An Architecture for Predictive Maintenance of Railway Points Based on Big Data Analytics

Giulio Salierno[1(✉)], Sabatino Morvillo[2], Letizia Leonardi[1], and Giacomo Cabri[1]

[1] Università di Modena e Reggio Emilia, Modena, Italy
{giulio.salierno,letizia.leonardi,giacomo.cabri}@unimore.it
[2] Alstom Ferroviaria S.p.A, Savigliano, Italy
sabatino.morvillo@alstomgroup.com

Abstract. Massive amounts of data produced by railway systems are a valuable resource to enable Big Data analytics. Despite its richness, several challenges arise when dealing with the deployment of a big data architecture into a railway system. In this paper, we propose a four-layers big data architecture with the goal of establishing a data management policy to manage massive amounts of data produced by railway switch points and perform analytical tasks efficiently. An implementation of the architecture is given along with the realization of a Long Short-Term Memory prediction model for detecting failures on the Italian Railway Line of Milano - Monza - Chiasso.

Keywords: Railway data · Predictive maintenance · Big data architecture

1 Introduction

In recent years, Big Data analytics has gained relevant interest from both industries and academia thanks to its possibilities to open up new shapes of data analysis as well as its essential role in the decision-making processes of enterprises. Different studies [8], highlight the importance of big data, among other sectors, for the railway industries. The insight offered by big data analytics covers different areas of the railway industry, including and not limited to *maintenance*, *safety*, *operation*, and *customer satisfaction*. In fact, according to the growing demand for railway transportation, the analysis of the huge amount of data produced by the railway world has a positive impact not only in the services offered to the customers but also for the railway providers. Knowledge extracted from raw data enables railway operators to optimize the maintenance costs and enforce the safety and reliability of the railway infrastructure by the adoption of

This work was partially supported by the EU H2020 program under Grant No. 734599 - FIRST project.

new analytical tools based on descriptive and predictive analysis. Maintenance of railway lines encompasses different elements placed along the railway track, including but not limited to signals and points. Our interest is mainly on predictive maintenance tasks that aim to build a variety of models with the scope of monitoring the health status of the points composing the line. Typical predicting metrics of Remaining Useful Life (RUL) and Time To Failure (TTF) enable predictive maintenance by estimating healthy status of objects and replacing them before failures occur. Despite unquestionable value of big data for the railway companies, according to [3] big data analytics is not fully adopted by them, yet due to different aspects mainly related with the lack of understanding on how big data can be deployed into railway transportation systems and the lack of efficient collection and analysis of massive amount of data. The goal of our work is to design a big data architecture for enabling analytical tasks typical required by the railway industry as well as enabling an effective data management policy to allows end-users to manage huge amounts of data coming from railway lines efficiently. As already mentioned, we considered predictive maintenance as the main task of our architecture; hence to show the effectiveness of the proposed architecture, we use real data collected from points placed over the Italian railway line (Milano - Monza - Chiasso). The complexity of the considered system poses different challenges for enabling efficient management of the huge amount of data. The first challenge concerns the collection of the data given the heterogeneity of the data sources. Multiple railway points produce distinct log files, which must be collected and processed efficiently. The second challenge is to deal with the data itself. Data collected from the system must be stored as raw data without any modification to preserve the original data in case of necessity (e.g., in case of failures, further analyses require to analyze data at a higher level of granularity). At the same time, collected data must be processed and transformed to be useful for analysis thus, data must be pre-processed and aggregated before fitting models for analytics. Finally, the data analysis performed by the end-users requires analytical models to perform predictive or descriptive analysis; thus, the architecture should enable model creation as well as graphical visualization of results.

The paper is organized as follows. Section 2 describes similar works that discuss the design of Big Data architectures for railways systems. Section 3 describes the kinds of data produced by a railway system. Sections 4 and 5 describe, respectively, the architectural design and its implementation. The Sect. 6 presents a real case scenario in which failure prediction is performed on real data. Section 7 draws some conclusions.

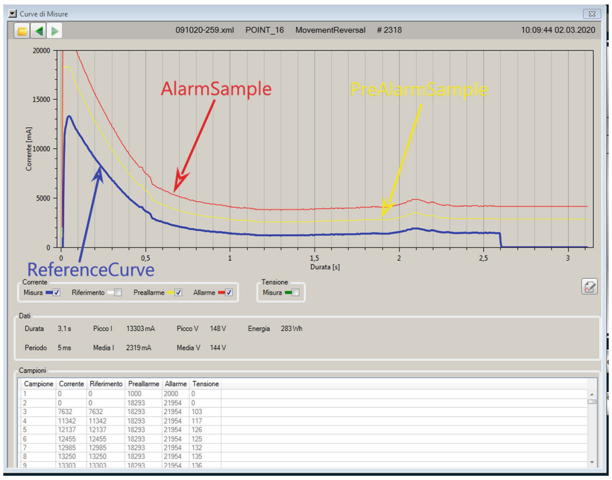
2 Related Work

To the best of our knowledge, few solutions take into consideration challenges arisen when deploying a big data architecture for railway systems. Most works focus on theoretical frameworks where simulations produce results without experimenting with real data. Moreover, researchers mainly focus on Machine

Learning algorithms as well as analytical models, giving less importance to the fundamental tasks related to data management, ingestion processing, and storage. Close to our work in [9], authors propose a cloud-based big data architecture for real-time analysis of data produced by on-board equipment of high-speed trains. However, the proposed architecture presents scalability issues since it is not possible to deploy large-scale computing clusters in high-speed trains; neither is it possible to deploy a fully cloud-based architecture due to bandwidth limitation of trains which make infeasible transferring huge amount of data to the cloud to perform real-time analysis. On the contrary, the scope of our work is to define a scalable Big Data architecture for enabling analytical tasks using railway data. For this reason and due to space limit, we have reported only work related to the railway scenario.

3 Data Produced by a Railway Interlocking System

In our work, we take into consideration the data log files produced by a railway interlocking system. A railway interlocking is a complex safety-critical system that ensures the establishment of routes for trains through a railway yard [1]. The computer-based interlocking system guarantees that no critical-safety condition (i.e., a train circulate in a track occupied by another train) will arise during the train circulation. Among other actions issued by the interlocking, before the route is composed, it checks the state of each point along the line. The interlocking system produces log files that store information about the command issued to the point as well as data about its behavior. Commands are issued by the interlocking through smart boards, which in turn control the physical point on the line and collect data about their status. Once data are collected, they are written into the data storage of the interlocking system as log files. These log files can be both structured and semi-structured data and contain diverse information about the behavior of the points upon the requests sent by the interlocking system. Requests may vary according to the logic that must be executed to set up a route (e.g., a switch point is moved from the normal to the reverse position or vice versa) and this implies that the information contained in log files may vary. A complete railway line is controlled by multiple interlocking systems, which in turn produce different log files according to the points they control. The analytical task which motivates the design of our architecture is the prediction of failures. A failure may occur when a mechanical part of points has a break. This kind of failure propagates negatively on the entire railway traffic; therefore, its prediction is desirable. Moreover, instead of doing maintenance when a failure occurs, it is also useful in particular, to estimate the RUL of point in order to enable predictive maintenance by estimating if a points will fail or not in a certain time-frame. Predicting failures of railway points requires to take into consideration the log files produced by the interlocking whenever a command is issued to a point. These log files are heterogeneous in type and contain different information resumed as:



Dati

Durata 3,1 s Picco I 13303 mA Picco V 148 V Energia 283 Wh

Periodo 5 ms Media I 2319 mA Media V 144 V

Campioni

Campione	Corrente	Riferimento	Preallarme	Allarme	Tensione
1	0	0	1000	2000	0
2	0	0	18293	21954	0
3	7632	7632	18293	21954	103
4	11342	11342	18293	21954	117
5	12137	12137	18293	21954	126
6	12455	12455	18293	21954	125
7	12985	12985	18293	21954	132
8	13250	13250	18293	21954	135
9	13303	13303	18293	21954	136

Fig. 1. Graphical visualization of data sample collected from a switch point

1. Timestamps respectively indicating the time recorded the logging server upon submitting the command, and the timestamp recorded by the smartboard when an action is performed.
2. Information about the smartboard that controls the point (channel number, name of the smartboard, sampling frequency).
3. Information about the operation issued by the interlocking (type of movement, total number of movements, number of current movement in a single day).
4. A set of raw data representing values of Voltage and Power supplied to the point to operate.

Data 3 and 4 are considered to train and evaluate the proposed model to estimate the health status of the points, thus to estimate its RUL (see Sect. 5). As an example, we report a sample collected from a railway switch point (Fig. 1). These samples contain three types of information:

1. **ReferenceCurve**: is a sample curve representing the behavior of the point upon the command issued by the interlocking. This curve is used to derive the following ones;
2. **PreAlarmSample**: is a pre-threshold curve, computed by adding to the ReferenceCurve an intermediate threshold value;
3. **AlarmSample**: is the alarm curve computed as the previous one by adding an alarm threshold value.

4 System Architecture

The Big Data architecture presented in this section covers all the fundamental stages of a big data pipeline [4] of:

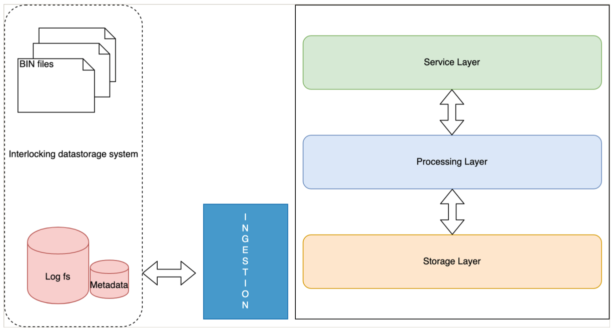


Fig. 2. Architecture for railway big data management

1. Data acquisition by implementing ingestion tasks for collecting data from external sources.
2. Information extraction and retrieval by processing ingested data and storing them in a raw format.
3. Data integration, aggregation, and representation by data table view as well as data aggregation functionalities to produce new data for analytics.
4. Modeling and analysis by providing a set of functionalities to build models to perform predictive analytics.
5. Interpretation of results by graphical visualization of data.

The architecture presented in Fig. 2 includes the following layers:

Storage layer is the layer responsible for implementing the data storage. It contains the storage platform to provide a distributed and fault-tolerant filesystem. In particular, this layer, should store data in their raw form. Therefore datasets for analytic models will be originated from the upper layers.

Processing layer provides all tasks for data manipulation/transformation useful for the analytical layer. In particular, this layer presents a structured view of the data of the Storage layer, allowing the creation of datasets by transforming raw data coming from the Storage layer. The structured view of data is implemented through table views of the raw data. The transformation of original data is performed through aggregation functions provided by this layer.

Service layer contains all components to provide analytics as a service to the end-users. This layer interacts with the processing layer in order to access data stored on the platform, manipulate and transform data to fit analytical models. In addition, it provides: 1) Data visualization functionalities for graphical displaying data 2) Models creation to perform analytical tasks.

Ingestion layer This layer implements all the tasks for the ingestion of data from external sources. It is based on ingestion tools, which enable the definition of dataflows. A dataflow consists of a variable number of processes that transform data by the creation of flow files that are moved from one processor to another through process relations. A relation is a connection between two processors to define data flow policies among data flow processes.

Our architecture is an implementation of the concept of a datalake [2]. To avoid situations in which data stored in the platform become not usable due to multiple challenges related to their complexity, size, variety, and lack of metadata, we adopt a mechanism of URI[1] abstraction to simplify data access, thus establishing a data governance policy. As an example, at the storage layer, the URI of a resource is simply its absolute path. In order, to avoid the definition of multiples URIs for each resource (since they can be used by multiple components at different architectural layer), we define a URI abstraction mechanism to simplify the access to resources since they are stored in a distributed manner (where keeping track of the physical location of a resources could be tricky). Therefore the *RealURI* refers to a resource stored on the distributed filesystem abstracting its physical location. A *RealURI* is bound to a single *VirtualURI*, which is in charge of abstracting the details of paths adopted by a particular implementation of a distributed filesystems. A *PresentationURI* is an optional URI created whenever a component of the Processing layer or Service layer uses a resource stored on the filesystem. As an example, the URI abstractions defined for a single resource are reported in Table 1. Each resource is identified by 1) a smartboard id, 2) a channel number that controls a specific point, 3) a point number which identifies the object on the line. These metadata are extracted from the data described in Sect. 3. through the tasks provided by the Ingestion layer described in the next section. In addition, the *VirtualURI* refers to the resource at a platform level, while the *PresentationURI* represents a HIVE table view of the data created by the processing layer (see Sect. 5). We stress the fact that while resources can be assigned to an unbounded number of *PresentationURI*, depending on the type of components that consume the data, the *VirtualURI* is mandatory and it refers to a single *RealURI*.

Table 1. URI abstractions for storage resources

URI type	URI	View level
RealURI	`hdfs://data_path/smart_board_number/channel/point_number`	Filesystem
VirtualURI	`adc://data_path/smart_board_number/channel/point_number`	Platform
PresentationURI	`adc:hive://data_path/smart_board_number/channel/point_number`	Analytics

5 Architecture Implementation

The architecture has been implemented mainly using components of the Hadoop[2] stack. Hadoop is a framework that allows for the distributed processing of large data sets across clusters of computers using simple commodity-hardware. Hadoop provides different components to implement a complete big data architecture. In particular, for this work, we considered:

[1] An Uniform Resource Identifier (URI) is a sequence of characters that uniquely identify resources on a system.

[2] http://hadoop.apache.org/.

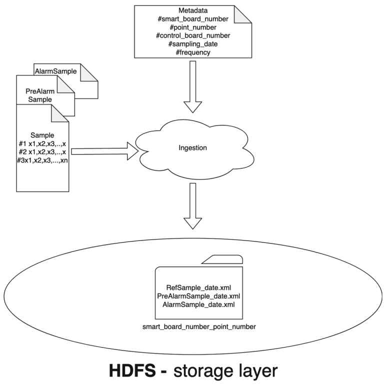


Fig. 3. Ingestion process to store data and related metadata

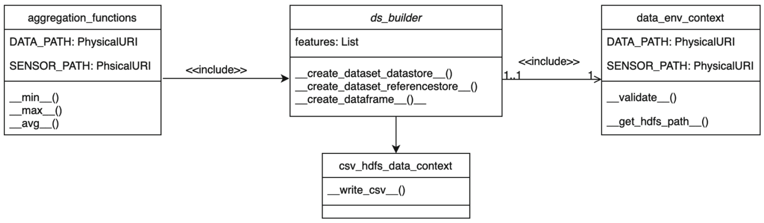
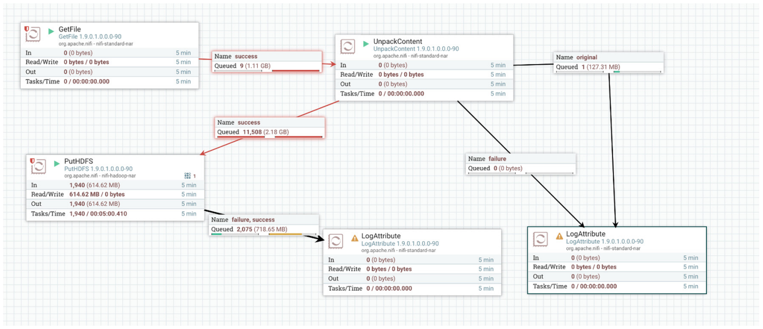


Fig. 4. Class diagram of the dataset builder module

Storage layer, which has been implemented using the Hadoop Filesystem named HDFS. HDFS is a fault-tolerant distributed filesystem that runs on cluster providing fault-tolerance and high availability of the data. HDFS stores raw data as ingested by the Ingestion layer, as represented in Fig. 3. In particular, the Ingestion layer performs extraction of data and metadata and aggregate data into a specific folder stored on HDFS representing data for a particular point. Data representing point behavior (see Sect. 3) are stored in their original format as XML files. Therefore these data must be processed and transformed to create new datasets. This task is performed by the processing layer.

Processing layer Before data can be employed into analysis must be transformed to fulfill the requirements of analytical models. The processing layer implements all the tasks required to build datasets from raw data. This layer has been implemented through the specification of two components. The first component has the scope of processing raw XML files by extracting relevant features and aggregating them into CSV files, thus producing new datasets. Results are written back to the HDFS in the folder of the original data provenance. This mechanism allows enriching the data available, producing aggregation of raw data as well as providing features extraction functionalities to extract/aggregate features to be used by analytical models.



GetFile
GetFile 1.9.0.1.0.0.0-90
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Name success
Queued 9 (1.11 GB)

UnpackContent
UnpackContent 1.9.0.1.0.0.0-90
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Name original
Queued 1 (127.31 MB)

Name success
Queued 11,508 (2.18 GB)

Name failure
Queued 0 (0 bytes)

PutHDFS
PutHDFS 1.9.0.1.0.0.0-90
org.apache.nifi - nifi-hadoop-nar
1
In 1,940 (614.62 MB) 5 min
Read/Write 614.62 MB / 0 bytes 5 min
Out 1,940 (614.62 MB) 5 min
Tasks/Time 1,940 / 00:05:00.410 5 min

Name failure, success
Queued 2,075 (718.65 MB)

LogAttribute
LogAttribute 1.9.0.1.0.0.0-90
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

LogAttribute
LogAttribute 1.9.0.1.0.0.0-90
org.apache.nifi - nifi-standard-nar
In 0 (0 bytes) 5 min
Read/Write 0 bytes / 0 bytes 5 min
Out 0 (0 bytes) 5 min
Tasks/Time 0 / 00:00:00.000 5 min

Fig. 5. Example of a NiFi dataflow pipeline which implement an ingestion task from a local filesystem

To fulfill this task, a dataset builder processes raw data and extracts relevant features (classes are reported in Fig. 4). This module extracts features provided as input and aggregates them into CSV files using an aggregation function (min, max, avg). Results are written back to the HDFS utilizing the HDFS context to get the original data path.

In addition, to enable an analysis of aggregated data, these files are imported into HIVE tables. HIVE is a data warehousing tool provided by the Hadoop stack, which includes a SQL-like language (HIVEQL) to query data. To import data into HIVE tables, we define a general schema to match the structure of data points. The table schema for representing data points is read from the aggregated CSV created by the dataset builder. A general table schema representing data for a generic point is structured as:

```
smart_board_number_point_number(RecordSampleTime DATE,
MovTime FLOAT, current_mA FLOAT, voltage_V FLOAT)
```

The designed HIVE tables store aggregated data containing the extracted features obtained from raw data. Results of aggregation extract four features respectively representing: 1) a timestamp in which sample was collected, 2) the estimated time to complete the operation, 3) the average current expressed in mA issued by the point 4) the average voltage V. Features 2, 3 and 4 are obtained by the aggregation of single measurements contained in the original data.

Service layer acts as a presentation layer. It implements all the tasks needed to build models for analytics as well as graphical visualize data. These tasks are fulfilled by Jupyter notebooks. Notebooks are designed to support the workflow of scientific computing, from interactive exploration to publishing a detailed record of computation. The code in a notebook is organized into cells, chunks which can be individually modified and run. The output from each cell appears directly below it and is stored as part of the document [5]. In addition, a variety of languages supported by notebooks allows integrating different open-source tools for data analysis like Numpy, Pandas, and Matplot [7]. These tools allow to parse data in a structured format and to perform data manipulation and visualization by built-in libraries. In addition, the data structure adopted by Pandas, named, DataFrame, is widely adopted as input format by a variety of analytical models offered via machine learning libraries like scikit-learn and SciPy.

Ingestion layer has been realized through Apache NiFi, a dataflow system based on the concepts of flow-based programming. Dataflows specify a path that describes how data are extracted from the external sources and stored on the platform. An example of DataFlow, which combines data coming from an external filesystem, is provided in Fig. 5. The flow files created by the dataflow are then written to the HDFS. In the reported example, the files read from a local filesystem are unpacked and then written into a specific folder on the HDFS. This folder is created by extracting meaningful information necessary to identify the smartboard where the data comes from as well as the point which produced that data.

The proposed architecture has been employed for the collection and processing of data of the railway line Milano-Monza-Chiasso. This line is composed of 72 points forming the railway track, which are managed by multiple smart boards that collect data. In particular, data are collected from 7 points which produces roughly 32 GB/month. Data produced by the system contains information about points status and type of commands issued by the interlocking to move points. The definition of a data management policy allows to collects, govern, and controls raw data as well as enabling data analysis for end-users. The proposed platform has been deployed in a test environment using a containerization technology Fig. 6.

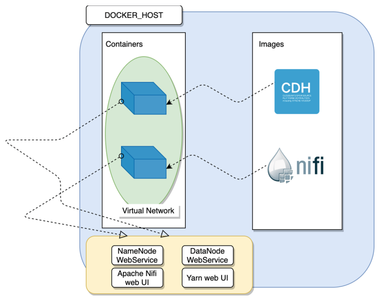


Fig. 6. Architecture deployment using containers.

We adopted two separate containers, which respectively implement the data storage & processing layers plus the ingestion layer in a separate environment. These containers communicate over a virtual network, which allows exchanging data in an isolated environment exposing web services to access the platforms and perform tasks. This deployment enables scalability of the architecture by moving containers on a cluster. Cloudera pre-built image has been adopted as a container implementing the Hadoop stack, while a separate container based on

Apache NiFi has been proposed to perform the ingestion tasks. A docker image containing the proposed platform is made available for further testing[3].

6 Example of Failure Detection Using LSTM

As an example to show the effectiveness of the proposed architecture, we report the creation of a Long Short- Term Memory (LSTM) model for failure detection of a specific railway point along the railway line Milano-Monza-Chiasso. LSTM models are a special kind of Recurrent Neural Networks (RNN) widely employed by both Academia and Industries for failure prediction [6]. A key aspect of RNN is their ability to store information or cell state for use later to make new predictions. Therefore these aspects make them particularly suitable for analysis of temporal data like analysis of sensor readings for detecting anomalies. For the considered scenario, we use sensor reading collected from a specific switch point positioned along the line. Data originated from the point includes measurements of power supplied to the object, voltage, and time of movement (to move from a normal to a reversal position or vice-versa) of types described in Sect. 3 and reported in Fig. 2. Once data are ingested, we use components composing the Processing Layer to produce useful datasets for anomalies detection using the techniques described in the previous sections. Results of the aggregation process produce a dataset consisting of 2443 samples, which are used as input for training the model. The evaluation part is performed using reference data, which represents threshold values above which failures occur (89 samples). Examples of features used for training the model are reported in Fig. 7, 8, 9.

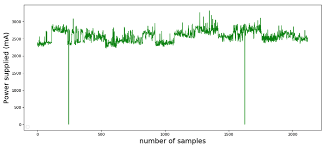


Fig. 7. Extracted feature used as model input, representing power supplied to a point

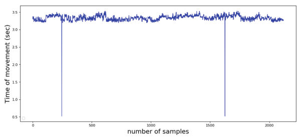


Fig. 8. Extracted feature representing time of movement in seconds

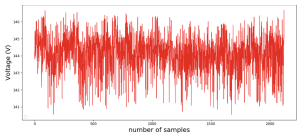


Fig. 9. Extracted feature representing emitted voltage

The autoencoder model used to make predictions learns a compressed representation of the input data and then learns to reconstruct them again. The idea is training the model on data not containing anomalies; therefore, the model will likely to be able to reconstruct healthy samples. We expect that until the model predicts healthy samples, its reconstruction error (representing the distance between the input and the reconstructed sample) is low. Whenever the model processes data outside the norm as the ones represented by the reference

[3] docker pull julio92sg/data:cloudera-hadoop-nifi.

data, which consist of threshold values, we expect an increase in the reconstruction error as the model was not trained to reconstruct these kinds of data. Therefore, we use the reconstruction error as an indicator for anomaly detection. Figure 10 reports anomalies detected by trying to predict reference values. In particular, we will see an increase of the reconstruction error on those values which are greater than a threshold of 0.25. This threshold was obtained by computing the error loss on the training set. Therefore, we identified this value suitable for the point considered as a case study, but it varies according to the particular behavior of the object. For example, considering two objects having the same characteristics (e.g., switch points) placed in different railway network topologies, may have different behaviors; therefore, they must be analyzed using different prediction models.

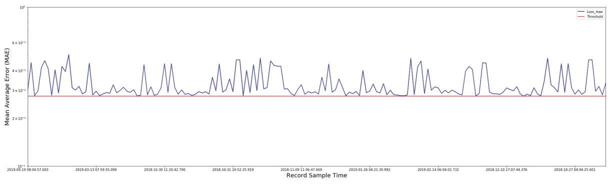


Fig. 10. Anomalies detection on a railway switch point of railway line Milano-Monza-Chiasso.

7 Conclusion

This paper proposes a novel architecture for big data management and analysis of railway data. Despite big data attract railways industries, many challenges have to be faced to enable effective big data analysis of railway data. This work proposes a four-layer architecture for enabling data analytics of railway points. Each layer is loosely coupled with others; therefore, it enables the integration of diverse data processing components intra-layers and inter-layers. To show the effectiveness of the proposed architecture, we reported the analysis of a railway switch points using predictive models for detecting failures. Nevertheless, instead of being task-oriented, the proposed architecture integrates different data processing tools to perform diverse analytical tasks as real-time data analysis. A data governance policy has been defined to deal with the variety and the complexity of railway data making them easily manageable at different granularity levels. A containerized deployment has been proposed to scale the architecture on a cluster, increasing up its scalability, thus enabling parallel data processing. Our architecture can also be extended according to the nature of the task to perform. In fact, it allows practitioners to extend architectural components to fulfil different tasks not limited to failure prediction. Moreover, in this work, we did not consider any real-time scenario in which data must be analyzed using streaming techniques, but the architecture flexibility also allows to deal with such cases. As future work, the analytical layer will be extended, proposing a comparison of different classes of predictive algorithms to measure their accuracy in diverse predictive maintenance tasks of a railway system. Moreover, we aim to extend the scope of the architecture by monitoring other kinds of infrastructures, including but not limited to power grids and highways intelligent systems.

References

1. Banci, M., Fantechi, A.: Geographical versus functional modelling bystatecharts of interlocking systems. Electron. Notes Theor. Comput. Sci. **133**, 3–19 (2005). https://doi.org/10.1016/j.entcs.2004.08.055. Proceedings of the Ninth International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2004)
2. Fang, H.: Managing data lakes in big data era: what's a data lake and why has it became popular in data management ecosystem. In: 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), pp. 820–824, June 2015. https://doi.org/10.1109/CYBER.2015.7288049
3. Ghofrani, F., He, Q., Goverde, R.M., Liu, X.: Recent applications of big data analytics in railway transportation systems: a survey. Transp. Res. Part C: Emerg. Technol. **90**, 226–246 (2018). https://doi.org/10.1016/j.trc.2018.03.010
4. Jagadish, H.V., et al.: Big data and its technical challenges. Commun. ACM **57**(7), 86–94 (2014). https://doi.org/10.1145/2611567
5. Kluyver, T., et al.: Jupyter notebooks ? A publishing format for reproducible computational workflows. In: Loizides, F., Scmidt, B. (eds.) Positioning and Power in Academic Publishing: Players, Agents and Agendas, pp. 87–90. IOS Press (2016). https://eprints.soton.ac.uk/403913/
6. Meyes, R., Donauer, J., Schmeing, A., Meisen, T.: A recurrent neural network architecture for failure prediction in deep drawing sensory time series data. Procedia Manuf. **34**, 789–797 (2019). https://doi.org/10.1016/j.promfg.2019.06.205. 47th SME North American Manufacturing Research Conference, NAMRC 47, Pennsylvania, USA
7. Nelli, F.: Python Data Analytics: With Pandas, NumPy, and Matplotlib, 2nd edn. Apress, USA (2018)
8. Thaduri, A., Galar, D., Kumar, U.: Railway assets: a potential domain for bigdata analytics. Procedia Comput. Sci. **53**, 457–467 (2015). https://doi.org/10.1016/j.procs.2015.07.323. iNNS Conference on BigData 2015 Program San Francisco, CA, USA 8-10 August 2015
9. Xu, Q., et al.: A platform for fault diagnosis of high-speed train based on big data - project supported by the national natural science foundation, china (61490704, 61440015) and the national high-tech. r&d program, China (no. 2015aa043802). IFAC-Papers OnLine **51**(18), 309–314 (2018). https://doi.org/10.1016/j.ifacol.2018.09.318. 10th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2018