



SELENE LEYA CERNA ÑAHUIS

**Comparative Analysis of XGBoost, MLP and LSTM
Techniques for The Problem of Predicting Fire Brigade
Interventions**

**Análise Comparativa das Técnicas XGBoost, MLP e LSTM para o
Problema de Prever Intervenções de Bombeiros**

SELENE LEYA CERNA ÑAHUIS

**Comparative Analysis of XGBoost, MLP and LSTM
Techniques for The Problem of Predicting Fire Brigade
Interventions**

**Análise Comparativa das Técnicas XGBoost, MLP e LSTM para o
Problema de Prever Intervenções de Bombeiros**

Dissertation presented to the São Paulo State University (UNESP) - School of Engineering - Campus of Ilha Solteira, in fulfilment of one of the requirements for obtaining the Master's degree in Electrical Engineering.

Field of Expertise: Automation.

Prof. Dr. Anna Diva Plasencia Lotufo

Advisor

Prof. PhD. Christophe Guyeux

Co-Advisor

Ilha Solteira

2019

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

C415c Cerna Ñahuis, Selene Leya.
Comparative analysis of XGBoost, MLP and LSTM techniques for the problem
of predicting fire brigade interventions / Selene Leya Cerna Ñahuis. -- Ilha
Solteira: [s.n.], 2019
76 f. : il.

Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de
Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2019

Orientador: Anna Diva Plasencia Lotufo
Coorientador: Christophe Guyeux
Inclui bibliografia

1. Firefighters. 2. Prediction. 3. XGBoost. 4. Long short-term memory. 5.
Multi-layer perceptron. 6. Mutual information regression.


Raiane da Silva Santos

Supervisora Técnica de Seção
Seção Técnica de Referência, Atendimento ao usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação
CRB/8 - 9999



UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: COMPARATIVE ANALYSIS OF XGBOOST, MLP AND LSTM TECHNIQUES
FOR THE PROBLEM OF PREDICTING FIRE BRIGADE INTERVENTIONS

AUTORA: SELENE LEYA CERNA ÑAHUIS

ORIENTADORA: ANNA DIVA PLASENCIA LOTUFO

COORIENTADOR: CHRISTOPHE GUYEUX

Aprovada como parte das exigências para obtenção do Título de Mestra em ENGENHARIA
ELÉTRICA, área: Automação pela Comissão Examinadora:

Prof. Dr. CHRISTOPHE GUYEUX
Université Franche Comté

Mauro de Souza Tonelli Neto

Pós-doutorando **MAURO DE SOUZA TONELLI NETO**

Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Kenji Nose Filho

Prof. Dr. **KENJI NOSE FILHO**

Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas / Universidade Federal do ABC

Ilha Solteira, 30 de agosto de 2019

A mi Oshito bonita, por su amor y paciencia.

ACKNOWLEDGEMENTS

I would like to express my thanks to my advisor, Anna Diva, for accepting me in the master and giving me the chance to meet this beautiful country, Brazil. For guiding me and encouraging me throughout my research with numerous advice and for the friendship and patience show up these years.

I thank my co-advisor Christophe Guyeux, for the trust placed in me and for giving me the precious opportunity to contribute to firefighters during this research. For being an example to me as a researcher with passion and commitment, for sharing his knowledge and teaching me with patience.

I am also grateful to Pierre-Cyrille Heam and Raphaël Couturier, professors in The University of Franche-Comté, for their welcome and support in the AND team, Femto-ST. And a special thanks to Cap. Guillaume Royer from SDIS25, for giving me the honour of work with such a great organization, the fire brigade of Doubs, whom I admire for their noble effort and dedication to the community good.

I would like to infinitely thank my parents, Eulalia and Fernando, who never measured efforts to give studies to their children and always encouraged us to be persevering and to strive for what we want. To my brother Edwin, for all his effort working and helping the family since his childhood. To my sister Liesel, for taking care of me and giving me a profession. And to my brother Fernando, for sharing his knowledge with me since I was a child and for showing me the scientific way.

I thank Héber, my special person, who patiently guided me and cheered me in the writing of my scientific texts, for accompanying me in long discussions about research and incoherences, with whom I discovered Brazil.

I thank my good friends Tania and Monara, who welcomed me affectionately in SINTEL, they understood my zero or poorly spoken Portuguese and helped me so kindly during the master's degree.

Finally, this study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001".

ABSTRACT

Many environmental, economic and societal factors are leading fire brigades to be increasingly solicited, and, as a result, they face an ever-increasing number of interventions, most of the time on constant resource. On the other hand, these interventions are directly related to human activity, which itself is predictable: swimming pool drownings occur in summer while road accidents due to ice storms occur in winter. One solution to improve the response of firefighters on constant resource is therefore to predict their workload, i.e., their number of interventions per hour, based on explanatory variables conditioning human activity. The present work aims to develop three models that are compared to determine if they can predict the firefighters' response load in a reasonable way. The tools chosen are the most representative from their respective categories in Machine Learning, such as XGBoost having as core a decision tree, a classic method such as Multi-Layer Perceptron and a more advanced algorithm like Long Short-Term Memory both with neurons as a base. The entire process is detailed, from data collection to obtaining the predictions. The results obtained prove a reasonable quality prediction that can be improved by data science techniques such as feature selection and adjustment of hyperparameters.

Keywords: Firefighters. Prediction. XGBoost. Long Short-Term Memory. Multi-Layer Perceptron. Mutual Information Regression. Principal Component Analysis.

RESUMO

Muitos fatores ambientais, econômicos e sociais estão levando as brigadas de incêndio a serem cada vez mais solicitadas e, como consequência, enfrentam um número cada vez maior de intervenções, na maioria das vezes com recursos constantes. Por outro lado, essas intervenções estão diretamente relacionadas à atividade humana, o que é previsível: os afogamentos em piscina ocorrem no verão, enquanto os acidentes de trânsito, devido a tempestades de gelo, ocorrem no inverno. Uma solução para melhorar a resposta dos bombeiros com recursos constantes é prever sua carga de trabalho, isto é, seu número de intervenções por hora, com base em variáveis explicativas que condicionam a atividade humana. O presente trabalho visa desenvolver três modelos que são comparados para determinar se eles podem prever a carga de respostas dos bombeiros de uma maneira razoável. As ferramentas escolhidas são as mais representativas de suas respectivas categorias em Machine Learning, como o XGBoost que tem como núcleo uma árvore de decisão, um método clássico como o Multi-Layer Perceptron e um algoritmo mais avançado como Long Short-Term Memory ambos com neurônios como base. Todo o processo é detalhado, desde a coleta de dados até a obtenção de previsões. Os resultados obtidos demonstram uma previsão de qualidade razoável que pode ser melhorada por técnicas de ciência de dados, como seleção de características e ajuste de hiperparâmetros.

Palavras-chave: Bombeiros. Previsão. XGBoost. Long Short-Term Memory. Multi-Layer Perceptron. Mutual Information Regression. Principal Component Analysis.

LIST OF FIGURES

Figure 1	Bias-Variance tradeoff in machine learning	23
Figure 2	Learning a tree on single variable	25
Figure 3	A Multi-Layer Perceptron. The S-shaped curves in the hidden and output layers indicate the application of “sigmoidal” nonlinear activation functions.	27
Figure 4	A recurrent neuron (left), unrolled through time (right).	30
Figure 5	LSTM cell	31
Figure 6	Standard scaler representation	33
Figure 7	One hot encoder representation	34
Figure 8	Procedure for estimating the MI	35
Figure 9	Transformation with PCA	36
Figure 10	Number of interventions per year	41
Figure 11	Frequency of number of interventions per hour	41
Figure 12	Outliers in the occurrence of interventions	41
Figure 13	Maximum number of interventions	42
Figure 14	Precipitation each 1h	43
Figure 15	L’Est Republicain reports on the storms	43
Figure 16	Precipitations peaks recorded from Basilea Station	43
Figure 17	Storm area	44
Figure 18	Precipitation each 3h	44
Figure 19	Ten minutes average wind speed	44
Figure 20	Features structure before modelling	48
Figure 21	XGBoost - Feature importance	54
Figure 22	XGBoost - Tree construction	55

Figure 23	MLP model architecture with 3 past hours	56
Figure 24	LSTM model architecture with 24 time steps	57
Figure 25	XGBoost with three past hours for predicting one hour	58
Figure 26	XGBoost with twelve past hours for predicting one hour	59
Figure 27	XGBoost with twenty-four past hours for predicting one hour	60
Figure 28	MLP with three past hours for predicting one hour	61
Figure 29	MLP with twelve past hours for predicting one hour	62
Figure 30	MLP with twenty-four past hours for predicting one hour	63
Figure 31	LSTM with three past hours for predicting one hour	64
Figure 32	LSTM with twelve past hours for predicting one hour	65
Figure 33	LSTM with twenty-four past hours for predicting one hour	66
Figure 34	XGBoost, MLP and LSTM with three past hours for predicting one hour	67
Figure 35	XGBoost, MLP and LSTM with twelve past hours for predicting one hour	68
Figure 36	XGBoost, MLP and LSTM with twenty-four past hours for predicting one hour	69

LIST OF TABLES

Table 1	Initialization parameters for each type of activation function	29
Table 2	Illustrated example of the dictionary	40
Table 3	Numerical variables	45
Table 4	Categorical variables	46
Table 5	List of features after applying mutual information regression selection .	47
Table 6	Differences between MIR and PCA	47
Table 7	XGBoost hyperparameters settings	49
Table 8	MLP hyperparameters Settings	50
Table 9	LSTM hyperparameters Settings	50
Table 10	XGBoost prediction results for next one hour on test data 2017	52
Table 11	MLP prediction results for next one hour on test data 2017	52
Table 12	LSTM prediction results for next one hour on test data 2017	52

LIST OF ABBREVIATIONS & ACRONYMS

XGBoost	Extreme Gradient Boosting
ANN	Artificial Neural Network
FNN	Feedforward Neural Network
MLP	Multi-Layer Perceptron
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
SVM	Support Vector Machine
GA	Genetic Algorithm
PSO	Particle Swarm Optimization
KDE	Kernel Density Estimation
MIR	Mutual Information Regression
PCA	Principal Component Analysis
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
SDIS25	Service Départemental d'Incendie et de Secours 25
ACC	Accuracy
DBN	Deep Belief Network
RBM	Restricted Boltzmann Machine
CART	Classification and Regression Trees
GBM	Gradient Boosting Machine
MSE	Mean Squared Error
DNN	Deep Neural Network
Tanh	Hyperbolic Tangent function
ReLU	Rectified Linear Unit function
RTRL	Real Time Recurrent Learning
BPTT	Back-Propagation Through Time
OHE	One-Hot Encoding
MI	Mutual Information
SVD	Singular Value Decomposition
ACC0E	Accuracy with margin of error 0
ACC1E	Accuracy with margin of error 1
ACC2E	Accuracy with margin of error 2
GRU	Gated Recurrent Unit

TABLE OF CONTENTS

1	INTRODUCTION	13
1.1	MOTIVATION	15
1.2	CONTRIBUTIONS	15
1.3	OUTLINE OF THE THESIS	16
2	MACHINE LEARNING TECHNIQUES	18
2.1	INTRODUCTION	18
2.2	GRADIENT BOOSTING MACHINES	21
2.2.1	OVERVIEW	21
2.2.2	EXTREME GRADIENT BOOSTING	21
2.3	ARTIFICIAL NEURAL NETWORKS	25
2.3.1	OVERVIEW	25
2.3.2	MULTI-LAYER PERCEPTRON	26
2.3.3	LONG SHORT-TERM MEMORY NEURAL NETWORK (LSTM)	29
2.4	ADVANCED FEATURES	32
2.4.1	FEATURE EXTRACTION	32
2.4.2	FEATURE SELECTION	34
2.4.3	MODEL SELECTION	36
3	DATA PREPROCESSING	38
3.1	DATA COLLECTION	38
3.2	DATA CLEANING	40
3.3	VARIABLES ENCODING	44
3.4	DATA SET DIMENSIONALITY REDUCTION	45
4	BUILDING PREDICTION MODELS	48

4.1	MODELLING WITH XGBOOST	49
4.2	CONSTRUCTING A MULTI-LAYER PERCEPTRON	49
4.3	CONSTRUCTING A LONG SHORT-TERM MEMORY NEURAL NETWORK	50
4.4	METRICS	50
4.5	EXPERIMENTAL RESULTS AND DISCUSSION	51
5	CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH	70
5.1	CONCLUSIONS	70
5.2	SUGGESTIONS FOR FURTHER RESEARCH	71
5.3	SCIENTIFIC PUBLICATIONS	71
	REFERENCES	73

1 INTRODUCTION

In the past decade, machine learning has provided effective web search, practical speech and image recognition, efficient statistical arbitrage, more accurate medical diagnostic systems, self-driving cars, and an immense improvement in understanding the human genome. Machine learning is so widespread that people use it every day without knowing it, changing lives and providing an irreplaceable service in the industry (GUPTA; GUSAIN; POPLI, 2016). Numerous researchers believe that is the best way to get to human level artificial intelligence.

Machine learning aims to spot a function based in a great set of data. The function is created with a first training group of data in order to forecast outputs of a previously unseen data set called testing set, always searching the highest possible accuracy. This could be seen as classification problems in the form of labels and classes or as regression problems in the form of continuous values (SJARDIN; MASSARON; BOSCHETTI, 2016). Nevertheless, it is often hard to find enough training data available, i.e., lack of suitable data, lack of access to the data, privacy problems, badly chosen tasks and algorithms, and a lack of resources; as a result, often several machine learning programs fail to deliver the expected value.

The goal of building these systems is that they could be adaptable, learn from their experience and being enough simple. With this purpose, two approaches stand out from machine learning: decision tree learning and artificial neural networks.

Decision trees have for a long time been used for classification purposes. A highly optimized classification system was obtained when Friedman *et al.* (FRIEDMAN; HASTIE; TIBSHIRANI, 1998) applied the boosting technique to perform decision tree ensembles. Later, the gradient descend approach was introduced to boosting (MASON *et al.*, 1999) and along with its stochastic variant (FRIEDMAN, 2002) the boosted tree model increased robustness against the overcapacity of the base learner by adding trees sequentially while fitting a simple parameterized function. Despite all its improvements and wide use, it still was not efficient in processing time and model complexity. In order to reduce the latter, Johnson *et al.* (JOHNSON; ZHANG, 2014) made a modification of gradient boosting, it was called Fully Corrective Greedy Algorithm, achieving higher accuracies and smaller models. Nevertheless, this was not enough. In order to cope with this complexity, it was created the Extreme Gradient Boosting (XGBoost), a highly scalable and accurate learning system (CHEN; GUESTRIN, 2016), which has gained a lot of attention in recent years for its simplicity and its great success in competitions such as Kaggle.

Alternatively, Artificial Neural Networks (ANNs) are versatile, powerful, and scalable, ex-

cellent to cope with large and tremendously complex machine learning tasks as, for example, categorizing billions of images as Google Images does or like DeepMind's AlphaGo that defeats the world champion and itself at the game of Go by learning and analyzing millions of past games (GÉRON, 2017).

Originally, ANNs were developed as a mathematical representation of biological brains capabilities when information is processed (MCCULLOCH; PITTS, 1988). Despite the fact that nowadays it is known that ANNs have little resemblance to real biological neurons, they are still popular as applied methods for the classification of patterns. A large number of varieties of ANNs have appeared over the years, with very diverse properties. A relevant difference is between ANNs which have acyclic connections and those with cyclic. ANNs without cycles are identified as feedforward neural networks (FNNs) and the most recognized example is the Multi-Layer Perceptron (MLP) that has its beginning with (ROSENBLATT, 1958). ANNs with cycles are described as recursive or better called Recurrent Neural Networks (RNNs). The main idea of RNNs is that they can manage contextual information when input and output sequences are mapped. Unluckily, in practice, conventional RNN architectures have a quite limited access to a range of context, because of the effect of vanishing gradient problem, in which the impact of an entry in the hidden layer(s) and, consequently, in the output decays or explodes exponentially as it travels through the recurrent connections in the network (GRAVES, 2012). One attempt to overcome this problem was proposed in 1997 by Hochreiter and Schmidhuber (HOCHREITER; SCHMIDHUBER, 1997) with Long Short-Term Memory (LSTM), which have emerged as an effective and scalable model for several learning problems related to sequential data (GREFF et al., 2017).

Around the world, fire brigades seek strategies to deal with the fact of response immediately to incidents. Researches based on the forecast of the number of interventions, response speed, materials and engine used by fire departments are scarce in the literature. However, we can find machine learning techniques applied to forecasting occupational accidents using Support Vector Machine (SVM) and Multi-Layer Perceptron (MLP) optimized by Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), where the aim was to predict the accident outcomes such as injury and property damage using occupational accident data (SARKAR et al., 2018). Another example, Ren *et al.* (REN et al., 2018) worked with a deep learning approach, using LSTM neural networks to predict the traffic accident risk, taking into consideration weather variables, periodical patterns and getting from the traffic accidents the spatial distribution patterns. The traffic accident data was discretized in space and time, and used as inputs to the deep model during training. Then, the model was provided with recent historical data to eventually obtain the real-time traffic accident risk prediction. Gerber (GERBER, 2014) developed a research using spatio-temporally tagged tweets with a Kernel Density Estimation (KDE) technique in order to predict crimes where the crime data collected belonged to the Chicago Police Department and the selected tweets were the ones tagged with the geographical positioning sys-

tem coordinates corresponding to the limits of Chicago and Illinois cities. In the work of (SHI et al., 2017), the author proposed an approach based on XGBoost to predict urban fire accidents using ten million samples as data set, an algorithm based on association rules to select features, as well as the Box-Cox transformation to clean outliers. As a result, they obtained 90% of accuracy. Finally, Pettet *et al.* (PETTET et al., 2017) developed a data-driven toolchain to forecast the likelihood of vehicular incidents in given time and location using incident data from the Nashville Fire Department of the United States of America, considering weather and road type information as added variables. The authors combined a Similarity Based Agglomerative Clustering to categorize incidents with similar characteristics, a survival analysis to learn the probability of incidents per cluster and a Bayesian Network inference technique to map the clusters to the spatial locations.

Thereby, most of the problems that firefighters face on are related to the management of the budget, rising call volume, and personnel and equipment shortages. On the other side, fire departments have been collecting data on their interventions; notwithstanding, few of them use data science to elaborate a decision making approach.

Knowing the number of interventions in the next hours can help to decrease the response time of firefighters and improve their ability to save lives and rescue people. In this research, it is developed three models for this purpose: one with XGBoost, another with MLP and a last one with LSTM. Finally, their performances are compared in order to find the most efficient in terms of prediction accuracy.

1.1 MOTIVATION

There is a need to build an accurate model that is capable of evaluating the possible occurrence of an incident using information that happened in the past. Therefore, it could help fire departments to establish well planned strategies with their allocated mobile and personnel resources. Thus, the response time of firefighters would be reduced and more lives would be safe.

In the literature, there are few researches related to the forecasts of interventions carried out by firefighters, that is the reason for which in this work three paradigms are constructed using different techniques of machine learning, and their performances are evaluated to discover which fits better to the considered data.

1.2 CONTRIBUTIONS

In this research, the aim is to develop, evaluate and compare the performance of three regression models applied to the prediction of the number of interventions that the fire brigade

SDIS25 in Doubs, France, would face on. The techniques used are:

- XGBoost, a software library which provides a gradient boosting framework;
- MLP, a classical machine learning method;
- LSTM, considered a popular and more advanced neural network algorithm.

The procedure establishes the following objectives:

- i) Collect and clean data from various sources. The initial data set is provided by the fire brigade “Service Départemental d’Incendie et de Secours” of Doubs. For complementary sources, it will be considered weather-related data, traffic hours, epidemiological data, academic vacations and holidays, astronomical variables as moon phase and others.
- ii) Encode and select data. As the initial data are not defined by float value pairs, it is necessary to applied feature extraction techniques such as Standard Scaler and One-Hot-Encoding from Scikit-Learn library in Python. And in order to recognize the relationship of all variables and if they all are needed for the modelling, it will be tested and compared two techniques: Mutual Information Regression (MIR) and Principal Component Analysis (PCA), also from Scikit-Learn library.
- iii) Build models with XGBoost, MLP and LSTM. During the construction of the models and to improve the predictions, it will be made a research to find the best configuration of hyperparameters. For XGBoost, it will be used a grid search and for MLP and LSTM, a random search.
- iv) Compare the models’ performances. This activity is made taking into account the Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Accuracy (ACC) as evaluation metrics.

Furthermore, this research shows the feasibility of the paradigms constructed, illustrates several variables not considered yet in the prediction and initializes discussions about these issues. Finally, it provides a path to the further development of predictive models for firefighters interventions considering machine learning techniques.

1.3 OUTLINE OF THE THESIS

The rest of the thesis is organized as follows:

-
- Chapter 2: gives an introduction to machine learning, explaining learning algorithms, its advantages, challenges and limitations considered in this research.
 - Chapter 3: describes how the data were collected, cleaned, encoded and selected before feeding the models, and depicts the tuning made to each model to improve their hyperparameters.
 - Chapter 4: exhibits the predicting models developed, the results obtained and a discussion interpreting the highlighting study results.
 - Chapter 5: concludes about the subject by offering important insights about the research problem and makes suggestions for future studies.

2 MACHINE LEARNING TECHNIQUES

2.1 INTRODUCTION

The Internet, the enormous computational power of the recently electronic devices and the fact that practically all process in the world make use of any kind of software are providing a great quantity of data every second. There are many things to do with this data such as analyze it to later take a decision, visualize it in a lot of ways or consider it as a source of experience to enhance the performance of the algorithms. These algorithms, which can learn from previous data, comprehends the field of Machine Learning (GARRETA; MONCECCHI, 2013).

Some definitions that try to encapsulate the ideal objective or ultimate aim of machine learning are expressed as follows:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E . (Mitchell, 1997, p.2)

Machine learning algorithms can figure out how to perform important tasks by generalizing from examples. This is often feasible and cost-effective where manual programming is not. As more data becomes available, more ambitious problems can be tackled. As a result, machine learning is widely used in computer science and other fields. (DOMINGOS, 2012, p.1)

As it is described in (GÉRON, 2017), there are many different types of machine learning algorithms and it is useful to classify them in broad categories based on:

- **Training with or without human supervision.**
 - **Supervised Learning.** During the training phase, the algorithm is fed by data called *Predictors* indicating the desired results named *Labels*. The task types could be classification, like classify new emails as spam or non-spam; or regression where the aim is to predict a numerical value, such as the price of a car. This last sort of prediction is also used for algorithmic trading. Some examples of supervised learning algorithms are: K-Nearest Neighbors, Linear Regression, Support Vector Machine, Decision Trees, Random Forests and several Neural Networks architectures (GÉRON, 2017; GOODFELLOW; BENGIO; COURVILLE, 2016).

- **Unsupervised learning.** Samples of the training data are unlabeled; therefore, the system attempts to learn without a guide. The model is trained with normal instances to later, during the prediction, recognizes abnormal ones. However, two main difficulties are known: the generalization that refers to the exponential growth of configurations during the learning process and the computational challenge where many algorithms imply intractable computations. As examples: Clustering, Visualization and dimensionality reduction, and Association rule learning are referred (GÉRON, 2017; GOODFELLOW; BENGIO; COURVILLE, 2016).
- **Semi-supervised.** The algorithms can handle a lot of unlabeled data with a few of labeled data. An example of these algorithms is the Deep Belief Networks (DBNs) that have the component Restricted Boltzmann Machines (RBMs) which is trained sequentially in an unsupervised manner and next fine-tuned using supervised learning techniques (GÉRON, 2017; GARRETA; MONCECCHI, 2013).
- **Reinforcement learning:** The system is named *Agent* and it has the ability to observe the environment, choose and execute actions to get rewards in return or penalties. Hence, the agent learns by itself the best strategy. (GÉRON, 2017; GARRETA; MONCECCHI, 2013).

- **Learning incrementally on the fly.**

When the system is trained using all data available and then launched into production without learning anything else, the process is called *Batch learning*. This is done offline and consumes plenty of time and resources. To update the system, it requires to be trained a new version of itself, including the new data and the old data; then, the old system would be stopped and replaced by the new one. In order to make the computation faster and more space efficient, it can be used another method called *Online learning*, where the system is trained sequentially by small groups called *Mini-batches*, learning new data on the fly (GÉRON, 2017; BURLUTSKIY et al., 2016).

- **Learning by generalization of examples.**

It is divided into two types: **Instance based-learning**, where samples are learned by heart and the generalization process is using a similarity criterion; and **Model-based learning**, where a model is built using a set of samples and the predictions are made based on this model (GÉRON, 2017).

For example, a state-of-the-art spam filter may learn on the fly using a deep neural network model trained; this makes it an online, model-based, supervised learning system.

Regardless of learning style or function, all combinations of machine learning algorithms consist of three components according to (DOMINGOS, 2012):

- **Representation.** The classifier must be depicted in a language that computer could understand.
- **Evaluation.** An evaluation function, also called objective function, is required to distinguish good classifiers from the least efficient ones. The objective function used locally by the algorithm may diverge from the external one that it is desired the classifier to optimize.
- **Optimization.** It is needed a method to optimize the classifiers and obtain the highest-scoring one. The selection of an optimization technique is essential to the efficiency of the learner, helping to discover more than one optimum.

As it was described before, there are different approaches to getting machines to learn. With all of their processing power, they are able to more quickly highlight or find patterns in big data that would have otherwise been missed by human beings. Machine learning is a tool that can be used to enhance humans' abilities to solve problems and make informed inferences on a wide range of situations.

However, there are some challenges and limitations to overcome, such as the historical overfitting and underfitting. This could be explained by dividing the generalization error into bias and variance, where the bias is a learner's tendency to constantly learn the same erroneous thing, resulting in underfitting, and variance is the proneness to learn random things regardless the real signal, resulting in overfitting. After overfitting, the most crucial problem in machine learning is the curse of dimensionality. This expression was coined by Bellman in 1961 to mention the fact that several algorithms that work well in low dimensions becomes unmanageable when the input is high-dimensional, consequently, it is more difficult to understand the data. Another characteristic to regard is the fact that when a learning algorithm is not giving good results, often the quicker path to successful outcomes is to feed the machine more data, which is the primary driver of progress in machine and deep learning algorithms in recent years. Nevertheless, this can lead to issues with scalability, in which more data is available but time to learn that data remains an issue (DOMINGOS, 2012).

A well-known approach that have made great gains over the past decade is Deep Learning, which is the study and design of machine learning algorithms for learning good representation of data at multiple levels of abstraction. Recent publicity of deep learning through DeepMind, Facebook, and other institutions has highlighted it as the "next frontier" of machine learning.

2.2 GRADIENT BOOSTING MACHINES

2.2.1 OVERVIEW

A Decision Tree is an analytical method with a schematic representation of alternatives that facilitates making better decisions. The goal of a decision tree is to learn a series of decision rules to infer the target labels. Utilizing an iterative algorithm, the procedure starts at the root level of the tree and splits the data recursively on the feature that gets the lowest impurity. The majority of scalable tree-based applications are based on Classification and Regression Trees (CART) introduced by Breiman, Friedman Stone and Ohlson in 1984. CART works building binary trees for classes or continuous variables, operating iteratively on given features and improving in a greedy way the results of an error metric expressed as impurity in order to finally obtain a prediction (SJARDIN; MASSARON; BOSCHETTI, 2016).

Decision trees are the basis for ensemble methods, such as bagging and boosting. Bagging creates several subsets of data from training sample chosen randomly, where each collection generates its own decision tree. All of these produce an ensemble of different models averaged, i.e., an aggregated predictor (BREIMAN, 1996). Boosting fits consecutive trees generated by a random sample, its goal is to solve for net error from the prior tree. When an input is incorrectly classified by a hypothesis, its weight is increased and therefore the next hypothesis will try to classify it in the most precise way. Thus, weak learners, which performs just slightly better than random guessing can be “boosted” into an accurate “strong” learner (FRIEDMAN; HASTIE; TIBSHIRANI, 1998; FREUND; SCHAPIRE, 1999).

When gradient descent procedure was combined with boosting technique, gradient boosting was created. Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating the error or loss, the weights are updated to minimize that error. Instead of parameters, weak learner sub-models or more specifically decision trees are considered. After calculating the loss, to perform the gradient descent procedure, a tree must be added to the model to reduce the loss. This is done by parameterizing the tree, then the parameters of the tree are modified and the algorithm moves in the right direction by reducing the residual loss and ameliorating the prediction. A specific number of trees are added, or the training terminates, once loss reaches a desirable level or when the results no longer improve on an external validation data set (MASON et al., 1999; FRIEDMAN, 2001; CHIO; FREEMAN, 2018).

2.2.2 EXTREME GRADIENT BOOSTING

XGBoost is a well engineered, distributed machine learning system to scale up tree boosting algorithms. It makes use of a novel regularization approach over the conventional Gradient

Boosting Machines (GBMs) to significantly decrease the complexity. The system is optimized for quick parallel tree construction and adapted to be fault tolerant under the distributed setting. It can handle millions of samples on a single node and manage billions of them with distributed computing (CHEN; GUESTRIN, 2016). Furthermore, unlike GBMs, Extreme Gradient Boosting makes use of a regularization method to considerably diminish the complexity of the model. One notable advantage is the availability of a set of parameters that are able to be modified during the training with the inputted data to enhance the model that is being constructed. (GUPTA; GUSAIN; POPLI, 2016). The process of how XGBoost works is described as follows.

a) Scoring function

In order to measure the performance of a model given a certain data set, XGBoost defines an objective function considering the training loss $L(\theta)$ and regularization $\Omega(\theta)$ terms, where the first measures how predictive the model is and the latter penalizes the complexity of the model and prevents the overfitting. This is represented in equation (1), where θ signifies the parameters that will be discovered during the training. For the mathematical structure, XGBoost makes use of a model based on decision tree ensembles, where the result model \hat{y}_i , at the i^{th} training example of a total of n , is the sum of the predictions of multiple trees together. Equation (2) shows K as the number of trees which are been combined, f as a function in the functional space F and x_i as the input. Thus, the new evaluation function to be optimized becomes in equation (3) (CHEN, 2014).

$$Obj(\theta) = L(\theta) + \Omega(\theta) \quad (1)$$

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), f_k \in F \quad (2)$$

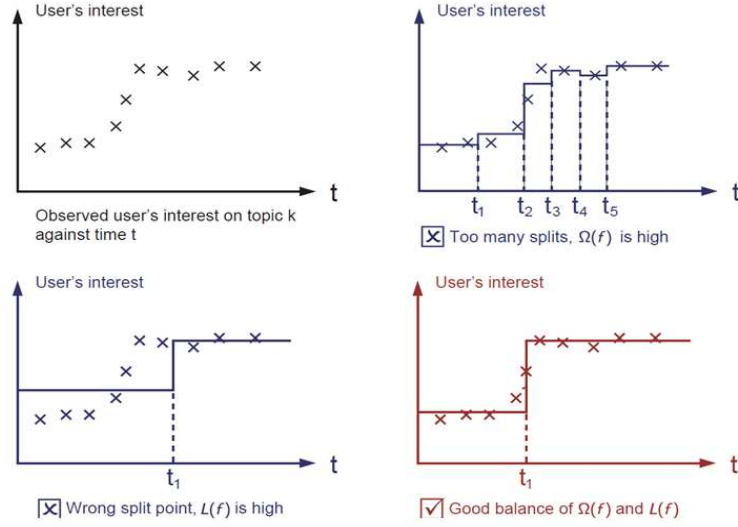
$$Obj = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k) \quad (3)$$

The aim is to find a trade-off between bias and variance, in other words, the model needs to be simpler and predictive. Simpler because it tends to have smaller variance making prediction stable, and predictive because it must fit well in training data. This can be visualized in Figure 1, where the upper image of the left corner shows the distribution of a variable that will be modelled, the upper image of the right corner presents the model with high complexity which derives in an overfitting, the bottom image of the left corner illustrates the model not learning enough patterns, i.e., an underfitting, and the bottom image of the right corner shows the model with a good balance between overfitting and underfitting.

b) Boosting

The additive strategy is applied during the training, one new tree that optimizes the system is added at a time to the model, in equation (4), $\hat{y}_i^{(t)}$ is described as the model at training

Figure 1 - Bias-Variance tradeoff in machine learning



Source: (CHEN, 2014)

the round t , $\hat{y}_i^{(t-1)}$ is the function added in the previous round and $f_t(x_i)$ is the new function (CHEN, 2014).

$$\hat{y}_i^{(t)} = \sum_{k=1}^t f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \quad (4)$$

c) Regularization

To determine the complexity of the tree $\Omega(f)$, (CHEN; GUESTRIN, 2016) proposed an approach that defines it as equation (5), where the first term γT evaluates the number of leaves T , taking γ as a constant, and the second term computes $L2$ norm of leaves scores w_j and λ is a very small constant value (CHEN, 2014).

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2 \quad (5)$$

d) Final structure outcome

As an example of an estimator, it is considered the Mean Squared Error (MSE) for the loss term; then it is taken the Taylor expansion of the loss to the second order, the objective function is described as equation (6) and displays how the splitting of the nodes would be done. G and H are defined in equations (7) and (8) respectively, where g_i and h_i are the first and second order partial derivatives after taking the Taylor expansion, $I_j = \{i | q(x_i) = j\}$ is the group of indices of data points attributed to the j -th leaf and $q(x)$ is the structure of the tree (CHEN, 2014).

$$obj^{(t)} = \sum_{j=1}^T [G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2] + \gamma T \quad (6)$$

$$G_j = \sum_{i \in I_j} g_i \quad (7)$$

$$H_j = \sum_{i \in I_j} h_i \quad (8)$$

Finally, in the objective function, it is taken the argument of the minimum and the minimum of the quadratic function for the single variable w_j , considering $q(x)$ as fixed. The outcomes are equations (9) and (10), where the last one assess how good a tree structure is, i.e., if the score is smaller, the structure will be better (CHEN, 2014).

$$w_j^* = -\frac{G_j}{H_j + \lambda} \quad (9)$$

$$obj^* = -\frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T \quad (10)$$

e) Greedy learning of the tree

In practice, it would be unmanageable to enumerate all possible tree structures and select the best one. Because of this trees are built in a greedy way, starting from a tree with depth zero; then, one level is optimized at the time until arriving at the maximum depth, this means, it is added a split for each leaf node of the tree. The gain is expressed as follows:

$$Gain = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma \quad (11)$$

In equation (11), the first term refers to the score of left child; the second is the score of right child; the third expresses the score if a split is not taken; and the last term consider a complexity cost for adding a leaf. From this, it is deduced that is better not to add a split if the gain is smaller than γ . This technique is called *Prunning* in tree based models.

In order to find the optimal split:

- i) For each node, all possible features are enumerated.
- ii) For each feature, instances are sorted by feature value.
- iii) A linear scan is used to decide the best split along that feature. This linear scan is taken from left to right and the split is done where values g_i and h_i give the best gain, this is calculated with equation (11).
- iv) The best split solution is taken along all the features.

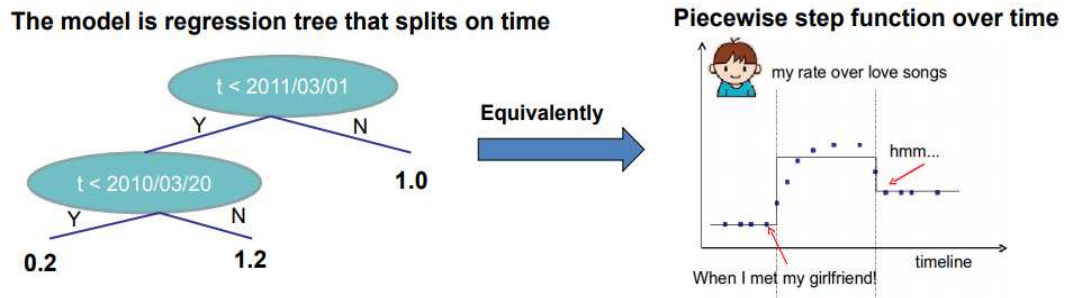
Finally, the $f_t(x)$ obtained by the new tree is added to the model. As it is shown in equation (12):

$$y_i^{(t)} = y_i^{(t-1)} + \varepsilon f_t(x_i) \quad (12)$$

The ε is called *Step-size* or *Learning Rate* and it usually takes values around 0.1. A complete optimization is not done in each step, like this, a chance of improvement is kept for future rounds to prevent the overfitting.

In Figure 2, it is illustrated an example of learning a tree on single variable, where the input variable is t (time) and the goal is to predict a person's preference of romantic music in a given time. The first image shows the construction of the tree and the second one shows how the data is being modelled through the time in order to obtain a final function.

Figure 2 - Learning a tree on single variable



Source: (CHEN, 2014)

For more details about the power of XGBoost method, its features and its algorithmic optimizations, (CHEN, 2014; CHEN; GUESTRIN, 2016; GUPTA; GUSAIN; POPLI, 2016) are strongly suggested for interested readers.

2.3 ARTIFICIAL NEURAL NETWORKS

2.3.1 OVERVIEW

The first occasion that ANNs were mentioned was in 1943 by the neurophysiologist Warren McCulloch and the mathematician Walter Pitts (MCCULLOCH; PITTS, 1988). They introduced an elementary computational model of how biological neurons from animal brains could work together to perform tasks through the use of propositional logic. This representation was known as the first artificial neural network. During ANNs' history, they passed for a long dark period in the course of the 1960s and 1970's where they were set aside. At the beginning of the 1980's, a revival interest in them returned allowing the development of new architectures and

better training techniques. For the next decade, better results and theoretical foundations were achieved by powerful machine learning techniques. Nowadays, with the tremendous increase in computing power, the enormous amount of data available, finer training algorithms and some theoretical limitations that have become advantageous in practice have permitted ANNs to be in a continuous evolution with accelerated progress and more fundings (GÉRON, 2017).

The basic structure of an ANN is compounded by small processing units or nodes with weighted associations which in the biological model represent neurons and the strength of the synapses between the neurons respectively. The network is stimulated by given inputs to some or all nodes, this effect is spread throughout the weighted connections in the network. The activation of an ANN node tries to model the average firing of the spikes observed as a result of the electrical activity in biological neurons (GRAVES, 2012).

From among all ANNs architectures, there is a special distinction ANNs whose connections are acyclic called Feedforward Neural Networks (FNNs), and those whose associations form cycles called Recurrent Neural Networks (RNNs). In the case of FNNs, the most representative is the Multi-Layer Perceptron (MLP) and for RNNs is the Long Short-Term Memory (LSTM) that in last years has demonstrated a highlighted performance. These two representations are described as follows (GRAVES, 2012).

2.3.2 MULTI-LAYER PERCEPTRON

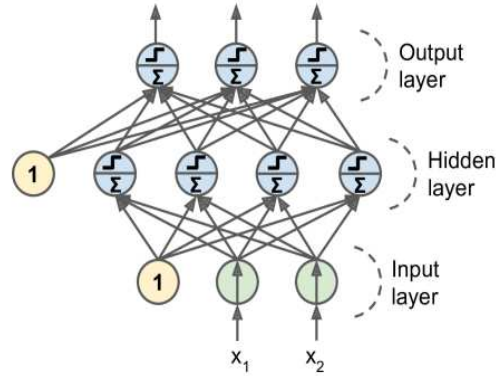
An MLP works with a set of units called neurons organized in an input layer, one or more hidden layers and an output layer, all fully connected; each layer contains a bias neuron. Thus, inputs are propagated from the first layer, passing through hidden layers until reaching the output layer, this is called the forward pass of the network and could be observed in Figure 3. When a ANN has two or more hidden layers is called Deep Neural Network (DNN) (GRAVES, 2012; GÉRON, 2017).

A brief description of how Multi-Layer Perceptron works is given in (GRAVES, 2012) as follows:

An MLP with a particular set of weight values defines a function from input to output vectors. By altering the weights, a single MLP is capable of instantiating many different functions. Indeed it has been proven that an MLP with a single hidden layer containing a sufficient number of nonlinear units can approximate any continuous function on a compact input domain to arbitrary precision. For this reason, MLPs are said to be “universal function approximators”. (GRAVES, 2012, p.13)

There are some principal characteristics that were born with MLP and must be considered in the modelling of an ANN:

Figure 3 - A Multi-Layer Perceptron. The S-shaped curves in the hidden and output layers indicate the application of “sigmoidal” nonlinear activation functions.



Source: (GÉRON, 2017)

a) Loss function.

Also called "Error Function", it is the function that is going to be minimized or maximized during the training phase, its selection depends on the particular application. For classification problems, the goal is to model the posterior probabilities of class membership conditioned on the input variables. The most utilized are Categorical Crossentropy or Binary Crossentropy. For regression problems, the basic goal is to model the distribution of the output variables also conditioned on the input variables. The most used are Mean Squared Error (MSE) and Mean Absolute Error (MAE). In the case of modelling count data, which characteristics are represented by a discrete distribution and non-negative predicted values, Poisson distribution is widely used (BISHOP, 1995; GRAVES, 2012; CHOLLET et al., 2015; LONG, 1997). In Keras, a library of open source neural networks written in Python, Poisson loss function is calculated by equation (13):

$$Loss = \frac{1}{n} \sum_{i=1}^n (\hat{y}^{(i)} - y^{(i)} \log(\hat{y}^{(i)})) \quad (13)$$

Where:

n : number of samples.

$\hat{y}^{(i)}$: prediction of sample i^{th} .

$y^{(i)}$: target of sample i^{th} .

b) Backpropagation.

In 1986, D. E. Rumelhart *et al.* found a way to train MLPs by introducing the backpropagation algorithm in (RUMELHART; HINTON; WILLIAMS, 1986). During the training phase, each instance is used by the algorithm to feed the network and calculate the output of each neuron in each successive layer. The network's error is computed by calculating the variation between the desired output and the real output of the network at the moment; then,

it is determined the quantity of error contribution that each neuron of the last hidden layer has in the output neuron's error. This process continues in each previous hidden layer until reach the input layer. In this manner, the error gradient is measured efficiently during the reverse pass through all the connection weights and finally, with the gradients calculated the algorithm makes a gradient descent step on all the connection weights (GÉRON, 2017).

According to (HAYKIN, 1998), for each iteration, weights can be updated following the next rule:

$$w_{ij} = \alpha w_{ij}(n-1) + \varepsilon \delta_j(n) y_i(n) \quad (14)$$

Where:

n : current iteration.

$n - 1$: previous iteration

w_{ij} : synaptic weight connecting neuron i to neuron j .

α : momentum.

ε : learning rate.

δ_j : local gradient.

y_i : input signal of neuron j .

c) Problem of vanishing/exploding gradients.

During the backpropagation in architectures with two or more layers, gradients often get more and more little as the algorithm progresses down to the first layers. Consequently, the lower layer connection weights stay without being updated by the gradient descent and the training phase does not converge to a good solution; or the opposite, when gradients become bigger and bigger and last layers get large weight updates making the algorithm diverges. In the literature, this is recognized as the vanishing or exploding gradients problem (GÉRON, 2017).

In (GLOROT; BENGIO, 2010) was analyzed this problem. Ideally, to avoid the vanishing gradient, the variance of each layer outputs should be the same with the variance of its inputs, also it would be required the gradients to have equivalent variance before and after flowing through a layer in reverse direction along the network. However, it is not possible to assure both; instead, what has worked very well on the practical level is the random initialization of the connections weights. This is described in Table 1, where n_{inputs} is the number of input connections and $n_{outputs}$ the number of output connections for the layer whose weights are being initialized (GÉRON, 2017).

Using the Xavier initialization procedure can accelerate training considerably. What is more, this method has been applied to deep learning giving excellent results (GÉRON, 2017).

d) Activation functions.

Table 1 - Initialization parameters for each type of activation function

Activation Function	Uniform Distribution $[-r, r]$	Normal Distribution
Logistic	$r = \sqrt{\frac{6}{n_{inputs} + n_{outputs}}}$	$\sigma = \sqrt{\frac{6}{n_{inputs} + n_{outputs}}}$
Hyperbolic Tangent	$r = 4\sqrt{\frac{6}{n_{inputs} + n_{outputs}}}$	$\sigma = 4\sqrt{\frac{6}{n_{inputs} + n_{outputs}}}$
ReLU (and its variants)	$r = \sqrt{2}\sqrt{\frac{6}{n_{inputs} + n_{outputs}}}$	$\sigma = \sqrt{2}\sqrt{\frac{6}{n_{inputs} + n_{outputs}}}$

Source: (GÉRON, 2017)

In order to work with the backpropagation, it had to be changed the step activation function by the logistic function, this is because of the Gradient Descent works in bumpy surfaces and the resulting flat segments of the step function do not allow it while the logistic function presents nonzero derivative everywhere admitting some progress at every step. Two other widespread activation functions are detailed in (GÉRON, 2017) as follows:

- **The hyperbolic tangent function (Tanh).** It shows an S-shaped, it is continuous and differentiable, its output value is on the limits of -1 and 1 making the layer's output more or less normalized which helps in the convergence.
- **The rectified linear unit function (ReLU).** It is continuous but not differentiable when the sum of the average weights is less or equal to zero, its range is between zero and infinity and tends to be fast to compute.

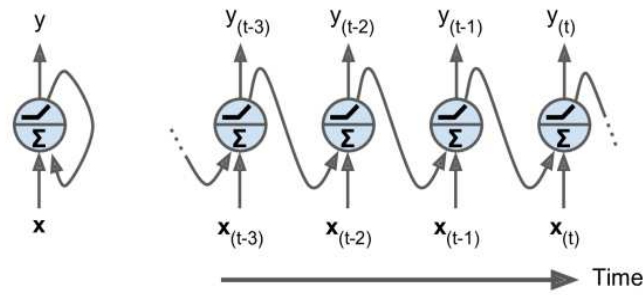
For more details about MLP architecture, (GRAVES, 2012; GÉRON, 2017; GLOROT; BENGIO, 2010; BISHOP, 1995) are recommended to interested readers.

2.3.3 LONG SHORT-TERM MEMORY NEURAL NETWORK (LSTM)

Unlike MLP that map from input to output vectors, RNN connections can save a memory of previous inputs and keep it in the network's internal state which impact on the output (GRAVES, 2012). This is similar to the fact of comprehending contexts based on the understanding of previous situations or completing phrases using first words as a basis.

The recurrent process is explained in Figure 4. On the left side, it is showed a recurrent neural network with one neuron which receives an input x , results in y and returns the outcome to itself. Generally, the activation function used is the hyperbolic tangent. On the right side of the picture, the recurrent network is unrolled and presents the recurrent neuron being fed by the inputs in different time steps, for each time step it is obtained a result that comes back as entry internally. The part of the network that holds some state over time steps is called *Memory Cell*. A single neuron or a layer of recurrent neurons is a *Basic Cell* (GRAVES, 2012; GÉRON, 2017).

Figure 4 - A recurrent neuron (left), unrolled through time (right).



Source: (GÉRON, 2017)

There are four kind of input and output sequences in recurrent neural networks:

- a) The network can receive a sequence of inputs and generate a sequence of outputs;
- b) The network can take a series of inputs, ignore all outputs, except for the last one;
- c) The network can admit a single entry at the first time step, complete with zeros the rest of time steps and return a sequence; and
- d) A sequence-to-vector network called *Encoder*, succeeded by a vector-to-sequence network called *Decoder*.

However, standard recurrent networks can access to a limited range of context due to the vanishing or exploding gradient problem explained in section 2.3.2.

In (HOCHREITER; SCHMIDHUBER, 1997), it was presented an improved version of RNN called Long Short-Term Memory, which is an effective and scalable model for solving problems related to sequential data such as handwriting recognition, speech recognition, language modelling, human activity recognition and traffic prediction. LSTM was designed to get over error backflow problems by truncating the gradient without affecting the training process. It is described by the authors in the following:

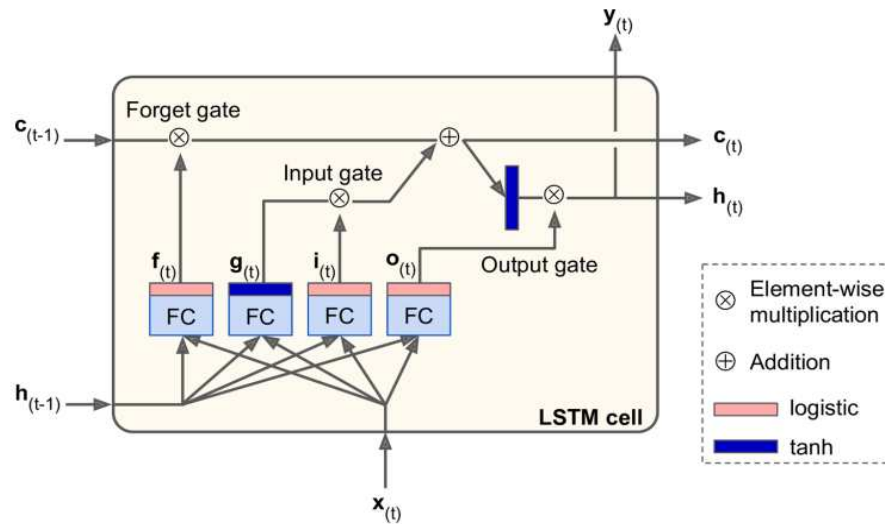
LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing constant error flow through “constant error carousels” within special units. Multiplicative gate units learn to open and close access to the constant error flow. In comparisons with RTRL, BPTT, Recurrent Cascade-Correlation, Elman nets, and Neural Sequence Chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long time lag tasks that have never been solved by previous recurrent network algorithms. (HOCHREITER; SCHMIDHUBER, 1997, p.1)

The LSTM architecture presents a set of subnets connected in a recurrent way, named memory blocks, each block has one or more auto-connected memory cells and three multiplicative

units called “gates” for input, output and forget, which represents operations such as writing, reading and restart for the cells (GRAVES, 2012).

In Figure 5, it is shown an LSTM cell, its state is divided into two: the short-term state $h_{(t)}$ and the long-term state $c_{(t)}$, the latter helps the network to recognize what to store, what to discard and what to read from it. As the long-term state $c_{(t-1)}$ travels the network from left to right, for each time step some memories are dropped when passing through the forget gate and new ones are added selected by the input gate via the addition operation, the resulting long-term state $c_{(t)}$ is duplicated, one copy is sent directly without any further change and another one is delivered to the tanh activation function and filtered by the output gate, producing the short-term state $h_{(t)}$. In this example, the resulting $h_{(t)}$ is equal to the cell’s output for this time step $y_{(t)}$ (GÉRON, 2017).

Figure 5 - LSTM cell



Source: (GÉRON, 2017)

But, where do new memories come from and how do the gates affect them?

- i) An input vector $x_{(t)}$ and the previous short-term state $h_{(t)}$ are entries to four fully connected layers.
- ii) The principal layer is the one that outputs $g_{(t)}$ which analyzes the inputs and the previous short-term states. It is partially stored in the $c_{(t)}$.
- iii) The input, forget and output layers are known as “gate controllers”. They make use of the logistic activation function, where their output values are between 0 and 1, “0” means closed gate and “1” opened gate. The forget gate $f_{(t)}$ manages which memories should be dropped from the long-term state. The input gate $i_{(t)}$ controls the resultant flow of $g_{(t)}$ that

should be partially added to the long-term state. Finally, at a specific time step, the output gate $o_{(t)}$ determines which memories of the long-term state should be resulting.

The equations to compute all LSTM process are:

$$i_{(t)} = \sigma(W_{xi}^T \cdot x_{(t)} + W_{hi}^T \cdot h_{(t-1)} + b_i) \quad (15)$$

$$f_{(t)} = \sigma(W_{xf}^T \cdot x_{(t)} + W_{hf}^T \cdot h_{(t-1)} + b_f) \quad (16)$$

$$o_{(t)} = \sigma(W_{xo}^T \cdot x_{(t)} + W_{ho}^T \cdot h_{(t-1)} + b_o) \quad (17)$$

$$g_{(t)} = \tanh(W_{xg}^T \cdot x_{(t)} + W_{hg}^T \cdot h_{(t-1)} + b_g) \quad (18)$$

$$c_{(t)} = f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)} \quad (19)$$

$$y_{(t)} = h_{(t)} = o_{(t)} \otimes \tanh(c_{(t)}) \quad (20)$$

Where:

$W_{xi}, W_{xf}, W_{xo}, W_{xg}$: are the weight matrices for their connection to the input vector $x_{(t)}$.

$W_{hi}, W_{hf}, W_{ho}, W_{hg}$: are the weight matrices for their connection to the previous short-term state $h_{(t-1)}$.

b_f, b_g, b_i, b_o : are the bias terms for each of the four layers.

For more details about LSTM and its architecture, (HOCHREITER; SCHMIDHUBER, 1997; GRAVES, 2012; GÉRON, 2017; KALCHBRENNER; DANIHELKA; GRAVES, 2015) are suggested.

2.4 ADVANCED FEATURES

2.4.1 FEATURE EXTRACTION

In supervised learning, which is the type of learning used in this research, a list of samples is organized in feature/value pairs called predictors and in one or more independent features called target classes which will be predicted based on the remaining features. Nevertheless, originally,

the source data is not structured like this, that is why, it is needed to apply feature extraction process to obtain the most useful ones where values can be integer, float, string, categorical, etc; and transform them to a specific format for the learning process such as categorical features into numerical values through the use of One-Hot Encoding (OHE) method where one attribute will be equal to “1” (hot), while the others will be “0” (cold). Consequently, the training time will be reduced (GARRETA; MONCECCHI, 2013).

In this research, StandardScaler and OneHotEncoder methods from Scikit-learn (PEDREGOSA et al., 2011) and Pandas (MCKINNEY, 2010) libraries respectively are used to convert features.

- **Standard scaler.** It is a method that standardizes features by rescaling the distribution of values to zero mean and unit variance. The standard outcome for each sample x is:

$$z = \frac{(x - u)}{s} \quad (21)$$

Where:

u : mean of the training samples, when the parameter *mean* = *False*, $u = 0$.

s : standard deviation of the training samples, when the parameter *std* = *False*, $s = 1$.

After centering and scaling, the mean and standard deviation are stored to be used on later data with the transform method.

In Figure 6, as an example, it is considered a data set to identify the type of the iris plant, using as features: sepal length, sepal width, petal length, petal width and the type of the plant as the target feature. The initial set of features is taken and transformed using the standard scaler method, resulting in the second data set.


Figure 6 - Standard scaler representation

	sepal length	sepal width	petal length	petal width	target		sepal length	sepal width	petal length	petal width	target
0	7.1	3.0	5.9	2.1	Iris-virginica		1.247439	-0.519122	0.991770	0.962425	Iris-virginica
1	6.9	3.1	4.9	1.5	Iris-versicolor		1.009832	-0.047193	0.446842	0.212483	Iris-versicolor
2	5.1	3.5	1.4	0.2	Iris-setosa		-1.128635	1.840524	-1.460409	-1.412390	Iris-setosa
3	6.3	3.3	6.0	2.5	Iris-virginica		0.297009	0.896665	1.046263	1.462386	Iris-virginica
4	6.3	2.9	5.6	1.8	Iris-virginica	→ Applying Standard Scaler →	0.297009	-0.991051	0.828292	0.587454	Iris-virginica
5	6.4	3.2	4.5	1.5	Iris-versicolor		0.415813	0.424736	0.228870	0.212483	Iris-versicolor
6	7.0	3.2	4.7	1.4	Iris-versicolor		1.128635	0.424736	0.337856	0.087493	Iris-versicolor
7	5.8	2.7	5.1	1.9	Iris-virginica		-0.297009	-1.934910	0.555827	0.712444	Iris-virginica
8	4.9	3.0	1.4	0.2	Iris-setosa		-1.366243	-0.519122	-1.460409	-1.412390	Iris-setosa
9	4.7	3.2	1.3	0.2	Iris-setosa		-1.603850	0.424736	-1.514902	-1.412390	Iris-setosa

Source: Developed by the author

- **One hot encoder.** It converts categorical variables into indicator features. For instance, if we have the categorical variable “country” with the values [‘Peru’, ‘Brazil’, ‘France’] can be encoded into a binary vector with 3 positions, having as value “1” where the country is present and the remaining positions as “0”. For a better visualization see Figure 7.

Figure 7 - One hot encoder representation



country		country_Brasil country_France country_Peru		
0	Peru	0	0	1
1	Brasil	1	0	0
2	France	0	1	0

Source: Developed by the author

2.4.2 FEATURE SELECTION

Usually, it is desired to use every available feature in the learning data set, since it is believed that using as much information would build a better model. However, there are two main reasons why the number of considered features should be restricted. First, it is possible that unimportant features could establish correlations between features and the target that arise just by chance and do not correctly model the problem, this may lead to poor generalization or add redundant information. And second, a large number of features could enormously increase the computation time without any improvement (GARRETA; MONCECCHI, 2013).

As a result, working with a smaller set of features may conduct to better results. For this reason, it is required to search an algorithmically way to discover the best features. Thus, for this work, it will be used “Mutual Information Regression” and “Principal Component Analysis” (PCA) methods from the open source library Scikit-learn (PEDREGOSA et al., 2011) to select features. Both of them will be evaluated and compared in order to find which method best captures the necessary features that will be used as inputs for the models developed with XGBoost, MLP and LSTM.

- **Mutual information regression.** It is a non-parametric method that evaluates the mutual correspondence between two variables. When the two variables are independent the output is zero, but higher outputs mean higher dependency (PEDREGOSA et al., 2011).

The method is based on information theory, where the Mutual Information (MI) is the quantity of uncertainty in a target variable that is removed by knowing a random variable, i.e., the MI is the amount of shared information, usually, measured in units called *bits* (SHANNON, 1948). However, the approach to calculate MI differs in the kind of variables, if they are discrete or continuous. Continuous values are more difficult to deal with, because basically, they are sparsely sampled. In the special case, where one variable is discrete and the other is continuous, which is the case of the data set used in this work, a method described in (ROSS, 2014) is used to overcome this problem using k-nearest neighbors. The approach computes the mutual information $I(X, Y)$ as the weighted form of I_i , where I_i is the Jensen-Shannon divergence (GROSSE et al., 2002) applied to each data point i . The data point i belongs to a list of (x, y) , where $x \in X$ and $y \in Y$, considering

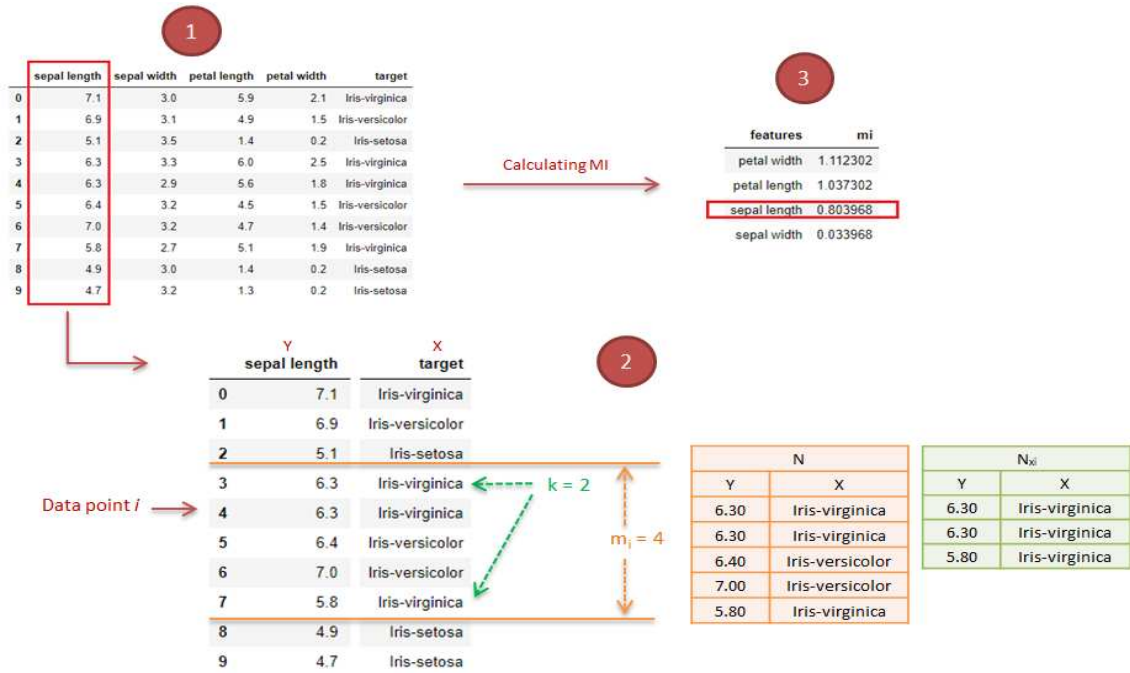
X as the discrete variable and Y as the real-valued variable. Initially, it is established the k^{th} -nearest neighbor to the data point i within all N_{xi} , where the parameter k is a fixed integer and N_{xi} are the data points whose value of the discrete variable equals xi . Then, it is counted the total number of neighbors m_i , including the ones whose value of the discrete variable does not equal xi but lie in the distance to the k^{th} -nearest neighbor. N are all data points in X and $\psi(\cdot)$ is the digamma function taken from (ABRAMOWITZ, 1974).

$$I_i = \psi(N) - \psi(N_{xi}) + \psi(k) - \psi(m_i) \quad (22)$$

$$I(X, Y) = \langle I_i \rangle = \psi(N) - \langle \psi(N_x) \rangle + \psi(k) - \langle \psi(m) \rangle \quad (23)$$

In Figure 8, from the iris plant data set, it is extracted the first column to illustrate how the k -nearest neighbors are taken in order to sample distributions. Later, it is applied the equation (23) and obtained its mutual information observed in the third part of the picture, considering the representation of each iris category as an integer number to make the computation. For more details about the mathematical analysis of the procedure, see (ROSS, 2014).

Figure 8 - Procedure for estimating the MI



Source: Developed by the author

- **Principal component analysis.** It is a data compression technique. It makes use of linear algebra through the Singular Value Decomposition (SVD) in order to find orthogonal directions of greatest variance, i.e, it discovers data combinations that retain the most information. SVD decomposes any matrix A into three pieces: U , Σ and V^t , where the

matrix U shows the eigenvectors with the most meaningful direction of the data and the matrix Σ depicts the total variance in the data. This is represented in equation (24). In scikit-learn library (PEDREGOSA et al., 2011), the attribute `n_components_` is the number of components selected and the attribute `explained_variance_ratio_` represents the variance percentage of each selected component.

$$A = U\Sigma V^t \quad (24)$$

Where:

U : left singular vectors (orthogonal matrix).

V^t : right singular vectors (orthogonal matrix).

Σ : singular values (diagonal matrix).

Thus, the data dimension can be reduced by generating transformed variables called *components* from the original ones. As an example and continuing with the data set of the iris plant, in Figure 9, the second data set demonstrates that the 90% of the variance is concentrated in two principal components that can be taken as a new data set. For more details about Singular Value Decomposition, see (STRANG, 2016) and about using the PCA technique, see (PEDREGOSA et al., 2011; GÉRON, 2017).

Figure 9 - Transformation with PCA

sepal length	sepal width	petal length	petal width	target		principal component 1	principal component 2	target	
0	7.1	3.0	5.9	2.1	Iris-virginica	0	-1.908412	-0.055298	Iris-virginica
1	6.9	3.1	4.9	1.5	Iris-versicolor	1	-0.918319	-0.333015	Iris-versicolor
2	5.1	3.5	1.4	0.2	Iris-setosa	2	2.724024	-1.163463	Iris-setosa
3	6.3	3.3	6.0	2.5	Iris-virginica	3	-1.353707	-1.108685	Iris-virginica
4	6.3	2.9	5.6	1.8	Iris-virginica	4	-1.231869	0.738291	Iris-virginica
5	6.4	3.2	4.5	1.5	Iris-versicolor	5	-0.357361	-0.565738	Iris-versicolor
6	7.0	3.2	4.7	1.4	Iris-versicolor	6	-0.720979	-0.799774	Iris-versicolor
7	5.8	2.7	5.1	1.9	Iris-virginica	7	-1.084514	1.830084	Iris-virginica
8	4.9	3.0	1.4	0.2	Iris-setosa	8	2.222975	1.130954	Iris-setosa
9	4.7	3.2	1.3	0.2	Iris-setosa	9	2.628161	0.326644	Iris-setosa

Applying PCA with
variance ratio = 90%

Source: Developed by the author

2.4.3 MODEL SELECTION

Selecting the model parameters, known as hyperparameters, is another important step during the training. In this research, the GridSearchCV method from Scikit-learn is used for select the best model to XGBoost; and for MLP and LSTM neural networks, a Random Search is implemented.

- **Grid search cross-validation.** It is a method that performs an exhaustive search through cross-validation using specific parameter values for an estimator, this means, the classifier

or estimator will be trained for each combination and get a cross-validation accuracy at evaluating each one. Thus, results will be shown and the best parameters could be identified (PEDREGOSA et al., 2011; GARRETA; MONCECCHI, 2013).

Commonly, it is used with three or fewer hyperparameters due to the fact that its computational cost grows exponentially as the number of hyperparameters increases (GOODFELLOW; BENGIO; COURVILLE, 2016).

- **Random search.** Another method to optimize hyperparameters is Random Search, which defines a marginal distribution for binary or discrete hyperparameters, or uniform distribution for positive real-valued ones on log-scale as an example. There is a principal reason that why random search converges much faster to good values than grid search and it is because the former does not misspend experimental executions (they would usually have different values), unlike the latter (GOODFELLOW; BENGIO; COURVILLE, 2016).

3 DATA PREPROCESSING

3.1 DATA COLLECTION

The departmental fire and rescue, SDIS 25, in the region Doubs-France has provided a set of information collected from 2012 to 2017 containing a quantity of 195 628 interventions. This data file contains: the identifier code of an intervention, date and time of the start and end of an intervention, date and time in which the first engine arrives, the community where the intervention happens, the classification of the incident, response time and duration of the intervention.

From the list of interventions gathered, it was extracted the date and time in order to detect tendencies correlated with these parameters. For instance, the number of car accidents increases on Saturday night because young people tend to drink more alcohol during this period of time.

Other factors considered in the occurrence of incidents are the weather conditions, that affects significantly the number of road accidents, fires and casualties; traffic hours, height of the main rivers in Doubs, epidemiological data, academic vacations, holidays, moonrise, moonset and moon phase in order to predict the number of interventions that will occur in the next hour. Therefore, at the beginning, it was created a dictionary with the extracted data from the fire department and the supplementary data imported from other sources together. The process is explained as follows:

- The dictionary is initialized containing keys ranging from “01/01/2012 00:00:00” until “31/12/2017 23:00:00” in the form “YYYYMMJJhhmmss”. The keys are generated by blocks of one hour.
- The following weather-related data reported by “Meteo France” (FRANCE, 2019) was imported from three stations located in Dijon-Longvic, Bale-Mulhouse, and Nancy-Ochey: temperature, pressure, pressure variation each one hour, barometric trend type, total cloudiness, humidity, dew point, precipitation in the last hour, precipitation in the last three hours, average wind speed for every ten minutes, average wind direction for every ten minutes, bursts over a period, horizontal visibility, and finally the current time. However, the data were not complete, some fields were missing. For this reason, it was applied a linear interpolation to fill the blanks. Finally, the meteorological data were added to the dictionary.
- It was introduced various temporal information: hour, day, day in the week, day in the

year, month and year.

- It was considered the height of twelve rivers, the most representative of the Doubs department, the data reported by “Hydro” (HYDRO, 2015) are from the stations: L’Allan à Courcelles-lès-Montbéliard, Le Doubs à Voujeaucourt, Le Doubs à Besançon, La Loue à Ornans, L’Ognon à Montessaux, L’Ognon à Bonnal, Le Dessoubre á Saint-Hippolyte, Le Doubs á Mouthe, Le Doubs á Mathay, Le Dugeon á Rivière Dugeon, Le Gland á Meslières and La Loue á Vuillafans. The dictionary was filled with the average of the readings closest to the time of the block considered, the standard deviation of variation of the water height on this block, the number of readings during this block, the maximum height occurred during this block of time with the alert 1 (true) if the height of the river pass a limit established as a flood alert or 0 (false) if not.
- From the list of interventions given by the fire brigade department, the interventions were organized according to the date of occurrence, grouping them by a period of one hour to add it to our dictionary.
- Holidays were considered as a binary variable initialized with 0 (false), that will be 1 (true) for any one hour block within an academic holiday period. Also, it was contemplated the start and end of vacations as a binary variable where it is 1 (true) for the days corresponding to the beginning and end of holiday periods and 0 (false) if not.
- Public holidays were added with values 1 or 0, for true or false respectively, as well as a second key that is set to 1 the days before public holidays, for the hours ranging from 3:00 pm to 11:00 pm (otherwise 0).
- It was included information related to the “Bison Futé” (FUTÉ, 2015) which is a system put in place in France to communicate to motorists all the recommendations of public authorities regarding traffic, traffic jams, bad weather, accidents, advices, etc. It classifies the days at risk according to several colors: green = fluid traffic, orange = dense traffic, red = difficult traffic, black = to avoid because of traffic jams and slow traffic. We integrate these information with two additional keys related to the departure and the return. They are 0, 1, 2 or 3, depending on whether the traffic forecasts correspond to green, orange, red or black.
- It was incorporated weekly epidemiological information organized by each given hour and related to the incidence of chickenpox, influenza and acute diarrhea, collected from the “Sentinelles” network (SENTINELLES, 2007).
- Finally, it was added to the dictionary a boolean variable to know if it is a day (0) or night (1) for each given hour. Moreover, we added another boolean variable to recognize if

the moon had already risen, this was determined considering the given hour plus thirty minutes, and what is its phase (an integer from 0 to 7, namely 0 for new moon, 2 for the first quarter, 4 for the full moon, and 6 for last quarter).

In Table2, it is shown how the dictionary looks like. It contains the information extracted from the list of interventions (number of interventions, hour, day, etc.) and the ones imported from external sources (meteorological, rivers height, ephemeris, traffic, diseases, vacations, etc.). Each line represents a block of one hour. Over the period of 6 years, for each day we have twenty four columns representing the 24 hours.

Table 2 - Illustrated example of the dictionary

ID	year	startEndHolidays	...	windDirectionBasilea	humidityDijon	temperatureNancy	nbInterventions
0	2012	0	...	240	97	283.35	7
1	2012	1	...	216	96	283.38	10
2	2012	0	...	193	95	283.45	9
...
52560	2017	0	...	200	96	277.45	10

Source: Developed by the author

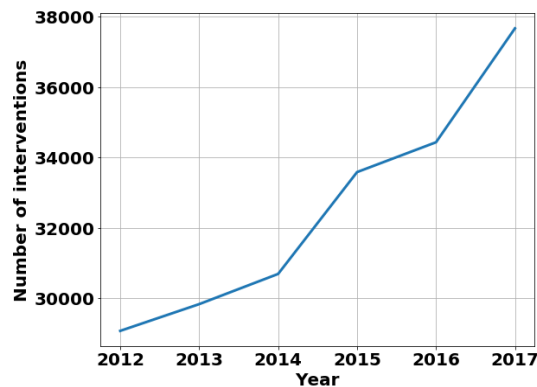
3.2 DATA CLEANING

In this subsection, it is explained how outliers, that can affect negatively the final results, were detected and removed. At first, it was analyzed the number of interventions per year in Figure 10 and it is observed a great increment of them over the 6 years, maybe related to the population-ageing and growth. Then, it was calculated the mean value of the number of interventions per hour, the average was 3.59 interventions/h. Moreover, it was found that the minimum number of intervention per hour is 0, while the maximum is 85. Finally, in 75% of cases the number of interventions is less than 5.

Leap years have an impact on the variable of the day in the year. For instance, June 21th (Music day in France) or July 14th (the National Day of France) are not the same day in the year when the month of February has 29 days. For this reason, February 29th of 2012 and 2016 were removed.

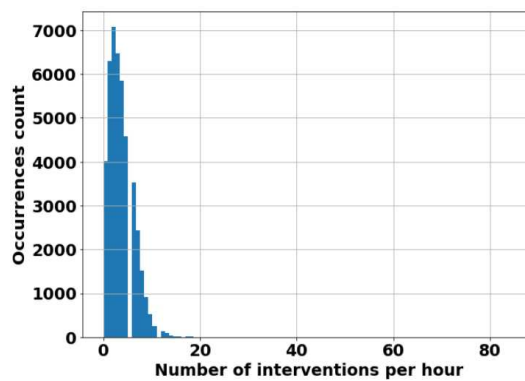
On the other side, looking at Figure 11 and in more detail Figure 12, there seems to be some very particular situations that generated a large number of interventions (more than 80). These events can affect the learning phase. Therefore, they were analyzed in more detail to know if they should or not be discarded. The hours were sorted, ranging from 0 to 52 559, in descending order according to their corresponding number of intervention. It was noticed that the first 7 IDs with the highest number of interventions correspond to the same period of time.

Figure 10 - Number of interventions per year



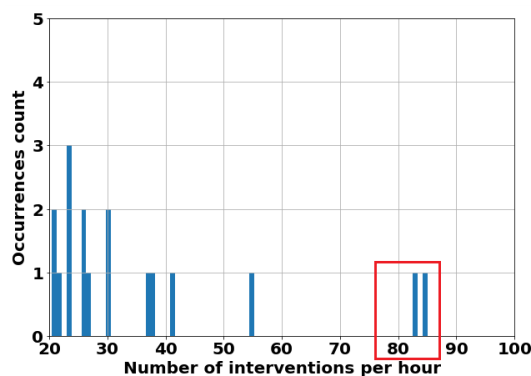
Source: Developed by the author

Figure 11 - Frequency of number of interventions per hour



Source: Developed by the author

Figure 12 - Outliers in the occurrence of interventions

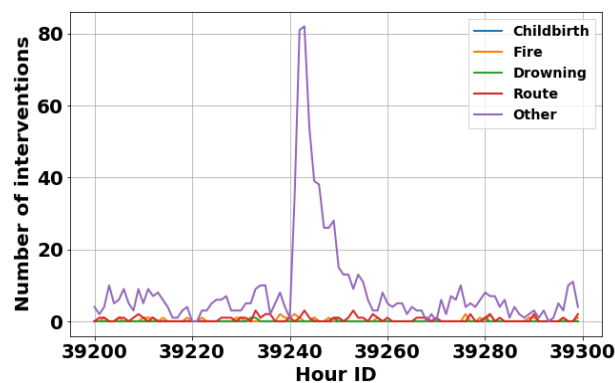


Source: Developed by the author

The ID number 39243 has the maximum number of interventions, that is 85, illustrated in Figure 13. The year, month, day, and hour of the 7 IDs were listed and it was noticed that they all belong to the night of 24th to the 25th of June 2016. In the list of interventions given by the fire department, the following main causes were noted for these days: exhaustion, floods,

protection of miscellaneous property, and accidents. It was found that there were very violent storms that night and that was recognized as a natural disaster in the region of Doubs. Therefore, it is necessary to evaluate if these outliers should be considered as a consequence of exceptional weather and be artificially smoothed or keep them to be predicted with weather variables. In what follows, a forecasting analysis of these picks will be made with meteorological data from Basilea, Dijon, and Nancy.

Figure 13 - Maximum number of interventions



Source: Developed by the author

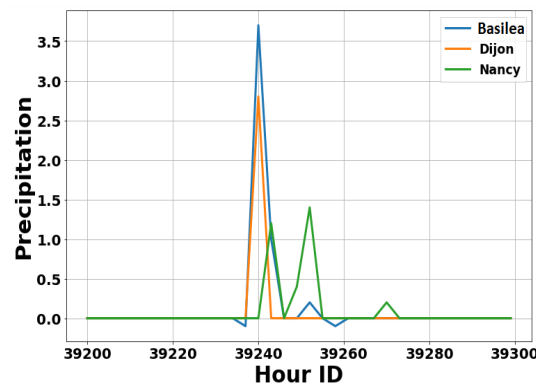
It is taken an interval of one hundred hours (approximately four days) centered around this storm. While checking the precipitation in the last one hour, Figure 14, it can be observed a peak in rainfall during the last hour, almost four millimeters, but not enough bigger than eighty millimeters that the article in Figure 15 from the “Est Republicain” (REPUBLICAIN, 2016) mentions. If it is compared the IDs of the peaks with the IDs of the maximum precipitations in the data provided by the weather station in Basilea (Figure 16), it seems that they are not unusual values. Although Basilea is closer to the storm location (between Sancey and L’Isle-sur-le-Doubs, Figure 17) than Dijon or Nancy, it is possible that its precipitations values are not very representative.

Nevertheless, it might be possible that a lot of water has fallen for a relative long time. Therefore, Figure 18 shows the precipitation over a period of three hours. A thunderstorm peak appears clearly, and the amount dropped twenty millimeters is closer to the eighty millimeters mentioned above. With the data, it is checked if such quantity twenty millimeters is something frequent and it turns out to be the fifth highest rainfall in three hours recorded from 2012 to 2016.

These precipitation data are very important for the prediction model, but they do not allow the prediction of the extreme situation of June 25th, 2016 due to weather measurements that are not sufficiently localized. Another variable analyzed was the wind speed. However, the maximum value in the defined time interval (Figure 19) is less than 15.9m/s which is the maximum wind speed from 2012 to 2017. With the weather data considered at the moment, it looks like

complicated to forecast such a peak of interventions. This kind of situation is very exceptional and has a big impact on the data studied. For instance, the number of incidents in June might be considerably overvaluated. For this reason, the chosen option is to artificially smooth the data on this date by putting the same number of interventions at the same time of the following day.

Figure 14 - Precipitation each 1h



Source: Developed by the author

Figure 15 - L'Est Republicain reports on the storms



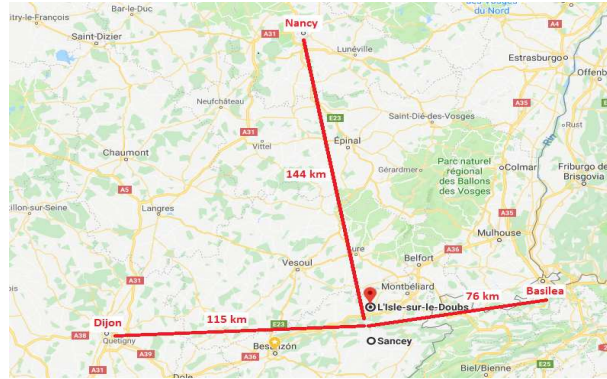
Source: (REPUBLICAIN, 2016)

Figure 16 - Precipitations peaks recorded from Basilea Station

	precipitations3hBasilea	year	month	day	hour
30237	32.300000	2015	6	14	21
22005	27.000000	2014	7	6	21
22004	23.000000	2014	7	6	20
30238	22.200000	2015	6	14	22
39240	21.500000	2016	6	25	0
30236	21.500000	2015	6	14	20

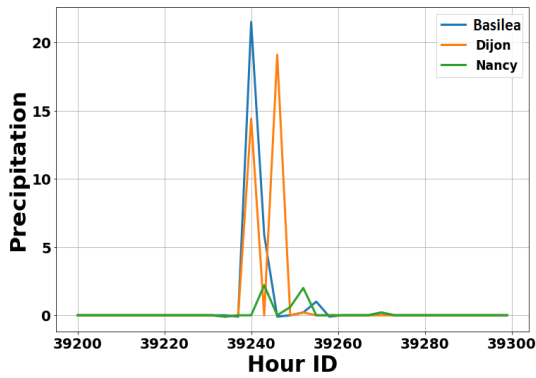
Source: Developed by the author

Figure 17 - Storm area



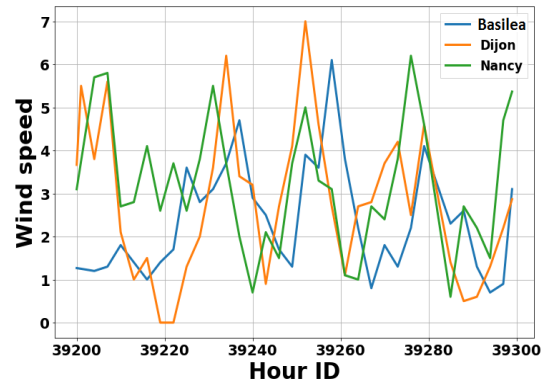
Source: Developed by the author

Figure 18 - Precipitation each 3h



Source: Developed by the author

Figure 19 - Ten minutes average wind speed



Source: From Author

3.3 VARIABLES ENCODING

First, the data was divided in two sets: learning (years 2012-2016) and testing (year 2017). It was used the StandardScaler method to extract features from the numerical variables. The mean and the standard deviation was computed with the learning set. Then, the standardization by centering and scaling was made with the complete data set, i.e., with the learning and testing sets. This process is made in order to discover changes in the data through the time and capture them during the training process. The numerical variables considered were: year, hour, wind direction, humidity, nebulosity, moon phase, dew point, precipitations, bursts, temperature, visibility, wind speed, chickenpox statistics, influenza statistics, acute diarrhea statistics, rivers height variables except by the alert variable. More details are presented in Table 3.

In the case of the categorical variables, they were encoded with the method OneHotEncoder. OHE was fitted to the complete data set in order to recognize all the categories for each variable. Then the data was transformed with the identified pattern. The categorical variables considered were: bison futé variables, time variables such as day, day of the week, day of the year and

month, holiday indicator, night indicator, barometric trend, river height alert variable. These variables are shown in Table 4.

Finally, the target feature “nbInterventions” was not standardized, after several previous tests, better results were obtained with original values. In total, 831 features were extracted.

Table 3 - Numerical variables

Variable : Description	
annee : year	varicelle_inc : number of registered incidents for chickenpox
heure : hour	varicelle_inc100 : number of registered incidents for chickenpox
directionVentBale : wind direction in Basilea	varicelle_inc100_low : number of registered incidents for chickenpox
directionVentDijon : wind direction in Dijon	varicelle_inc100_up : number of registered incidents for chickenpox
directionVentNancy : wind direction in Nancy	varicelle_inc_low : number of registered incidents for chickenpox
humiditeBale : humidity in Basilea	hLoueVuillafansMax : maximum records value of Loue river's height at Vuillafans
humiditeDijon : humidity in Dijon	hAllanCourcellesMean : records average of Allan river's height at Courcelles
humiditeNancy : humidity in Nancy	hDoubBesanconMean : records average of Doubs river's height at Besancon
nebulositeBale : nebulosity in Basilea	hDoubVoujeaucourtMean : records average of Doubs river's height at Voujeaucourt
phaseLune : moon phase	hLoueOrnansMean : records average of Loue river's height at Ornans
pointRoseeBale : dew point in Basilea	hOgnonBonnalMean : records average of Ognon river's height at Bonnal
pointRoseeDijon : dew point in Dijon	hOgnonMontessauxMean : records average of Ognon river's height Montessaux
pointRoseeNancy : dew point in Nancy	hAllanCourcellesStd : records std of Allan river's height at Courcelles
precipitations1hBale : precipitation in the last 1h in Basilea	hDoubBesanconStd : records std of Doubs river's height at Courcelles
precipitations1hDijon : precipitation in the last 1h in Dijon	hDoubVoujeaucourtStd : records std of Doubs river's height at Voujeaucourt
precipitations1hNancy : precipitation in the last 1h in Nancy	hLoueOrnansStd : records std of Loue river's height at Ornans
precipitations3hBale : precipitation in the last 3h in Basilea	hOgnonBonnalStd : records std of Ognon river's height at Bonnal
precipitations3hDijon : precipitation in the last 3h in Dijon	hOgnonMontessauxStd : records std of Ognon river's height at Montessaux
precipitations3hNancy : precipitation in the last 3h in Nancy	hAllanCourcellesNb : records number of Allan river's height at Courcelles
pressionBale : pressure in Basilea	hDoubBesanconNb : records number of Doubs river's height at Besancon
pressionDijon : pressure in Dijon	hDoubVoujeaucourtNb : records number of Doubs river's height at Voujeaucourt
pressionNancy : pressure in Nancy	hLoueOrnansNb : records number of Loue river's height at Ornans
pressionMerBale : pressure at sea level in Basilea	hOgnonBonnalNb : records number of Ognon river's height at Bonnal
pressionMerDijon : pressure at sea level in Dijon	hOgnonMontessauxNb : records number of Ognon river's height at Montessaux
pressionMerNancy : pressure at sea level in Nancy	hAllanCourcellesMax : maximum records value of Allan river's height at Courcelles
pressionVar3hBale : pressure variation in the last 3h in Basilea	hDoubBesanconMax : maximum records value of Doubs river's height at Besancon
pressionVar3hDijon : pressure variation in the last 3h in Dijon	hDoubVoujeaucourtMax : maximum records value of Doubs river's height at Voujeaucourt
pressionVar3hNancy : pressure variation in the last 3h in Nancy	hLoueOrnansMax : maximum records value of Loue river's height at Ornans
rafalesSurl1perBale : gusts in Basilea	hOgnonBonnalMax : maximum records value of Doubs river's height at Besancon
rafalesSurl1perDijon : gusts in Dijon	hOgnonMontessauxMax : maximum records value of Doubs river's height at Besancon
rafalesSurl1perNancy : gusts in Nancy	hDessoubreHippoMean : records average of Dessoubre river's height at Saint-Hippolyte
temperatureBale : temperature in Basilea	hDoubMathayMean : records average of Doubs river's height at Mathay
temperatureDijon : temperature in Dijon	hDoubMoutheMean : records average of Doubs river's height at Mouthe
temperatureNancy : temperature in Nancy	hDrugeonRiviereMean : records average of Drugeon river's height at La Rivière
visibiliteBale : horizontal visibility in Basilea	hGlandMeslieresMean : records average of Gland river's height at Meslières
visibiliteDijon : horizontal visibility in Dijon	hLoueVuillafansMean : records average of Loue river's height at Vuillafans
visibiliteNancy : horizontal visibility in Nancy	hDessoubreHippoStd : records std of Dessoubre river's height at Saint-Hippolyte
vitesseVentBale : 10 min average wind speed in Basilea	hDoubMathayStd : records std of Doubs river's height at Mathay
vitesseVentDijon : 10 min average wind speed in Dijon	hDoubMoutheStd : records std of Doubs river's height at Mouthe
vitesseVentNancy : 10 min average wind speed in Nancy	hDrugeonRiviereStd : records std of Drugeon river's height at La Rivière
diarhee_inc : number of registered incidents for diarrhea	hGlandMeslieresStd : records std of Gland river's height at Meslières
diarhee_inc100 : number of registered incidents for diarrhea	hLoueVuillafansStd : records std of Loue river's height at Vuillafans
diarhee_inc100_low : number of registered incidents for diarrhea	hDessoubreHippoNb : records number of Dessoubre river's height at Saint-Hippolyte
diarhee_inc100_up : number of registered incidents for diarrhea	hDoubMathayNb : records number of Doubs river's height at Mathay
diarhee_inc_low : number of registered incidents for diarrhea	hDoubMoutheNb : records number of Doubs river's height at Mouthe
diarhee_inc_up : number of registered incidents for diarrhea	hDrugeonRiviereNb : records number of Drugeon river's height at La Rivière
grippe_inc : number of registered incidents for influenza	hGlandMeslieresNb : records number of Gland river's height at Meslières
grippe_inc100 : number of registered incidents for influenza	hLoueVuillafansNb : records number of Loue river's height at Vuillafans
grippe_inc100_low : number of registered incidents for influenza	hDessoubreHippoMax : maximum records value of Dessoubre river's height at Saint-Hippolyte
grippe_inc100_up : number of registered incidents for influenza	hDoubMathayMax : maximum records value of Doubs river's height at Mathay
grippe_inc_low : number of registered incidents for influenza	hDoubMoutheMax : maximum records value of Doubs river's height at Mouthe
grippe_inc_up : number of registered incidents for influenza	hDrugeonRiviereMax : maximum records value of Drugeon river's height at La Rivière
varicelle_inc_up : number of registered incidents for chickenpox	hGlandMeslieresMax : maximum records value of Gland river's height at Meslières

Source: Developed by the author

3.4 DATA SET DIMENSIONALITY REDUCTION

The standardized data set used presents a high dimension: 831 features. For this reason, the methods: Mutual information regression and PCA were tested in a parallel way to evaluate and record the most relevant characteristics and use the regarded variables (outcomes from MI) or components (outcomes from PCA) to optimize the modelling process in time and memory without missing the integrity of the data. Naturally, it is necessary to bear in mind that features

Table 4 - Categorical variables

Variable : Description	
bisonFuteDepart : departure traffic level	tendanceBaromNancy : barometric trend Nancy
bisonFuteRetour : return traffic level	vacances : vacations
debutFinVacances : start/end vacations	veilleFerie : days before holidays
ferie : holidays	hAllanCourcellesAle : warning for Allan river's height at Courcelles
jourSemaine : day in the week	hDoubsBesanconAle : warning for Doubs river's height at Besançon
jour : day	hDoubsVoujaucourtAle : warning for Doubs river's height at Voujaucourt
jourAnnee : day in the year	hLoueOrnansAle : warning for Loue river's height at Ornans
mois : month	hOgnonBonnalAle : warning for Ognon river's height at Bonnal
luneApparente : sunrise or dusk	hOgnonMontessauxAle : warning for Ognon river's height at Montessaux
nuit : day or night	hDessoubreHippoAle : warning for Dessoubre river's height at Saint-Hippolyte
tempsPresentBale : weather conditions in Basilea	hDoubsMathayAle : warning for Doubs river's height at Mathay
tempsPresentDijon : weather conditions in Dijon	hDoubsMoutheAle : warning for Doubs river's height at Mouthe
tempsPresentNancy : weather conditions in Nancy	hDugeonRiviereAle : warning for Dugeon river's height at Rivière
tendanceBaromBale : barometric trend type in Basilea	hGlandMeslièresAle : warning for Gland river's height at Meslières
tendanceBaromDijon : barometric trend type in Dijon	hLoueVuillafansAle : warning for Loue river's height at Vuillafans

Source: Developed by the author

that look unimportant in isolation, i.e., when applying Mutual information regression or PCA, might be important in combination, i.e., during the modelling.

To get the Mutual information regression score, the entirety data set with the 831 extracted features was processed. The variables were divided in features and target, and 800 neighbors were taken into account to calculate the score. The outcomes were scaled between 0 and 1. Finally, a threshold to select the features was established with the value 0.01, i.e., all features with scores higher than 0.01 will be considered as input to the future models developed. Hence, 56 features and the target were taken to train the model. The selected features with their respective scores are shown in Table 5.

PCA was used as an alternative method to reduce the dimension of the standardized data set. The model was fitted with the learning set and it was retained the 95% of the variance. The transformation was applied to both data sets (learning and testing), i.e., the application of the dimensionality reduction to the data set with the 831 extracted features resulted in 83 principal components and the target.

The division on the learning and testing sets during the PCA fitting process corresponds to the idea of applying a real-world case, where initially, the testing set would not exist and the model would be fitted just with the learning set. Eventually, the future data (the testing set) would be transformed with the built model. In comparison with Mutual information regression, where it is possible to use the complete data to quantify the information between variables (because within the process samples are taken for the selection of variables), there would be no need to transform their values. As a summary, after analyzing this part of the process, a differentiation of both methods is made in Table 6.

Table 5 - List of features after applying mutual information regression selection

Feature : Mutual information regression score	
heure : 1.000000	luneApparente_1 : 0.027064
humiditeBale : 0.316402	nuit_1 : 0.026946
humiditeDijon : 0.230820	luneApparente_0 : 0.023712
humiditeNancy : 0.195237	pressionVar3hDijon : 0.021664
temperatureBale : 0.148263	ferie_0 : 0.018265
temperatureDijon : 0.144453	directionVentNancy : 0.018011
temperatureNancy : 0.122189	pointRoseeDijon : 0.014356
rafalesSur1perBale : 0.114104	tempsPresentNancy_0 : 0.014349
tempsPresentBale_0 : 0.091858	ferie_1 : 0.013768
rafalesSur1perNancy : 0.087239	hDrueonRiviereMean : 0.013719
rafalesSur1perDijon : 0.080465	pointRoseeBale : 0.013154
veilleFerie_1 : 0.078794	tempsPresentDijon_0 : 0.013146
veilleFerie_0 : 0.077291	hDrueonRiviereMax : 0.013036
vitesseVentBale : 0.066838	hDessoubreHippoMax : 0.012802
directionVentDijon : 0.064387	tempsPresentDijon_10 : 0.012493
visibiliteBale : 0.059371	tempsPresentBale_10 : 0.012339
vitesseVentNancy : 0.057049	precipitations3hBale : 0.012093
vitesseVentDijon : 0.047521	hDessoubreHippoMean : 0.011828
hDoubsBesanconNb : 0.046924	hDoubsBesanconMean : 0.011741
directionVentBale : 0.045515	hDoubsBesanconMax : 0.011203
visibiliteNancy : 0.042089	tendanceBaromBale_1 : 0.011116
visibiliteDijon : 0.039114	hOgnonBonnalMax : 0.010492
hDoubsBesanconStd : 0.036746	pointRoseeNancy : 0.010385
annee : 0.034604	hOgnonBonnalMean : 0.010360
tempsPresentBale_2 : 0.032631	pressionVar3hNancy : 0.010296
nebulositeBale : 0.032262	hDoubsMoutheMax : 0.010232
nuit_0 : 0.028023	tendanceBaromDijon_1 : 0.010222
pressionVar3hBale : 0.027969	tempsPresentNancy_10 : 0.010109

Source: Developed by the author

Table 6 - Differences between MIR and PCA

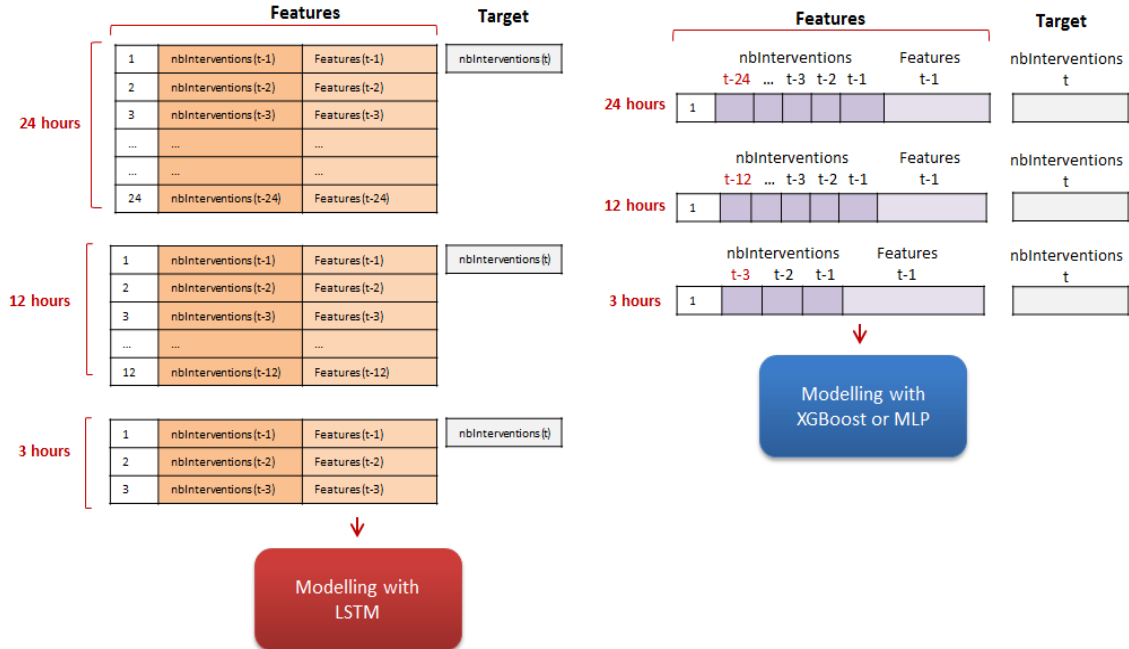
Mutual Information Regression	Principal Component Analysis
1. It is a non-parametric method, i.e, it does not need for previous standardization. However, in order to keep the same treatment of data to both methods, the standardization was made.	1. Standardization is needed to avoid emphasizing variables with higher variances than variables with low variances while searching for principal components.
2. Its base is the calculus of kth-neighbors and mutual information.	2. Its base is the calculus of SVD (linear algebra).
3. Relevant variables are choosen.	3. The variables and its values are transformed.
4. The new data set is reduced with the selected variables.	4. The new data set is made up of the new values (principal components).

Source: Developed by the author

4 BUILDING PREDICTION MODELS

After collecting, cleaning, encoding and selecting the features, data were split into three sets for the three kind of techniques: the training set (2011-2015), the validation set (2016) and the testing set (2017). The first and second sets are used to build a model that will predict the number of interventions for each one hour block since it is needed an immediate response to firefighters, and the last set is used to verify the accuracy of these predictions. LSTM works with time steps, in this way, it was used three, twelve and twenty-four hours as time steps with the goal of discovering which quantity of past hours give us better results. On the other hand, it was built a "similar" structure for XGBoost and MLP techniques which consists in defining a number of past interventions as new features for the next hour prediction in order to try to give an artificial memory to the models. The organization of the features for each technique is depicted in Figure 20.

Figure 20 - Features structure before modelling



Source: Developed by the author

The programming language used was Python, the gradient boosting models and the artificial neural networks were developed with XGBoost (CHEN; HE; KHOTILOVICH, 2016) and Keras (CHOLLET et al., 2015) libraries respectively. In order to run the codes, it was employed a GeForce GTX TITAN X, Intel® Xeon® CPU E5-2623 v4 @2.60GHz with an architecture

x86_64.

4.1 MODELLING WITH XGBOOST

In the training phase of the XGBoost model, the hyperparameters specified for the Grid-SearchCV were "max_depth": [3, 4, 5, 6, 7, 8], "learning_rate": [0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08], "colsample_bytree": [0.6, 0.7, 0.8, 0.9, 1], "n_estimators": [100, 200, 300, 400] and "objective": "count:poisson". The "count:poisson" corresponds to poisson regression for count data and uses "poisson-nloglik" as an internal evaluation metric where the prediction will be better if the probability of occurrence is higher. After finishing the execution on the server, the hyperparameters of the best model were tuned by hand in order to enhance the result. The values used to model our data are specified in Table 7. In (CHEN; HE; KHOTILOVICH, 2016) can be found more details about the hyperparameters specifications.

Table 7 - XGBoost hyperparameters settings

Hyperparameters	Description	Value
base_score	The initial prediction score of all instances	0.5
booster	It produces a tree based model	tree based model
colsample_bylevel	Subsample ratio of columns for each level	1
colsample_bytree	Subsample ratio of columns when constructing each tree	[0.6, 1]
gamma	Minimum loss reduction required to make a partition on a leaf	0
learning_rate	Step size shrinkage used in update	[0.02, 0.08]
max_delta_step	Absolute regularization that restricts weights	0
max_depth	Maximum depth of a tree	[3, 8]
min_child_weight	Minimum sum of instance weight to continue splitting	1
n_estimators	Number of trees	[100, 400]
n_jobs	Number of parallel threads	1
objective	Learning task and the corresponding learning objective	count:poisson
reg_alpha	L1 regularization term on weights	0
reg_lambda	L2 regularization term on weights	1
scale_pos_weight	It controls the balance of weights	1
subsample	Subsample ratio of the training instances	1

Source: Developed by the author

4.2 CONSTRUCTING A MULTI-LAYER PERCEPTRON

During the training phase of the MLP models, a random search with 50 iterations was performed with a range of values detailed in Table 8. The optimizer used was "Adam", which is an extension to stochastic gradient descent, and the loss function was "Poisson". The maximum number of epochs is 5000, but the callback "early stopping", with a value of fifteen epochs, was used to end training if the validation set loss had stopped improving.

Table 8 - MLP hyperparameters Settings

Hyperparameter	Values
Neurons first layer	[300, 400]
Neurons second layer	[420, 520]
Neurons third layer	[520, 620]
Neurons fourth layer	1
Epochs	5000
Batch size	[60, 80]
Learning rate	[0.0001, 0.008]

4.3 CONSTRUCTING A LONG SHORT-TERM MEMORY NEURAL NETWORK

In order to model with LSTM, the hyperparameters were chosen within a range as described in Table 9, using "Adam" as optimizer and "Poisson" as a loss function. Fifty iterations were performed in a random search to discover which combination of hyperparameters produce the best model. The maximum number of epochs was 5000, but the "early stopping" method was set to fifteen epochs ,i.e., if after fifteen epochs the loss function of validation set had not improved, the LSTM stopped.

Table 9 - LSTM hyperparameters Settings

Hyperparameter	Values
Units first layer	[3, 10]
Units second layer	[50, 70]
Units third layer	[50, 100]
Units fourth layer	1
Epochs	5000
Batch size	[55, 160]
Learning rate	[0.000001, 0.001]

Source: Developed by the author

4.4 METRICS

The metrics used to measure the performance of the models are described below:

- **RMSE:** The Root Mean Square Error calculates the standard deviation of the errors obtained during the predictions, highlighting the variance of the frequency distribution of errors by taking their square before they are averaged. A lower value is better (GÉRON, 2017).

$$RMSE = \sqrt{\frac{1}{m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)})^2} \quad (25)$$

where:

m is the number of instances.

$x^{(i)}$ are the predictors vector of i^{th} instance.

h is the predicting function.

$y^{(i)}$ is the label of i^{th} instance.

- **MAE:** The Mean Absolute Error evaluates the average magnitude of errors' predictions, where each error contributes in proportion to the scoring rule. A lower value is better (GÉRON, 2017).

$$MAE = \frac{1}{m} \sum_{i=1}^m | (h(x^{(i)}) - y^{(i)}) | \quad (26)$$

where:

m is the number of instances.

$x^{(i)}$ are the predictors vector of i^{th} instance.

h is the predicting function.

$y^{(i)}$ is the label of i^{th} instance.

- **ACC0E:** It is the accuracy of the predictions' results with a margin of error 0, i.e. percentage of exact predictions.

$$ACC0E = \left(\frac{\#predictions_margin_0}{\#total_predictions} \right) 100\% \quad (27)$$

- **ACC1E:** It is the accuracy of the predictions' results with a margin of error equal or less than 1.

$$ACC1E = \left(\frac{\#predictions_margin_1}{\#total_predictions} \right) 100\% \quad (28)$$

- **ACC2E:** It is the accuracy of the predictions' results with a margin of error equal or less than 2.

$$ACC2E = \left(\frac{\#predictions_margin_2}{\#total_predictions} \right) 100\% \quad (29)$$

4.5 EXPERIMENTAL RESULTS AND DISCUSSION

In this section, the eighteen results are presented and analyzed. Table 10, 11 and 12 present results for XGBoost, MLP and LSTM models with the two types of reduction techniques, respectively. It was tested different past hours: three, twelve and twenty-four to discover which one improves the prediction of the number of interventions that firefighters will face in the next one hour. The best performance is marked in bold considering the accuracy metrics: ACC0E, ACC1E and ACC2E as the most representative in real life. In this context, the best XGBoost

model has a MAE of 1.6787, corresponding to twenty-four past hours with MIR reduction method, while for MLP and LSTM the MAE is 1.7325 with three past hours and 1.6960 with twenty-four past hours, respectively, both with MIR reduction method.

Table 10 - XGBoost prediction results for next one hour on test data 2017

Method	Past hours	RMSE	MAE	ACC0E (%)	ACC1E (%)	ACC2E (%)
MIR	3h	2.3013	1.6942	20.64	55.62	76.75
	12h	2.3020	1.6895	20.97	55.79	76.91
	24h	2.2815	1.6787	21.03	55.90	77.03
PCA	3h	2.4789	1.8644	17.58	49.98	73.50
	12h	2.4408	1.8182	18.55	52.12	74.36
	24h	2.3520	1.7534	19.09	53.52	75.72

Source: Developed by the author

Table 11 - MLP prediction results for next one hour on test data 2017

Method	Past hours	RMSE	MAE	ACC0E (%)	ACC1E (%)	ACC2E (%)
MIR	3h	2.3413	1.7325	19.97	54.34	76.05
	12h	2.3810	1.7523	19.82	54.10	75.88
	24h	2.3899	1.7587	19.90	54.46	75.13
PCA	3h	2.4526	1.8021	19.85	53.85	74.18
	12h	2.3945	1.7974	18.79	50.79	75.41
	24h	2.3879	1.7621	20.16	53.96	74.94

Source: Developed by the author

Table 12 - LSTM prediction results for next one hour on test data 2017

Method	Past hours	RMSE	MAE	ACC0E (%)	ACC1E (%)	ACC2E (%)
MIR	3h	2.3237	1.6989	20.99	56.04	76.55
	12h	2.3088	1.7010	20.59	55.29	76.71
	24h	2.2838	1.6960	20.06	55.05	76.96
PCA	3h	2.3586	1.7557	19.40	53.79	75.31
	12h	2.3639	1.7416	19.77	54.97	75.94
	24h	2.3390	1.7290	20.08	54.69	76.14

Source: Developed by the author

Figure 25a and Figure 25b illustrate XGBoost results with MIR, Figure 25c and Figure 25d show XGBoost results with PCA. The right images exhibit 200 samples with 3 past hours trying to predict one hour horizon and the left images demonstrate a histogram with the number of predictions per error from 0 to 30 errors. The same definitions are applied to Figure 26 to Figure 33 changing the corresponding technique with both reduction methods and the number of past hours specified. Considering a total of 8760 samples from the testing set (year 2017) and a margin of error equal or less than 2 (ACC2E), the best XGBoost model achieved 6748 correct predictions, which represents 77.03% of accuracy. The best MLP model got 6662 correct predictions which means 76.05% of accuracy. And the best LSTM model predicted correctly 6742 samples which depicts 76.96% of accuracy. In contrast with ACC0E, predictions with a

margin of error zero, the best model obtained 21.03% and it was with XGBoost technique and MIR method.

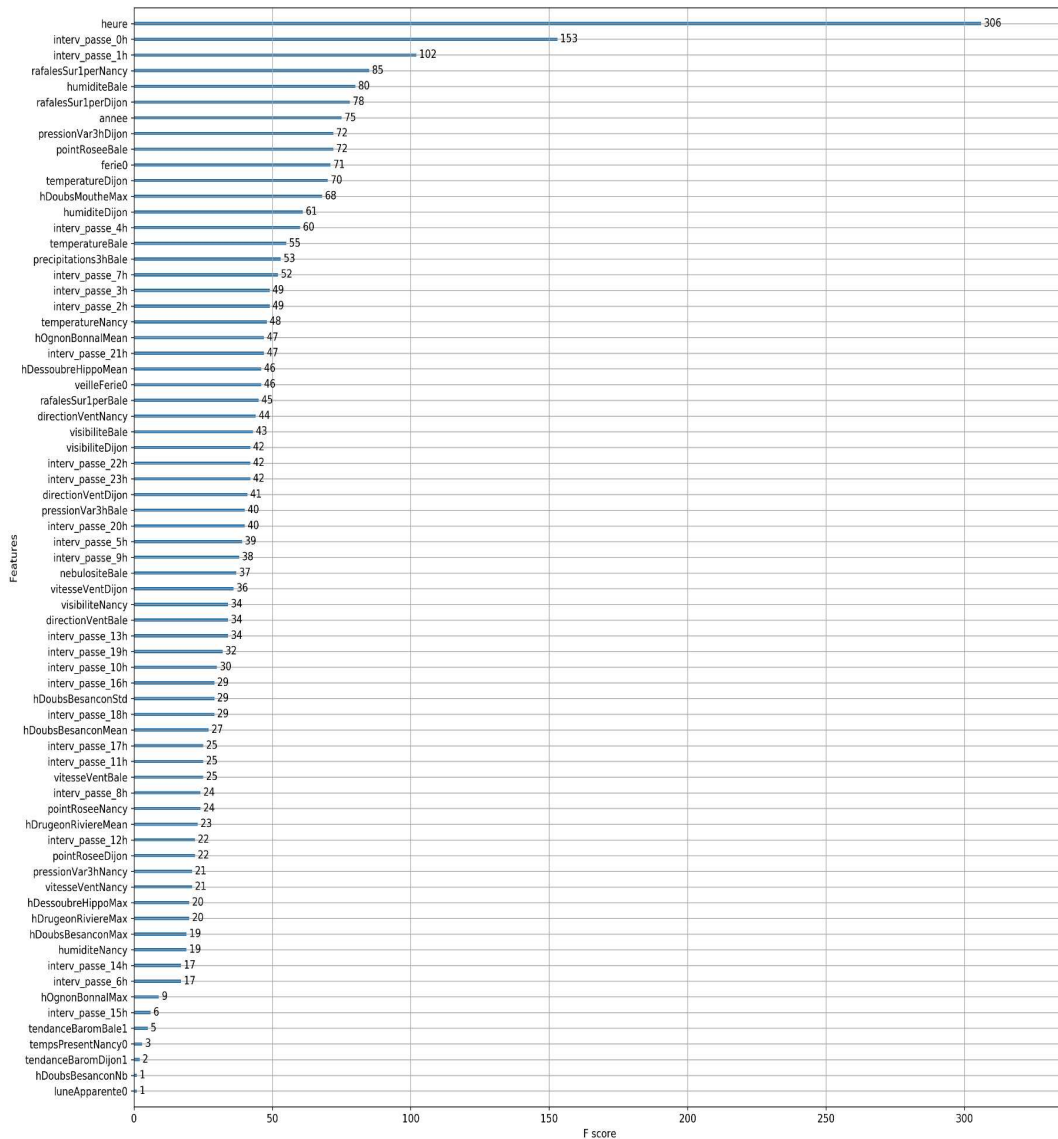
The best model for XGBoost was achieved with a learning rate: 0.034999, max_depth: 4, n_estimators: 212 and colsample_bytree: 1, with 24 past hours. In Figure 21, it is exhibited the importance value of each feature reported by XGBoost. This operation is based on counting the number of times a feature is used to split the data across all trees without considering zero-importance feature (CHEN; HE; KHOTILOVICH, 2016). As an example, in Figure 22, it is showed the splits made in the first tree built. The ten first features with higher score are: hour (heure), number of interventions in t-1 (interv_passe_0h), number of interventions in t-2 (interv_passe_1h), bursts reported from Nancy station (rafalesSur1perNancy), humidity reported by Basilea station (humiditeBale), bursts reported from Dijon station (rafalesSur1perDijon), year (annee), average pression in the last three hours reported by Dijon station (pressionVar3hDijon), dew point reported by Basilea station (pointRoseeBale) and holidays (ferie0). Most of them make reference to meteorological data, followed by the time, the two previous number of interventions and if the day is a holiday or not. Therefore, it can be deduced that by adding more representative weather variables, the models could have better precision. What is more, detecting and including other kinds of variables with higher importance would also enhance further the forecast.

Figures 23 and 24 show the architectures of the best models found for MLP and LSTM taking three and twenty-four time steps respectively, both generated by Keras library. The MLP architecture consists of four dense layers, after each one a ReLu activation function is applied and a dropout rate of 0.5 to avoid overfitting and gradient vanishing. The first layer has 359 neurons, the second one 448 neurons, the third one 527 neurons and the last only 1 neuron as output. All trained with a batch size of 63 and a learning rate of 0.0017443359. In the case of LSTM architecture, the best is defined with 4 layers, the first three are LSTM layers with 9, 60, 87 neurons and the last is a dense layer with 1 output neuron. The dropout rate after the first layer was 0.2 and after the second was 0.5. It was not used dropout for the third layer and for the last the activation function was linear. The batch size and learning rate employed were 103 and 0.0000758669, respectively. It can be seen that the learning rate in neural networks is smaller compared to that used in XGBoost.

Finally, Figures 34 to 36 show the comparison of the three techniques, each one with the two reduction methods and detailed by the number of past hours: three, twelve and twenty-four. It can be observed that as the time steps increase the results for XGBoost and LSTM improved and they were very close to each other, maintaining the highest accuracies. Furthermore, it is noticed in Figure 34a, Figure 35a and Figure 36a that XGBoost technique is a little more robust in recognizing peaks of the interventions, conversely, MLP maintains a more constant pattern. It should be noted that the best results obtained were applying the Mutual Information Regression

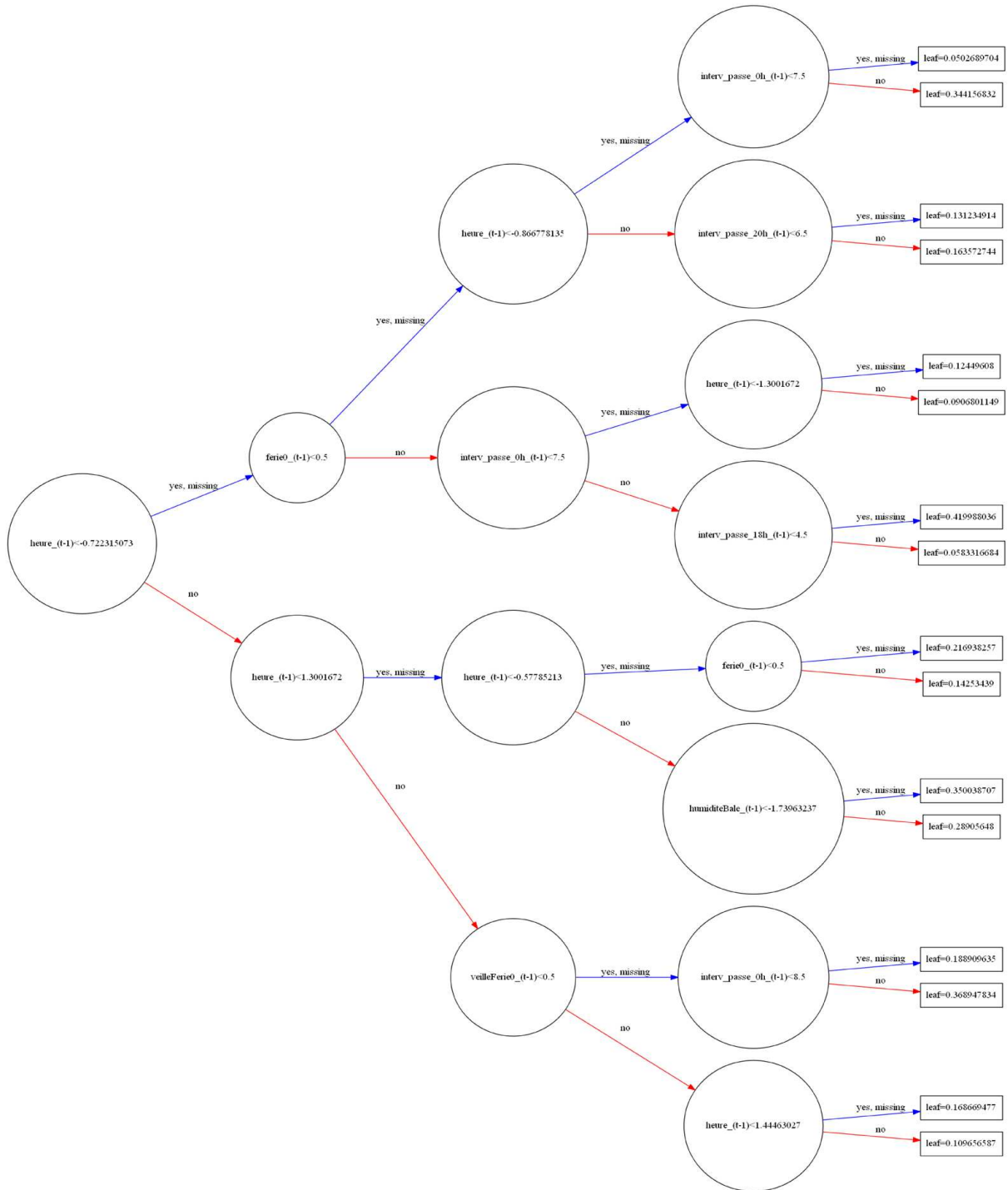
reduction method, where 56 features were used with the target. In average the time consuming of the random search with fifty iterations for LSTM and MLP were around 72 hours and 36 hours, respectively, while for XGBoost and its grid search with four validations per iteration from a total of 840 iterations were roughly 48 hours. Nevertheless, the use of deeper layers and more time steps could improve results in data generalization. This is also denoted in the literature, where LSTM is famous for resolving sequential data problems and MLP widely used for pattern recognition in several categories. On the other hand, the simplicity, robustness, less time consumption and less computational costs of XGBoost method are well appreciated in real-time applications. As it was examined in this research, the XGBoost works by optimizing with quick parallel tree construction in comparison with LSTM and MLP that increased complexity proportionally by expanding the number of layers and/or neurons.

Figure 21 - XGBoost - Feature importance



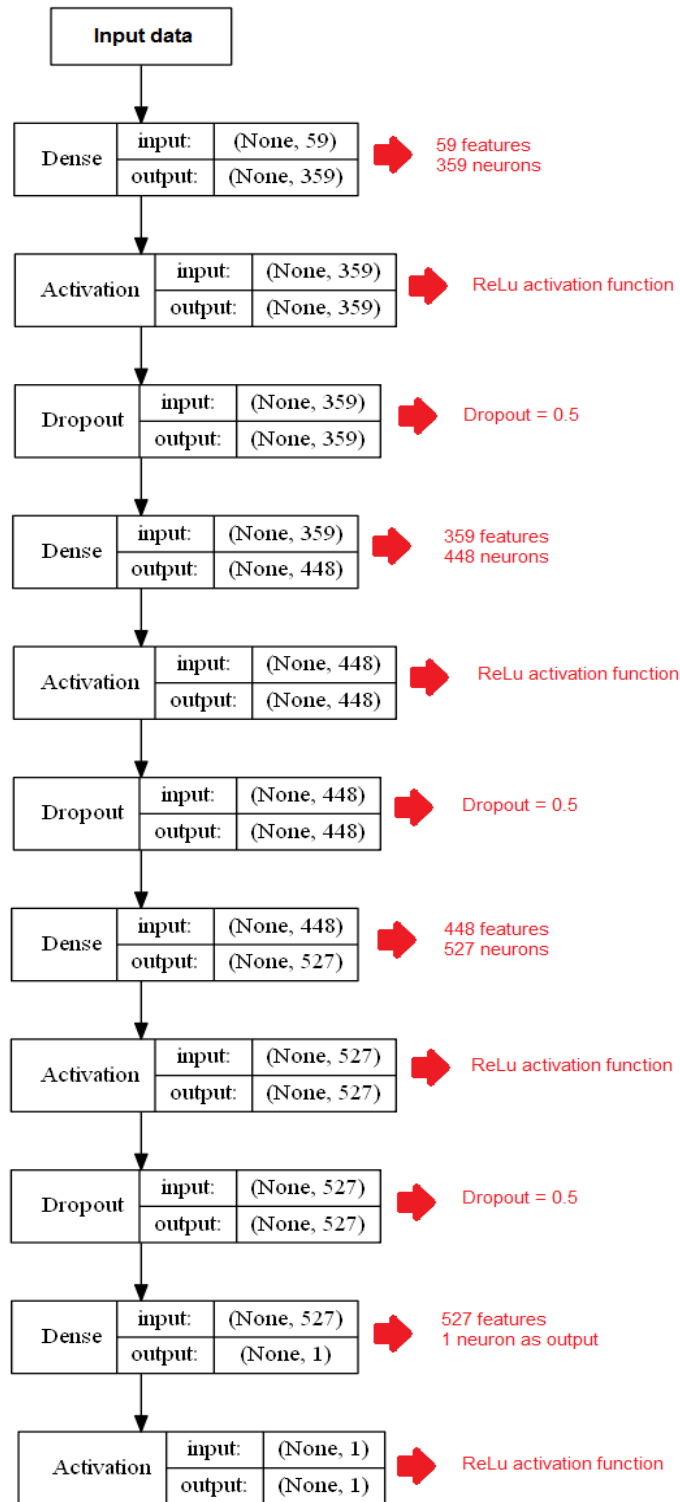
Source: From Author

Figure 22 - XGBoost - Tree construction



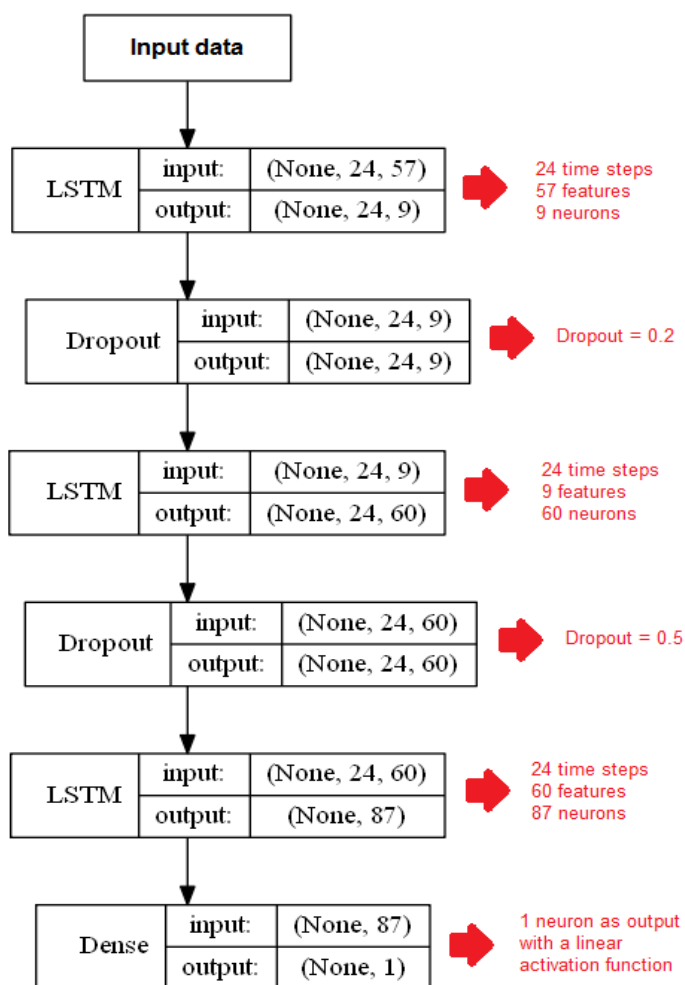
Source: From Author

Figure 23 - MLP model architecture with 3 past hours



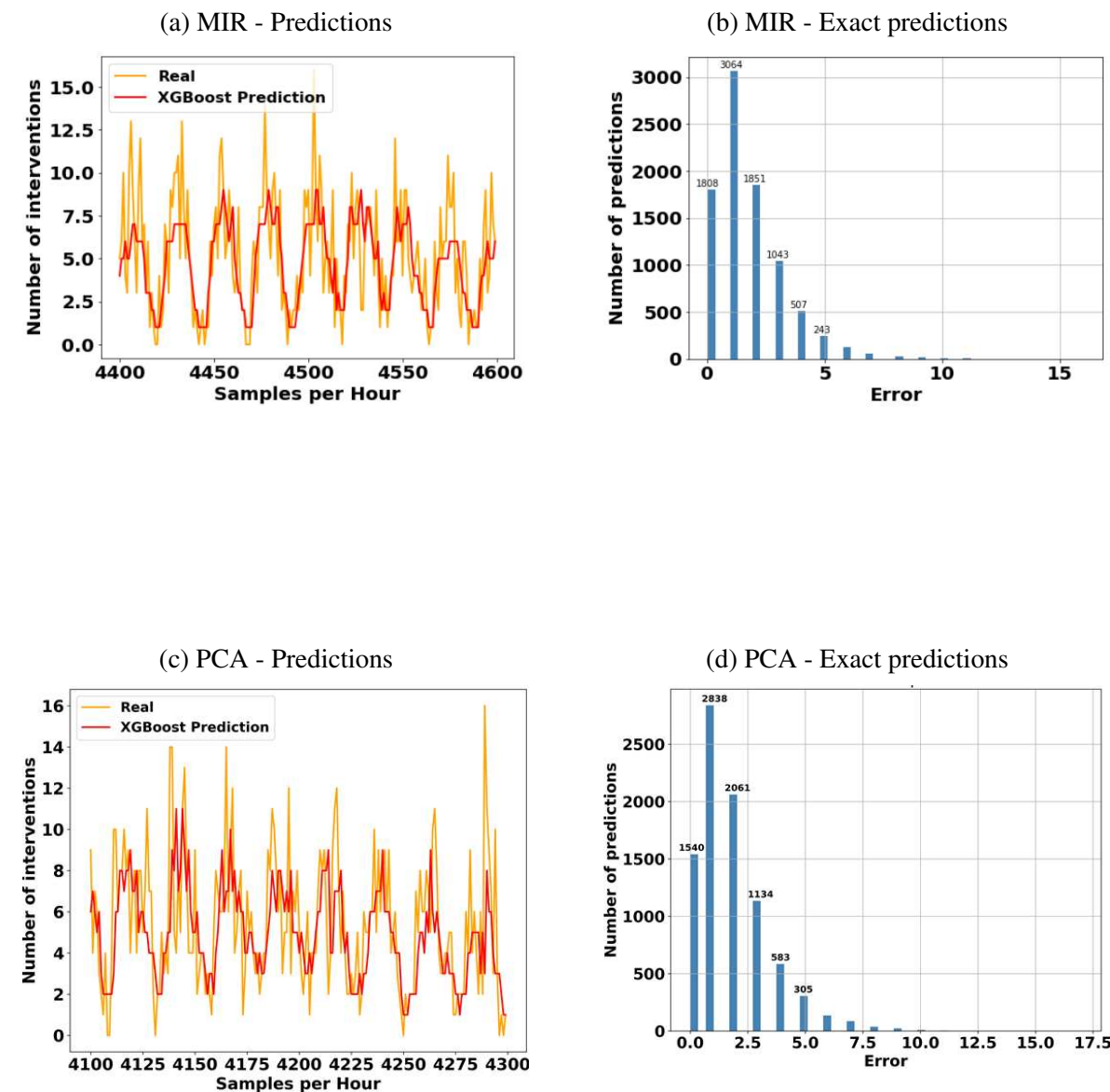
Source: Developed by the author

Figure 24 - LSTM model architecture with 24 time steps



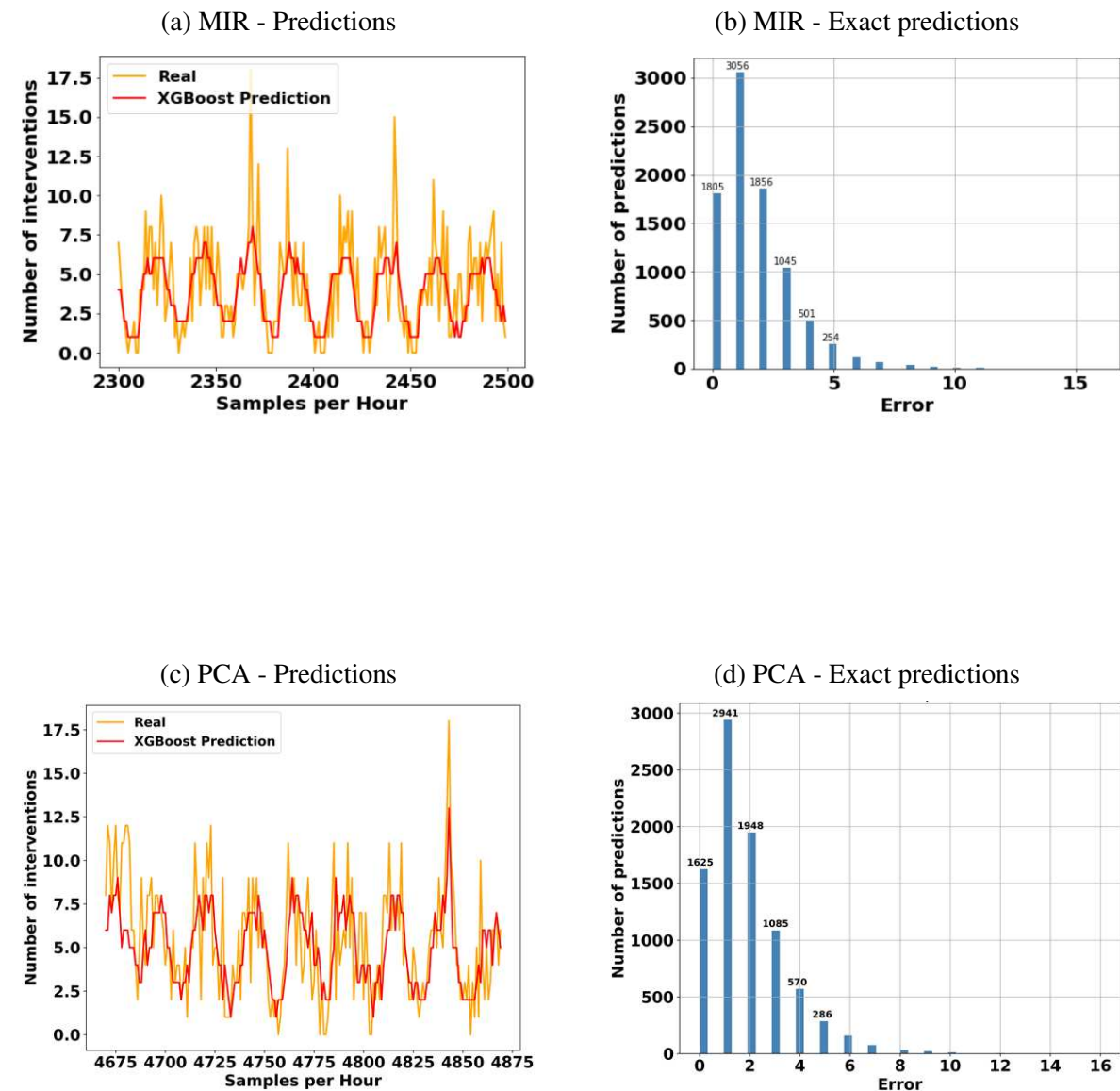
Source: Developed by the author

Figure 25 - XGBoost with three past hours for predicting one hour



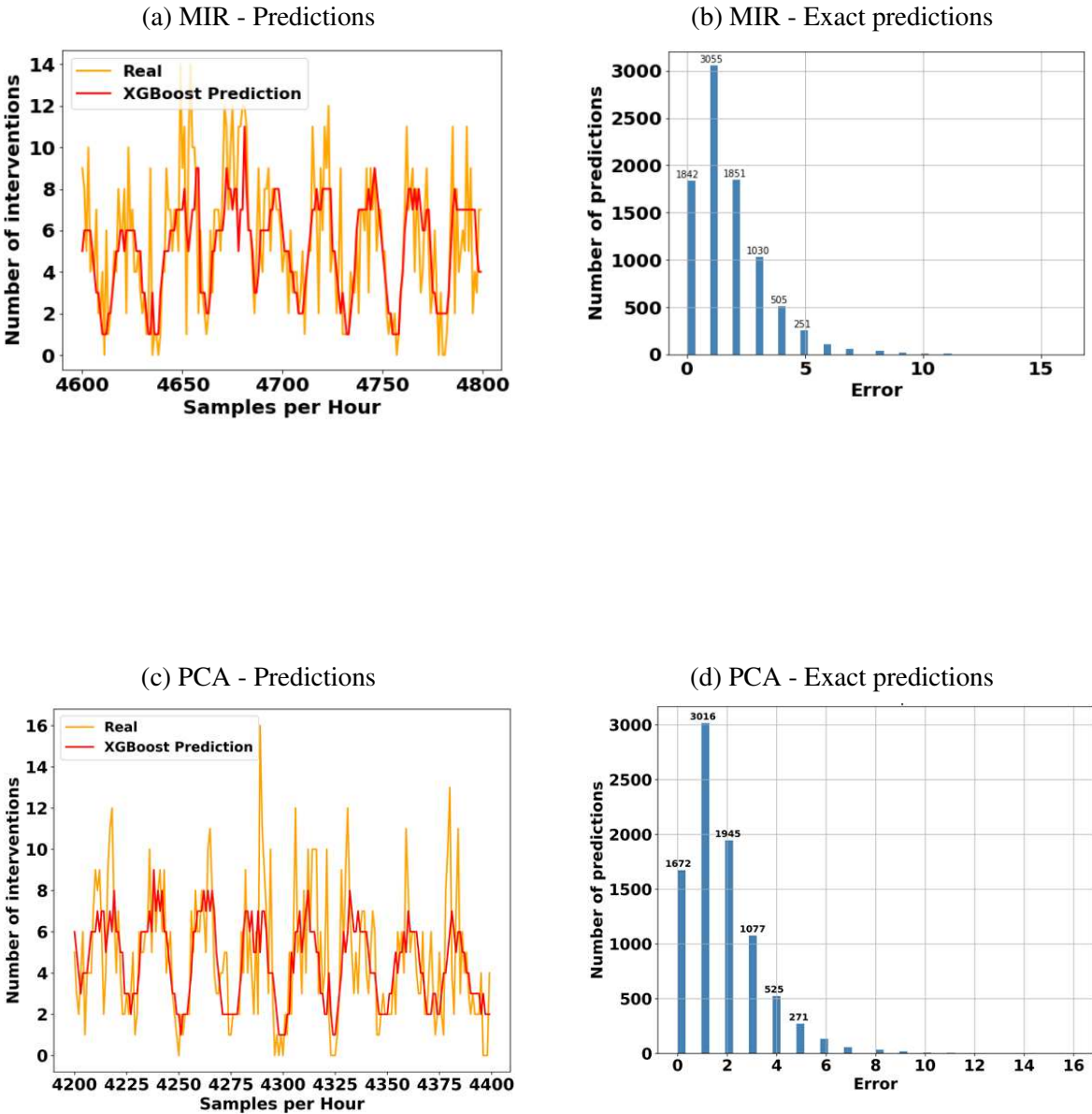
Source: Developed by the author

Figure 26 - XGBoost with twelve past hours for predicting one hour



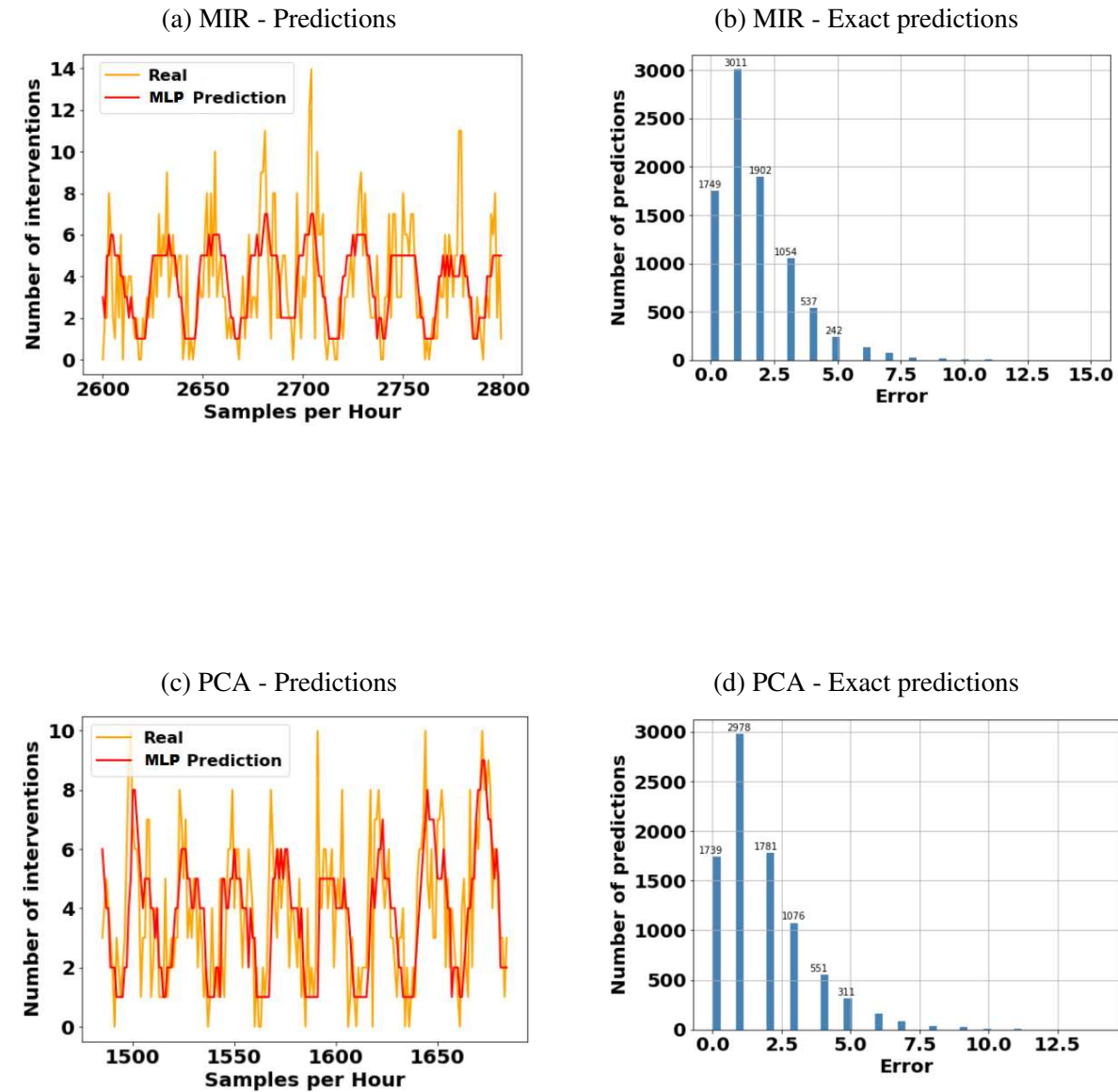
Source: Developed by the author

Figure 27 - XGBoost with twenty-four past hours for predicting one hour



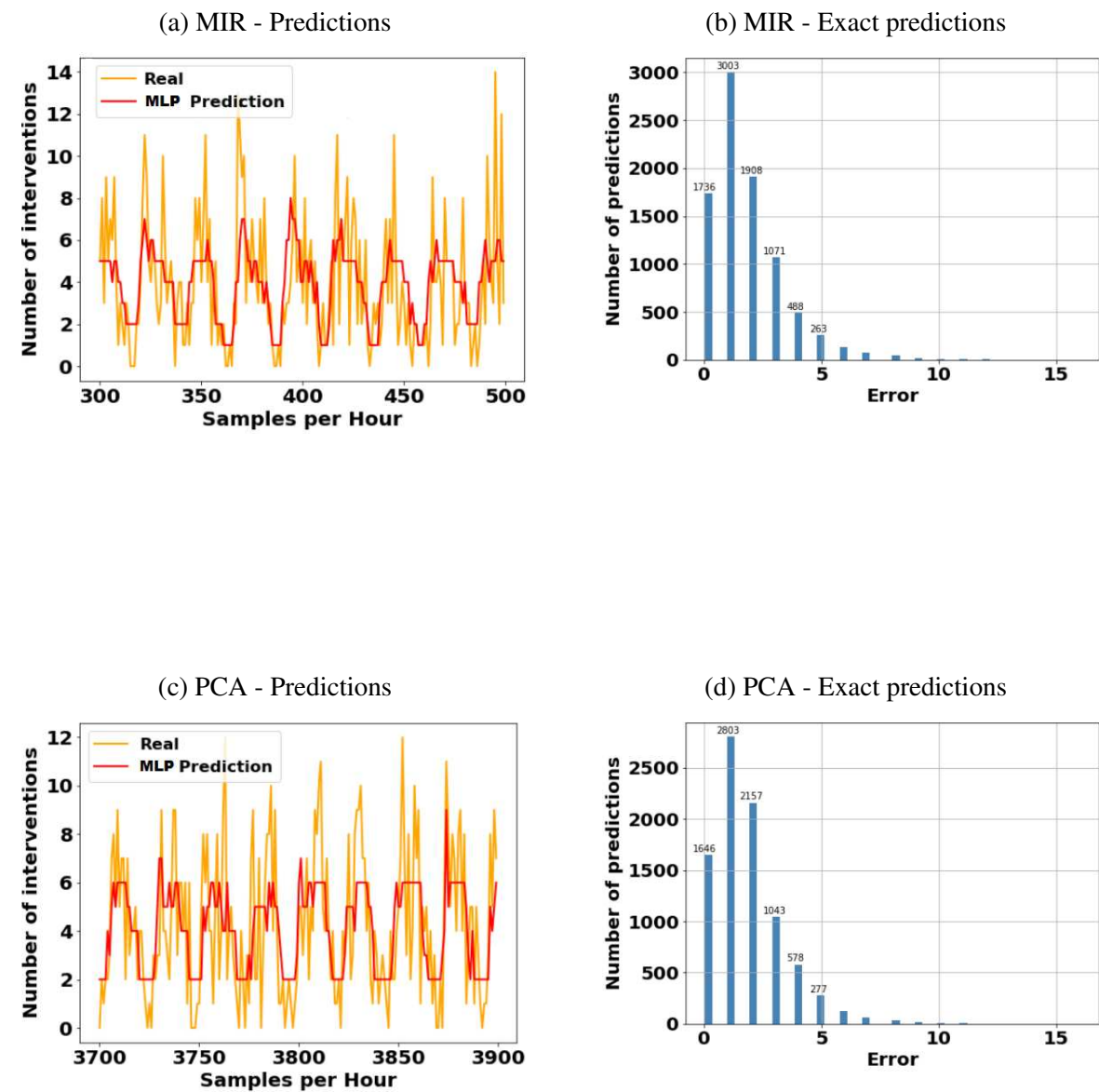
Source: Developed by the author

Figure 28 - MLP with three past hours for predicting one hour



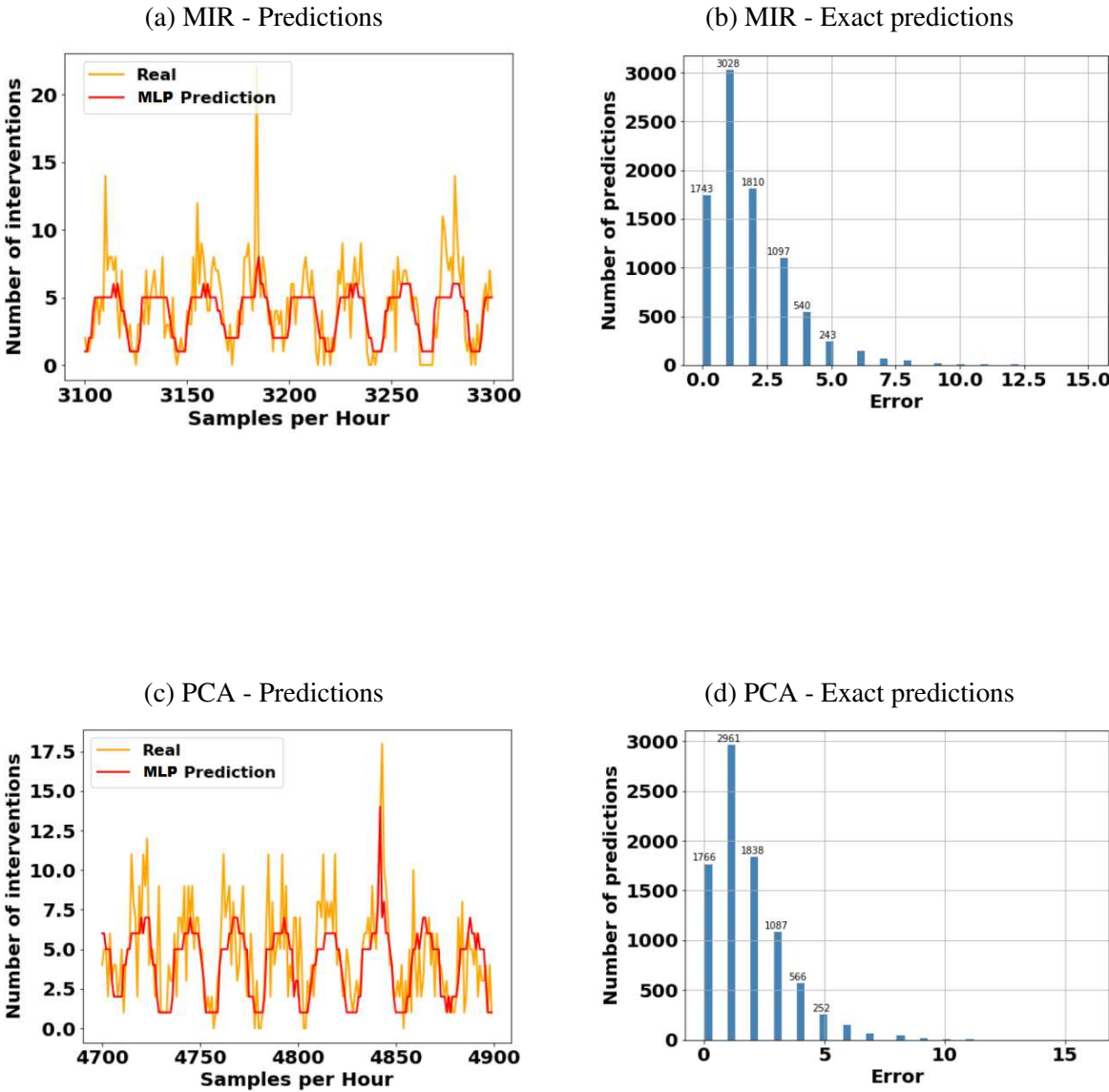
Source: Developed by the author

Figure 29 - MLP with twelve past hours for predicting one hour



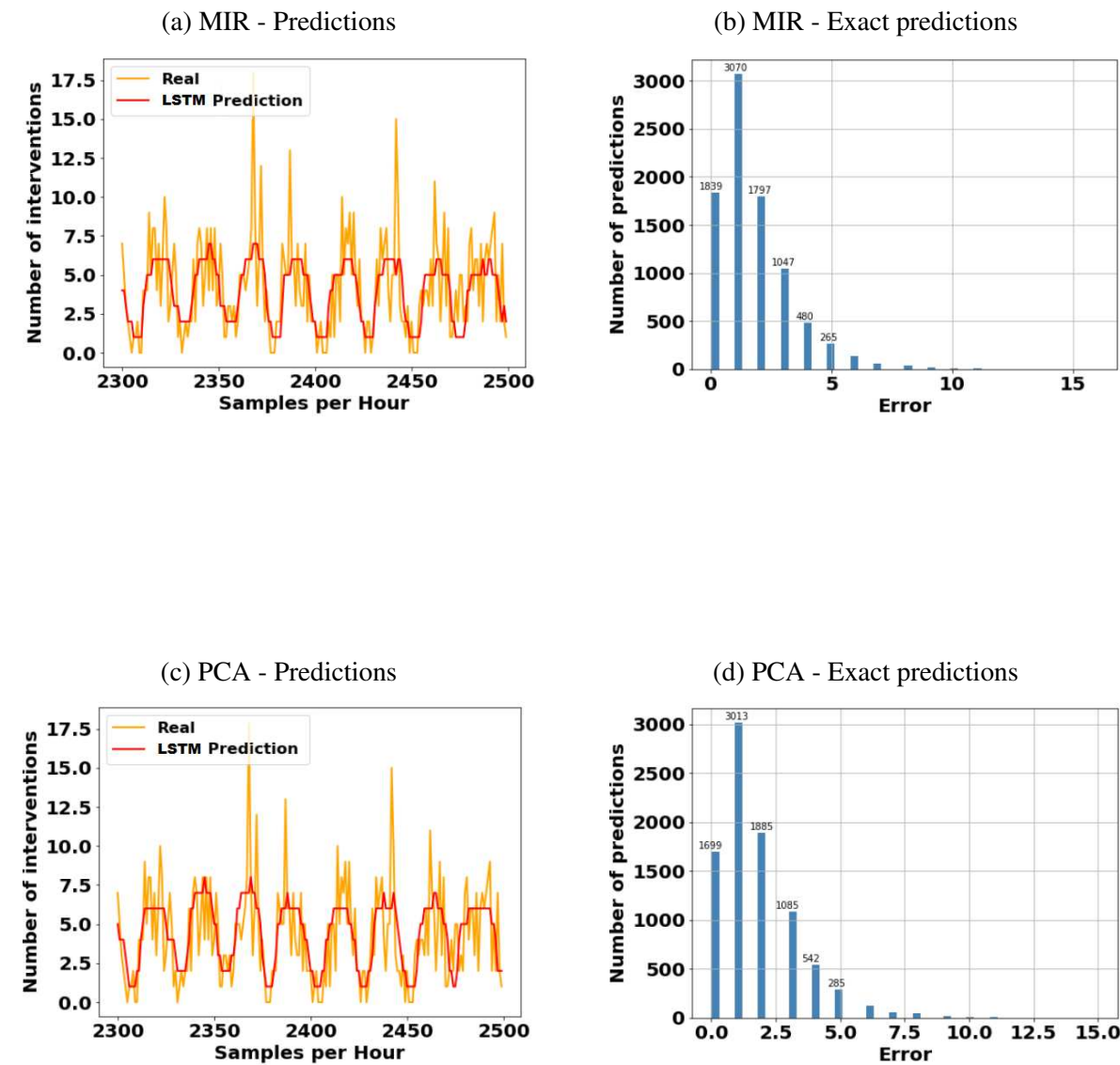
Source: Developed by the author

Figure 30 - MLP with twenty-four past hours for predicting one hour



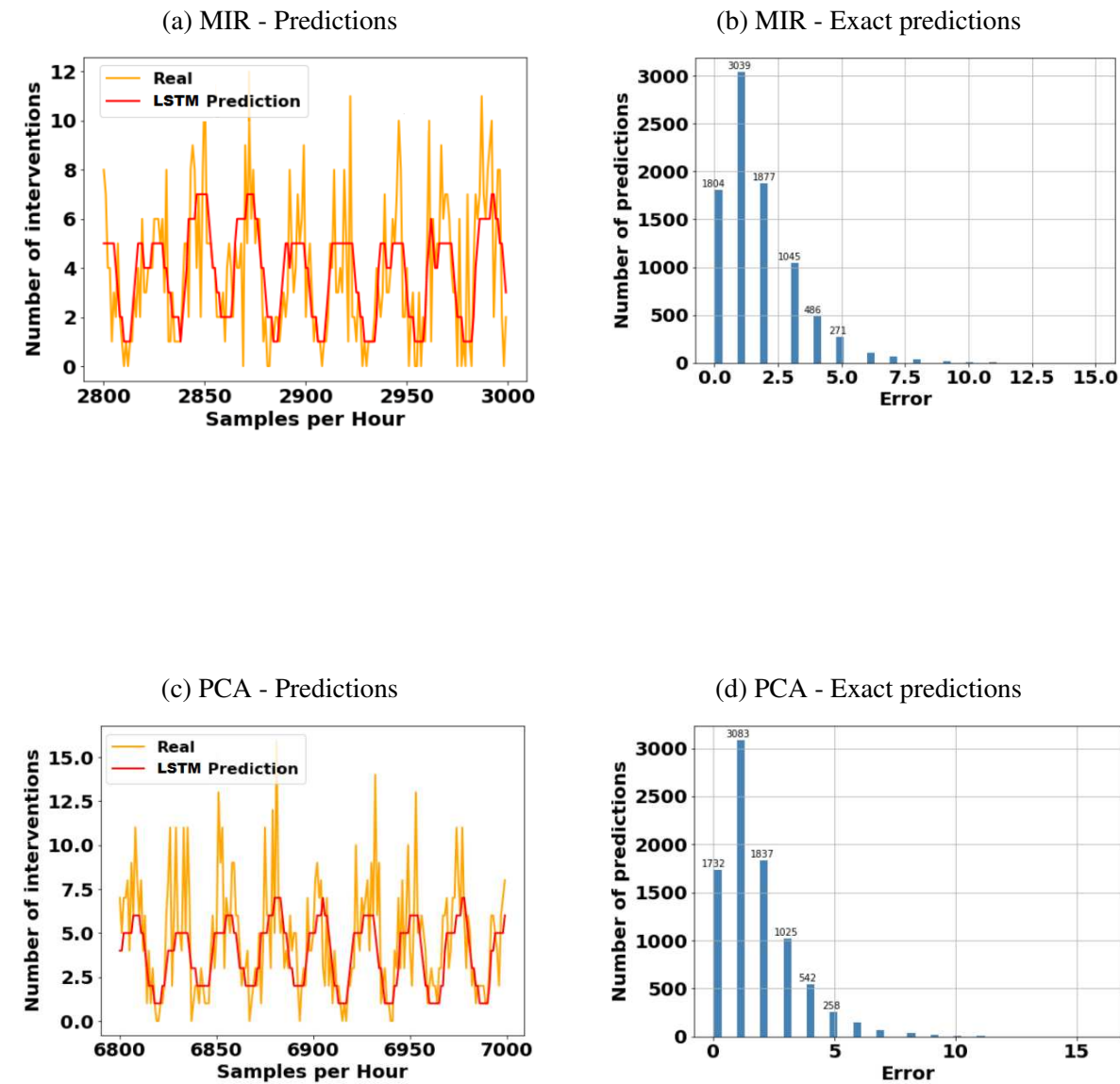
Source: Developed by the author

Figure 31 - LSTM with three past hours for predicting one hour



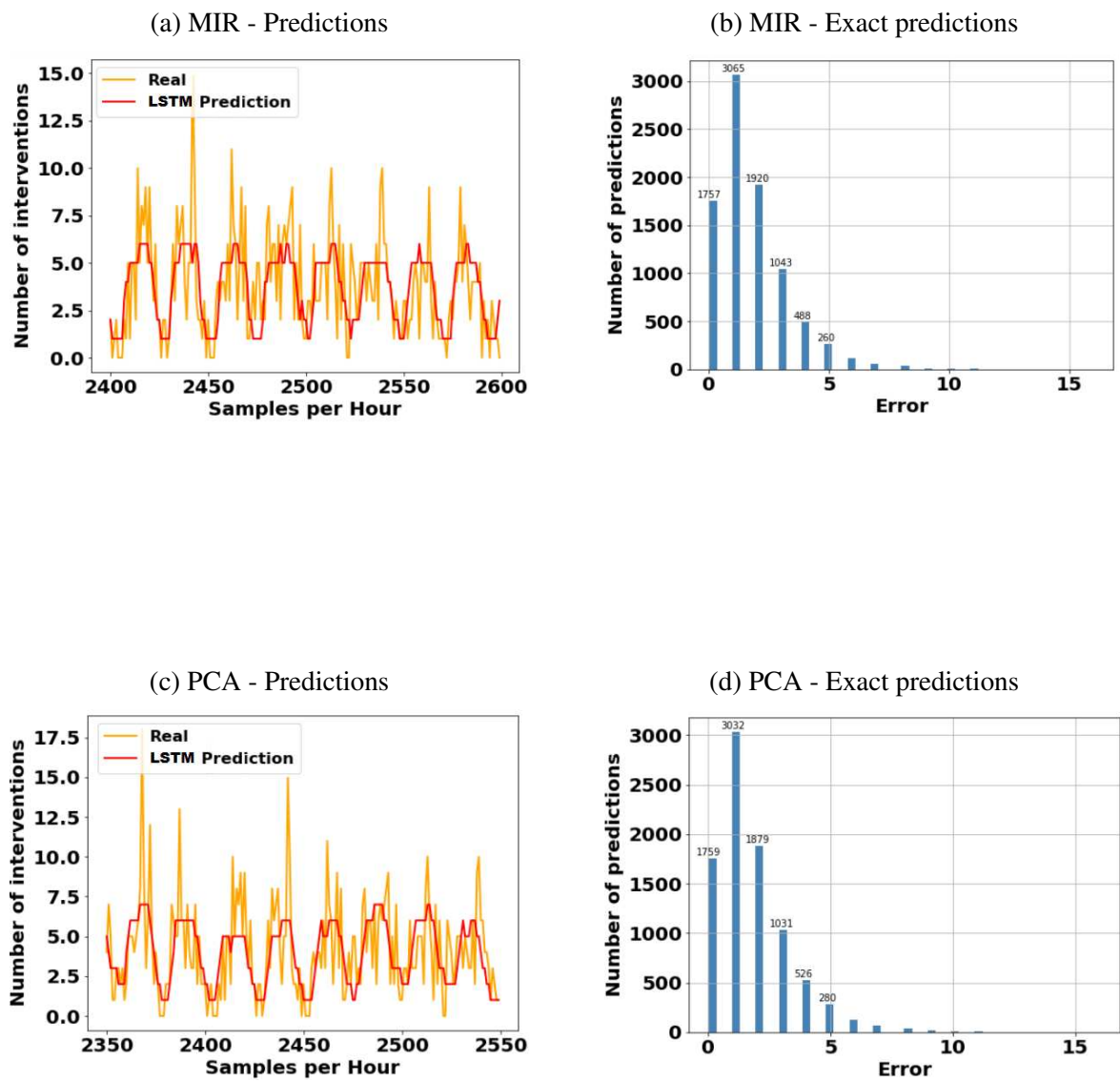
Source: Developed by the author

Figure 32 - LSTM with twelve past hours for predicting one hour



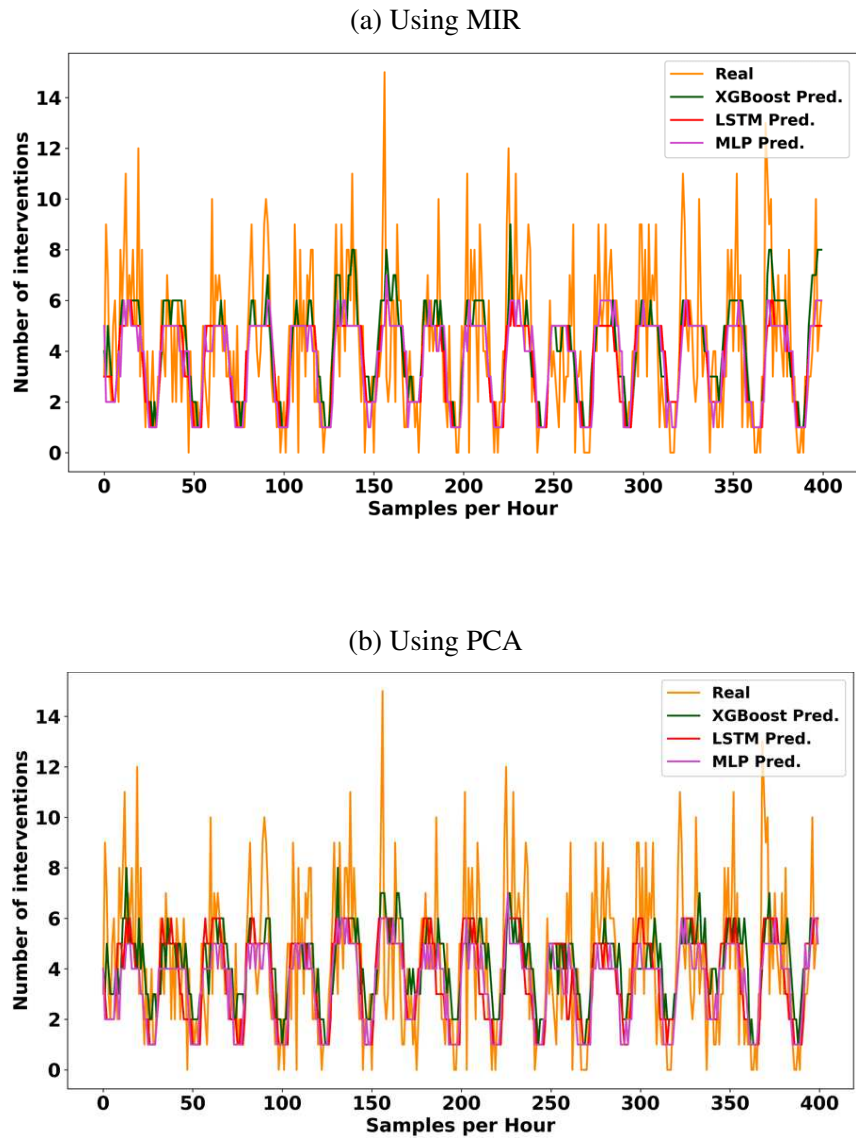
Source: Developed by the author

Figure 33 - LSTM with twenty-four past hours for predicting one hour



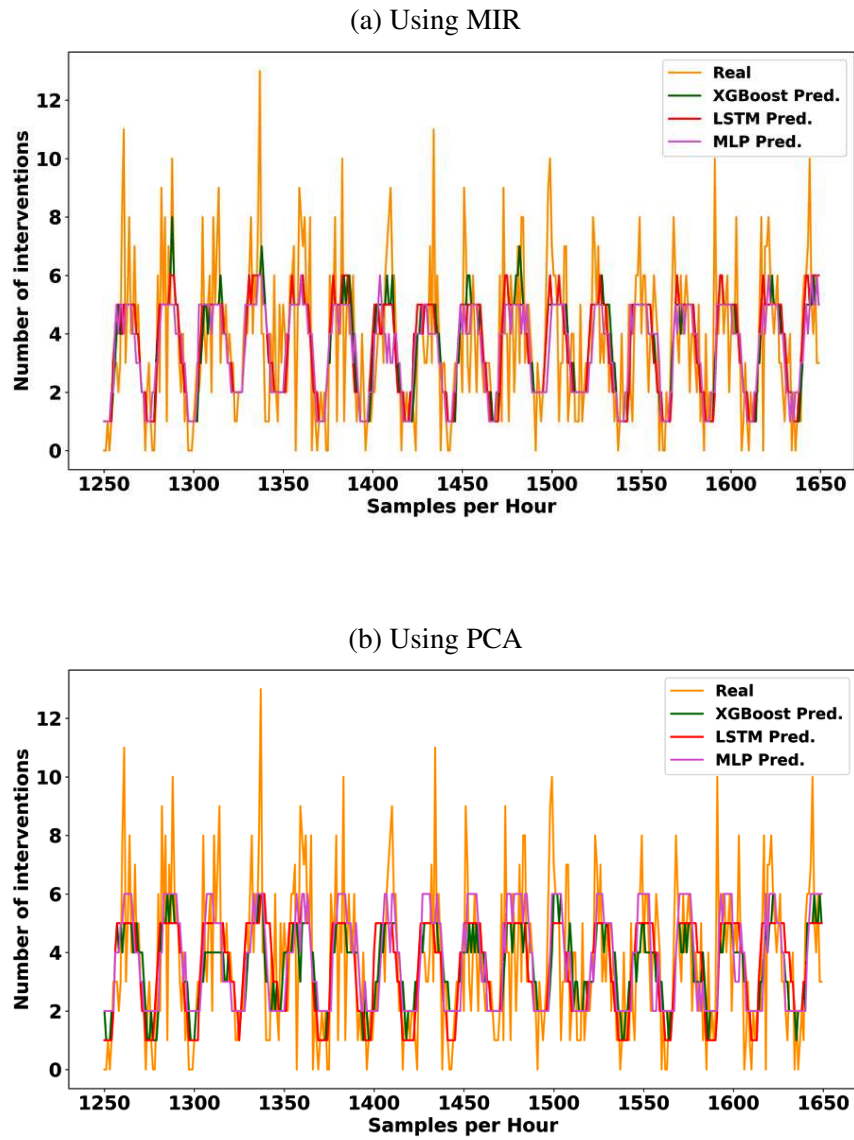
Source: Developed by the author

Figure 34 - XGBoost, MLP and LSTM with three past hours for predicting one hour



Source: Developed by the author

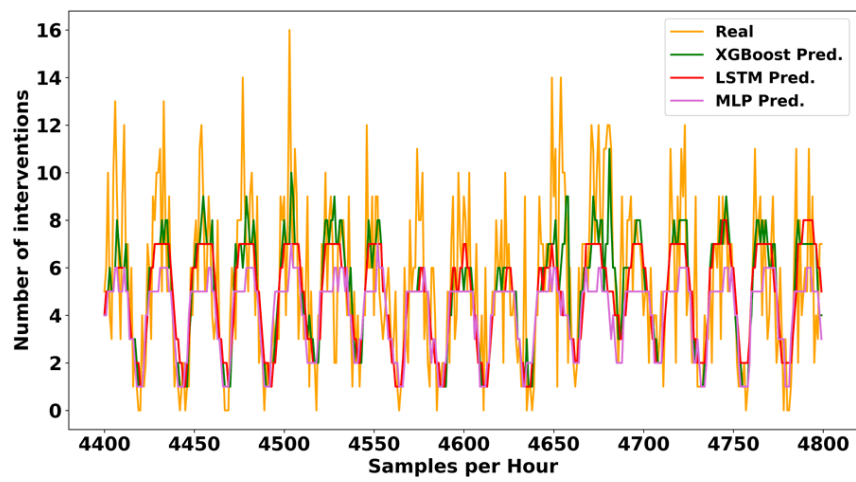
Figure 35 - XGBoost, MLP and LSTM with twelve past hours for predicting one hour



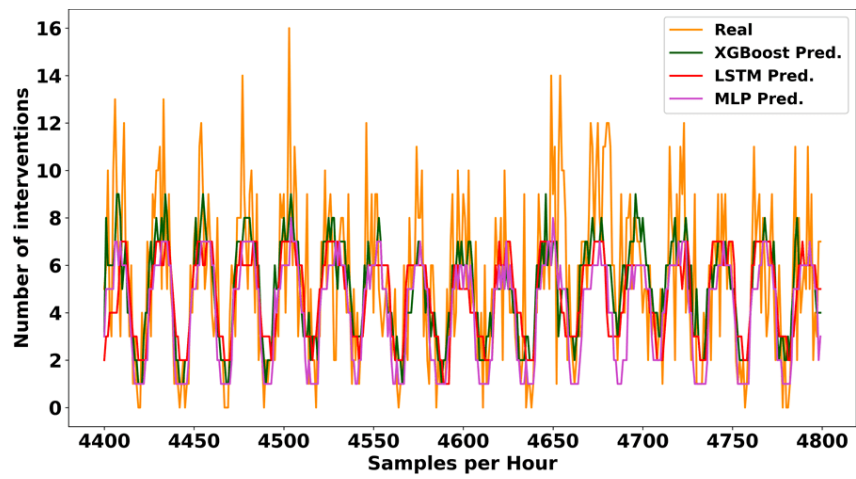
Source: Developed by the author

Figure 36 - XGBoost, MLP and LSTM with twenty-four past hours for predicting one hour

(a) Using MIR



(b) Using PCA



Source: Developed by the author

5 CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

5.1 CONCLUSIONS

The development of intelligent systems to predict the number of interventions at a given time could help fire brigades around the world to deal more efficiently with forthcoming incidents and to improve the management of human and mobile resources. This work presents three approaches of machine learning techniques applied successfully in the literature, i.e., the Extreme Gradient Boosting, the Multi-Layer Perceptron and the Long Short-Term Memory neural network, to predict the number of interventions of firefighters in the region of Doubs-France for the next one hour.

Models were fed with data collected during six years, provided by the fire brigade SDIS 25 located in Doubs-France and external variables such as weather, holidays, rivers height and moon phase. Before sending data to the training phase, they were previously analyzed to validate the frequency of the number of interventions. As a result, it was observed critical outliers in the set which in real life represented a natural disaster in the Doubs region. These outliers had to be smoothed considering the number of incidents registered the next day since the meteorological variables considered were not enough predictive to recognize this phenomenon, leaving the idea that by adding more significant weather variables this abnormal pattern could be identified and with others the final results could be improved. Furthermore, after standardizing and categorizing the data, the 143 initial variables were transformed into 831 features, resulting in a high-dimensional data set. In order to differentiate and gather the foremost characteristics from the irrelevant ones, speed up the training phase and avoid the random predictions owing to the noise caused by the unimportant features, MIR and PCA methods were tested in parallel.

The analysis of the models' results revealed that the three techniques (XGBoost, MLP and LSTM) are trying to find a sequence pattern for the regression problem using a different quantity of past hours to predict one future hour, taking in consideration that the number of attended incidents by firefighters incremented over the 6 years what affects the pattern recognition with the variables considered. The LSTM and XGBoost models showed better performance. XGBoost tries to reach slightly more the peaks of higher numbers of interventions during the training and this can be observed in the compared results. Furthermore, in most cases, it can be seen that as the number of time steps increases, performance improves. This might suggest that testing with more than twenty-four hours, better results could be achieved. On the other side, according to the time and resource consumption, LSTM used more memory and twice as much training time as MLP executing fifty iterations during the random search of the best model.

In comparison with XGBoost, that performed a grid search with four validations per iteration (cross-validation) from a total of 840 iterations, its memory consumption was lower and the training time was greater than MLP and less than LSTM, proving that XGBoost was more swift and effective than LSTM and MLP.

As regards the methods applied for dimensionality reduction, it was observed that PCA achieved good results with LSTM and twenty-four past hours. Nevertheless, MIR allowed obtaining more precision with the three modelling techniques as it is demonstrated in the percentage of exact predictions and with a margin of error equal or less than 1. The most outstanding combination of techniques is XGBoost with MIR, followed by LSTM with MIR and finally, MLP with MIR. Notwithstanding, better results can be achieved concentrating more efforts on the tuning procedure and on arranging features.

Finally, it is demonstrated that with the use of the developed machine learning techniques, it can be possible forecast firemen interventions with a reasonable accuracy showing its feasibility for practical purposes such as a decision making assistant, i.e., human experience working together with an artificial intelligence system, it ought to improve results in real applications.

5.2 SUGGESTIONS FOR FURTHER RESEARCH

For future works, it would be fascinating explore more recurrent architectures like Gated Recurrent Unit (GRU), test other different machine learning methods such as Deep Neural Decision Forest which consists in a combination of Deep Convolutional Neural Network and Decision Trees, analyze the importance and distribution of each variable considering to add new ones and apply other feature selection techniques to avoid redundant information and decrease computation time. Another attractive direction is to develop techniques capable of forecasting the type and location of the interventions and integrate them to the prediction of the number of interventions to work like this as a toolchain.

5.3 SCIENTIFIC PUBLICATIONS

During this research, one scientific contribution were published in the Sixth (6th) International Conference on Control, Decision and Information Technologies (CoDIT 2019). And a second one will be submitted to the Forty-sixth International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2020).

CERNA, S. L. Ñ.; GUYEUX, C.; ARCOLEZI, H. H.; COUTURIER, R.; ROYER, G.; LOTUFO, A. D. P. Long Short-Term Memory for Predicting Firemen Interventions. In: 6th International Conference on Control, Decision and Information Technologies (CODIT). CoDiT, 2019.

CERNA, S. L. Ñ.; GUYEUX, C.; ARCOLEZI, H. H.; COUTURIER, R.; ROYER, G.; LOTUFO, A. D. P. A Comparison of Long Short-Term Memory and Extreme Gradient Boosting Techniques for Predicting Firemen Interventions. In: 46th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM). SOFSEM, 2020.

Furthermore, the author took part as co-author in a scientific article published in the previously mentioned CoDIT 2019 conference as:

ARCOLEZI, H.H.; NUNES, W. R. B. M.; CERNA, S. L. Ñ.; SANCHES, M. A. A.; TEIXEIRA, M. C. M.; CARVALHO, A. A. de. A RISE-based Controller Fine-tuned by an Improved Genetic Algorithm for Human Lower Limb Rehabilitation via Neuromuscular Electrical Stimulation. In: 6th International Conference on Control, Decision and Information Technologies (CODIT). CoDiT, 2019.

REFERENCES

- ABRAMOWITZ, M. **Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables**. New York: Dover Publications, 1974.
- BISHOP, C. M. **Neural Networks for Pattern Recognition**. Oxford: Oxford University Press, 1995.
- BREIMAN, L. Bagging predictors. **Machine Learning**, Hingham, v. 24, n. 2, p. 123–140, 1996.
- BURLUTSKIY, N.; PETRIDIS, M.; FISH, A.; CHERNOV, A.; ALI, N. An investigation on online versus batch learning in predicting user behaviour. *In: Research and Development in Intelligent Systems XXXIII*. New York: Springer International Publishing, 2016. p. 135–149.
- CHEN, T. **Introduction to Boosted Trees**. [S.l.: s.n.], 2014. Available in: <https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf>. Access on: 29 jan. 2019.
- CHEN, T.; GUESTRIN, C. XGBoost: A scalable tree boosting system. *In: INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING (SIGKDD)*, 22nd., USA. **Proceedings [...]** New York: ACM, 2016. p. 785–794.
- CHEN, T.; HE, T.; KHOTILOVICH, V. **XGBoost eXtreme Gradient Boosting**. [S.l.: s.n.], 2016. Available in: <https://github.com/dmlc/xgboost>. Access on: 12 feb. 2019.
- CHIO, C.; FREEMAN, D. **Machine Learning and Security: Protecting Systems with Data and Algorithms**. California: O'Reilly Media, 2018.
- CHOLLET, F. et al. **Keras**. [S.l.: s.n.], 2015. Available in: <https://keras.io>. Access on: 12 feb. 2019.
- DOMINGOS, P. A few useful things to know about machine learning. **Communications of the ACM**, New York, v. 55, n. 10, p. 78–87, 2012.
- FRANCE, M. **Météo France**. [S.l.: s.n.], 2019. Available in: <http://www.meteofrance.com/accueil>. Access on: 20 oct. 2018.
- FREUND, Y.; SCHAPIRE, R. E. A short introduction to boosting. *In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, 16th., New York. **Proceedings [...]** California: Morgan Kaufmann, 1999. p. 1401–1406.
- FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. Additive logistic regression: a statistical view of boosting. **Annals of Statistics**, Maryland, v. 28, p. 2000, 1998.
- FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. **The Annals of Statistics**, Maryland, v. 29, n. 5, p. 1189–1232, 2001.

- FRIEDMAN, J. H. Stochastic gradient boosting. **Computational Statistics & Data Analysis**, New York, v. 38, n. 4, p. 367–378, 2002.
- FUTÉ, B. **Bison Futé**. [*S.l.: s.n.*], 2015. Available in: <https://www.bison-fute.gouv.fr>. Access on: 20 oct. 2018.
- GARRETA, R.; MONCECCHI, G. **Learning scikit-learn: Machine Learning in Python**. Birmingham: Packt Publishing, 2013.
- GERBER, M. S. Predicting crime using twitter and kernel density estimation. **Decision Support Systems**, Oxford, v. 61, p. 115–125, 2014.
- GÉRON, A. **Hands-On Machine Learning with Scikit-Learn and TensorFlow**. California: O'Reilly Media, 2017.
- GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. **Journal of Machine Learning Research**, Massachusetts, v. 9, p. 249–256, 2010.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Massachusetts: MIT Press, 2016.
- GRAVES, A. **Supervised Sequence Labelling with Recurrent Neural Networks**. Berlin: Springer Berlin Heidelberg, 2012.
- GREFF, K.; SRIVASTAVA, R. K.; KOUTNIK, J.; STEUNEBRINK, B. R.; SCHMIDHUBER, J. LSTM: A search space odyssey. **Transactions on Neural Networks and Learning Systems**, New York, v. 28, n. 10, p. 2222–2232, 2017.
- GROSSE, I.; BERNAOLA-GALVÁN, P.; CARPENA, P.; ROMÁN-ROLDÁN, R.; OLIVER, J.; STANLEY, H. E. Analysis of symbolic sequences using the jensen-shannon divergence. **Physical Review E**, Maryland, v. 65, n. 4, 2002.
- GUPTA, A.; GUSAIN, K.; POPLI, B. Verifying the value and veracity of extreme gradient boosted decision trees on a variety of datasets. *In*: INTERNATIONAL CONFERENCE ON INDUSTRIAL AND INFORMATION SYSTEMS (ICIIS), 11th., China. **Proceedings [...]** New York: IEEE, 2016.
- HAYKIN, S. **Neural Networks: A Comprehensive Foundation (2nd Edition)**. New Jersey: Prentice Hall, 1998.
- HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, Massachusetts, v. 9, n. 8, p. 1735–1780, 1997.
- HYDRO. **Hydro**. [*S.l.: s.n.*], 2015. Available in: <https://www.hydro.eaufrance.fr>. Access on: 20 oct. 2018.
- JOHNSON, R.; ZHANG, T. Learning nonlinear functions using regularized greedy forest. **Transactions on Pattern Analysis and Machine Intelligence**, New York, v. 36, n. 5, p. 942–954, 2014.
- KALCHBRENNER, N.; DANIHELKA, I.; GRAVES, A. Grid long short-term memory. **CORR**, New York, abs/1507.01526, 2015.

- LONG, J. S. **Regression Models for Categorical and Limited Dependent Variables (Advanced Quantitative Techniques in the Social Sciences)**. California: SAGE Publications, 1997.
- MASON, L.; BAXTER, J.; BARTLETT, P.; FREAN, M. Boosting algorithms as gradient descent. *In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS*, 12nd., USA. **Proceedings [...]** Massachusetts: MIT Press, 1999.
- MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *In: Neurocomputing: Foundations of Research*. Massachusetts: MIT Press, 1988. p. 15–27.
- MCKINNEY, W. Data structures for statistical computing in python. *In: WALT, S. van der; MILLMAN, J. (ed.). PYTHON IN SCIENCE CONFERENCE (SCIPY 2010)*, 9th., USA. **Proceedings [...]** [S.l.: s.n.], 2010. p. 51 – 56.
- PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, Massachusetts, v. 12, p. 2825–2830, 2011.
- PETTET, G.; NANNAPANENI, S.; STADNICK, B.; DUBEY, A.; BISWAS, G. Incident analysis and prediction using clustering and bayesian network. *In: IEEE SMARTWORLD, UBIQUITOUS INTELLIGENCE & COMPUTING, ADVANCED & TRUSTED COMPUTED, SCALABLE COMPUTING & COMMUNICATIONS, CLOUD & BIG DATA COMPUTING, INTERNET OF PEOPLE AND SMART CITY INNOVATION (SMARTWORLD/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI)*, 2017, China. **Proceedings [...]** New York: IEEE, 2017.
- REN, H.; SONG, Y.; WANG, J.; HU, Y.; LEI, J. A deep learning approach to the citywide traffic accident risk prediction. *In: INTERNATIONAL CONFERENCE ON INTELLIGENT TRANSPORTATION SYSTEMS (ICIIS)*, 21st., USA. **Proceedings [...]** New York: IEEE, 2018.
- REPUBLICAIN, L. **Doubs : gros dégâts après les orages de la nuit**. [S.l.: s.n.], 2016. Available in: <https://www.estrepublikain.fr/faits-divers/2016/06/25/violents-orages-dans-le-doubs-de-nombreux-sinistres>. Access on: 20 oct. 2018.
- ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, Washington, p. 65–386, 1958.
- ROSS, B. Mutual information between discrete and continuous data sets. **PloS one**, California, v. 9, p. e87357, 2014.
- RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation. *In: Parallel Distributed Processing – Explorations in the Microstructure of Cognition*. Massachusetts: MIT Press, 1986. p. 318–362.
- SARKAR, S.; VINAY, S.; RAJ, R.; MAITI, J.; MITRA, P. Application of optimized machine learning techniques for prediction of occupational accidents. **Computers & Operations Research**, Oxford, 2018.

SENTINELLES, R. **INSERM/Sorbonne Université**. [*S.l.: s.n.*], 2007. Available in: <https://www.sentiweb.fr/?page=table>. Access on: 20 oct. 2018.

SHANNON, C. E. A mathematical theory of communication. **The Bell System Technical Journal**, New Jersey, v. 27, n. 3, p. 379–423, 1948.

SHI, X.; LI, Q.; QI, Y.; HUANG, T.; LI, J. An accident prediction approach based on XGBoost. *In: INTERNATIONAL CONFERENCE ON INTELLIGENT SYSTEMS AND KNOWLEDGE ENGINEERING (ISKE)*, 12th., China. **Proceedings [...]** New York: IEEE, 2017.

SJARDIN, B.; MASSARON, L.; BOSCHETTI, A. **Large Scale Machine Learning with Python**. Birmingham: Packt Publishing, 2016.

STRANG, G. **Introduction to Linear Algebra**. Massachusetts: Wellesley-Cambridge Press, 2016.