

Mestrado em Métodos Analíticos Avançados

Master Program in Advanced Analytics

Time Series Classification for Device Fingerprinting

Internship project at a telecommunications and
technology company

Ana Sofia Tavares Jordão Alves

Internship report presented as partial requirement for
obtaining the master's degree in Data Science and Advanced
Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

TIME SERIES CLASSIFICATION FOR DEVICE FINGERPRINTING

by

Ana Sofia Tavares Jordão Alves

Internship report presented as partial requirement for obtaining the master's degree in Data Science and Advanced Analytics

Advisor: *Prof. Doutor* Flávio Luis Portas Pinheiro

Co Advisor: Hock Ng, Ph.D.

September 2020

ACKNOWLEDGEMENTS

This report reflects the synergy among myself and different players I crossed paths within this journey. In this way, I could not leave this page blank without acknowledging each of them.

I would like to start by thanking my supervisor, Professor Flávio Pinheiro, for providing useful inputs and the required guidance to build this document. I should also extend my gratitude to all the professors I encountered in my educational path whose shared expertise somehow contributed to the conclusion of this cycle.

To my supervisor at the company, Hock Ng, who accepted this challenge and introduced me to the project and to the telecommunications field and was always available for any question I had despite the geographical distance. I appreciate the data science knowledge and supervision given to me throughout this period.

I would also like to thank my manager, Sumathi Mohanselman, as well as all the members of the team that welcomed me at the company, for the learning experience and the opportunity given to me. In particular, to my colleagues, Tatiane Espindola and Hamid Khosravani, for all the wise words, ideas and interest in this project.

Finally, to my parents, Ana and Mário, and my brother Luis, for the endless caring, continuous encouragement and support.

To each and every one of you – Thank you!

ABSTRACT

The telecommunication service providers seek an accurate insight into the devices that are connected within a home network, in order to provide a better in-home experience. In this way, the goal of the internship was to develop a machine learning model for fingerprinting of Amazon devices. This can be translated to a timeseries binary classification problem and assumes an exploration background of understanding the employment of bytes received by the router over time as an indicator of the internet usage to detect the Amazon devices. A feature-based analysis was conducted to make it possible to apply the most common and simple classifiers, which is relevant within a company context. The available data presented some challenges, namely a high imbalance and number of missing values. For this, it was studied several combinations of different techniques to increase the importance of the minority class and to impute the unknown values. In addition, multiple models were trained, whose results were evaluated and compared. The achieved performance of the best model was not considered satisfactory to correctly identify the Amazon devices, which lead to the conclusion that other approaches, algorithms and/or variable(s) need to be considered in a future iteration. The project contributed to a better understanding of the path to take on the identification of the devices and introduced new approaches and reasoning when dealing with similar data as the timeseries in analysis.

KEYWORDS

Machine Learning; Telecommunications; Time Series; Binary Classification; Supervised Learning; Feature-Based Analysis; Imbalanced Data; Decision Trees; Ensemble Methods; Random Forest; AdaBoost; XGBoost; Logistic Regression; SVMs

INDEX

1. Introduction.....	1
1.1. Company Overview	2
1.2. The Team and Activities	2
1.3. Internship Goals.....	3
2. Theoretical Framework	5
2.1. Machine Learning	5
2.1.1. Categories of Machine Learning Problems	5
2.1.2. Overview of the Machine Learning Process.....	6
2.1.3. Data Preparation	6
2.1.4. Feature Engineering	7
2.1.5. Imbalanced Dataset.....	7
2.1.6. K-fold Cross Validation	9
2.1.7. Machine Learning Algorithms	9
2.1.8. Evaluation Measures	18
2.2. Time Series	19
2.2.1. Feature-Based Analysis	20
3. Tools and Technology.....	24
3.1. Tsfresh	24
3.2. Xgboost.....	25
4. The Project.....	26
4.1. Timeline	26
4.2. Device Fingerprinting Project	26
4.2.1. Motivation	28
4.2.2. Data Description	28
4.2.3. Model and Analysis	30
4.2.4. Results and Discussion	33
4.2.5. Conclusions.....	37
5. Conclusions.....	38
5.1. Connection to the Master Program	38
5.2. Internship Evaluation	38
5.3. Limitations	39
5.4. Lessons Learned	39
5.5. Future Work.....	39

6. Bibliography.....	41
7. Annexes	44

LIST OF FIGURES

Figure 1: SMOTE. The axes represent the values of features x_1 and x_2 . A) is a point from the minority class where the lines correspond to the distance to the nearest neighbors. B) is a chosen neighbor from the 3 nearest neighbors. C) is the new generated data point....	8
Figure 2: 3-Fold Cross Validation. In each iteration, the original train data is divided into 3 folds: one is held as the unseen data and the remaining as the train data. After training a model on the train data, the chosen evaluation measure is calculated based on the results of the model on the unseen data. The final performance is given by the average of the values of the chosen evaluation measure along the different iterations.....	9
Figure 3: Decision Tree architecture. From the root node, the observations are divided accordingly with the decision taken on the decision node. The observations on a leaf node will not be further divided.....	12
Figure 4: SVM for a 2-class problem. The axes represent the values of the features x_1 and x_2 . A) and B) are support vectors and the dashed lines correspond to the boundary of the respective class. C) is the decision boundary that maximizes D) the margin.	17
Figure 5: Confusion Matrix for a 2-class problem. The lines of the matrix correspond to the number of observations within the actual class value while the columns represent the number of observations within the respective predicted class value. The values for TN, TP, FN and FP allow to evaluate the correctness of the predictions made by the classifier. .	18
Figure 6: Fourier Transform. The time series function (in black) is decomposed into simpler functions (in blue and grey) with a given frequency and amplitude.	21
Figure 7: Wavelet transform properties. A) Scaling the wavelet to detect different frequency resolutions. B) Translation of the wavelet over time	22
Figure 8: Tsfresh. The original dataset with the time series of each device is transformed into a dataset with the respective extracted features of each time series.	25
Figure 9: Example of the time series with a clear pattern of bytes received by the router over time for an Amazon device	27
Figure 10: Example of the time series of bytes received by the router over time for a non-Amazon device	27
Figure 11: Number of missing values on the train set. A high number of devices have at least one missing value.	29
Figure 12: Train set data imbalance. Only 5% of devices are Amazon devices.....	30
Figure 13: Example of the overfitting of the decision tree models in the scenarios where SMOTE and ADASYN were followed to balance the data and for different balancing rates.	

The models have a good performance on the train set and behave poorly on the test set.	34
Figure 14: Performance on the test set of each model following a cost-sensitive analysis. The Random Forest model from scenario C is the only model with f1-score above 50%.....	35
Figure 15: Precision-Recall curve on test set of the models in scenario C. The tree-based models show better performance than logistic regression and SVM.....	35
Figure 16: Evaluation measures on train set and on test set along 5 different seeds. The performance of the best model on the test set does not differs significantly from the performance on the train set.....	36

LIST OF TABLES

Table 1: Differences between the scenarios regarding the followed techniques to impute the missing values and to balance the data.	30
Table 2: Grid search tuned hyperparameters per algorithm.....	32
Table 3: Number of selected features accordingly with the missing values imputation method.	33
Table 4: Confusion matrix of the best model. The number of FN remains high.	36
Table 5: Tuned hyperparameters of the best model.....	37

LIST OF ABBREVIATIONS AND ACRONYMS

AdaBoost	Adaptive Boosting
ADASYN	Adaptive Synthetic
AUC	Area Under the Curve
CART	Classification and Regression Tree
CPE	Customer Premises Equipment
FN	False Negative
FP	False Positive
Hctsa	Highly Comparative Time-Series Analysis
IIoT	Industrial Internet of Things
IoT	Internet of Things
LOCF	Last Observation Carried Forward
MAC	Media Access Control
NFL	No Free Lunch
NOCB	Next Observation Carried Backward
OUI	Organizationally Unique Identifier
RFE	Recursive Feature Elimination
SEMMA	Sample, Explore, Modify, Model and Assess
SMOTE	Synthetic Minority Over-sampling Technique
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TR-181	Technical Report 181
Tsfresh	Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests
XGBoost	Extreme Gradient Boosting

1. INTRODUCTION

Nowadays it is no surprise that companies, especially the ones that deal directly with the end-user, need to better understand their users. More in-depth knowledge of the users opens the doors for companies to have customized offers that match their client needs. This allows them to provide more value and fight customer attrition rates. The telecommunication service providers, hereinafter referred only as *service providers*, are not an exception as they also starve to have insights on their customers.

One of the areas of interest for the service providers is to enhance the in-home experience: we all want to avoid that painful experience of not having a perfect access to the internet while at home. This can lead to endless (and sometimes useless) support field trips and calls to the service providers' customer care, unsatisfied customers and complaints. And, of course, all of this comes with a cost for the service provider. In this way, they should focus their attention on avoiding home network issues, particularly when facing an IoT revolution, with several in-home devices that should be working on their full potential, being connected 24/7 and collecting and exchanging data non-stop (after all, Alexa should always be there for us, right?!).

Service providers may ask for help in this matter to other firms, such as the company where the project described in this document was conducted, onwards referred to as *telecommunications and technology company*. To be successful in this mission the telecommunications and technology company provides an in-house developed product and presents the insights that result from the analysis of the data collected by it.

Particularly in this scenario, the service providers aim to more accurately identify the type of devices connected to each home network. This information allows them to trigger an alarm when the network usage from the devices differ from the expected pattern, which may indicate that there is a connectivity problem undergoing, leading to further investigation on it. In this way, the process of detecting issues in the home network becomes easier and faster.

The project in hands consisted in classifying each device as being an Amazon device or not, based on the traffic transferred overtime from the device to the residential gateway. Amazon has a diverse portfolio of devices: from always-connected devices, such as Alexa and Echo, to mobile devices like Kindle and Fire tablets, and stationary devices as the Fire TV. Besides this, there are also different models for each of the devices. So, although just the fact of identifying if a device is from Amazon, given its behavior, is an early stage and a small piece on this huge puzzle, it will definitely improve the service providers insights on the make of the device that their customers use and thus on the behavior expected from them.

The present document describes the application of the knowledge learned throughout the 1st year of the master program in order to solve a real-world problem that was proposed during a 6-month internship at the telecommunications and technology company. The report is organized in several chapters that aim an easy reading: first an introduction to the project and company (Chapter 1); then the theoretical framework that allows for a contextualization and the awareness of the existing related work (Chapter 2); followed by an overview of the technologies that were used (Chapter 3); and by the explanation of the applied approaches, and how the challenges were overcome in order to reach a final model (Chapter 4); and the last conclusions on the internship project and next steps (Chapter 5).

1.1. COMPANY OVERVIEW

The internship was conducted at a European multinational company that is one of the global leaders in the areas of technology, telecommunications and software.

A big slice of its customers is composed of communication service providers in over 100 countries, that are offered with a portfolio of network equipment, software, services and licensing that aims to improve their service on the new digital era. This translates to more than 6 million subscriptions that are hit by the work delivered by the company. Besides providing networking solutions to public and private enterprises, the end-users can also have access to the technology and devices developed by the company.

Nowadays, the company is innovating in terms of IoT and IIoT solutions, cloud services and, specially, developing 5G technology. In this last one, significant investments are being made and developments were achieved, making the company an active player in this matter.

The internship took place in the Portugal office, which is considered to be a center of innovation and where part of the software development of the company is done.

1.2. THE TEAM AND ACTIVITIES

The team is mainly responsible for the analysis of customer care data. The elements of the team have different backgrounds, ranging from data scientists to solution architects and big data engineers. The daily activities include IoT related tasks, such as the analysis of the data that are produced by it, which involves working with the software products and technology developed by the company.

In this context, it is possible to highlight a product that the team works close with, as it corresponds to the source of data used for this project. The product consists of an engine that is capable of collecting data from the CPEs located at the service providers' customers' homes. These home devices include set-top boxes, gateways, extenders, routers, end-user devices, among others. The data is collected accordingly with the Broadband Forum's TR-181 data model, which gives several features on the home network usage, not only on the device but also on the associated devices to that particular device, like computers, televisions and phones. Some examples of the available variables are the software version, manufacturer OUI and model name of the device, the hostname, MAC address, signal strength of each of the associated devices, packets/bytes sent and received, among many others.

Then, it can build insights on the data collected, such as "The wi-fi coverage is poor". On top of that, it provides recommendations to the service providers to solve any existing issue based on the insights, for example, "Add a repeater". Hence, it is possible to store the historical insights and recommendations, allowing a better knowledge of each home's profile.

The main goal of this product is to provide a better in-home experience to the end-customers by being able to troubleshoot connectivity problems, before the customer realizes there is an issue undergoing. In this way, the customer care becomes a more proactive and predictive sector, because it is possible to detect when an issue will happen and the best solution to solve it based on the past events of each home and its network usage. An enhanced customer experience and care service along with the reduction of faults and dispatches makes it possible for the service provider to witness a drop on the costs that they incur with the broadband home.

Throughout the internship, the key responsibility was to analyze time series data received from the product described above. This was achieved by developing and evaluating the performance of machine learning models to solve real-world problems that would allow the service providers to have a deeper comprehension of the home network usage.

1.3. INTERNSHIP GOALS

In order to broaden the knowledge on the make of the associated devices used by the end-customers, hereinafter referred also as *devices*, one can analyze how they behave over time in terms of internet usage.

In this way, it was observed the internet traffic data from the devices, namely the bytes received by the home gateway from the device. The bytes received consist of the amount of data exchanged between an associated device and the respective router. In other words, every time we browse the internet, an interaction of our device with the router is set in the form of bytes. For example, if we download a file from the web, the bytes sent will increase as it was requested information from the router to the associated device. On the other hand, if we upload a file, for instance as an attachment to an email, the bytes that are received by the router will increase because it was given information to the router.

Thus, if one can associate a certain behavior of the traffic data of the device to a specific manufacturer, the make of the device could be traceable and afterwards, the knowledge on which devices are used in each home network would increase. Moreover, it could also help to identify network problems when the expected pattern is not verified, leading to further investigation.

In particular, previous data analysis may indicate that some devices that belong to Amazon seem to have a particular signature, showing a peak of bytes received from time to time (which is suspected to be a sort of a keep alive or status message sent to the router). This pattern is more visible on the devices that are connected for long periods of time. This may be a clue that knowing the fingerprint of the device in relation to the bytes received, can lead us to the manufacturer, at least in the case of the Amazon devices.

Hence, the main goal of the internship was to develop a machine learning algorithm that could tell if a given device, that is connected over wi-fi to a router, is from Amazon or not based on the bytes received overtime. More precisely, the project aims to explore the possibility of building a model that is able to identify the Amazon devices based on the bytes received by the router and in parallel enrich the data science pipeline with approaches and methods to deal with similar data.

In this way, if a suitable model is achieved, it will be possible to:

- a) Provide a more accurate count of the Amazon devices in the population of connected devices (mostly when no information on the maker is available);
- b) Have insights on the relevant characteristics that make the internet behavior of the Amazon devices distinct from the remaining ones.

Translating to data scientist terms, one single variable is measured in different time periods, making it a univariate timeseries classification problem. To tackle it, the mainstream SEMMA approach was followed: this includes the steps of describing the sample available for machine learning purposes, performing some data exploration in order to better understand the phenomena in study, creating new features accordingly to methods used for timeseries present in the literature, training different algorithms and afterwards evaluating the performance of each model to find a final winner. Each of these steps will be further developed in Chapter 4.

2. THEORETICAL FRAMEWORK

In this section is presented the theoretical knowledge relied on in order to classify the devices into Amazon or not based on their fingerprints of bytes received by the router over time. In this matter, bellow is provided an introduction to machine learning and some of its concepts and algorithms that are relevant for the end purpose, as well as an understanding of time series data, the importance of a feature-based analysis and pointers to relevant work and literature in these fields. These topics are important to better understand the rationale behind the experiments performed throughout this project.

2.1. MACHINE LEARNING

Machine learning is a branch of Artificial Intelligence that aims to develop algorithms and techniques related with computational and statistics methods, that allow computers to “learn” [1]. In this way, it is useful to extract information from data automatically, *i.e.* without using explicit instructions as these may not be trivial to produce. Lately, the term machine learning has been subject to the technological hype. In fact, we can see its applications in several domains, including telecommunications. In this sector, it is common to apply machine learning for recommendation systems, service personalization and media recognition (*e.g.* image, video, sound pattern classification) purposes [2]. Regarding the project described in this document, machine learning will be an essential tool to find the underlying pattern of network usage that make the Amazon devices distinct from the other devices.

In order to better understand the role of machine learning, is important to clarify some concepts that are relevant for this use case.

2.1.1. Categories of Machine Learning Problems

Firstly, one should identify the nature of the task in hands: when the available data is already labeled, meaning the correct answer is already tagged, we are in the presence of **supervised learning**. This project falls in this category of machine learning problems, as there is already a flag stating whether a device is from Amazon (assuming the value 1) or not (assuming the value 0).

On the other hand, the **unsupervised learning** refers to problems where there are no pre-existing labels on the data and therefore, the goal is to partition the data based on its similarity.

Moreover, the supervised learning is commonly associated with **classification problems**, where the goal is to find a classifier f that correctly maps the data X into the respective label y , that is $f(X) = y$. In other words, it aims to allocate each observation to the respective group and thus, the target variable, y , is categorical. The **binary classification problems** are a subset of the classification problems, where the labels only assume the values 0 (or False) for a negative observation of the group in study and 1 (or True) for a positive observation. In this way, the project described in this document corresponds of a supervised binary classification problem.

The **regression problems** are also included on the supervised learning category and are related to problems where the target variable is continuous.

2.1.2. Overview of the Machine Learning Process

The machine learning process should start by partitioning the original dataset in, at least, two subsets that should be as independent as possible: the training and the test sets. The exploration of the data begins in the training subset where the necessary actions are taken accordingly to the findings (for instance, how to deal with the missing values or inconsistencies in the data). This includes a range of techniques for data preparation and feature engineering, which are done in order to improve the performance of the algorithms. These actions will be reproduced on the test set accordingly with the conclusions taken on the training data. Afterwards, the models are trained observing only the preprocessed data from the train set while the test set data is used later on to evaluate the performance of the trained models in an unseen set of data. The evaluation of the performance of a model relies in a range of evaluation measures, that will be further detailed in this chapter. The chosen model should be the one with the best and stable behavior on the unseen data.

2.1.3. Data Preparation

In order to ensure the independence between both the train and test sets, any duplicated observation should be initially removed before the splitting to avoid any data leakage to the test set.

Additionally, **feature scaling** is also important, especially when dealing with distance-based algorithms. When having features with different scales and ranges, the models will attribute a higher importance to the ones with higher values. Feature scaling will change the range of the numerical features, giving an equal weight to each of them. Normalization and standardization are two common methods to perform feature scaling. The first one scales the variables to have values between 0 and 1 (by applying, for instance, Equation 1), while the second one transforms the variables to have mean 0 and standard deviation 1 (by using the Equation 2, for example).

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (1)$$

$$x' = \frac{x - \mu}{\sigma} \quad (2)$$

Furthermore, one should take into consideration that most of the methods in the machine learning pipeline and algorithms do not deal with the existence of unknown values in the data. Therefore, there are several techniques to perform **missing values imputation**, such as considering the attribute's mean value of the respective observation or a global constant (like an impossible value given the context) [3]. To note that when dealing with time series data, is also possible to consider the previous (LOCF) and next values (NOCB) of the series to fulfill the missing ones [4].

2.1.4. Feature Engineering

Another issue that requires attention is the so-called curse of dimensionality. This happens when the data has too many features, which can bring unnecessary information and can lead the model to overfit. In fact, each added feature will make the model to learn the particularities and the appearance of the training data and consequently, the algorithm learns too well the training data but has a poor performance on unseen data. Thus, when in the presence of a higher dimensionality space, it is more complicated for the algorithm to achieve an optimal solution [5]. The curse of dimensionality has a bigger impact on the algorithms that partition the vector space and rely on computing distances between data points.

In order to avoid the curse of dimensionality, one can select the features that are more relevant for the classification problem. There are several methods for **feature selection**, such as the correlation matrix and the RFE [6]. The correlation matrix allows to visualize the features that are highly correlated with each other and thus not bringing new information to the model. By keeping only one of the features that are correlated, one is performing feature selection. In what concerns to the RFE, it trains an estimator in recursively smaller sets of features [7]. In other words, the estimator is fit on the data and the features are ranked based on their coefficients. Then, the less important feature, *i.e.* the one with smaller discrimination ability, is discarded, and the model is fit on the remaining features. The process is repeated until a predefined number of features is achieved.

In parallel with feature selection, it may also be relevant to perform **feature extraction**. This consists in creating new useful variables, based on the existing ones and/or on the field knowledge, bringing additional and more relevant information [6]. In this way, the feature extraction can also contribute to feature selection as the new variables can provide a summary of the information contained on the original set of features (which can be later discarded).

2.1.5. Imbalanced Dataset

When using real-world data and, in the context of a classification problem, it is common to happen to have more observations of a certain class (usually the negative class), making the dataset imbalanced.

In the presence of an imbalanced dataset, one can conduct a **cost sensitive analysis**, where is possible to assign different weights on the classification errors of each class, instead of the proportional weights assumed by default [8][9]. In this way, the goal is to give a higher penalization (or cost) in the loss function of the classifier when observations from the minority group are misclassified as the model seeks the minimization of the cost.

In alternative, under-sampling and oversampling are both techniques that should be taken into consideration to help balancing the data [10].

The **under-sampling** means to reduce the number of observations of the majority class, by randomly dropping a percentage of them, for example.

The **oversampling** consists of increasing the class with less observations by means of, for instance, repetition of the available observations of this class, SMOTE or ADASYN.

2.1.5.1. SMOTE

The SMOTE method consists of the creation of ‘artificial’ observations similar to the ones of the class we want to increase the presence. In each iteration, it randomly chooses a data point, x , belonging to the minority class and one of the k nearest minority class neighbors, x_k . Then, it generates a synthetic data point, x' , on the line defined between the two instances (see Figure 1). This is done by taking the difference between the two points, multiplying the difference by a random number between 0 and 1 and add it to the selected sample (see Equation 3)[11].

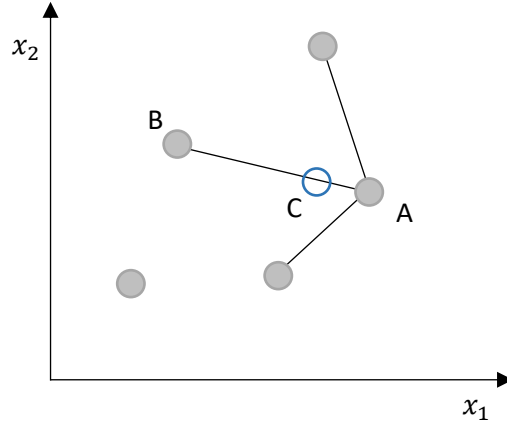


Figure 1: SMOTE. The axes represent the values of features x_1 and x_2 . A) is a point from the minority class where the lines correspond to the distance to the nearest neighbors. B) is a chosen neighbor from the 3 nearest neighbors. C) is the new generated data point.

$$x' = x + rand(0, 1) \times |x - x_k| \quad (3)$$

2.1.5.2. ADASYN

The ADASYN approach, follows a similar reasoning to that of SMOTE, but in addition it generates more data for the harder to learn observations [12]. In this way, it starts by defining the neighborhood of each minority instance, which is comprised by both majority and minority observations, and calculates the ratio of dominance of the majority class for each, r_i (see Equation 4). Then, this ratio will weight on the number of observations to be created in each neighborhood, such that a higher number of synthetic examples will be generated in the neighborhoods with higher dominance of the majority class. The new observations are then generated using the same linear combination between the minority observations within the same neighborhood, as in SMOTE (see Equation 3).

$$r_i = \frac{\text{number of neighbors of the majority class}}{\text{number of neighbors}} \quad (4)$$

2.1.6. K-fold Cross Validation

The k-fold cross validation aims to foresee the behavior of a model on unseen data [13]. As such, it splits the train set into k groups and for each unique group, holds out the respective group as unseen data and fits the model on the observations of the remaining groups (see Figure 2). Afterwards, the model is evaluated on the unseen set and the evaluation measure is kept. In the end, the performance of the model is given by the average value along the groups. In this way, the unseen data sets should resemble the test set and be as independent as possible from the train data, thus any preparation of the data should be made inside the loop. In addition, when using the stratified k-fold cross validation, one can ensure that the unseen set has the same proportion of observations from the minority class as the train set. This procedure is also advantageous when the available data is limited and usually presents less biased estimations of the performance of the model than the simple train-test split method.

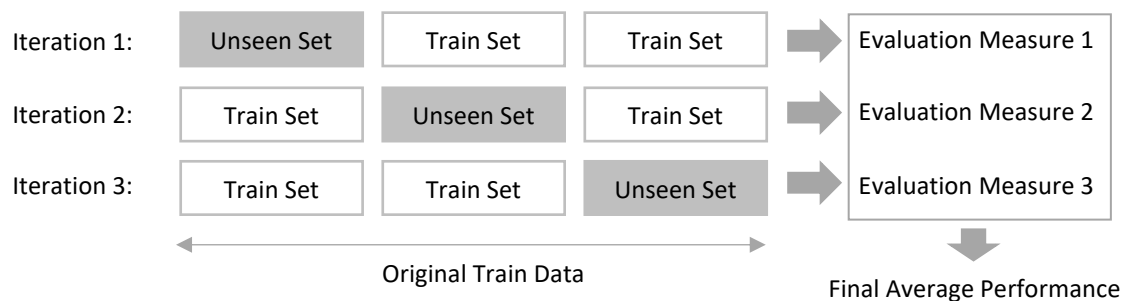


Figure 2: 3-Fold Cross Validation. In each iteration, the original train data is divided into 3 folds: one is held as the unseen data and the remaining as the train data. After training a model on the train data, the chosen evaluation measure is calculated based on the results of the model on the unseen data. The final performance is given by the average of the values of the chosen evaluation measure along the different iterations.

The k-fold cross validation is commonly performed simultaneously with the optimization of the hyperparameters of the model. In this matter, grid search can be applied to tune the model, where all possible combinations between the hyperparameters values are explored in order to find the configuration that results in the best performance of a model, given an evaluation measure.

2.1.7. Machine Learning Algorithms

In what concerns to the machine learning algorithms, there are several ones specific for classification problems purposes. In accordance with the NFL theorem, there is not an algorithm that performs better than the others in a given problem and dataset, so several models should be trained. Below are detailed the ones that are relevant for this project.

2.1.7.1. Logistic Regression

The logistic regression is a technique with its root on the statistics field, although can be also seen in a machine learning perspective. Following similar reasoning of the linear regression, the logistic regression is considered to be a linear method as it aims to fit a regression to the data. However, it has the particularity of being applied in binary classification problems, being the appropriate method of describing the relationship between one binary dependent variable and one or more independent variables (that can be discrete or continuous) [14].

For this purpose, the logistic regression fits a logistic function to the data, which limits the predictions to assume probability values between 0 and 1 of belonging to the positive class. In order to overcome these boundaries and be able to fit a line where the dependent variable can assume any value (just like in linear regression), the analysis is conducted on the log of the odds space of belonging to the positive class (see Equation 5) instead of the probability scale. This implies that the residuals (distance from the data points to the line) can as well assume values from negative to positive infinity. In this way, it is employed the maximum likelihood in order to find the best fitting line to the data.

$$\log(\text{odds}) = \log\left(\frac{\text{probability}}{1-\text{probability}}\right) \quad (5)$$

The maximum likelihood starts by generating a candidate line on the log of the odds space. When projecting the data points onto this line, one can obtain the respective value of the log of the odds, which can be converted to the probability by means of Equation 6. Once all the probabilities of the data instances are obtained, a logistic regression is defined. Afterwards, it is possible to calculate the log-likelihood of this candidate logistic line by adding all the logs of individual likelihoods of the data points. In this matter, the predicted probability given by the candidate line corresponds to the probability of the data point belonging to the positive class, and thus, the likelihood of an observation belonging to the positive class is the same as the predicted probability, while for an observation of the negative class it corresponds to 1 minus the predicted probability. In this way, a line that better fits the data (and therefore has less misclassified data points) have a higher log of likelihood. Then, a new candidate line on the log of the odds space is drawn by rotating the previous one and the same procedure is followed in order to achieve the log of likelihood of the new candidate. The line on the log of the odds space keeps rotating towards the maximization of log likelihood of the associated logistic line and the best fitting line is the one with higher log likelihood.

$$\text{probability} = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}} \quad (6)$$

The usage of the logistic regression implies the verification of some assumptions such as a binary dependent variable, low multicollinearity (meaning the correlation of the dependent variables should be low) and the existence of a linear relationship between the independent variables and the log of odds. However, when conducting an analysis on a machine learning point of view these assumptions are more flexible as the priority is the performance of the model on unknown data.

On the other hand, when interpretation of the obtained regression is relevant, one can obtain the coefficients on the generated line on log of odds space, which can be defined by Equation 7. In this way, the intercept β_0 can be interpreted as the log of odds when all the input variables assume the value 0, while the coefficients $\beta_1, \beta_2, \dots, \beta_i$ can be seen as the effect of one-unit increase of the respective variable, $x_1, x_2, \dots, x_i$, in the log of odds. For better interpretability, it is possible to convert these effects in the log of odds to the odds, by means of e^β and thus, obtaining the percentual effect on the odds of belonging to the positive class, given a one-unit increase in the respective variable. When in the presence of discrete variables, the algorithm will generate a line for each category, representing the associated log of odds of each one. The coefficients are obtained by Equation 8 (as an example of a binary input), where the first component correspond to the effect of the reference group, while the second one represents how much the non-reference group increases or decreases the odds of belonging to the positive class, on a log scale. To note that to identify if these coefficients are statistically significant for the model, one can additionally perform some statistical tests (which some software automatically calculate with the output of the model).

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_i x_i \quad (7)$$

$$\log(\text{odds reference group}) x_1 + \log\left(\frac{\text{odds non-reference group}}{\text{odds reference group}}\right) x_2 \quad (8)$$

Where x_1 corresponds to the reference group and x_2 to the non-reference group of a binary variable.

Moreover, it is possible to ascertain if independent variables used in the model are good indicators of the class to predict by calculating the R^2 , which takes the values of the log likelihood of the fitted line and the log-likelihood of the worst fitting line that, on the log scale, corresponds to the log of odds without taking into account any input variable (see Equation 9) [15]. In the same way, one can also obtain the respective p-values with the help of the test statistic given by Equation 10, which follows a χ^2 distribution with the degrees of freedom defined by the difference between the number of parameters in both models [16].

$$R^2 = \frac{\text{Log Likelihood of worst fit} - \text{Log Likelihood of fitted line}}{\text{Log Likelihood of worst fit}} \quad (9)$$

$$2 \times (\text{Log Likelihood of fitted line} - \text{Log Likelihood of worst fit}) \quad (10)$$

The logistic regression is compatible with several optimizers for the cost function [17] and has a hyperparameter, C, that aims to control overfitting by penalizing large magnitudes of the parameters.

In order to build a cost-sensitive logistic regression, the respective adjusted weights of each class are taken into consideration in the loss function used to optimize the coefficients of the fitted line [18]. Thus, a larger weight will lead to higher errors and therefore to the need of changing the coefficients of the regression to minimize it.

Finally, once achieved a best fit model, the predictions are made based on the respective probabilities. In this way, accordingly with the default threshold, if the probability is higher than 0.5 than the observation is predicted as belonging to the positive class, otherwise the prediction falls on the negative class.

2.1.7.2. Decision Tree

Decision trees are a supervised method that can be used to solve both classification and regression problems. It differs from the remaining models due to its tree-like graphic that comprises of a series of binary questions (sets of if-then-else rules) that when answered in sequence ultimately achieve a prediction. In this way, the decision trees have a particular terminology (see Figure 3) [19]:

- A **node** corresponds to a group of the data that met a particular rule. The first node where the decision tree starts, and thus includes all the observations, is called the **root node**. This node is subsequently divided into sub-nodes, which can be recursively divided again. A node that originates sub-nodes is known as **decision node**. Finally, a node that is not divided is called **leaf**;
- **Splitting** is the process of dividing the observations of a node accordingly to a defined rule;
- **Branch** is a sub-section of the decision tree that includes all the decisions taken from the root node until one leaf;
- **Pruning** is the process of removing sub-nodes.

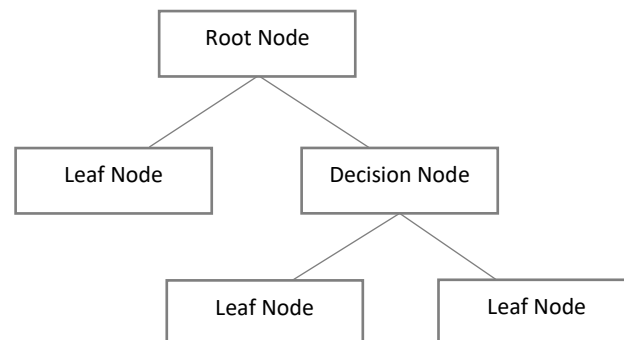


Figure 3: Decision Tree architecture. From the root node, the observations are divided accordingly with the decision taken on the decision node. The observations on a leaf node will not be further divided.

Starting from the root node, the decision tree classifier is built interactively by one split at a time. Each time a node is divided there is a decision on which is the best variable to split on and where in the range of values of the feature to split. The splitting process continues until all the observations in each partition are exclusively of one category or until some stopping criteria is met (such as the maximum depth of the decision tree) [19].

The decision on the splitting attribute is done by a criterion that maximizes the segregation of the target value, this is, that separates the data in a way that the observations at each node are increasingly dominated by just one category. The information gain and the Gini gain are two examples of decision criterions [3].

The information gain can be seen as the expected reduction in the target values diversity of the group by splitting the observations accordingly to a certain feature. This can be calculated based on entropy which measures the impurity of the data. The information gain of a particular attribute is computed accordingly with the following Equations 11, 12 and 13 [3].

$$Gain(A) = Ent(D) - Ent_A(D) \quad (11)$$

Where

$$Ent(D) = - \sum_{i=1}^C p_i \log_2(p_i) \quad (12)$$

Where D is the complete dataset, C represents the classes of the target variable, p_i is the probability of an observation being from the class i , and

$$Ent_A(D) = \sum_{j=1}^m \frac{|D_j|}{|D|} \times Ent(D_j) \quad (13)$$

Where m represents the different values that the attribute A can assume, $|D_j|$ corresponds to the number of observations where the attribute A has the value j , $|D|$ is the length of the dataset and $Ent(D_j)$ is the entropy on the subset of the dataset where the variable A has the value j .

In this way the minimum value that entropy can assume is 0, meaning that all the observations belong to one single category, while the maximum value is 1, which can happen when half of the observations belong to one category and the second half to another category [3].

The Gini gain is similar to the information gain but uses the Gini impurity to measure the impurity present in the data. The Gini impurity corresponds to the likelihood of incorrectly classify an observation, when that observation is randomly classified accordingly with the distribution of the target values. The formulas for the computation of the Gini gain of a feature are identical to the ones of the information gain and can be seen in the Equations 14, 15 and 16 [3].

$$Gain(A) = Gini(D) - Gini_A(D) \quad (14)$$

Where

$$Gini(D) = 1 - \sum_{i=1}^C p_i^2 \quad (15)$$

And

$$Gini_A(D) = \sum_{j=1}^m \frac{|D_j|}{|D|} Gini(D_j) \quad (16)$$

The Gini impurity ranges from 0 to 0.5, with 0 occurring when all observations are from the same category and 0.5 occurring when each category has an equal number of observations.

Thus, the decision trees can also be used as a method for feature selection, where the most important features, the ones with higher values of information or Gini gain, appear at the top of the built tree [19].

A cost-sensitive tree can be built by taking into account the importance of each class when calculating the splitting criterion, such that a larger weight will have a higher impact on the node purity [20].

In addition, there are various decision tree algorithms where all the process described above is implemented with slight differences (namely in the way of dealing with the different types of variables). An example of an algorithm is the CART which is implemented on the Python library Scikit-Learn and essentially differs from the remaining algorithms as it supports numerical target features [19].

The major advantages of the decision trees are the ease of interpretation, as they can be displayed graphically and are similar to the human decision-making process, and the least effort for data preprocessing. However, decision trees are data sensitive and thus can easily overfit. This drawback can be overcome by means of pruning, which can be achieved by, for example, defining a maximum depth of the tree or a minimum number of observations in each leaf. In this way, having final nodes with a narrow number of observations that denotes very particular cases on the data is prevented. Another solution to control the decision trees data overfitting issue is to use the ensembles methods, as they join the output from several models.

2.1.7.3. Ensemble Methods

The ensemble methods of learning consist of combining the decisions of several models in order to improve the classification performance. This is based on the principle that taking into consideration the outputs of a group of weak models can originate a strong model capable of surpassing the performance of an individual one. In other words, it assumes that these independent models can only see a slice of the data but not the entire picture, thus when joined together it is possible to have a broader perspective of the data [13][3]. The ensembles are composed of individual algorithms, known as base learners, which should be as diverse as possible. The classification of a new data tuple is done by the combination, for example the majority, of the predicted classes by each of the base learners. The following are examples of ensemble methods.

2.1.7.3.1. Random Forest

The random forest has as its base learner the decision trees. Each of these classifiers is applied on a sub-sample of the original dataset, where the observations are sampled with replacement by default. In addition, at each node it is also randomly selected a predefined number of attributes to determine the split. To note that the base learners are independent from each other. The final scores of a random forest are given by the average of the votes from the classification trees [3].

2.1.7.3.2. AdaBoost

The AdaBoost is an ensemble method that can receive as base learner other algorithms than just decision trees and, as a boosting method, it has the particularity of focus its attention on the misclassified data instances generating dependency between the base classifiers [3].

In this way, for each estimator, a sample of observations is selected from the original dataset with replacement. This selection is based on the weight given to each of the data tuples, which at the starting point will be equal for all the observations and then is interactively adjusted accordingly with the classification accuracy obtained by the current estimator (see Equation 17 for the incorrectly classified samples and Equation 18 for the correctly classified samples) and normalized such that the sum is equal to 1. Thus, it is given a higher weight to the tuples where the model failed to predict the class, which gives them a higher probability of being selected as input for the following estimator. Due to this process, there is still a risk of overfitting, as the misclassified samples will increase their presence on the selected observations by means of repetition.

$$\text{new sample weight} = \text{current sample weight} \times e^{\text{classifier weight}} \quad (17)$$

$$\text{new sample weight} = \text{current sample weight} \times e^{-\text{classifier weight}} \quad (18)$$

Where,

$$\text{classifier weight} = \frac{1}{2} \log \left(\frac{1 - \text{total error}}{\text{total error}} \right) \quad (19)$$

And

$$\text{total error} = \sum \text{misclassified samples weight} \quad (20)$$

A weight is also assigned to the vote of each classifier based on how well it performed, such that the more accurate the prediction, the higher the weight (see Equations 19 and 20). For each class, the respective scores are added, and the final predicted class is given by the vote with the highest score.

2.1.7.3.3. XGBoost

Similar to the AdaBoost, the XGBoost is an ensemble method that applies the concept of boosting and thus, the selection of the subset of observations to learn from is not completely random as the consequent individual model will put more focus on the misclassified data instances with a higher error associated and the final predicted class is a weighted decision of the votes obtained by each individual model as well. The XGBoost is most commonly used with decision trees as base learners. The main differences from the previous presented ensemble methods is that it follows the gradient descent to minimize the loss function and introduces a regularization parameter to prevent overfitting [21] [13].

In this way, the XGBoost algorithm starts by making an initial prediction for all the selected sample instances, which by default takes the value 0.5, and calculates the residuals in light of this prediction, this is, the difference between the observed and predicted values of each instance. Afterwards, a decision tree is fit to the residuals. For the calculation of the output values of the leaves, the gradient descent formula is used with the introduction of the regularization parameter, λ , at the denominator (Equation 21) and the new predictions correspond to the probability associated to the sum of the log of the odds of the initial prediction and of each of the generated trees scaled by the learning rate, ε (Equations 22 and 23). To note that the learning rate controls the overfitting as well, by shrinking the influence of each individual tree. Then, the residuals can be recalculated taking into consideration the new probabilities and a second tree can be built fitting the new residuals. However, in each of the subsequent trees, the previous probabilities used in the formulas change accordingly. This process is repeated until a maximum number of trees is reached or the residuals achieved a very small number.

$$Output\ Value = \frac{\sum Residual_i}{\sum (Previous\ Probability_i) \times (1 - Previous\ Probability_i) + \lambda} \quad (21)$$

$$Predicted\ Probability = \frac{e^{\sum \log(odds)}}{1 + e^{\sum \log(odds)}} \quad (22)$$

Where,

$$Sum\ Log(odds) = \log\left(\frac{initial\ probability}{1 - initial\ probability}\right) + \sum(\varepsilon \times Output\ Value_i) \quad (23)$$

As mentioned before, XGBoost introduces several hyperparameters that aim to control the overfitting, through methods that lead to the pruning of the trees. To mention a few, it is possible to define the minimum number of residuals in each leaf and to modify the regularization parameter to decrease the sensitivity to isolated observations. In this matter, higher values of regularization will contribute to the reduction of the gain. In the same way, one can define the value of the minimum loss reduction required to partition a node.

The XGBoost also presents the advantage of being able to handle missing values [21]. For this, when an observation does not belong to either sides of the threshold split due to the unknown values on the particular feature, still the target value is known and thus, the algorithm will try both the assignments and will chose the one that minimizes the loss.

In addition, the XGBoost has a hyperparameter called *scale_pos_weight* that makes it possible to conduct a cost sensitive analysis, by giving more impact on the classification errors of the positive instances, and thus, leading the algorithm to focus more in these instances to reduce the respective residuals [22].

2.1.7.4. SVM

The SVM classifier work towards the definition of a decision boundary that segregates the observations from each class in a way such as the margin, *i.e.* the distance between the extreme points on the dataset (the ones that are closed to the opposite class), is maximized. These points are called support vectors (see Figure 4).

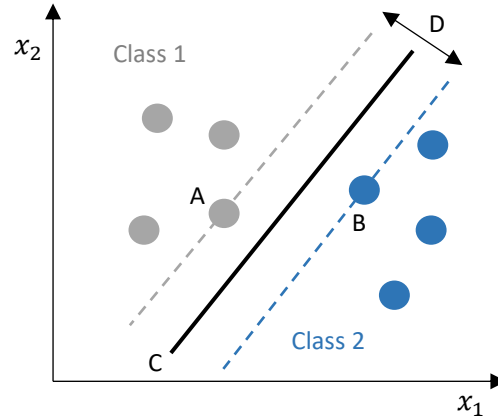


Figure 4: SVM for a 2-class problem. The axes represent the values of the features x_1 and x_2 . A) and B) are support vectors and the dashed lines correspond to the boundary of the respective class. C) is the decision boundary that maximizes D) the margin.

In fact, the algorithm starts by defining a random line or hyperplane and then, interactively moves it towards the direction of the misclassified observations and taking into consideration the maximization of the margin through vector calculus [3][23]. By maximizing the margin, the algorithm aims to minimize the misclassification of unseen observations.

However, not always is possible to obtain a clear segregation of the classes and thus, errors have to be accepted. In this way, the algorithm includes a hyperparameter, C , that controls the tradeoff between having a large margin and the occurrence of classification errors. Large values of C will result in low-margin lines or hyperplanes in order to reduce the misclassifications, while low values will produce large margin ones. This hyperparameter also allows to build a cost sensitive SVM when weighted by the importance of each class, such that larger weights correspond to a larger C value [24]. A regularization hyperparameter can be also considered to emphasize the weight of misclassified observations in the loss function.

Nevertheless, when the data is not linearly separable, the SVM algorithm has the particularity of mapping the data points into a higher dimensional space using kernels. A kernel consists of a function that takes the vectors in the original space and returns the dot product of the vectors in the high-dimensional space. In this way, the algorithm is able to proceed with the necessary calculations without actually perform the transformations on the data, which makes it less computationally expensive. This is based on the assumption that it is possible to find a higher dimensional space where the data is linearly separable. The polynomial, radial basis function and sigmoid are commonly used kernels, although one can also specify the function to use [3].

Moreover, it is possible to obtain the importance of the features when applying a linear SVM. This is given by the coordinates of the vectors that are orthogonal to the best boundary whose direction also indicates the respective predicted class [25]. When using the remaining kernels this option is not available and thus, the SVM behaves like a black box.

2.1.8. Evaluation Measures

In what concerns to the evaluation measures of the performance of a model, the most common one is the accuracy. However, this may not be enough to understand the behavior of the model and thus the accuracy is usually paired up with other measures such as precision, recall and f1-score (that considers both precision and recall) and, also, by visualizing the confusion matrix (see Figure 5 and Equations 24 to 27) [26][27] or the precision-recall curve. This last one represents the tradeoff between precision and recall at different threshold levels and is often used in binary classification problems, especially when there is a class imbalance. A high area under the curve means that both precision and recall scores are high and thus the classifier labels correctly the observations.

	Predicted C = 0	Predicted C = 1
Actual C = 0	TN	FP
Actual C = 1	FN	TP

TN = Number of elements that do not belong to C (C = 0), that have not been classified as C (C = 1)

TP = Number of elements belonging to C (C= 1), classified as C (C = 1)

FN = Number of elements belonging to C (C = 1), that have not been classified as C (C = 1)

FP = Number of elements that do not belong to C (C = 0), classified as C (C=1)

Figure 5: Confusion Matrix for a 2-class problem. The lines of the matrix correspond to the number of observations within the actual class value while the columns represent the number of observations within the respective predicted class value. The values for TN, TP, FN and FP allow to evaluate the correctness of the predictions made by the classifier.

$$accuracy = \frac{TP+TN}{TP+FP+TN+FN} = \frac{\text{Number of elements correctly classified}}{\text{Total number of elements}} \quad (24)$$

For a given class C,

$$precision = \frac{TP}{TP+FP} = \frac{\text{Number of elements belonging to C,classified as C}}{\text{Number of elements classified as C}} \quad (25)$$

$$recall = \frac{TP}{TP+FN} = \frac{\text{Number of elements belonging to C,classified as C}}{\text{Number of elements belonging to C}} \quad (26)$$

$$f1 \text{ score} = 2 \times \frac{precision \times recall}{precision+recall} \quad (27)$$

In fact, taking into consideration several measures other than accuracy has its advantages: it helps to make a better evaluation of the algorithms even when the dataset is not balanced [26] [27]. In this scenario, only looking at the accuracy may lead to the conclusion that the algorithm is performing well when in fact it is classifying all the observations into the majority class making the accuracy high, but the real performance is not. The precision, recall, f1-score, confusion matrix and precision-recall curve will give us more understanding on the performance of the model in each of the classes [27].

Finally, if the model performs well on the unseen data, meaning the values presented by the evaluation measures are acceptable, then it is possible to conclude that the model has a good generalization ability. In the opposite case, it may be an indicator that the model is **overfitting** (the algorithm learns too well the training data but has a poor performance on the test). To note that the presence of duplicated observations on the train set can lead to overfitting as well.

In this matter, during the training phase of a model, there are some measures that may reduce the overfitting, such as select the most relevant features, stop the learning process earlier and perform a k-fold cross validation.

2.2. TIME SERIES

There are several real-world phenomena that change over time: from the stock-market prices, to the weather or even to the electro-cardiograms [28]. The common factor in these examples is that one (or more) variables are measured along time, and so, they are called time series. In a nutshell, a group of data can be considered as a time series whenever there is a time stamp associated with the phenomena in study, creating a sequence of values.

In this project, only one variable, the bytes received, is being collected every hour, making it a univariate time series. Formally, one can define a univariate time series of length n as a sequence of numeric values $T = (x_1, x_2, x_3, \dots, x_n)$ where $x_i \in \mathbb{R}$ for $1 \leq i \leq n$ [26]. On the other side, if several variables are being measured over time, it is called a multivariate time series.

Moreover, it can happen that several time series are collected. For instance, it is possible to measure the temperature over time in different locations, so that there is one time series for each place. In this scenario, we are in the presence of so-called multiple time series.

In this way, the data used in the project at hands can be more precisely defined as a multiple univariate time series because the variable in study is measured for different devices along time.

When facing a supervised classification problem with time series data the goal is to find a function f , also called as classifier, that maps each sequence to the respective group [26][28]. In other words, given a pair of $(T_j, y(T_j))$ where T_j is a univariate time series $T_j = (x_{j1}, x_{j2}, x_{j3}, \dots, x_{jn})$ where $x_{ji} \in \mathbb{R}$ for $1 \leq i \leq n$ and $1 \leq j \leq m$, being m the total number of time series in the dataset, and $y(T_j)$ the respectively associated group label, one wants to find a function f such that $f: T_j \rightarrow y(T_j)$ [26]. This mapping aims to be as accurately as possible for every time series. Ideally, we want $f(T_j) = y(T_j)$ for all T_j [26]. Several studies have shown the application of time series classification in different areas, such as in the detection of credit frauds [29] or in the detection of abnormalities and faults in processes/systems [30][31].

Time series data can be decomposed into several components each representing an underlying pattern that affect the observed values at a given time: the trend, seasonal, cycle and irregular variations [32]. The trend is the overall long-term increase or decrease in the data, either linear or non-linear. The seasonal variations correspond to intra-year regular and periodic patterns that recur every year (for example, variations due to the season of the year), while the cyclical component consists of fluctuations without a fixed frequency comprised of successive phases of expansion and contraction (such as the “economic cycle”). In this way, the length of a cycle is longer than the length of a seasonal variation. The irregular component includes all the remaining random, irregular and unpredictable variations in the data after removing the trend, seasonal and cycle components. The decomposition of a time series in its components is helpful to better understand the phenomena in study and to improve the forecast accuracy and can be accomplished through several methods [32].

The analysis of time series data can bring some challenges: a) the length of each time series may not be uniform [26], for instance, in location A one can have the daily temperature for two weeks, while in location B there is data only for one week; b) the length of the time series can be quite large and thus more expensive in terms of resources when applying some machine learning algorithms [28]; and c) subsequent observations are usually naturally correlated [26], for example, the temperature at a specific hour is expected to be similar to the temperature in the following hour, which makes the use of certain algorithms unfeasible.

2.2.1. Feature-Based Analysis

In order to overcome the challenges that arise when analyzing time series data, one proposed approach by the literature is to perform a feature-based analysis. This consists in extracting features from the time series data that incorporate the existing properties of the original data. The classification will be further conducted based on these extracted variables, which makes it possible the usage of the most conventional and simple methods and algorithms [33]. Moreover, extracting features can also present the advantages of being less sensitive to noise and to reduce the training times when the number of extracted features is less than the length of the time series data [33].

These new extracted features can be of different natures and, as proposed in [34], combining different methods can be beneficial in terms of performance on unseen data. For the sake of this project, the time series were decomposed in components of different domains. An exhaustive list of all the features extracted by the library `tsfresh`, described in Chapter 3, and that were the input for the algorithms that were used in the project, can be seen in [35].

To name a few, statistical features such as the mean of the values in the time series, the count of the number of values below and above the mean, the maximum and minimum values, the number of peaks, the autocorrelation in different lags, among others, were calculated [35].

In more detail, regarding the number of peaks in the time series, a value is considered a peak when is bigger than its n neighbors to the left and to the right (also referred as support). Different values for the support can be used in the calculations and the function returns the count of observations that are considered peaks of the respective support.

The autocorrelation in a time series measures the dependency between a series and its lagged version and can be computed by Equation 28 for a given time t .

$$\frac{1}{(n-l)\sigma^2} \sum_{t=1}^{n-l} (x_t - \mu)(x_{t+l} - \mu) \quad (28)$$

Where n is the length of the time series $X_t = (x_1, x_2, x_3, \dots, x_n)$, being σ^2 its variance and μ its mean and l is the lag.

In addition, properties from the Fourier and wavelet transformations were also taken into consideration, which are mathematical methods to detect periodic signals on a time series [35].

In a nutshell, the Fourier transformation decomposes a periodic function in the time domain to simpler functions in the frequency domain that take the form of sines and cosines waves (see Figure 6) [36][37]. In other words, it decomposes the original function in a weighted summation of sinusoids (see Equation 29). The Fourier transform coefficients can be obtained by means of Equations 30 and 31, where for each frequency bin, k , it returns a complex number (see Equation 32) from which is possible to extract the vector's magnitude (that will correspond to the amplitude of the wave at that frequency bin) and the phase angle (that will represent the shift of the respective wave). Thus, the original signal can be expressed by sinusoid waves of different amplitudes and at different frequencies.

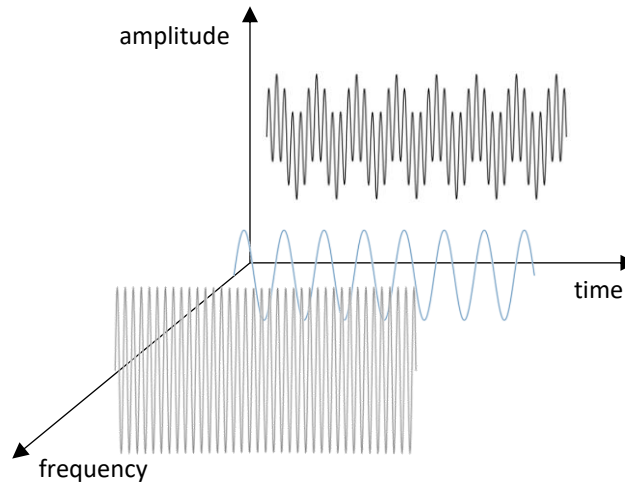


Figure 6: Fourier Transform. The time series function (in black) is decomposed into simpler functions (in blue and grey) with a given frequency and amplitude.

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos 2\pi kt + b_k \sin 2\pi kt) \quad (29)$$

Where k denotes the frequency and a_k, b_k for $k \in \{1, 2, 3, \dots, \infty\}$ are the coefficients of the Fourier transform.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{j2\pi kn}{N}} \quad (30)$$

Where N is the number of samples and $e^{-\frac{j2\pi kn}{N}}$ is a complex exponential that can be simplified Euler's formula:

$$e^{jx} = \cos x + j \sin x \quad (31)$$

$$X_k = A_k + B_k j \quad (32)$$

On the other hand, the wavelet transformation looks for similarities between a given function (or small wave, in this case the Ricker wavelet [38]) and the time series data, with the particularity of being able to scale and translate the wavelet function, and thus, considers both time and frequency domain (see Figure 7) [36][39]. The scale consists in compressing (generating a high frequency wavelet) or stretching (generating a low frequency wavelet) the wavelet, in order to detect fast and low variations on the signal, respectively. The translation parameter is responsible for the shifting of the wavelet along the time. In this way, for different values of scale, a , and position, b , it takes into consideration the value of the signal in the time domain and the wavelet function, and returns the values of the coefficients of the wavelet, $C_{(a,b)}$ (see Equation 33). A spectrum of magnitudes of the coefficients can be built to better visualize the changes on the original signal over time and in different frequencies. Thus, the original signal can be transformed into several wavelets with different scales and at different positions.

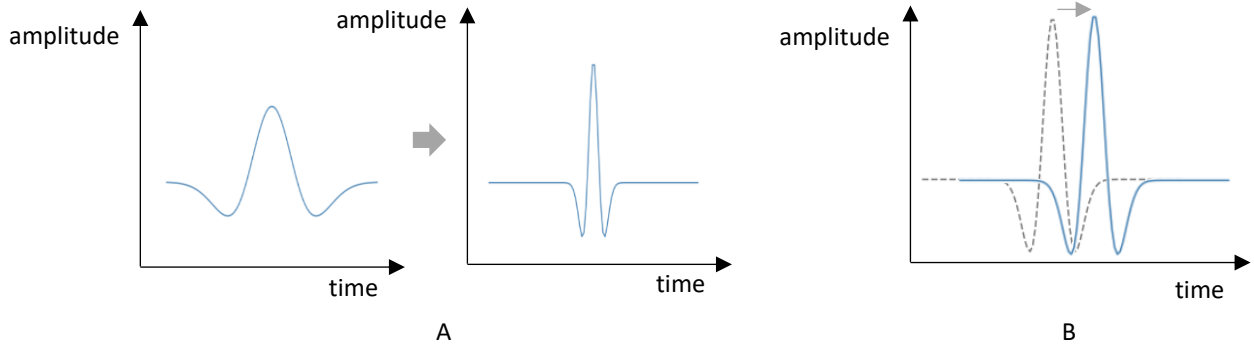


Figure 7: Wavelet transform properties. A) Scaling the wavelet to detect different frequency resolutions. B) Translation of the wavelet over time

$$C_{(a,b)} = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} f(t) \cdot \psi\left(\frac{t-b}{a}\right) dt \quad (33)$$

Where $f(t)$ is the signal and ψ is the mother wavelet.

Due to the high number and diversity of features that can be extracted from a time series, the feature extraction process is usually followed by methods of feature selection. These aim to reduce the number of features to the ones that are the most useful to predict the class that an observation belongs to. For this, they can rely on the discrimination ability or in the relation between the features, for instance.

To sum up, when in the presence of a classification problem with time series data, the related work shows that a common approach is to follow a feature-based analysis. This involves extracting features from the time series and then conduct the analysis based on the new variables. In this way, it is possible the usage of conventional machine learning classifiers such as decision trees, ensemble methods, logistic regressions and SVMs. Moreover, the literature also refers to multiple methods to mitigate the presence of problems that may affect the performance of the models, namely the existence of missing values, the imbalance on the data and the overfitting.

3. TOOLS AND TECHNOLOGY

The support of tools and technologies is essential to perform machine learning tasks. The present project relied majorly on the ones presented throughout the curricular year of the Master's degree, namely the Jupyter Notebook tool and the Python programming language. On the last one, the following libraries can be highlighted:

- SciPy, NumPy, Pandas and Matplotlib mostly for the data analysis of the dataset;
- Scikit-Learn which contains a considerable portfolio of implementations of machine learning algorithms, as well as utility functions for data preprocessing.

Besides the libraries mentioned above, the tsfresh [35] and XGBoost [22] Python packages were also used, which are further detailed in this chapter.

3.1. TSFRESH

Tsfresh is a Python package that performs feature extraction from a time series dataset, with additional support for doing feature selection over the calculated features [40]. In a nutshell, this library comprises of 65 functions to create new variables related with basic statistics, correlations, distribution and Fourier and wavelet transforms. The mean value, number of peaks, quantile and skewness are just some examples of these functions. Thus, the number of final calculated features can be larger than 65 (by default it outputs 794 time series features), considered that for the same function, different values of the parameters are used by default, although there is the option of manually defining these values. The list of all the methods to calculate features as well as their description and parameters can be accessed in the documentation page of the package [35].

For each univariate timeseries present in the dataset, tsfresh calculates the portfolio of features, and then, in order to filter out the irrelevant ones for the classification task, it will additionally perform significance hypothesis tests which will output an array of p-values and, finally, apply the Benjamini Hochberg procedure to control the number of false positives (the ones that accordingly with the p-value are considered relevant, but they are actually not) [35]. In this way, it is possible to transform the original dataset that contains multiple time series of one or more variables, into a dataset with several features that characterize those series. A visual support of the whole process can be seen in Annexure 1.

Currently, there is an alternative to the tsfresh package named hctsa, which was developed for MATLAB (although it is also possible to use it with Python through pyopy). Just like tsfresh, its ultimate goal is to extract features from time-series data, and, for this purpose, it includes around 155 functions that allow to create more than 7700 features, when using different values in the parameters of each one of them. Hctsa also contains visualizations and functions to analyze the results and to support the feature selection process [41][42].

Nevertheless, the tsfresh package presents some leverages. Firstly, it can be directly used with Python, the programming language most commonly used for data science at the telecommunications and technology company. Moreover, it is compatible with Scikit-Learn, which implies that it can be easily integrated in machine learning pipelines and it is fast in calculating the features [40].

Taking into consideration that the majority of the data that the telecommunications and technology company deals with has a timestamp, this package is seen as a useful and time saving technology because it creates a diversity of time series features in one single line of code, instead of having to generate them individually.

In the present project, the tsfresh was used to perform feature engineering by calculating new features for the network traffic received from each device. In addition, the option of extract the relevant features was used. The transformations on the original dataset can be visualized in Figure 8. In this way, the new variables incorporate the time factor and can later be used as input for the common machine learning classifiers, which will be deepened in the next chapter.

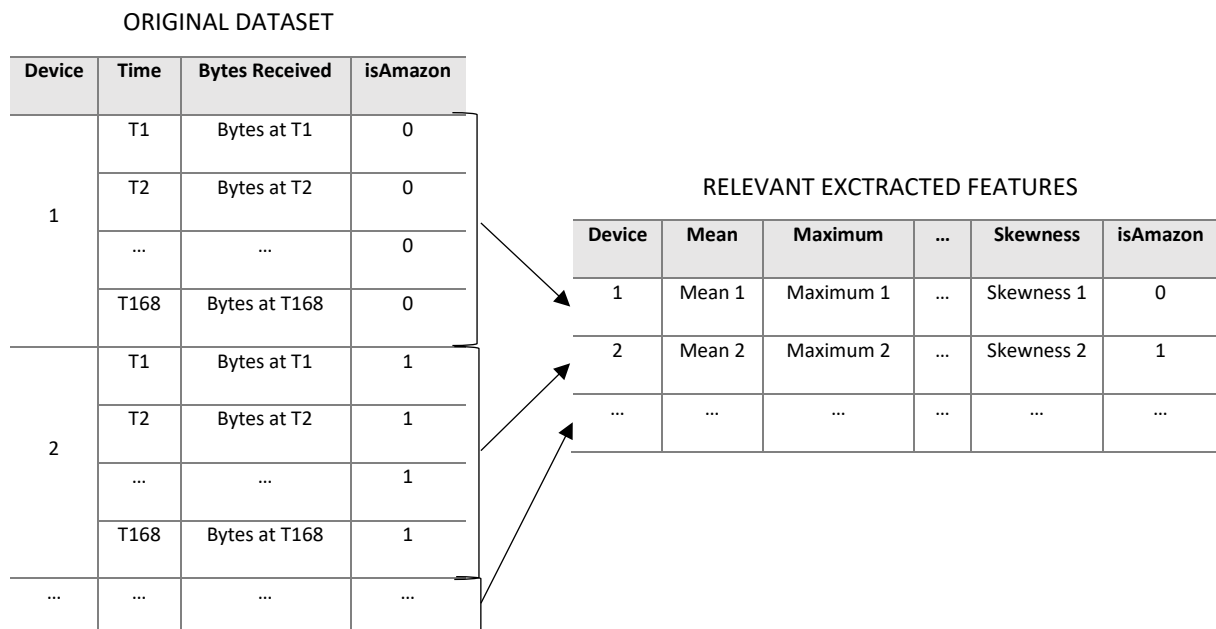


Figure 8: Tsfresh. The original dataset with the time series of each device is transformed into a dataset with the respective extracted features of each time series.

3.2. XGBOOST

XGBoost is a library that provides a high-performance implementation for gradient boosting decision trees (see section 2.1.7.3.3.). The introduction of systems and algorithmic optimizations, such as the handling of missing values, the parallel and distributed computing, makes it a scalable and efficient resource [21]. It also has implementations in several programming languages, including Python, where it provides integration with Scikit-Learn [22]. Currently, there are similar libraries to build gradient boosting trees, such as CatBoost [43] and LightGBM [44], that implement slightly different variations of the algorithm, as for instance on the procedures to handle categorical features. The XGBoost library was applied in this project in order to implement the gradient boosting algorithm (XGBoost).

4. THE PROJECT

The internship consisted of one single project that aimed to develop a machine learning model for device fingerprinting, which can be translated into a multiple time series classification problem. This section aims to present the details of the project and to explain the whole execution process from the reasoning behind on each decision to the obtained results and final conclusions.

4.1. TIMELINE

The project described in the present document lasted for a period of 6 months, which can be split into two phases:

- The first half of the period was dedicated to domain knowledge, to research on related work and to define the strategy to take;
- The remaining half was reserved to the implementation of the defined strategy and, when needed, to proceed to changes on it, as well as to the analysis of the final results.

To be more precise, the domain knowledge included the understanding of the operation of the company's product, which includes the data collection process, as well as getting familiar to telecommunications related data. The related work had also an important role as timeseries classification problems are not a very common type of data science problem and thus the experience on this topic was limited. Based on this research and after some exploratory data analysis, it was possible to plan the approaches to tackle the problem in hands, being that this was a dynamic process, suffering changes along time, while the implementation was taking place.

4.2. DEVICE FINGERPRINTING PROJECT

The project consisted in developing a machine learning model for device fingerprinting, in other words, to associate a certain pattern to a specific device. In particular, the model should be able to classify if a device is from Amazon or not based on the bytes received by the home router over time.

The reason why this project targets the Amazon devices instead of devices from any other maker and the focus on the variable bytes received by the router in place of any other available feature, is due to a previously conducted analysis. In accordance to the analysis, some Amazon devices seem to present a pattern, showing regular peaks of bytes received by the router from time to time, what can be considered as 'stay alive' messages. Figure 9 shows an example from an Amazon device where this happens. On the other hand, the devices from remaining makers seem to have a more random pattern, as is possible to see on Figure 10. The difference between the patterns is more visible when the devices are always connected to the network. Either way, it is not possible to generalize, this is, to state that these patterns are verified for all the Amazon and non-Amazon devices, as the analysis was conducted on a small sample and on a restricted number of makers. Nevertheless, the analysis may indicate that the variable bytes received could lead to the maker of a device, at least in the case of the Amazon devices.

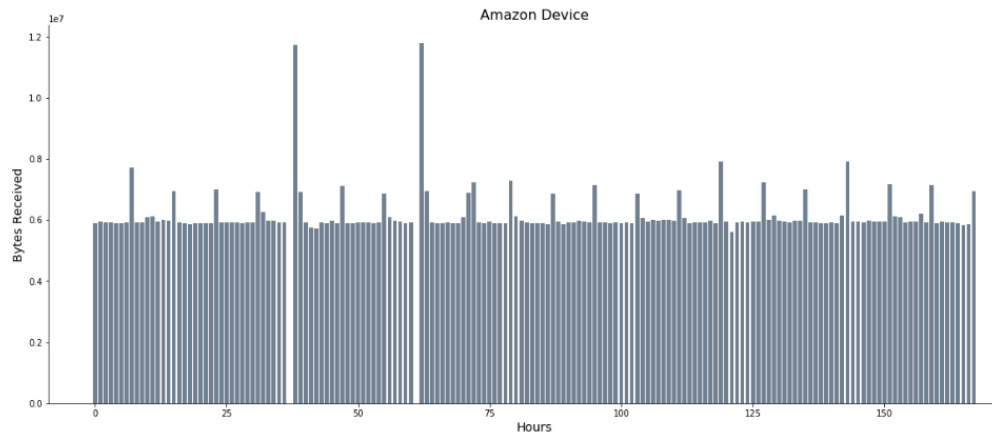


Figure 9: Example of the time series with a clear pattern of bytes received by the router over time for an Amazon device

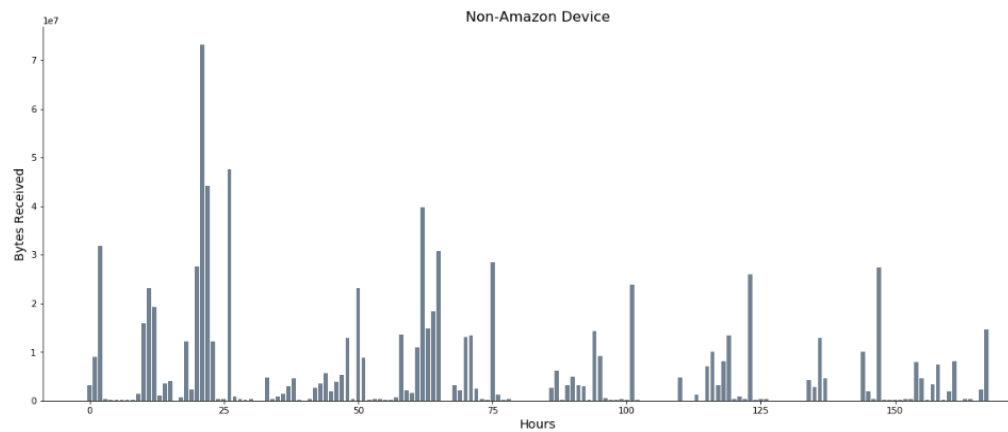


Figure 10: Example of the time series of bytes received by the router over time for a non-Amazon device

In this way, the main goal is to check if it would be possible to build an accurate machine learning model to identify the Amazon devices, based on the variable bytes received by the router over time. If so, one could recognize these devices more accurately and also have an idea of the most important characteristics that make them distinguishable from the remaining ones, for future exploration and association of a signature pattern to them. This would additionally allow to set an alarm when the expected pattern is not verified, leading to further investigation.

Along the process of developing the model, two main challenges can be pointed out: the lack of field data because it was not possible to access to more data for this purpose; and the quality of the data, as, due to the nature of the devices, they can be frequently disconnected, causing a high number of missing values. Regardless of the measures taken to mitigate these challenges, there will always be an impact on the final results obtained.

4.2.1. Motivation

The device fingerprinting project has an academic nature and its purpose is to provide more accurate knowledge on which devices are used in each home network to the service providers. Currently, the information on the maker of a device is not always available and thus, it is necessary to find an approach that could provide a truthful identification of the manufacturer of these devices. A solution in study is to develop a model that clearly associates certain characteristics of the internet usage to the Amazon devices. In this way, it would be possible to identify more precisely the devices, at least the ones from Amazon, even when the maker is not trackable.

Therefore, the main takeaway of this project will be to verify if a machine learning model that uses the bytes received by the router over time, as an indicator of the network usage of a device, is able to identify the Amazon devices precisely. If successful, the study can be deepened with the analysis of the characteristics that make these devices distinguishable from the remaining ones in order to associate a pattern to them.

4.2.2. Data Description

The provided dataset is composed by a total of 3768 observations (or devices) and 169 columns: the binary target, that assumes the value 1 if the respective device is from Amazon and 0 otherwise; and the remaining 168 columns that have the information of the bytes received by the respective router per hour and along one week (from 08/24/2019 to 08/31/2019).

The first step taken was the elimination of duplicated information present in the original dataset in order to prevent data leakage, this is, to avoid having the same information simultaneously on the train and test set and/or having several repeated information in the training set. These scenarios can contribute to the lack of diversity, to overfitting in this subset of data and to an unrealistic performance of the model, and therefore, should be mitigated. In this way, it was found 32 rows with duplicated information. None of these devices were from Amazon and a big slice of them consists of constant arrays with the value 0. In sum, these observations hold only 0.85% of the dataset and by being eliminated the original dataset remained with 3736 devices.

Afterwards, the next decision taken corresponds to the splitting of the original dataset into the train and the test sets. This was done in a way where the first set holds 70% of the observations (which corresponds to 2615 devices) and the second set holds the remaining 30% of the observations (which corresponds to 1121 devices). In addition, both sets have a similar distribution of the values as well as the same proportion of Amazon devices (around 5%), to avoid, for instance, having a test set only with non-Amazon devices. From this point forward, all the conclusions are taking into consideration only the training set.

After the exploratory data analysis, it was possible to check that there is no errors nor inconsistencies in the data, such as negative values. It was also verified that there are devices that present very high values in some hours, but as the respective values were plausible, these observations ended up being considered just as extreme values.

At this point, two major risks were detected: a) the high number of missing values and b) the degree of imbalance of the dataset.

In what concerns to a), the big majority of the devices have at least one missing value throughout the whole week, as perceived in Figure 11, and there are not devices where all the values are null. Missing values occur when there is no reported value from the router, which can happen when there is a data collection problem from the service provider side, or when the devices are disconnected to the network, either intentionally (for instance, users may disconnect the router when on holidays) or not (for example, in the presence of network failures). However, it is not possible to state the real cause. Given that a high percentage of the observations in the dataset has at least one missing value, dropping these observations did not seem to be a valid solution. In addition, 0 means that no information was exchanged between the device and the router (although the device is connected), and thus it differs from having a null value. Hence, the possibility of fulfilling the missing values with 0 was also discarded. In this way, three solutions were taken into consideration in order to mitigate this problem: 1) fulfill with -1, which is an impossible value with no meaning in this context, representing the value for the missing data, 2) using mean imputation which is a common technique used across many types of problems and datasets, and 3) fulfill with a mix of the previous (LOCF) and next values (NOCB) which is a common technique used in time series data.



Figure 11: Number of missing values on the train set. A high number of devices have at least one missing value.

Regarding b), as it is shown on Figure 12, only 5,12% of the observations belong to Amazon devices. This has an influence in the learning process because the model does not see enough examples from this minority class, and thus it may not learn the characteristics from these devices. Once that it was not possible to obtain more field data to use for this purpose, to minimize the imbalance problem were used methods that aim to create synthetic observations based on the existing data. In this matter, the over-sampling techniques were prioritized over the under-sampling techniques because the last ones have the disadvantage of causing information loss, in particular when used alone, which is something that was avoided as the original dataset is already not very large. In addition, there is also the option of conducting a cost sensitive analysis by defining a specific weight to each class and thus increasing the importance of the minority class on the loss function used by the algorithms, without the need of creating artificial observations. In this way, in order to increase the number of Amazon devices, as well, as to bring diversity into the dataset, the following techniques were taken into consideration: 1) SMOTE, 2) ADASYN and 3) cost sensitive learning (using the class weight option).

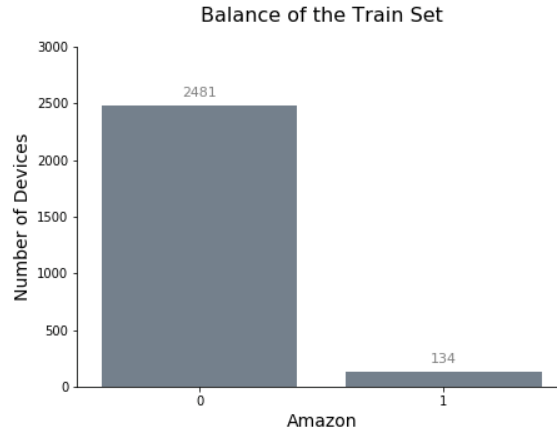


Figure 12: Train set data imbalance. Only 5% of devices are Amazon devices.

Once that these threats to the quality of the data were identified, the analysis can continue with the implementation of the mentioned approaches to soften them and the training phase of the models.

4.2.3. Model and Analysis

Given the different combinations of paths that can be pursued to mitigate the problems of having a big number of missing values and an imbalanced dataset, several scenarios were built. The differences between them can be seen in Table 1. For each of the scenarios, the pipeline sequence of steps, which will be detailed below, is similar as well as the trained machine learning algorithms: Decision Trees, Random Forest, Adaboost, XGBoost, Logistic Regression and SVM.

Table 1: Differences between the scenarios regarding the followed techniques to impute the missing values and to balance the data.

Scenario	Missing Values Imputation	Technique to balance the dataset
A	-1	SMOTE
B	-1	ADASYN
C	-1	Cost Sensitive
D	Mean Value	SMOTE
E	Mean Value	ADASYN
F	Mean Value	Cost Sensitive
G	LOCF & NOCB	SMOTE
H	LOCF & NOCB	ADASYN
I	LOCF & NOCB	Cost Sensitive

For each of the scenarios above, the splitting of the original dataset was followed by the imputation of the missing values with the respective technique and by the extraction of features from the time series by means of the `extract_features()` function from `tsfresh` package (see Chapter 3). The extracted features are from different natures, ranging from basic statistical ones such as mean, minimum and maximum values, to advanced features like the coefficients from the Fourier transformations.

Afterwards, a pipeline to prepare the data to input into a model was built. This pipeline is composed of several steps, starting with the selection of the relevant features on the train set (which will be later filtered on the test set). This is accomplished by the `tsfresh` package's `select_features()` function. Then, in the scenarios where SMOTE and ADASYN methods are applied, the variables were normalized before, in order to ensure that all had the same weight (as they rely on distance-based methods which are scale dependent) and inversed again after. When using these methods, it was considered a number of neighbors of 3 and two balance rates: a perfect balance of 50% of observations for both the minority and majority classes; and a balance of 40% of observations for the minority class and remaining 60% for the majority class. Additional steps were added when training logistic regression and SVM: one of feature selection using RFE, applied before SMOTE, in order to reduce the number of selected features to 20 to avoid overfitting and unnecessary complexity; and a step to rescale the data (either normalization or standardization) in order to improve their performance as distances are calculated. To note that it was also tested the performance of these algorithms without these additional options. In addition, care was taken, especially when dealing with Logistic Regression and SVMs, to identify and eliminate any feature that shown a high correlation (> 0.9) with other(s), in order to avoid the multicollinearity problem. The following step of the pipeline comprises of the model's training phase.

Regarding the selected group of algorithms to use, two factors were key: the ease of interpretation of the final results and the variety of exploring different types of relationship (linear and non-linear). In this way, the choice was reduced to tree-based algorithms, Logistic Regression and SVM. As for the tree-based algorithms, these consisted of Decision Tree, Random Forest, AdaBoost and XGBoost; thus, ranging from simpler forms to more complex and high-performance ones.

The training phase of the models includes the hyperparameter tuning, which was accomplished by means of stratified k-fold cross validation grid search over a parameter grid. The k-fold cross validation is a procedure to evaluate the performance of the model on unseen data and generally results in less biased estimations. For this, the training set was split into 4 folds, all with the same proportion of observations from the minority class, and for each combination, one-fold was hold out as the validation set and the remaining 3 as the train set. In this way, the pipeline described above was incorporated inside the k-fold process to avoid any possible data leakage to the validation sets created inside the k-fold cross validation and thus, to ensure they resemble the independence from the train set as much as possible, just like the test set. To note that whenever features were selected on the train set, the same variables were filtered on the validation set. For each grouping of folds, after the steps from the pipeline were applied, the grid search started to explore the best combination of values for the hyperparameters. To do so, the model was trained on the train set using all possible combinations of hyperparameter's values, given the defined grid of hyperparameters, and later the performance was evaluated on the respective validation set. The optimization was made in order to maximize the f1-score on the validation set. The grid search then returns the combination of values for the hyperparameters that present the best average performance along the validation, this is the combination that returns the highest average f1-score.

In what concerns to the grid search, the tuned hyperparameters differs according to the algorithm (see Table 2). These hyperparameters are the ones shown to have a higher impact on the training results. To note that the class weight hyperparameter (scale_pos_weight for the XGBoost) was also included in the grid on the respective scenarios, where different rates were tested, including the inverse of the class distribution. Moreover, for each of the hyperparameters, several values were used, including the default ones and the different options available (for example, the criterion for tree-based models can assume the values 'Gini' and 'Entropy', thus both of these options were used on the grid search). These values were changed accordingly with the need shown by the obtained results. For instance, when overfitting was present, the grid values were adjusted in order to minimize it. In the case of tree-based algorithms, the possible values for the maximum depth, maximum features and minimum samples per leaf were reduced, as well as the number of estimators and the learning rate when these were available options. In the opposite case, when the model was showing some underfitting, these values were increased. In addition, when training the AdaBoost algorithm, it was used a simple decision tree, with a maximum value of maximum depth of 3, as the base learner. In what concerns to the XGBoost, the added hyperparameters eta, gamma, and lambda were also taken into consideration to prevent overfitting. For the remaining algorithms the number of iterations and the C hyperparameter were modified accordingly. In the case of the SVM, only the linear kernel was used in order to obtain the importance of the features.

Table 2: Grid search tuned hyperparameters per algorithm.

Algorithm	Tuned Hyperparameters
Decision Tree	• Criterion • Max_depth • Max_features • Min_Samples_Leaf
Random Forest	• Criterion • Max_depth • Min_Samples_Leaf • Max_samples • Max_features • N_estimators
AdaBoost	• N_estimators • Learning_rate
XGBoost	• Eta • Gamma • Max_depth • N_estimators • Lambda
Logistic Regression	• Solver • C • Max_iter
SVM	• C • Max_iter

Once the hyperparameters were optimized, the pipeline was replicated to the totality of the train set and the models were trained with the respective best hyperparameters and seeing all the training data as well. The model was then used to predict the device maker of the observations in the test set and the obtained results are resumed in the next points. All of the reasoning described was followed in each scenario.

4.2.4. Results and Discussion

After applying the pipeline steps and training the models in each scenario, the obtained results lead to several conclusions. Firstly, the measures taken to address the missing values issue and the imbalance of the data had an impact on the selected features and the total number of observations, respectively. In this way, when using -1 as imputation value for the missing data, the number of selected features was higher than the remaining methods (see Table 3); while the number of observations varied from around 4100 or 4900, depending on the technique (SMOTE or ADASYN) applied and the balance rate (50% or 40% of positive observations).

Table 3: Number of selected features accordingly with the missing values imputation method.

Missing Values Imputation	Number of Selected Features
-1	266
Mean Value	112
LOCF & NOCB	148

In what concerns to the results achieved by the models, it is possible to separate the scenarios into two groups. In the scenarios that did not follow a cost-sensitive analysis, the models show non-trustable results as overfitting is present. Although measures were taken to reduce the effect of overfitting, it was not possible to eliminate it completely. This can be due to the noise/randomness introduced by the oversampling techniques. In Figure 13 can be seen the difference between the evaluation measure f1-score in the train and test set, of the pruned decision tree models in the different scenarios and for the different balance rates. To note that the remaining algorithms had a similar behavior as shown in Figure 13. Nevertheless, it was also checked that for the majority of the models, the results on the test sets were slightly improved when using a 0.4 rate balance for the oversampling and the missing values were imputed with -1. On the other side, there were not big differences between the oversampling techniques (SMOTE vs ADASYN) and the imputation of the missing values with -1, the mean value or the previous (LOCF) and next values (NOCB). In these scenarios and ignoring the overfitting, the tree-based models overperformed the logistic regression and SVM models on the test set. These last two algorithms indicated better results when the data was standardized and only 20 features were inputted.

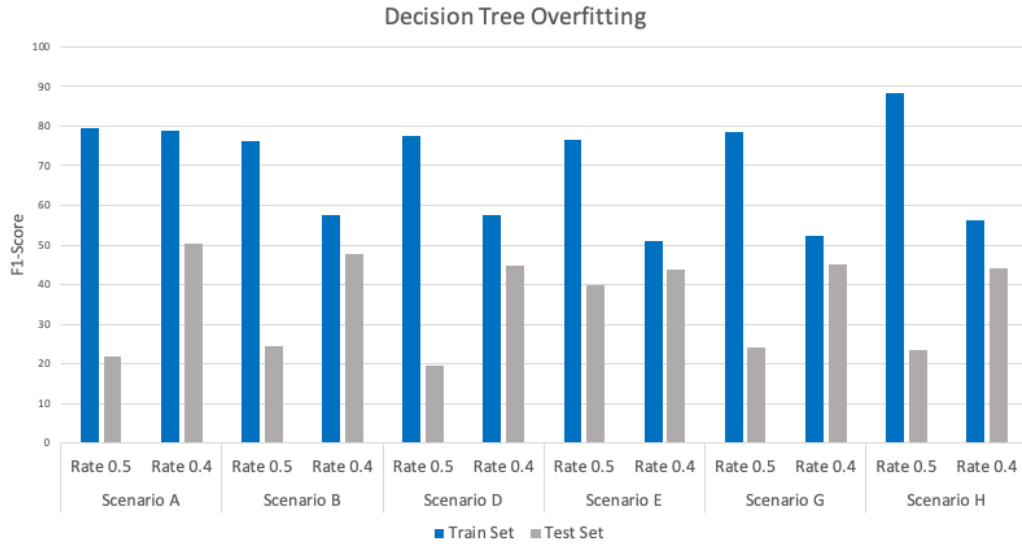


Figure 13: Example of the overfitting of the decision tree models in the scenarios where SMOTE and ADASYN were followed to balance the data and for different balancing rates. The models have a good performance on the train set and behave poorly on the test set.

The best results in terms of stability were observed in the scenarios where the cost-sensitive analysis was conducted (independently of the followed technique to impute the missing values). In these scenarios it was possible to control the overfitting through the hyperparameters tuning. Hence, after comparing the performance of each of the models in these scenarios on the unseen data, one can conclude that the tree-based algorithms present better results than the logistic regression and SVM (see Figure 14, where the results from logistic regression and SVM are considered using the standard feature scaling and RFE feature selection). Although there are not major differences between the performance of the tree-based models, the random forest from scenario C achieved a slightly higher f1-score on the test set, and thus it was considered the best model. Moreover, the achieved best model has the highest AUC on the precision-recall curve (0.60), thus maximizing both precision and recall, when comparing with the remaining models from the same scenario (see Figure 15).

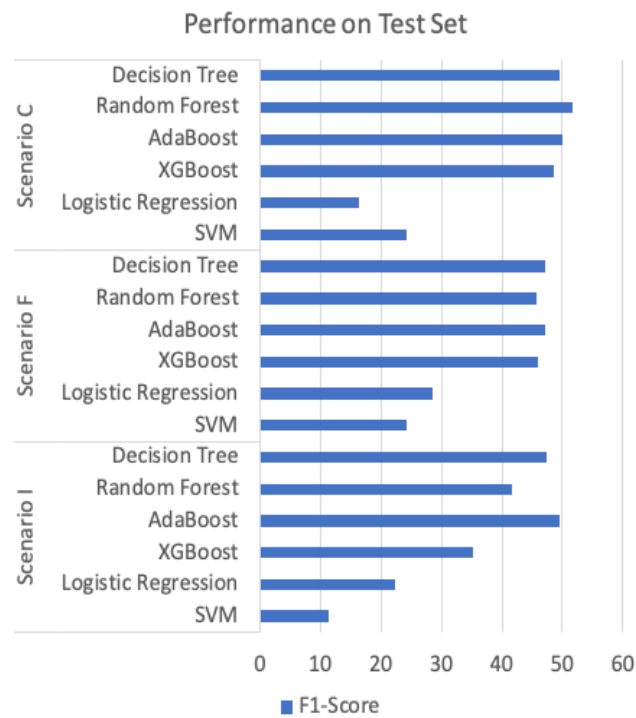


Figure 14: Performance on the test set of each model following a cost-sensitive analysis. The Random Forest model from scenario C is the only model with f1-score above 50%.

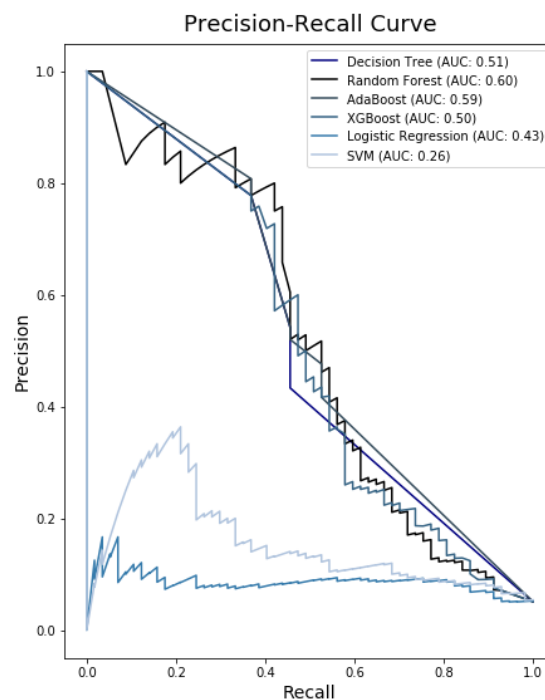


Figure 15: Precision-Recall curve on test set of the models in scenario C. The tree-based models show better performance than logistic regression and SVM.

Looking to the best model in more detail, it is possible to check that there is not overfitting and the results are stable along different seeds as well (see Figure 16). From Table 4, is possible to see that the model still shows some difficulties to identify correctly the Amazon devices, having a recall below 50% (38,6%) and a precision of 78,6%. The best values for the hyperparameters resulting from the grid search can be seen on Table 5. These values allowed for a better performance when comparing with a base random forest model with the default values for all the hyperparameters that classifies all the observations to the majority class.

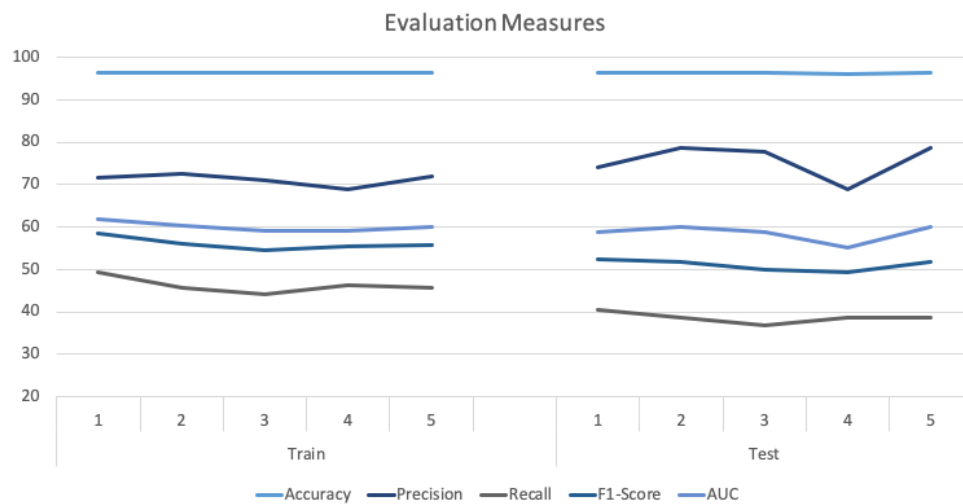


Figure 16: Evaluation measures on train set and on test set along 5 different seeds. The performance of the best model on the test set does not differs significantly from the performance on the train set.

Table 4: Confusion matrix of the best model. The number of FN remains high.

		Predicted:	
		Non-Amazon	Amazon
Actual:	Non-Amazon	1058	6
	Amazon	35	22

Table 5: Tuned hyperparameters of the best model.

Hyperparameter	Value
Criterion	Gini
Max_depth	2
Max_Samples	0.8
Max_Features	0.8
N_estimators	100
Class_Weight	{0: 1.0, 1: 5.0}

Moreover, accordingly with the best model, the most important features to distinguish the Amazon devices from the non-Amazon devices are majorly the number of peaks with a support of 5 (with 60,8% of the worth), this is the number of peaks of bytes received by the router with a value higher than the respective values from the previous and later 5 hours. Although with significant less importance, the autocorrelation of lag 8 (9,3% of the worth) and quantile of 40% of the timeseries (6,3% of the worth) can be highlighted as part of the top 3 most relevant features.

4.2.5. Conclusions

The path to develop a machine learning model for device fingerprinting came with an exploratory aspect of understanding if it would be possible to associate a pattern of wi-fi usage of a device to the respective maker, taking the Amazon devices and the bytes received per hour as example.

In fact, taking into consideration all the steps described in the current chapter, where feature extraction plays a central role to transform the classification of time series data into a traditional classification problem by removing the time dimension and different algorithms were applied as well as several techniques to reduce the impact of the imbalance of the data and the number of missing values, it was possible to find a potentially good approach to take. In this matter, the methods to balance the data seem to have a high impact in the performance of the models being the cost sensitive analysis the one that presented better results, while regarding the imputation of the missing values, there was not a method that stand out among the ones applied. In what concerns to the algorithms, the tree-based models presented a higher and more stable performance.

However, the achieved performance of the best model was considered not sufficient to identify accurately the Amazon devices, as it shows some ambiguity in identifying correctly those devices, leading to the impression that some of the devices may present a common pattern while others not.

In conclusion, the project introduced a process to carry out a time series classification problem and the achieved results may suggest that the variable chosen is not the best indicator for device fingerprinting. In this way, the project did not proceed forward with this configuration, and efforts will need to be made in order to find the best mean of identifying the maker of a given device based on the wi-fi usage pattern of the device.

5. CONCLUSIONS

The project developed during the internship consisted of creating a machine learning model for device fingerprinting, namely, to explore the identification of the Amazon devices based on the bytes received by the router over time in order to associate a pattern with certain characteristics to these devices.

However, it was concluded that the best model did not present reasonable results as it could not easily distinguish the Amazon devices from the remaining devices of different makers which can indicate that either these devices in general do not present a particular pattern and/or the variable chosen (bytes received) may not be the best choice for this purpose.

Either way, the project allowed the coverage of all data science steps, from the data cleaning to the evaluation of the trained models, as well as the exploration of a feature-based analysis for a time series classification problem. In this chapter, some additional notes on the project and the period of the internship are presented.

5.1. CONNECTION TO THE MASTER PROGRAM

The master program provided the foundation knowledge on concepts, techniques, tools and algorithms to apply on the proposed project that covers all the phases from the traditional end-to-end data science process. In this context, it is possible to highlight the following learned takeaways that helped to meet the final goals of the project:

- Identification of the type of problem in hands and the algorithms that are the best fit for it;
- Build the data science process avoiding common mistakes at each step;
- Identification of the problems that affect the data quality, such as the number of missing values, duplicates and the balance of the data, and possible approaches to tackle them;
- Identification of problems during the training process of a model, such as overfitting or underfitting, and alternatives to prevent it given the algorithm in training;
- Selection of the appropriate evaluation measures of the models' performance considering the type of problem.

5.2. INTERNSHIP EVALUATION

Overall, the internship was an opportunity to apply the data science knowledges presented in the master's program and to have contact with state-of-the-art tools and technologies, while in the context of a company. In addition, it was also a learning experience as the proposed project consisted of a type of problem (the classification of time series data) that was not explored during the curricular year of the master's program, which lead to the search of alternative techniques and models that could be applied in this particular case.

However, the team was starting to introduce data science in the daily work and thus both the know-how and the existence of meaningful projects with the available data and in the domain of data science were limited.

5.3. LIMITATIONS

Due to the nature of the data, having missing values in a considerable number is unavoidable and thus this will have an impact on the quality of the available data. Moreover, the small number of examples of Amazon devices also decreased the quality of the data. The possibility of adding more data could have helped to mitigate the quality issue and allowed the models to observe more examples, however, it was not possible to obtain access to extra field data for this purpose and in useful time. Alongside these, as the project was not a priority some depreciation was felt, and it was difficult to learn the standard data science approaches used in other projects as the team was recent in this area as well. Hence, all of the above points worked in their own way as a limitation while executing the project during the internship.

5.4. LESSONS LEARNED

There are some key points learned during the internship and that will be taken into account in future projects. To start, real-world field data is not perfect and thus all the steps taken before inputting it to a model should be carefully carry on as they can bias the final conclusions and one should also consider in this stage that not all the options can be tested and choices need to be made. The domain knowledge is at this point a must in order to understand the data, detect any inconsistency, as well as to identify possible future projects to explore.

In addition, one should be aware that the commonly used techniques to prevent issues that the data may present are not always the best fit for the concerned problem and data, while the least favorable alternatives are worth the try and may show surprising results.

In what concerns to dealing with time series data, a feature-based analysis seems an interesting approach where feature engineering takes a central role to be able to select simpler (yet powerful) algorithms and consequently ease the training process and the interpretation of the models which is particularly important when in the context of a company.

5.5. FUTURE WORK

In line with the conclusions obtained in the present project and the still existing need of accurately identify the devices associated to a router, there are some steps left to be explored in the future.

Firstly, this project was focus on a feature-based analysis, hence it could be interesting to study the performance of an approach based on the real values of the time series (without the feature extraction stage) and compare the results with the ones obtained in this project.

Secondly, the exploration of different methods, namely for the imbalance present in the data (for example, under-sampling techniques or threshold moving) and for the missing values imputation (in this matter, more complex techniques can be taken into account, such as the interpolation of missing data) are welcomed. Gather more field data can potentially help in both points as well. It could also be relevant to study alternative algorithms than the ones mentioned and a more detailed feature extraction process.

Thirdly, one can consider performing small mutations on the scope of the project, such as considering other variable(s) than the bytes received (or add new variables), that may reflect better the existence of a potential pattern of network usage. Expand the classifiers to identify the remaining makers can also be an option for future developments.

Lastly, if reasonable results are achieved, it would also be challenging to implement the achievements into the company product.

6. BIBLIOGRAPHY

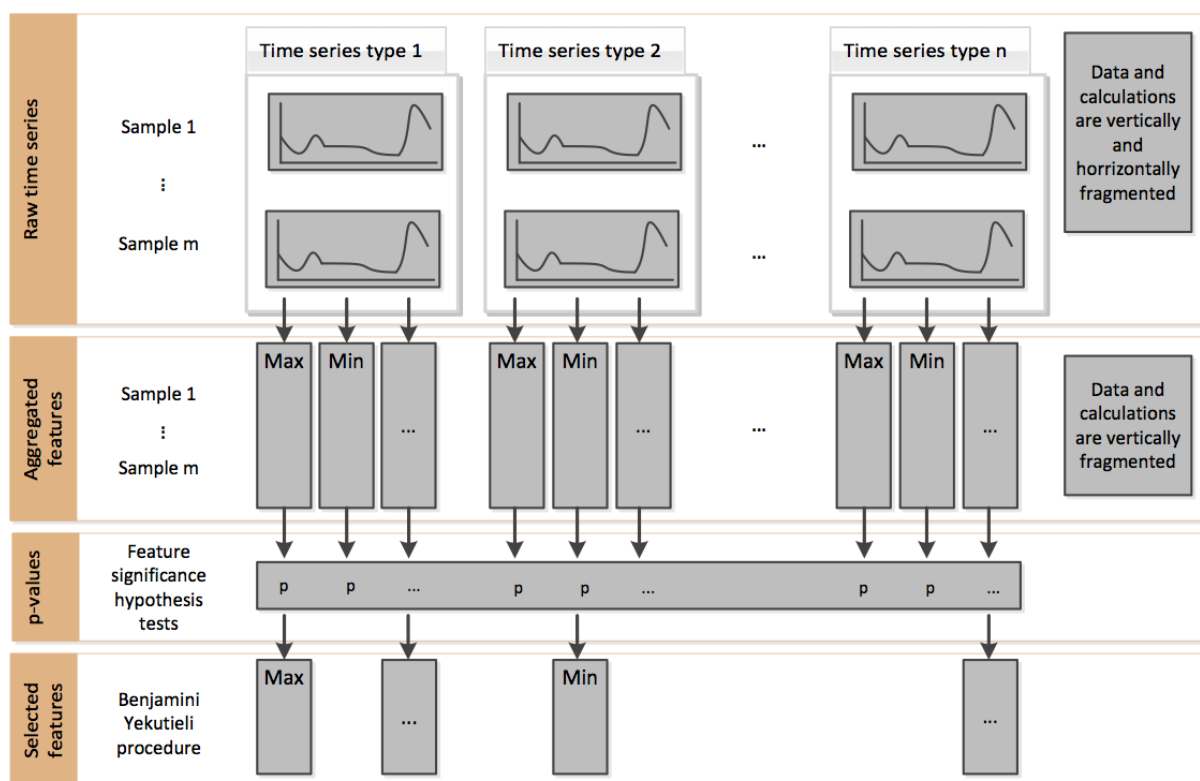
- [1] T. Mitchell, *Machine Learning*, vol. 17. 1997.
- [2] M. Svensson and J. Söderberg, "Machine-learning technologies in telecommunications," *Ericsson Rev. (English Ed.)*, vol. 85, no. 3, pp. 29–33, 2008.
- [3] J. Han, M. Kamber, and J. Pei, *Data mining: Data mining concepts and techniques*, 3rd ed. 2012.
- [4] J. M. Engels and P. Diehr, "Imputation of missing longitudinal data: A comparison of methods," *J. Clin. Epidemiol.*, vol. 56, no. 10, pp. 968–976, 2003.
- [5] M. C. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.
- [6] B. Ghogh et al., "Feature Selection and Feature Extraction in Pattern Analysis: A Literature Review," 2019.
- [7] J. Gosic and T. Łukaszuk, "Application of the Recursive Feature Elimination and the Relaxed Linear Separability Feature Selection Algorithms To Gene Expression Data Analysis," *Adv. Comput. Sci. Res. Appl.*, vol. 10, pp. 39–52, 2013.
- [8] C. X. Ling and V. S. Sheng, "Cost-Sensitive Learning and the Class Imbalance Problem," *Encycl. Mach. Learn.*, pp. 231–235, 2008.
- [9] N. Thai-Nghe, Z. Gantner, and L. Schmidt-Thieme, "Cost-sensitive learning methods for imbalanced data," *2010 Int. Jt. Conf. Neural Networks*, pp. 1–8, 2010.
- [10] M. S. Shelke, P. R. Deshmukh, and V. K. Shandilya, "A Review on Imbalanced Data Handling Using Undersampling and Oversampling Technique," *Int. J. Recent Trends Eng. Res.*, vol. 3, no. 4, pp. 444–449, 2017.
- [11] W. P. K. Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, "SMOTE: Synthetic Minority Over-sampling Technique," *J. Artif. Intell. Res.*, no. 16, pp. 321–357, 2002.
- [12] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," *Proc. Int. Jt. Conf. Neural Networks*, no. July 2008, pp. 1322–1328, 2008.
- [13] F. Chollet, *Deep Learning with Python*. 2018.
- [14] J. Peng, K. Lee, and G. Ingersoll, "An Introduction to Logistic Regression Analysis and Reporting," *J. Educ. Res. - J EDUC RES*, vol. 96, pp. 3–14, 2002.
- [15] E. Shtatland, K. Kleinman, and E. Cain, "One more time about R2 measures of fit in logistic regression," *NESUG 15 Proc.*, vol. 15, 2002.
- [16] V. Bewick, L. Cheek, and J. Ball, "Statistics review 14: Logistic regression," *Crit. Care*, vol. 9, pp. 112–118, 2005.
- [17] "Linear Models." [Online]. Available: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression. [Accessed: 19-Dec-2019].
- [18] G. King and L. Zeng, "Logistic Regression in Rare Events Data," *Polit. Anal.*, vol. 9, 2002.
- [19] L. Rokach and O. Maimon, "Decision Trees," in *Data Mining and Knowledge Discovery*

- Handbook*, O. Maimon and L. Rokach, Eds. Boston, MA: Springer US, 2005, pp. 165–192.
- [20] Kai Ming Ting, “An instance-weighting method to induce cost-sensitive trees,” *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 3, pp. 659–665, May 2002.
 - [21] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” 2016, pp. 785–794.
 - [22] “xgboost Documentation.” [Online]. Available: <https://xgboost.readthedocs.io/en/latest/index.html>. [Accessed: 16-Jun-2020].
 - [23] T. Evgeniou and M. Pontil, “Support Vector Machines: Theory and Applications,” 2001, vol. 2049, pp. 249–257.
 - [24] A. Iranmehr, H. Masnadi-Shirazi, and N. Vasconcelos, “Cost-sensitive support vector machines,” *Neurocomputing*, vol. 343, pp. 50–64, 2019.
 - [25] J. Brank, M. Grobelnik, N. Milic-Frayling, and D. Mladenic, “Feature Selection Using Support Vector Machines,” *WIT Trans. Inf. Commun. Technol.*, vol. 28, 2002.
 - [26] K. Buza, “Time series classification and its applications,” *ACM Int. Conf. Proceeding Ser.*, no. June, 2018.
 - [27] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, “Handling imbalanced datasets : A review,” *GESTS Int. Trans. Comput. Sci. Eng.*, vol. 30, 2006.
 - [28] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, “Feature-based Classification of Time-series Data,” *Int. J. Comput. Res.*, vol. 0056, 2001.
 - [29] R. Devaki and V. Kathiresan, “Credit Card Fraud Detection using Time Series Analysis,” *IJCA Proc. Int. Conf. Simulations Comput. Nexus*, vol. ICSCN, no. 3, pp. 8–10, 2014.
 - [30] N. Mehdiyev, J. Lahann, A. Emrich, D. Enke, P. Fettke, and P. Loos, “Time Series Classification using Deep Learning for Process Planning: A Case from the Process Industry,” *Procedia Comput. Sci.*, vol. 114, pp. 242–249, 2017.
 - [31] S. Dheepadharshani, S. Anandh, K. B. Bhavinaya, and R. Lavanya, “Multivariate time-series classification for automated fault detection in satellite power systems,” *Proc. 2019 IEEE Int. Conf. Commun. Signal Process. ICCSP 2019*, pp. 814–817, 2019.
 - [32] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2014.
 - [33] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, “Feature-based Classification of Time-series Data,” *Int. J. Comput. Res.*, vol. 10, pp. 49–61, 2001.
 - [34] L. Ge and L.-J. Ge, “Feature extraction of time series classification based on multi-method integration,” *Optik (Stuttg.)*, vol. 127, no. 23, pp. 11070–11074, 2016.
 - [35] M. Christ and Blue Yonder GmbH, “tsfresh Documentation,” 2018. [Online]. Available: <https://tsfresh.readthedocs.io/en/latest/>. [Accessed: 15-Dec-2019].
 - [36] C. Herff and D. J. Krusienski, “Extracting Features from Time Series,” in *Fundamentals of Clinical Data Science*, P. Kubben, M. Dumontier, and A. Dekker, Eds. Cham: Springer International Publishing, 2019, pp. 85–100.
 - [37] P. J. Olver, *Introduction to Partial Differential Equations*. Springer International Publishing, 2013.

- [38] Y. Wang, "Frequencies of the Ricker wavelet," *GEOPHYSICS*, vol. 80, pp. A31–A37, 2015.
- [39] A.-H. Najmi, J. Sadowsky, O. Morlet, and W. Transform, "The Continuous Wavelet Transform and Variable Resolution Time-Frequency Analysis," vol. 18, 1997.
- [40] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time Series Feature Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018.
- [41] "hctsa package." [Online]. Available: <https://hctsa-users.gitbook.io/hctsa-manual/>. [Accessed: 15-Dec-2019].
- [42] B. D. Fulcher and N. S. Jones, "hctsa: A Computational Framework for Automated Time-Series Phenotyping Using Massive Feature Extraction," *Cell Syst.*, vol. 5, no. 5, pp. 527-531.e3, 2017.
- [43] "CatBoost Documentation." [Online]. Available: <https://catboost.ai/docs/>. [Accessed: 01-Aug-2020].
- [44] "LightGBM Documentation." [Online]. Available: <https://lightgbm.readthedocs.io/en/latest/>. [Accessed: 01-Aug-2020].

7. ANNEXES

Annexure 1: tsfresh's process to extract relevant features



Source: https://tsfresh.readthedocs.io/en/latest/text/feature_filtering.html

Accessed: 19-Dec-2019

