.07 ± 30.29	0.73	1171.78
April	0.8709	0.0947	4.99 ± 9.47	1.74	19.87	156.47 ± 250.42	11.16	1094.51	299.45 ± 289.27	39.76	2001.82
May	0.9953	0.0030	0.73 ± 0.58	0.15	12.42	34.65 ± 25.87	0.48	937.87	43.37 ± 27.33	0.73	1143.39
June	0.9970	0.0018	0.82 ± 0.62	0.13	13.38	38.21 ± 29.40	0.44	1023.59	48.31 ± 31.10	0.65	1261.02
July	0.9300	0.0415	1.06 ± 0.93	0.30	12.75	44.22 ± 39.44	0.87	819.10	60.17 ± 42.75	1.31	1075.16
August	0.9993	0.0004	2.25 ± 1.42	0.14	20.96	15.77 ± 11.69	0.60	153.92	19.70 ± 12.35	0.81	199.56

With regards to granularity and time window influence in the models performances, Figure 5.11 presents the variations of MAPE and  $R^2$ .

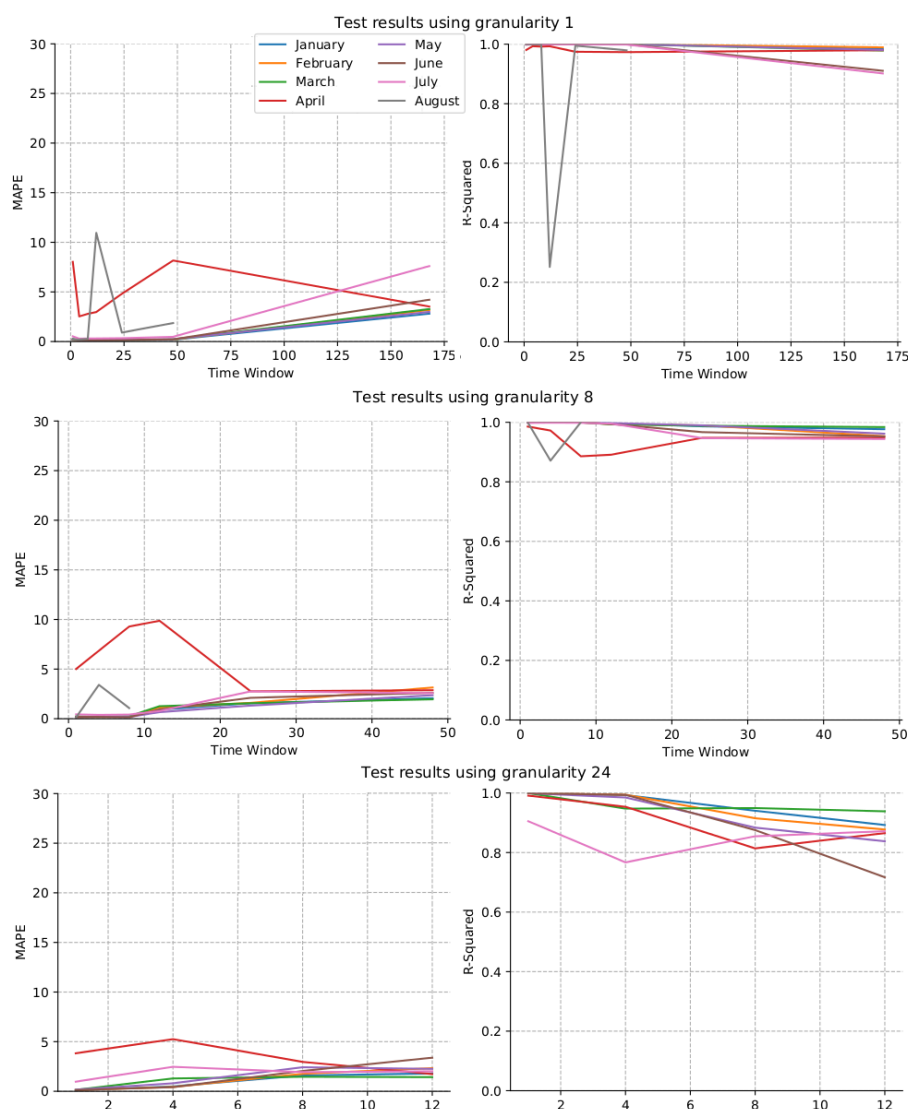


Figure 5.11: Influence of granularity and time window in MAPE and R-Squared for linear regression with 10 folds cross-validation on enhanced dataset

Most months benefited from the enhanced dataset. For a granularity 1, the best results

were found up to the 24-hour mark. MAPE distribution was below 1% in most months up to a time window of 48. For granularity 8, the best time window performance was up to time window 8. Similar to the basic scenario, increase in granularity rose the oscillations of values for MAPE and  $R^2$ , the relation/influence between these two metrics remained however intact.

Lastly, for granularity 24, in the majority of months, the best performing time window was 4. Correlation between MAPE and  $R^2$  remained intact. Similar to the basic scenario, with increased granularity the magnitude of MAPE fluctuation through  $R^2$  influence decreased.

When using enhanced dataset, months with outliers such as July seem to be handled better by linear regression. However, months with correction procedures (wrong values were substituted by average values) such as April seem to have more difficulties.

## Artificial neural networks

Table 5.9: Result synthesis for all months using ANN, enhanced dataset and 10 folds cross-validation

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
January	0.9907	0.0062	0.91 ± 0.75	0.09	12.86	37.06 ± 30.13	0.31	656.54	47.80 ± 31.99	0.41	875.73
February	0.9636	0.0218	1.38 ± 1.47	0.14	13.28	52.55 ± 45.92	0.49	943.07	70.14 ± 49.26	0.68	1149.88
March	0.9864	0.0091	1.16 ± 0.92	0.11	13.47	40.63 ± 30.46	0.34	527.79	50.87 ± 32.15	0.45	673.13
April	0.9311	0.0426	6.30 ± 18.83	1.10	32.09	191.20 ± 546.64	4.96	821.86	597.05 ± 684.75	12.92	5325.50
May	0.9876	0.0083	1.14 ± 0.89	0.15	13.33	45.69 ± 38.16	0.47	848.46	59.66 ± 40.66	0.64	1187.86
June	0.9787	0.0140	1.25 ± 1.10	0.13	14.03	55.96 ± 53.77	0.40	1067.60	78.08 ± 58.22	0.55	1690.30
July	0.9495	0.0378	1.24 ± 1.05	0.08	13.69	46.55 ± 39.08	0.23	656.15	61.04 ± 41.74	0.39	821.39
August	0.9884	0.0088	1.41 ± 1.03	0.25	13.75	15.29 ± 10.88	0.74	316.97	18.80 ± 11.44	1.03	380.41

Identical to linear regression, ANN models had an overall better performance with the enhanced dataset. Table 5.9 presents the results synthesis across all months with cross-validation and enhanced dataset.

In terms of average  $R^2$ , results in all months were above 0.9311. Average overfitting results remained overall low with variations in values having almost no influence in the models. For the average MAPE, the enhanced dataset led to an increase in performance of the models in all months. Nonetheless, performing still worse than linear regression. The best models achieved mostly MAPE values below 0.25%. With regards to granularity and time window influence in MAPE and  $R^2$ , through Figure 5.12 it is possible to verify their overall impact.

As illustrated in Figure 5.12, an increase in time window when using a granularity of 1 increased the MAPE value, the best models were built with a time window lower than 24. Results were similar in most months. For granularity of 8, the best models were created with a time window lower than 8. While for granularity 24, the majority of months achieved their best models with a time window lower than 4.

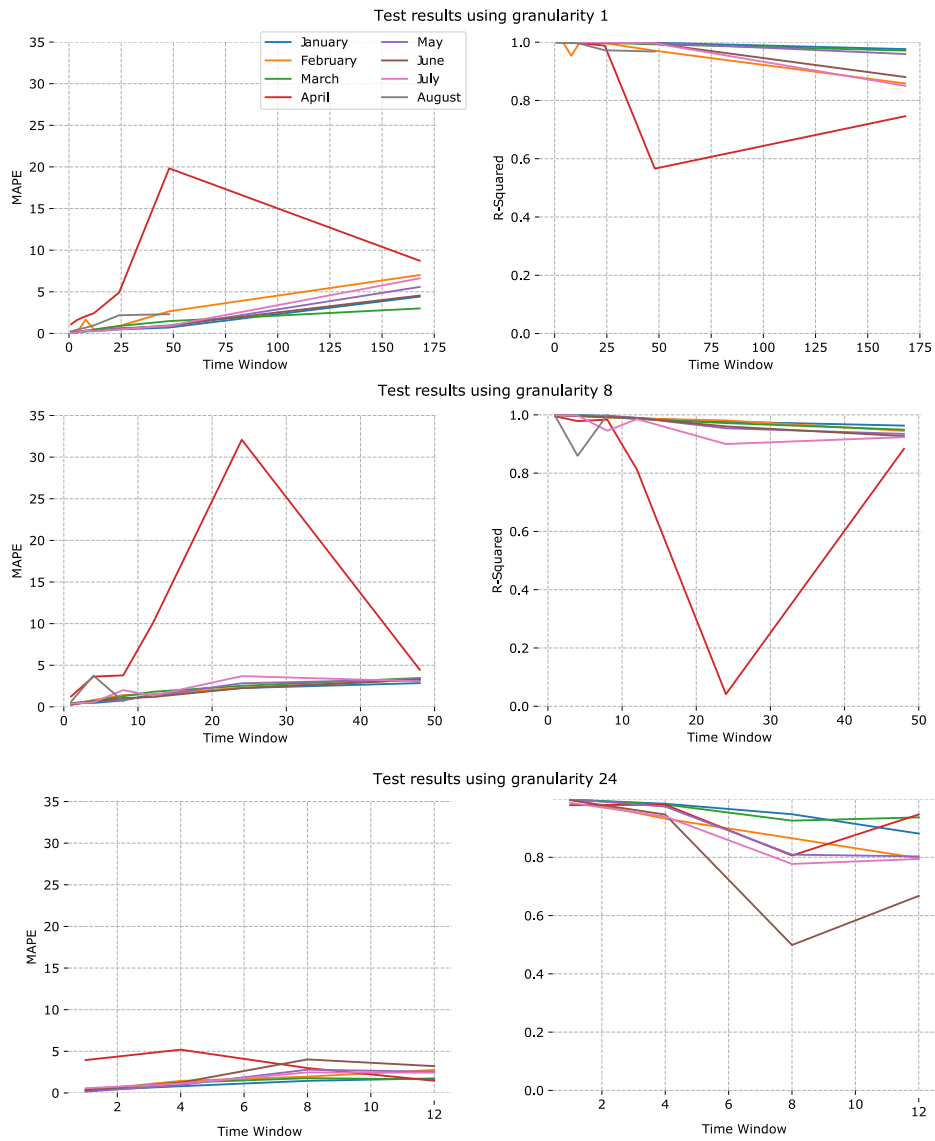


Figure 5.12: Influence of granularity and time window in MAPE and R-Squared for ANN with 10 folds cross-validation on enhanced dataset

## Support vector machine

Table 5.10: Result synthesis for all months using SVM, enhanced dataset and cross-validation 10 folds

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
January	0.9979	0.0014	0.65 ± 0.55	0.15	12.23	29.61 ± 24.36	0.53	838.47	38.38 ± 25.90	0.77	1073.84
February	0.9907	0.0055	0.86 ± 0.66	0.18	13.32	40.61 ± 29.71	0.64	795.97	50.74 ± 31.45	0.99	875.48
March	0.9968	0.0021	0.68 ± 0.52	0.21	11.94	26.33 ± 19.20	0.69	615.61	32.67 ± 20.23	0.96	762.64
April	0.9483	0.0292	3.08 ± 5.64	1.06	16.92	119.95 ± 235.51	11.37	1511.21	271.73 ± 281.79	58.06	2352.37
May	0.9965	0.0022	0.75 ± 0.57	0.18	12.51	33.73 ± 23.67	0.56	941.24	41.32 ± 24.87	0.86	1118.90
June	0.9864	0.0082	0.87 ± 0.76	0.17	13.61	42.01 ± 38.88	0.53	1031.36	57.39 ± 41.83	0.78	1362.89
July	0.9876	0.0076	0.96 ± 0.78	0.34	12.97	35.32 ± 26.85	1.01	633.24	44.46 ± 28.38	1.51	787.93
August	0.9977	0.0017	0.93 ± 0.76	0.23	13.15	11.68 ± 9.78	0.70	306.05	15.24 ± 10.42	1.03	402.86

Akin to linear regression and ANN, the enhanced dataset provided a better performance in models. All months attained better results in the metrics of average  $\bar{R}^2$  and overfitting. In terms of average MAPE, results were also significantly better in all months. Overall average MAPE values were predominately under the 1% mark. As for minimum MAPE, results also improved immensely with most models achieving a MAPE below 0.34.

With reference to granularity and time window impact an increase in performance was verified, as can be seen in Figure 5.13.

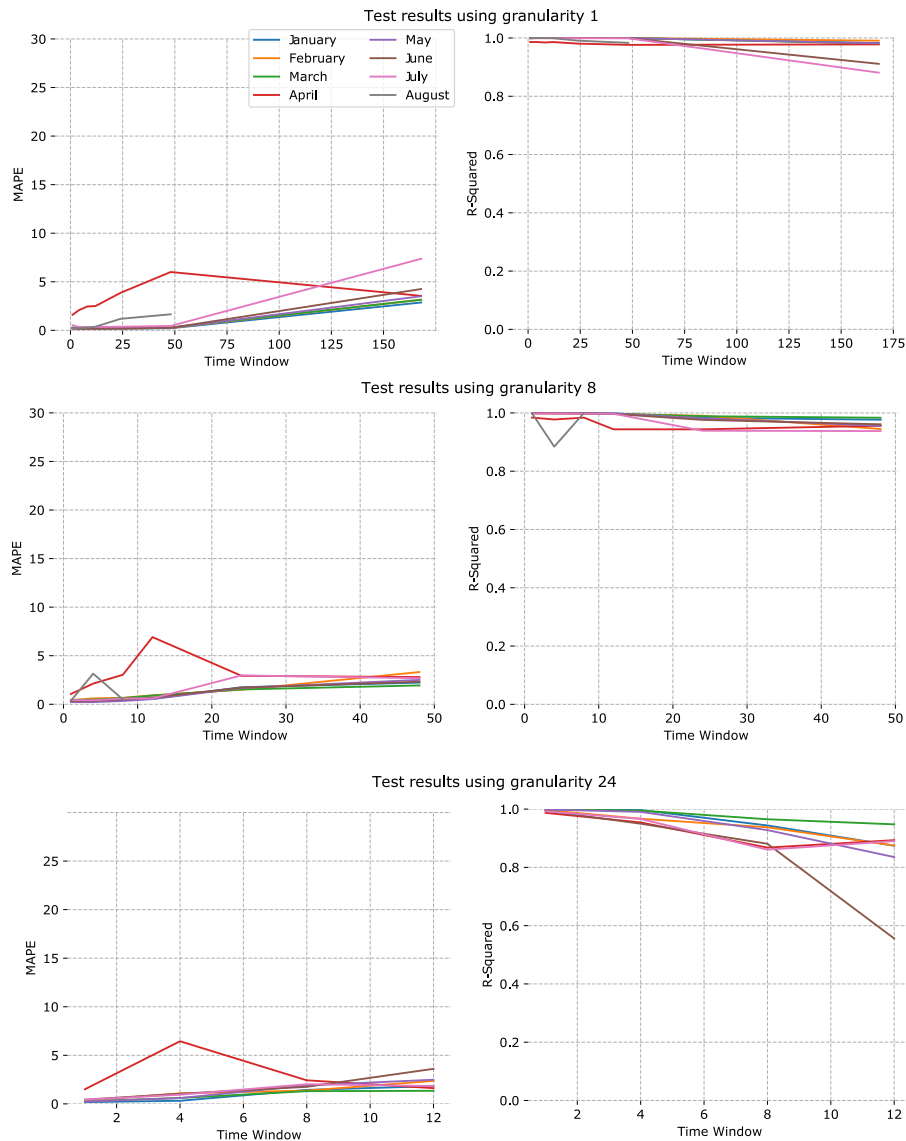


Figure 5.13: Influence of granularity and time window in MAPE and R-Squared for SVM with 10 folds cross-validation on enhanced dataset

With granularity 1, all months decreased their overall MAPE, the time windows in which the months performed better lied between 4 and 48. After that the error would increase gradually with the time window. For a granularity of 8, all months with the exception of April, achieved results always below the 5% mark, for April even though it breached the 5% mark results were still an improvement when compared to the basic

dataset. With regards to optimal time window, results showed that this lied between 4 and 8. Last of all, for granularity 24, the best performing time window was found to be 12 for April and 4 for the rest of the months.

## Model comparison

For the sake of selecting the best model for every month, Table 5.11, containing the best models per month per learning method was created.

Table 5.11: Best models with respective time window and granularity configuration for Linear regression, ANN and SVM, using 10 folds cross-validation on enhanced dataset

Month	Learning Method	Granularity	Time Window	$\bar{R}^2$	Model Overfitting	MAPE	MAE	RMSE
January	LR	24	1	0.999614	0.000106	0.11	9.10	11.13
	<b>ANN</b>	<b>1</b>	<b>4</b>	<b>0.999942</b>	<b>0.000003</b>	<b>0.09</b>	<b>0.44</b>	<b>0.56</b>
	SVM	1	4	0.999889	0.000005	0.15	0.53	0.77
February	LR	24	1	0.999209	0.000234	0.11	9.57	15.37
	<b>ANN</b>	<b>1</b>	<b>4</b>	<b>0.999934</b>	<b>0.000001</b>	<b>0.14</b>	<b>0.58</b>	<b>0.81</b>
	SVM	1	4	0.999827	0.000009	0.18	0.64	1.02
March	LR	24	1	0.999680	0.000088	0.14	10.70	13.59
	<b>ANN</b>	<b>1</b>	<b>4</b>	<b>0.999939</b>	<b>0.000003</b>	<b>0.11</b>	<b>0.34</b>	<b>0.45</b>
	SVM	1	8	0.999831	0.000016	0.21	0.69	0.96
April	LR	24	12	0.884281	0.069431	1.74	137.79	231.43
	<b>ANN</b>	<b>1</b>	<b>4</b>	<b>0.998905</b>	<b>0.000054</b>	<b>1.10</b>	<b>7.13</b>	<b>18.11</b>
	SVM	8	1	0.982440	0.001596	1.06	74.53	364.70
May	LR	1	4	0.999877	0.000006	0.15	0.48	0.73
	<b>ANN</b>	<b>1</b>	<b>4</b>	<b>0.999892</b>	<b>0.000005</b>	<b>0.15</b>	<b>0.47</b>	<b>0.65</b>
	SVM	1	8	0.999823	0.000017	0.18	0.56	0.86
June	LR	24	1	0.998955	0.000299	0.13	9.82	15.37
	<b>ANN</b>	<b>1</b>	<b>4</b>	<b>0.999904</b>	<b>0.000005</b>	<b>0.13</b>	<b>0.40</b>	<b>0.55</b>
	SVM	1	4	0.999806	0.000010	0.17	0.53	0.78
July	LR	1	4	0.999376	0.000030	0.30	0.88	1.31
	<b>ANN</b>	<b>1</b>	<b>4</b>	<b>0.999944</b>	<b>0.000003</b>	<b>0.08</b>	<b>0.29</b>	<b>0.39</b>
	SVM	1	4	0.999164	0.000040	0.34	1.01	1.52
August	<b>LR</b>	<b>8</b>	<b>1</b>	<b>0.999518</b>	<b>0.000321</b>	<b>0.14</b>	<b>3.29</b>	<b>3.92</b>
	ANN	1	4	0.998235	0.000612	0.25	1.36	1.80
	SVM	1	4	0.999356	0.000223	0.23	0.71	1.06

With the enhanced dataset, the best models were created all, with exception of one month, with ANN. The remaining month achieved its best model with linear regression.

For the months of January, February, March and April, ANN was the clear choice since it outperformed the other learning methods in every metric.

In May, the choice was not so obvious as linear regression and ANN achieved similar results, with ANN having the upper hand in most metrics. The main differential factor for selecting ANN was nevertheless the better ability of handling outliers.

For June, ANN and linear regression tied in MAPE, however in all other metrics ANN performed better, hence why ANN was the learning method selected for June.

In July, ANN similar to some previous months outperformed completely the other learning methods. Lastly for August, linear regression was the best performing learning method, it outperformed all other learning methods in every metric.

Overall, ANN was the best performing learning method, a different outcome to the split ratio scenario, where linear regression and ANN were the best performing learning methods.

With regards to time window and granularity of the best models per learning method, a clear trend is visible, Figure 5.14 demonstrates that lower time windows usually perform better.

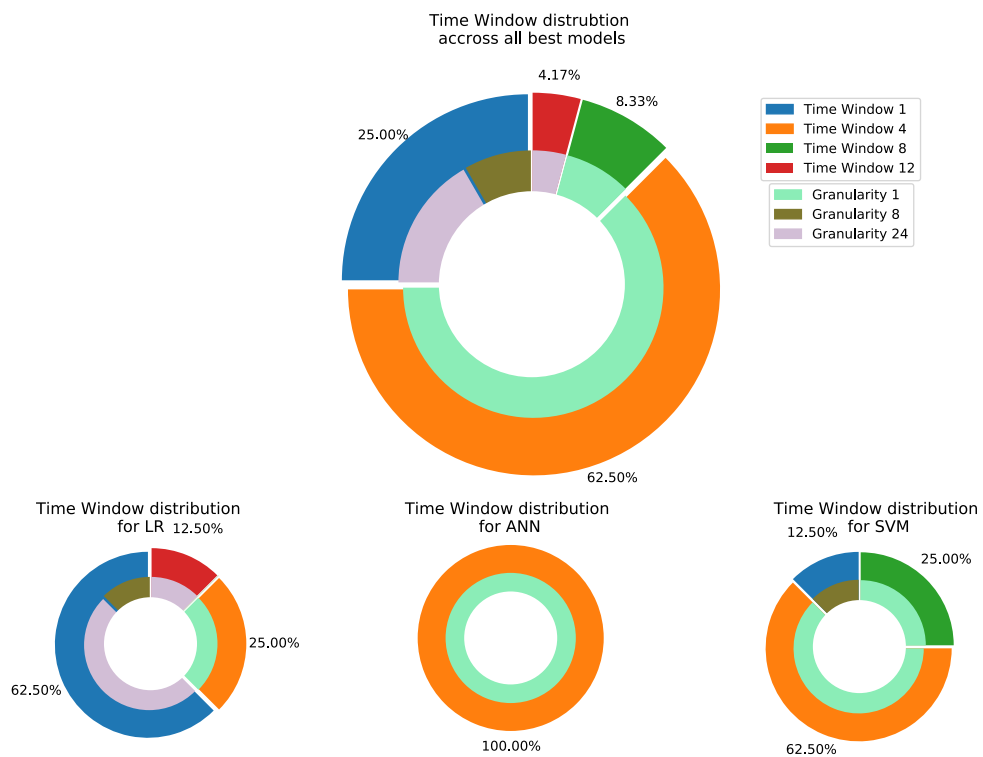


Figure 5.14: Time window distribution across all best models for single month with 10 folds cross-validation on enhanced dataset

For linear regression the best models were created with a time window of 1, 62.50% of the times. It is important to mention that whenever time window 1 occurred, granularity was either 8 or 24. The rest of linear regressions' models were created with time window 4, 25% of the times and time window 12, 12.5% of the times. For ANN, results were uniform, with all months achieving their best models with time window 4 and granularity 1. Lastly for SVM, time window 4 with granularity 1 was again the best configuration, with 62.5%. Followed by time window 8 and 1 with 25% and 12.5% respectively.

Overall time window 4 with granularity 1 was the best configuration, with 62.5% of all models being created with this configuration. Time window 1 occupied 25% of the spectrum with either granularity 8 or 24. The least used configurations were, time window 8 and 12 with 8.33% and 4.17% respectively. In terms of granularity, the value of 1 was the best choice.

To conclude, in order to better contrast how the use of the enhanced dataset affected the various models, a MAPE distribution was made and presented in Figure 5.15.

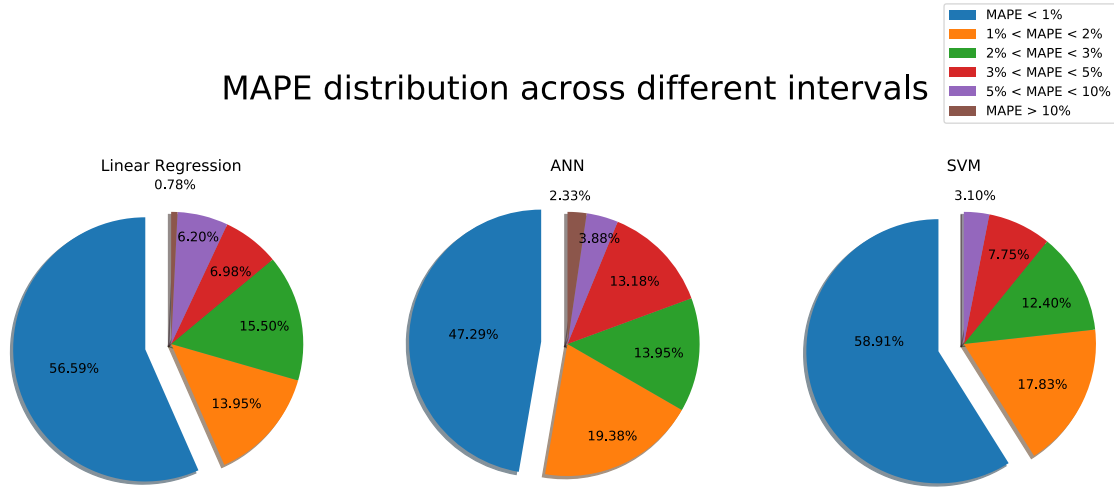


Figure 5.15: Time window distribution across all best models for single month with 10 folds cross-validation on enhanced dataset

Contrary to the selected best models, ANN was not the best performing learning method, when it comes to create low MAPE models. SVM was the clear winner in this aspect, with 58.91% of all models achieving a MAPE below 1%. SVM was followed by linear regression and ANN with 56.59% and 48.29% respectively. In terms of models with a MAPE above 10%, ANN was again the worst performing learning method, with 2.33% of all models surpassing this mark. Linear regression came next with 0.78% and SVM came last, with no created model surpassing the 10% MAPE mark.

### 5.3.3 Test runs and analysis for single year dataset

When analysing the performance of each learning method when using the yearly dataset, some similarities to the single month approach were verified. Table 5.12 synthesis the results of the test runs.

Table 5.12: Average and best results for single year dataset using 10 folds cross-validation

Result synthesis for single year dataset

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
LR	0.8811	0.0538	3.01 ± 8.08	1.12	8.18	145.63 ± 372.20	5.09	775.02	403.42 ± 453.72	26.27	1752.81
ANN	0.9774	0.0078	2.63 ± 3.84	0.47	10.97	133.61 ± 274.05	1.69	576.58	306.52 ± 324.52	5.26	1110.96
SVM	0.9417	0.0373	1.40 ± 0.48	0.60	6.88	73.59 ± 197.02	3.94	576.24	215.58 ± 244.09	26.14	866.36

Best models with respective configuration

Learning Method	Granularity	Time Window	$\bar{R}^2$	Model Overfitting	MAPE	MAE	RMSE
LR	24	1	0.987886	0.000451	1.12	101.71	256.39
ANN	1	4	0.999377	0.000004	0.47	1.69	5.37
SVM	24	1	0.991665	0.000310	0.48	54.03	262.27



Identical to the single month test scenario, ANN was the best performing learning method, from an absolute point of view. It achieved the lowest MAPE values with 0.47%, followed by SVM with 0.60% and linear regression with 1.12%. In terms of average MAPE, SVM was the best learning method, with all learning methods achieving average values below the 3.01%. With respect to MAPE and  $R^2$  variations, Figure 5.16 presents the variations.

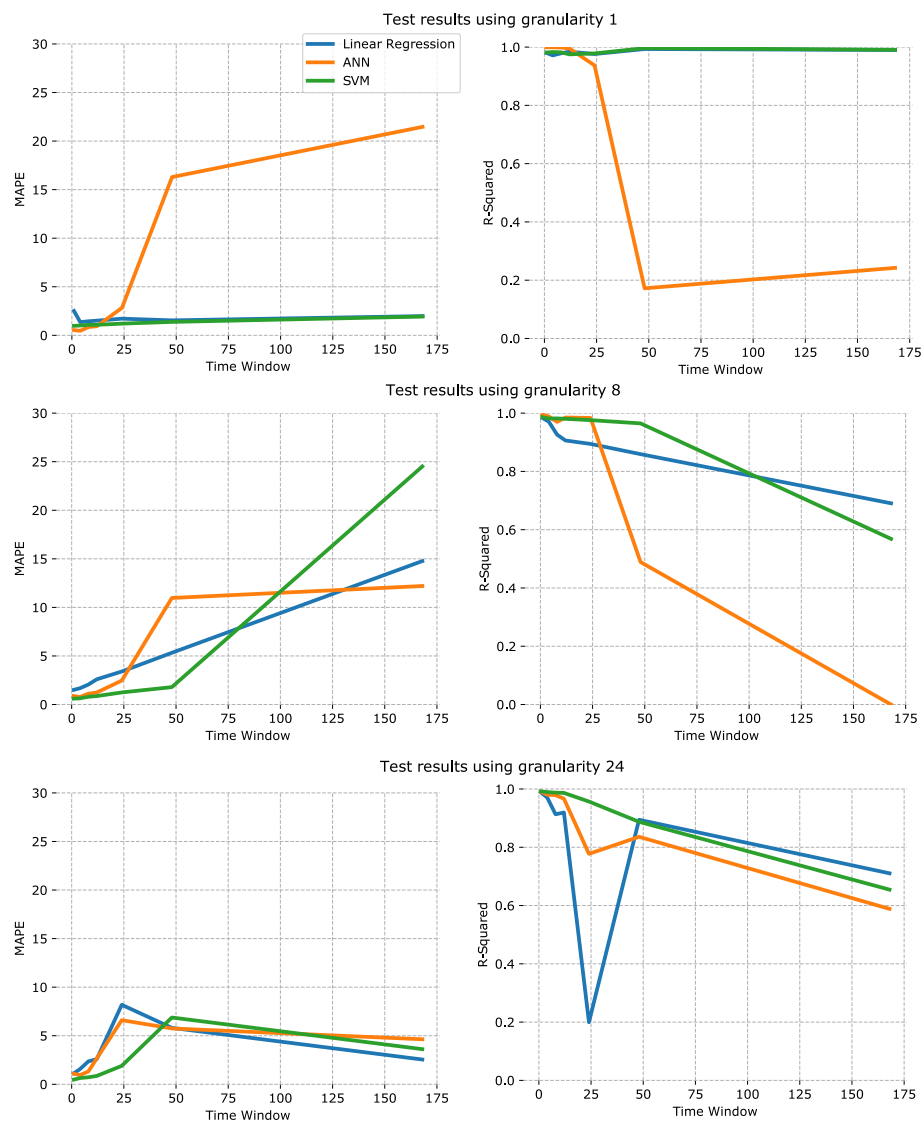


Figure 5.16: Time window distribution across all best models for single month with 10 folds cross-validation on enhanced dataset

For granularity 1 linear regression and SVM performed very similar, with ANN standing out with higher MAPE values, the best performing time windows were found below 12. For granularity 8 the best performing time windows were 4 and 8, with SVM performing better amongst all learning methods. With a granularity of 24 a time window of up to 4 was the best choice, with MAPE values peaking at time window 24 for linear regression and ANN and 48 for SVM.

To conclude, a MAPE distribution was created and depicted in Figure 5.17.

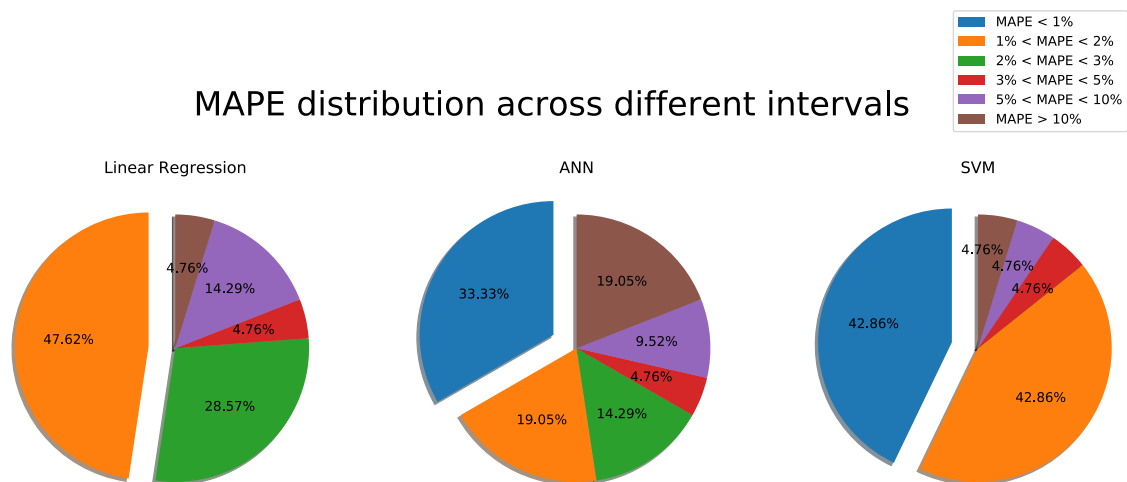


Figure 5.17: Time window distribution across all best models for single year with 10 folds cross-validation on enhanced dataset

In this category linear regression was the worst performing learning method, with no model achieving a MAPE below 1%. SVM was the best performing learning method with 42.86%, followed by ANN with 33.33%. In terms of models with a MAPE above 10%, ANN was the worst learning method with 19.05%, followed by linear regression and SVM, which tied with 4.76%. In summary, the enhanced dataset with cross-validation from an average perspective brought an immense increase in performance, reducing average errors, overfitting and increasing  $\bar{R}^2$  as well as the learning methods ability to handle outliers.

## 5.4 Conclusion

In this chapter the test results of the server load prediction case study were presented and analysed. Through this analysis it was possible to identify that the addition of extra attributes to the basic or enhanced dataset increased significantly the learning methods performances. The enhanced dataset outperformed the basic dataset in most occasions. The use of cross-validation fortified the approach taken from a statistical point of view. With regards to time window and granularity, a lower granularity presented better results in most cases, while time windows up to 24 were the most optimal ones. Lastly, the best performing learning method was SVM, with linear regression coming close second. ANN was the worst performing learning method from an average point of view. An approach for granularity and time window optimization through genetic algorithms was developed and published. It is available for deeper analysis if desired.

# Chapter 6

## Energy load forecast and model exchange: experimental setup and tests

In the energy load forecast and model exchange chapter, experimental setup, test and results of the energy load forecast, as well as, model exchange test cases are presented and analysed. On a first stage, the test environment is explained, focus is made on the difference between server load prediction and energy load forecast. Succeeding the experimental setup comes the test and results analysis of energy load forecasting test and model exchange scenario. The chapter is closed with a chapter conclusion containing the synthesis of the most relevant information.

### 6.1 Test environment

Identical to the server load prediction environment, data preparation and analysis steps were taken to produce an initial usable dataset. These steps were the same as in server load prediction, being the only difference the energy data in processing.

The resulting dataset had also two different variations: a basic/standard one, containing the most basic attributes and a enriched/enhanced one, in which the attributes were added to increase the models performances. Figure 6.1 presents an example of EDP datasets variations.

As can be verified, the basic version only contains a single attribute, *Current*, as explained in Chapter 4. This was due to the limited quality of the raw data. In the case of the enhanced dataset, additional attributes to the ones used in the final enhanced dataset were tested in terms of model performance. These attributes were: date and time, day of week, holiday, weekend flag and average, minimum and maximum values for humidity, pressure, wind and precipitation rate. All these attributes were tested individually and in conjunction with each other. These attribute were finally excluded as their presence proved to have a detrimental effect the models performance.

Each created dataset variation was transformed in accordance to the same test configurations presented in Table 5.1 of previous chapter. The transformation process in the energy

### **Baseline scenario dataset**

Current
318
316
320
308
274

### **Enriched dataset scenario**

Current	Average_ current	Weekend_stdev_ current	Weekday_stdev_ current	Minimum_ temperature	Maximum_ temperature	Average_ temperature
318	79.5	4.52	6.10	18	23	20.5
316	79	4.09	5.52	18	23	20.5
320	80	4.53	6.12	18	23	20.5
308	77	4.71	6.36	18	23	20.5
274	68.5	6.20	8.38	18	23	20.5

Figure 6.1: Example of EDP's dataset variations

scenario originated a total of 84 different datasets, per dataset variation, to be used in the evaluation process.

In the modelling phase predictive models using linear regression, ANN and SVM were created. These models were designed to predict the next hour of current with a split ratio of 70/30 and 10 folds cross-validation. For the split ratio method, the records order remained unaltered after the splitting. Similar to server load prediction, results between split ratio and cross-validation were very similar, hence why it was again opted to only present the cross-validation results. Split ratio results were published and are available if a deeper analysis is desired.

For the sake of comparing the impact of the data transformation process in the two test scenarios, parametrization of the learning methods remained the same as the one used in server load prediction.

Lastly, identical tests to server load prediction were conducted with normalized and non-normalized data, results were again very similar, and it was opted to just present the results of the non-normalized data.

## **6.2 Energy load forecast on basic dataset with cross-validation**

The first test scenario embraces a basic dataset containing data referent to the electrical consumption in the city of Leiria – Portugal in the month of April 2016. As mentioned above, the base dataset consisted only of one attribute: current. The tests conducted ahead were all built on this base dataset.

Due to data availability limitation a multiple month's scenario was not possible to be conducted. As a result, the testing approach taken will follow the same procedures as the ones taken in the yearly dataset tests in the server load prediction case study. The configurations used in the test scenario were 1, 8 and 24 for granularity and 1, 4, 8, 12, 24, 48 and 168 for time window/history, non-normalization of data (as mentioned above, normalized test runs were also done, but achieved identical results, hence why it was opted to only present the non-normalized ones).

## 6.2.1 Results analysis

Alike to previous approaches a synthesis of results was made and displayed in a form of a table, Table 6.1.

Table 6.1: Average and best results for EDP basic dataset using 10 folds cross-validation

Result synthesis of all test runs

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Ampere)	Min. MAE (Ampere)	Baseline MAE (Ampere)	Avg. RMSE (Ampere)	Min. RMSE (Ampere)	Baseline RMSE (Ampere)
LR	0.7373	0.0469	12.15 ±15.10	6.55	21.35	326.58 ±291.10	18.66	1029.38	438.66 ±312.14	24.13	1283.96
ANN	0.7268	0.0428	14.56 ±16.45	7.55	26.77	440.21 ±382.96	21.35	1723.47	584.51 ±409.44	27.56	2191.36
SVM	0.7908	0.0299	13.06 ±13.89	6.37	22.64	395.65 ±285.04	18.36	1558.93	492.52 ±301.67	24.03	1678.94

Best models with respective configuration for EDP basic dataset

Learning Method	Granularity	Time Window	$\bar{R}^2$	Model Overfitting	MAPE	MAE	RMSE
LR	1	48	0.9322	0.0047	6.55	18.66	24.13
ANN	1	8	0.9140	0.0009	7.55	21.99	27.81
SVM	1	48	0.9336	0.0047	6.37	18.36	24.03

As can be verified from the table, the average values of MAPE, average  $\bar{R}^2$  and overfitting had a worse performance when compared to the server load counterpart. A somewhat expected result when taking into consideration that the base dataset only contained data from a single month and a single attribute.

Average MAPE values fluctuated between  $12.15 \pm 15.10$  and  $14.56 \pm 16.45$ . As for  $\bar{R}^2$  results varied between 0.7268 and 0.7908, while for overfitting the results were very promising, with no model surpassing on average the 0.47% mark.

With regards to the best models created by the learning methods, SVM was the best with a MAPE of 6.37, a  $\bar{R}^2$  of 0.9336 and a overfitting of only 0.47%. The ability of handling outliers was also good as MAE and RMSE did not differ much. Linear regression came close second with a MAPE of 6.55,  $\bar{R}^2$  of 0.9322 and a overfitting also of only 0.47%. ANN was the worst performing learning method, its MAPE was of 7.55,  $\bar{R}^2$  of 0.9140 and overfitting of only 0.09%, the lowest overfitting amongst all learning methods. Overall, all learning methods performed well in all metrics with exception of MAPE. All learning methods had an overfitting below the 0.5% mark, a  $\bar{R}^2$  above the 91% and MAE and RMSE close to each other, an indicative of high outlier handling capability.

Subsequently to the results synthesis analysis, an average analysis with regards to granularity and time window was performed, Figure 6.2 presents the variations.

With granularity 1, linear regression and SVM were almost identical in terms of results. Their MAPE decreases with the increase in time window, but just up to a time window of 48, which was the best performing time window, after that MAPE gradually increased. For ANN the best performing time window was 4. After that, MAPE had an overall tendency to increase with time window.  $\bar{R}^2$  accompanied the MAPE variations across all time windows.

For granularity of 8, linear regression and SVM did not overlapped results. However, the best performing time window was identical in both learning methods: time window

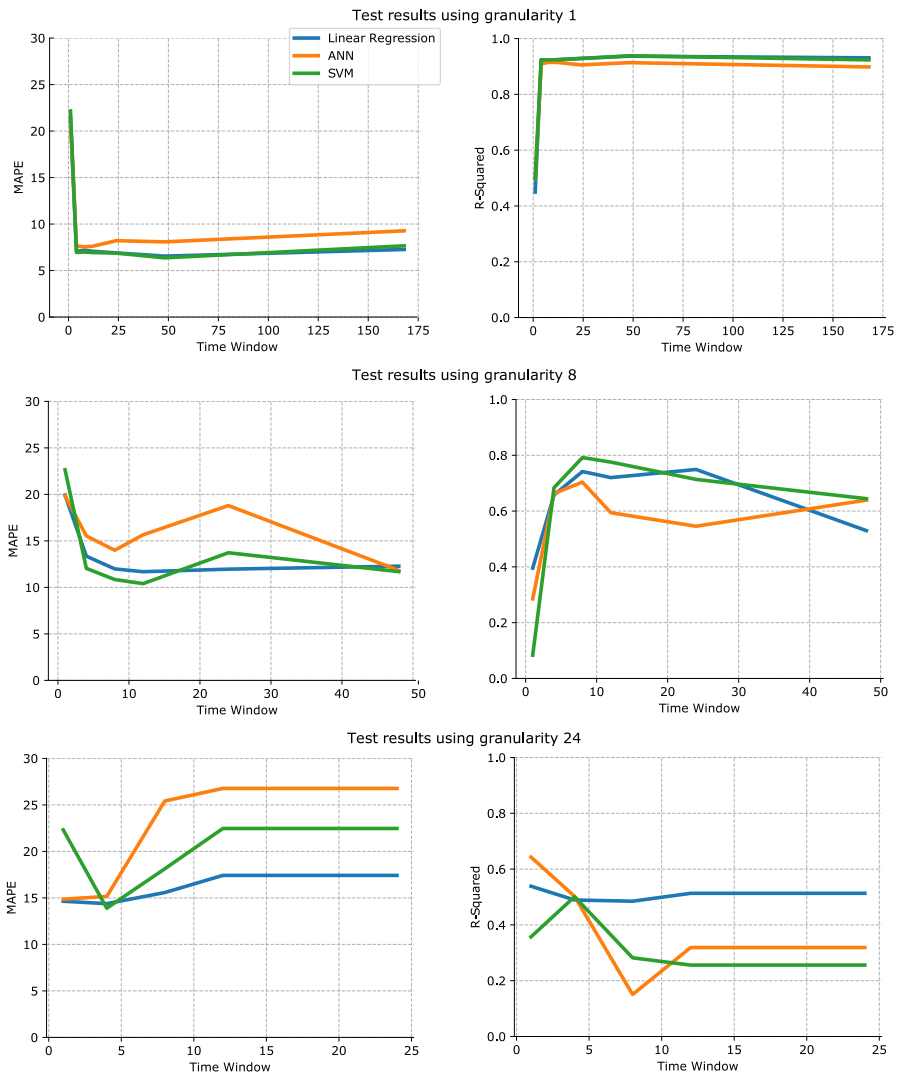


Figure 6.2: Influence of granularity and time window in MAPE and R-Squared for EDP basic dataset

12. For ANN the best performing model was found with a time window of 48. The worst performing window however, was found to be in all learning methods 24.  $R^2$  accompanied most MAPE variations,

Finally, with a granularity of 24, all learning methods presented similar patterns. Time window 4 was the best performing time window in all learning methods. After that, MAPE increased drastically with time window, stabilizing its MAPE value at time window 12. These patterns were verified in all learning methods. The influence of  $R^2$  was also much smaller when compared to lower granularities.

To conclude the results analysis of the first test scenario, a MAPE distribution appraisal was made, Figure 6.3 presents these distributions.

As it is possible to verify, in terms of overall MAPE distribution, all learning presented the same distribution, an expected result when taking into consideration that average MAPE and minimum MAPE values were all close across the three learning methods. This outcome indicates that although time window and granularity have a very positive effect in

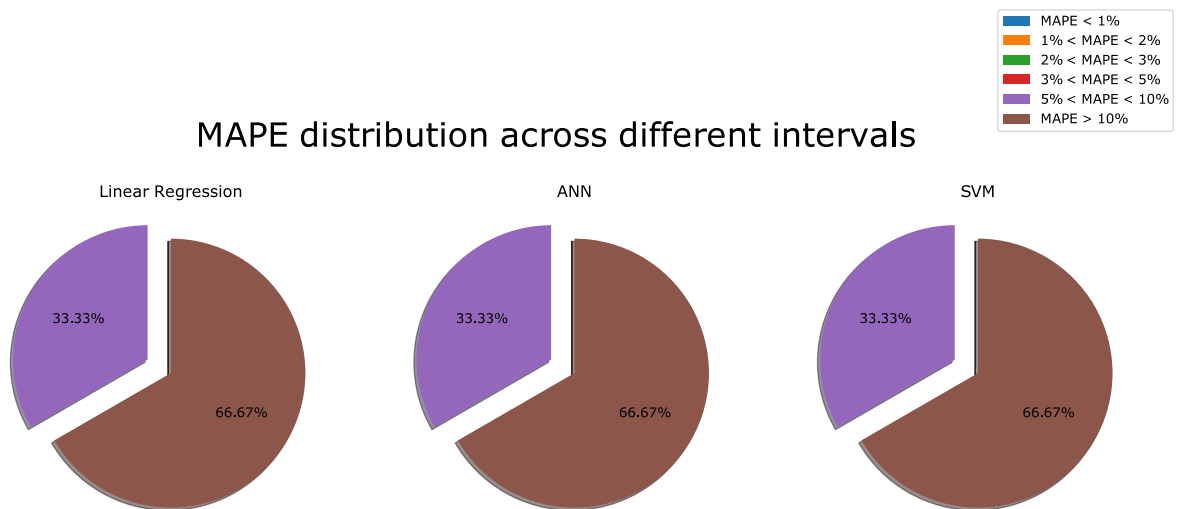


Figure 6.3: Time window distribution for EDP basic dataset

the learning models performance, decreasing initial MAPE value of 22.64% to 6.37%, in the case of SVM. The shortage in data still is the biggest challenge for this test scenario, a problem that will be tackled with the enriched dataset test scenario.

## 6.3 Energy load forecast on enhanced dataset with cross-validation

Similarly, to the basic dataset scenario, different configurations for time window and granularity were applied. Testing and analysis procedures were overall identical to the basic test scenario, being the main difference the feature analysis of the dataset. Identical to server load prediction this procedure was not compulsory for the basic scenario since the dataset only contained a single attribute.

### 6.3.1 Feature analysis

As aforesaid different attributes were tested before a final selection of attributes to be used in the enhanced dataset was made. Similar to server load prediction statistical attributes were the most efficient, with exception for the temperature attributes. From the different tests conducted emerged the final enhanced dataset depicted in Figure 6.1. Nonetheless, in order to validate the attributes chosen from a statistical point of view, a feature analysis was conducted. Figure 6.4 presents the correlation matrix of the enhanced dataset attributes.

As evidenced, all features are positively correlated among themselves. Only *Average\_current* with *Current*, achieved a correlation higher than 0.95. Usually a high correlation, above 0.95, could imply that the two attributes had more or less the same impact in the models performance, sharing the same information, redeeming one of the attributes redundant and useless. Nevertheless, the learning methods were tested with and without these attributes and it was concluded that removing one of the attribute weakened the models performance.

	Current	Average_ Current	Min_temp	Max_temp	Average_ temp	Weekday_ stdev_current	Weekend_ stdev_current
Current	1						
Average_current	0.99	1					
Min_temp	0	0.01	1				
Max_temp	0.06	0.06	0.24	1			
Average_temp	0.05	0.05	0.65	0.9	1		
Weekday_stdev_current	0.1	0.08	0.07	0.41	0.42	1	
Weekend_stdev_current	0.09	0.07	0.06	0.4	0.41	0.85	1

Figure 6.4: Correlation matrix for EDP's enhanced dataset

On the contrary, a lower correlation coefficient can indicate that the variables share very little information between themselves, an indicator that the variables might just be noise to the learning method. Test were also conducted for lower correlation on each attribute. Results concluded that removing any of the selected attributes would have a detrimental effect in the overall models performance.

### 6.3.2 Results analysis

Having certified the validation of the enhanced dataset, testing and analysis were held. The testing and evaluation procedures, as well as the dataset configurations, were equivalent to the ones used with the basic dataset.

Table 6.2: Average and best results for EDP enhanced dataset using 10 folds cross-validation

Result synthesis of all test runs

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Ampere)	Min. MAE (Ampere)	Baseline MAE (Ampere)	Avg. RMSE (Ampere)	Min. RMSE (Ampere)	Baseline RMSE (Ampere)
LR	0.9153	0.0363	4.19 ±9.07	0.05	14.09	203.51 ±83.81	1.23	1011.38	227.54 ±88.33	1.46	1014.76
ANN	0.9216	0.0338	5.66 ±9.74	0.27	20.63	260.92 ±108.03	1.55	1587.23	288.75 ±112.64	10.90	1759.93
SVM	0.9238	0.0326	4.13 ±9.11	0.05	16.30	185.75 ±79.82	1.22	1170.12	212.48 ±85.47	1.45	1175.09

Best models with respective configuration for EDP enhanced dataset

Learning Method	Granularity	Time Window	$\bar{R}^2$	Model Overfitting	MAPE	MAE	RMSE
LR	8	4	0.99988	0.00011	0.05	1.23	1.46
ANN	24	4	0.99911	0.00056	0.27	20.44	24.18
SVM	8	4	0.99988	0.00011	0.05	1.22	1.45

As shown in Table 6.2, the appliance of the enhanced dataset led to a dramatic increase in all learning methods performances. All average MAPE values fell under the 6% mark, a vast increase in performance when considering that the basic dataset averaged under the 15% mark. Average  $\bar{R}^2$  also increased significantly with all learning methods breaching the 0.90 mark. From an absolute perspective linear regression and SVM tied, with both methods achieving in all metrics equivalent results. With regard to MAPE values, both achieved 0.05%. The models were also created by using a granularity of 8 and a time



window of 4.  $\bar{R}^2$  and overfitting also increased significantly, with values of 0.99988 and 0.00011 respectively. ANN was the least performing method, with all metrics falling behind linear regression and SVM. The results from an overall perspective were nonetheless impressive, with a MAPE of only 0.27 and a  $\bar{R}^2$  and overfitting of 0.99911 and 0.00056 accordingly. The best ANN model was created with a granularity of 24 and a time window of 4.

Relative to the MAPE and  $R^2$  variations, Figure 6.5 presents an global overview.

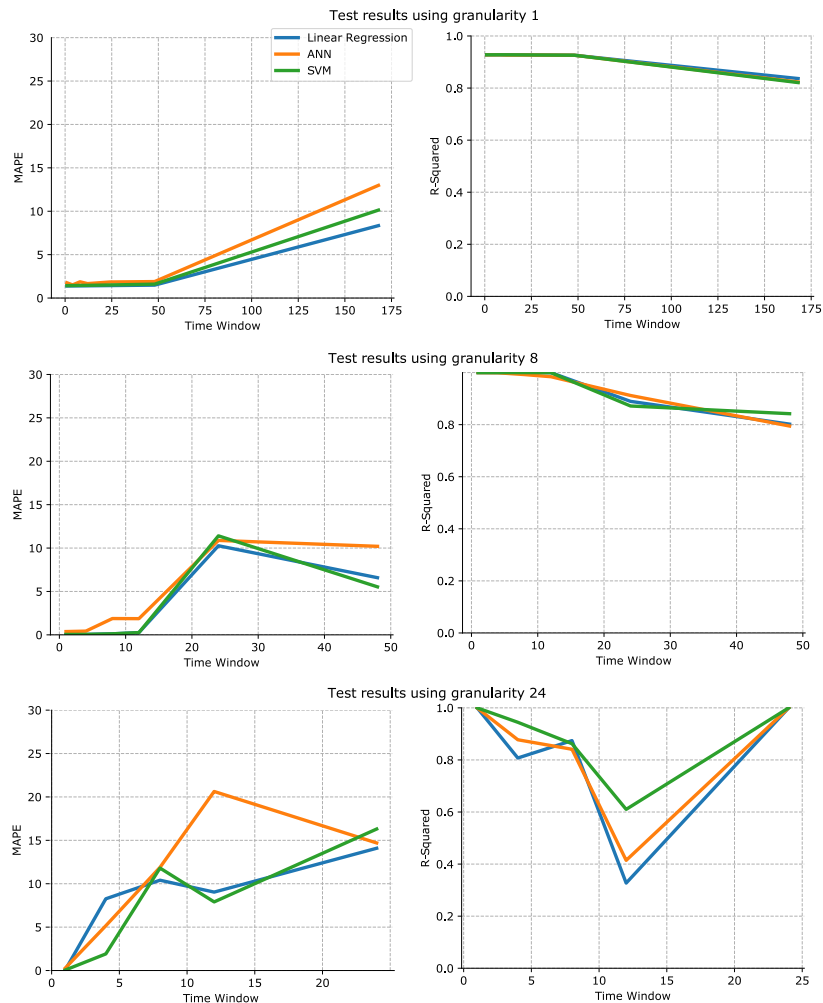


Figure 6.5: Influence of granularity and time window in MAPE and R-Squared for EDP enhanced dataset

For granularity 1, all learning methods performed very identical, up to time window 48. After that, MAPE increased progressively in all learning methods, with ANN registering the highest increase, followed by SVM and linear regression respectively. The best performing time window was in all cases 4.

With a granularity 8, the best performing time window was again 4, with linear regression and SVM performing very alike and ANN performing slightly worse. MAPE values started to increase after time window 12, peaking at time window 24 and then decreasing steadily. At last with a granularity of 24, time window 4 was again the best choice, with MAPE values increasing after that.

In all test scenarios, when using the enhanced dataset, the relation/dependence between MAPE and  $R^2$  was clearly verified.

To conclude, a MAPE distribution analysis, depicted in Figure 6.6, was as well performed.

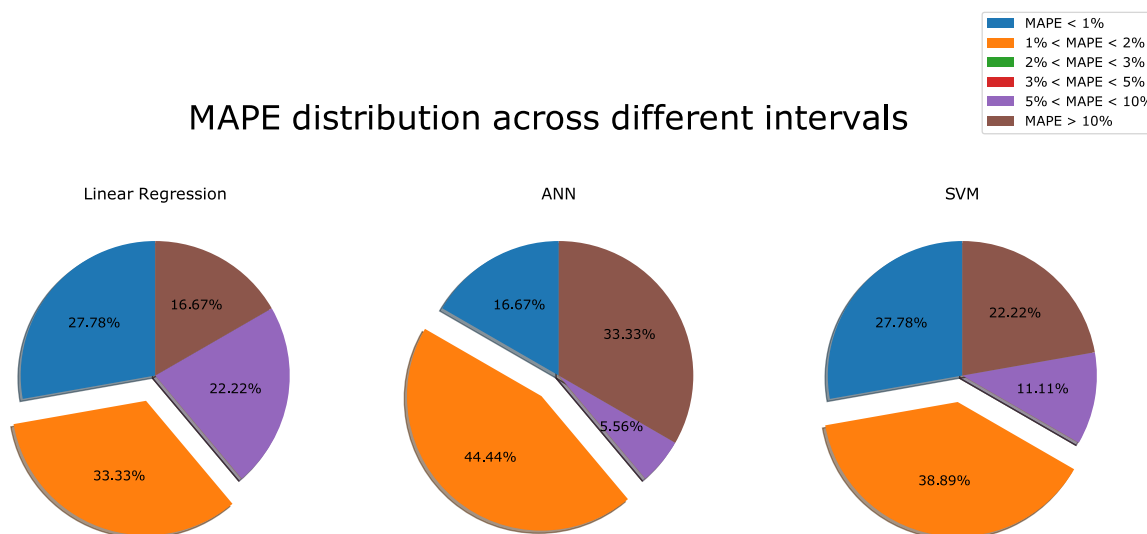


Figure 6.6: Time window distribution for EDP enhanced dataset

Differently from the basic dataset scenario, with the enhanced dataset differences in MAPE distribution between learning methods were clearly verified. Linear regression and SVM were the best learning methods with regards to MAPE values below 1%, with 27.78% of all test runs beating this mark. As for ANN, only 16.67% of the test runs achieved this goal. In terms of models with a MAPE above 10%, ANN was once more the least performing learning method with 33.33%, followed by SVM and linear regression with 22.22% and 16.67% respectively. Overall, all learning methods performed great with the enhanced dataset, a lower time window in conjunction with a granularity of 8 seemed to bear the best results in terms of model performance.

## 6.4 Model exchange

With the conclusion of the server load prediction and energy load forecasting case studies, the last stage of this work is initiated. This stage consists in training a new model for each case study and later on trade the test datasets of each field and test them on the model trained with data from the opposite field. Figure 6.7 shows the architecture of the above mentioned approach.

As seen in the figure, the first stage involves analysing all the results that originated from the two case studies. Having conducted a deep analysis of these results, a selection is then made, taking into consideration the best model configuration for both test scenarios, i.e., the objective of this selection is to find the common dataset configuration that achieves the best performance for server load prediction and energy load forecast simultaneous.

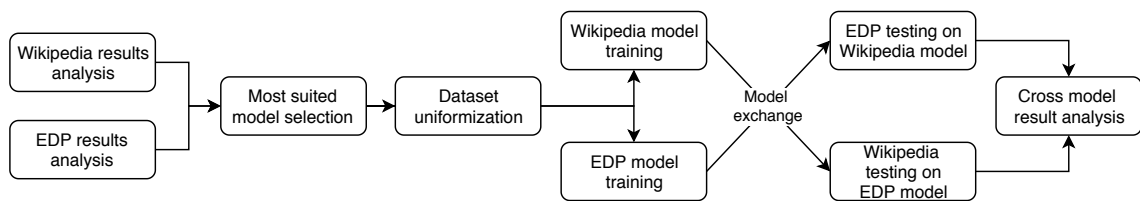


Figure 6.7: Model exchange architecture

With the model selection concluded, the dataset uniformization step is commenced. This step is one of the most crucial for model exchange. The objective of this procedure is to remove the attributes that are not mutual to both cases studies datasets, be it the basic dataset or enhanced dataset. If this procedure was not taken, it would be impossible to test a dataset trained by a different scenario, e.g., the server load prediction and energy load forecasting basic dataset were altered to only contain the load as attribute, if the server dataset would have remained the same, it would be impossible to test it in the trained energy model, since it contains load and request as attributes and the energy model only knows load. As such, the uniformed basic dataset contains only the load attribute and the uniformed enhanced dataset only contains load, average load and standard deviation of load as attributes.

Having validated the uniformed datasets, the next step is to train and test each model with the dataset corresponding to its scenario. After that the datasets are switched and tested in their opposite trained model. The last step is to do a cross model analysis, containing the results of the baseline test, the uniformed basic and enhanced datasets trained and tested in their respective scenarios and lastly, the testing of the uniformed basic and enhanced datasets in the opposite model.

### 6.4.1 Basic dataset result analysis

The first test scenario will be the uniformed basic dataset, starting with Wikipedia prediction and succeeding with EDP prediction.

#### Wikipedia prediction

To better understand the results of this testing step an overview of the Wikipedia prediction results was created and presented in Table 6.3.

As can be verified three different model types for Wikipedia predictions were tested per learning method. The first type was the baseline dataset, this dataset contains no dataset configuration steps applied, having a granularity of 1 and time window of 1. Secondly, the uniformed basic dataset of Wikipedia is used for training and testing of the model. Lastly, a model is trained using the EDP's basic uniformed dataset and tested with the Wikipedia uniformed dataset. In order to successfully train and test this last type of model, the data needed to be normalized, as the difference in greatness between the load in Wikipedia and EDP led to the model only been able to predict values in the same greatness as the Wikipedia load, a prediction range not viable when predicting EDP's load.

Table 6.3: Model exchange results overview for Wikipedia prediction with linear regression, ANN and SVM on basic dataset

Linear regression results

Model type	Avg. $\bar{R}^2$	Overfitting	MAPE(Gb)	MAE(Gb)	RMSE(Gb)
Baseline	0.73655	0.1510	14.64	1134.41	1383.89
Wikipedia prediction with Wikipedia trained model	0.99209	0.0011	3.81	12.07	16.53
Wikipedia prediction with EDP trained model	0.93349	0.0047	8.10	0.03	0.04

ANN results

Model type	Avg. $\bar{R}^2$	Overfitting	MAPE(Gb)	MAE(Gb)	RMSE(Gb)
Baseline	0.71645	0.1830	18.90	468.16	606.37
Wikipedia prediction with Wikipedia trained model	0.98212	0.0025	5.62	18.09	25.06
Wikipedia prediction with EDP trained model	0.53374	0.0038	10.39	0.04	0.05

SVM results

Model type	Avg. $\bar{R}^2$	Overfitting	MAPE(Gb)	MAE(Gb)	RMSE(Gb)
Baseline	0.73456	0.1740	18.90	468.16	606.37
Wikipedia prediction with Wikipedia trained model	0.99192	0.0011	3.76	11.93	16.73
Wikipedia prediction with EDP trained model	0.54897	0.0028	7.40	0.03	0.04

The baseline model was the worst performing type in all three learning methods, followed by the model trained by the dataset of the opposite scenario. As expected training and testing on the same scenario produced the best results, nevertheless through the use of granularity and time window it was possible to create a generic enough model that operates cross-scenario wise and still outperforms the baseline results.

The MAPE values of the different model types were computed and compared through a bar chart, as shown in Figure 6.8.

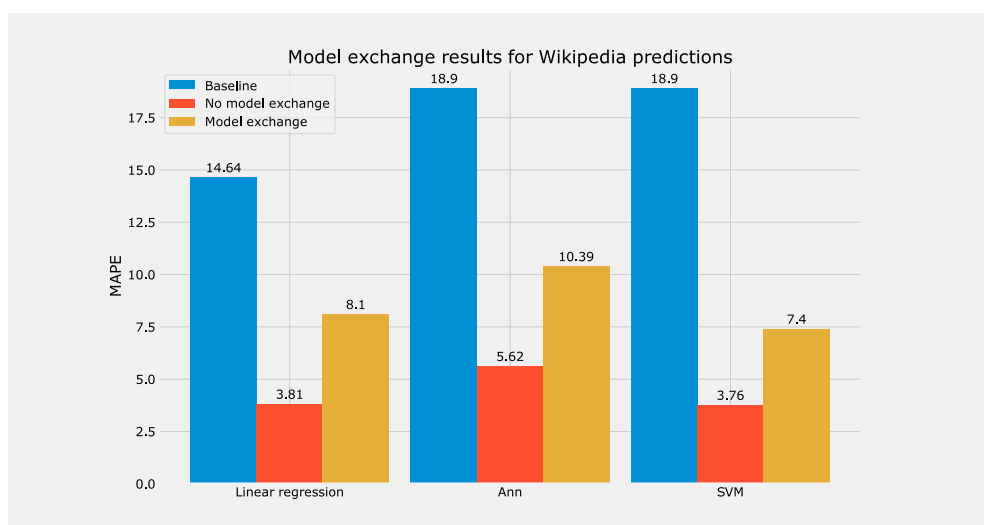


Figure 6.8: Wikipedia prediction results overview for basic dataset

As seen, through the application of a granularity and time window in conjunction with the uniformization of the datasets, it is possible to create a generic enough model that can cope with datasets of a different load scenario.

## EDP prediction

Identical to Wikipedia prediction, an overview was also made for the EDP prediction, Table 6.4. The different model types and testing approaches were identical to the ones used in Wikipedia.

Table 6.4: Model exchange results overview for EDP prediction with linear regression, ANN and SVM on basic dataset

Linear regression results					
Model type	Avg. $\bar{R}^2$	Overfitting	MAPE(Ampere)	MAE(Ampere)	RMSE(Ampere)
Baseline	0.69078	0.1640	21.35	1029.38	1283.96
EDP prediction with EDP trained model	0.99209	0.0011	7.81	12.07	16.53
EDP prediction with Wikipedia trained model	0.97017	0.0021	8.93	0.03	0.04

ANN results					
Model type	Avg. $\bar{R}^2$	Overfitting	MAPE(Ampere)	MAE(Ampere)	RMSE(Ampere)
Baseline	0.68161	0.1751	26.77	1723.47	2191.36
EDP prediction with EDP trained model	0.98212	0.0025	8.62	18.09	25.06
EDP prediction with Wikipedia trained model	0.93448	0.0046	11.71	0.04	0.06

SVM results					
Model type	Avg. $\bar{R}^2$	Overfitting	MAPE(Ampere)	MAE(Ampere)	RMSE(Ampere)
Baseline	0.69796	0.1654	22.64	1558.93	1678.94
EDP prediction with EDP trained model	0.99192	0.0011	7.76	11.93	16.73
EDP prediction with Wikipedia trained model	0.97005	0.0021	8.83	0.03	0.04

As seen in the table, the baseline model was again the worst model type. However, different to what happened when predicting Wikipedia's load, on the EDP scenario, model results were very similar with and without model exchange. Nevertheless, results vastly outperformed the baseline model. As can also be verified, the models created with and without model exchange have a very high  $\bar{R}^2$  and low overfitting, a good indicator of the high level of generalization that the models possess. Figure 6.9 shows the different MAPE values per learning method and model type. The similarity in results with and without model exchange is clearly visible, another indicator of the high level of generalization that the approach developed and applied in this work creates.

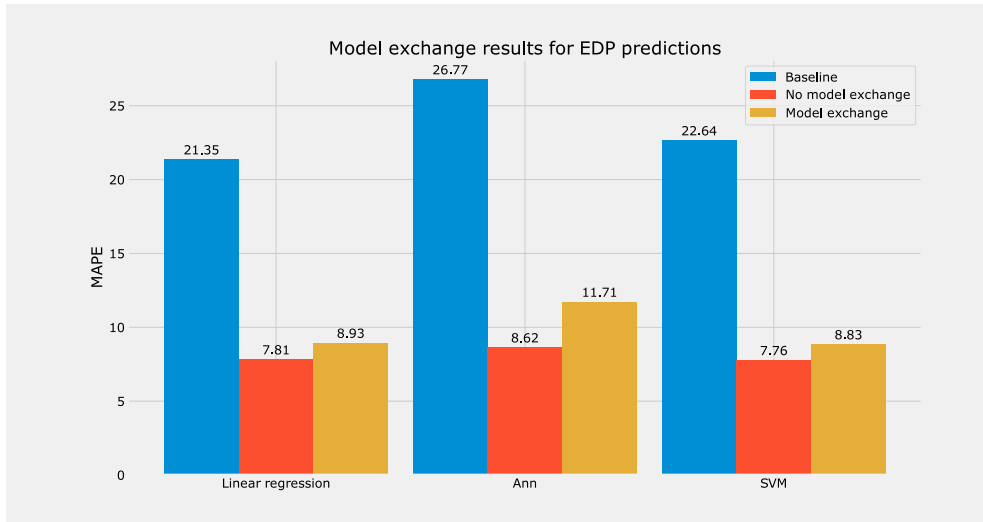


Figure 6.9: EDP prediction results overview for basic dataset

## 6.4.2 Enhanced dataset result analysis

Lastly, the same testing approach is taken for the uniformed enhanced dataset. As mentioned above, this dataset contains the load, average load and standard deviation of the load.

### Wikipedia prediction

Table 6.5 presents the results overview for the Wikipedia prediction with the enhanced dataset.

Table 6.5: Model exchange results overview for Wikipedia prediction with linear regression, ANN and SVM on enhanced dataset

#### Linear regression results

Model type	Avg. $\bar{R}^2$	Overfitting	MAPE(Gb)	MAE(Gb)	RMSE(Gb)
Baseline	0.81351	0.0352	19.87	1094.51	2001.82
Wikipedia prediction with Wikipedia trained model	0.98842	0.0012	1.07	2.25	9.87
Wikipedia prediction with EDP trained model	0.95417	0.0047	9.48	0.01	0.02

#### ANN results

Model type	Avg. $\bar{R}^2$	Overfitting	MAPE(Gb)	MAE(Gb)	RMSE(Gb)
Baseline	0.75616	0.0562	32.09	821.86	5325.50
Wikipedia prediction with Wikipedia trained model	0.98504	0.0015	1.44	3.00	11.26
Wikipedia prediction with EDP trained model	0.99120	0.0009	8.06	0.00	0.01

#### SVM results

Model type	Avg. $\bar{R}^2$	Overfitting	MAPE(Gb)	MAE(Gb)	RMSE(Gb)
Baseline	0.81342	0.0349	16.92	1511.21	2352.37
Wikipedia prediction with Wikipedia trained model	0.98923	0.0011	0.50	0.57	9.52
Wikipedia prediction with EDP trained model	0.96241	0.0039	6.47	0.00	0.02

As expected, the baseline model was again the worst performing model type, with some learning methods performing worse than their basic counterpart. The model without model exchange clearly outperforms the remaining model types. Nonetheless, as seen through the  $\bar{R}^2$  and overfitting values, a high model generalization was again guaranteed. Figure 6.10 depicts in a simple manner how different model types differ immensely in terms of MAPE.

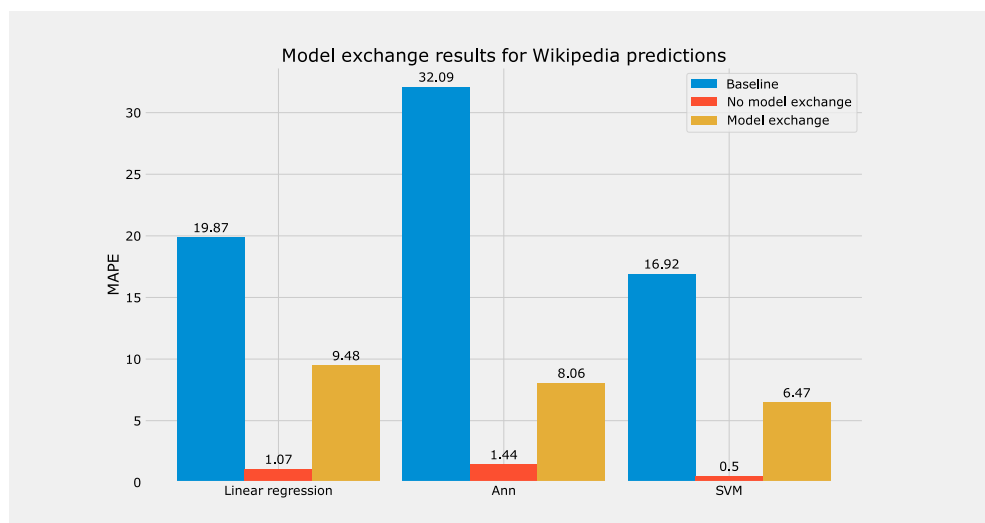


Figure 6.10: Wikipedia prediction results overview for enhanced dataset

## EDP prediction

To conclude the model exchange case study, the uniformed enhanced dataset for EDP was also tested, Table 6.6 illustrates the results overview.

Table 6.6: Model exchange results overview for EDP prediction with linear regression, ANN and SVM on enhanced dataset

### Linear regression results

Model type	Avg. $\bar{R}^2$	Overfitting	MAPE(Ampere)	MAE(Ampere)	RMSE(Ampere)
Baseline	0.79494	0.0465	14.09	1011.38	1014.76
EDP prediction with EDP trained model	0.84711	0.0157	2.86	18733.66	3900340.06
EDP prediction with Wikipedia trained model	0.98842	0.0012	0.95	0.01	0.03

### ANN results

Model type	Avg. $\bar{R}^2$	Overfitting	MAPE(Ampere)	MAE(Ampere)	RMSE(Ampere)
Baseline	0.81342	0.0316	20.63	1587.23	1759.93
EDP prediction with EDP trained model	0.92236	0.0080	2.38	16126.55	2773813.57
EDP prediction with Wikipedia trained model	0.98532	0.0015	1.91	0.01	0.03

### SVM results

Model type	Avg. $\bar{R}^2$	Overfitting	MAPE(Ampere)	MAE(Ampere)	RMSE(Ampere)
Baseline	0.79114	0.0461	16.30	1170.12	1175.09
EDP prediction with EDP trained model	0.92500	0.0077	2.23	15410.69	2692709.58
EDP prediction with Wikipedia trained model	0.98923	0.0011	0.11	0.00	0.03

This last test run was the most surprising, as all learning methods achieved their best results when making EDP load predictions on a Wikipedia trained model. Results in terms of generalization were also the best with the Wikipedia trained model. With regards to MAPE, SVM was able to achieve results that rival its absolute best model of only 0.05 MAPE. Figure 6.11 depicts the astonishing performance of the Wikipedia trained model when compared to the remaining model types.

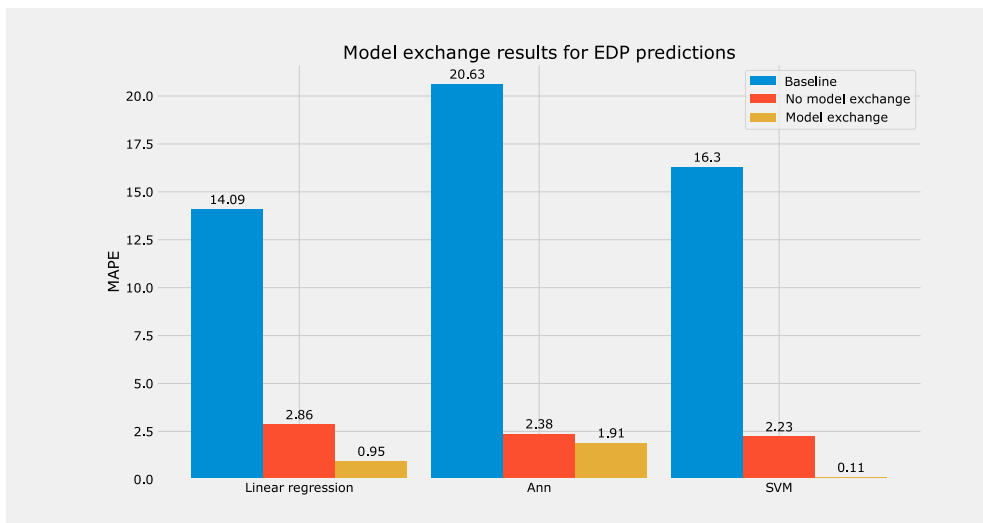


Figure 6.11: EDP prediction results overview for enhanced dataset

## 6.5 Conclusion

The concluded chapter presented and analysed the test results of the energy load forecast case study, along with the case study of model exchange. Result analysis in both case studies confirmed that the addition of extra attributes to the basic and enhanced dataset, increased significantly the learning methods performances, as well as, their generalization capability. Similar to server load prediction, the enhanced dataset outperformed the basic dataset in most occasions. With regards to time window and granularity, identical to server load prediction, a lower granularity presented better results in most cases, while time windows up to 24 for enhanced dataset and 48 for basic dataset were the best option. Lastly, the best performing learning method was again SVM, with linear regression coming close second. ANN was the worst performing learning method from an average point of view. In terms of model exchange, the Wikipedia trained models were the best performing when using the uniformed enhanced dataset, either for Wikipedia predictions or EDP predictions. Results of this last case study once again confirmed the high level of generalization that the dataset transformation procedure integrates in the model. Further testing, evaluation and confirmation of this outcome is found in the papers published.



# Chapter 7

## Conclusions and Future work

In this work, a new method for improving model performance and generalization by adding extra temporal information to the datasets was proposed. Different approaches in the field of load forecasting were studied and presented. Two different and independent forecasting solutions emerged from said work. These solutions were based in the above mentioned integration of temporal information to a dataset through granularity and time window. The results achieved by these solutions can challenge other approaches of similar problem spectra. The published per-reviewed papers provide a deeper analysis and confirm of this statement.

For the web server/datacenter solution, real data from the Wikipedia servers, publicly made available by Wikipedia through the Wikimedia project, was used for model training and testing. The created models were able to predict the hourly load on Wikipedia's servers with a MAPE on average below the 0.15 mark, for the months of January to August of the year 2016. For the yearly dataset, a MAPE on average below the 0.75 mark for hourly load prediction was achieved. All the created Wikipedia models were able to achieve these results while keeping a high level of generalisation.

For the energy problem, the data used was provided by EDP and contained real life data regarding the energy consumption levels of the city of Leiria for the month of April 2016. The created models predicted the energy consumption levels on an hourly basis. Similar to the Wikipedia case study, research regarding this specific dataset is not publicly available, hence why results comparison can be just made to problems of similar problem spectrum but with different data. Studies regarding these results comparisons can also be found on the published papers. The resulting model was able to predict the next hour consumption levels with a MAPE of only 0.05, while keeping a level of overfitting extremely low and consequently a high level of generalization. On both case studies, a time window up to 24 and lower granularity achieves the best results.

Each final model was tested as well, with data from the opposite field. This approach had the goal to analyse how both models of load forecasting would behave when the environment in which they were used was switched. Results concluded that the level of generalization that the integration of temporal information gave to the models was extremely high, with the Wikipedia trained enhanced dataset applied to EDP, surpassing the trained and tested enhanced EDP model.

Throughout the developed work , a set of tools were needed to facilitate and accomplish several tasks. These tools were later on adapted and published in the form of API for other researchers to use.

With regards to future work, switching from CPU usage to GPU usage would be the first step to take, since the actual procedure contains models that took between one and two months of training time. Increasing the range of different values for granularity and time window would also be important, as it could add robustness and validity to the applied approach. Further optimization of model parameters may also result in better model performance. The addition of different learning methods would most certainly provide a better understanding of how well the applied approach overall performs. Lastly, the application of the developed approach to different problems/case studies, as well as, cross-testing the different trained models, would be advantageous.

# Bibliography

- [1] A. Buszta and J. Mazurkiewicz, "Climate changes prediction system based on weather big data visualisation," in *Theory and Engineering of Complex Systems and Dependability*. Springer, 2015, pp. 75–86.
- [2] C. Chatfield, *Time-series forecasting*. Chapman and Hall/CRC, 2000.
- [3] S. A. hady Soliman, *Electrical load forecasting : modeling and model construction*. Oxford: Elsevier Inc., 2010.
- [4] D. C. Park, M. El-Sharkawi, R. Marks, L. Atlas, and M. Damborg, "Electric load forecasting using an artificial neural network," *IEEE transactions on Power Systems*, vol. 6, no. 2, pp. 442–449, 1991.
- [5] S. Chaidaroon, J. Y. Kim, and J. Seo, "Server load prediction."
- [6] D. J. Dalrymple, "Sales forecasting practices: Results from a united states survey," *International Journal of Forecasting*, vol. 3, no. 3-4, pp. 379–391, 1987.
- [7] A. Hipni, A. El-shafie, A. Najah, O. A. Karim, A. Hussain, and M. Mukhlisin, "Daily forecasting of dam water levels: comparing a support vector machine (svm) model with adaptive neuro fuzzy inference system (anfis)," *Water resources management*, vol. 27, no. 10, pp. 3803–3823, 2013.
- [8] E. I. Vlahogianni, J. C. Golias, and M. G. Karlaftis, "Short-term traffic forecasting: Overview of objectives and methods," *Transport reviews*, vol. 24, no. 5, pp. 533–557, 2004.
- [9] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM, 2012, p. 7.
- [10] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *Journal of grid computing*, vol. 12, no. 4, pp. 559–592, 2014.
- [11] A. Hooper, "Green computing," *Communication of the ACM*, vol. 51, no. 10, pp. 11–13, 2008.
- [12] L. Di Persio, A. Cecchin, and F. Cordini, "Novel approaches to the energy load unbalance forecasting in the italian electricity market," *Journal of Mathematics in Industry*, vol. 7, no. 1, p. 5, 2017.

- [13] P. Wallström, “Evaluation of forecasting techniques and forecast errors: with focus on intermittent demand,” Ph.D. dissertation, Luleå tekniska universitet, 2009.
- [14] K. B. Song, Y. S. Baek, D. H. Hong, and G. Jang, “Short-term load forecasting for the holidays using fuzzy linear regression method,” *IEEE transactions on power systems*, vol. 20, no. 1, pp. 96–101, 2005.
- [15] J. V. Beck and K. J. Arnold, *Parameter estimation in engineering and science*. James Beck, 1977.
- [16] E. A. Feinberg and D. Genethliou, “Load forecasting,” in *Applied mathematics for restructured electric power systems*. Springer, 2005, pp. 269–285.
- [17] G. Gross and F. D. Galiana, “Short-term load forecasting,” *Proceedings of the IEEE*, vol. 75, no. 12, pp. 1558–1573, 1987.
- [18] E. Kacapyr, *Economic forecasting: the state of the art*. ME Sharpe, 1996.
- [19] H. Gürkök, G. Hakvoort, and M. Poel, *Evaluating user experience with respect to user expectations in brain-computer interface games*. na, 2011.
- [20] H. Chen, C. A. Canizares, and A. Singh, “Ann-based short-term load forecasting in electricity markets,” in *Power Engineering Society Winter Meeting, 2001. IEEE*, vol. 2. IEEE, 2001, pp. 411–415.
- [21] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, “Arima models to predict next-day electricity prices,” *IEEE transactions on power systems*, vol. 18, no. 3, pp. 1014–1020, 2003.
- [22] B. Choi, *ARMA model identification*. Springer Science & Business Media, 2012.
- [23] C. Kongcharoen and T. Kruangpradit, “Autoregressive integrated moving average with explanatory variable (arimax) model for thailand export,” in *33rd International Symposium on Forecasting, South Korea, 2013*, pp. 1–8.
- [24] H. J. Bierens, “Armax model specification testing, with an application to unemployment in the netherlands,” *Journal of Econometrics*, vol. 35, no. 1, pp. 161–190, 1987.
- [25] H. T. Yang, C. M. Huang, and C. L. Huang, “Identification of armax model for short term load forecasting: An evolutionary programming approach,” in *Power Industry Computer Application Conference, 1995. Conference Proceedings., 1995 IEEE*. IEEE, 1995, pp. 325–330.
- [26] R. M. Golden, “Exploring the diversity of artificial neural network architectures,” *Journal of Mathematical Psychology*, vol. 41, no. 3, pp. 287–292, 1997.
- [27] V. N. Vapnik, “An overview of statistical learning theory,” *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.

- [28] E. A. Feinberg, J. Hajagos, and D. Genethliou, "Statistical load modeling," in *Proceedings of the 7th IASTED International Multi-Conference: Power and Energy Systems*, 2003, pp. 88–91.
- [29] R. H. Teunter and L. Duncan, "Forecasting intermittent demand: a comparative study," *Journal of the Operational Research Society*, vol. 60, no. 3, pp. 321–329, 2009.
- [30] E. S. Gardner Jr, "Exponential smoothing: The state of the art—part ii," *International journal of forecasting*, vol. 22, no. 4, pp. 637–666, 2006.
- [31] A. A. Syntetos and J. E. Boylan, "The accuracy of intermittent demand estimates," *International Journal of forecasting*, vol. 21, no. 2, pp. 303–314, 2005.
- [32] S. A. DeLurgio, *Forecasting principles and applications*. Irwin Professional Publishing, 1998.
- [33] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks for short-term load forecasting: A review and evaluation," *IEEE Transactions on power systems*, vol. 16, no. 1, pp. 44–55, 2001.
- [34] N. Jia, R. Yokoyama, Y. Zhou, and Z. Gao, "A flexible long-term load forecasting approach based on new dynamic simulation theory—gsim," *International journal of electrical power & energy systems*, vol. 23, no. 7, pp. 549–556, 2001.
- [35] K. Nagasaka and M. Al Mamun, "Long-term peak demand prediction of 9 japanese power utilities using radial basis function networks," in *Power Engineering Society General Meeting, 2004. IEEE*. IEEE, 2004, pp. 315–322.
- [36] S. Parkpoom, G. Harrison, and J. Bialek, "Climate change impacts on electricity demand," in *Universities Power Engineering Conference, 2004. UPEC 2004. 39th International*, vol. 3. IEEE, 2004, pp. 1342–1346.
- [37] F. Cavallaro, "Electric load analysis using an artificial neural network," *International Journal of Energy Research*, vol. 29, no. 5, pp. 377–392, 2005.
- [38] M. Falvo, R. Lamedica, and A. Prudenzi, "Meteorological parameters influence for medium term load forecasting," in *2005/2006 IEEE/PES Transmission and Distribution Conference and Exhibition*, 2006, pp. 1296–1301.
- [39] H. Y. Yang, H. Ye, G. Wang, J. Khan, and T. Hu, "Fuzzy neural very-short-term load forecasting based on chaotic dynamics reconstruction," *Chaos, Solitons & Fractals*, vol. 29, no. 2, pp. 462–469, 2006.
- [40] N. Kandil, R. Wamkeue, M. Saad, and S. Georges, "An efficient approach for short term load forecasting using artificial neural networks," *International Journal of Electrical Power & Energy Systems*, vol. 28, no. 8, pp. 525–530, 2006.
- [41] S. Yang and N. Li, "Power demand forecast based on optimized neural networks by improved genetic algorithm," in *Machine Learning and Cybernetics, 2006 International Conference on*. IEEE, 2006, pp. 2877–2881.

- [42] D. Asber, S. Lefebvre, M. Saad, and C. Desbiens, "Modeling of distribution loads for short and medium-term load forecasting," in *Power Engineering Society General Meeting, 2007. IEEE*. IEEE, 2007, pp. 1–5.
- [43] B. Xue and J. Geng, "Dynamic transverse correction method of middle and long term energy forecasting based on statistic of forecasting errors," in *IPEC, 2012 Conference on Power & Energy*. IEEE, 2012, pp. 253–256.
- [44] F. Cavallaro, "Electric load analysis using an artificial neural network," *International Journal of Energy Research*, vol. 29, no. 5, pp. 377–392, 2005.
- [45] S. Khatoon, A. K. Singh *et al.*, "Effects of various factors on electric load forecasting: An overview," in *Power India International Conference (PIICON), 2014 6th IEEE*. IEEE, 2014, pp. 1–5.
- [46] M. Ghiasi, M. Irani Jam, M. Teimourian, H. Zarrabi, and N. Yousefi, "A new prediction model of electricity load based on hybrid forecast engine," *International Journal of Ambient Energy*, pp. 1–8, 2017.
- [47] P. Reyneau, "Predicting load on residence circuits," *Electrical World*, vol. 71, no. 19, pp. 969–971, 1918.
- [48] C. Plumley and E. F. Seaman, "Cost comparisons decide houston network," *Electrical World*, vol. 95, no. 24, pp. 1211–1213, 1930.
- [49] M. M. Samuels, "System plan alternatives visualized for executives," *Electrical World*, vol. 98, no. 18, pp. 778–782, 1931.
- [50] G. Read, "Forecasting the revival phase of kilowatt-hour consumption," *Electrical World*, vol. 102, no. 25, pp. 780–784, 1933.
- [51] P. Jennings, "Predicting load with no-growth curve," *Electrical West*, vol. 85, no. 6, pp. 42–44, 1940.
- [52] M. Hatch, "Estimate secondary loads from customer kilowatt-hours," *Electrical West*, vol. 116, no. 4, pp. 32–33, 1941.
- [53] A. Davison, "The weatherman can help in loadforecasting for utilities," *Edison Elec. Inst. Bul.*, vol. 12, no. 11, pp. 361–363, 1944.
- [54] P. Schiller, "Relation between daylight illumination and system load," *with a Mathematical Appendix by Nl Johnson, ERA Report, Ref. K*, vol. 115, p. 1945, 1945.
- [55] H. P. Seelye, "Trend prediction important in planning," *with a Mathematical Appendix by Nl Johnson, ERA Report, Ref. K*, pp. 79–81, 1946.
- [56] R. Lacy, "Variations of the winter means of temperature, wind speed and sunshine, and their effect on the heating requirements of a house," *Meteorological magazine*, vol. 80, p. 161, 1951.

- [57] C. Williams and F. Leslie, "Load forecasts reflect weather," *Electrical World*, vol. 6, pp. 86–88, 1953.
- [58] A. Thomas and J. Drummond, "The role of weather correction in load forecasting," *Edison Electrical Inst., Bul.*, vol. 21, no. 8, pp. 306–310, 1953.
- [59] F. W. Brooks, "load forecasting—a method based on economic factors," *Electrical World*, vol. 139, no. 2, pp. 63–65, 1953.
- [60] D. Gillies, B. Bernholtz, and P. Sandiford, "A new approach to forecasting daily peak loads," *Transactions of the American Institute of Electrical Engineers. Part III: Power Apparatus and Systems*, vol. 75, no. 3, pp. 382–387, 1956.
- [61] M. Davies, "The relationship between weather and electricity demand," *Proceedings of the IEE-Part C: Monographs*, vol. 106, no. 9, pp. 27–37, 1959.
- [62] E. Vedere, "Further work in connection with the influence of weather conditions on the paris and paris region networks'," in *International Union of Producers and Distributors of Electrical Energy, Tenth Congress*, vol. 7, 1955, pp. 1–12.
- [63] H. Nissen, "Influence of weather conditions on the maximum demand in the hamburg district," in *International Union of Producers and Distributors of Electrical Energy, Tenth Congress*, vol. 7, 1955, pp. 1–11.
- [64] W. J. A. Kuipers and A. W. Zuidweg, "The influence of atmospheric conditions on the consumption of electrical energy in the netherlands," in *International Union of Producers and Distributors of Electrical Energy, Tenth Congress*, vol. 7, 1955, pp. 1–6.
- [65] L. Perrin, "Effect of temperature on the consumption of electrical energy," in *International Union of Producers and Distributors of Electrical Energy, Tenth Congress*, vol. 7, 1955, pp. 1–3.
- [66] R. Hooke, "Forecasting the demand for electricity," *Electrical Engineering*, vol. 75, no. 2, pp. 132–132, 1956.
- [67] W. L. Carey, "Short-range peak system planning," *Eltric Light and Power*, pp. 44–47, 1961.
- [68] R. E. Kalman, "A. lazzari," *Electrical World*, vol. 159, pp. 38–40, 1962.
- [69] E. D. Farmer, "A method of prediction for non-stationary processes and its application to the problem of load estimation," *Trans. IFAC*, pp. 47–54, 1963.
- [70] S. Darlington, "Linear least-squares smoothing and prediction, with applications," *Bell Labs Technical Journal*, vol. 37, no. 5, pp. 1221–1294, 1958.
- [71] G. K. Chen and P. R. Winters, "Forecasting peak demand for an electric utility with a hybrid exponential model," *Management Science*, vol. 12, no. 12, pp. B–531, 1966.

- [72] G. Heinemann, D. Nordmian, and E. Plant, "The relationship between summer weather and summer loads-a regression analysis," *IEEE Transactions on Power Apparatus and Systems*, no. 11, pp. 1144–1154, 1966.
- [73] E. Farmer and M. Potton, "Development of online load-prediction techniques with results from trials in the south-west region of the cegb," in *Proceedings of the Institution of Electrical Engineers*, vol. 115, no. 10. IET, 1968, pp. 1549–1558.
- [74] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [75] P. Vähäkylä, E. Hakonen, and P. Leman, "Short-term forecasting of grid load using box-jenkins techniques," *International Journal of Electrical Power & Energy Systems*, vol. 2, no. 1, pp. 29–34, 1979.
- [76] K. N. Stanton and P. C. Gupta, "Forecasting annual or seasonal peak demand in electric utility systems," *IEEE Transactions on Power Apparatus and Systems*, no. 5, pp. 951–959, 1970.
- [77] P. C. Gupta and K. Yamada, "Adaptive short-term forecasting of hourly loads using weather information," *IEEE Transactions on Power Apparatus and Systems*, no. 5, pp. 2085–2094, 1972.
- [78] F. Galiana and F. Schweppe, "Weather dependent probabilistic model for short-term load forecasting," in *Ieee Transactions On Power Apparatus And Systems*, no. 4. Ieee-inst Electrical Electronics Engineers Inc 345 E 47Th St, New York, Ny 10017-2394, 1972, p. 1728.
- [79] J. Davey, J. Saacks, G. Cunningham, and K. Priest, "Practical application of weather sensitive load forecasting to system planning," *IEEE Transactions on Power Apparatus and Systems*, no. 3, pp. 971–977, 1973.
- [80] S. Corpening, N. Reppen, and R. Ringlee, "Experience with weather sensitive load models for short and long-term forecasting," *IEEE Transactions on Power Apparatus and systems*, no. 6, pp. 1966–1972, 1973.
- [81] C. Asbury, "Weather load model for electric demand and energy forecasting," *IEEE Transactions on Power Apparatus and Systems*, vol. 94, no. 4, pp. 1111–1116, 1975.
- [82] R. P. Thompson, "Weather sensitive electric demand and energy analysis on a large geographically diverse power system application to short term hourly electric demand forecasting," *IEEE Transactions on Power Apparatus and Systems*, vol. 95, no. 1, pp. 385–393, 1976.
- [83] W. Christiaanse, "Short-term load forecasting using general exponential smoothing," *IEEE Transactions on Power Apparatus and Systems*, no. 2, pp. 900–911, 1971.
- [84] K. N. Stanton, "Medium-range, weekly and seasonal peak demand forecasting by probability methods," *IEEE Transactions on Power Apparatus and Systems*, no. 3, pp. 1183–1189, 1971.



- [85] P. Gupta, "A stochastic approach to peak power-demand forecasting in electric utility systems," *IEEE Transactions on Power Apparatus and Systems*, no. 2, pp. 824–832, 1971.
- [86] K. Sharma and M. Ak, "Application of adaptive estimation technique for short-term load prediction," in *Ieee Transactions On Power Apparatus And Systems*, no. 6. Ieee-inst Electrical Electronics Engineers Inc 345 E 47Th St, New York, Ny 10017-2394, 1972, p. 2258.
- [87] S. Vemuri, D. Hill, and R. Balasubramanian, "Load forecasting using stochastic models," in *Proceeding of the 8th power industrial computing application conference*, vol. 1, 1973, pp. 31–37.
- [88] K. Sharma and A. Mahalanabis, "Recursive short-term load-forecasting algorithm," in *Proceedings of the Institution of Electrical Engineers*, vol. 121, no. 1. IET, 1974, pp. 59–62.
- [89] N. Prasad, J. M. Perkins, and G. Nesgos, "A markov process applied to forecasting. part ii. the demand for electricity," in *Proceedings of the Power Engineering Society Summer Meeting, Vancouver, July 1973*, 1974.
- [90] E. Spumberg, "Load forecasting techniques for power-system planning," in *Ieee Transactions On Power Apparatus And Systems*, no. 6. Ieee-inst Electrical Electronics Engineers Inc 345 E 47Th St, New York, Ny 10017-2394, 1974, pp. 1734–1734.
- [91] K. Sharma, "An application of nonlinear adaptive estimation theory in short-term load prediction," *IEEE Proc. of PES, SM, Anaheim (1974-7)*, 1974.
- [92] A. Ferrandiz, M. Penel, and Y. Pioger, "Short, medium and long-term load forecasting models," in *Ieee Transactions On Power Apparatus And Systems*, vol. 94, no. 6. Ieee-inst Electrical Electronics Engineers Inc 345 E 47Th St, New York, Ny 10017-2394, 1975, pp. 1916–1916.
- [93] K. Srinivasan and R. Pronovost, "Short term load forecasting using multiple correlation models," *IEEE Transactions on Power Apparatus and Systems*, vol. 94, no. 5, pp. 1854–1858, 1975.
- [94] M. Davies, P. Matthewman, H. Nicholson, E. Farmer, and M. Potton, "Techniques for load prediction in the electricity-supply industry and developments of online load-prediction techniques with results from trials in the southwest region of the cegb," in *Proceedings of the Institution of Electrical Engineers*, vol. 117, no. 4. IET, 1970, pp. 823–824.
- [95] M. Sachdev and S. Ibrahim, "Short-term on-line load forecasting," in *Ieee Transactions On Power Apparatus And Systems*, no. 6. Ieee-inst Electrical Electronics Engineers Inc 345 E 47Th St, New York, Ny 10017-2394, 1972, p. 2257.
- [96] D. Lijesen and J. Rosing, "Adaptive forecasting of hourly loads based on load measurements and weather information," *IEEE Transactions on Power Apparatus and Systems*, no. 4, pp. 1757–1767, 1971.

- [97] T. Takenawa, A. M. Schneider, and D. A. Schiffman, "A computer program for 24-hour electric utility load forecasting," *Energy*, vol. 5, no. 7, pp. 571–585, 1979.
- [98] G. Friedlander, "Power: Planning for a brighter future: Modeling and simulation, load forecasting, frequency control, stability, security, and reliability are important factors," *IEEE Spectrum*, vol. 14, no. 3, pp. 61–68, 1977.
- [99] N. D. Uri, "Forecasting peak system load using a combined time series and econometric model," *Applied Energy*, vol. 4, no. 3, pp. 219–227, 1978.
- [100] E. Menge, V. Wilreker, and J. Northcote-Green, "Electrical loads can be forecasted for distribution planning.[time-series and multivariate spatial analyses]," in *Proc. Am. Power Conf.:(United States)*, vol. 39, no. CONF-770403-. Westinghouse Electric Corp., Pittsburgh, 1977.
- [101] N. D. URI and J. S. MAYBEE, "Forecasting the load duration curve using box-jenkins time series analysis," *Engineering Optimization*, vol. 3, no. 4, pp. 193–199, 1978.
- [102] S. Maybee and N. Uri, "Time series forecasting of utility load duration curves," *Canadian Electrical Engineering Journal*, vol. 4, no. 1, pp. 4–8, 1979.
- [103] B. De Martino, G. Fusco, E. Mariani, R. Randino, and P. Ricci, "A medium and short term load forecasting model for electrical industry," in *Power Industry Computer Applications Conference, 1979. PICA-79. IEEE Conference Proceedings.* IEEE, 1979, pp. 186–191.
- [104] J. C. Sweeney, "Energy modeling and forecasting: Implications for strategic planning," *Business Economics*, pp. 21–27, 1978.
- [105] T. Peng, N. Hubele, and G. Karady, "Conceptual approach to the application of neural network for short-term load forecasting," in *Circuits and Systems, 1990., IEEE International Symposium on.* IEEE, 1990, pp. 2942–2945.
- [106] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.
- [107] H. White, "Learning in artificial neural networks: A statistical perspective," *Neural computation*, vol. 1, no. 4, pp. 425–464, 1989.
- [108] K. Lee, Y. Cha, and J. Park, "Artificial neural network methodology for short-term load forecasting," in *NSF Workshop on Artificial Neural Network Methodology in Power System Engineering.* Clemson University, SC, 1990, pp. 9–10.
- [109] D. Prabhakar, S. Viswanathan, M. Raoot, P. Pentayya, and N. Nallarsan, "Implementation aspects of artificial neural networks based short term load forecasting method-experience of wrldc," in *National power system conference, Indian Institute of Technology, Kanpur*, 1986, pp. 355–359.

- [110] T. Peng, N. Hubele, and G. Karady, "Advancement in the application of neural networks for short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 7, no. 1, pp. 250–257, 1992.
- [111] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavior science," *Unpublished Doctoral Dissertation, Harvard University*, 1974.
- [112] A. Bakirtzis, V. Petridis, S. Kiartzis, M. Alexiadis, and A. Maissis, "A neural network short term load forecasting model for the greek power system," *IEEE Transactions on power systems*, vol. 11, no. 2, pp. 858–863, 1996.
- [113] A. Khotanzad, R. Afkhami-Rohani, T.-L. Lu, A. Abaye, M. Davis, and D. J. Maratukulam, "Annstlf-a neural-network-based electric load forecasting system," *IEEE Transactions on Neural networks*, vol. 8, no. 4, pp. 835–846, 1997.
- [114] A. Khotanzad, R. Afkhami-Rohani, and D. Maratukulam, "Annstlf-artificial neural network short-term load forecaster-generation three," *IEEE Transactions on Power Systems*, vol. 13, no. 4, pp. 1413–1422, 1998.
- [115] A. D. Papalexopoulos, S. Hao, and T.-M. Peng, "An implementation of a neural network based load forecasting model for the ems," *IEEE transactions on Power Systems*, vol. 9, no. 4, pp. 1956–1962, 1994.
- [116] T. Chow and C.-T. Leung, "Nonlinear autoregressive integrated neural network model for short-term load forecasting," *IEE Proceedings-Generation, Transmission and Distribution*, vol. 143, no. 5, pp. 500–506, 1996.
- [117] H. Mori and N. Kosemura, "Optimal regression tree based rule discovery for short-term load forecasting," in *Power Engineering Society Winter Meeting, 2001. IEEE*, vol. 2. IEEE, 2001, pp. 421–426.
- [118] S. E. Skarman, M. Georgiopoulos, and A. J. Gonzalez, "Short-term electrical load forecasting using a fuzzy artmap neural network," in *Applications and Science of Computational Intelligence*, vol. 3390. International Society for Optics and Photonics, 1998, pp. 181–192.
- [119] B. Schölkopf and A. Smola, "Support vector machines and kernel algorithms," pp. 5328–5335, 2002.
- [120] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.
- [121] J. Fan and J. McDonald, "A real-time implementation of short-term load forecasting for distribution power systems," *IEEE Transactions on Power Systems*, vol. 9, no. 2, pp. 988–994, 1994.
- [122] P. F. Pai and W. C. Hong, "Support vector machines with simulated annealing algorithms in electricity load forecasting," *Energy Conversion and Management*, vol. 46, no. 17, pp. 2669–2688, 2005.

- [123] Y. c. Li, T. j. Fang, and E. k. Yu, "Study of support vector machines for short-term load forecasting [j]," *Proceedings of the Csee*, vol. 6, p. 010, 2003.
- [124] M. Mohandes, "Support vector machines for short-term electrical load forecasting," *International Journal of Energy Research*, vol. 26, no. 4, pp. 335–345, 2002.
- [125] B. Dong, C. Cao, and S. E. Lee, "Applying support vector machines to predict building energy consumption in tropical region," *Energy and Buildings*, vol. 37, no. 5, pp. 545–553, 2005.
- [126] W. C. Hong, "Electric load forecasting by support vector model," *Applied Mathematical Modelling*, vol. 33, no. 5, pp. 2444–2454, 2009.
- [127] E. E. Elattar, J. Goulermas, and Q. H. Wu, "Electric load forecasting based on locally weighted support vector regression," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 4, pp. 438–447, 2010.
- [128] M.-G. Zhang, "Short-term load forecasting based on support vector machines regression," in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, vol. 7. IEEE, 2005, pp. 4310–4314.
- [129] Y. Wang, Q. Xia, and C. Kang, "Secondary forecasting based on deviation analysis for short-term load forecasting," *IEEE Transactions on Power Systems*, vol. 26, no. 2, pp. 500–507, 2011.
- [130] M. Espinoza, J. A. Suykens, R. Belmans, and B. De Moor, "Electric load forecasting," *IEEE Control Systems*, vol. 27, no. 5, pp. 43–57, 2007.
- [131] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in neural information processing systems*, 1997, pp. 155–161.
- [132] P. F. Pai and W. C. Hong, "Forecasting regional electricity load based on recurrent support vector machines with genetic algorithms," *Electric Power Systems Research*, vol. 74, no. 3, pp. 417–425, 2005.
- [133] M. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Eighth Annual Conference of the Cognitive Science Society, 1986*, 1986, pp. 513–546.
- [134] P. F. Pai and W. C. Hong, "Support vector machines with simulated annealing algorithms in electricity load forecasting," *Energy Conversion and Management*, vol. 46, no. 17, pp. 2669–2688, 2005.
- [135] R. Weron, "Electricity price forecasting: A review of the state-of-the-art with a look into the future," *International journal of forecasting*, vol. 30, no. 4, pp. 1030–1081, 2014.

- [136] B.-M. Hodge, A. Florita, K. Orwig, D. Lew, and M. Milligan, “Comparison of wind power and load forecasting error distributions: Preprint,” National Renewable Energy Laboratory (NREL), Golden, CO., Tech. Rep., 2012.
- [137] S. Aggarwal *et al.*, “Forecasting issues in present day power system,” 2017.
- [138] G. Irisarri, S. Widergren, and P. Yehsakul, “On-line load forecasting for energy control center application,” *IEEE Transactions on Power Apparatus and Systems*, no. 1, pp. 71–78, 1982.
- [139] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, “A review of auto-scaling techniques for elastic applications in cloud environments,” *Journal of grid computing*, vol. 12, no. 4, pp. 559–592, 2014.
- [140] A. Hooper, “Green computing,” *Communication of the ACM*, vol. 51, no. 10, pp. 11–13, 2008.
- [141] D. Ardagna, S. Casolari, M. Colajanni, and B. Panicucci, “Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems,” *Journal of Parallel and Distributed Computing*, vol. 72, no. 6, pp. 796–808, 2012.
- [142] T. V. T. Duy, Y. Sato, and Y. Inoguchi, “A prediction-based green scheduler for datacenters in clouds,” *Ieice Transactions on Information and Systems*, vol. 94, no. 9, pp. 1731–1741, 2011.
- [143] H. Kim, W. Kim, and Y. Kim, “A pattern-based prediction model for dynamic resource provisioning in cloud environment,” *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 5, no. 10, pp. 1712–1732, 2011.
- [144] C. Santana, J. C. Leite, and D. Mossé, “Power management by load forecasting in web server clusters,” *Cluster Computing*, vol. 14, no. 4, pp. 471–481, 2011.
- [145] D. J. Dean, H. Nguyen, and X. Gu, “Ubl: Unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems,” in *Proceedings of the 9th international conference on Autonomic computing*. ACM, 2012, pp. 191–200.
- [146] Y. Tan, H. Nguyen, Z. Shen, X. Gu, C. Venkatramani, and D. Rajan, “Prepare: Predictive performance anomaly prevention for virtualized cloud systems,” in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 2012, pp. 285–294.
- [147] M. A. Iverson, F. Ozguner, and L. C. Potter, “Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment,” in *Heterogeneous Computing Workshop, 1999.(HCW’99) Proceedings. Eighth*. IEEE, 1999, pp. 99–111.
- [148] N. R. Herbst, N. Huber, S. Kounev, and E. Amrehn, “Self-adaptive workload classification and forecasting for proactive resource provisioning,” *Concurrency and computation: practice and experience*, vol. 26, no. 12, pp. 2053–2078, 2014.

- [149] S. Naseera, G. Rajini, N. A. Prabha, and G. Abhishek, "A comparative study on cpu load predictions in a computational grid using artificial neural network algorithms," *Indian Journal of Science and Technology*, vol. 8, no. 35, 2015.
- [150] Y. Yu, X. Zhan *et al.*, "Server load prediction based on improved support vector machines," in *IT in Medicine and Education, 2008. ITME 2008. IEEE International Symposium on.* IEEE, 2008, pp. 838–842.
- [151] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.
- [152] J. Hamilton, "Cooperative expendable micro-slice servers (cems): low cost, low power servers for internet-scale services," in *Conference on Innovative Data Systems Research (CIDR'09)(January 2009)*. Citeseer, 2009.
- [153] N. Vasić, M. Barisits, V. Salzgeber, and D. Kostic, "Making cluster applications energy-aware," in *Proceedings of the 1st workshop on Automated control for data-centers and clouds*. ACM, 2009, pp. 37–42.
- [154] J. Murty, *Programming amazon web services: S3, EC2, SQS, FPS, and SimpleDB*. O'Reilly Media, Inc., 2008.
- [155] Y. Wadia, *AWS Administration—The Definitive Guide*. Packt Publishing Ltd, 2016.
- [156] T. Mather, S. Kumaraswamy, and S. Latif, *Cloud security and privacy: an enterprise perspective on risks and compliance*. " O'Reilly Media, Inc.", 2009.
- [157] D. Sanderson, *Programming google app engine: build and run scalable web apps on google's infrastructure*. " O'Reilly Media, Inc.", 2009.
- [158] T. Redkar, T. Guidici, and T. Meister, *Windows azure platform*. Springer, 2009.
- [159] M. Ebbers, W. Bosch, H. J. Ebert, H. Hellner, J. Johnston, M. Kroll, W. Mild, W. O'Brien, B. Ogden, I. Salm *et al.*, *Introduction to the New Mainframe: IBM Z/VSE Basics*. IBM Redbooks, 2016.
- [160] G. M. Connolly, S. Alexander, P. Bari, A. Botura, M. Ecker, K. Park, C. Perzl, H. Xia, and M. Yamazaki, "Ibm enterprise workload manager v2. 1," *IBM Redbook*, pp. 3–11, 2006.
- [161] B. Schroeder and M. Harchol-Balter, "Web servers under overload: How scheduling can help," *ACM Transactions on Internet Technology (TOIT)*, vol. 6, no. 1, pp. 20–52, 2006.
- [162] V. Cardellini, E. Casalicchio, M. Colajanni, and S. Tucci, "Mechanisms for quality of service in web clusters," *Computer Networks*, vol. 37, no. 6, pp. 761–771, 2001.
- [163] T. Vercauteren, P. Aggarwal, X. Wang, and T.-H. Li, "Hierarchical forecasting of web server workload using sequential monte carlo training," *IEEE transactions on signal processing*, vol. 55, no. 4, pp. 1286–1297, 2007.

- [164] T. M. Mitchell, *The discipline of machine learning*. Carnegie Mellon University, School of Computer Science, Machine Learning Department, 2006, vol. 9.
- [165] E. Alpaydin, *Introduction to machine learning*. MIT press, 2014.
- [166] N. Collier, N. T. Son, and N. M. Nguyen, “Omg u got flu? analysis of shared health messages for bio-surveillance,” *Journal of biomedical semantics*, vol. 2, no. 5, p. S9, 2011.
- [167] M. Di and E. M. Joo, “A survey of machine learning in wireless sensor networks from networking and application perspectives,” in *Information, Communications & Signal Processing, 2007 6th International Conference on*. Citeseer, 2007, pp. 1–5.
- [168] B. Ma, Y. Su, and F. Jurie, “Covariance descriptor based on bio-inspired features for person re-identification and face verification,” *Image and Vision Computing*, vol. 32, no. 6-7, pp. 379–390, 2014.
- [169] G. Tur and R. De Mori, *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons, 2011.
- [170] K. M. Alhawiti, “Advances in artificial intelligence using speech recognition,” *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 9, no. 6, pp. 1351–1354, 2015.
- [171] G. Tur and R. De Mori, *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons, 2011.
- [172] P. K. Chan and S. J. Stolfo, “Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection.” in *KDD*, vol. 98, 1998, pp. 164–168.
- [173] E. Aleskerov, B. Freisleben, and B. Rao, “Cardwatch: A neural network based database mining system for credit card fraud detection,” in *Computational Intelligence for Financial Engineering (CIFER), 1997., Proceedings of the IEEE/IAFE 1997*. IEEE, 1997, pp. 220–226.
- [174] S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick, “Credit card fraud detection using bayesian and neural networks,” in *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*, 2002, pp. 261–270.
- [175] I. Kononenko, “Machine learning for medical diagnosis: history, state of the art and perspective,” *Artificial Intelligence in medicine*, vol. 23, no. 1, pp. 89–109, 2001.
- [176] K. R. Foster, R. Koprowski, and J. D. Skufca, “Machine learning, medical diagnosis, and biomedical engineering research-commentary,” *Biomedical engineering online*, vol. 13, no. 1, p. 94, 2014.
- [177] I. Kononenko, I. Bratko, and M. Kukar, “Application of machine learning to medical diagnosis,” *Machine Learning and Data Mining: Methods and Applications*, vol. 389, p. 408, 1997.

- [178] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.
- [179] N. M. Oliver, B. Rosario, and A. P. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 831–843, 2000.
- [180] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, vol. 32, pp. 323–332, 2012.
- [181] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.
- [182] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.
- [183] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
- [184] G. E. Box and G. C. Tiao, *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011, vol. 40.
- [185] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied mathematics and computation*, vol. 214, no. 1, pp. 108–132, 2009.
- [186] M. Dorigo, M. Birattari, C. Blum, M. Clerc, T. Stützle, and A. Winfield, *Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22-24, 2008, Proceedings*. Springer, 2008, vol. 5217.
- [187] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*. Springer, 2011, pp. 760–766.
- [188] J. G. Carbonell, "Learning by analogy: Formulating and generalizing plans from past experience," in *Machine learning*. Springer, 1983, pp. 137–161.
- [189] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
- [190] N. R. Draper and H. Smith, *Applied regression analysis*. John Wiley & Sons, 2014, vol. 326.
- [191] D. F. Specht, "Probabilistic neural networks," *Neural networks*, vol. 3, no. 1, pp. 109–118, 1990.
- [192] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.
- [193] S. Weisberg, *Applied linear regression*. John Wiley & Sons, 2005, vol. 528.



- [194] L. H. Tsoukalas and R. E. Uhrig, *Fuzzy and neural approaches in engineering*. John Wiley & Sons, Inc., 1996.
- [195] D. F. Specht, “Probabilistic neural networks,” *Neural networks*, vol. 3, no. 1, pp. 109–118, 1990.
- [196] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.
- [197] H. Daraei, M. Irandoust, J. B. Ghasemi, and A. R. Kurdian, “Qspr probing of na+ complexation with 15-crown-5 ethers derivatives using artificial neural network and multiple linear regression,” *Journal of Inclusion Phenomena and Macrocyclic Chemistry*, vol. 72, no. 3-4, pp. 423–435, 2012.
- [198] H. Sompolinsky, “The theory of neural networks: The hebb rule and beyond,” in *Heidelberg colloquium on glassy dynamics*. Springer, 1987, pp. 485–527.
- [199] L. H. Tsoukalas and R. E. Uhrig, *Fuzzy and neural approaches in engineering*. John Wiley & Sons, Inc., 1996.
- [200] H. Drucker, D. Wu, and V. N. Vapnik, “Support vector machines for spam categorization,” *IEEE Transactions on Neural networks*, vol. 10, no. 5, pp. 1048–1054, 1999.
- [201] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
- [202] D. Bertsekas, A. Nedić, and A. Ozdaglar, “Convex analysis and optimization (belmont, ma: Athena scientific),” 2003.
- [203] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.
- [204] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [205] M. Frank and P. Wolfe, “An algorithm for quadratic programming,” *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.
- [206] K.-S. Shin, T. S. Lee, and H.-j. Kim, “An application of support vector machines in bankruptcy prediction model,” *Expert Systems with Applications*, vol. 28, no. 1, pp. 127–135, 2005.
- [207] A. Berlinet and C. Thomas-Agnan, *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.
- [208] B. Schölkopf, A. Smola, and K. R. Müller, “Nonlinear component analysis as a kernel eigenvalue problem,” *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

- [209] D. R. Hardoon and J. Shawe-Taylor, “Decomposing the tensor kernel support vector machine for neuroscience data with structured labels,” *Machine learning*, vol. 79, no. 1-2, pp. 29–46, 2010.
- [210] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.
- [211] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “Tensorflow: a system for large-scale machine learning.” in *OSDI*, vol. 16, 2016, pp. 265–283.
- [212] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen *et al.*, “Mllib: Machine learning in apache spark,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1235–1241, 2016.
- [213] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.
- [214] R. Collobert, S. Bengio, and J. Mariéthoz, “Torch: a modular machine learning software library,” *Idiap, Tech. Rep.*, 2002.
- [215] K. Lee and M. Son, “Deepspotcloud: leveraging cross-region gpu spot instances for deep learning,” in *Cloud Computing (CLOUD), 2017 IEEE 10th International Conference on*. IEEE, 2017, pp. 98–105.
- [216] F. Seide and A. Agarwal, “Cntk: Microsoft’s open-source deep-learning toolkit,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 2135–2135.
- [217] B. C. Ooi, K. L. Tan, S. Wang, W. Wang, Q. Cai, G. Chen, J. Gao, Z. Luo, A. K. Tung, Y. Wang *et al.*, “Singa: A distributed deep learning platform,” in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 685–688.
- [218] R. High, “The era of cognitive systems: An inside look at ibm watson and how it works,” *IBM Corporation, Redbooks*, 2012.
- [219] M. Hofmann and R. Klinkenberg, *RapidMiner: Data mining use cases and business analytics applications*. CRC Press, 2013.
- [220] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The weka data mining software: an update,” *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.
- [221] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.
- [222] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

- [223] B. C. Ooi, K. L. Tan, S. Wang, W. Wang, Q. Cai, G. Chen, J. Gao, Z. Luo, A. K. Tung, Y. Wang *et al.*, “Singa: A distributed deep learning platform,” in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 685–688.
- [224] T. Winograd, “Understanding natural language,” *Cognitive psychology*, vol. 3, no. 1, pp. 1–191, 1972.
- [225] A. Gheith, R. Rajamony, P. Bohrer, K. Agarwal, M. Kistler, B. W. Eagle, C. Hambridge, J. B. Carter, and T. Kaplinger, “Ibm bluemix mobile cloud services,” *IBM Journal of Research and Development*, vol. 60, no. 2-3, pp. 7–1, 2016.
- [226] R. Bouman and J. van Dongen, “Pentaho solutions,” *Business Intelligence and Data Warehousing with Pentaho and MYSQL*, 2009.
- [227] C. Miltin Mboh, C. Montzka, R. Baatz, and H. Vereecken, “A novel partial grid search approach for handling complex multi-dimensional parameter estimation and state improvement at the catchment scale.” in *EGU General Assembly Conference Abstracts*, vol. 16, 2014.
- [228] J. Baker and A. Collinson, “Electrical energy storage at the turn of the millennium,” *Power Engineering Journal*, vol. 13, no. 3, pp. 107–112, 1999.
- [229] A. Zeh and R. Witzmann, “Operational strategies for battery storage systems in low-voltage distribution grids to limit the feed-in power of roof-mounted solar power systems,” *Energy Procedia*, vol. 46, pp. 114–123, 2014.
- [230] S. Ruzic, A. Vuckovic, and N. Nikolic, “Weather sensitive method for short term load forecasting in electric power utility of serbia,” *IEEE Transactions on Power Systems*, vol. 18, no. 4, pp. 1581–1586, 2003.
- [231] N. Kamel and Z. Baharudin, “Short term load forecast using burg autoregressive technique,” in *Intelligent and Advanced Systems, 2007. ICIAS 2007. International Conference on*. IEEE, 2007, pp. 912–916.
- [232] C. E. Borges, Y. K. Peña, I. Fernández, J. Prieto, and O. Bretos, “Assessing tolerance-based robust short-term load forecasting in buildings,” *Energies*, vol. 6, no. 4, pp. 2110–2129, 2013.
- [233] Y. Sakamoto, M. Ishiguro, and G. Kitagawa, “Akaike information criterion statistics,” *Dordrecht, The Netherlands: D. Reidel*, vol. 81, 1986.
- [234] I. Eidhammer, H. Barsnes, G. E. Eide, and L. Martens, *Computational and statistical methods for protein quantification by mass spectrometry*. Wiley Online Library, 2013.

*This page has been intentionally left blank.*

# Appendix A

## Server Load Prediction with split ratio

### A.1 Server Load Prediction on basic dataset for split ratio

The first test scenario presented was conducted using the basic Wikipedia dataset with a split ratio of 70/30. The analysis of results(subchapters) was split between quantity of data(single month data or whole year data) and learning model.

#### A.1.1 Test runs and analysis for multiple months dataset

The months used for the multiple months scenario were from January 2016 to August 2016. At the time of the study these datasets were the most recent available through the Wikimedia foundation. As for the test setup, the months were tested individually not sharing any information between them. The configurations used were 1, 8 and 24 for granularity, 1, 4, 8, 12, 24, 48 and 168 for time window/history and normalization and non-normalization of data.

#### Linear Regression

The linear regression test results were synthesised in Table A.1. The results presented show the average, minimum and maximum values divided by month. The prediction scope used lied always within the next hour.

From the results presented, it is possible to verify that across all months the addition of temporal attributes/time windows enabled the models to have an  $\bar{R}^2$  above the 91% mark, i.e., from an average perspective the approach applied enabled the models to explain the variation in the prediction values by 91% and above through the predictors/input variables. The created models had also a low level of noise classification/overfit, varying average results between 0,2% and 5,3% of overfitting. From a percentual error perspective, the approach created models that reduced the error significantly as can be confirmed when comparing the average MAPE values with the baseline MAPE values. From real life perspective this represents that through the application of this methodology , for instance

Table A.1: Result synthesis for all months using linear regression, basic dataset and split ratio 70/30

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
January	0.988	0.002	2.17 ± 1.57	0.86	25.36	84.69 ± 54.46	4.76	2185.40	101.74 ± 57.19	6.22	2496.68
February	0.975	0.017	2.53 ± 1.47	1.00	21.88	104.18 ± 42.03	3.54	1439.34	118.04 ± 44.41	4.53	1439.34
March	0.985	0.003	3.29 ± 1.53	0.87	21.07	121.20 ± 43.26	3.70	804.37	130.34 ± 44.34	4.77	830.18
April	0.943	0.039	2.09 ± 1.54	0.67	16.31	75.72 ± 50.23	3.60	1212.01	91.67 ± 52.81	4.69	1322.29
May	0.975	0.007	2.49 ± 1.66	1.15	17.89	79.71 ± 40.88	4.71	644.08	91.04 ± 42.51	6.46	706.24
June	0.910	0.053	2.10 ± 1.52	1.02	15.78	85.07 ± 52.33	3.22	1225.32	103.61 ± 55.68	4.16	1230.79
July	0.929	0.014	7.18 ± 3.79	1.86	31.90	275.47 ± 139.42	5.63	2351.61	311.71 ± 144.29	7.68	2828.92
August	0.965	0.015	2.08 ± 1.15	0.61	13.71	25.73 ± 7.54	4.81	446.92	27.89 ± 7.84	5.60	446.92

in the month of January, an initial prediction error of 2185.40 gigabytes was reduced, on average to an error of about  $84.69 \pm 54.46$  gigabytes. From an absolute perspective it was even possible to create a model that predicted the next hour of Wikipedia traffic with an error of only 4.76 gigabytes. Additionally, when analysing the results in terms of data quality it was possible to verify through the MAE and RMSE values, that although outliers were present in the data, their quantity was moderate, the month which had the biggest percentage of outliers was July and which might be one reason for its higher average MAPE value.

From the average analysis it was also possible to observe how granularity in conjunction with time window influenced the MAPE value. Figure A.1 presents these variations along with its effect on  $\bar{R}^2$ .

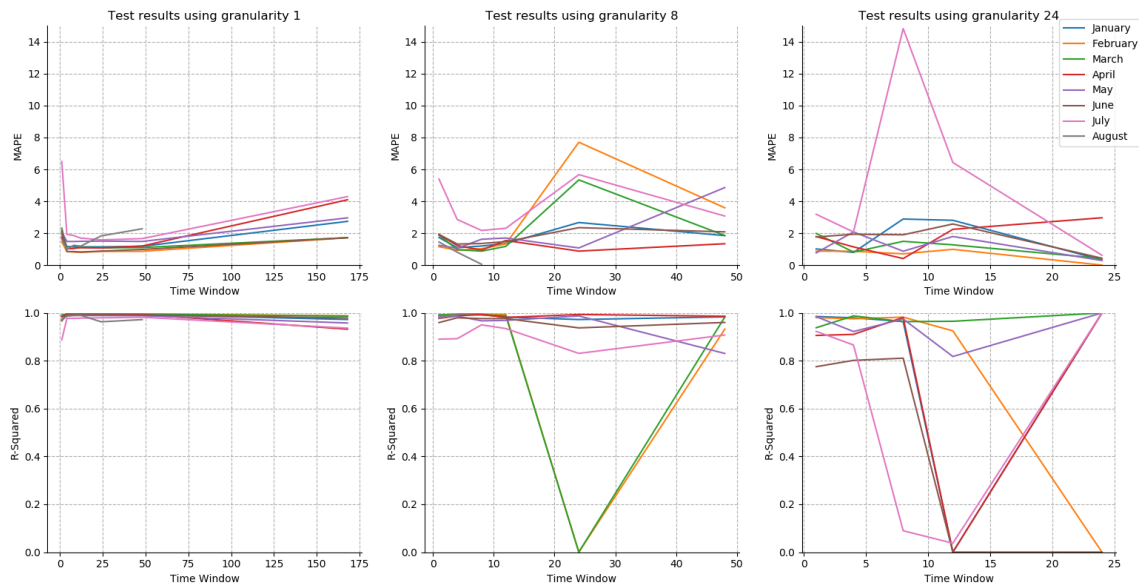


Figure A.1: Influence of granularity and time window in MAPE and R-Squared for linear regression

As can be seen, the use of a time window influenced the values of MAPE and  $\bar{R}^2$ . For the month of August, the results presented are only referent to about a third of the month, this is due to availability of data.

When using a granularity of just 1 the increase of time window reduced the value of MAPE, but only up to the 48/50-hour mark, after that, the increase in time window

lead to a worse performance of the model. The variations of  $R^2$  accompanied the MAPE variations, i.e., from a general perspective when the MAPE increased the  $R^2$  value reduced in a proportional way.

When using a granularity of 8 the increase in time window had a beneficial effect in a much shorter lifespan. The model performed better between the 4 and 12 mark in all months, with the exception of April and May, which have presented better performance for a time window of 24. In terms of  $R^2$ , the variations were of higher magnitude when compared to the use of granularity of 1. Which again translates in high MAPE values for low  $R^2$  values.

For a granularity of 24, the models performance increased from its predecessor of 8, with the exception of the month of July which had an increase in its overall error. The reasons for this higher error value might be the increased quantity of outliers in the data, as can be verified in the average RMSE value, which was the highest amongst all months. In terms of MAPE values, the models performance seems to behave best with a time window of 24. For lower a time window, the MAPE value fluctuates on average between the 1 and 2 percent. In terms of  $R^2$  the increase of granularity seems to have a worsening effect since the fluctuation is the highest among all the three granularity configurations. The impact of  $R^2$  however in terms of model performance appears to be lower, as the variations of  $R^2$  have a reduced impact in the MAPE value.

It is important to mention that the different scales in the  $x$  axis referent to the different granularities where due to effect of data compression when increasing the granularity. When increasing the granularity, a record would contain more information regarding different hours, while using the same available data. This as a result decreases the range of time window, since a time window of 24 with granularity 1 has the same total data as a time window of 3 with a granularity of 8.

## Artificial neural networks

Similar to the linear regression, the ANN test results were synthesised in a table, Table A.2.

Table A.2: Result synthesis for all months using ANN, basic dataset and split ratio 70/30

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
January	0.985	0.004	2.43 ± 1.63	1.36	15.36	79.25 ± 57.06	4.74	758.85	98.08 ± 60.15	6.29	1012.59
February	0.927	0.054	2.34 ± 1.49	1.22	18.92	99.18 ± 44.52	4.35	1553.17	115.53 ± 47.59	5.73	1553.17
March	0.983	0.003	2.92 ± 1.52	0.84	17.61	114.87 ± 44.98	3.68	1315.27	125.18 ± 46.30	4.74	1392.58
April	0.914	0.064	2.61 ± 1.81	1.30	16.03	85.48 ± 51.37	4.76	1186.54	100.74 ± 53.76	6.04	1343.25
May	0.972	0.006	3.13 ± 2.17	1.61	17.14	109.61 ± 65.21	4.93	1132.15	129.17 ± 68.21	6.60	1436.76
June	0.883	0.076	2.50 ± 1.81	1.11	15.88	87.70 ± 56.38	3.52	1166.64	105.75 ± 59.27	4.63	1417.78
July	0.795	0.061	6.65 ± 3.72	1.61	19.02	208.63 ± 93.53	5.99	1413.00	231.95 ± 96.60	7.70	1476.06
August	0.965	0.016	2.08 ± 1.20	0.74	14.35	22.50 ± 8.93	3.43	183.12	25.16 ± 9.32	4.21	205.35

In most months the addition of temporal attributes/time windows increased  $\bar{R}^2$ , only June and July presented an average  $\bar{R}^2$  below the 90% mark. The created models had also an overall low level of noise classification/overfit, which was nevertheless higher when compared to linear regression. The average results varied between 0,4% and 7.6%. From a percentual error perspective, the approach created models that reduced the error

significantly as can be confirmed when comparing the average MAPE values with the baseline MAPE values. The average MAPE values were similar to the linear regression ones.

When analysing the results in terms of data quality it was possible to verify through the MAE and RMSE values, that outliers had a bigger impact factor in terms of model performance when compared directly the linear regression models.

In terms of granularity and time window influence, Figure A.2 presents the variations.

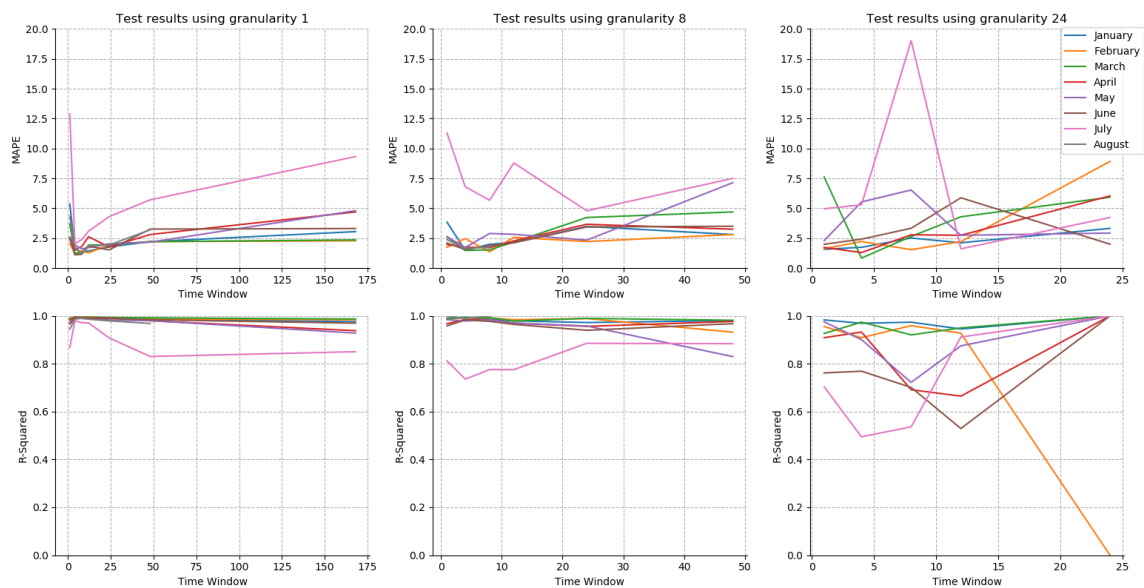


Figure A.2: Influence of granularity and time window in MAPE and R-Squared for ANN

As illustrated by the figure, the use of a time window influenced the values of MAPE and  $R^2$ . When using a granularity of just 1 the increase of time window reduced the value of MAPE visibly up to the 24-hour mark, after that an increase in time window lead to a worse performance of the model, although not in the same magnitude as in the linear regression. The month of July presented an exception to this pattern, being again one the most probable reason, the impact of outliers in the models. The variations of  $R^2$  again conveyed the MAPE variations, with the impact of outliers again present in the month of July.

For a granularity of 8, the models performed better with a time window of 8, presenting the months of February and July also good results with a time window of 24. July was again an outlier in terms of results. With regard to  $R^2$ , the variations were minimal with different time windows, with the exception of July.

Finally, for granularity 24, the results were more chaotic, with all months having higher MAPE variations. The best time windows were however 8 and 12. The impact of  $R^2$  in the final MAPE outcome was higher, when compare to the same situation in linear regression.



## Support vector machine

Akin to the previous analysis, a condensed version of the test results was presented in a table, as seen in Table A.3.

Table A.3: Result synthesis for all months using SVM, basic dataset and split ratio 70/30

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
January	0.989	0.002	2.06 ± 1.40	1.41	13.14	79.23 ± 46.62	4.88	1117.57	92.56 ± 48.57	6.30	1208.67
February	0.979	0.014	1.86 ± 1.04	0.94	19.19	85.45 ± 32.08	3.35	1575.38	96.76 ± 34.16	4.32	1575.38
March	0.985	0.003	2.95 ± 1.26	1.09	16.89	114.12 ± 34.08	3.51	1254.98	120.25 ± 34.72	4.54	1285.40
April	0.949	0.034	2.09 ± 1.47	0.96	15.66	75.08 ± 48.25	3.51	1160.03	89.88 ± 50.56	4.49	1304.98
May	0.976	0.006	2.38 ± 1.62	0.74	17.64	72.41 ± 39.03	4.76	662.08	83.13 ± 40.58	6.57	709.55
June	0.903	0.060	1.93 ± 1.53	1.02	15.21	70.51 ± 54.51	3.20	1114.36	89.93 ± 57.97	4.15	1343.91
July	0.907	0.027	6.47 ± 3.17	1.89	15.34	218.76 ± 82.47	5.71	960.23	238.88 ± 85.28	7.69	990.47
August	0.966	0.016	1.94 ± 0.93	1.17	13.88	23.70 ± 5.01	4.75	376.48	24.72 ± 5.12	5.59	376.48

When compared to ANN, it was possible through the use of SVM to obtain an average  $\bar{R}^2$  above the 90%. When it comes to noise classification/overfit, the SVM models presented similar results to linear regression, varying on average between 0.2% and 6%. With regard to percentual errors, the created models on average reduce their error 10%, when compared to its initial baseline. The data quality impact in the model was similar to linear regression and lower to ANN, as can be verified from the MAE and RMSE difference. In terms of granularity and time window influence, Figure A.3 shows these variations.

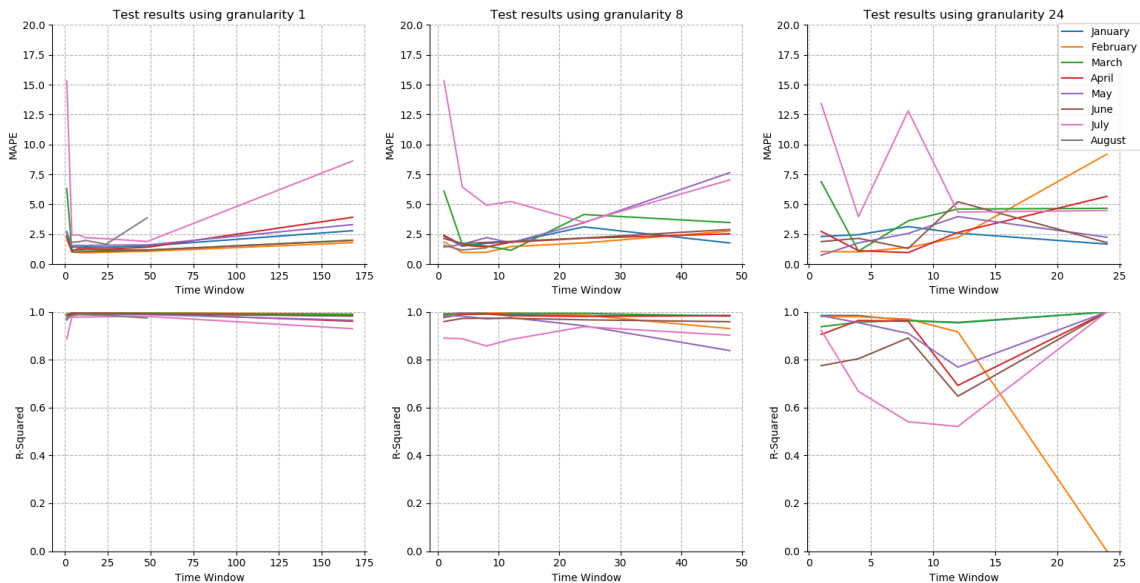


Figure A.3: Influence of granularity and time window in MAPE and R-Squared for SVM

As can be seen, the MAPE and  $R^2$  variations with SVM were of less magnitude in contrast to the ANN variations. The month of July confirms this trend, as its error variations were the most outstanding amongst all months.

When using a granularity of 1, the increase of time window reduced the value of MAPE, with a comparable range as in linear regression, i.e., up to the 48/50 mark. An increase in

time window, after this period, lead to an inferior performance of the model. The variations of  $R^2$  again, went along the MAPE variations, although with a much smoother variation than the other learning models.

With a granularity of 8, a time window between 4 and 12 was the best choice, with exemption of July which achieved a better performance with a time window of 24. The  $R^2$  variations were minor with different time windows, being July again an omission to this pattern.

Lastly, for granularity 24, the results were more disordered, all months had higher MAPE variations. The best results were proven to be found with a lower time window, being on average the best results with 4 and 8 time window. The exception to this norm was this time also found in another month with the exception of July, March. July and March presented very similar symptoms with lower time windows, they started with higher MAPE value, decreased, reaching a lower peak at around 4 time window and then start to increase again. March however, contrary to July, does not reach the high level of MAPE value on its next peak, which almost hits baseline level.

The variations of  $R^2$  accompanied the MAPE until around a time window of 12, after that its impact was significantly lower. July was non-restricted by this pattern.

## **Model comparison**

With the individual learning method analysis concluded, a comparison is made between all learning models in order to find the best possible model. An initial pre-selection was made containing all the best models per learning method, from this pre-selection a single final model was chosen per month. Table A.4, presents these pre-selections.

For the majority of months, the decision of an ideal month was an more or less straight forward choice. Between the learning methods of each month the difference between the ideal model and the others were usually significant enough to justify the choice. However, for the months of March, June, July and August this choice was not so obvious.

In the case of March, an initially choice might fall for the model created through ANN, since the MAPE of this model is lower than the one using linear regression. However, basin the decision of the best model just on MAPE might be deceiving. Although ANN presents a lower MAPE, the values of  $\bar{R}^2$  and model overfitting, present us with overall better understanding of the model. The model using ANN has an lower error when making prediction on the testing data used, however due to the lower value of  $\bar{R}^2$  and higher of overfitting it is rational to assume, that the model doesn't take as much advantage of the input data as linear regression and also presents a higher level of noise classification, which on the long run might create more impactful errors with the use of new data.

As for the month of June, the crucial factor of decision was the capacity to handle outliers. In this month, the MAPE,  $\bar{R}^2$  and model overfitting couldn't be used as differential factor since between the models created through linear regression and SVM all values are equal, as such, the metrics MAE and RMSE were used as differential. As shown in Table A.4, the MAE for both learning models is the same, through the difference between MAE and RMSE it is possible to extract information about the impact of outliers, the lower this

Table A.4: Best models with configuration for Linear regression, ANN and SVM with basic dataset and split ratio

Month	Learning Method	Granularity	Time Window	$\bar{R}^2$	Model Overfitting	MAPE	MAE	RMSE
January	<b>LR</b>	<b>24</b>	<b>4</b>	<b>0.979</b>	<b>0.002</b>	<b>0.86</b>	<b>4.76</b>	<b>6.22</b>
	ANN	1	8	0.993	0.0004	1.36	4.74	6.29
	SVM	1	8	0.992	0.0004	1.41	4.91	4.11
February	LR	1	12	0.997	0.003	1	3.54	4.53
	ANN	1	4	0.995	0.0001	1.22	4.34	5.72
	<b>SVM</b>	<b>1</b>	<b>12</b>	<b>0.996</b>	<b>0.0003</b>	<b>0.94</b>	<b>3.35</b>	<b>2.72</b>
March	<b>LR</b>	<b>24</b>	<b>4</b>	<b>0.988</b>	<b>0.003</b>	<b>0.87</b>	<b>3.7</b>	<b>4.77</b>
	ANN	24	4	0.916	0.0104	0.83	62.06	53.88
	SVM	24	4	0.929	0.0088	1.09	79.71	33.3
April	<b>LR</b>	<b>24</b>	<b>8</b>	<b>0.981</b>	<b>0.009</b>	<b>0.67</b>	<b>3.6</b>	<b>4.69</b>
	ANN	24	4	0.896	0.0129	1.3	51.86	112.17
	SVM	24	8	0.892	0.0134	0.96	74.09	61.11
May	LR	24	1	0.982	0.002	1.15	4.71	6.46
	ANN	1	8	0.988	0.0007	1.61	4.93	4.39
	<b>SVM</b>	<b>24</b>	<b>1</b>	<b>0.982</b>	<b>0.0022</b>	<b>0.74</b>	<b>54.22</b>	<b>51.75</b>
June	LR	1	4	0.993	0.002	1.02	3.22	4.16
	ANN	1	4	0.991	0.0002	1.11	3.51	3.01
	<b>SVM</b>	<b>1</b>	<b>4</b>	<b>0.993</b>	<b>0.0002</b>	<b>1.02</b>	<b>3.22</b>	<b>2.63</b>
July	<b>LR</b>	<b>1</b>	<b>48</b>	<b>0.967</b>	<b>0.015</b>	<b>1.86</b>	<b>5.63</b>	<b>7.68</b>
	ANN	24	12	0.911	0.0423	1.61	119.6	56.98
	SVM	1	48	0.965	0.015	1.89	5.14	7.69
August	LR	8	8	0.995	0.011	0.61	4.81	5.6
	<b>ANN</b>	<b>24</b>	<b>1</b>	<b>0.999</b>	<b>0.0002</b>	<b>0.61</b>	<b>51.1</b>	<b>51.1</b>
	SVM	24	1	0.966	0.011	1.17	26.77	9.77

value, the less the impact of outliers is present in the model. From this analysis the model chosen for June was the one using SVM.

With regards to the month of July, the approach taken for deciding a model was similar to the one used in June. Although on initial analysis the model using linear regression might seem the obvious choice, the proximity of the values achieved through the use of SVM is significant enough to perform a deeper analysis. From the standpoint of  $\bar{R}^2$  and overfitting, the difference between these two learning models is almost inexistent. In terms of MAPE, the difference of just 0.03% is not sufficient to justify the choice of one in detriment of the other. As such, the impact of outliers must again be used as differentiator, from this point of view linear regression presents a better performance than its SVM counterpart.

To conclude, for the month of August the decision for the model created by ANN was justified, by all other metrics with the exception of MAPE, which was the only value that achieved equal results between linear regression and ANN. All other metrics were in favour of the model created through ANN.

Having selected the best models for each month, an overall analysis of these models was performed in terms of the impact of time window and error distribution across the

learning methods. This analysis granted a better insight from the overall performance of the models in the case study. By this very nature, the graphics presented in Figure A.4 were created regarding this time window distribution.

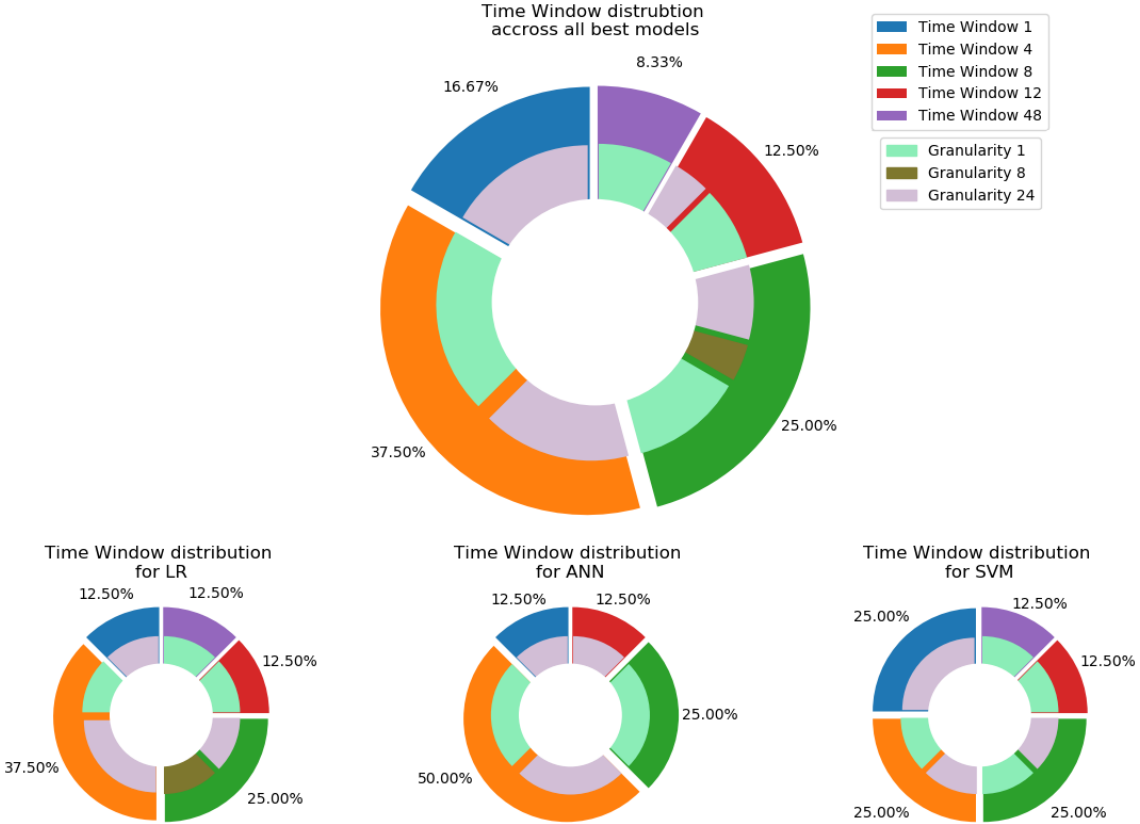


Figure A.4: Time window distribution across all best models for single month with split ratio

As shown in the figure, in terms of time window distribution and disregarding the different learning models, a time window of 4 was in 37.50% of the cases the best choice, followed up by a time window of 8 with 25%. Curiously, some months, 16.67%, had their optimal results for certain learning models when no time window was applied, these cases however never used a granularity of 1. In general, the best results were achieved when using a time window lower than 12, an increase in time window after this mark had lower chances of success as 12.5% of the learning models achieved optimal results with time window of 12 and only 8.33% achieved it for a time window of 48.

In terms of results per learning model, a time window of 4 and 8 was the best choice, comprising about 62.5%, 75% and 50% of the results for linear regression, ANN and SVM respectively. It was also possible to confirm that SVM had more ease in handling higher time windows than other learning methods, with ANN specially having these difficulties, as no optimal model used a time window of 48.

When analysing the MAPE distribution regardless of the learning method and/or if it was an optimal model, it is possible to get a big picture of how the use of granularity and time window affects the performance of created models. Figure A.5 shows the pie charts regarding this information.

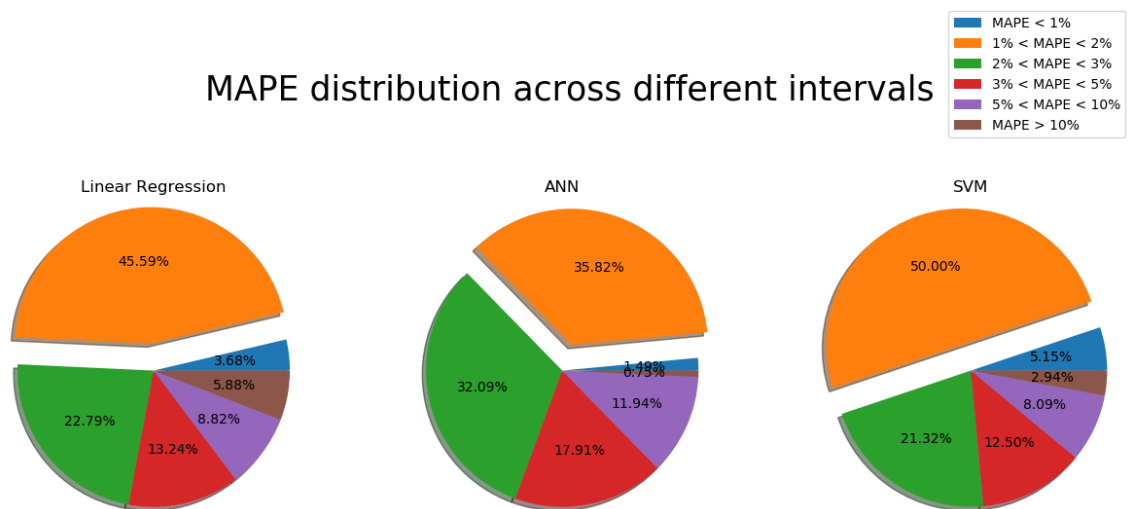


Figure A.5: Time window distribution across all best models for single month with split ratio

In terms of distribution, linear regression and SVM presented very similar patterns. Most MAPE values lied in the range of 1% and 2%. For linear regression and SVM this was their dominant range of values, having 45,59% and 50% respectively. As for ANN this distribution was also the most predominant of the error spectrum. In terms of errors above the 10 percent mark, linear regression was clearly the worst model with 5,88%, which was almost double of SVM which had 2,94% and almost six times more than ANN which had only 0,75%. Having ANN the lowest percentage of errors above 10%, when it comes to errors below 1%, ANN unfortunately also presented the lowest percentage with only 1,49%, significantly lower than linear regression which achieved 3,68% and SVM which had 5,15%.

Summing up, linear regression and SVM presented very similar outputs, having the last one a slight advantage in terms of performance. As for ANN the models created were mostly average, i.e., the percentage of models with errors above the 10% were very low, however the models with MAPE below 1% were also very low. A total of 67.91% of the models created with ANN had a MAPE between 1 and 3 percent.

## A.1.2 Test runs and analysis for single year dataset

For the single year test, the datasets from the single month scenario were aggregated and used as a singular year dataset. The prediction scope for this dataset similar to the previous test scenario remained for the next hour. From the point of view of granularity and time window, the configurations were kept identical. However, differently from the single month test scenario the results in this scenario were not split by learning method. As such the generic and absolute analysis was summarized in a single table, Table A.5.

As shown in the table, the baseline MAPE improved significantly when compared to the single month scenario. SVM specifically stands out with a MAPE of only 6.91%, an impressive increase in performance when taking into consideration that the lowest baseline MAPE for the single month was of 13.14%, almost double. This result might indicate a

Table A.5: Average and best results for single year dataset using split ratio

Result synthesis for single year dataset

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
LR	0.9222	0.0363	2.50 ±1.96	1.20	11.94	101.79 ±75.90	4.12	454.80	127.47 ±80.20	6.22	524.59
ANN	0.8678	0.0297	4.60 ±3.51	1.24	13.37	233.01 ±166.21	4.24	1101.86	287.56 ±175.12	6.26	1228.26
SVM	0.8951	0.0459	2.20 ±1.79	1.22	6.91	93.72 ±73.27	4.18	380.97	119.34 ±77.68	6.26	465.50

Best models with respective configuration

Learning Method	Granularity	Time Window	$\bar{R}^2$	Model Overfitting	MAPE	MAE	RMSE
LR	1	48	0.9932	0.0004	1.20	4.14	6.22
ANN	1	8	0.9936	0.0001	1.24	4.24	6.26
SVM	1	24	0.9933	0.0002	1.22	4.18	6.28

bigger adaptability from SVM to the year dataset scenario when compared to the other learning methods.

In terms of average perspective of overfitting the results got worse, with average values of 0.9222, 0.8678, 0.8951 for linear regression, ANN, SVM respectively. This outcome though, was somewhat expected since the increase in data in conjunction with the time window appliance would eventually amplify subtle noise and/or redundancy found in the single months. This amplifying pattern was also found, although not in the same magnitude, in the impact of outliers, which from the average perspective had also increased. Nonetheless, for months with huge outlier impact such as June the aggregation of months had a benevolent, mitigating effect.

In terms of absolute results, the best models fit in between the results of the single month scenario. These results presented an increase in performance for months with a higher percentage of outliers, such as, June, but had slightly higher MAPE when compared to the “regular” months. The  $\bar{R}^2$ , which although with not much of a difference, was higher in the year scenario, which contradicted to some extent the average results landscape. Figure A.6 shows the graphical representation of the baseline classification and the best model classification.

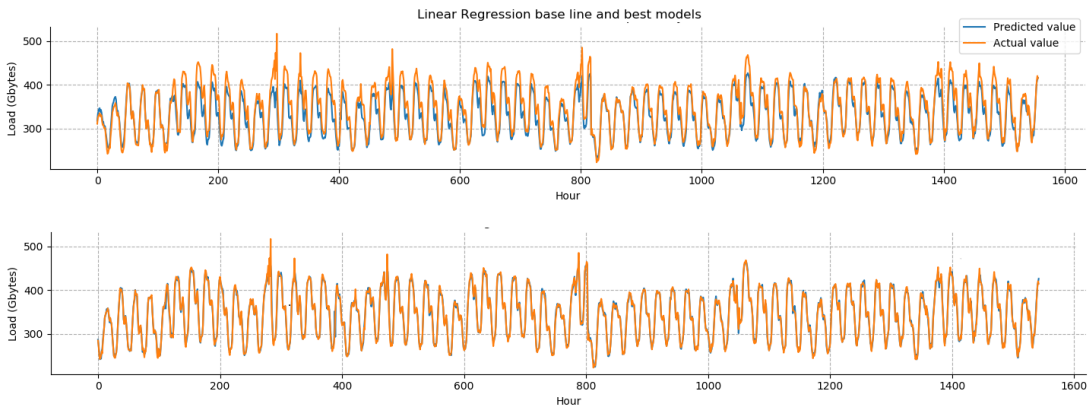


Figure A.6: Linear regression learning method with actual and predicted values for baseline and best models.

As it is possible to verify, the baseline model, predicted with a deficit, which in the case of the Wikipedia could represent the not allocation of resources in need. However, with the best model, the curves overlap in most situations. When analysing the overall performance of the learning methods with regards to the influence of granularity and time window a better big picture is possible to be taken, Figure A.7 presents the graphical representation of this influence.

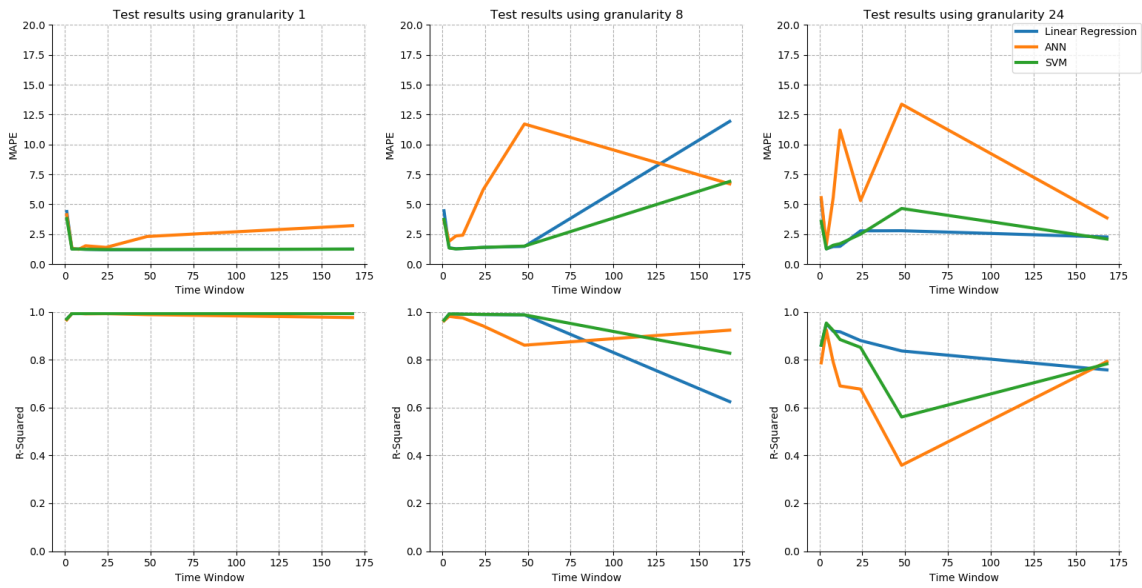


Figure A.7: Time window distribution across all best models for single month with split ratio

One of the things that stands out in this test scenario is the relation between  $R^2$  and MAPE, i.e., contrary to the single month scenario, the increase in granularity does not weaken the bond between these two metrics. Throughout the whole test scenario, a lower MAPE comes with a higher  $R^2$  and every subtle variation in one of the metrics effects the other.

When observing the results with granularity one, it is possible to confirm that the lowest MAPE values are present in this range, which translates also in the highest  $R^2$  values. The models seem to perform better between a time window of 4 and 24, a decrease in range when compared to the single month test scenario. In terms of learning methods, linear regression and SVM had very similar output, which is proven by the fact that the line plot regarding linear regression is totally overlaid by the line plot of SVM. ANN was the worst performing model, which is a trend maintained and amplified with the increase in granularity.

Opposing to the previous test scenario, an increase in granularity did not reduced the range of time window in which the model performed better. For linear regression and SVM this range window lied between 4 and 48/50 time window. ANN did not follow this pattern however, its MAPE peaks at around 48 time window, decreasing gradually after that, never to reach however the same values as the other learning methods in their prime time window.



Lastly, with a granularity of 24 the models created presented the highest MAPE values. ANN was again the worst performing model, presenting however again a decrease in MAPE after its peak, which occurred again at around time window 24. MAPE and  $R^2$  had the biggest oscillations in this granularity.

Regarding the MAPE values distribution a synthesis was created and illustrated through pie charts, as shown in Figure A.8.

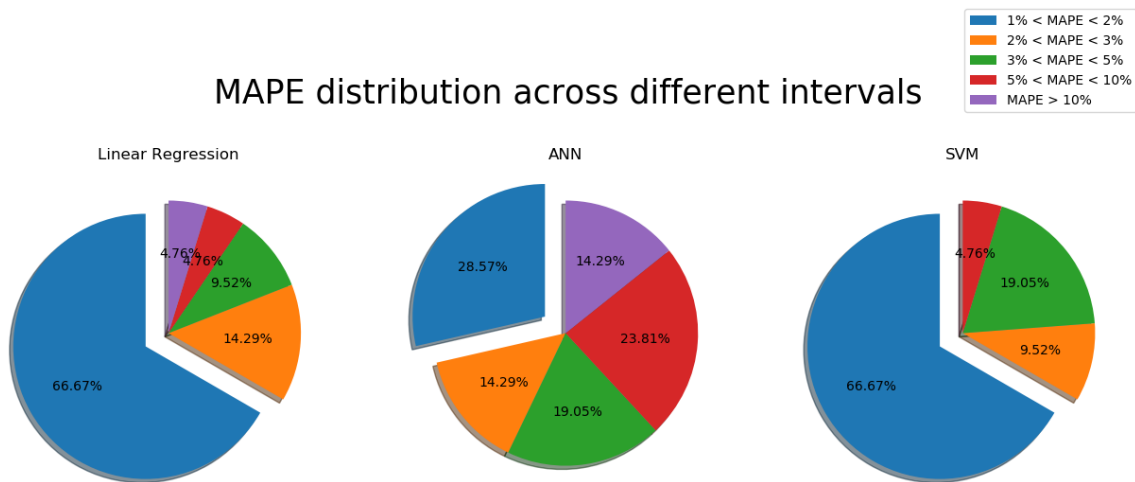


Figure A.8: Time window distribution across all best models for single year with split ratio

Confirming the initial hypothesis form granularity and time window influence, it is possible to verify that linear regression and SVM, present very similar results. Both learning models have 66.67% of MAPE values between 1 and 2 percent. ANN had the worst results, with comparably only 28.57% between 1 and 2 percent. The main difference between linear regression and SVM in terms of results, lied in the percentage of MAPE above the 10 percent, which SVM had none and linear regression had 4.76. The biggest percentage of MAPE values above 10 was achieved with ANN. The distribution of MAPE in ANN was overall more evenly distributed.

## A.2 Server Load Prediction on enhanced dataset for split ratio

Equivalent to the basic dataset scenario, different configurations for time window and granularity were used. The testing and analysis procedures were overall similar to the basic one. The difference between these two test scenarios were the enhanced dataset used and an initial feature analysis of the dataset, this procedure was not compulsory for the basic scenario since the dataset only contained 2 attributes in its initial form.

### A.2.1 Feature analysis

On an initial approach, different attributes along with different formats for the same attributes were tested. For instance, a date attribute was tested as string, numeric, time



stamp, and split into 4 different attributes: hour, day, month and year. All these variations for date had a detrimental effect on the models performance, focus was on MAPE values. Flag attributes such as indicators of week days or weekends were also tested, as well, as period of the day: morning, afternoon, evening, midday and midnight, all these attributes had a detrimental effect in the end result. These initial test results led to the exclusion of said attributes. Tests runs showed that statistical attributes were the most efficient. A reason for this might be the creation process of the dataset. Since the dataset is an aggregate of English language pages, the non-statistical attributes may not perform so well in non-region locked data, i.e., non-statistical attributes such as time of the day focus on human activity patterns, since the records were aggregates of data from different parts of the world, human patterns would most likely not be transferred to the record, thus rendering these attributes inefficient to the learning methods.

From the testing phase emerged a final dataset, the enhanced dataset depicted in Figure 5.1. Nevertheless, in order to validate the attributes chosen from a statistical point of view, a feature analysis was conducted. A correlation matrix between attributes was created, as seen in Figure A.9.

	sum_requests1	avg_requests1	avg_size1	Weekend_stdev_requests1	Weekend_stdev_size1	Population_stdev_requests1	Population_stdev_size1	language_occur1
sum_requests1	1							
avg_requests1	0.76	1						
avg_size1	0.77	0.83	1					
Weekend_stdev_requests1	0.56	0.86	0.79	1				
Weekend_stdev_size1	0.58	0.85	0.82	0.78	1			
Population_stdev_requests1	0.56	0.86	0.79	0.81	0.78	1		
Population_stdev_size1	0.58	0.85	0.82	0.78	0.81	0.78	1	
language_occur1	0.78	0.18	0.24	0.03	0.07	0.03	0.07	1

Figure A.9: Correlation matrix for Wikipedia’s enhanced dataset

All the features presented in the figure are positively correlated among themselves, that is, when a value of a variable increases or decrease, i.e., changes its direction, all correlated features change their values in the same direction. The impact of this on each attribute is usually influenced by the correlation coefficient between these features. When analysing features in terms of correlation, usually a high correlation, e.g., above 0.95, implies that the two attributes have more or less the same impact in the models performance, sharing lot a of the same information. In situations like these the exclusion one of the attributes or the fusion of the two attributes into one might be options. Nevertheless, the resulting model should always be tested before and after the changes, as high correlated attributes may share a lot information, however, the information that they do not share might be crucial for the models performance. Analogous a lower correlation coefficient indicates that the variables share very little information between them, which may indicate low redundancy of information for the learning method. Testing before and after is again a necessity, as the low correlation coefficient might just be a symptom from one of the attributes being noise to the learning method and thus having a detrimental effect in the models performance.

In the information presented in Figure A.9 the correlation between attributes varies between 0.03 and 0.86, which indicates that low correlated attributes may be just noise to the learning method, while high correlated ones suggest an eventual redundancy in information. Test were conducted for each attribute in order two measure their overall impact in the models performance, the results concluded that removing any attribute would worsen the models performance.

## A.2.2 Test runs and analysis for multiple months dataset

With the validation of the enhanced dataset, testing and analysis processes are initiated. The testing and evaluation methodologies as well as the dataset configurations are identical to the basic dataset.

### Linear Regression

Table A.6: Result synthesis on enhanced dataset for all months using linear regression with split ratio

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
January	0.9964	0.0035	1.58 ± 1.07	0.10	16.44	59.20 ± 39.62	0.55	459.39	71.60 ± 41.57	0.79	546.26
February	0.9997	0.0002	1.54 ± 0.70	0.06	13.51	78.24 ± 24.85	0.46	1108.48	85.70 ± 25.99	0.59	1108.48
March	0.9955	0.0043	2.40 ± 1.29	0.15	18.47	104.99 ± 53.00	0.62	1184.28	118.77 ± 54.84	0.81	1340.56
April	0.7534	0.1689	7.43 ± 5.30	2.27	29.72	224.56 ± 158.76	7.11	809.68	277.74 ± 167.77	9.60	1058.62
May	0.9938	0.0060	2.13 ± 1.27	0.10	14.94	75.52 ± 36.84	0.47	497.66	85.13 ± 38.14	0.75	523.40
June	0.9987	0.0010	2.09 ± 1.55	0.14	13.09	84.18 ± 56.61	0.43	603.08	102.20 ± 59.50	0.64	653.61
July	0.9406	0.0573	2.86 ± 1.70	0.62	12.97	94.93 ± 51.80	1.79	533.89	109.57 ± 53.94	2.35	611.31
August	0.9989	0.0009	3.61 ± 1.76	0.21	17.81	51.54 ± 15.65	0.59	316.59	54.84 ± 16.03	0.66	331.46

As can be seen in Table A.6, linear regression's performance across all months increased in almost all metrics when compared to the basic dataset. The month of April was the only one in which linear regression performed worse. All months with exception of April and June achieved average  $\bar{R}^2$  above 0.9938. These results indicate that the addition of the statistical attributes helped the models to classify the missing variations in the predictions from the basic dataset scenario. In terms of model overfitting the results remained very similar to the basic dataset, which suggests that the increase in  $\bar{R}^2$  and  $R^2$  were not related to noise classification.

The minimum MAPE and MAE values were the most obvious differences in the models, as all months with exception of April and July achieved a MAPE below 0.21%, which translates to around 0.59 Gigabytes difference in the actual load value. These results are also reflected in the average MAPE values, all months with exception of April and June presented lower average values when compared to the basic dataset scenario.

With regards the time window and granularity influence, most of the months benefited from the enhanced dataset, repeating the above presented pattern, Figure A.10 illustrates the variations.

In terms of most effective time window range, results remained identical for granularity 1, this is, up to the 48/50-hour mark. However, when it comes to MAPE distribution a different outcome is visible. Results through the enhanced dataset were able to achieve more extreme values, i.e., the minimum and maximum values of linear regression decreased and increased respectively. As for  $R^2$  its influence in MAPE was clear, the oscillations of values increased when compared to the basic scenario.

For granularity 8, the time window performance range remained almost identical to the basic scenario, with exception of time window 8. Time window 8 achieved the highest MAPE values in all months. Similar to the basic scenario increase in granularity increased

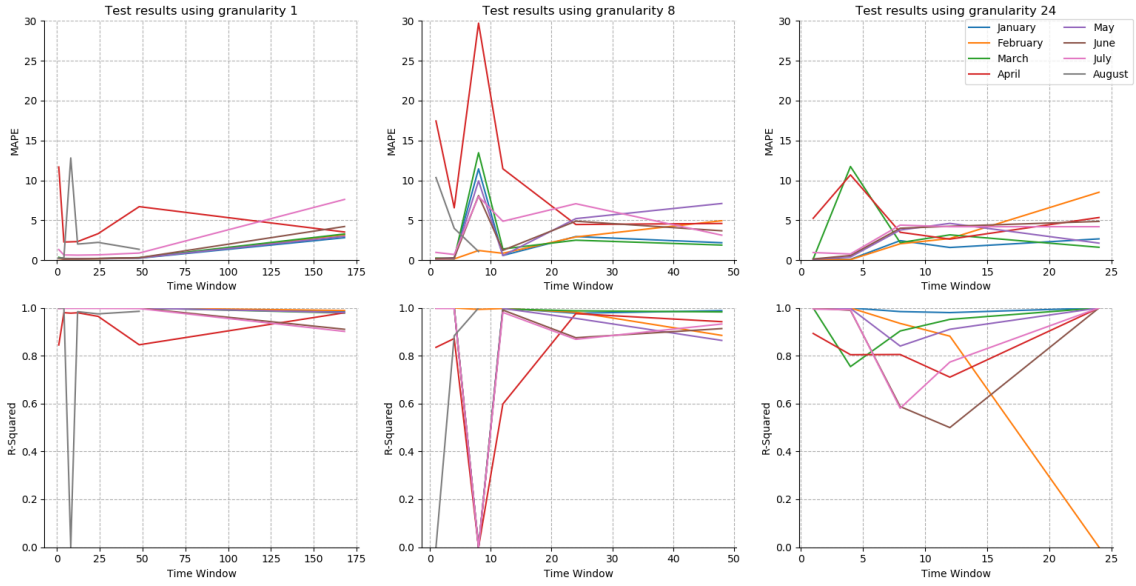


Figure A.10: Influence of granularity and time window in MAPE and R-Squared for linear regression on enhanced dataset with split ratio

the oscillations of values for MAPE and  $R^2$ , the relation/influence between these 2 metrics remained however intact.

Lastly, for granularity 24 the minimum and maximum values decreased and increased as well. Correlation between MAPE and  $R^2$  remained intact. Similar to the basic scenario, with increased granularity the magnitude of MAPE fluctuation through  $R^2$  influence decreased.

When using enhanced dataset, months with outliers such as July seem to be handled better by linear regression, however, months with correction procedures(wrong values were substituted by average values) such as April seem to have more difficulties.

## Artificial neural networks

Table A.7: Result synthesis on enhanced dataset for all months using ANN with split ratio

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
January	0.9630	0.0365	$1.52 \pm 1.00$	0.18	10.08	$55.08 \pm 37.76$	0.62	460.92	$67.62 \pm 39.90$	0.81	635.05
February	0.9933	0.0059	$2.82 \pm 1.63$	0.22	19.42	$91.39 \pm 37.60$	0.79	859.41	$103.42 \pm 39.57$	0.95	859.41
March	0.8931	0.1058	$1.49 \pm 0.95$	0.15	9.28	$52.74 \pm 29.64$	0.42	584.86	$61.59 \pm 30.96$	0.58	690.87
April	0.9280	0.0378	$5.45 \pm 3.81$	0.54	16.60	$172.83 \pm 107.76$	1.71	820.11	$205.51 \pm 112.84$	2.31	878.22
May	0.9520	0.0475	$1.98 \pm 1.24$	0.16	12.70	$69.13 \pm 38.29$	0.50	454.62	$80.03 \pm 39.87$	0.64	547.15
June	0.9906	0.0083	$2.19 \pm 1.76$	0.20	10.87	$91.57 \pm 68.81$	0.62	706.72	$115.18 \pm 72.83$	0.92	909.27
July	0.9937	0.0034	$2.72 \pm 1.54$	0.30	11.70	$95.72 \pm 46.92$	0.89	636.61	$108.22 \pm 48.64$	1.22	732.56
August	0.9975	0.0020	$1.29 \pm 0.85$	0.24	7.71	$17.44 \pm 5.38$	0.65	268.94	$19.21 \pm 5.68$	0.88	268.94

From Table A.7 it is possible to verify that average and standard deviation MAPE values decreased when compared to the basic scenario. As for  $\bar{R}^2$  the enhanced dataset had an overall increasing effect. The performance of ANN in terms of overfit had also increased, with the exception in the months of March and May. The best achieving models

from a MAPE perspective also significantly increased their performance, attaining in all months values below the 0.54% mark. Finally, when analysing ANN performance in handling outliers, the results remained similar to the basic dataset. The use of an enhanced dataset, similar to linear regression, had an overall beneficial effect when compared to its basic counterpart.

When it comes of granularity and time window influence, most months performed similarly. Contrary to the basic dataset July, which contained most outliers, did not perform worst. Identically to linear regression April was the worst performing month. Figure A.11 all the MAPE and  $\bar{R}^2$  variations.

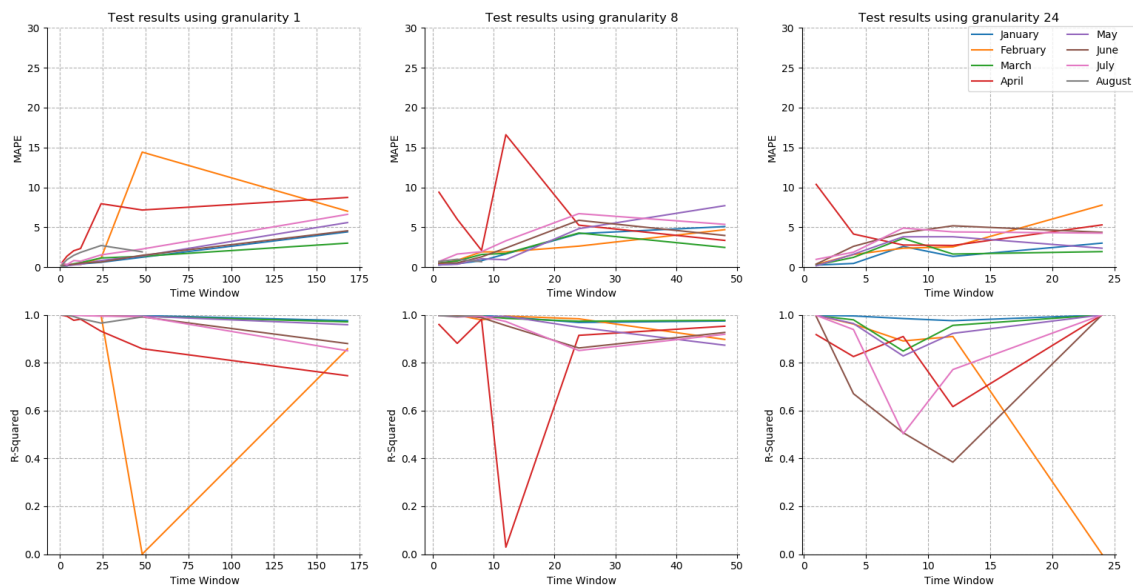


Figure A.11: Influence of granularity and time window in MAPE and R-Squared for ANN on enhanced dataset with split ratio

With granularity 1 a lower time window presented a better performance in all months with exception of February and April. Through most time window variations,  $R^2$  values oscillated in the same manner as MAPE. When increasing granularity to 8, a lower time window achieved better results, peaking the MAPE values at time window 24, after that MAPE decreased gradually, however, never reaching same levels as with a lower time window. With granularity 8 the only month with out of the norm behaviour was April. To conclude, with a granularity of 24, results were more consistent when compared to the basic. A lower time window was again a synonym of lower MAPE. With granularity 24, the month of April presented identical variations to other months. An increase in granularity seem to diminish the malevolent effects on the models performance, present in lower granularities.

### Support vector machine

In like manner as linear regression and ANN, SVM presented an increase in performance when compared to the basic dataset version, Table A.8 present the synthesis of the results.

Table A.8: Result synthesis on enhanced dataset for all months using SVM with split ratio

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
January	0.99262	0.00713	1.09 ± 0.74	0.20	8.03	49.01 ± 30.33	0.75	634.39	58.42 ± 31.85	1.12	796.07
February	0.99899	0.00072	1.65 ± 0.75	0.17	13.55	84.40 ± 28.88	0.58	1112.17	93.08 ± 30.20	0.76	1112.17
March	0.99082	0.00857	1.07 ± 0.66	0.28	8.15	40.59 ± 22.98	0.90	594.99	47.74 ± 24.12	1.15	668.19
April	0.89752	0.09264	4.17 ± 3.07	0.56	16.03	168.26 ± 110.76	4.30	1231.41	201.96 ± 115.85	5.39	1467.32
May	0.99041	0.00928	1.68 ± 0.96	0.18	11.92	65.34 ± 32.40	0.57	519.79	74.28 ± 33.70	0.92	549.18
June	0.99698	0.00220	1.91 ± 1.48	0.21	11.83	89.24 ± 59.93	0.64	908.24	108.57 ± 63.11	0.98	991.92
July	0.92728	0.07068	2.42 ± 1.35	0.55	12.41	84.96 ± 41.49	2.03	538.49	95.75 ± 42.92	2.65	616.91
August	0.99852	0.00118	1.34 ± 0.79	0.13	9.37	24.61 ± 7.63	0.37	219.71	26.60 ± 7.90	0.56	226.14

April was again the only month that did not benefited from the enhanced dataset, but just from an average perspective, in terms of best possible model the enhanced dataset was also a success.

With exception of April, all months achieved average  $R^2$  above 0.92728, a slight increase to the basic dataset. In terms of average MAPE and respective standard deviation, SVM performed better in all months, excluding April. The minimum achievable metrics MAPE, MAE and RMSE were the only metrics were all months, including April, presented far better results than its basic counterpart. Proving that although from an average point April might perform worst, when it comes to best possible model it is still possible to create a very efficient model. Lastly, with regards to metric analysis, the enhanced dataset also diminished the impact of outliers, as can be verified by the increase in performance of July, the month that contains most outliers.

Concerning the time window and granularity influence, Figure A.12 presents identical to previous analysis the variations.

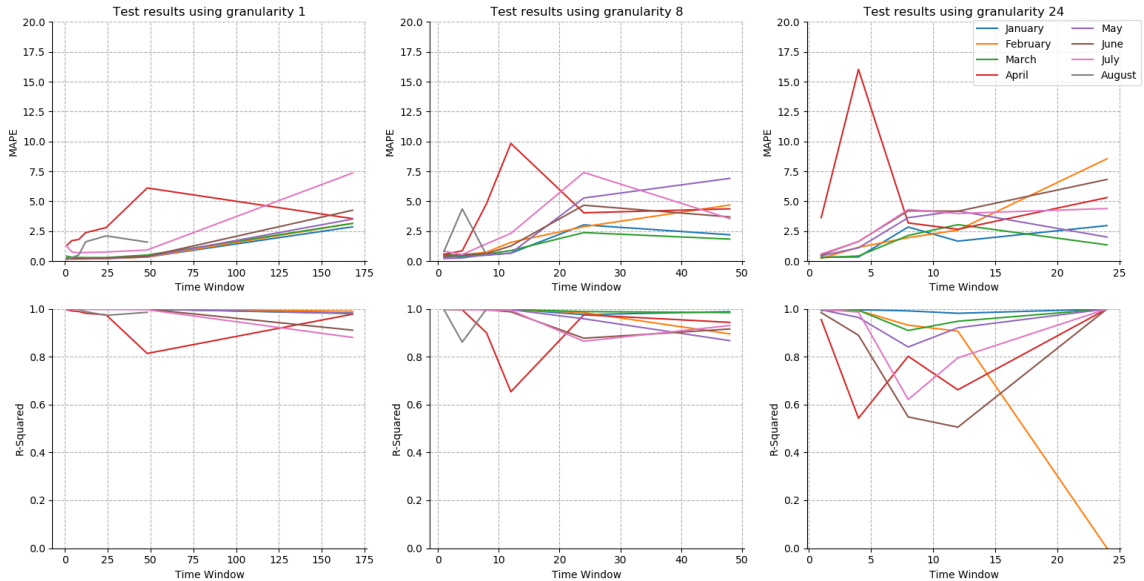


Figure A.12: Influence of granularity and time window in MAPE and R-Squared for SVM on enhanced dataset with split ratio

Figure A.12 confirms the outlier behaviour of April, indicted by the metrics analysis. With a granularity 1, SVM presents the same pattern found in the basic scenario. The

main difference relies in the range of values, i.e., the pattern is identical in both scenarios, however, the values presented by the enhanced scenario are lower.  $R^2$  behaviour remained also identical.

For a granularity of 8, a lower granularity achieves a better model performance, peaking the error at 24 time window, after 24 the error decreases again, however never reaching the same level as found in time window 8 and lower.  $R^2$  were inversely proportional to MAPE.

To conclude for granularity 24, similar to what happened in linear regression and ANN, the chaotic results achieved with the basic dataset were smoothen out. Contrary to ANN, April kept its outliers results even with a higher granularity. In terms of lower MAPE, a time window of 4 presents in most cases the best results.

## Model comparison

The best achieving models per learning method were selected and presented in Table A.9.

Table A.9: Best models with configuration for Linear regression, ANN and SVM on enhanced dataset with split ratio

Month	Learning Method	Granularity	Time Window	$\bar{R}^2$	Model Overfitting	MAPE	MAE	RMSE
January	<b>LR</b>	<b>24</b>	<b>4</b>	<b>0.99918</b>	<b>0.000537</b>	<b>0.10</b>	<b>8.42</b>	<b>11.02</b>
	ANN	1	4	0.99988	0.000018	0.18	0.62	0.82
	SVM	1	4	0.99979	0.000034	0.20	0.75	1.13
February	<b>LR</b>	<b>24</b>	<b>4</b>	<b>0.99994</b>	<b>0.000633</b>	<b>0.06</b>	<b>4.81</b>	<b>5.96</b>
	ANN	1	4	0.99993	0.000012	0.22	0.79	0.95
	SVM	1	4	0.99991	0.000003	0.17	0.58	0.76
March	LR	24	4	0.99980	0.000320	0.15	11.62	13.81
	<b>ANN</b>	<b>1</b>	<b>4</b>	<b>0.99997</b>	<b>0.000001</b>	<b>0.15</b>	<b>0.42</b>	<b>0.58</b>
	SVM	24	4	0.99643	0.002333	0.42	31.53	33.60
April	LR	1	4	0.97637	0.003882	2.27	7.11	9.60
	<b>ANN</b>	<b>1</b>	<b>4</b>	<b>0.99895</b>	<b>0.000039</b>	<b>0.54</b>	<b>1.71</b>	<b>2.31</b>
	SVM	8	4	0.99915	0.000263	0.56	14.89	19.91
May	<b>LR</b>	<b>24</b>	<b>4</b>	<b>0.99972</b>	<b>0.000454</b>	<b>0.10</b>	<b>6.99</b>	<b>9.94</b>
	ANN	24	4	0.99760	0.001572	0.16	11.97	13.07
	SVM	1	4	0.99975	0.000039	0.18	0.57	0.92
June	<b>LR</b>	<b>1</b>	<b>4</b>	<b>0.99983</b>	<b>0.000028</b>	<b>0.14</b>	<b>0.43</b>	<b>0.64</b>
	ANN	1	4	0.99974	0.000010	0.20	0.62	0.92
	SVM	1	4	0.99964	0.000060	0.21	0.64	0.98
July	LR	1	12	0.99667	0.001637	0.62	1.79	2.35
	<b>ANN</b>	<b>1</b>	<b>4</b>	<b>0.99944</b>	<b>0.000090</b>	<b>0.30</b>	<b>0.89</b>	<b>1.22</b>
	SVM	8	4	0.99400	0.003921	0.55	12.97	16.93
August	LR	1	4	0.99992	0.000020	0.21	0.59	0.66
	ANN	1	4	0.99950	0.000134	0.24	0.65	0.89
	<b>SVM</b>	<b>1</b>	<b>4</b>	<b>0.99980</b>	<b>0.000053</b>	<b>0.13</b>	<b>0.37</b>	<b>0.56</b>

All months with exception of August achieved their best models with either linear regression or ANN. All months were able to achieve models with  $\bar{R}^2$  above 0.998 and a MAPE below 0.54. All months achieved their best model with a time window of 4 and granularity 1 or 24.

For January the best model was achieved with linear regression, were it achieved the best results in terms of MAPE. However, in terms of  $\bar{R}^2$ , overfit and outlier handling linear regression was the worst performing learning method. The decision to nevertheless use linear regression relied on the fact that the detail needed to measure these differences was so high that in the end their effect on the end model was almost residual. In the month of February, the best performing learning method was also linear regression. The reasons for choosing linear regression were identical to the month of January. For the month of March linear regression and ANN achieved identical results in terms of MAPE and  $\bar{R}^2$ , the differentiating factor was the models overfit, were ANN performed better, thus leading to the selection of ANN. In April the selection of an ideal model was relatively easy, as ANN outperformed the other learning methods in every metric. For the months of May, June and July the best performing learning methods were linear regression for May and June and ANN for July. The reason for selecting these learning methods was identical to April, as these learning methods in their respective months outperformed the other learning methods in every metric. Lastly, for August the best performing method was SVM. SVM completely outperformed ANN in every metric and just performed worse than linear regression in overfit, although just slightly higher.

With regards to the overall performance of the best performing models, the impact of time window and error distribution across the learning methods was studied. The graphics presented in Figure A.13 show the time window distribution for these models.

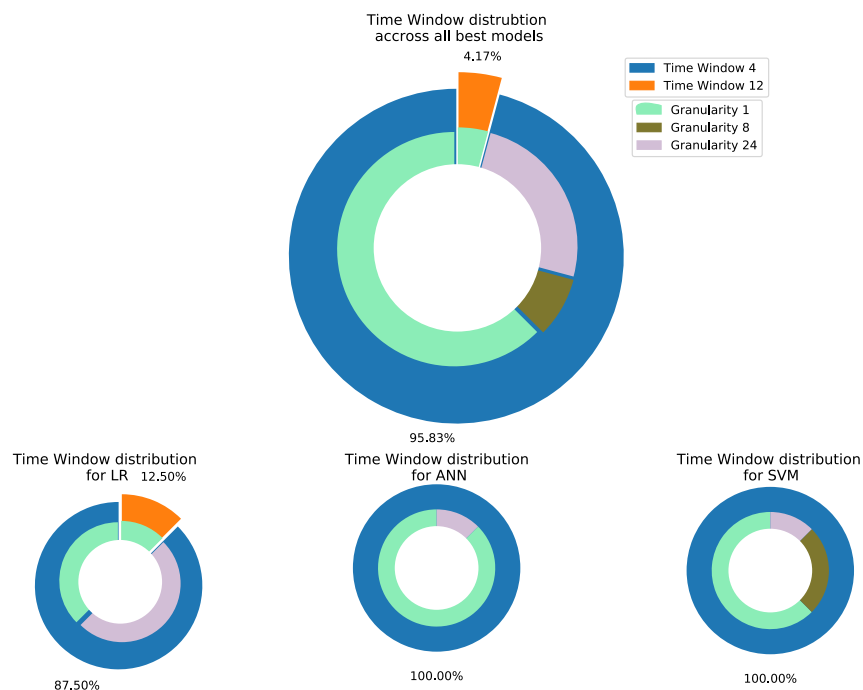


Figure A.13: Time window distribution across all best models for single month with split ratio and enhanced dataset

As evidence from the figure, most models, with 95.83%, were achieved with a time of 4. Only one model used a different time window, linear regression for the month of July with a time window of 12. In terms of granularity, a granularity 1 was the best performing



overall. Contrary to the basic scenario, with the enhanced dataset the optimal time window seems to be consistent in most month, while in terms of granularity the diversity of values remains similarly to what happens in the basic scenario. In both scenarios a granularity of 8 was the least occurring value when it comes to optimal models.

For the MAPE distribution, across all test runs, a study was also carried out, results are shown in Figure A.14.

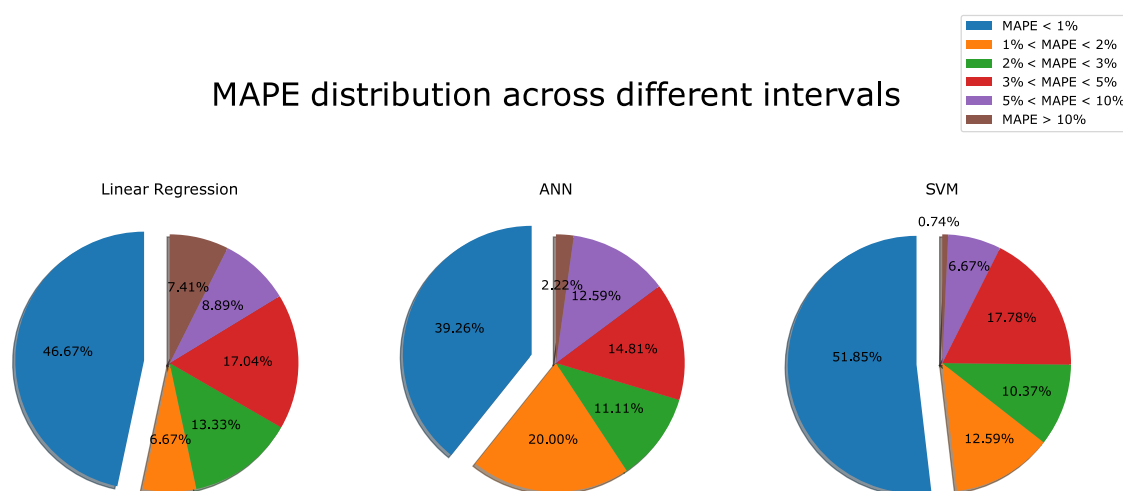


Figure A.14: Time window distribution across all best models for single month with split ratio and enhanced dataset

Different to what Table A.9 indicted, SVM was not the worst performing learning method, from an average perspective. Although only August achieved its best model with SVM, SVM outperformed linear regression and ANN in terms of average MAPE distribution. From all the conducted test runs, 51.85% of the runs were able to create a model with a MAPE below 1%, a significant increase when compared to the 46.67% and 39.26% of linear regression and ANN respectively. Moreover, SVM was also the best performing month in terms of least models with a MAPE above 5% and 10% with 6.67% and 0.74% respectively. Linear regression had 7.41% of the models above the 10% MAPE and ANN had 2.22%. In terms of average performance SVM was the clear winner, followed by linear regression and ANN.

### A.2.3 Test runs and analysis for single year dataset

For the single year test scenario, Table A.10 presents the average and absolute results.

With the enhanced yearly dataset, linear regression and SVM increased their performance in all metrics, ANN was the only method to perform worse than its basic dataset counterpart, from an average point of view. All methods were able to create optimal models with a MAPE below 1%. In terms of time window and granularity, a time window under 12 achieved better results while 24 and 1 were the best choices for granularity. With regards to  $\bar{R}^2$  linear regression was the only month with a slightly worse results, with ANN and SVM achieving  $\bar{R}^2$  above 0.9982. Linear regression was the overall worst model, achieving lower  $\bar{R}^2$  higher overfit and MAPE and presenting the most difficulties with outliers in



Table A.10: Average and best results for single year enhanced dataset using split ratio

Result synthesis for single year dataset

Month	Avg. $\bar{R}^2$	Avg. overfitting	Avg. MAPE	Min. MAPE	Baseline MAPE	Avg. MAE (Gb)	Min. MAE (Gb)	Baseline MAE (Gb)	Avg. RMSE (Gb)	Min. RMSE (Gb)	Baseline RMSE (Gb)
LR	0.9746	0.0153	2.68 $\pm$ 2.18	0.94	10.17	89.07 $\pm$ 63.10	78.11	780.00	109.29 $\pm$ 66.29	105.98	942.47
ANN	0.8692	0.0974	16.00 $\pm$ 16.67	0.55	15.83	419.00 $\pm$ 597.12	1.88	837.47	735.98 $\pm$ 677.15	2.90	1616.17
SVM	0.9933	0.0041	2.60 $\pm$ 2.06	0.51	14.37	94.42 $\pm$ 64.83	42.18	1134.57	114.64 $\pm$ 67.92	47.45	1360.69

Best models with respective configuration

Learning Method	Granularity	Time Window	$\bar{R}^2$	Model Overfitting	MAPE	MAE	RMSE
LR	24	8	0.8454	0.1321	0.94	78.11	105.98
ANN	1	12	0.9986	0.0001	0.55	1.88	2.90
SVM	24	1	0.9982	0.0002	0.51	42.18	47.45

data. ANN and SVM achieved very similar results, with SVM having a slight advantage over ANN, being the best overall learning method for the enhanced yearly dataset.

With regards to granularity and time window influence in the overall models performance, Figure A.15 illustrates the variations.

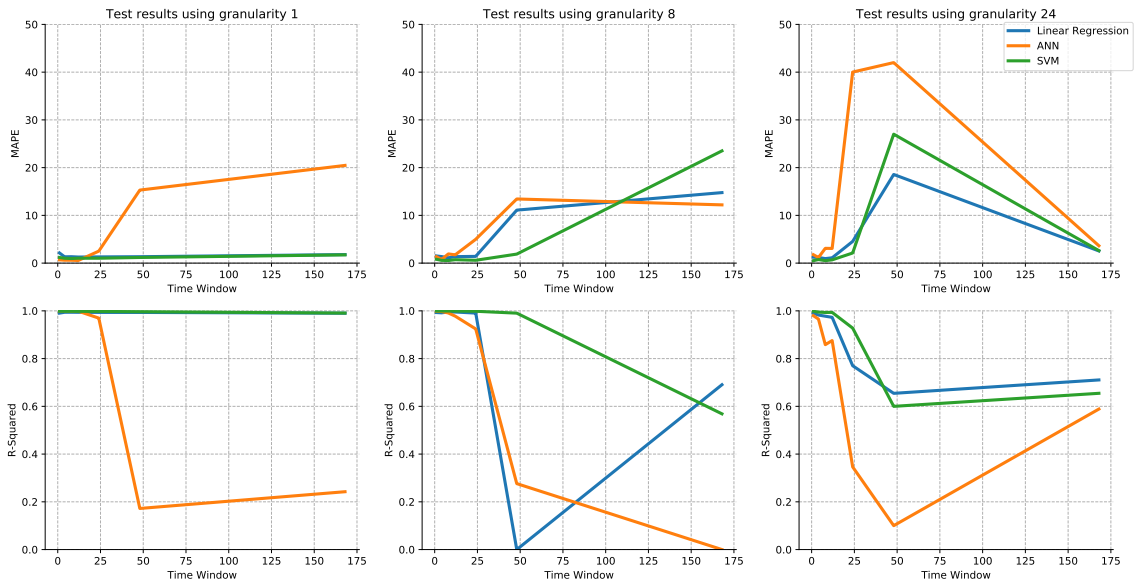


Figure A.15: Time window distribution across all best models for yearly dataset with split ratio and enhanced dataset

Through the analysis of the figure, it is possible to see that from an average point of view ANN was the worst performing learning method. Across all granularities, ANN was always the first month to increase its MAPE drastically with the increase in time window. For granularity 1 and 24, ANN was also the worst performing month in terms of MAPE, only with granularity 8, was ANN outperformed in terms of high MAPE by SVM.

Through the figure in is also possible to identify a clear pattern of lower time windows performing better. With the increase in granularity, the time windows efficiency range

seems to decrease. Throughout most runs a proportionality inverse relationship between MAPE and  $R^2$  was verified.

To conclude, the MAPE distribution was also studied, Figure A.16 depicts the results.

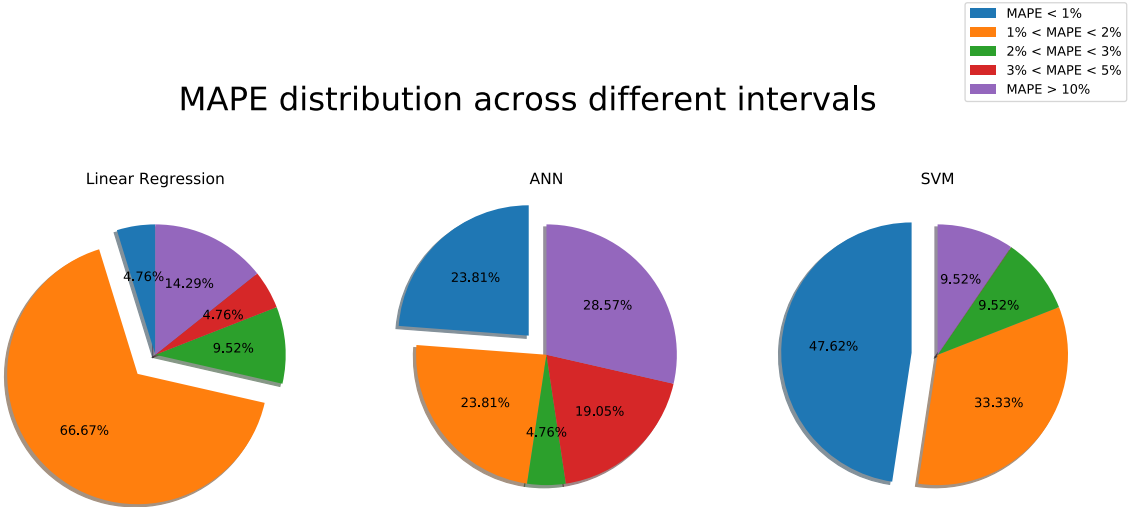


Figure A.16: Time window distribution across all best models for single year with split ratio and enhanced dataset

From an average perspective, SVM outperformed linear regression and ANN. SVM was able to achieve a MAPE below 1% in 47.62% the runs conducted. A great difference when compared to 4.76% and 23.81% for linear regression and ANN respectively. Between linear regression and ANN, linear regression was more even in its results, it had less models with MAPE below 1% but also less models above 10% MAPE than ANN. When compared to the basic dataset, the performance also improved as no learning method with basic dataset was able to achieve models with a MAPE below 1%.