National and Kapodistrian University of Athens
Department of Physics and Informatics and Telecommunications

Master Thesis in Control and Computing

Application of Machine Learning techniques in Cloud Services

Antonia Pelekanou
Student ID: 2015518

Supervisor
Assistant Professor Anna Tzanakaki

Athens, October 2017

Master Thesis in Control and Computing

Title: "Application of Machine Learning techniques in Cloud Services"

Author: Antonia Pelekanou

Student ID: 2015518

Supervisor: Assistant Professor Anna Tzanakaki

Evaluation Committee: Assistant Professor Anna Tzanakaki,
Associate Professor Dionysios I. Reisis,
Associate Professor Hector E. Nistazakis

Master Thesis submitted October 2017

Key words: Big Data, Cloud, IoT, Machine Learning, Neural Networks, Forecasting

ABSTRACT

This thesis focuses on machine learning techniques that can be used in support of cloud services. The motivation behind this work is the necessity of manipulation and processing data comprising multiple sensor measurements. These measurements were collected by sensors that were installed on the Reims tramway. The concepts of Big Data are studied, while different Data Mining methods and Machine Learning algorithms are presented. A review of Cloud Computing, Internet of Things and SiteWhere are also provided. Two different neural network structures, Multilayer Perceptrons and Long Short-Term Memory to process and forecast the data composed of various physical quantities of the Reims tramway are studied. The results produced indicate that Long Short-Term Memory is more suitable than Multilayer Perceptrons for this forecasting problem.

ACKNOWLEDGEMENTS

I would like to express my gratitude to Professor Anna Tzanakaki, my thesis supervisor, for her guidance and encouragement. The door to Prof. Tzanakaki's office was always open whenever I had a question about my research or writing. She allowed this paper to be my own work, while with her useful critiques steered me in the right direction.

I would also like to extend my thanks to Dr. Markos Anastasopoulos for his general advices and his suggestions concerning the way I should approach certain aspects of this research work.

Finally, I wish to thank my family for their continuous support throughout the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Author

Antonia Pelekanou

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Machine Learning and Cloud Computing are gaining increased popularity over that past few years. The reason behind this is the rapid increase of the volume of data and the necessity of their fast processing, in order to extract useful information.

The goal of this master thesis is to investigate the application of machine learning techniques in support of cloud services. In this work, we focus our effort to improve the process and analysis of the data of a real railway system (the Reims tramway) by forecasting the train power consumption. We develop a machine learning technique based on Neural Networks to process the dataset composed of various physical quantities related with the Reims tramway, each one measured periodically every second during the period of a day. The measurements were collected by different sensors, which were installed on the train. Initially, we provide the required theoretical background and then we present some relevant experimental results. The theoretical background involves the introduction of the concept of Big Data, the presentation of different Data Mining methods and Machine Learning algorithms that can be used in data processing and a review on Cloud Computing, Internet of Things (IoT) and SiteWhere.

In Chapter 2, the problem definition is presented and the terms of Cloud Computing and Internet of Things are introduced. In Chapter 3, we introduce the notions of Big Data, Data Mining and Machine Learning. In Chapter 4, we begin with the presentation of the architecture and capabilities of SiteWhere and we conclude with the visualization of the train data, using a Graphical User Interface (GUI), referred to as MongoDB Compass. The train dataset was stored in the MongoDB database, which is supported by the SiteWhere platform. Chapter 5 provides a brief overview of Neural Networks and the relevant basic algorithms. In Chapter 6 we introduce the concept of Time Series, implement two different types of Neural Networks, the Multilayer Perceptrons (MLP) and the Long Short-Term Memory (LSTM) on the train dataset and we present a set of relevant results.

Chapter 7, provides the summary of the thesis and the conclusions derived and presented in the previous chapters.

Chapter 2 Scenario Description

The increasing advances in hardware technology, such as the development of miniaturized sensors, GPS-enabled devices and accelerometers result in greater access and availability of different kinds of sensor data. This produces the collection of enormous amounts of data, that can be mined for a variety of analytical insights. Hence, sensor data introduce a lot of challenges in terms of data collection, storage and processing. In order to provide ubiquitous and embedded sensing there is a requirement for data collection from different devices that are connected and accessible through the Internet. This paradigm is referred to as the Internet of Things (IoT). The IoT term is used to describe "a world – wide network of interconnected objects uniquely addressable, based on standard communication protocols" [1] [2], whose point of convergence is the Internet [3].

In this thesis, we describe and propose the SiteWhere, an IoT platform in order to collect, store and process data that consists of multiple sensor measurements from Reims tramway. The Reims tramway plan is depicted in Figure 1. SiteWhere provides the required functionalities for ingestion, storage, processing and integration of device data and deal with all above-mentioned challenges caused by sensor data. It also utilizes datastore technologies, like MongoDB, Apache HBase and InfluxDB to store data. In this thesis we store the data in the MongoDB database and retrieve, store locally and process them.

An important problem related with the sensor processing is the multivariate time series modeling and forecasting. In this work, we focus our effort to improve the process and analysis of the train's data, in part by forecasting the train power consumption. Accurate modeling and prediction of the Reims tramway data is therefore of great importance and is the motivation behind this thesis.

In Chapter 6, an empirical comparison between MLP and LSTM neural networks is performed. We consider two different LSTM types and compare their performance with MLP for making a one step-ahead prediction.

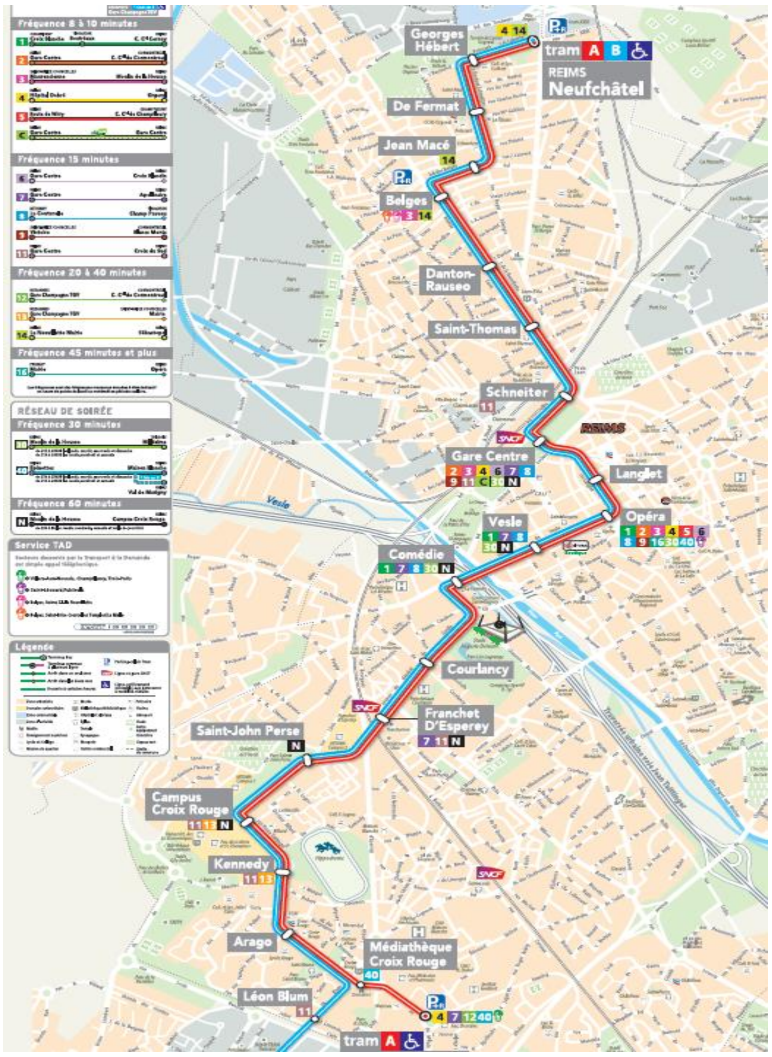


Figure 1: Reims tramway plan.

The dataset used, is part of a larger dataset, collected from sensors on the Reims tramway. This part of the dataset contains the following quantities:

i. External Temperature (measurement spot 1)
ii. External Temperature (measurement spot 2)
iii. Longitude
iv. Latitude
v. Station
vi. Velocity
vii. CO_2 Level (inside the coaches)
viii. Total Power

2.1 Cloud Computing

With the significant advances in processing and storage technologies and the success of the Internet the realization of the cloud computing has been enabled. This computing model, provides the essential computing services to support a wide variety of services available to organizations, businesses and the public. In the cloud computing model resources are provided as general utilities that can be leased and released by users in an on-demand basis [4].

One definition of cloud computing provided by the National Institute of Standard and Technologies (NIST):

"*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to shared pool of configurable computing resources (e.g., networks, servers, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction* [5].*"*

In the context of cloud computing the elements of the network representing the rendered services are invisible to the end users, as if it is obscured by a cloud [6]. This concept is depicted in Figure 2.

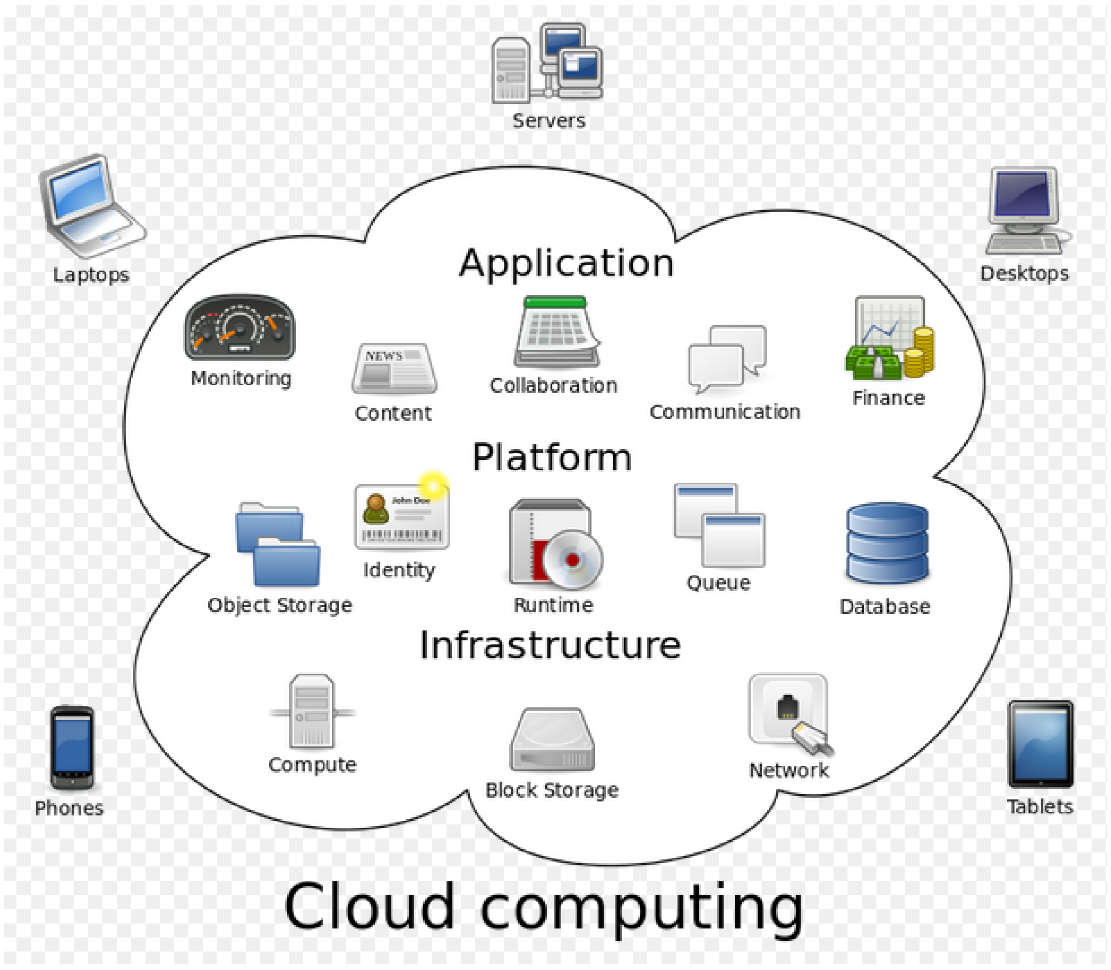


Figure 2: Cloud computing metaphor [6].

The cloud model is composed of five cloud computing characteristics, three different service models and four types of clouds [5].

Cloud Computing Characteristics

1. *On demand self-service.* A consumer can allocate or deallocate computing resources automatically, in an on-demand fashion without requiring human interaction. This automated resource management attains high performance and agility.

2. *Broad network access.* Computing capabilities are accessible through the Internet. Hence cloud services are available to be used by any device with Internet connectivity (e.g. mobile phones, tablets, and laptops).

3. *Resource pooling.* The provider's computing resources are pooled, thus they can be dynamically assigned and reassigned to multiple resource consumers according to demand. Each costumer can specify the location of the provided resources only at a higher level of abstraction (e.g. country, state, or datacenter) while he is not able to know the exact location.

4. *Rapid elasticity.* Computing capabilities are elastically provisioned and released in order to scale rapidly in accordance to the level of demands.

5. *Measured services.* Cloud computing employ a pay-per-use model, so computer systems ought to use a metering capability at some level of abstraction vary from service to service. As a result, resource usage can be monitored, controlled and reported providing transparency for both the provider and resource consumer.

Service Models

1. *Software as a Service (SaaS).* The consumer is capable to use the provider's applications. These applications are accessible from various client devices and are running on a cloud infrastructure.

2. *Platform as a Service (PaaS).* The resources provided to the consumer include operating system support and software development frameworks. Therefore, the consumer is able to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services and tools.

3. *Infrastructure as a Service.* IaaS provide various computing resources (e.g. processing, storage, and networks) where arbitrary software can be deployed and run by the consumer.

Types of clouds

1. *Private cloud.* Private clouds are offered for exclusive use by a single organization. They may be built and managed by the organization or other external providers. It may exist on or off premises of cloud provider.

2. *Community cloud.* Community clouds are designed for exclusive use by a specific community of organizations that have shared concerns. It may be built and managed by one or more organizations in the community or by external providers. It may exist on or off premises of cloud provider.

3. *Public cloud.* A public cloud can be used by the general public and it may be built and managed by a business, academic, or government organization, a third party or combination of them. It exists on the premises of the cloud provider.

4. *Hybrid cloud.* Hybrid cloud is a combination of two or more cloud infrastructures (e.g. private, community, or public). In a hybrid cloud the entities preserve their uniqueness.

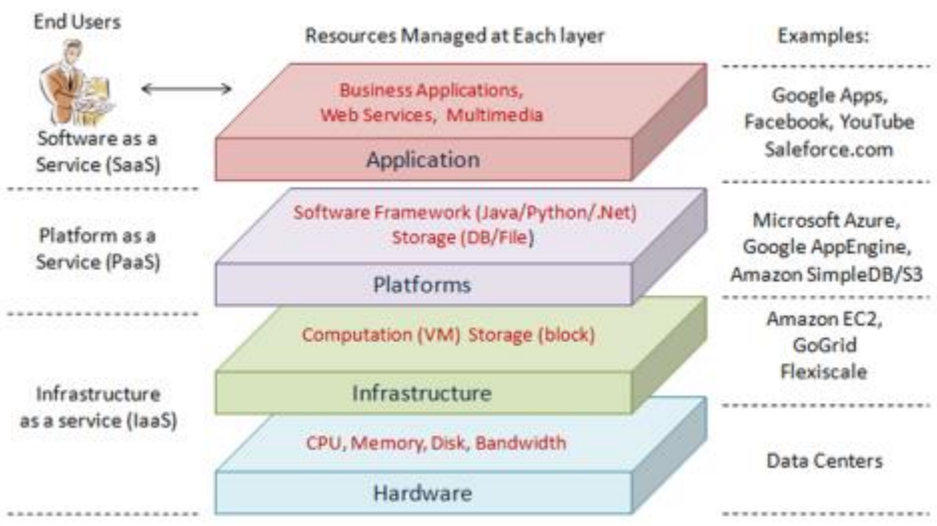

End Users
Resources Managed at Each layer
Examples:
Software as a Service (SaaS)
Business Applications, Web Services, Multimedia
Application
Google Apps, Facebook, YouTube Saleforce.com
Platform as a Service (PaaS)
Software Framework (Java/Python/.Net) Storage (DB/File)
Platforms
Microsoft Azure, Google AppEngine, Amazon SimpleDB/S3
Infrastructure as a service (IaaS)
Computation (VM) Storage (block)
Infrastructure
Amazon EC2, GoGrid Flexiscale
CPU, Memory, Disk, Bandwidth
Hardware
Data Centers

Figure 3: The architecture of cloud computing environment can be divided into 4 layers that are illustrated in above figure [4]***.***

2.2 Internet of Things

The IoT term is used to describe “a world – wide network of interconnected objects uniquely addressable, based on standard communication protocols” [1] [2], whose point of convergence is the Internet [3]. These objects are able to identify, measure and modify the physical environment acting as sensors and actuators.

aspects that are related to the Internet of Things are the Radio Frequency Identification (RFID), the sensor networks, the addressing and the middleware.

RFID systems are attached to objects to provide the automatic identification of them and allow them to be assigned to unique digital identities, to be integrated into a network and to be associated with digital information and services [3]. RFID systems composed of two parts, the transponders, also known as RFID tags, and the readers. The RFID tags are small microchips that comprise a memory to store electronically information and an antenna. They can be active, passive or battery-assisted passive. The active tags have battery, so they periodically transmit ID signals. In contrast, passive tags have no battery, so they use only radio energy transmitted by the reader. Finally, if tags are battery-assisted passive, they transmit ID signals when a RFID reader appears. The RFID readers usually trigger the tag transmission by generating an appropriate signal, querying for possible presence of tags and for the reception of their unique IDs [1].

Sensor networks along with RFID systems contribute to gain more information about the status of things, since they are capable to take measurements and get information about the location, movement and temperature. Sensor networks can be composed of a large amount of sensing nodes that can communicate to each other in a wireless network. Wireless sensor networks (WSNs) may provide useful data and are used in several application scenarios. However, WSNs and generally sensor networks have to tackle many communication and resources problems, such as short communication range, security, privacy, power considerations, storage, processing capabilities and bandwidth availability.

Addressing refers to the ability to uniquely identify the interconnected objects. This issue is very crucial for the success of IoT since this allows to uniquely address a huge number of devices and control them through the Internet [3].

Middleware is a software layer between the object and the application layer composed of three subs–layers, known as object abstraction, service management and service composition layer. These sub—layers are utilized in order to face the heterogeneity of the participating objects, their limited storage and limited processing capabilities and the large amount of applications that are included in the application layer. In addition, middleware has to deal with security issues and provide functionalities that manage the privacy and security of all the exchanged data. These functions could be included in an additional layer or distributed through the stack between the object abstraction and service composition layer. The last solution is used more often. An IoT paradigm and the SOA-based architecture for the IoT middleware are depicted in the Figure 4.

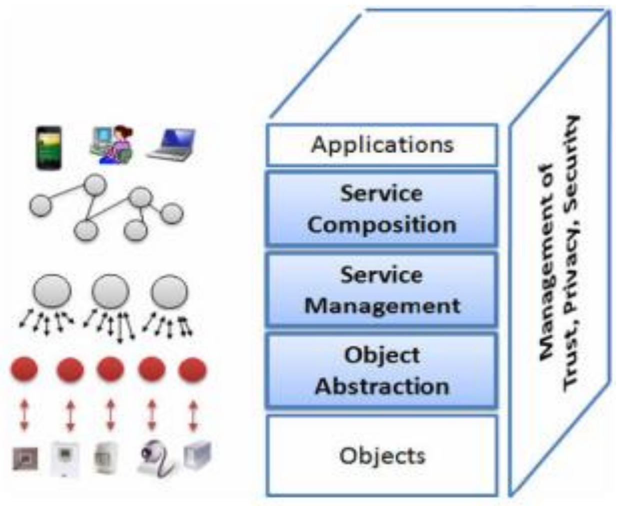


Figure 4: IoT paradigm and the SOA-based architecture for the IoT middleware [1]**.**

Chapter 3 Theoretical Background: Big Data, Data Mining, Machine Learning

The goal of this chapter is to provide the basic definitions, properties and tools which are related with the concept of *Big Data*, *Data Mining* and *Machine Learning.* These terms are often used interchangeably, so it is decided to discuss these in the same chapter. Hadoop and all the components included in the Hadoop ecosystem are significant tools correlated with Big Data. The essence of data mining and machine learning are introduced and various data mining methods and machine learning algorithms are summarized. Data mining is the process of learning patterns and identifying predictive models from large-scale datasets. It usually uses machine learning algorithms in order to extract valuable information from the data.

3.1 Big Data

A widely used definition of Big Data is derived from Gartner. In 2012, Gartner defined the exploding amount of data as follows:

"Big data is high volume, velocity, and/or variety information assets, that require new forms of processing to enable enhanced decision making, inside discovery and process optimization [7]*."*

According to the above definition one understands that *volume*, *velocity* and *variety* are three basic characteristics of the big data. The terms volume, velocity and variety referred to the huge amount of data, the speed that data are stored, managed and manipulated at the right time and the variety of data sources, types and structures respectively.

3.1.1 Big Data Analytics

Big data analysis is the data mining field that aims to manage fast vast amounts of data. In this section, we summarize different types of data analytics.

Big data analytics are separated into two types, basic analytics and advanced analytics.

Basic analytics are used to investigate the data and it may include visualizations or simple statistics. Some significant operations of basic analytics involve: the division of data into smaller sets of data that ease the examination of them, the track of large volumes of data in real time, and the anomaly detection, for example an event with actual observation different from expected.

On the contrary, advanced analytics provide some algorithms and a lot of data mining techniques, such as innovative statistical models, neural networks and other machine learning methods for more complicated analysis. Some examples of advanced analytics include: predictive models composed of algorithms and techniques used on large volume of data to predict future outcomes, text analytics and other statistical and data mining methods and algorithms, like forecasting, classification, clustering and optimization. Nowadays, advanced analytics are becoming increasingly famous, because they achieve substantial power increase and enable new algorithm development that makes the manipulation of big data easier and more efficient.

3.1.2 Data Mining Methods

As mentioned in the previous paragraph, advanced analytics utilize various data mining techniques. Data mining is a computing process that includes basic algorithms and methods that enable gaining knowledge from big data. In fact, it is a part of a wider knowledge investigation process, which comprises preprocessing and postprocessing tasks. Some examples of preprocessing tasks are data extraction, data cleaning, data fusion, data reduction and feature construction, while some examples of postprocessing tasks are pattern and model explanation, data visualization and online updating. Data mining is an interdisciplinary subfield that merges concepts from database systems, statistics, machine learning and pattern recognition [8].

Some basic data mining postprocessing methods are presented below.

Classification

The goal of the classification process is to assign an unknown item to one of some predefined classes. This is achieved by defining a function f based on a training set that comprises many items and their corresponding class. The function defined by the training set is called classifier. After the classifier is trained, it is possible to predict the class of any unknown item. As far as machine learning goes, perspective classification is a supervised learning.

Regression

Contrary to classification process, in regression analysis the output variable takes values in an interval in the real axis. Hence, given a training set, the task is to estimate a function f, whose graph fits the data. After this function is estimated, an unknown input point can be assigned to a specific output value [9].

Clustering

Clustering is the problem of partitioning data into groups so that each group contains similar objects. The more similar the objects within a group are to each other and more dissimilar from the objects in other groups, the better the clustering is. In terms of machine learning, the exploration of clusters is an unsupervised learning because cluster analysis needs to capture the hidden structure from unlabeled data in order to group them.

Forecasting

Forecasting is a process of making predictions of future values of a variable based on past and present observations. The term prediction is similar to forecasting. The main difference between them is that forecasting is used for the estimation of values at specific future times, while prediction is used for more general estimations [10].

3.2 Machine Learning

As it has been already mentioned in the introductory paragraph, the data mining process usually uses machine learning algorithms. In this section, we will explain the basic concepts someone needs for a full understanding of machine learning by introducing basic definitions and different kinds of machine learning algorithms.

3.2.1 Machine Learning Algorithms

Machine learning algorithms can be separated into groups depending on the form by which the function f is represented. Below we present the most popular machine learning algorithms.

Decision trees: In this category, the form of the function f is a tree. Decision trees are a collection of nodes, each of which has a function f of x. There is a root node, interior nodes and the leaves nodes (nodes that don't have children). To classify a feature vector x, we start from the root node and according to the value of $f(x)$ we decide to which child or children the search must proceed. The classification task is completed when a leaf is reached. Decision trees are suitable for binary and multiclass classification and regression problems. Some examples of decision tree algorithms are the Classification and Regression Tree, the Iterative Dichotomizer3, the Decision Stump and Conditional Decision Trees.

Perceptrons: Perceptron is a function which is associated with a vector of weights $\vec{w}=[w_1,w_2,w_3,...,w_n]$ with real valued components and applied to a feature vector $\vec{x}=[x_1,x_2,x_3,...,x_n]$ with real valued components, too. Perceptron has a threshold θ. The output of the perceptron is equal to +1 if the inner product $\vec{w}\cdot\vec{x}$ is grater that threshold θ or otherwise it is equal to -1. Perceptron is suitable for binary classification [11].

Neural nets: Neural nets are networks composed of layers, each of which contains one or more perceptrons. The output of some perceptrons can be used as input to next layer's perceptrons. Neural nets are suitable for binary or multiclass classification. Multiclass classification is supported since many perceptrons can be used as outputs by associating each of them with a specific class. More details about neural nets are given in Chapter 5.

Instance – based learning algorithms: An instance – based algorithm uses a similarity measure to compare new feature vectors with the data seen during the training procedure. According to the results of the comparison, it makes predictions and classifies new data properly. An important kind of instance based algorithms is the k-Nearest Neighbor algorithm (kNN).

Support – vector machines: The goal of support vector machines is to find a hyperplane $\vec{w}\cdot\vec{x}+\vec{b}=0$ that separates the points of the training set into two classes and maximizes the distance γ between the hyperplane and the points simultaneously. Support vectors are the points that are at distance γ from the hyperplane [11]. Support vector machines are used for classification and regression analysis.

3.2.2 Training Set

The training set contains the data that are given to machine learning algorithms for learning. It comprises a set of pairs $\left(\vec{x}, y\right)$, where $\vec{x}$ is a vector of categorical or numerical values, known as feature vector and y is the label that denotes the class in which $\vec{x}$ belongs. There are many types y of that define the type of the machine learning problem. To be more specific, if y is a real number, then the machine learning problem is a regression problem. In contrast, if label y is a Boolean value -1 (false) or +1 (true) or a member of some finite set, then the machine learning problem is a binary or a multiclass classification problem respectively. Finally, the label y may be a member of potentially infinite set, such as a parse tree for $\vec{x}$ [11].

3.2.3 Supervised, Unsupervised and Semi - Supervised Learning

Machine learning can be categorized into supervised, unsupervised and semi – supervised depending on the type of the given training set.

Supervised learning: the training set is composed of input variables $x_1, x_2, x_3, ..., x_n$, known as features and output variables $y_1, y_2, y_3, ..., y_n$, known as labels. In this case, the algorithm learns how to map each input x_i to the corresponding label y_j and designs a function, which is capable of predicting the output label given any new input. Some supervised learning algorithms are the linear regression, random forest and support vector machines.

Unsupervised learning: the training set comprises only the input variables $x_1, x_2, x_3, ..., x_n$ and no corresponding labels. The unsupervised learning algorithms have to model the structure or the distribution in the input data. Some examples of unsupervised machine learning algorithms are the clustering and the association rule problem.

Semi – supervised learning: the training set is composed of a large amount of unlabeled input data. In semi – supervised learning class only a small percentage of the input data include their corresponding labels.

3.2.4 Data Preprocessing

Data preprocessing is required before machine learning algorithms are applied to training sets, because the quality of the data and the information that they contain are two factors that affect the learning performance. Below we present some needful preprocessing techniques. Some of them will be used in Chapter 6.

Missing values: missing values in real world datasets are a common incident that reduce the representativeness of the samples and can cause unpredictable results if they are ignored. There are several preprocessing techniques to treat the missing data, like mean imputation or simply removing the features or samples that contain missing values. To eliminate features and samples with missing values is a convenient approach, but if many features or samples are removed from dataset, it can lead to the loss of valuable and useful information. In contrast, mean imputation is a safer approach. Mean imputation is an interpolation technique that substitutes missing values by the mean value of the total feature column.

Training set and test set: Dividing dataset into training and test set is an effective approach to avoid overfitting. Overfitting occurs when the machine learning algorithm performs well only on the training set, but it is unable to generalize well to new data. So, to avoid overfitting the training set is used to train and optimize the model, while the test set is used to estimate the accuracy of the final model and indicate if the model is overfitting the data. The separation of the dataset into training and test set must reserve the valuable information in the training set. However, smaller test sets may lead to less accurate estimation of the generalization error. The most commonly used splits are 60-40, 70-30 or 80-20 calculate on size of the initial dataset.

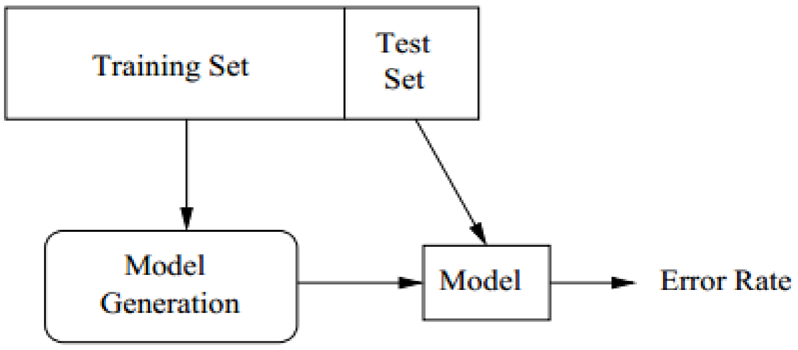


Figure 5: Training set derives the model and test set evaluate the final model.

Feature scaling: Feature scaling is an essential step in data pre-processing, since machine learning algorithms behave better if the values of feature vectors are on the same scale. We present an example to understand the importance of feature scaling in machine learning.

Example: If there are two features, one of which is measured on scale from 1 to 10 and the second on scale from 1 to 100,000, then the squared error function in a classifier will focus on optimizing the weights to larger errors in the second feature [12].

The well-known techniques for feature scaling are the normalization and standardization techniques.

In normalization, the min – max scaling is applied to each feature column, so that each sample x_i is replaced by the new value x_{norm}, where x^i_{norm} is calculated as follows:

$x^i_{norm} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$, where x_i is a particular sample, $x_{\min}$ is the minimum value in the feature vector and $x_{\max}$ is the maximum value in the feature vector.

Standardization is a technique that can be more practical for many linear models which are used to initialize weights to zero or small random values close to zero. It centres the feature columns at mean zero with standard deviation equal to one, thus the feature columns take the form of a normal distribution which in turns eases the algorithm to learn the weights. In addition, standardization makes the algorithms less sensitive to outliers by holding useful information about them. The new value x^i_{std} of each sample x_i is calculated as follows:

$x^i_{std} = \frac{x_i - \mu_x}{\sigma_x}$, where μ_x is the sample mean of a particular feature vector and σ_x the corresponding standard deviation [12].

Regularization: The goal of a machine learning algorithm is to find a set of weight coefficients that minimizes the error on the training data. Regularization is a preprocessing technique that adds a penalty term to the initial error in order to penalize complex models. Finally, instead of the error on the data, the learning algorithm must minimize an augmented error which is given by the equation:

$$E = error\ on\ data + \lambda \cdot \mathrm{mod}el\ complexity$$

The parameter λ gives the weight of the penalty term [13].

Dimensionality reduction: Feature selection and feature extraction are two fundamental techniques for dimensionality reduction. In feature selection, algorithms intend to create a subset composed of the most relevant to the problem original features and remove the irrelevant features. In this way, they achieve to reduce the dimensions of the initial feature subspace. In feature extraction, algorithms derive the valuable information from the initial dataset and project the data onto a new feature subspace of lower dimensionality. Feature extraction can be understood as data compression with the goal of maintaining most of the relevant information [12]. The Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are basic algorithms that are widely used for feature extraction and dimensionality reduction.

3.3 MapReduce and Hadoop Ecosystem

In this part, we will outline the MapReduce model and other tools, like Apache Hadoop and the Hadoop-related Apache projects Pig, Hive, HBase and Mahout. Big data analysis requires large amount of data to be managed with high speed. Computing clusters and Distributed File Systems (DFS), enable the parallelization of various operations, like computations and data transfer. Based on the DFS, many higher-level programing systems have been developed, with most significant the MapReduce system.

MapReduce

MapReduce is a programing model, that is used for data administration and processing. It is composed of two functions, the Map and the Reduce functions. The Map function applies an operation to a specific part of data and provides a set of intermediate key/value pairs. The Reduce function takes an intermediate key and a set of values related with that key, merges together these values and provides the final output.

The simplest example to get an understanding of MapReduce is the word count example, in which the only operation is to count how many times each word appears in a collection of documents. In other words, the task is to create a list of words and identify the frequency with which they appear in order to find the relative importance of certain words. In the following example, the input of the Map function is a line of text. At first, the Map function analyses the text string into individual words and produces a set of key/value pairs of the form <word, 1>. Finally, the Reduce function sums up the 1 values and provides the <word, count> pair, which is the final output. The MapReduce example is illustrated in Figure 6 [14].

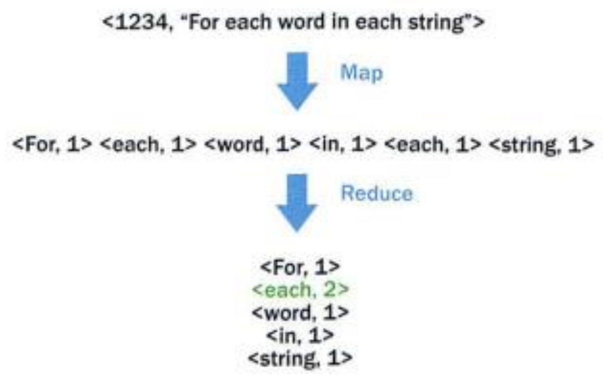

<1234, "For each word in each string">
Map
<For, 1> <each, 1> <word, 1> <in, 1> <each, 1> <string, 1>
Reduce
<For, 1>
<each, 2>
<word, 1>
<in, 1>
<string, 1>

Figure 6: MapReduce example [14]**.**

Apache Hadoop

Apache Hadoop is a project which develops open-source software for reliable, scalable, and distributed computing [15]. Hadoop successfully executes the operations of the MapReduce model at a high level. The base of Hadoop is the Hadoop Distributed File System (HDFS), in which data are distributed through a computing cluster. HDFS separates each file into fixed size blocks and stores them across a computing cluster. In addition, it creates three replicates of each block, that are stored on different nodes in the Hadoop cluster. As a result of replicates, Hadoop is capable of responding even in the case of a hardware failure and it is more flexible in determining which machine to use for the map step on a specific block. The manipulation of the data access in HDFS is accomplished by three background processes, usually called Java Daemons, each of which has a particular task. The three Java Daemons are known as Name Node, Data Node and Secondary Name Node. Name Node determines where the blocks of data are stored on a single machine, Data Node manages the stored data on each machine and the Secondary Name Node can perform some of the Name Node tasks in order to reduce its workload. Machines are classified into two categories by processes that they run. The first category is the master nodes and the second category is the worker nodes. Master nodes run the Name Node and the Secondary Name Node while worker nodes run the Data Nodes.

Apache Pig

Apache Pig is a high-level platform that contains an environment to execute the Pig code and a data flow language, called Pig Latin. The main advantage of Pig is the simplicity of developing and executing a MapReduce task. Pig instructions are translated into MapReduce jobs in background. Additionally, in Pig the execution of several common data manipulations, such as inner and outer joins between files, is the same as that of the SQL language used in a relational database. Finally, Pig language can be extended using user-defined functions (UDFs), thus some operations can be coded in other programing languages, for instance Python, JavaScript or Groovy, and then executed in the Pig environment [14].

Apache Hive

Apache Hive is a data warehouse structure built on top of Hadoop that provides data summarization, query and analysis [16]. The language of Hive is the HiveQL (Hive Query Language) and it is like the SQL language. Hive provides table structures with rows and columns. Each row represents a record, transaction or a particular entity. The value of the corresponding column represents the attribute of the row. Hive is a good tool to be used if the table is the appropriate structure for data visualization. However, Hive is not the suitable tool for real – time queries because a Hive query has to be translated into a MapReduce job and then submitted to the Hadoop cluster [14].

Apache HBase

Apache HBase is Hadoop database, that provides real-time read and write access to the big data. The HBase design is based on Google's 2006 paper on BigTable [14]. It is constructed upon HDFS. HBase achieves real-time operations very quickly by dividing the workload over different nodes in a distributed cluster. More specifically, each table is separated into a few regions and a worker node is responsible for a specific range of data in the table column. As the table size increases or the user load increases, additional nodes and region splits can be added to scale the cluster properly. The form that the contents are stored on an HBase table is a key/value form. The values represent the data stored at the intersection of the row, column and version. The key contains the next elements:

1. Row
2. Row length
3. Column family
4. Column family length
5. Column qualifier
6. Version
7. Key type

Row is used as the primary attribute to access the data of an HBase table. The way that data are distributed depends on it, so the structure of the row has to be designed according to how the data will be accessed.

Column family provides grouping for the column qualifiers.

Version or timestamp make it possible to maintain different values of the contents in an intersection of a row and a column.

Key type indicates whether a write operation corresponds to add data into the HBase table or delete data from the table [14].

Apache Mahout

Apache Mahout provides executable Java code that implements various machine learning algorithms and data mining techniques, that can be used for big data analysis. It contains classification algorithms, such as Logistic Regression, Naïve Bayes, Random forests and Hidden Markov models. It also contains clustering algorithms, like canopy, K-means, fuzzy K-means, dirichlet, mean-shift and some recommenders -

collaborative filtering, such as non-distributed recommenders and distributed item-based collaborative filtering.

Chapter 4 SiteWhere: An Internet of Things Platform

In the beginning of this chapter we describe SiteWhere, an Internet of Things solution used to manage and analyze device data. The next topic presented in this chapter is data visualization, which is performed using a GUI for MongoDB database, the MongoDB Compass.

4.1 SiteWhere

The middleware solutions discussed above are referred to as IoT platforms. In this section, we will focus our attention to SiteWhere, an open source Internet of Things platform that provides all the required functionalities for ingestion, storage, processing and integration of device data.

The core of SiteWhere relies on many proven and open source technologies in order to carry out all the required functions mentioned in the previous paragraph. An important technology adopted is that of the Tomcat Server, a java servlet container in which SiteWhere runs as a Web Application Archive (WAR) file. Some other technologies are the Spring Framework, Spring Security and Hazelcast. In addition, SiteWhere utilizes datastore technologies, like MongoDB, Apache HBase and InfluxDB to store data. Finally, it uses complementary technologies like the Mule AnyPoint Platform and Apache Spark.

SiteWhere architecture is illustrated in Figure 7. It consists of various components that will be analyzed further in the following paragraphs.

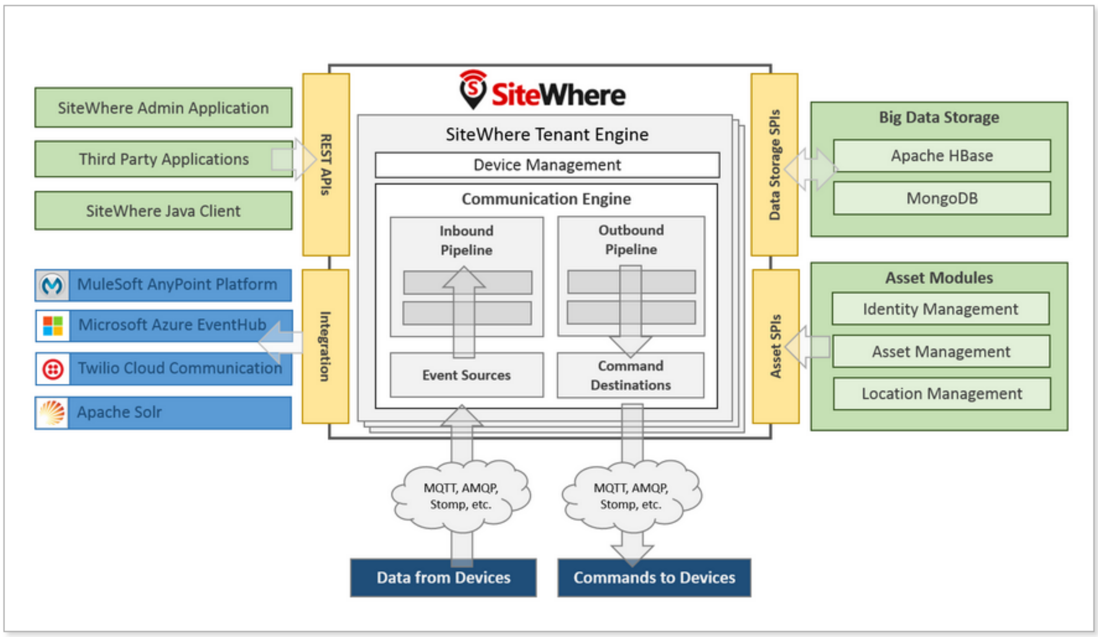


Figure 7: SiteWhere architecture. [17]*.*

A significant component is the SiteWhere Server, a central component which controls all other system's components and the processing of device data.

Another component is the Tenant Engine. A specific SiteWhere instance is capable to have multiple tenants and thus to serve multiple IoT applications. Each system tenant can have different data storage and processing pipeline without affecting other tenants. In addition, it is configured to communicate with devices over any number of most popular communication protocols, like Message Queue Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP) and Stomp. Although the default communication protocol is the MQTT.

Communication engine is a component that is encapsulated in the Tenant Engine. It ensures the internal event handling. Its operations are the registration of new or existing devices, the receipt of events from connected devices and the delivery of commands to them.

It is also important to mention that events are the core data that SiteWhere revolves around. SiteWhere manages different types of events such as measurements, locations, alerts and command invocations and responses.

Figure 8 illustrates the flow of data in the Communication Engine component.

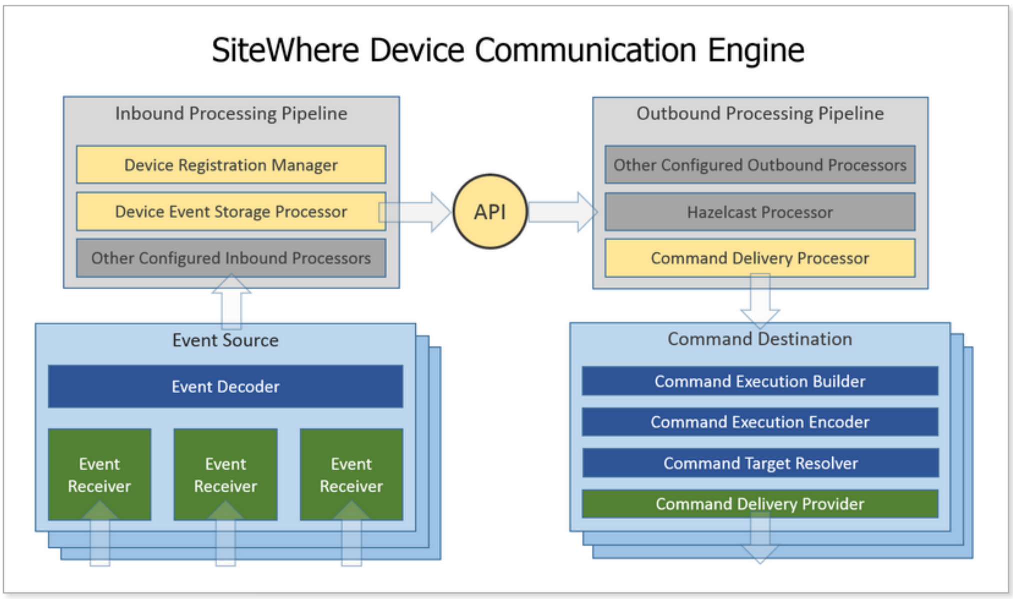

SiteWhere Device Communication Engine
Inbound Processing Pipeline
Device Registration Manager
Device Event Storage Processor
Other Configured Inbound Processors
API
Outbound Processing Pipeline
Other Configured Outbound Processors
Hazelcast Processor
Command Delivery Processor
Event Source
Event Decoder
Event Receiver
Event Receiver
Event Receiver
Command Destination
Command Execution Builder
Command Execution Encoder
Command Target Resolver
Command Delivery Provider

Figure 8: Diagram of data flow in Communication Engine component [17]***.***

SiteWhere is able to send and receive data either from devices or from other agents via Representational State Transfer (REST) services. REST services can be used by authenticated users to create, view, update or delete entities in the system. They can also be used by devices and directly interact with SiteWhere.

In addition, SiteWhere cooperates with NoSQL databases to store and retrieve the data that are collected from devices. MongoDB, Apache HBase and the InfluxDB are three databases which supported by SiteWhere. The MongoDB is a document oriented database with great performance, availability and high scalability. The HBase database is a NoSQL and distributed database in which device events are stored as time series data. A brief description of HBase has already been pointed out in Chapter 3. The InfluxDB is a time series database that supports advanced clustering and provide high scalability. The InfluxDB database also provides a tool for data visualization, reffered to as Grafana [17].

Furthermore, SiteWhere facilitates the integration with other platforms through Hazelcast, an in-memory data grid. This way, SiteWhere's functionalities and capabilities can be improved and extended.

Another important feature of SiteWhere is its capability to support several external device platforms. These include the Android Development Kit, the Arduino Development Kit and the Java Agent. As a result, the interaction and the interchange of events between the server and the Android devices, Arduino devices or Java agents are easy to take place. Moreover, the Mule AnyPoint Platform and the Apache Spark are two external platforms that can connect to a SiteWhere server instance and exchange data with it via Hazelcast services. The Mule AnyPoint is an ESB platform which manages interactions across varied systems since it supports a lot of communication protocols and the Apache Spark is a computing framework which is used to large – scale data processing.

Finally, SiteWhere makes it easy to associate the devices with physical assets, like people, places and things and describe the hardware information or configuration. Each device can be registered along with its unique hardware id and device specific metadata [17].

In summary, SiteWhere is a platform suitable to support the work of this thesis. SiteWhere can be used to facilitate network connectivity of train sensors. Through this sensor measurements can be stores in the MongoDB database of SiteWhere. If the data are stored, can then be retrieved and stored locally allowing their offline processing. In Chapter 6 we propose the use of Neural Networks for forecasting using the sensor data retrieves from the MongoDB database and present some relevant experimental results. If the sensors are connected to SiteWhere, Neural Networks can be adopted and used for real time forecasting via services provided by Hazelcast. Finally, data visualization providing immediate insight into the server status and query performance can be also provided, using various visualization tools. In the next section, we will use the MongoDB Compass and visualize the train data that we have stored in the MongoDB database.

4.2 Data Visualization

The database that was used in this thesis to store the collected train data was MongoDB. We also used MongoDB Compass for data visualization. MongoDB Compass is a graphical tool introduced by MongoDB.

It allows the user to analyze and visualize its database scheme through histograms, which represent the frequency of the data and their distribution in a collection. Figure 9 illustrate the histograms of the train data that are stored in a MongoDB database.

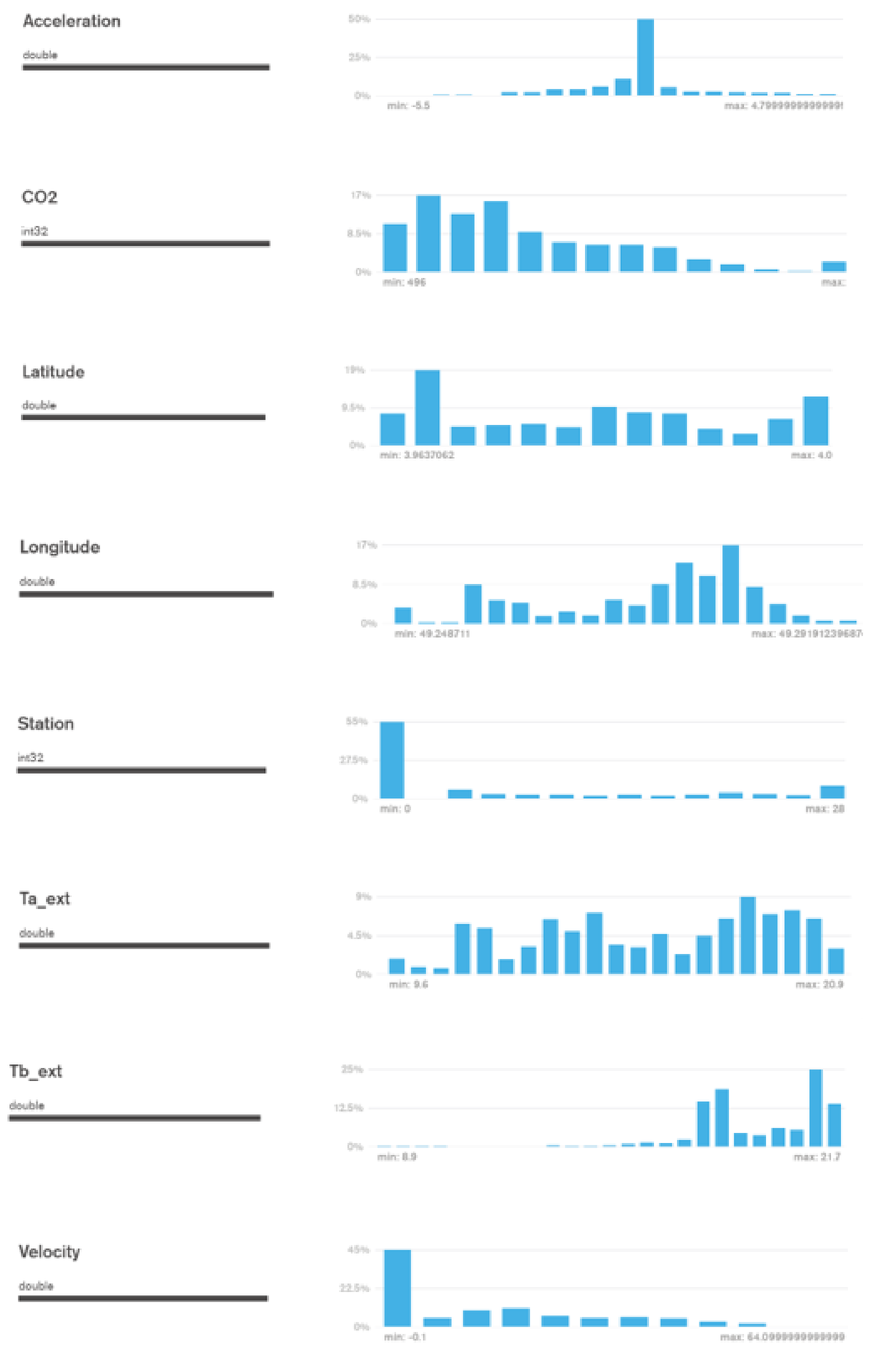
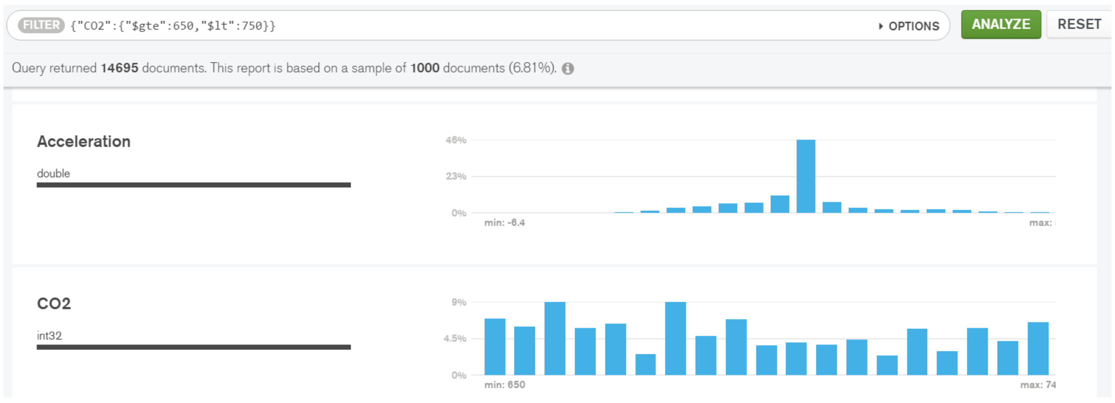


Figure 9: Histograms of the train data as represented by MongoDB Compass.

In addition, MongoDB Compass makes easier the MongoDB queries. Users can simply click on the chart or on a range of charts of the data histogram and as a result a MongoDB query will be automatically built in the query bar. Figure 10 shows results of a query about the CO2 that was graphically built using MongoDB Compass.



```
{"CO2":{"$gte":650,"$lt":750}}
```

Query returned 14695 documents. This report is based on a sample of 1000 documents (6.81%).

Figure 10: Results of a MongoDB query that was graphically built using MongoDB Compass.

Finally, MongoDB Compass can use the geospatial functions provided by MongoDB. If a geospatial index introduced in the MongoDB collection, the MongoDB Compass shows a map with identified the points of interest. This map is illustrated in Figure 11. To create the geospatial data depicted below we combined the latitude and longitude data from the collected train dataset.

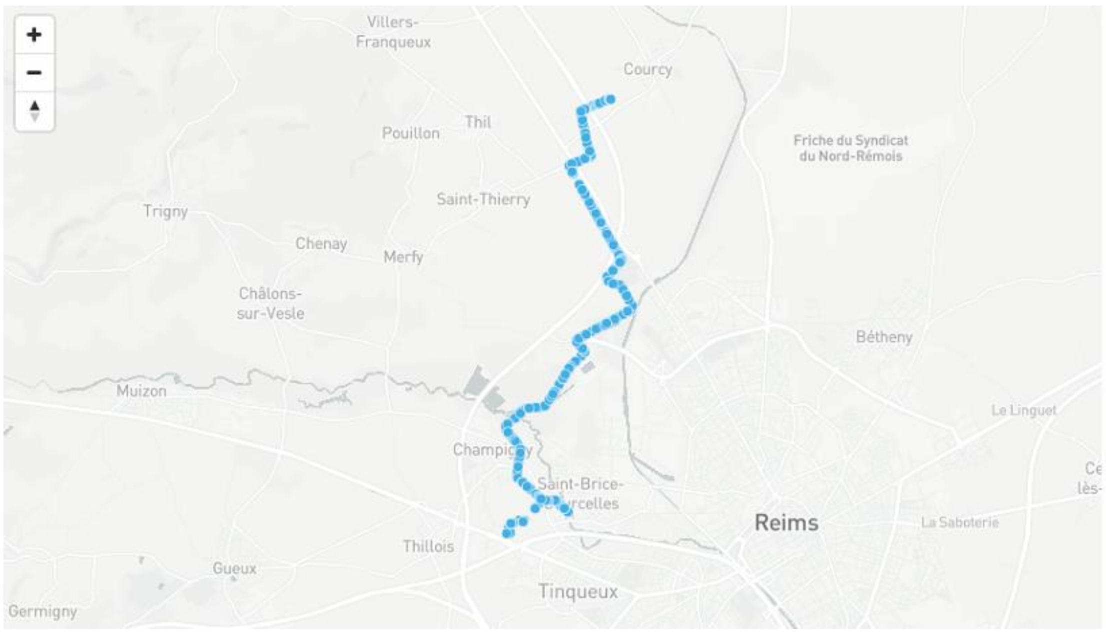
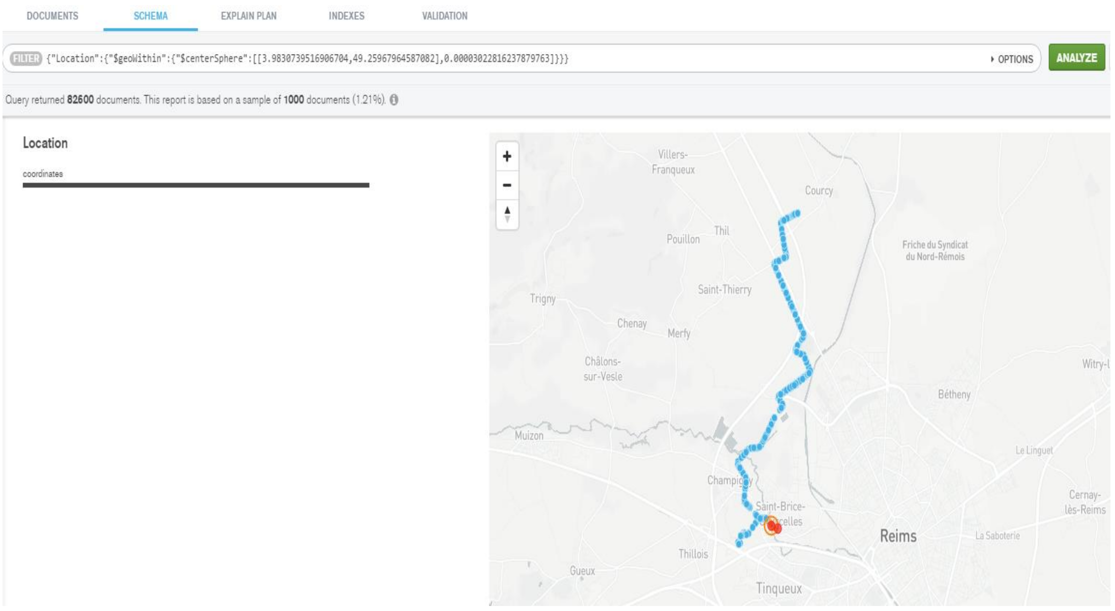


Figure 11: Map with points of a route of the Reims train.

Since geospatial data are supported by MongoDB Compass, users are able to build geospatial queries and extract the results graphically and as a JSON documents. Figure 12 shows the implementation of a query and Figure 13 shows the graphical results for the location and the power consumption of the train.s



```
{"Location":{"$geoWithin":{"$centerSphere":[[3.9830739516906704,49.25967964587082],0.00003022816237879763]}}}
```

Query returned **82600** documents. This report is based on a sample of **1000** documents (1.21%).

Figure 12: Geospatial query graphically built using MongoDB Compass.

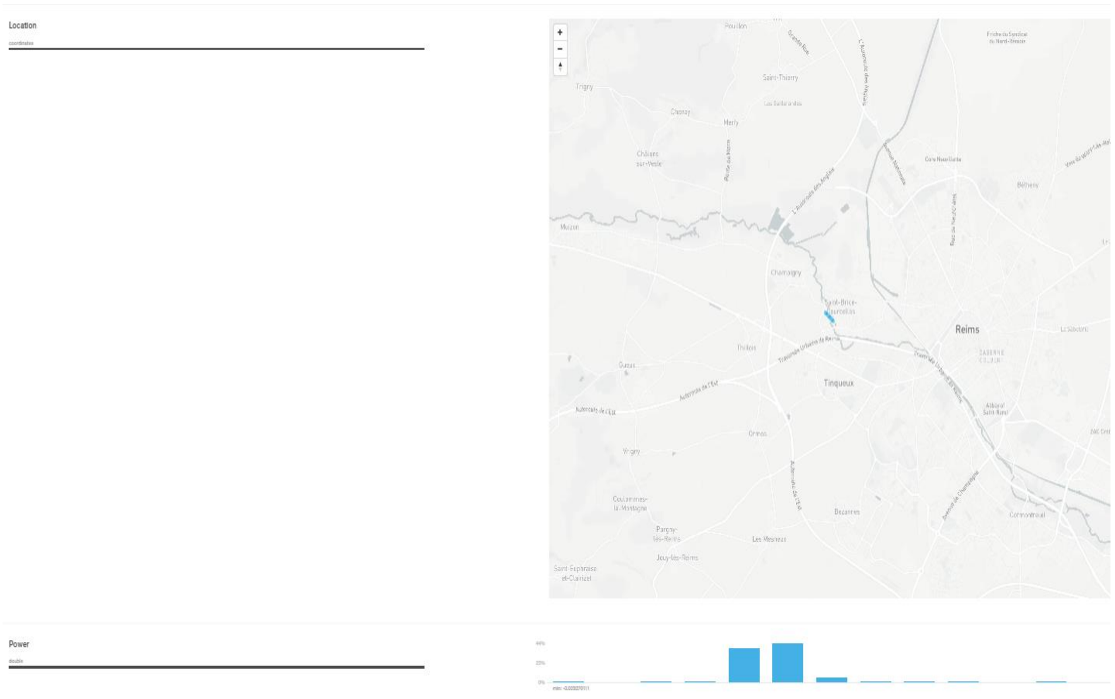


Figure 13: Graphically results of the geospatial query.

Chapter 5 Overview of Neural Networks

In this chapter, we provide an overview of Neural Networks. We introduce the different Neural Networks' architectures and some algorithms related to them, such as the backpropagation and gradient descent algorithm. Finally, we analyze further the Multilayer Perceptrons and the Long Short-Term Memory neural network.

5.1 Introduction

Before trying to give a definition of Artificial Neural Networks, we will provide a review of biological brain. The human brain consists of many interconnected neurons. Neurons are computational cells, that are responsible for processing, transmitting signals and accomplishing recognition tasks. “Warren McCullock and Walter Pitts published the first concept of a simplified brain cell, the so-called McCullock-Pitts (MCP) neuron, in 1943”. Warren McCullock and Walter Pitts correlated the nerve cell with the logic gate. These logic gates get chemical and electrical signals as input and if they exceed a threshold, then a specific output signal is generated [12].

Artificial Neural Networks, that are usually referred to as “Neural Networks”, are a replica of Biological Neural Networks. Neural Networks are composed of a group of interconnected computational units, that are called neurons and they can interact with the environment. The knowledge is stored in synaptic weights by interneurons connection strengths. Finally, knowledge is accomplished through a learning algorithm. A learning algorithm is a function that updates the synaptic weights during the learning operation [18].

There are three basic elements of the neural model:

1. A collection of synapses or connecting links between neurons. Each synapse has its own weight.
2. An adder for summing the neuron’ s input signals. Before summing, each signal has been multiplied by the corresponding synaptic weight. The aggregation is linear.

3. An activation function for reducing the amplitude of the neuron’ s output. An activation function is a non-linear transformer, that limits the amplitude of the output into a specific range of finite values.
4. An offset vector, usually called as bias, for increasing or decreasing the triggering of activation function. The bias vector is equal to synaptic weight w_{k0} and it does not depend on any input value.

The neuron model is depicted in Figure 14.

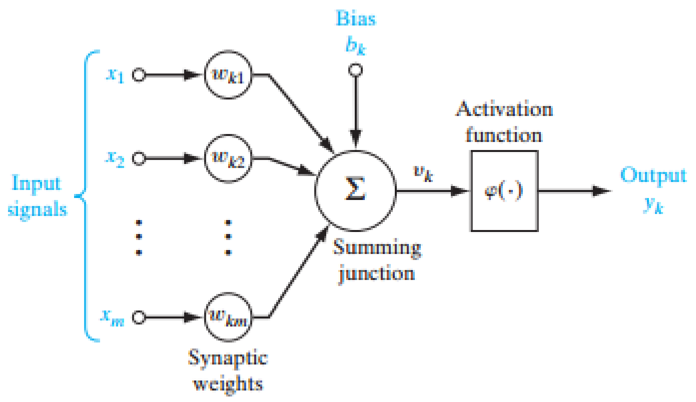


Figure 14: Non-linear model of a neuron [18].

5.2 Activation Functions

There are a plenty of activation functions. Some of them will be inferred in this part.

Sigmoid Function

The sigmoid function is a non-linear function with the following mathematical form:

$$\sigma(x) = \frac{1}{1+e^{-x}}.$$

It accepts a set of real-valued numbers and limits them into a range between 0 and 1.

Tanh function

The tanh function is a non-linear function, which squashes each real-valued number into a range between -1 and 1.

Rectified Linear Unit or ReLU function

The mathematical form of ReLU function is $f(x) = \max(0, x)$. The symbol x in the ReLU' s formula is the real-valued input of the neuron [5].

5.3 Neural Network Architectures

Neurons of a neural network are grouped in layers. While the network layout is closely associated with the learning algorithm used to train the neural network, three basic structures of neural networks are referenced below.

Single-Layer Feedforward Networks

A single-layer neural network contains two layers. The first and the second layer are composed of input nodes, that can be called "source" nodes, and output nodes respectively. Even though there are two layers, it is considered that there is a single layer of neurons, the output layer. The reason is that only the output nodes perform computations. The source nodes of the network project input vectors straight onto output neurons. The input signal does not flow backward as the network architecture is a feedforward type. A single layer feedforward neural network is illustrated in Figure 15 [18].

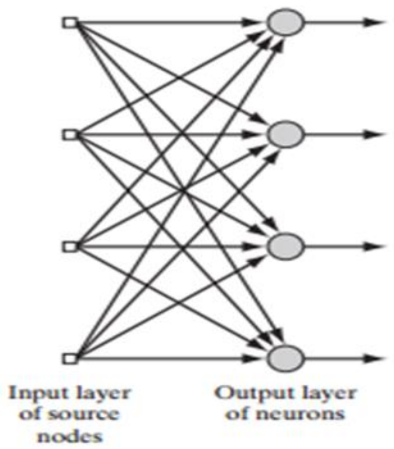


Figure 15: Feedforward network with a single layer of neuron [18]**.**

Multilayer Feedforward Networks

The second category of neural networks architecture comprises two or more hidden layers. The neurons of a hidden layer do not communicate directly with the input and output layers. The source nodes transfer the input data to the second layer. Then the output of second layer is transmitted to the input of the third layer and so on. It is important to realize that if a network has more neurons and layers, then parallel computations can be accomplished because the information is distributed to more neurons. Like the first class, the input signal does not flow backward. A multilayer feedforward network is illustrated in Figure 16. The illustrated neural network is a fully connected one. The term "fully connected" refers to the fact that every node in each layer is linked with each node in the next layer. There are multilayer feedforward networks that are partially connected [18].

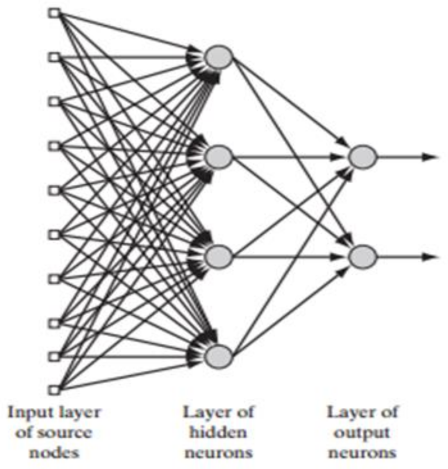


Figure 16: Fully connected feedforward network with one hidden layer and one output layer [18]**.**

Recurrent Networks

The difference between the above-mentioned feedforward networks and the recurrent networks, usually called RNN, is that the latter has at least one feedback loop. More specifically, the recurrent network may have a layer, in which neurons feed other neurons of the network with their outputs. This feedback loop allows information to persist. A recurrent network with a hidden layer is illustrated Figure 17. Hidden neurons and output neurons feed into other neurons as it is depicted below. Because of feedback loops, some synapses are composed of time delayed units so the network has non-linear performance [18] [6].

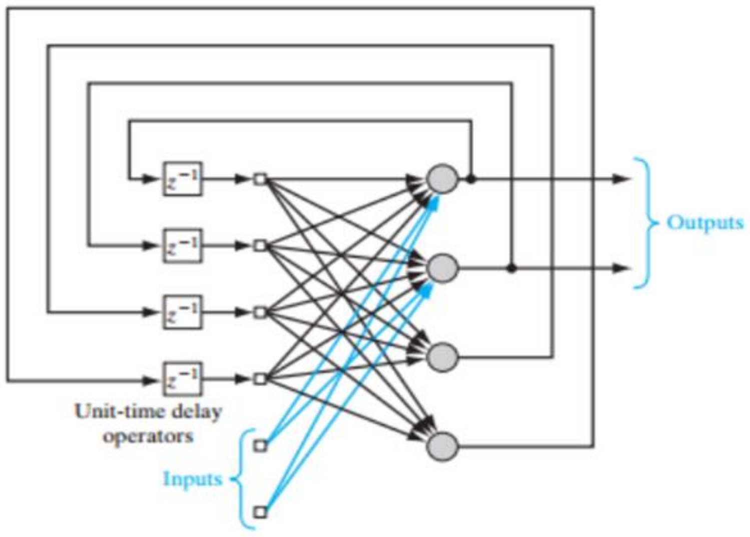


Figure 17: Recurrent Network with hidden neurons [18]***.***

5.3.1 Multilayer Perceptrons

The basic properties of Multilayer Perceptrons neural networks are:

1. Each neuron of the MLP network has an activation function that is non-linear and differentiable.
2. The network is composed of one or more hidden layers.
3. The neurons are highly connected and the extent of their connectivity is determined by the value of their synaptic weights.

A Multilayer Perceptron is shown in Figure 18, in which the MLP network contains one hidden layer.

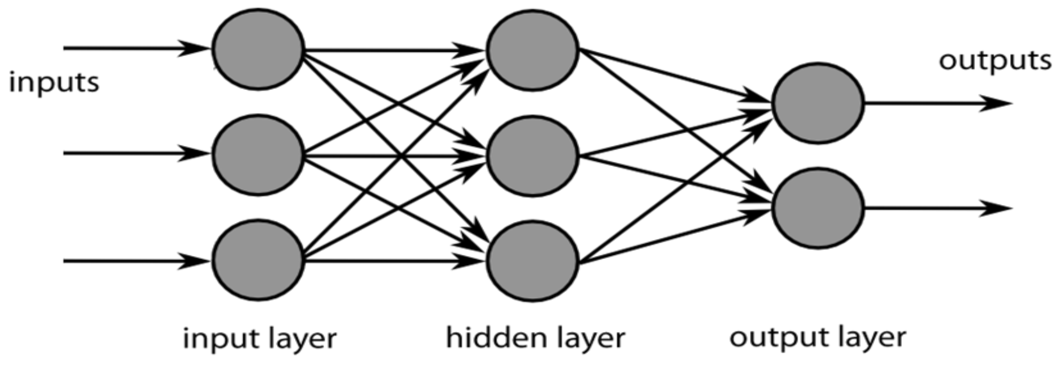


Figure 18: Multilayer Perceptrons Network with one hidden layer.

3.3.1.1 Backpropagation algorithm

The Backpropagation algorithm is the most popular learning algorithm for multilayer perceptron training. While a Backpropagation algorithm is used, the training procedure is separated into two phases, the forward phase and the backward phase. Through the first phase, the input signal is transmitted from the input to output layer, layer by layer, and the synaptic weights' values are constant. In the second phase, the comparison between the output and the desired response leads to an error signal. The error signal is propagated through the network backwards, starting from the output, and then the synaptic weights are changed to minimize the loss function. The loss function is a function that calculates the divergence between predicted and expected network's response values. The back-propagation algorithm is an efficient and famous algorithm, because it is simple to compute locally and it executes stochastic gradient descent [18].

3.3.1.2 Gradient Descent

Gradient descent is an algorithm that is used to "optimize" an objective function finding its local minimum. The local minimum can be found using gradient descent. Steps associated with the function's gradient are taken in order to converge to the local minimum.

3.3.1.3 Stochastic gradient descent

Stochastic gradient descent, is stochastic approximation of the gradient descent optimization algorithm for minimizing an objective function that is written as a sum of differentiable functions [4].

Adaptive Moment Estimation (Adam) is a class of stochastic gradient descent algorithm. In this optimization algorithm, running averages of both gradients and then the second moments of the gradient are used [4].

5.3.2 Long Short - Term Memory

LSTM (short for Long Short-Term Memory) networks are a class of recurrent neural networks, that were mentioned above. LSTMs contain a set of recurrent blocks, known as memory blocks. Each block is composed of one or more memory cells and each cell contains three units, the input, output and forget gates. LSTMs are able to learn long-term dependencies, because they remember information that was earned in previous epochs. There are some basic phases through the training procedure of LSTM network. During the first phase, LSTM decides which information it is going to discard. The unit that makes this decision is a sigmoid layer, known as “forget gate” layer. After that, the networks choose the information that they are going to keep in the cell state. In this phase, there is a combination of two layers, the sigmoid layer, usually called “input gate” layer, that decides which values will be updated and the tanh layer that selects which candidate values will be added to the state. In the next phase, the old cell state is updated to a new cell state. Finally, LSTM decides what information it is going to output. In this last phase, the sigmoid layer decides which parts of the cell state it is going to output and the tanh layer outputs them [6] [19]. Figure 19 provides an illustration of LSTM memory block with one cell.

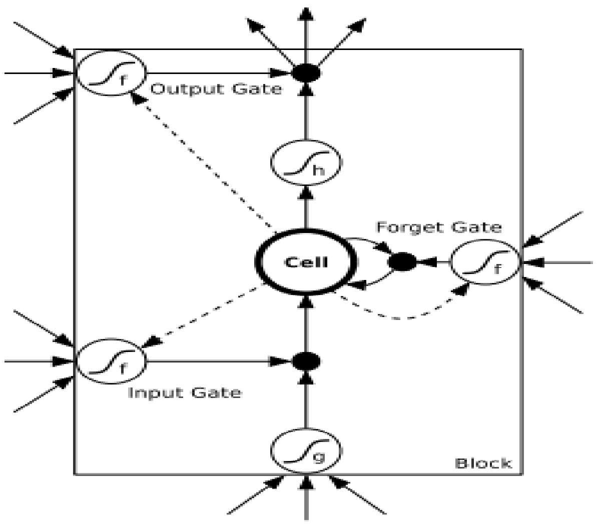


Figure 19: LSTM memory block [6]**.**

Chapter 6 Forecasting based on Neural Networks (NN) techniques

In this chapter, we introduce the notion of time series and summarize the results of the implementation of neural networks in time series forecasting. We used two different types of Neural Networks, the Multilayer Perceptrons and the LSTM network. Finally, we compare the results of these implementations.

6.1 Time Series

Time series is a set of data points, each of which is recorded at a specific time. There are two categories of time series. First class is the discrete-time time series, which is a sequence taken at fixed time intervals and second class is the continuous-time time series, which is a sequence of observations recorded continuously over some time interval [20].

Forecasting future values of an observed time series can be challenging in many fields of science and engineering. Forecasting can be carried out using different statistical methods and models, such as auto – regressive, moving average and vector auto – regressive models. Although, recent years machine learning models, like Artificial Neural Networks (ANN), LSTM Recurrent Neural Networks, Decision Trees and Support Vector Machines are widely used for time series forecasting. In next sections, we approach the time series forecasting problem using two different types of neural networks, the MLP and LSTM.

6.2 Results of MLP implementation

In this section, we will present the experimental results of the MLP implementation on the train dataset. We used MLP networks for the power consumption forecasting.

Before data pre-processing, we selected some quantities that we considered more important for power forecasting. We took line plots across time and analyzed these quantities of the initial dataset visually. The plots are illustrated in Figure 20.

The first step was to split the dataset into two parts, the training set and the test set. The training set contains 80% of the total dataset and the remaining 20% comprises the test set. The second step was to transform the time series problem into a supervised problem, which means that we used the observation at the previous time step as the model's input and the observation at the current time step as the model's output. Finally, we rescaled the data to a range of [-1,1], because machine learning and optimization models behave better, if data are on the same scale. In the following experiments, we calculate one-step ahead forecast using the test set.

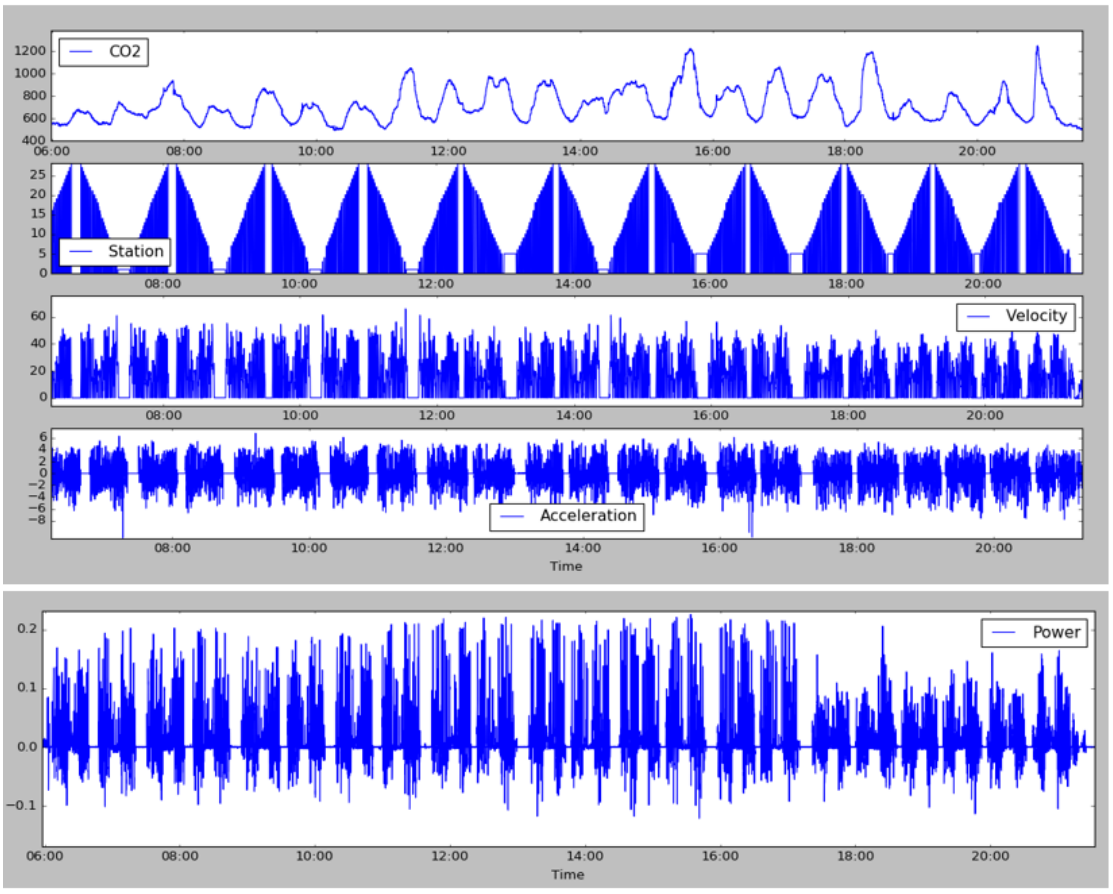


Figure 20: Plots of parameters CO2, Station, Velocity, Acceleration and Power.

In order to tune the network, throughout each experiment we changed a specific parameter of the network and estimated the forecast error. To evaluate these errors, we utilized the root-mean-squared-error metric (RMSE). Below we will present the results that have been derived from our network's parameter analysis.

In the beginning, we investigated the number of epochs. The number of epochs determine the maximum number of passes over the training dataset. Unless we set a maximum number of iterations, the weights will never stop updating. Figure 21 depicts RMSE against the number of epochs. As we can see, RMSE starts to get small values after 100 epochs, shows an increment at 150 and 200 epochs and finally converges after 250 epochs. Thus, we set the number of epochs to 250 for the rest of the MLP experiments. While we ran the experiment to determine the number of epochs, number of neurons, batch size and window size was set to 10, 50 and 1, respectively.

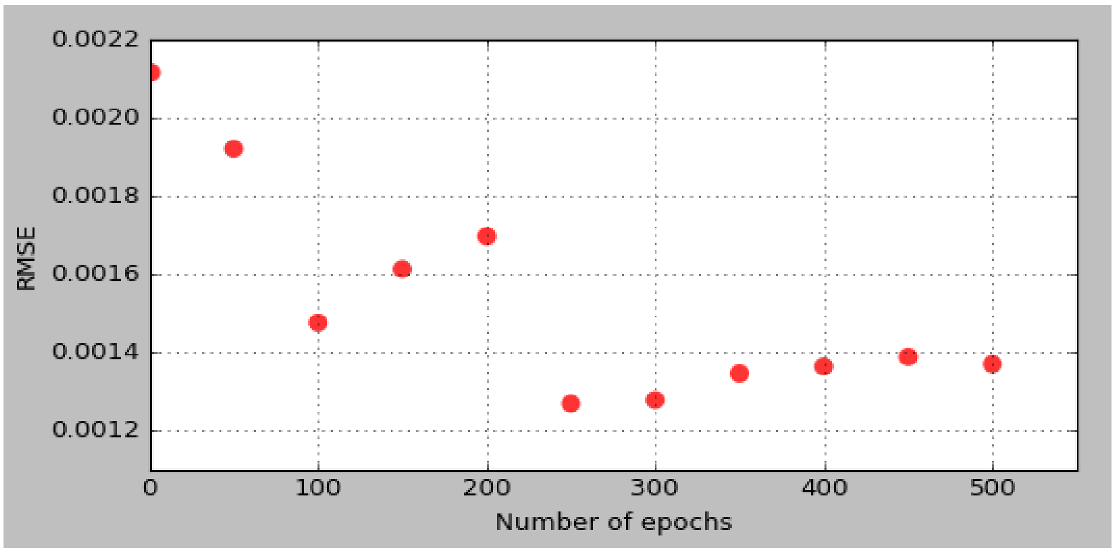


Figure 21: RMSE metric for different number of epochs for the MLP neural network.

Then, the batch size parameter has been investigated. The batch size is the number of training instances used in each iteration. The larger the size of the batch, the faster the training procedure is, because the weights are updated after each batch propagation. The results are plotted in Figure 22. During this experiment, the number of neurons, the number of epochs and the window size was set to 10, 250 and 1, respectively. Based on these results, the batch size has been set to 150 for the rest of the MLP experiments.

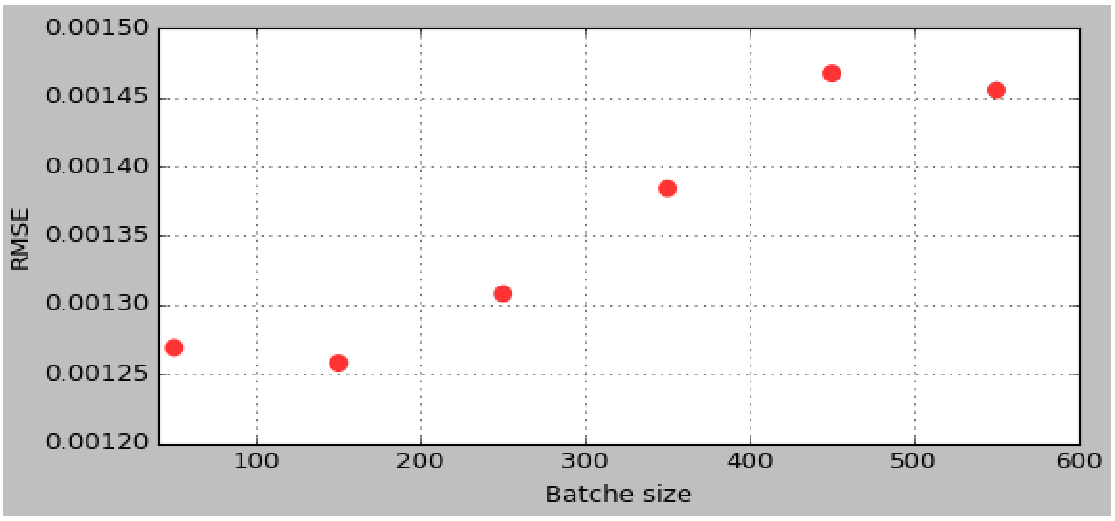


Figure 22: Plot that illustrates the results of RMSE metric for different batch size for the MLP neural network.

Following this, the number of neurons in a network with a single hidden layer and the number of hidden layers is evaluated for number of epochs, batch and window size equal to 250, 150 and 1, respectively. The results are plotted in Figure 23 and Figure 24 and summarized in following tables. It is concluded that the network architecture with one hidden layer and twenty neurons succeed the most accurate power consumption forecasting.

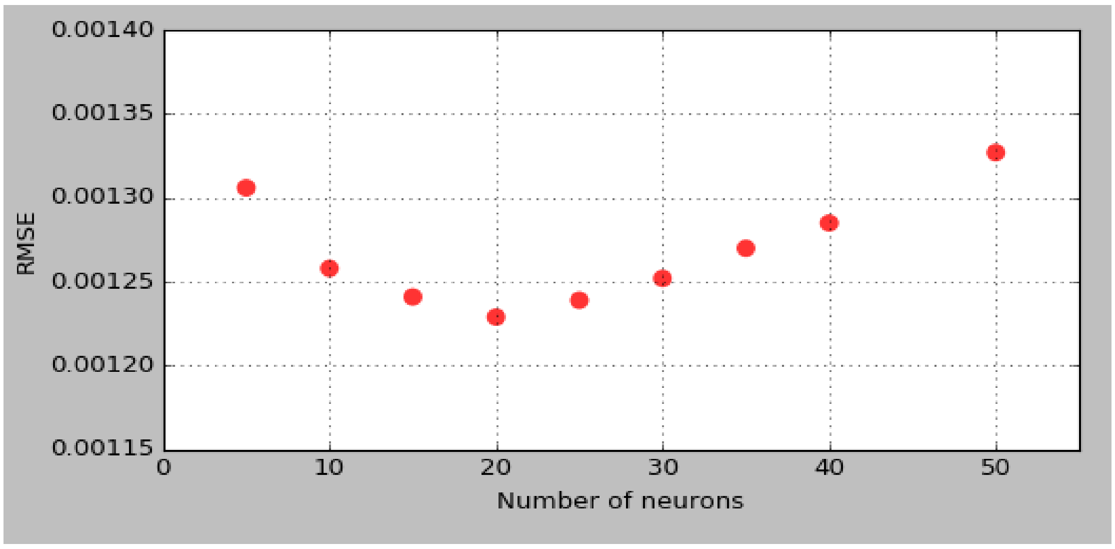


Figure 23: Plot that illustrates the results of RMSE metric for different number of neurons for the MLP neural network with a single hidden layer.

Table 1: Results of RMSE metric for different number of neurons for the MLP neural network with a single hidden layer.

Number of neurons	Number of epochs	Batch size	Window Size	RMSE
5	250	150	1	0.001306
10	250	150	1	0.001258
15	250	150	1	0.001241
20	250	150	1	0.001229
25	250	150	1	0.001239
30	250	150	1	0.001252
35	250	150	1	0.001270
40	250	150	1	0.001285
50	250	150	1	0.001327

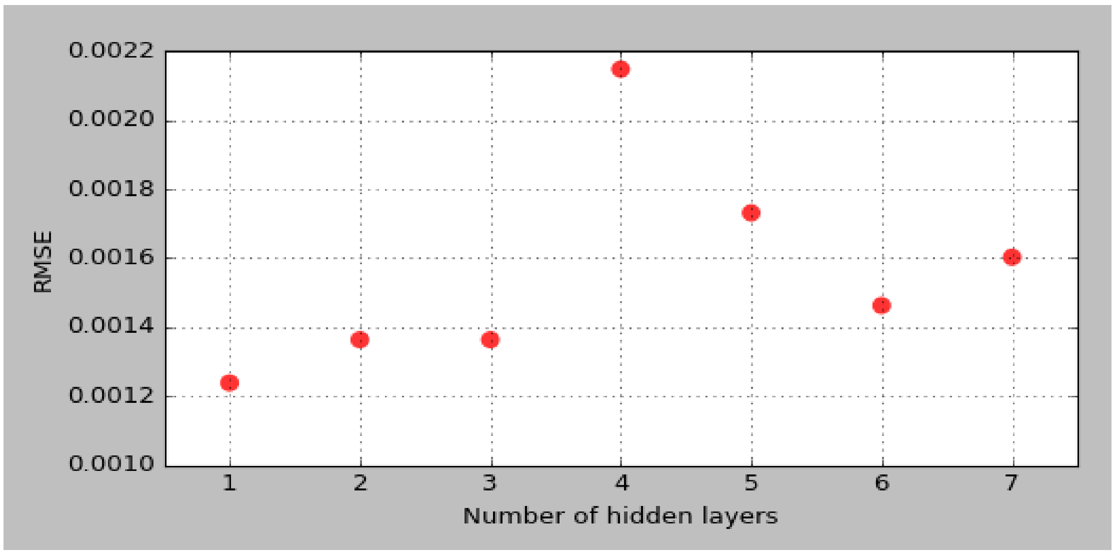


Figure 24: Plot that illustrates the RMSE versus the number of hidden layers for the MLP neural network.

Table 2: Results of RMSE metric for different hidden layers for the MLP neural network.

Number of hidden layers	Number of epochs	Batch size	Window Size	RMSE
1 hidden layer 20 neurons	250	150	1	0.001229
2 hidden layers 25-50 neurons	250	150	1	0.001364
3 hidden layers 10-20-30 neurons	250	150	1	0.001414
4 hidden layers 10-20-30-40 neurons	250	150	1	0.002147
5 hidden layers 40-50-60-70-80 neurons	250	150	1	0.001731
6 hidden layers 40-50-60-70-80-90 neurons	250	150	1	0.001463
7 hidden layers 40-50-60-70-80-90-100 neurons	250	150	1	0.001603

Finally, the window size has been evaluated. On a time series forecasting problem, the window size corresponds to the number of previous time steps that are used as input to the network. The results of the experiments are summarized in the following Table, and plotted in Figure 25. Smaller RMSE derived from the previous time step, while the error is significantly increased for larger windows.

Table 3: Comparative 0predictive performance for different values of window size. The MLP network was composed of 1 hidden layer with 20 neurons, the number of epochs was 250 and the batch size was 150.

Number of neurons	Number of epochs	Batch size	Window Size	RMSE
20	250	150	1	0.001229
20	250	150	2	0.013037
20	250	150	3	0.017000
20	250	150	4	0.020464
20	250	150	5	0.022947

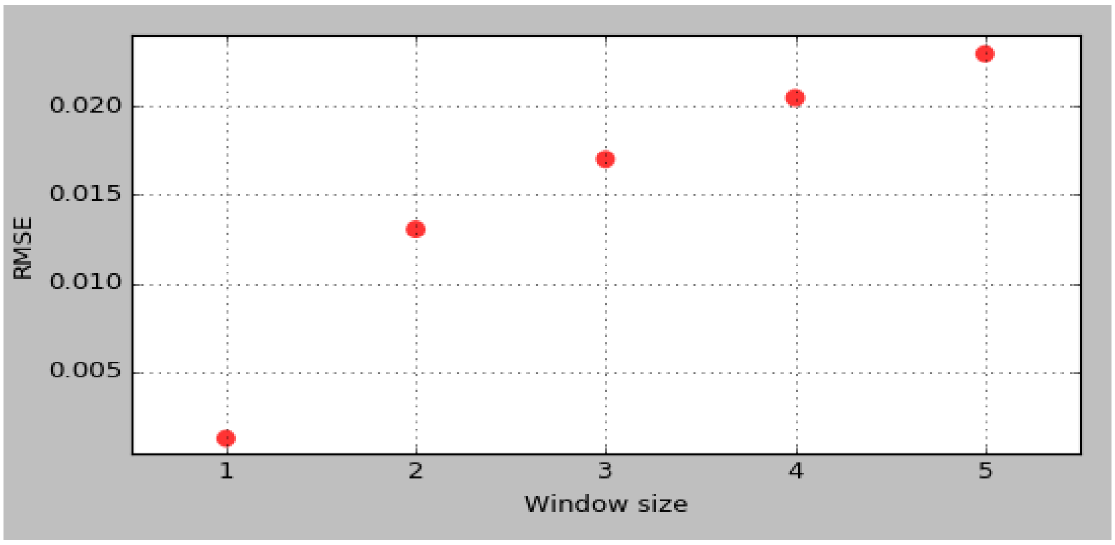


Figure 25: Results for different values of window size. The MLP network was composed of 1 hidden layer with 20 neurons, the number of epochs was 250 and the batch size was 150.

Once we have preprocessed the data and tuned the network with the appropriate network's parameters, a feature selection has been made. Feature selection is important for selecting a subset of features that are most relevant to the problem and as a result to improve computational efficiency or reduce the generalization error of the model by removing irrelevant features or noise. Moreover, selecting a subset of features results to dimensionality reduction and thus to overfitting prevention [12]. In previous experiments, we used only the power feature as input to the MLP neural network. Afterwards, we experimented with different subsets of features that we could use as input to the neural network to find the one that will lead to better performance. The subset of features that ended up with better results was the [longitude, latitude, velocity, acceleration] with RMSE equal to 0.017508.

Based on the above analysis, we deduce that a MLP neural network model cannot succeed adequate predictive performance on power forecasting. The MLP model achieved to forecast the power consumption of the train with RMSE 0.001229 using as input the observation of previous time step of the power. Finally, our MLP model was able to forecast the power consumption of the train with RMSE 0.017508 using as input the observations of previous and current time step of the features longitude, latitude, velocity and acceleration. Given the fact that the mean value of the power is equal to 0.006411, we can evaluate that the total error of the forecasting is 19.17% at first case, which is considered a non-acceptable predicted error and 273% at second case, which is a very high predicted error.

6.3 Results of LSTM implementation

In this section, we will present the experimental results of the LSTM implementation on the train dataset. We used LSTM neural networks for the power consumption forecasting. Two different LSTM neural network models were developed and compared, a stateless and a stateful LSTM. The main difference between the stateless and stateful LSTM is that the latter uses as an initial state for each sample at a specific position in a batch, the last state of the sample at the same position in the previous batch.

Like the MLP implementation, before feeding the LSTM model with the dataset, we preprocessed it properly. At first step, we split the dataset into training set and test set, so that the training set comprises the 65% and 70% of the total dataset while the remaining 35% and 30% constitutes the test set for the stateless and the stateful LSTM respectively. At second step, we transformed time series data in order to be stationary. Then we transformed the time series problem into a supervised problem, using the observations of a sequence of time steps as the model's input and the observation at the current time step as model's output. Finally, we rescaled the data to a range of [-1,1]. In the following experiments, we calculated one-step ahead forecast using the test set.

Once we had preprocessed the data, we investigated the network's parameter for each LSTM model.

The first parameter that was studied was the number of neurons. In Figure 26, we illustrate our results for the stateless LSTM and in Figure 27 for the stateful LSTM.

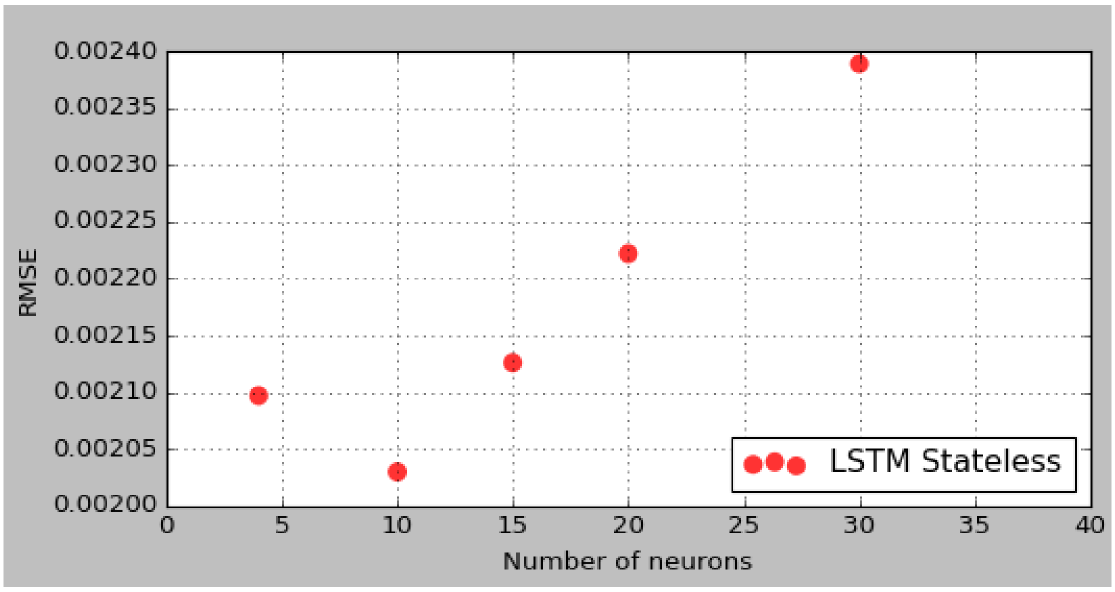


Figure 26: Results for different values of neurons for the stateless LSTM neural network. The LSTM was composed of 1 hidden layer, the number of epochs was equal to 1, the batch size equal to 50 and the sequence length equal to 50.

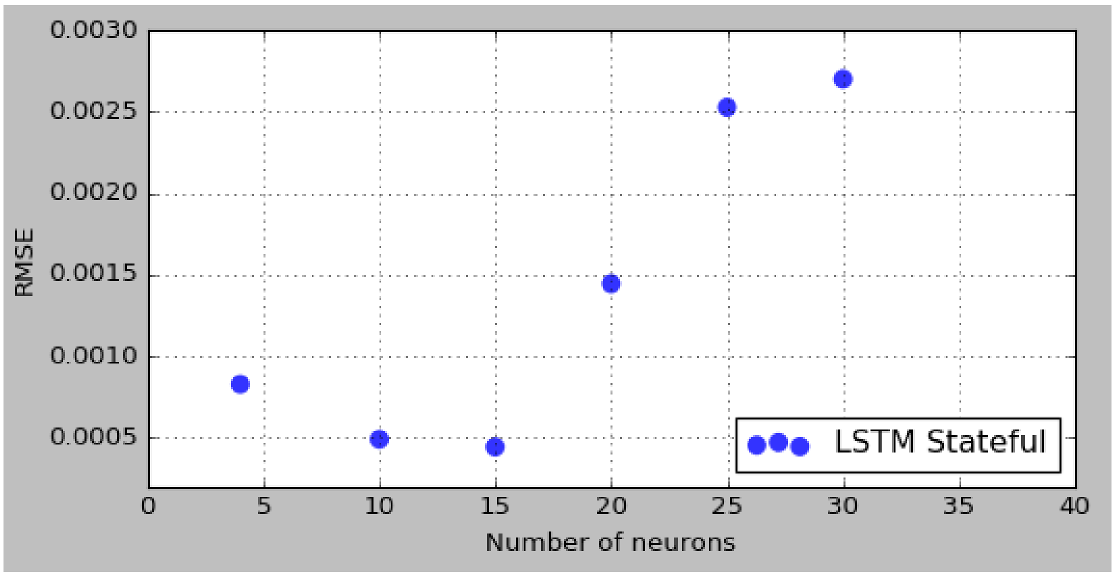


Figure 27: Results for different values of neurons for the stateful LSTM neural network. The LSTM was composed of 1 hidden layer, the number of epochs was equal to 1, the batch size equal to 50 and the sequence length equal to 50.

During this experiment, we used one hidden layer, one epoch, while batch size and sequence length were set to 50. According to the plots, we set the number of neurons to 10 for the stateless LSTM and to 15 for the stateful LSTM.

The next parameter that was researched was the length of the sequence. As the length of the sequence increases, the available learning information for the neural network increases too. Feeding the LSTM network with sequences results in teaching itself how to construct pattern of the sequences based on the previous window received. Sequences are sliding windows and shift by one at a time, causing a constant overlap of the prior windows. We used only one epoch, in which the LSTM cycled through all the sequence windows in the training set once. The results of this experiment are illustrated in Figure 28 for the stateless LSTM and in Figure 29 for the stateful LSTM.

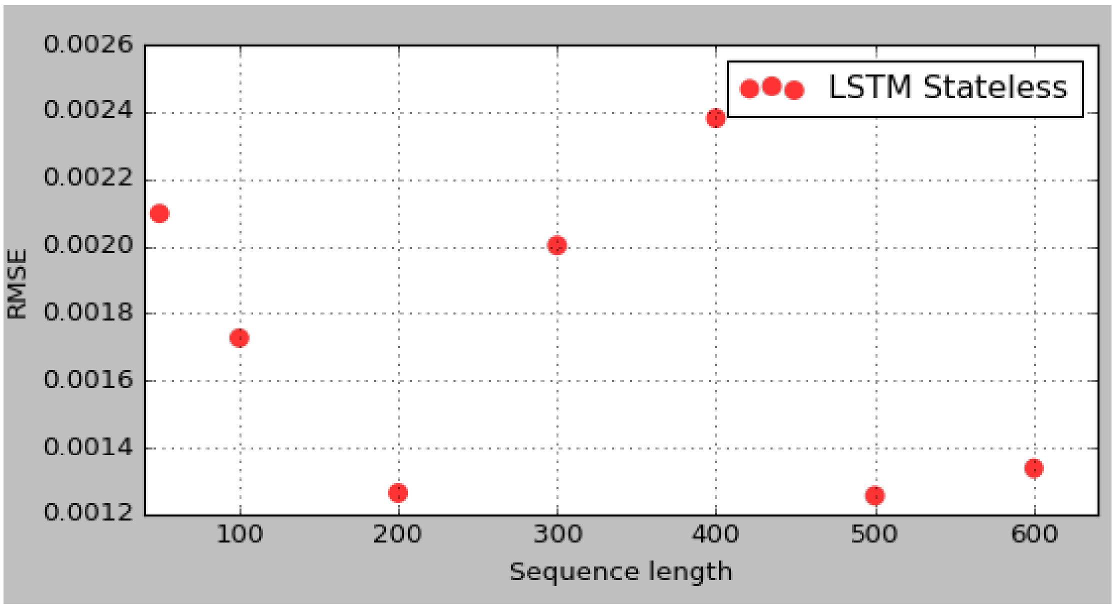


Figure 28: Results for different lengths of sequence for the stateless LSTM neural network. The LSTM was composed of 1 hidden layer with 10 neurons, the number of epochs was equal to 1 and the batch size equal to 50.

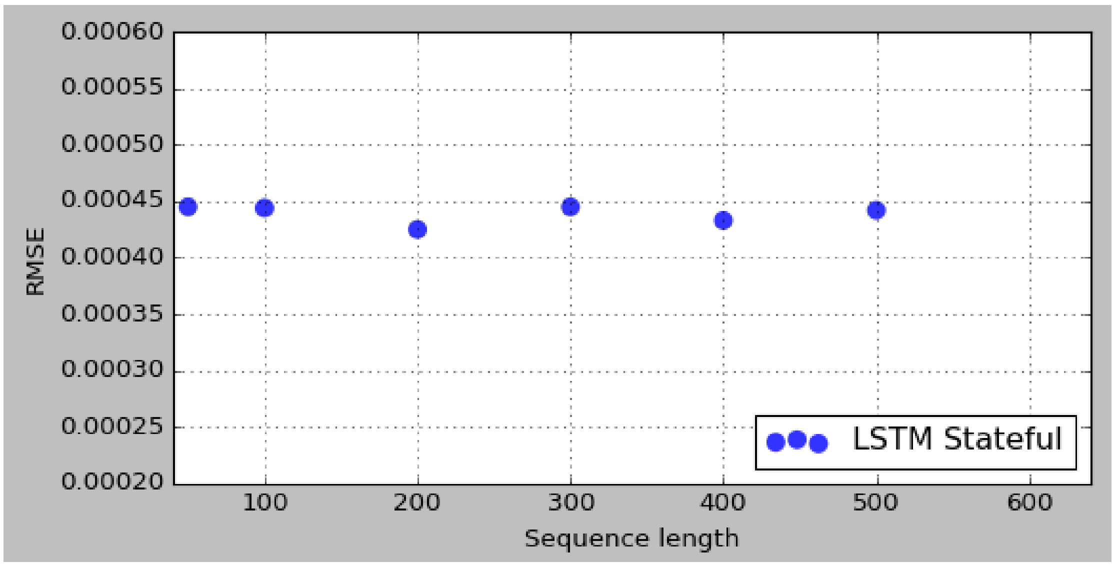


Figure 29: Results for different lengths of sequence for the stateful LSTM neural network. The LSTM was composed of 1 hidden layer with 15 neurons, the number of epochs was equal to 1 and the batch size equal to 50.

According to the plots above, we set the parameter “sequence length” to 500 for stateless LSTM and to 200 for stateful LSTM.

The last parameter that was investigated was the number of hidden layers. We register the results in the following tables and illustrate the RMSE versus the number of hidden layers in Figure 30, Figure 31 for the stateless and the stateful LSTM respectively. From the presenting tables and figures we conclude that the best architecture for stateless LSTM is the one with four hidden layers, each of which composed of ten neurons and for stateful LSTM is the one with ten hidden layers each of which composed of four neurons.

Table 4: Results of RMSE metric for different hidden layers for the stateless LSTM neural network.

Number of hidden layers	Sequence length	Batch size	Number of epochs	RMSE
1 hidden layer 10 neurons	500	50	1	0.001256
2 hidden layers 10 neurons each	500	50	1	0.000528
3 hidden layers 10 neurons each	500	50	1	0.000616
4 hidden layers 10 neurons each	500	50	1	0.000123
5 hidden layers 10 neurons each	500	50	1	0.000469
6 hidden layers 10 neurons each	500	50	1	0.000544

Table 5: Results of RMSE metric for different hidden layers for the stateful LSTM neural network.

Number of hidden layers	Sequence length	Batch size	Number of epochs	RMSE
1 hidden layer 4 neurons	500	50	1	0.000425
2 hidden layers 4 neurons each	500	50	1	0.000227
4 hidden layers 4 neurons each	500	50	1	0.000148
5 hidden layers 4 neurons each	500	50	1	0.000127
6 hidden layers 4 neurons each	500	50	1	0.000053

7 hidden layers **4 neurons each**	500	50	1	0.000048
8 hidden layers **4 neurons each**	500	50	1	0.000048
9 hidden layers **4 neurons each**	500	50	1	0.000036
10 hidden layers **4 neurons each**	500	50	1	0.000017
15 hidden layers **4 neurons each**	500	50	1	0.000111

s

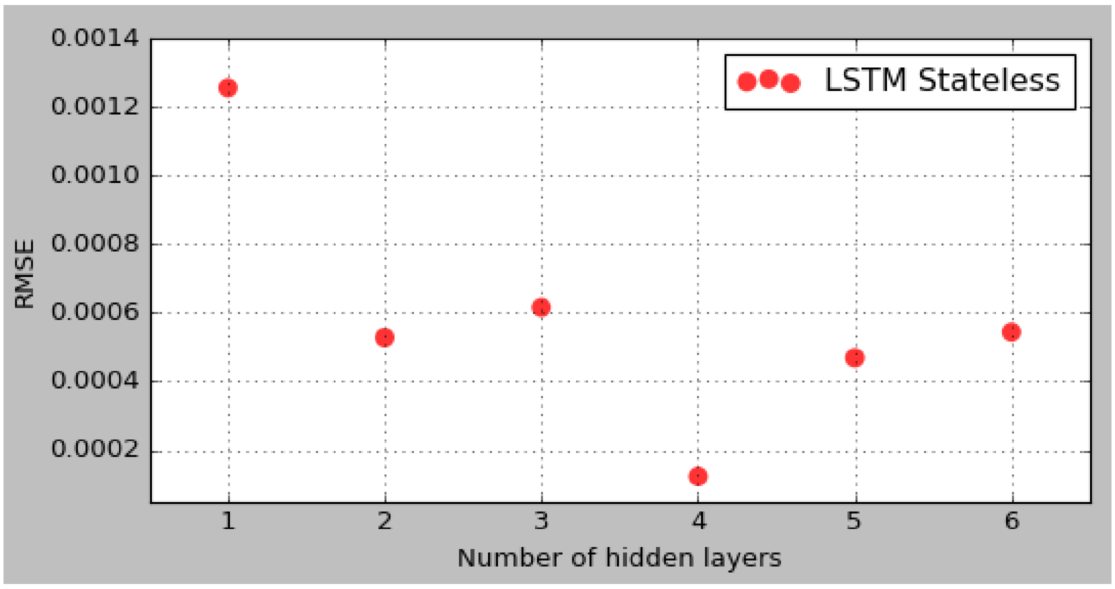


Figure 30: Plot that illustrates the RMSE versus the number of hidden layers for the stateless LSTM neural network.

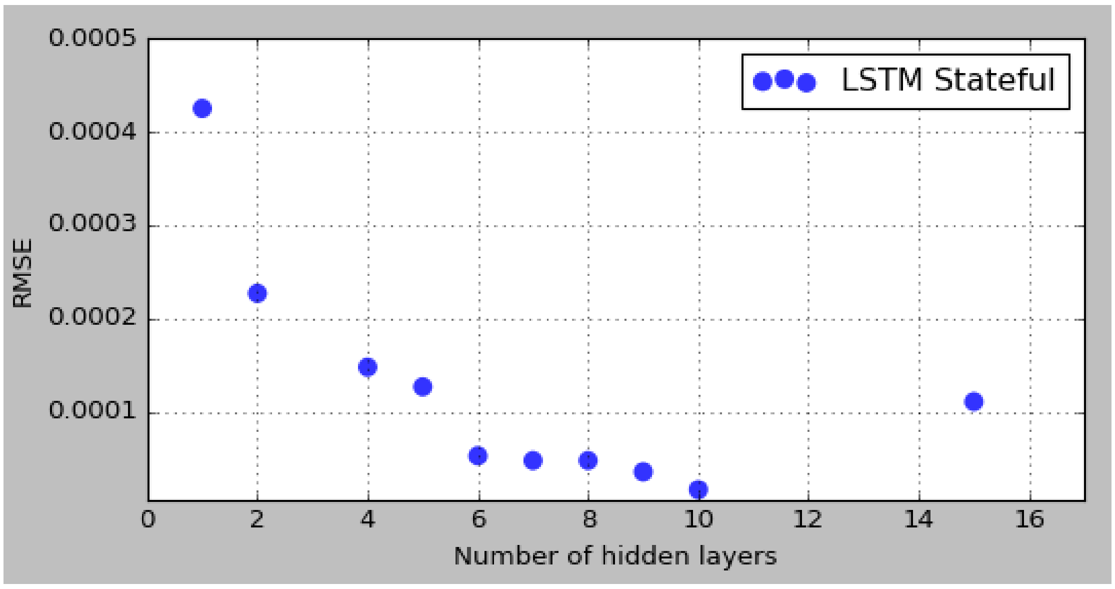


Figure 31: Plot that illustrates the RMSE versus the number of hidden layers for the stateful LSTM neural network.

In previous experiments, we used only the sequences of the power feature as input to the stateless and stateful LSTM model to predict the next time step power forecasting. Afterward, we experimented with different subsets of features that we could use as input to the neural network to identify the one that would lead to power forecasting with acceptable RMSE. So, we fed the stateless LSTM with different subset of features and estimated the RMSE for each of them. The results are presented in the following table and in the Figure 32.

Table 6: RMSE values for different subset of features that used as input to the stateless LSTM neural network.

Subset of features	Number of hidden layers	Sequence length	Batch size	Number of epochs	RMSE
[Longitude, Latitude, CO2, Velocity, Acceleration]	4 hidden layers 4 neurons each	1	50	100	0.00304
[Longitude, Latitude, Velocity, Acceleration]	4 hidden layers 4 neurons each	1	50	100	0.00294
[Longitude, Latitude, Acceleration]	4 hidden layers 4 neurons each	1	50	100	0.00278
[Velocity, CO2, Acceleration]	4 hidden layers 4 neurons each	1	50	100	0.00223
[Velocity, Acceleration]	4 hidden layers 4 neurons each	1	50	100	0.00230
[Velocity]	4 hidden layers 4 neurons each	1	50	100	0.00226
[Acceleration]	4 hidden layers 4 neurons each	1	50	100	0.00027

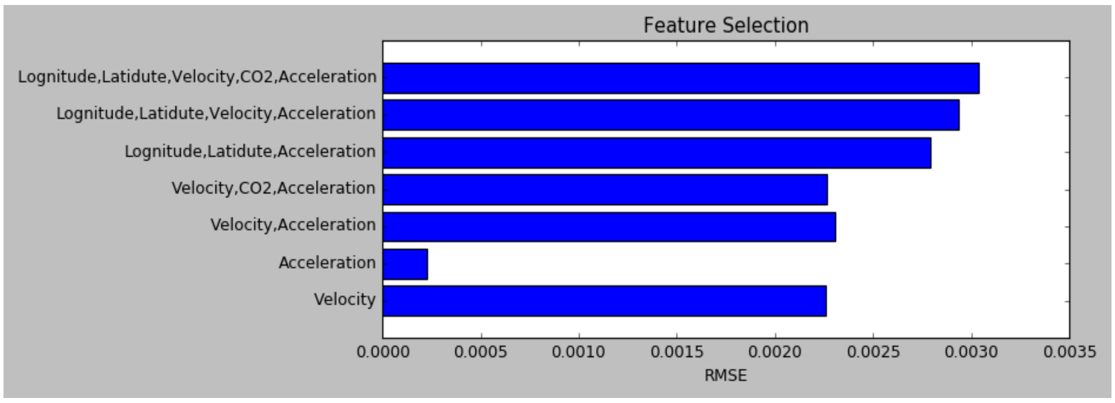


Figure 32: Plot of RMSE values for different subset of features that used as input to the stateless LSTM neural network.

In this multivariate continuous time series problem, we chose to go with 100 epochs and a length of sequence equal to 1. If those variables are highly correlated and there are no long-term dependencies, it is preferred to use many hidden units and smaller sequence length. More hidden units help to capture the correlation between features and smaller sequence length allowing the back-propagation algorithm to be faster, as the gradient is calculated at a smaller number of time steps.

Taking into consideration the results presented in Table 6, we concluded that acceleration is the feature that provided improved results with RMSE equal to 0.00027. To compare the performance of two LSTM models, we fitted the stateful LSTM to the acceleration feature too. Stateful LSTM achieved to predict the power consumption with RMSE equal to 0.000065.

In summary, the LSTM recurrent neural network can succeed sufficient predictive performance on the power forecasting. Stateless LSTM achieved to forecast the power consumption of the train with RMSE equal to 0.000123, which means that the error was equal to 1.69%, when accepted a sequence of previous time steps of the power as input. It also predicted the power with RMSE equal to 0.000227, which means that the error was equal to 3.13%, when accepted the observation of previous and current time step of the acceleration as input. Finally, stateful LSTM achieved to forecast the power of the train with RMSE equal to 0.000017 (error equal to 0.25%) when accepted a sequence of previous time steps of the power as input. It also predicted the power with RMSE equal to 0.000065 (error equal to 0.95%) when accepted the observation of previous and current time step of the acceleration as input. According to the last results, we concluded that the performance of the LSTMs is acceptable.

6.4 Discussion

In this section, we present a comparison of MLP and LSTM on power forecasting. Two variants of LSTM were considered, the stateless and the stateful LSTM. The experiments on our real train data show that LSTM is much better than MLP at predicting the power consumption of the train. Moreover, we found that the stateful LSTM neural network outperformed the stateless LSTM.

In Figure 33, a histogram of RMSE for power consumption forecasting is presented. For MLP the input was the exact previous value of power while for the LSTMs the input was a sequence of previous power values. The same histogram is presented in Figure 34. Although in this case the input for MLP was the exact previous and the current time step of features [longitude, latitude, velocity, acceleration] while for the LSTMs was the acceleration.

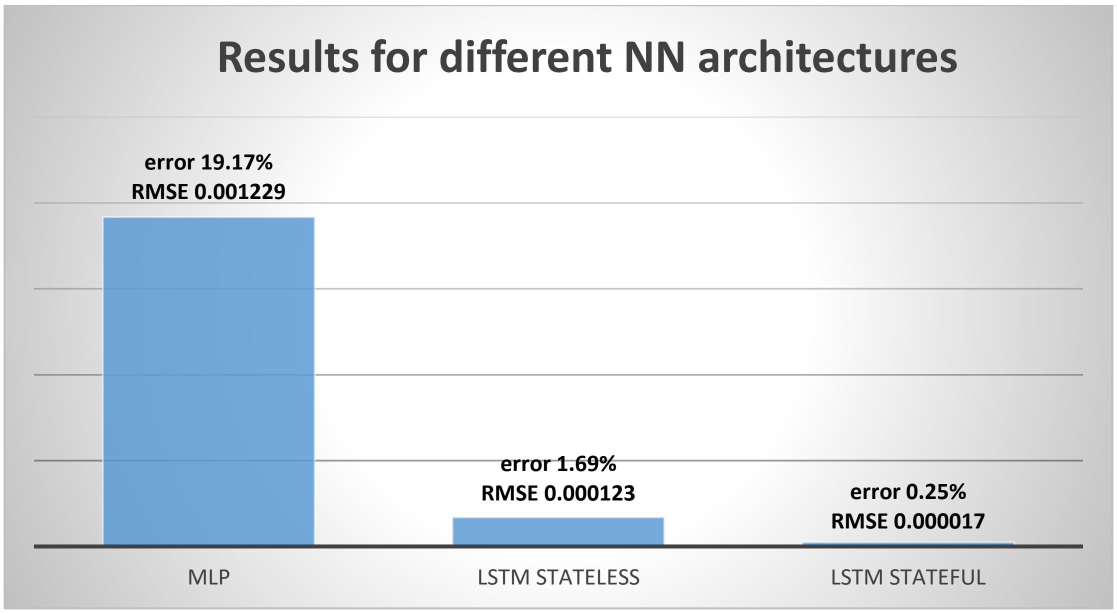


Figure 33: Histogram plot of RMSE for MLP, stateless LSTM and stateful LSTM (input power). LSTM models outperformed MLP model and stateful LSTM outperformed stateless LSTM in terms of error performance.

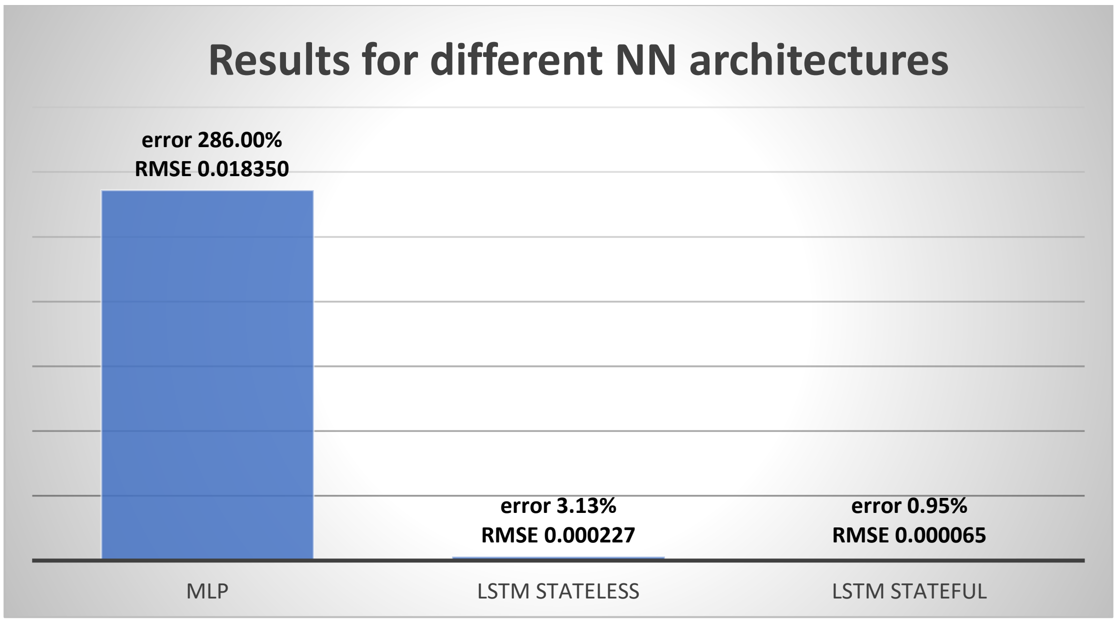


Figure 34:Histogram plot of RMSE for MLP, stateless LSTM and stateful LSTM. (input subset of features [longitude, latitude, velocity, acceleration] for MLP and feature acceleration for LSTMs). LSTM models outperformed MLP model and stateful LSTM outperformed stateless LSTM in terms of error performance.

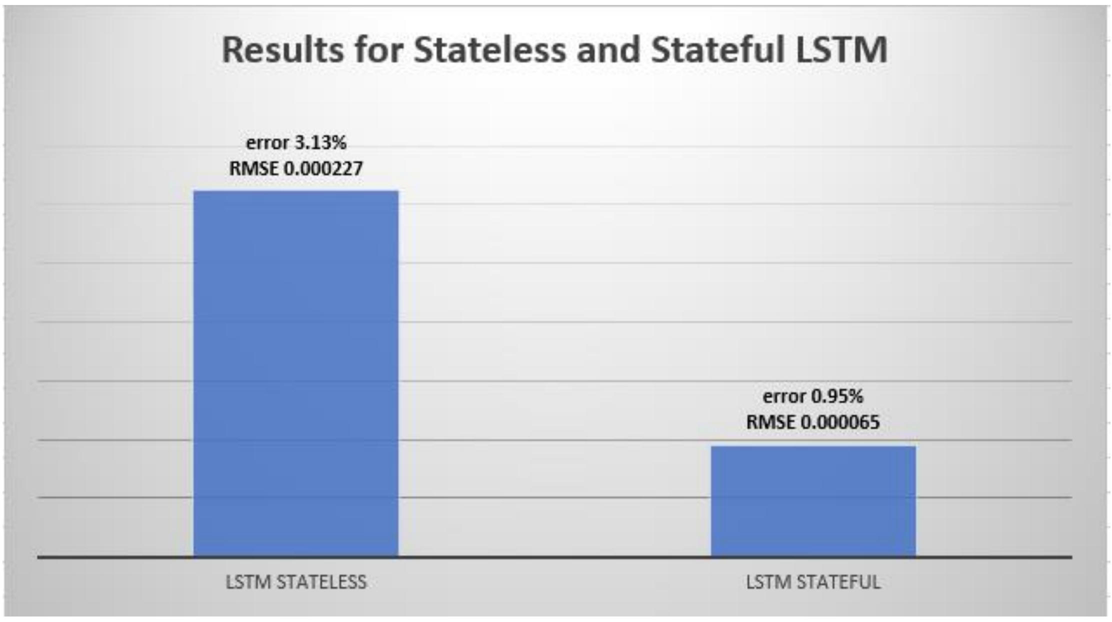


Figure 35: Histogram plot of RMSE for stateless LSTM and stateful LSTM. (input acceleration). LSTM stateful outperformed LSTM stateless model.

Chapter 7 Summary

This master thesis, introduced the concepts of Cloud Computing and Internet of Things and the basic notions related Big Data, Data Mining and Machine Learning. The MapReduce model and other tools capable to handle big data, such as Apache Hadoop, Pig, Hive, HBase, and Mahout are also described. Then SiteWhere, an IoT solution used to manage and analyze device data. SiteWhere was was used to manage and analyze device data collected from sensors supporting a railway use case. The theoretical background of Neural Networks was provided, while the processing of the dataset was composed of various quantities related with the railway use case was described.

The first step of the processing involved storage of the train dataset in MongoDB, a database supported by SiteWhere. After data were stored, they were visualized using a GUI, the MongoDB Compass.

The second step was the retrieval of the data from MongoDB in order to feed different types of Neural Networks, the MLP and LSTM, and forecast the train power consumption. We experimented with NNs by giving them different quantities as input. In the beginning, we fed the MLP and LSTMs with the exact previous time step of the power parameter. The errors derived from this experiment were 19.17% for the MLP, 1.69% for the LSTM Stateless and 0.25% for the LSTM Stateful Neural Network. Then, we fed the NNs with various combinations of the quantities longitude, latitude velocity, CO2 and acceleration. MLP optimal results were derived when the subset [longitude, latitude, velocity, acceleration] was given as input. In this case, the error was equal to 286% which is a very high and insufficient error. LSTMs best results were obtained when the acceleration parameter was given as input with errors 3.13% for the Stateless and 0.95% for the Stateful LSTM. In experiments that were presented above, the output of the NNs was the value of the power at current time step. According to previous errors, we concluded that LSTM outperformed the MLP in that time series forecasting problem in accordance to the theoretical analysis performed.

References

[1] [Online]. Available: https://en.wikipedia.org/wiki/Big_data.

[2] J. Z. Mohammed and M. J. Wagner, "Data Mining and Analysis: Fundamental Concepts and Algorithms," Cambridge University Press, 2014.

[3] S. Theodoridis, Machine Learning A Bayesian and Optimization Perspective, Elsevier, 2015.

[4] [Online]. Available: https://en.wikipedia.org/wiki/Forecasting.

[5] J. Leskovec, A. Rajaraman and J. D. Ullman, Mining of Massive Datasets, 2014.

[6] S. Raschka, Python Machine Learning, 35 Livery Street, Birmingham B3 2PB, UK: Packt Publishing Ltd, 2016.

[7] E. Alpaydın, Introduction to Machine Learning, Massachusetts Institute of Technology, 2010.

[8] EMC Education Services, Data Science & Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data, Indianapolis, Indiana: John Wiley &Sons, 2015.

[9] [Online]. Available: http://hadoop.apache.org/.

[10] J. Venner, Pro Hadoop, Apress, 2009.

[11] Q. Zhang, L. Cheng and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," J Internet Serv Appl (2010) 1: 7–18, 2010.

[12] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST Special Publication 800-145, 2011.

[13] [Online]. Available: https://en.wikipedia.org/wiki/Cloud_computing.

[14] L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A survey," Elsevier, 2010.

[15] A. Bassi and G. Horn, "Internet of Things in 2020: A Roadmap for the Future," European Commission: Information Society and Media, 2008.

[16] A. Botta, . W. d. Donato, V. Persico and A. Pescapé, "Integration of Cloud computing and Internet of Things: A survey," Elsevier, 2015.

[17] [Online]. Available: http://www.sitewhere.org.

[18] S. Haykin, Neural Networks and Learning Machines, Upper Saddle River, New Jersey 07458: Pearson Education, 2009.

[19] A. Graves, "Supervised Sequence Labelling with Recurrent Neural Networks".

[20] P. J. Brockwell and R. A. Davis, Introduction to Time Series and Forecasting, New York: Springer-Verlag New York, 2002.

[21] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Google, Inc..

[22] H. Goel, I. Melnyk, N. Oza, B. Matthews and A. Banerjee, "Multivariate Aviation Time Series Modeling: VARs vd. LSTMs".

[23] W. Fan and A. Bifet, "Mining Big Data: Current Status, and Forecast to the Future".

[24] G. Bontempi, S. B. Taieb and Y.-A. L. Borgne, "Machine Learning Strategies for Time Series Forecasting".

[25] H. Lütkepohl, New Introduction to Multiple Time Series Analysis, Berlin: Springer-Verlag, 2005.

[26] J. Hurwitz, A. Nugent, F. Dr. Halper and M. Kaufman, Big Data for Dummies, Hoboken, New Jersey: John Wiley & Sons, 2013.

[27] [Online]. Available: http://www.jakob-aungiers.com/articles/a/LSTM-Neural-Network-for-Time-Series-Prediction.

[28] [Online]. Available: http://machinelearningmastery.com/time-series-forecasting/.

[29] [Online]. Available: http://machinelearningmastery.com/understanding-stateful-lstm-recurrent-neural-networks-python-keras/.

[30] J. Guth, U. Breitenbücher, M. Falkenthal, F. Leymann and L. Reinfurt, "Comparison of IoT Platform Architectures: A Field Study based on a Reference Architecture," IEEE, 2016.

[31] J. Mineraud , O. Mazhelis, X. Su and S. Tarkoma, "A gap analysis of Internet-of-Things platforms".

[32] [Online]. Available: https://www.mongodb.com.

[33] [Online]. Available: http://cs231n.github.io/neural-networks-1/.

[34] [Online]. Available: https://en.wikipedia.org/wiki/Stochastic_gradient_descent.

[35] [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.