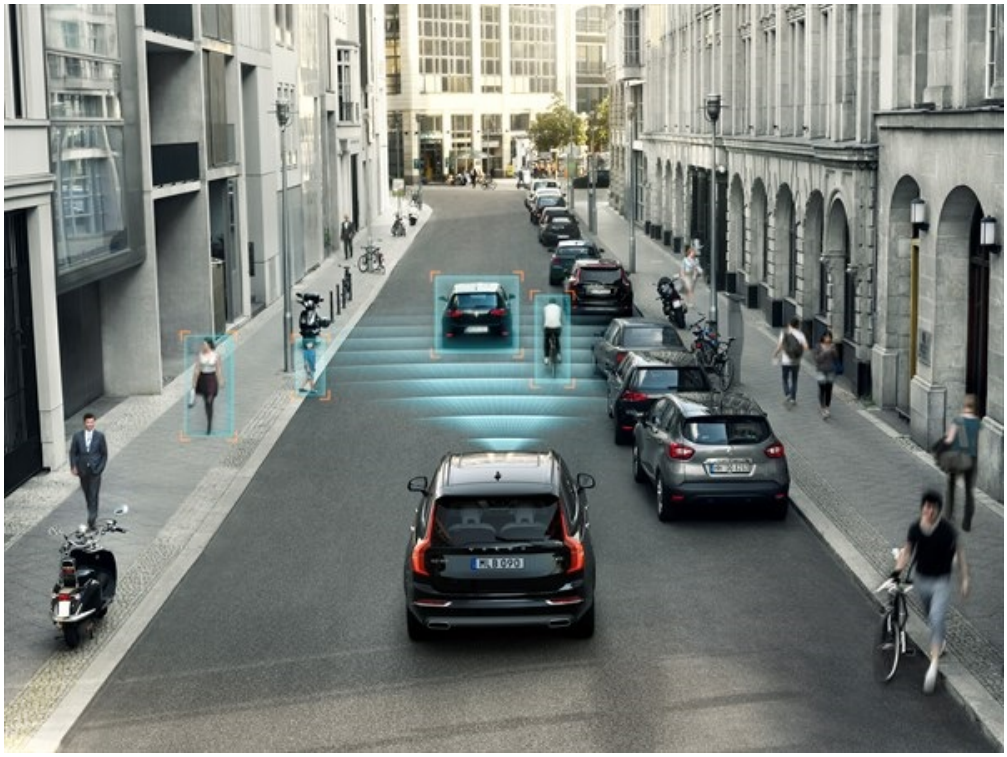


City Safety Event Classification using Machine Learning

A binary classification of a multivariate time series sensor data

Master's thesis in Applied Data Science

Natalia Jurczyńska

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

Master's thesis 2019

City Safety Event Classification using Machine Learning

A binary classification of a multivariate time series sensor data

Natalia Jurczyńska

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2019

City Safety Event Classification using Machine Learning
A binary classification of a multivariate time series sensor data
Natalia Jurczyńska

Supervisor: Alexander Schliep, Department of Computer Science and Engineering
Advisors: Srikar Muppirisetty, Andreas Runhäll and Sohini Roy Chowdhury, Volvo Cars
Examiner: Christos Dimitrakakis, Department of Computer Science and Engineering

Master's Thesis 2019
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The City Safety system visualisation. Source: Volvo Cars, internal resources.

Typeset in LaTeX
Gothenburg, Sweden 2019

City Safety Event Classification using Machine Learning
A binary classification of a multivariate time series sensor data
Natalia Jurczyńska
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

City safety technology aims to reduce vehicle collisions using activated warnings and braking based on automated detection of environmental threats. However, automatic detection of tentative collisions may differ from driver perception, leading to false positive activations. This work analyses vehicle on-board sensor suite in the event of City Safety activations and learns the optimal features responsible for activation classifications. From the 152 activation events, 8 second multivariate logs containing 316 signals are mined to achieve around 98% of ROC_AUC score in event classification. Thus, supervised and semi-supervised classifications significantly bridge the gap between automated and human perception for autonomous driving functionalities.

Keywords: Data Science, machine learning, time series analysis, binary classification, data pre-processing, feature engineering, thesis.

Acknowledgements

I would like to express my gratefulness to all of the people who were involved in this thesis work.
Firstly, I would like to thank my university supervisor, Alexander Schliep, who provided valuable and constructive suggestions during the whole period of this research work. Thank you for your time and help.
Secondly, I send my special thanks to my company supervisors, Andreas Runhäll, Srikar Muppirisetty, and Sohini Roy Chowdhury, for sharing a lot of domain knowledge, useful critiques and guidelines. Your advice, especially during data preprocesses, has been very much appreciated.
Furthermore, I wish to acknowledge the university and academic staff for the last two years of studies. In particular, I am grateful for all of the help and assistance that I received from my program supervisor, Richard Johansson.
Additionally, I would like to express my great appreciation to Volvo Cars for giving me the opportunity to develop a machine learning project, which can help in improving the existing system and, eventually, lead to an increase in car safety.
Finally, I would like to thank my family and my friends for supporting me during that time. Exclusively, thanks Weronika and Fionn for corrections and your feedback. Dziękuję!

Natalia Jurczyńska, Gothenburg, June 2019

Contents

List of Figures

List of Tables

1

Introduction

Advances in technology have always influenced human lives and created plentiful opportunities for boosting effectiveness in many different areas of applications such as education, manufacturing or business. Besides specific uses in concrete domains, technologies, if carefully employed, can also lead to an increase in safety. This is in particular crucial in industries, such as the automotive industry, where the risk for injuries or casualties is higher than normal.
Nowadays, the automotive industry is moving towards autonomous driving and Volvo, among other companies like Mercedes-Benz or Tesla, is one of the active research organisations in this area. However, before an autonomous dream will be achieved, all its small elements have to be flawless. City Safety is one of them.
In a nutshell, City Safety is a technology introduced by Volvo and is an umbrella name for a mechanism that aims to reduce car collisions. The system sends a set of warnings to a driver if an object has been detected in front of a car that may be the cause of the potential collision. Ultimately, if the driver does not respond within a sufficient amount of time, the car should automatically brake (auto-brake), as triggered by City Safety (see Figure 1.1). Hence, if a system has sent a warning or started to auto-brake, this is understood as a City Safety activation. The area in front of a car is explored by sensors and cameras. However, on rare occasions, it may happen that the sensors' perception of the environment does not correspond to the driver's perception. This can lead to City Safety activation that is perceived as false.

1.1 Goal

The primary goal is to classify with high accuracy whether the City Safety system was activated correctly (there was a potential collision, and the system activation prevented it) or incorrectly (there was no potential collision) in a given road event. A road event is represented by a multivariate time series with a length of 8 seconds. More information about the data can be found in section 3.4.

The secondary goal is to have a deep understanding of what influences misclassifications. What are the most significant features that are associated with misclassification? Answering these questions will help to improve the system in the future.

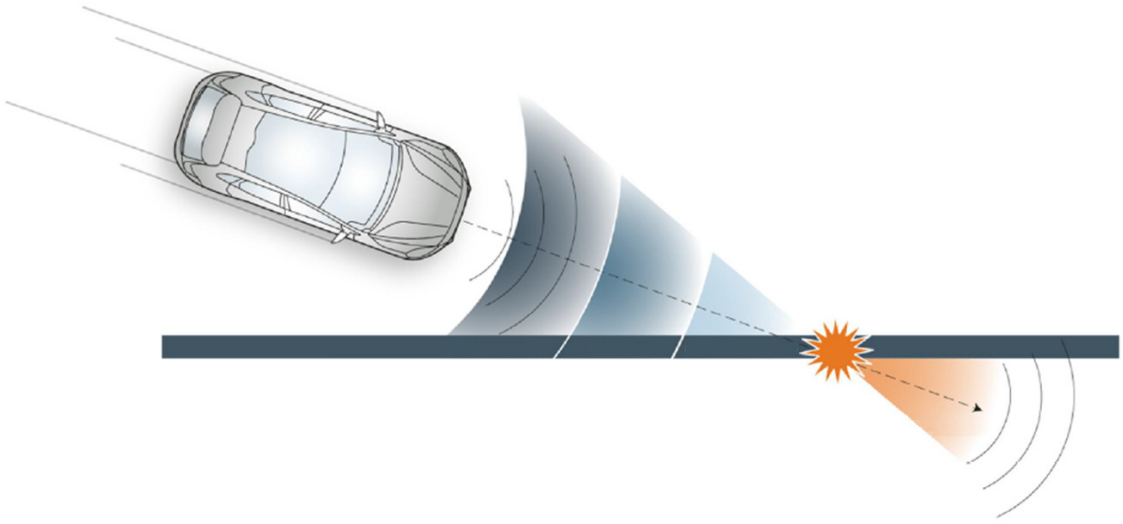


Figure 1.1: City Safety visualisation. Source: [46]

1.2 Approach

In this thesis, binary classification of multivariate time series is performed. As most of the available data is unlabelled, a semi-supervised learning[1] approach is used. This model is named *the pseudo-labelling model*.
Moreover, two *baseline models* are constructed for a benchmark against the main model. The first baseline model outlines how the false events are classified at the company now. The second baseline model is a simple machine learning model that is applied to almost unprocessed multivariate time series data. For the sake of clarity, the first model is called *the annotator model*, while the second one is called *the baseline model*.

1.3 Motivation for this analysis - *the annotator model*

Firstly, City Safety activations are mostly analysed upon request at present, when there is a need to understand the cause for it. This procedure is extremely time-consuming and inefficient. Human annotation of one event can take up to half of a working day. Additionally, an annotator needs to be trained beforehand, so that his/her decision will be consistent and of high quality. Nevertheless, assuming that the company is already in a possession of such an asset, thus no training or knowledge sharing needs to be done, the accuracy of one event annotation approximates 100%. Such an excellent score is challenging and almost impossible to achieve for an artificially created classifier. Yet, constructing even a simple algorithm that is able to predict false activation with fairly good accuracy can act as a helpful counterpart to human annotation due to significant time reduction.

[1]The definition of semi-supervised learning is given in Problem chapter.

Secondly, this slow pace of event annotations limits the possibility of grouping false events with the same root cause. Being able to cluster similar events would make a good start for improving City Safety system algorithms in the future. Being able to detect which factors influence false activations is a proactive approach for system improvement.
Finally, it seems to be interesting to scrutinise existing machine learning techniques in this problem setup. Using them instead of the traditional approaches may result in detecting new patterns that are not as easy to be noticeable by humans.

1.4 Research questions

This thesis aims to answer the following research questions:

1. Does machine learning, with all its semi-automatic procedures, achieve better performance than a simple model constructed with the help of expert knowledge within this field?
2. Does a semi-supervised approach reach better results than a supervised approach? What are the limitations and assumptions of each of them? What is the confidence level of each of the approaches?
3. What are the most significant signals for a true/false activation City Safety system?
4. To what extent can we believe that labelling of unlabelled events using a semi-supervised approach is adequate? Is the distribution of labelled events a good indicator of inference about the whole population? Which method is best?
5. What dimensionality reduction method performs the best and to what extent it provides meaningful and interpretable results?

1.5 Scope

The thesis is organised into several chapters.
The first chapter, *Background*, describes the related work to this research topic. It outlines the concept of safety in the vehicles and Auto-Braking and describes the devices which collect the data. In the second chapter, *Problem*, Machine Learning principles and the data used for the analysis is explained in detail. The third chapter, *Methods*, gives a summary of methods that are used for the modelling. This includes the description of data pre-processing methods, the selection of learning algorithms and an overview of evaluation techniques. In the next part, *Results*, the outcomes of modelling are demonstrated and discussed. Moreover, the research questions are answered. In the end, the future work recommendations are outlined. Finally, the last chapter, *Conclusions*, summarises this thesis.

2

Background: Auto-Braking

The purpose of this chapter is to introduce the societal and industrial setting of the industry, car safety and Volvo's City Safety System. We indicate current trends in the industry and characterise sensors that gather data. Last but not least, we summarise relevant studies and discuss ethical concerns in this research.

2.1 Road Accidents

According to the World Health Organisation (WHO), 1.35 million people die annually in road accidents. Today, road accidents constitute the 9th leading cause of death and cause 2.2% of all deaths in the world. Additionally, if no action in this area is taken, WHO predicts that road accident will become the 5th leading cause of death by 2030 [68].
These are only a few statistics associated with road crashes and their impact on individuals and societies. Although the aspect of roads safety is complex and there are usually many factors involved, there is always a common facet - a vehicle. For these very reasons, keeping roads safe should be the main concern for automotive companies. Consequently, it is one of the goals of *The 2030 Agenda for Sustainable Development* under the 11th goal of *Sustainable cities and communities*[1] [62].

2.2 Safety in vehicles

Within the automotive industry, we can distinguish two types of safety: (1) passive safety and (2) active safety.
Passive safety is a term that refers to systems installed in vehicles that are called to action during an accident. Therefore, they are not working while normal driving. Examples of passive safety system are seat belts or airbags.
Active safety, on the other hand, is a term that refers to systems that are monitoring the vehicle constantly in the background while driving, hence accidents can be prevented. Examples of active safety system are a driver assistant or a collision warning system. City Safety is an example of an active safety system.
Experts in the automotive industry agree that passive safety system has attained its maturity [10]. That means that supplementary improvements in this area are both expensive and complicated. Therefore, more focus and resources are given for research in developing active safety technologies nowadays.

[1] One of the targets under this goal is to increase safety in the roads.

2.3 City Safety System

The City Safety system consists of three levels: Collision Warning, Brake Assistance and Auto-Brake. As shown in Figure 2.1, each level has a different moment of activation.
The first level is the Collision Warning. It is activated if there is an object detected by sensors which aligns with the predicted path of a host vehicle and which may be a potential threat. If the driver brakes as a response to the Collision Warning, but not enough to avoid a collision, Brake Assistance amplifies the driver induced braking (the second level). Ultimately, if there is no reaction from a host vehicle driver within a sufficient amount of time and a collision is very likely to happen, then the third level is activated, which is an Automatic Brake (Auto-Brake).
In this thesis, an activation of City Safety is understood if the last level (Auto Brake) was activated.

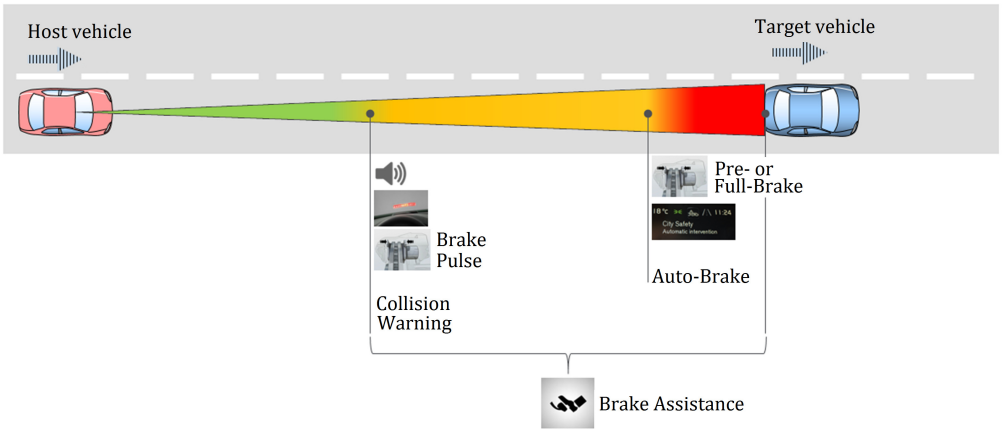


Figure 2.1: City Safety. Source: [46]

2.4 Sensor Data Fusion

As stated by Sangorrin et al. [50], active safety system interventions in road events are automatic, therefore there is a necessity for extremely reliable programs that estimate a potential collision with high accuracy. According to many researchers in this area [33], sensor fusion is a prominent technique in this kind of application.
The idea of sensor fusion is quite simple. Vehicles are equipped with various types of sensors, such as radars, LIDARs or cameras, that explore the area around them. Radar is a radio waves emitter and is used to detect objects and their speed. A LIDAR, on the other hand, is a sensor that uses light for estimating a distance between objects. A camera records a driver perception and, with the help of classification algorithms, a camera recording can be used to classify detected objects. Analysing

outputs from each type of sensors separately lacks a comprehensive understanding of an environment around a vehicle. However, if all of these outputs (called usually raw data) are employed simultaneously and consciously of own advantages and limitations, the overview of the environment might be enhanced. This step refers to Data Fusion and requires an implementation of processing algorithms [50].

In this thesis, the analysis is performed on processed data (not raw data from sensors).

2.5 Related work

The research in the Auto-Braking area is conducted on many levels because the complexity of such systems is relatively high. The first subsection refers to sensors as they gather signal data and, in this thesis, the modelling uses such data. However, to give a broader overview of methods discussed in the community, another technique, hazard analysis, is also presented.

2.5.1 Sensors

As there are different sources of data in the Active Safety system, the analysis of false Auto-Brake activations can be conducted on several stages. For instance, one would be interested in investigating sensors data separately. An example of this kind of approach was presented in 2019 by Saghafi et al. [48]. The researchers aimed to classify events with and without road accidents that were recorded by dash cameras in Taiwan. They decided to apply state-of-the-art deep learning techniques to these video recordings, that is Convolutional Long Short-Term Memory Networks, to image sequence classification. The choice of this method was motivated by preserving the spatial and temporal relationships. Similarly to this thesis, the authors did not have a large sample of labelled data.

In contrary to the first example, Kaempchen et al. [28] investigated false active safety activations in order to propose a new mechanism for an Auto-Brake trigger. They based their work on a novel approach within the sensor fusion domain, thus different road scenarios could be addressed (e.g. rear-end collisions, etc.). Deviating from previous work in this area, they focused on consolidating an orientation of vehicles when estimating the possibility of collision. This paper provided meaningful insights on how active safety systems work in detail and which signals may have the highest influence for a particular road situation.

Last but not least, Rakhshan et al. [44] discussed an important aspect of active safety systems. Depending on individual driving styles, there may be different brake response time expected by drivers. For this reason, the authors suggested algorithms for tuning an active safety system in accordance with the pre-studied estimation of brake response time distribution. According to their analysis, they managed to reduce false activations by more than 50%.

2.5.2 Safety analysis using hazard analysis

Safety analysis can be also tested in an engineering approach, such as risk and hazard analysis. This approach can be employed to evaluate whether an intensively complex IT system is resistant to hazardous factors that may cause a system failure. An example of such analysis was published by Sulaman et al. [57]. The authors compared qualitatively two hazard analysis methods on Forward Collision Avoidance System (FCA)[2] that detect potential system failures. As a result of such hazard analysis, the system's components vulnerability can be addressed, so that IT system reliability is ensured.

In contrast to this thesis, risk and hazard analysis is frequently used before an IT system is released. As City Safety is already installed in new generation cars, there is a need for post-release analysis of past events that are represented by data from sensors. Finding a system's bottlenecks thought data it generates, can help to address issues that the existing IT system faces.

2.6 Active Safety - a step ahead

Active Safety systems are only a part of a big picture of trends noticeable in the car industry. An increasing amount of automation, that is offered to a driver partly to facilitate a journey, intends not only to provide a reliable Auto-Braking technology but also to enable fully autonomous driving in the future.

Although attempts of autonomous vehicle development already began in 1920 [53], providing a vehicle that would act closely as a normal driver was (and still is) challenging and required advanced algorithms that imitated the normal environment. Therefore, the rapid development of such vehicles boosted in the last decades [11] [17] [61], mostly due to simultaneous advancements in computational capabilities, Image Analysis and Machine Learning.

At the moment, there are four levels of vehicles autonomy [15] distinguished in the community:

- Level 0: No automation is installed in cars.
- Level 1: Only some specific functions are automated. For instance, braking assistance. Volvo vehicles with the City Safety system are in this category.
- Level 2: This level requires alignment of two automatic functions, for example, a lane centring[3] and adaptive cruise control[4].
- Level 3: Cars are able to operate almost autonomously with limited help of the driver and convenient transition time.
- Level 4: Fully autonomous cars, no human interaction is needed.

[2]FCA works the same as City Safety.

[3]Lane centring is a system that aims to keep a host vehicle in the centre of the lane while driving.

[4]Adaptive cruise control aims to keep a safe distance between other road users by manipulating a host vehicle's speed.

2.6.1 Challenges

Despite intensive research and ongoing tests in this area, several issues need to be addressed before autonomous driving will become part of our lives.
Firstly, autonomous driving needs solid and transparent regulations made by countries both locally and globally. A wide range of laws has to be reconsidered to provide a rightful judgement in case of an accident with a driver-less car. For example, should it be the same penalty if an accident happens with an unoccupied car, an occupied car or a car carrying children? How to determine to what extent a driver has an influence in almost fully autonomous driving? Moreover, the difficulty in addressing such issues corresponds also how to regulate international law if the journey of an autonomous vehicle passes through different countries?
Secondly, it is commonly believed that autonomous driving will decrease the number of accidents. Ideally, it would be true if all vehicles worked accordingly to the same algorithms, thus there would be no unexpected behaviours in the environment. However, analysing the current trends and the number of companies that attempt to develop this technology, an autonomous driving monopoly seems to be quite dubious to expect. Additionally, driver-less cars will be introduced in cities and highways gradually, so the autonomous driving algorithms have to be extremely reliable in order to operate in a semi-automatic environment. This raises the question if this technology is mature enough to be launched? Moreover, most of these algorithms are based on continuous GPS readings. Therefore, in the near future only highly mapped environments will be able to embrace autonomous cars.
Thirdly, with the initiation of autonomous driving, wise steering of economies will be needed. At the moment, there are around 5 million people in Europe employed in the transportation industry, working in positions such as a bus, taxi or truck driver [4]. Although it is said that the autonomous driving cars and trucks will be instituted gradually, there is a social responsibility of state leaders to govern the economy is a way to provide the smoothest transition as it is possible.

2.7 Ethics

Data ethics is a novel term in the Data Science terminology [18] that refers to proper usage of data. According to the ethical principles, data science projects should follow ten simple rules in order to assure that research is socially responsible. This includes, for instance, the acknowledge that data is a description of people and can harm, or the people's privacy is a virtue [71].
City Safety event data analysis was therefore scrutinised for ethical violations that could potentially concern the investigation. As the data used for analysis was fully anonymous, no ethical violations were detected.

3

Problem: Machine Learning and Data

In this chapter, we give a detailed description of the problem. In the first part, the machine learning principles are outlined, so that the later chapters will be more accessible. In the second part, the data used for the analysis is characterised.

3.1 Machine learning

Machine learning (ML) is an essential part of data science. It is a technique which aims to discover and learn patterns from historical data and by that predict future data. As stated by Arthur Samuel in 1959 [49], ML is *"Field of study that gives computers the ability to learn without being explicitly programmed".*

3.1.1 Process

A typical machine learning process is composed of several steps. The standard and high-level process is shown in Figure 3.1.

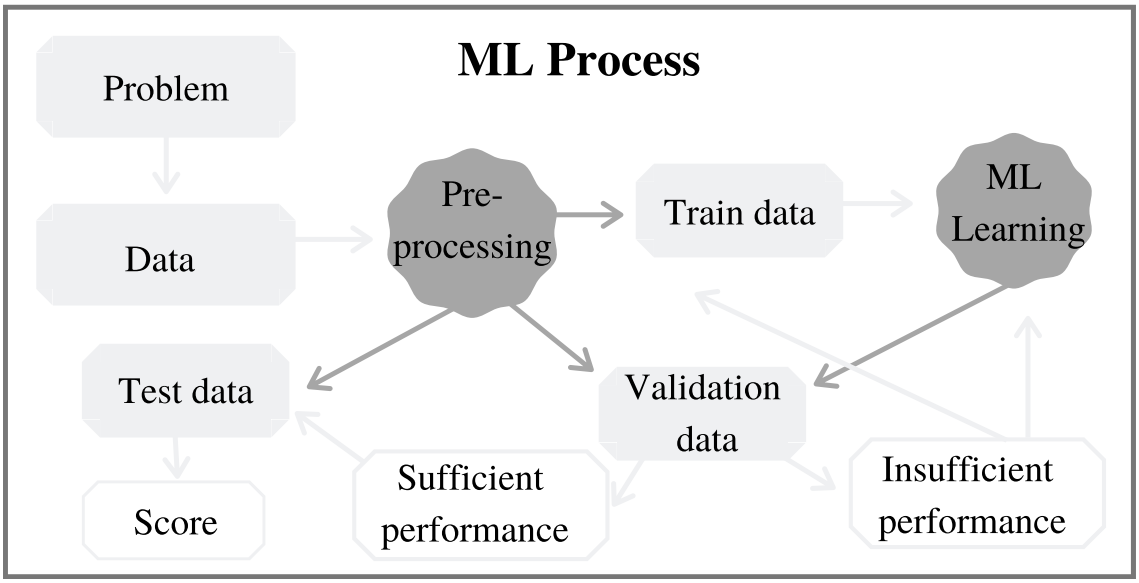


Figure 3.1: The visualisation of the typical machine learning process.

The first step of the ML process is to formulate a problem. In this thesis, the

problem is binary classification, as the algorithm aims to predict only one of two classes (true or false activation of City Safety).
The second step is to identify the data, such as to provide a proper data structure and to pre-process the data. The pre-processing is considered to be a time-consuming and an important phase. It also includes data cleaning, integration, transformation or reduction.
Once appropriate data is ready for analysis, the data is split into train, validation and test set. A training set is an input for an ML algorithm. A validation is used for tuning the parameters and a test set is used for evaluating the classification performance.
When the training (called also *fitting*) is finished, then the algorithm parameters are evaluated on the validation set. If the results are satisfactory, then the learning is accomplished. Next, the modelling is evaluated on the test set. If the results on validation sets are not satisfactory, then supplementary improvements can be applied in order to reach better classification. There are different possibilities of how to achieve it. For instance, the algorithm's parameters can be tuned, the pre-processing techniques can be changed or another ML algorithm can be selected. In the end, the algorithm's performance is calculated on the test set. It is especially important to have the test set for in such iterative process, to avoid data *over-fitting*.

3.1.2 Algorithms and learning

Algorithms form the central part of ML. Depending on whether labels are available during training or not, we speak about *supervised learning* (labels available) or *unsupervised learning* (labels unavailable). A common situation in data science projects is that only some data has labels, whilst the rest (often the majority) does not. In this case, it is called *semi-supervised learning*.
Furthermore, depending on a type of output, if it is continuous or categorical, learning is divided into *regression* and *classification*.
Two ML algorithms types are employed in this thesis. The baseline model is built on supervised learning algorithms, while the pseudo-labelling model on semi-supervised learning algorithms. The details of these two models are described in the Methods chapter. Due to the categorical type of output, both algorithms are classification algorithms, called in machine learning terminology *classifiers*. As the output values are limited to only two categories (called *classes*), we refer to this as binary classification.

3.2 Data

Volvo is gathering road data from new generation cars where the City Safety mechanism are installed. As the main goal in this thesis is to classify the City Safety activations, the data was limited only to the road situations that triggered City Safety Auto Brake. One observation x_n is called an event in this thesis and is a collection of up to 316 signals $i = 1, 2, ..., 316$ represented as a time series of 8 seconds. As shortly mentioned in the Introduction section, the first 4 seconds represent the time before the system was activated, while the next 4 seconds after it was activated.

As measurements are done five times per second, every time series constitutes to 40 data points $t = 1, 2, ..., 40$.
Events have labels y and this results in an output vector $\mathbf{y}_n$, where $\mathbf{y}_n \in \{\varnothing, 0, 1\}$. If, for example, $y_1 = \varnothing$ this means that the event x_1 does not have a label.
One event can be expressed by a matrix:

$$\mathbf{x}_{1,i,t} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1t} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & x_{i3} & \dots & x_{it} \end{bmatrix}$$

3.2.1 Data characteristics

There are several data characteristics associated with this analysis:

1. In total there are 223 labelled events. Initially, there were only 152 labelled events available, hence these events were used for the train and validation sets. After some months since the beginning of work for this thesis, 71 extra events were annotated. These 71 events were used for testing the algorithms' performance.
2. If an event has a label, then there are two classes $\{0, 1\}$ possible in the output vector $\mathbf{y}$, where $\{0\}$ implies an incorrect activation and $\{1\}$ implies a correct activation. Moreover, there are two events for which a class was labelled as "Nuisance". This is because according to the system principles the activation was correct, yet in these particular events the activation was not necessary. These two cases are treated as "True" events ($y = 1$).
3. The classes are slightly imbalanced.
4. The length of a time series is fixed ($t = 1, 2, ..., 40$). The activation of the City Safety system always occurs when $t = 20$.
5. Some signals i are missing in some events. This is because different City Safety upgrades were installed since the beginning of system usage. There are 132 signals that appear in all of the labelled events, and therefore they will be used for the modelling.
6. Given all described characteristics above, one signal is visualised in Figure 3.2.

3.2.2 Signal description

A wide range of signals is measured with the help of car sensors to evaluate if a car should auto-brake to avoid a potential collision. Table 3.1 contains the description of several example signals. A host vehicle refers to a Volvo car (see Figure 2.1), while a target object refers to the object which is detected by sensors and can be a probable threat.

3.2.3 Train-(validation)-test split

Typically in ML projects, the available data is divided into two or three sets: a training set (the algorithm learns on this set) and a testing set (the performance

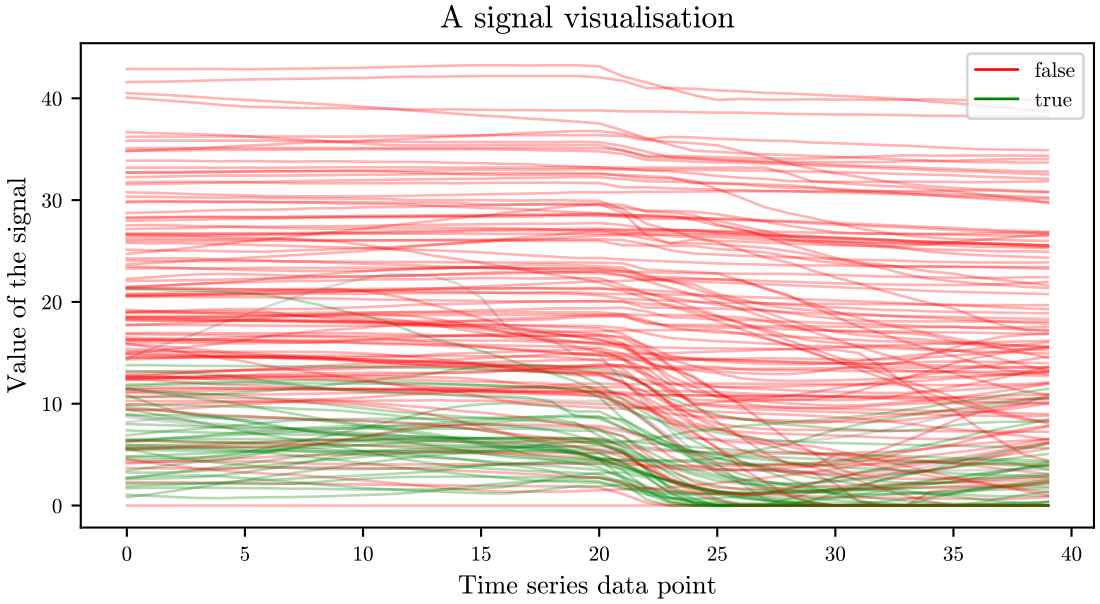


Figure 3.2: Example visualisation of one of the signals over 8 seconds (40 time stamps) period for all 152 annotated events. The green label indicates the correct activation of City Safety, while the red label indicates the incorrect activation. The City Safety system activations occur when time series data point equals 20.

No	Signal name	Unit	Description
1	Lat_acc	m/s^2	Lateral vehicle acceleration
2	Long_acc	m/s^2	Longitudinal vehicle acceleration
3	Lat_pos	m	Lateral position of target
4	Long_pos	m	Longitudinal position of target
5	Yaw_rate	rad/s	The vehicle yaw rate
6	Lat_v	m/s	Lateral velocity of the target
7	Long_v	m/s	Longitudinal velocity of the target
8	Ass_gain	–	Gain to multiply driver intended brake with in safety critical situation.
9	V_tar	m/s	Velocity of a target.

Table 3.1: Example signals with units and description

of the algorithm is evaluated on this set), and a validation set (for the algorithm's parameters tuning). In most articles, if data is split to two sets, usually 70-30 ratio is used, where 70% of data is training data, and 30% is testing data. Setting the golden ratio for training and testing set is a trade-off between the learning generalisation and accuracy of prediction. On one hand, the more data is given for training, the algorithm should learn more or generalise better. On the other hand, the more observations there are in the testing set, the more comprehensively the score measures how well the algorithm predicts new observations. If there is an enormous amount of data available (commonly called *big data*), the decision of what split ratio to choose becomes less important. Generally, algorithms tend to

generalise better if they are fitted on bigger amounts of data.

3.2.3.1 Splitting techniques

Moreover, there exist different approaches of how to split the data into two sets. For example, some of them aim to select observations in such a way to preserve the structure of data, while others aim to reduce the variance of a model. A few of the most common approaches are random splitting, trial-and-error methods or stratified random sampling [45].
Random splitting is a technique that is most commonly used. Observations are assigned to one of the sets randomly, thus the algorithm is quite efficient and easy to implement. In other words, each observation in data has an equal probability to be assigned to one of the sets. Despite its simplicity, this method has a serious disadvantage when applied to the data with the unequal class distribution. Random splitting can lead to an increased variance of the model's error estimate, as training data may not capture the data distribution properly [45].
The trial-and-error method is more sophisticated than random sampling. It aims to reduce the variance of the model's error estimate by performing multiple random sampling. Then, the results are averaged, thus the new set properly represents the characteristics of the data. One of the approaches is to provide a similar mean and variance to in the new set to the one occurring in the data [9]. This method is time-consuming and computationally expensive.
Stratified random sampling is a modified random sampling. It aims at dividing observations based on common attributes in order to preserve the structure of data. In the beginning, the data is explored and divided into clusters of similar characteristics. Later, a random sampling from each created cluster. This approach is widely used in demography, so the structure of the population is kept.

3.2.3.2 Train-validation-test split

As mentioned in 3.2.1, initially, there were only 152 labelled events available. Thus, in the beginning the data was split to two sets and the train-test split ratio was set to 70-30, following [65] and [70]. Here, 70% of labelled events constituted to 106 events.
However, as the number of labelled events increased to 223 along with this thesis' research, a slight modification of train-test data was applied. Therefore, the 30% of data which was initially used for testing was transformed to the validation set. This enabled to tune models parameters. Newly labelled events (71 instances) were used for testing the algorithm's parameters to provide an unbiased method of evaluation. In addition, the random splitting was used due to its simplicity. There were 100 random train-validation splits performed (called *data setups*) to reduce the bias of a training set. As mentioned above, 71 events were used for testing the algorithms.

4
Methods: Algorithms

This chapter describes two ML models: *the baseline model* and *the pseudo-labelling model*. It is organised into four sections: software used, data pre-processing, learning algorithms and model evaluation. Each of these sections has a specific subsection that refers to both models.

4.1 Software

Most of the analysis is performed in Python and its packages, such as *Pandas* [35], which helps to keep the adequate data structures, *Matplotlib* [25] and *Seaborn* [66], which visualise data or *NumPy* [63], which is used for mathematical transformations of the data. In addition, a set of machine learning packages is employed. *scikit-learn* [41] provides machine learning pipelines, learning algorithms and decomposition methods. *tslearn* [60] is applied in the Baseline model for time series classification. Feature engineering is performed with a use of *tsfresh* [14]. Finally, the implementation of *TPOT* [39] enables for an automatic search for the optimal learning algorithm. Addionally, the mRMR scores are calculated in R [43], using *praznik* package [29]. To enable R interface within Python environment, the *rpy2* package [22] is also employed.

4.2 Models

4.2.1 The baseline model

The baseline model is constructed using simple pre-processing and learning algorithms. There are four variants of the baseline model (see Table 4.1).

Model name	**Data Pre-processing**	**Input**
Baseline_11 + pre-processing	yes	11 signals
Baseline_11 + no pre-processing	no	11 raw signals
Baseline_all + pre-processing	yes	132 signals
Baseline_all + no pre-processing	no	132 raw signals

Table 4.1: The baseline model variants

Baseline_11 uses only those 11 signals which are the most relevant according to a pre-study within Volvo Cars together with a domain expert. It is also the least

automatic model, as *a priori* it is based on a limited subset of signals. Therefore, it could be potentially biased.

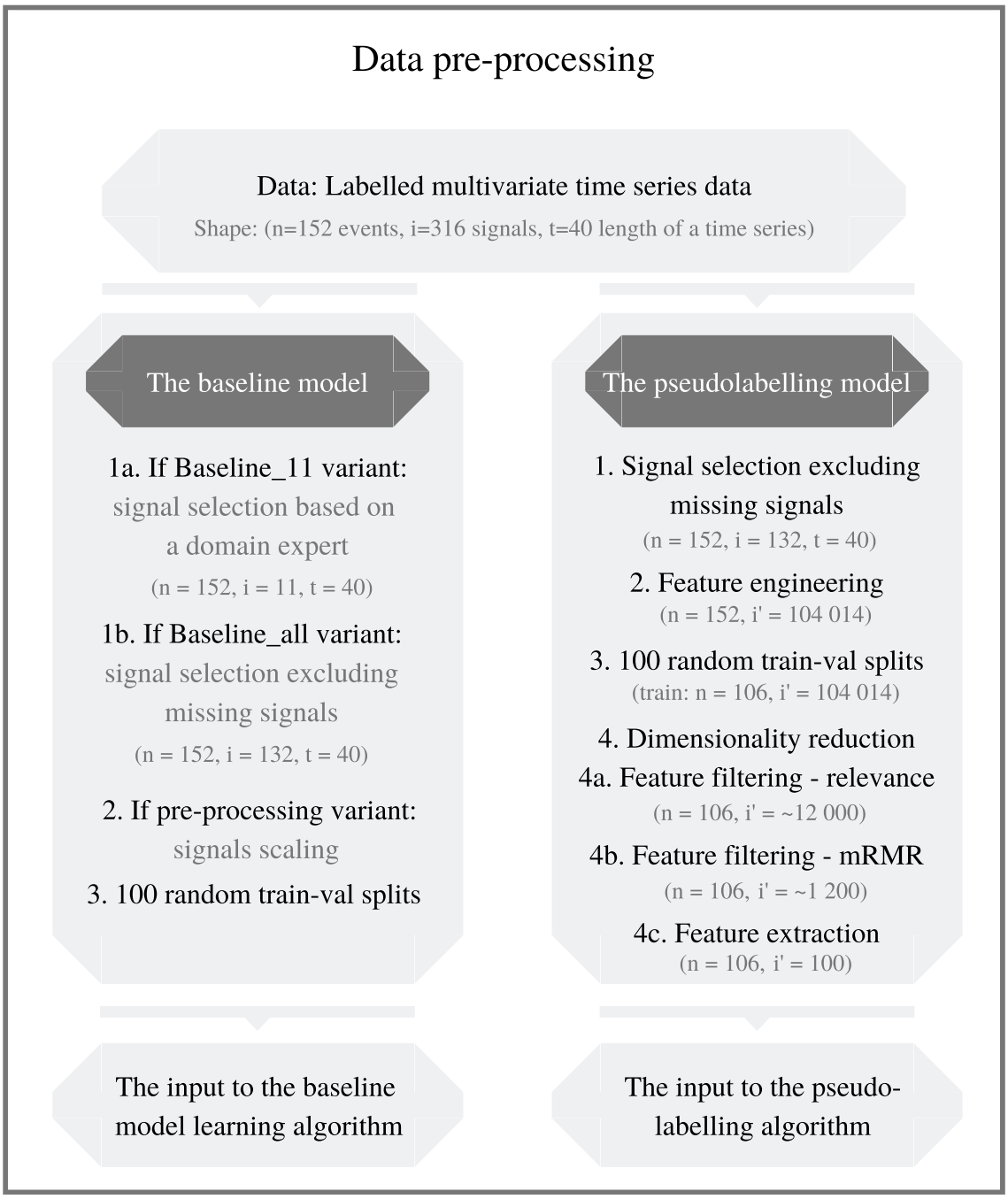


Figure 4.1: Data pre-processing steps.

In contrast to *Baseline_11*, *Baseline_all* uses all 132 signals. Having more input variables can be beneficial, due to an increased amount of information that a model may learn from. However, a higher number of variables can also lead to information redundancy or even increased error rates if a variable is irrelevant. Therefore, these two variants are constructed to test how differently the number of input signals affects the classification score.

In addition, two variants are investigated to explore the effect of input data modifications (i.e. scaling).

4.2.2 The pseudo-labelling model

The pseudo-labelling model is more automatic in comparison to the baseline one. It relies on heavy data pre-processing and more advanced learning algorithms.

4.3 Data pre-processing

Data pre-processing is an important step in data science projects as only high-quality data can properly describe the reality [21]. The data, which is to be applied to a learning algorithm needs to be screened beforehand in order to exclude redundant, noisy or irrelevant information that can hinder the modelling. In particular, careful data pre-processing should be conducted if a sample size is not sufficiently large, because the distribution of available data may not be a representative sample of the unknown population of a particular variable. Inappropriate assumption of distribution may result in higher error rates when making predictions.
There are multiple advantages of using data pre-processing before modelling. According to [21], data pre-processing benefits a researcher as it helps to understand the data, can decrease calculation time or algorithmic complexity of modelling and increase the accuracy of predictive tasks. Depending on the ML task to perform, different data pre-processing methods can be chosen.
In this thesis, we employ distinct pre-processing methods in both models. They are outlined in Figure 4.1 and detailed in the following subsections.

4.3.1 Pre-processing of the baseline model

As mentioned in the Problem section, the baseline model uses the raw signals. The input to this model is a three-dimensional data of a shape of $[152, 11, 40]$ (for Baseline_11) or $[152, 132, 40]$ (for Baseline_all), where: 152 is the number of events, 11 (132) is the number of signals and 40 is the length of a time series. Pre-processing is applied to two variants of the baseline model.
Each event is rescaled by its mean and variance in each dimension, according to the following equation:

$$\forall_{n=1,2,...,152}\forall_{i=1,2,...,11}\forall_{t=1,2,...,40} rescaled\_x = \frac{(x_{n,i,t} - mean_{ni})}{std_{ni}}, \tag{4.1}$$

where:

- $mean_{ni} = \frac{\sum_{t=1,2,...,40} x_{n,t,i}}{40}$, and
- $std_{ni} = \sqrt{\frac{\sum (x_{n,i,t} - mean_{ni})^2}{40}}$.

As a result, the structure of signals is preserved, whilst the magnitude of data points in a time series is equal among signals (see Figure 4.2).

Figure 4.2: The comparison of two raw signals standardisation. Standardisation reduces the influence of a signal magnitude.

4.3.2 Pre-processing of the pseudo-labelling model

The pseudo-labelling model is constructed based on extensive data pre-processing. This approach includes feature engineering, feature filtering and dimensionality reduction. Each subsection gives a short overview of the pre-processing method used, and finally, explains how these techniques were applied to the model. The pre-processing steps are outlined in Figure 4.1. In opposite to Figure 3.1, in this thesis a slightly modified procedure is applied.

Feature engineering means that new features are created based on raw input data, whilst feature extraction means to transform input data into another features space.

4.3.2.1 Missing signals and values

As mentioned in subsection 3.2.1, the data is represented by 316 distinct signals that are collected by the car sensors. Due to different versions of the data recorder, it may happen that some signals are missing in some events. For this reason, the first step of data pre-processing provides the same set of signals for each event to combat the imbalance. This results in shrinking the signal set to the common 132 signals. An example of signal is shown in the subsection 3.2.1 in Figure 3.2.

Moreover, data was assessed with respect to the missing values. No missing values were detected at this stage.

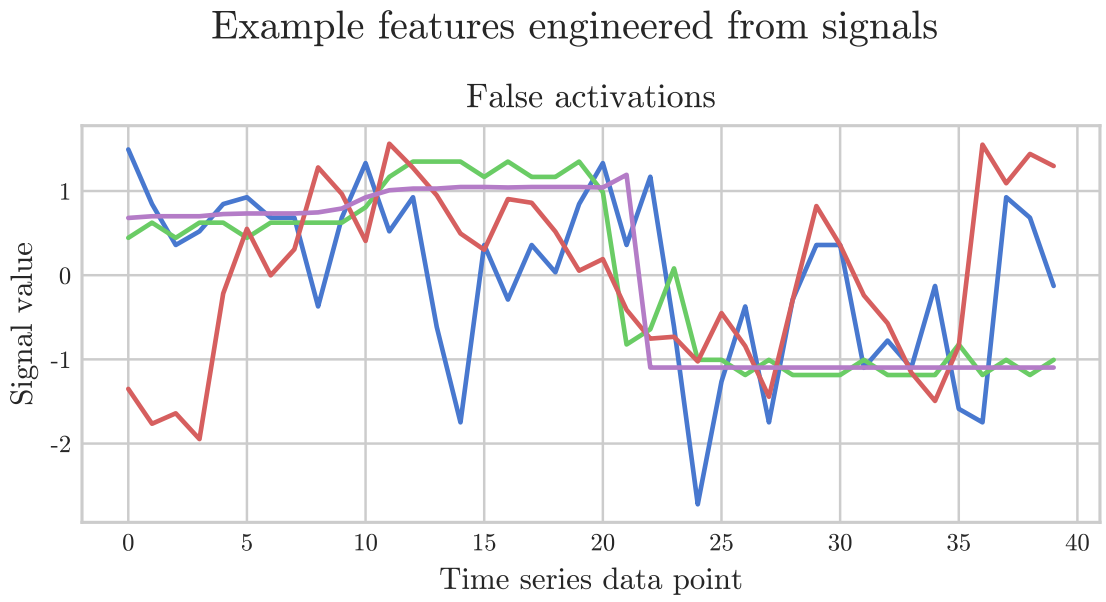
4.3.2.2 Feature engineering

Analysing multivariate time series (i.e. three-dimensional data) is challenging and in some sense limited, as there are not that many models applicable to this kind of data. Additionally, three-dimensional data may require questioning many prior assumptions (e.g. ARIMA model assumes *stationarity*[1] of time series) or a vast amount of computational power that would handle complex algorithms. Last but not least, consecutive values in a time series very often contain redundancy, because they are not independent, thus highly correlated to each other [36].
Feature engineering, a technique that transforms the input data to its representative descriptive attributes [56], solves the issues pointed out above. It enables to reduce noise or correlation or to compress a time series to a smaller format, therefore only the most relevant information is kept. According to [36], algorithms operating on engineered features can also reach better results as well as speed up the computation, if the appropriate features are engineered. Finally, feature engineering can also make a model more interpretable.
As motivated above, various features were calculated for each signal and event, using an automatic feature extraction package *tsfresh* [60]. Three example features are shown in Figure 4.3. The calculated features aim at describing multivariate time series in order to reduce the dimensionality of data.
There are two types of features that are calculated: (1) simple features, which return a single value; and (2) combined features, which return a set of features depending on parameters existing in a feature formula. Moreover, features can be also divided into two groups based on the complexity of formulas: (1) basic features, e.g. minimum, number of peaks or sum of all squared values in a time series; and (2) complex features, e.g. entropy, the descriptive statistics of the absolute Fourier Transform spectrum or auto-correlation. The full list of calculated features is presented in [13]. There are 65 formulas provided in the automatic feature extraction package. However, many of them are combined features that result in more than one return value. Given that there are 132 signals of length 40 that are the input for the feature extraction and for each of them the 65 formulas are applied, feature engineering step produces a vector of 104 014 features for each input event. Nevertheless, many of these 104 014 features contain null values or infinite values, thus these features are automatically reduced from the feature space.

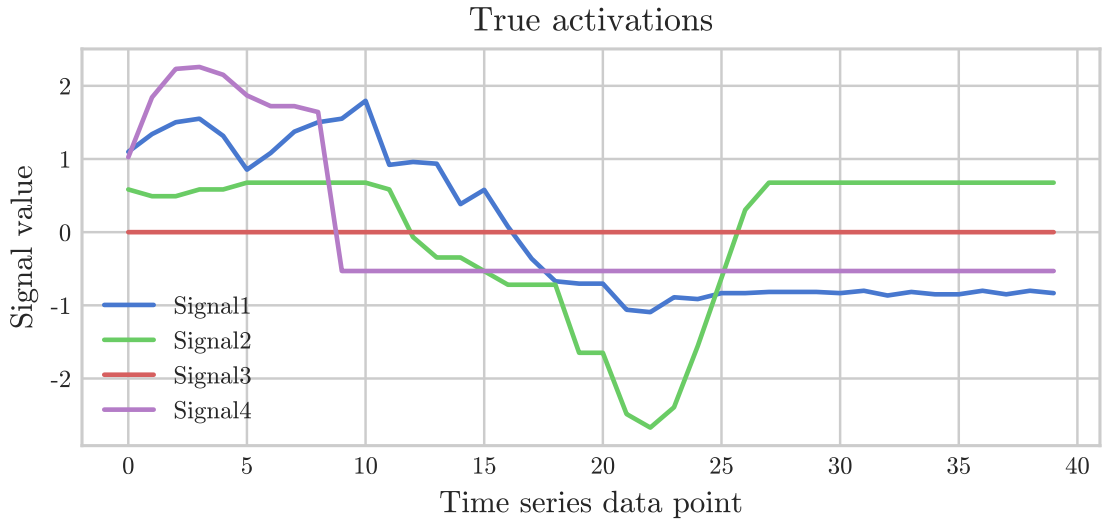
4.3.2.3 Dimensionality reduction

The high-dimensionality problem refers to the situation when the number of attributes p (further referred to as features) exceeds the number of observations n (referred to as events) [20]. As a result of feature engineering, the number of features p equals 104 014, whilst the number of events depends on the train-val set split ratio.
The number of labelled events for training and validating is $n = |X| = 152$. If, for instance, applying a train-val split of 70-30 ratio to all labelled events creates a

[1] In time series analysis terminology, a stationary time series is a time series which is constant over time. It means that the time series properties, such as mean or variance, are constant if the time is shifted.



	Sample entropy	Minimum	Skewness
Signal1	2.06	-2.72	-0.79
Signal2	1.3	-1.19	-0.01
Signal3	2.33	-1.95	-0.23
Signal4	1.06	-1.1	-0.16



	Sample entropy	Minimum	Skewness
Signal1	1.26	-1.09	0.51
Signal2	0.89	-2.67	-1.44
Signal3	inf	0.0	0
Signal4	0.49	-0.53	1.47

Figure 4.3: Example features engineered from signals. From two events (one true event and one false event) four signals were sampled from the signal space. For these signals, three example features were calculated. Signals were scaled to facilitate visualisation.

train set of 106 events ($|X_{train}| = 106$) and, correspondingly, a validation set of 46 events ($|X_{val}| = 46$). Therefore, after the feature engineering step, a severe high-dimensionality problem is faced, as the number of features exceeds the number of events in the training set 981 times.
There are two approaches to solve the high-dimensionality problem. The first approach seeks to find a projection of existing features to a new and smaller feature space. This technique is known as feature extraction and a common example of it is Principal Component Analysis (PCA) or Linear Discriminant Analysis. The second approach aims to reduce the existing feature space by selecting some features based on specific criteria and is called feature selection.
In this thesis, both approaches have been tested, and the subsequent subsections describe them in more detail.

4.3.2.3.1 Curse of dimensionality

In the literature, the problem of having an immense number of features in comparison to a number of observations is known as the *curse of dimensionality*. It is a common problem in contemporary data analysis partly due to advancements in microelectronics that caused rapid development of competitive sensors, which can collect cheaply different type of data. Although the increased amount of data can enhance modelling, it may also happen that pieces of information will be irrelevant or useless [64].
There are two difficulties in analysing high-dimensional data: (1) features are less intuitive to understand, as visualised geometrical properties can contradict themselves [64]; and (2) many methods, especially linear data tools, are designed for low-dimensional datasets.
The classical formula that can exemplify this problem is an estimate of covariance matrix or its inverse [69] in the multivariate statistics. A sample covariance S is calculated using the formula:

$$S = \frac{1}{n} \sum_{i=1}^{n} \left(X^{(i)} - \overline{X} \right) \left(X^{(i)} - \overline{X} \right)', \tag{4.2}$$

where:

- X is a p-dimensional vector of instances $(X_1, X_2, ..., X_p)$,
- n is a number of independent vectors X, and
- $\overline{X}$ is a mean value of X.

To calculate the estimate of covariance, the number of parameters which needs to be calculated equals $(p(p+1)/2)$. For $n = 106$ and p=104 014, estimating a covariance matrix would require the computation of around 5.41×10^9 parameters. That would be an unnecessary computational effort; therefore, a complex feature selection and dimensionality reduction methods should be applied beforehand.

4.3.2.3.2 Feature selection

Feature selection is a technique to select a subset of relevant features based on some conditions. It assumes that a selected subset of features properly represents the whole dataset, thus there is no information loss. There are three groups that

feature selection methods can be categorised into (1) supervised, (2) unsupervised and (3) semi-supervised, depending on whether labels are considered during decision making. Furthermore, supervised feature selection algorithms can be also divided again into (1) filtering, (2) wrapper and (3) embedded algorithms [59].

4.3.2.3.2.1 Filtering, wrapper & embedding

Filtering selects a subset of features based on a certain property, such as a measure of consistency, distance, correlation or dependency. The main advantage of this approach is that filtering algorithms do not rely on a learning algorithm, thus its bias does not affect the result of feature selection algorithms [59]. Additionally, filtering methods are considered to be simple and computationally inexpensive [59]. Nevertheless, most of the filtering methods are univariate, which is one of the most common disadvantages of this approach. Univariate filtering means that each feature is individually compared to a target variable. Therefore, an interaction between some features can be omitted [47].

The most common filtering algorithms are: ReliefF, that evaluates the distance between observations with the same label and an average across other labels; Information Gain, which comes from a family of information theory methods and aims to select the most informative features based on reduction of an entropy of observations of the same class; or F-test, which comes from a family of class variation methods and calculates between and within class variation.

The wrapper algorithms evaluate whether to select a feature within a learning algorithm. In other words, subsets of features are chosen as an input to a learning algorithm. For each chosen subset, the error rates of the learning algorithm are compared. The smallest error rate of the model indicates which subset of features is the most informative. There are different methods of how to divide the feature space into subsets. For instance, all possible combinations can be listed. Although this approach would give the most comprehensive results, it is very seldom used in practice due to high time complexity [31]. Therefore, usually, the modification of this approach is used, such as Recursive Feature Elimination. This method initially trains the learning algorithm on all features and then iteratively eliminates features that are less important. To choose the optimal number of features the error rate is monitored during iterations [67].

The embedded algorithms bridge filtering and wrapper feature selection methods. The feature selection process is incorporated in a learning algorithm, hence it commonly reaches a higher accuracy compared to previously mentioned approaches [59]. Embedded feature selection method aims to find a subset of features that generalises data the most. The popular examples of this approach are decision trees or LASSO with L1 penalty [30].

4.3.2.3.3 Feature selection methods used

In this thesis, the two-step feature selection is employed. Firstly, the number of features is reduced by testing statistical significance and, secondly, by an information-based filtering method.

4.3.2.3.3.1 Statistical Significance Tests

The first step of feature selection is based on statistical testing. For each feature, a p-value vector is estimated with respect to the target variable vector. Then, the significance of p-value is tested, and only significant features are selected.

The p-value of each feature is estimated individually and independently. As the target variable is binary and features are real, the p-values are evaluated with the help of the Kolmogorov–Smirnov (K-S) test. The basic variant of this non-parametric[2] test aims to compare a sample probability distribution to the reference probability distribution (one sample K-S). Since in this case there are two possible values of the target variable, a modified K-S test is applied. In this variant, the data is divided into two subsets depending on the value of the target vector. For each X variable, a sample probability distribution of the first subset is compared to a sample probability of the second subset, hence the separation of classes can be evaluated. The comparison of distribution is evaluated by calculating a distance between empirical distributions of both samples [1]. It is a commonly chosen variant, as the shape and location of two distribution can have a strong influence on a test score. Additionally, it is worth noticing that this test does not specify what kind of empirical distribution the samples have, yet if they are alike.

Once a vector of p-values is estimated by the K-S test, the significance of features is tested using Benjamini-Yekutieli (B-Y) procedure [7]. This procedure estimates if the null hypothesis (a result of K-S test) can be rejected with the respect to the ratio between false rejection to all rejection, called a *global discovery false rate*. The authors of the B-Y procedure state that this approach is much more powerful than traditional procedures, by proving that a false discovery rate (FDR) can be controlled for independent test statistics. The number of features that can be considered as irrelevant, yet not rejected by significance testing, is controlled by a parameter FDR_level and was set to 50% in this thesis.

As the result of statistical tests, the number of features $p = 104\ 014$ is reduced to $p \approx 10\ 000$, depending on a global discovery false rate FDR_level and a train-val split seed. A global discovery false rate is set to 0.5 in this thesis.

4.3.2.3.3.2 mRMR

As mentioned in 4.3.2.3.2, the main disadvantage of most filtering methods is that they ignore the interaction between features. An example of such method is the Mutual Information (MI). The MI measure evaluates how much mutual dependency exists between two variables by calculating an *entropy*. *Entropy* is a term from the Shannon's Information Theory and is a quantified uncertainty about an output class [55]. In this case, each feature is contrasted with the categorised target variable resulting in a vector of MI scores of the length of a number of features. In contrary to the traditional correlation method, MI can discover also some non-linear dependencies [5]. For these reasons, MI is a widely used technique and it poses a base for many more advanced uncertainty measures.

There are multiple modifications of the MI algorithm and one of them, Minimum Redundancy Maximum Relevance (mRMR), was used in this thesis. mRMR was introduced in 2005 by Peng et al. [42]. Similarly to the standard MI algorithm, it

[2] A non-parametric test does not assume the distribution of a variable.

uses a dependency criterion to indicate important features. However, in contrast to the standard MI, it is a multivariate feature selection method. It means that features are investigated twofold: (1) firstly, the standard dependency is evaluated, hence the maximum relevance between a feature and a target variable is calculated; and (2) secondly, the maximum relevance scores are depreciated based on redundancy in between features, thus less correlation in a feature space is encountered [59]. Since maximal dependency can be problematic to implement [59], the approximation of maximal dependency between the joined distribution of variables is performed in the mRMR calculations.
In this thesis, mRMR was computed using the package *praznik* available in *R*. The algorithm works as follows: (1) the MI score is calculated for all features, (2) the features are sorted according to the MI score, (3) thereupon, the features are added iteratively and greedily to the selected features set S based on the maximal value of the formula below:

$$J(X) = I(X;Y) - \frac{1}{|S|} \sum_{W \in S} I(X;W), \tag{4.3}$$

where:

- X is the training set matrix,
- Y is the target vector,
- $J(X)$ is the MRMR function of the input matrix X,
- $I(X;Y)$ is the MI function between an input X and a target vector Y,
- S is the set of selected features, and
- W is the feature evaluated per iteration.

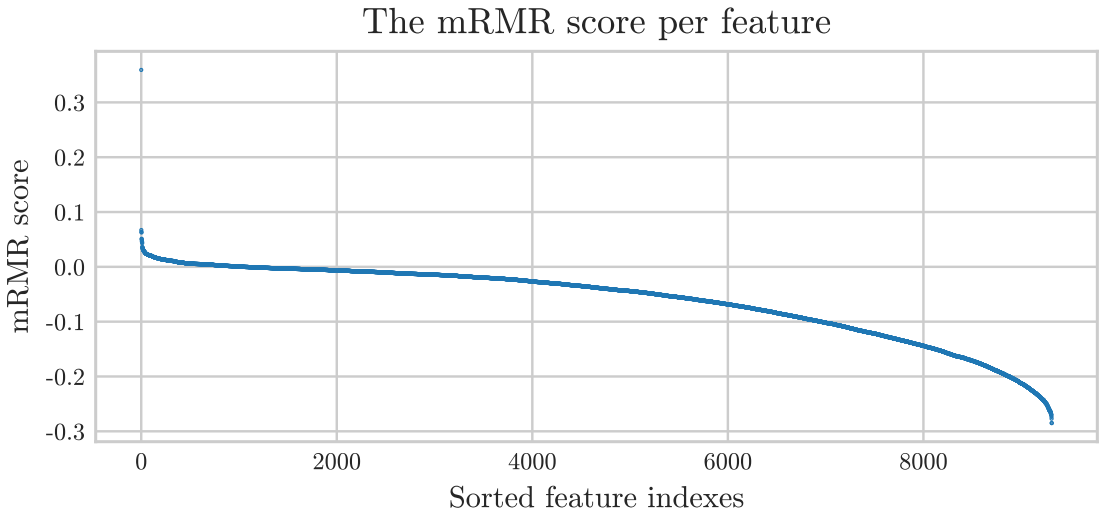


Figure 4.4: The mRMR score per feature. The features were sorted by the score value, thus the number of features to be selected can be easily approximated for a cut-off point.

The mRMR scores are within a range of $[-1, 1]$ and the higher the score, the higher relevance and smaller redundancy of the feature. The mRMR scores per features

are shown in Figure 4.4. The cut-off point was set to 0 which results in shrinking the feature space from $p \approx 9500$ to around $p \approx 2000$, depending on the train-val split seed and the outcome of the previous feature selection.
To evaluate how well the mRMR filtering algorithm separated data by the target class, two features which scored (1) high, and (2) low, were plotted. The results are shown in Figure 4.5.

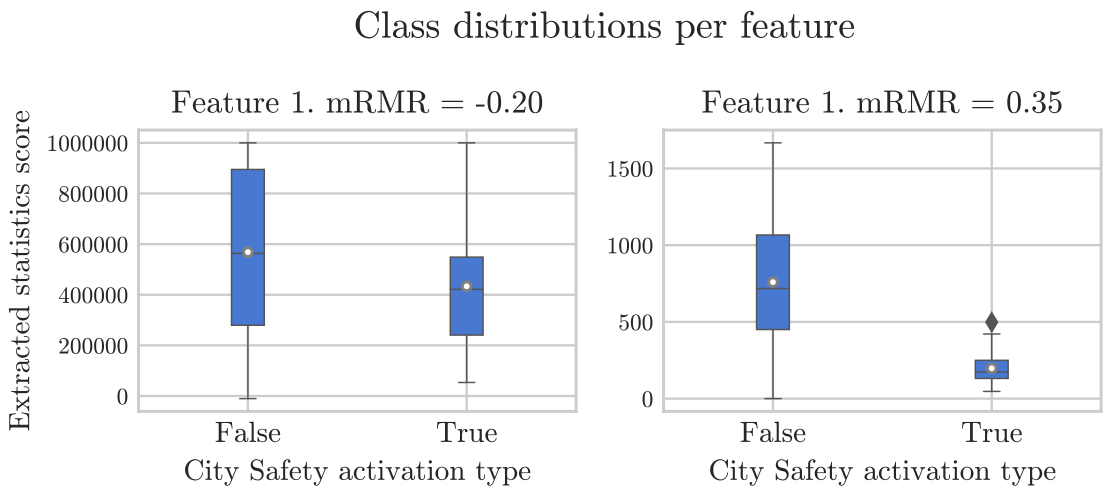


Figure 4.5: Class distribution per feature. The left plot visualises the class distribution of a feature which struck one of the lowest mRMR scores, whilst the right plot - the highest mRMR score. Class distribution is overlapping for the left plot, while it is separable for the right one.

4.3.2.3.4 Feature extraction

After applying the twofold feature selection method, the number of features p still exceeds the number of events n. Stricter filtering could cause a higher information loss, therefore different dimensionality reduction methods are employed thereafter. As introduced at the beginning of this section, the feature extraction methods aim at finding the appropriate projection of existing features. The essence of this approach is to provide new features, called *components*, that could maintain as much variance of input features as possible. As a consequence, the components are uncorrelated with each other [27]. This is the second considerable advantage of applying this algorithm, with many learning algorithms assuming independent features.

4.3.2.3.4.1 PCA

The standard method within this approach is the Principal Component Analysis (PCA). It was formulated by Karl Pearson in 1901 [40], yet some researchers also refer to Harold Hotelling who exploited it further and defined in 1930 [27]. Due to its straightforward mathematical transformation, which enables an instinctive interpretation (most commonly in the form of visualisation), PCA achieved widespread popularity among statisticians. Additionally, it is also successfully adapted to many other variants where the application of the classical PCA form is limited.

Algorithm
In short, the classical PCA algorithm calculates an orthogonal linear transformation of the input matrix X. Depending on the parameters (i.e. some matrix P, that X will be multiplied with) chosen during this step, the coordinate system of an initial matrix X is transformed to a new coordinate system. The new coordinate system can be the same shape as the old (only matrix X rotation) or smaller (dimensionality reduction of matrix X). The classical PCA has the following assumptions: linear correlation between the features, the number of observations exceeding the number of features and the absence of outliers.

PCA modifications
In case some of the assumptions are violated, the modified PCA can be used. The non-linear features dependency can be solved, for instance, by the kernelPCA [51]. The term *kernel* refers to a *kernel function*, which is a definition introduced in the mathematical operator theory. In kernel PCA, *kernels* specify what kind of dependency is investigated during PCA decomposition.
Another variant of the PCA is the SparsePCA (SPCA). This algorithm overcomes the shortcoming of the classical PCA which requires each calculated component to be a linear combination of the p features. Moreover, loadings[3] should also be non-zero [72]. This becomes a problem if a sparse matrix is to be transformed into a reduced size. For this reason, solving this issue was an active area of research, especially crucial in the fields where operations on sparse matrices are frequent [72]. Although the main motivation of this method was to provide a method that will handle the zero loadings, it also solved another issue of the classical PCA. In the standard PCA, if the features number exceeds the number of observations, the method will not be consistent. Thus, another advantage of the SPCA algorithm is that the input matrix can have high-dimensionality and the result will retain consistency [26].
In this thesis, SPCA was applied to the output of the filtering methods and the number of components was set to 100.

4.3.2.4 Data transformation - scaling

Many learning algorithms require that input data should be standardised beforehand.
Standardisation is a technique to transform the raw data into standard and normally distributed data, e.g. mean equals to 0 and variance equals to 1. Whereas the goal of the machine learning tasks is usually not to detect the distribution of data, the shape of the data distribution is ignored. Therefore, raw data is commonly only centred by subtracting the mean value and then scaled by the standard deviation value. Mathematically, it can be expressed by the formula:

$$z = \frac{(x - \mu)}{s}, \tag{4.4}$$

[3] A *loading* is a term used in the PCA terminology that describes a weight of each initial value of a feature to calculate its principal component.

where:

- x is the feature vector,
- μ is the mean of the x,
- s is the standard deviation of the x, and
- z is the scaled vector.

The other and yet popular technique is *normalisation*. Similarly to *standardisation*, it transforms the input vector into a modified form by scaling its values to the range $[0, 1]$. The values in the input vector are subtracted by the minimum value and then scaled by the difference between the maximum and minimum value (see Equation 4.5),

$$z = \frac{x - x_{min}}{x_{max} - x_{min}}. \tag{4.5}$$

Data transformation aims at reducing the variability between features so that the learning algorithms are not misled by the input values' magnitudes. It is especially important if the algorithms depend on distance metrics, such as the Euclidean distance, to decide how to classify instances. Moreover, scaled features are crucial for the PCA transformation. The objective of PCA is to reduce the variance between features and if the features are not scaled, then their variance is also high. Not scaling the features before the PCA algorithm, would skew the components towards features with large magnitude. Finally, smaller features values can also speed up computation as less information needs to be stored in the computer temporary memory [21].
In this thesis, the scaling is performed before statistical test filtering and before SPCA.

4.3.2.5 Class imbalance

If classes are not balanced we refer to this as *class imbalance*. Most of the ML algorithms require or assume a balanced class distribution, therefore applying data to them before checking this constraint would be a serious error. It is especially important, if the distribution of classes in highly unequal, sometimes even extremely, such as 100:1 or 1000:1 [24]. In this thesis, the class distribution of labelled data is 2.4:1, thus it is a moderate class imbalance.
The goal of an ML algorithm is to minimise the objective function, so if this function does not penalise incorrect classification it may lead to the situation that classifier will always predict the majority class for new instances. Moreover, sometimes the cost of incorrect prediction of false and positive class differs. For example, in the health care domain detecting cancer patients may be more urgent than detecting healthy patients.
Generally, there are two approaches to handling this dilemma. The first refers to the data level. It means that unequal classes are artificially re-sampled by under-sampling the majority class or over-sampling the minority class. Although this is a fairly simple approach, it has some disadvantages, for instance, the probability of disregarding some potentially useful data (under-sampling) or to the risk of over-fitting (over-sampling). The second approach refers to the algorithm level. As

shortly mentioned before, the objective function of the ML algorithm can have a penalisation for misclassifying an instance.
In this thesis, the algorithm level of handling the class imbalance was chosen. In a training phase, that will be described in detail in the next section, the algorithm sets weights per each class prediction. Therefore, a class prediction probability is discounted by its weight factor.

4.4 Learning algorithms

Learning algorithms are a fundamental part of machine learning. Choosing a learning algorithm depends on what kind of machine learning task is performed (e.g. supervised or unsupervised). For classification in the supervised learning approach, the purpose of the learning algorithm is to find patterns of the input data, and, in consequence, to classify the instances. The classification algorithms can be grouped into the following categories (see Table 4.2).

Classifier	**Description**	**Example models**
Linear Classifiers	Instances are separated based on the linear combination of the features. A hyperplane that isolates instances is constructed.	Logistic Regression, the Naïve-Bayes Classifier, Support Vector Classifier
Quadratic Classifiers	In contrast to Linear Classifiers, in this approach instances are isolated by a quadratic surface.	Quadratic Discriminant Analysis
Decision Trees	Instances are grouped into nodes based on decision rules.	CART, Random Forest Classifier
Neural networks (NN)	A class of instances is predicted by a complex combination of many ML algorithms and no specific rules. Despite its increased complexity, NN can handle much more advanced inputs, such as images or video recordings.	Perceptron, Convolutional Neural Networks, Recurrent Neural Networks
Nearest Neighbours	A class of a new instance is assessed based on the distance measurement between k nearest neighbours of the instance.	k-Nearest Neighbours

Table 4.2: The categories of the classification algorithms.

4.4.1 The baseline model - k-NN

The learning algorithm of the baseline model is based on *k-Nearest Neighbours*. The basic k-Nearest Neighbours (k-NN) is a straightforward non-parametric algorithm

which is considered as a *lazy learner*[4]. The k-NN pseudo-code is noted in the Algorithm 1.

Algorithm 1 The basic k-nearest neighbours

Require: $(\mathbf{X}_n, \mathbf{y}_n, x, k)$, where $\mathbf{X}_n$ is a matrix of n training instances, $\mathbf{y}_n$ is a label vector of n training instances, x is a new observation to classify and k parameter is a number of neighbours
for $i = 1$ to n **do**
compute the distance between x and $\mathbf{X}_i$
end for
select k nearest instances around x and calculate how many times each class occurred x with a majority class label

In this thesis, the modified k-Nearest Neighbours algorithm is used which takes three-dimensional data (raw time series) as input. Therefore, the distance between each data point at a particular time point in a time series is compared to the other data point in another time series.

4.4.1.1 Parameters

There are three parameters which can be chosen in the k-NN algorithm:

- k: the number of neighbours. Usually, an odd number is chosen, so that there is always one majority class of labels in binary classification. The higher the k, the simpler class boundary and the noise in data more suppressed. Choosing the optimal k is not simple, as if too small k is chosen, then the model can be over-fitted. On the other hand, setting a higher k may lead to model simplifications. In such scenario, the data will be under-fitted.
- distance metric: a function which calculates the distance between two instances. The common metrics are: Euclidean distance, Squared Euclidian distance, City-block[5] distance or Minkowski distance. To calculate the distance between time series the *dynamic time wraping* algorithm (DTW) can be used as well. This method considers also temporal fluctuations that may vary in speed. In other words, if there are two peaks in two time series but shifted in time, the similarity between them should still be relatively high.
- weight function: *uniform* or *distance*. The majority vote of neighbours decides which label to assign to a new observation. If weights are uniform, then each label has as weight equal to 1, whilst if the weight is distance, each label weight is discounted by distance factor of neighbours. This promotes the labels that are nearer to x.

4.4.1.2 Advantages and disadvantages

Notwithstanding the simplicity of k-NN, it is widely used in the community, as it does not call for any underlying assumptions on the data distribution and is quite

[4]In data science, a *lazy learner* algorithm is an algorithm that not have a discriminative function (called often a *model*) which after a learning step predicts a class of a new instance.

[5]Called Manhattan as well.

intuitive. Moreover, it can relatively easily adjust if new data is added and it does not require a training step to estimate a model. It can also be applied to the regression tasks and multi-class classification.
Nonetheless, k-NN also has some constraints and disadvantages. The distance computations become computationally exhaustive if the data dimensionality or the number of observations is large. Choosing the optimal number of k is also highly dependent on the nature of the data. In addition, the algorithm is sensitive to outliers and unscaled feature values.

4.4.2 The pseudo-labelling model

Pseudo-labelling is an intuitive semi-supervised technique known also as *self-labelling* or *self-training* [12]. The idea behind it is fairly simple. In the beginning, if labelled instances are available, the model is trained only on labelled data. Once training is completed, the model predicts the class of the remaining unlabelled instances. The output of this prediction is a probability score of the observation belonging to one of the class. Based on the chosen decision function, some instances are added to the training data (with a *pseudo-labelling* class) and the model is retrained. The procedure is repeated until the user-defined stop criterion, for instance, a fixed number of iterations.

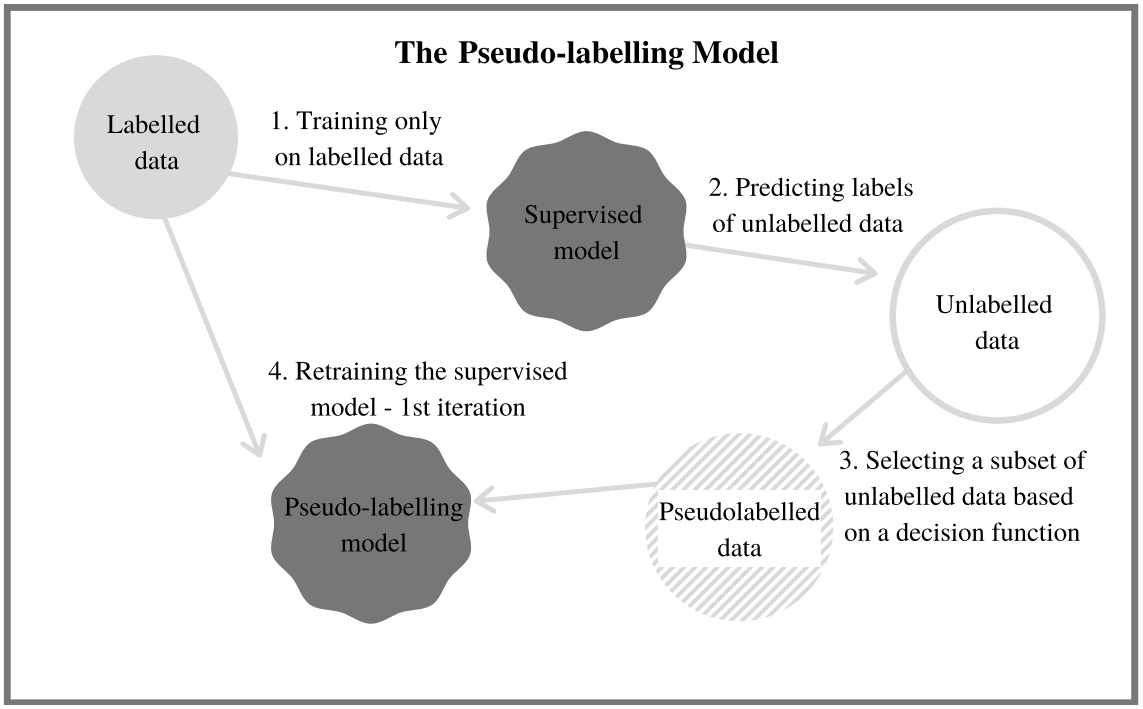


Figure 4.6: The pseudo-labelling model visualisation.

4.4.2.1 Pseudo-labelling

This basic approach is one of the first semi-supervised methods that appeared in the Information Theory literature around 50 years ago in the publications in [3], [19] or

[52]. There, the researchers discussed the Bayesian approach of estimating the unknown parameters of class distributions by repetitively adding new observations to a statistical model (see Figure 4.6). With the increased exploration of machine learning techniques, *pseudo-labelling* algorithm was also applied to deep neural networks. For example, Lee [32] investigated this approach for semi-supervised classification of the MNIST handwritten digit dataset [16].
Although the popularity of pseudo-labelling surfaced mostly due to its simplicity, this approach has several disadvantages. Since the classifier is based on the labelled data, the classification error in consequent iterations of the algorithm can be reinforced. What is more, if the classifier's objective function is to minimise the empirical risk, the unlabelled data will not have any effect on the results. On the other hand, if the objective is to maximise the margin between the separation of two classes, the boundary will be shifted from unlabelled data [12].

4.4.2.2 Model description

In this thesis, a slightly modified pseudo-labelling approach was used. Firstly, due to the small sample size of the training instances, 100 random train-val data splits were performed to minimise the bias in data. Then, for each data setup, the optimal learning algorithm was sought. After this step, several (for example 15) best performing algorithms (evaluated by the score on validation sets) was selected and run again on all setups. Once one learning algorithm (classifier C) was chosen, the prediction on unlabelled data (which was previously pre-processed) was performed (see the pseudo-code Algorithm 2).

4.4.2.2.1 Learning algorithm search

Hundreds of algorithms are analysed with the help of *TPOT* package [39], which automatically scrutinises numerous combinations of learning pipelines[6]. An automatic pre-processing and model selection is implemented in the package. After the convergence, the package exports one learning pipeline. Due to an extensive brute force that finds the best algorithmic architecture, this step is relatively time-consuming.
Exemplary learning algorithms which are evaluated in this step: Gradient Boosting Classifier, Naive Bayes Classifier, Random Forest Classifier, Extra Randomised Trees Classier or k-Nearest Neighbours Classifier.

4.4.2.2.2 Learning algorithm selection

There are two selections performed in the Algorithm 2.
Selection 1: Since 100 train-val setups are created, there are as many *best* learning pipelines which are optimised only for a specific setup. Therefore, several of those are selected and run on all setups. The selection of pipelines is based on the test score value.
Selection 2: At this step, there are several learning pipelines that are applied to all setups. The selection of one *best* pipeline is based again on average val score value, i.e. the mean value and variance are considered.

[6] A *pipeline* is a term used in Machine Learning which a work-flow of ML algorithms.

Algorithm 2 Pseudo-labelling

Require: $(setup, L_{train}, L_{val}, U, T, iter)$, where *setup* is an index of train-val data setup, L_{train} is labelled data for training, L_{val} is labelled data for validating, U is unlabelled data, T is a threshold for selection, *iter* is the number of pseudo-labelling iterations

```
for i = 1 to setup do
    search for the best learning algorithm on L_train
    return the best learning algorithm for setup
end for
choose several best learning algorithm from the previous loop (selection 1)
for i = 1 to setup do
    train on the chosen learning algorithms
    return train score of each learning algorithm
end for
select the learning algorithm C for the main pseudo-labelling part (selection 2)
for i = 1 to setup do
    for i = 1 to iter do
        train classifier C on data L_train
        predict U
        add instances predicted instances to L_train if predicted class probability
            is higher than T
        return evaluate the model on L_test
    end for
end for
```

4.5 Model evaluation methods

In a supervised binary classification task, there is a wide range of methods to evaluate how well the chosen classifier performs. There are only two classes (1 and 0)[7]. Since the true class of an instance is available, each predicted class can be compared to the true class. If the predicted class is the same as the true class, it means that the classifier classified the instance correctly. The sum of all correct predictions over the number of all prediction is an *accuracy* score, represented by the formula below;

$$accuracy = \frac{number\ of\ correct\ predictions}{number\ of\ all\ predictions} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{4.6}$$

There are two forms of correct predictions: true positives TP (when a model predicts 1 and a true class is also 1) and true negatives TN (similarly, the model predicts 0 and a true class is also 0); as well as there are two forms of incorrect predictions: false positives[8] FP (when the model predicts 1 and a true class is 0) and false negatives[9] FN (the model predicts 0 and a true class is 1). It is commonly represented as a *confusion matrix* (Table 4.3).

	PREDICTED = 1	PREDICTED = 0
TRUE = 1	TP	FN
TRUE = 0	FP	TN

Table 4.3: Confusion matrix

Therefore, if data is imbalanced, it is usually more relevant to use other formulas to state the predictability of a classifier, such as *Precision*, *Recall*, *True Positive Rate* (called also *Sensitivity*), *False Positive Rate* or *Specificity*.

$$Precision = \frac{TP}{TP + FP} \tag{4.7}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.8}$$

$$True\ Positive\ Rate = Sensitivity = \frac{TP}{TP + FN} \tag{4.9}$$

$$False\ Positive\ Rate = \frac{FP}{FP + TN} \tag{4.10}$$

[7] In the literature often a class 1 class is noted as *true*, and a class 0 as *false*. Here 1 and 0 are used to not to confuse with a true (real) label of the predicted instance.

[8] Known as Type I error.

[9] Known as Type II error.

$$Specificity = \frac{TN}{TN + FP} \tag{4.11}$$

The above metrics are preferred if the cost of FP (or FN) is higher/lower than of FN (or FP) so that some wrong predictions are more acceptable than the others. Additionally, since in some projects the trade-off between them is needed, another broadly used metric is a *F1 score*. It is a harmonic mean between the precision and recall scores;

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \tag{4.12}$$

Furthermore, since most of the classifiers do not explicitly return one class, but a probability score of belonging to one of two classes, a classifier selects a class depending on a cut-off threshold. For example, initially, a true instance can be predicted as 40% probable to be 0 and 60% probable to be 1. If a cut-off threshold is 50%, then the instance will be predicted as 1. However, the default threshold is changed to e.g. 30%, then the same instance will be predicted as 0. Different values of this threshold are used, for example, if there is a slight class imbalance in the training set. Thus, the random guess of the class equals is not biased.

The diagnostic of how the threshold affects the classification performance is depicted by the *Receiver Operating Characteristic curve* (ROC curve) (see Figure 4.7).

When comparing multiple models, the analysis of many ROC curves becomes hardly possible. Therefore, in practice, the *Area Under Curve* (AUC) score is calculated. It is an integral of ROC which takes values of the range $[0, 1]$ and the larger the value of AUC, the better the algorithm performs.

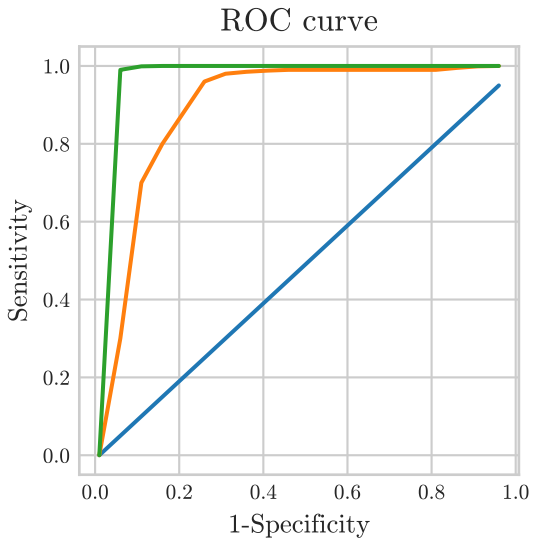


Figure 4.7: ROC Curve. The shape of the curve illustrates Sensitivity vs. False Positive Rate ($100 - Specificity$) for different values of a cut-off threshold. Ideally, if there is a complete class separation, the curve passes through the upper left corner (the green line). If the ROC curve shape is asymmetric, it means that distributions of False Positives and False Negatives represent unequal widths [34] (slightly the orange line). A blue line indicates a random classification.

5

Results: Models Comparison

In this chapter, we examine the results of the two models. The first section introduces the results of the baseline model, whilst the second section describes the pseudo-labelling model's results. Afterwards, we discuss the results in terms of effectiveness, limitations and usage. Furthermore, research questions are reviewed. Finally, we suggest further research directions and discuss the issues raised during the thesis defence.

5.1 Baseline model

As mentioned in the Methods chapter, there were 3 parameters[1] tested in the baseline model's learning algorithm (which was k-NN). Each distance metric, weight function and k in the k-nearest neighbours classifier was investigated in 100 data setups (of the split ratio 70-30) and in four baseline variants. The classification results performed on the validation sets for 100 data setups are shown in Figure 5.1, Figure 5.2 and Figure 5.3. The value of the ROC_AUC score was used to evaluate performance of different parameters setting. In the end, the overall performance of the chosen parameters is calculated based on additional 71 test instances.

5.1.1 Parameters tuning

This section refers to the parameters tuning on 100 validation sets.

5.1.1.1 Parameters *k* and distance metrics

Figure 5.1 depicts the ROC_AUC scores of four variants of the baseline model[2] with respect to the parameter k and the distance metric.

City-block metric vs. other metrics
The City-block metric tends to perform best in the *Baseline_11 + no pre-processing* variant. It reaches around 90% of mean ROC_AUC score for small values of k ($k \in \{3, 5\}$). When increasing k, the performance drops to around 87% of the mean ROC_AUC score (for $k = 21$). Yet, it still surpasses the other tested parameters and model variants. Additionally, the training on raw data using this metric seems to be more stable than using other metrics. The box-plot whiskers are within the range of

[1] The number of k neighbours, the distance metric and the distance weight function.

[2] The models are: *Baseline_11* on unscaled data, *Baseline_11* on scaled data, *Baseline_all* on unscaled data and *Baseline_all* on scaled data.

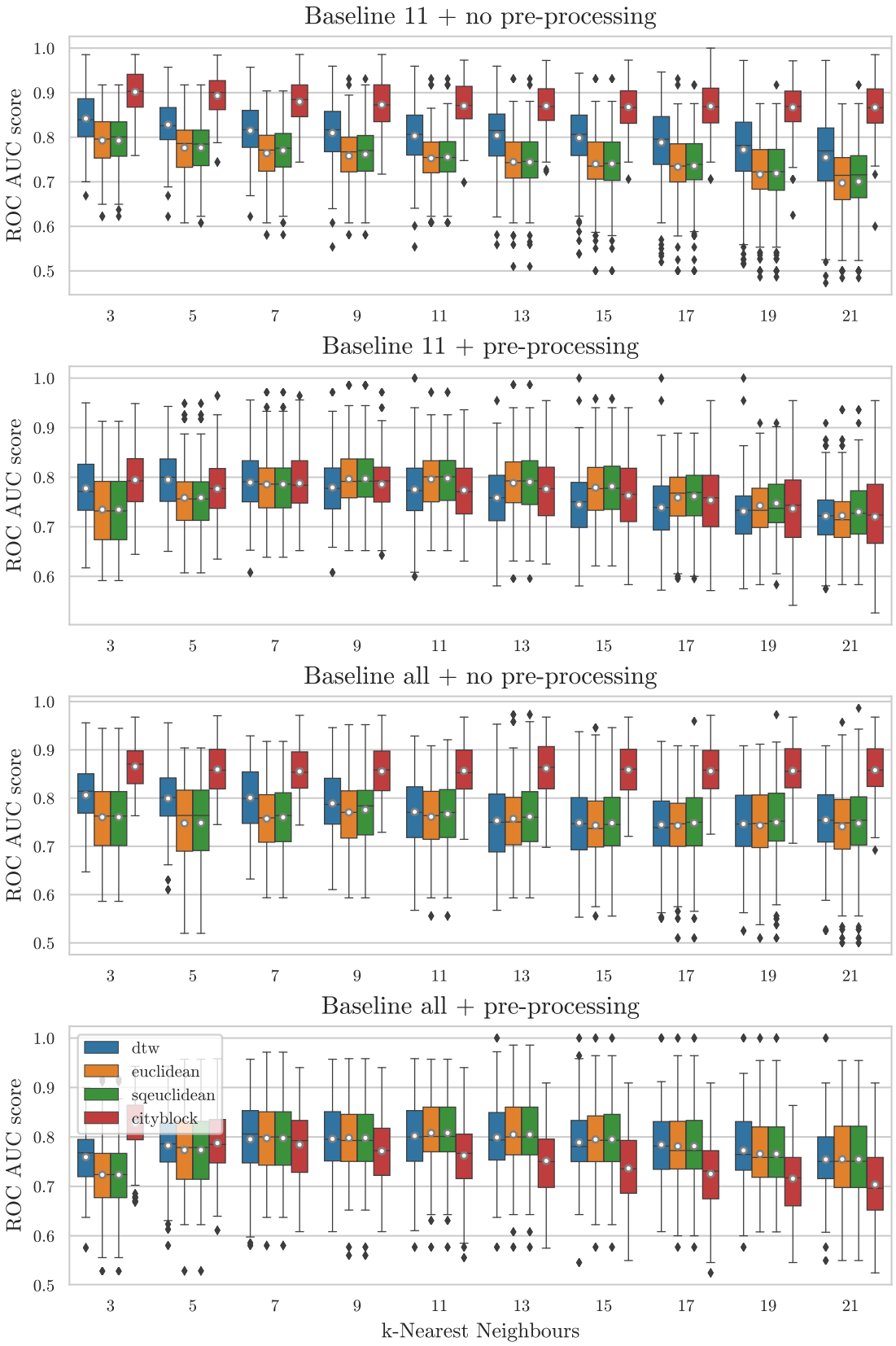


Figure 5.1: Four baseline model variants. The box-plots of the ROC_AUC scores tested against different k numbers and distance metrics.

73-100% of the ROC_AUC score, regardless of the value of k. By comparison, the DTW metric's whiskers are within the range of 52-98%. Interestingly, generally, the City-block metric does not perform better than other metrics (Euclidean, Squared Euclidean, DTW) if the training is done on pre-processed data.

Baseline_all vs. Baseline_11
The ROC_AUC scores of the *Baseline_all* variants (both with and without pre-processing) are on average lower than the corresponding ones in the *Baseline_11* variants. That means that including additional information in the *Baseline_all* did not improve the performance. The highest mean value of ROC_AUC score for the *Baseline_all* variant is around 87% for the City-block metric, while for the *Baseline_11* variant it is around 90%.

Data pre-processing vs. no data pre-processing
All tested metrics seem to achieve similar results on the pre-processed data in both *Baseline_11* and *Baseline_all* variants, ranging in the values of ROC_AUC score from 70% to 80% (depending on the k number). Additionally, the City-block metric appears to be more sensitive than other metrics in the higher dimensions (*Baseline_all* variant) and higher k numbers, as the mean value of ROC_AUC score for $k = 21$ is lower by 5 percentage points than the corresponding value for other metrics.

Euclidean vs. Squared Euclidean metric
The difference between Squared Euclidean and Euclidean distance is that Squared Euclidean metric does not take a root of a distance between two datapoints. Thus, in a basic k-NN algorithm, the results of these two metrics should be the same as if $|x| > |y|$ then $|x|^2 > |y|^2$, assuming the same random data splits for two metrics. However, there are some minor differences between these two metrics visible in Figure 5.1. In this thesis the modified k-NN algorithm is used, which takes a time series data format as the input. This probably causes a slight difference between these two metrics.

Small k values vs. higher k values
Finally, although small values achieve better results (across all metrics), choosing the parameter k to equal 3 or 5, poses a risk of over-fitting the model. The small k values create a complex decision boundary of the classification. If the training data is noisy, the model will not generalise well on the testing data.

5.1.1.2 Distance metrics and weight functions

Figure 5.2 illustrates how different distance metrics and weight functions affect the ROC_AUC score of the four variants of the baseline model. As choosing the optimal value of parameter k is relatively hard, its value was ignored in this section. Therefore, the remaining two parameters can be discussed.

Uniform vs. Distance weight function
Generally, there is no much discrepancy in the ROC_AUC score between uniform and distance weight functions per metric and model variant. Especially, the scores are nearly identical for the *Baseline_all + pre-processing* variant. Nevertheless, the

distance metric seems to slightly outperform the uniform one, for instance for the DTW metric in the *Baseline_11 + no pre-processing*, where the difference between the mean value of uniform and distance weight is around 2 percentage points.
Moreover, there are fewer outliers in models with the distance weight function, as well as the boxplots' whisker range is usually smaller in models with the uniform weight function.

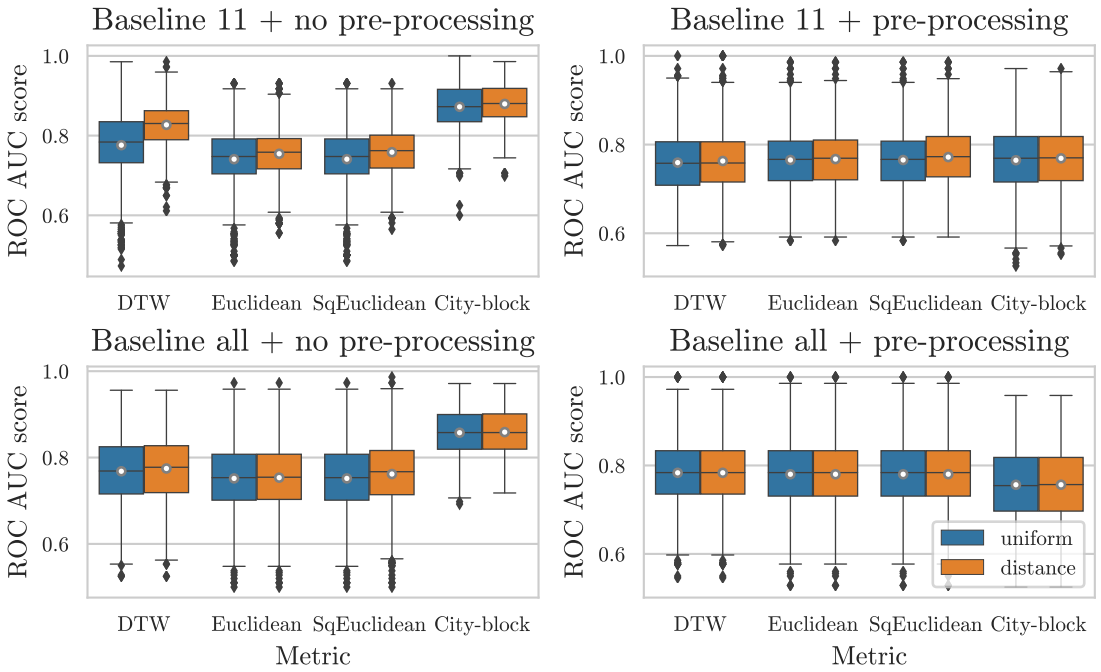


Figure 5.2: Four baseline model variants. The box-plots of the ROC_AUC score tested against different distance metrics and weight functions.

5.1.1.3 Parameters discussion for the Baseline_11 model

The *Baseline_11* variant, the model which input data was reduced to only 11 most influential signals according to the domain expert, averagely performs better than the *Baseline_all*. Hence, further analysis was limited only to this model.

5.1.1.3.1 Weight function

As shown in Figure 5.3, the difference in performance between the uniform and distance weight function gets reinforced with the increased number of k parameter for all tested metrics.
Although both weight functions perform quite similarly, it seems that the distance weight function should be chosen for modelling over the uniform one. This weight function is more stable since the mean value of ROC_AUC score does not drop as much as for the uniform weight function when increasing k. Additionally, the whisker range is usually lower for this weight function than for the uniform weight function.

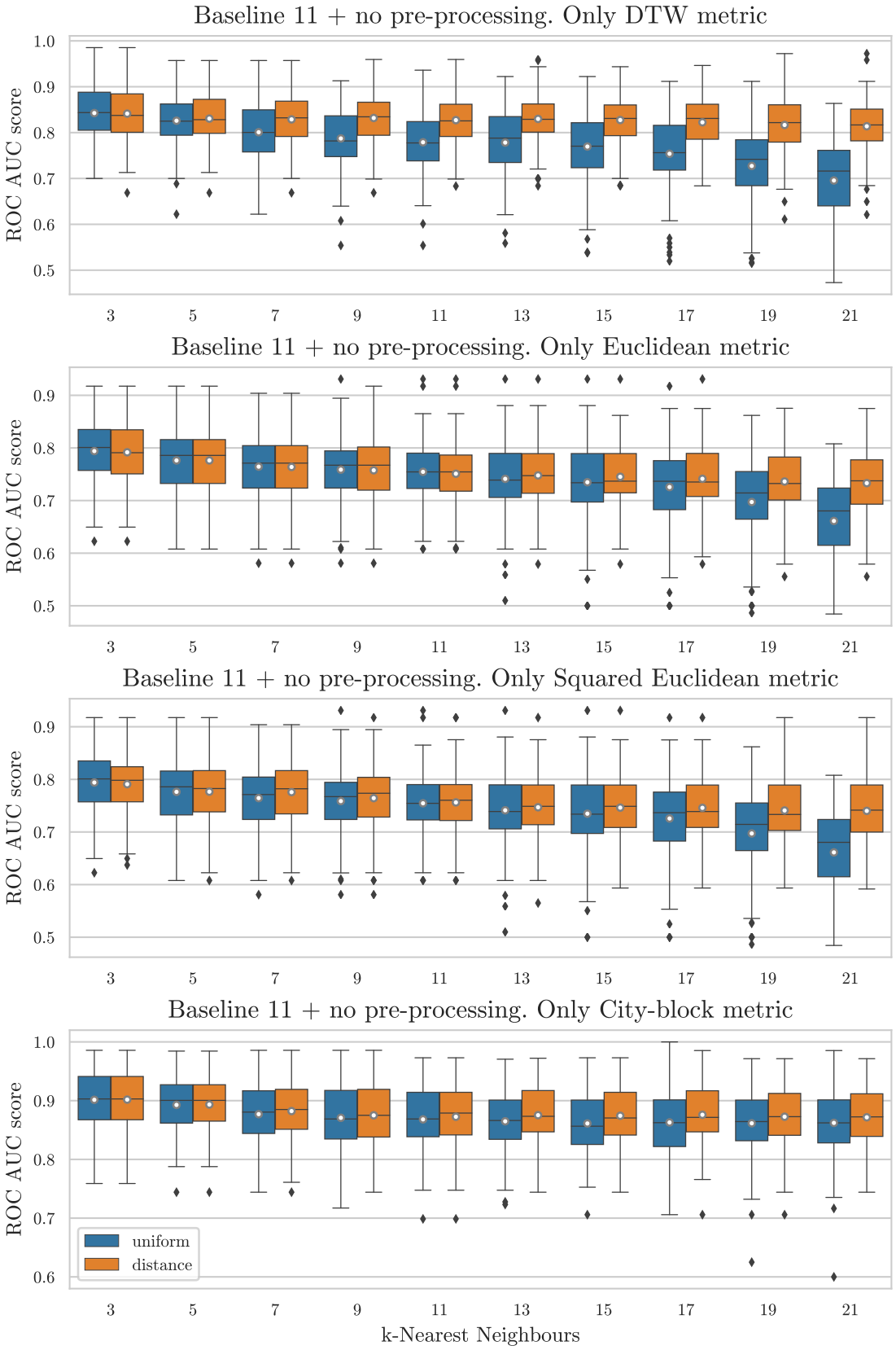


Figure 5.3: The Baseline_11 + no pre-processing variant in four different metrics. The box-plots of the ROC_AUC scores tested against different k numbers and weight functions.

5.1.1.3.2 Distance metric

As partly discussed in the previous subsection, the City-block metric models the data with the highest efficiency. Regardless of the weight function or the k parameter, the mean value of ROC_AUC score oscillates around 88-90%, having also relatively lesser fluctuations in range in comparison to the Euclidean or the Squared Euclidean metric.

5.1.1.3.3 Parameter *k*

There is not much fluctuation of the mean values across the ROC_AUC scores for all tested k parameters for the distance weight function and the City-block metric. These values range from 87% to 90%. Undoubtedly, smaller k values, such as $k = 3$ or $k = 5$, are superior. Nevertheless, $k = 3$ seem to over-fit, on account of the higher whisker range in comparison to the one of $k = 5$. Thus, the optimal number of k, based on visual analysis, is chosen to be 5.

5.1.2 Train-val split ratio

Last but not least, the best variant of the baseline model (*Baseline_11 + no pre-processing*) was examined for the different train-val ratio split (see Figure 5.4). Two train-val splits were inspected: 70-30 and 60-40, where the first number corresponds to the training size fraction. Likewise, the same parameters, i.e. the distance weight function, the City-block metric and the $k = 5$ nearest neighbours, were applied to both splits.

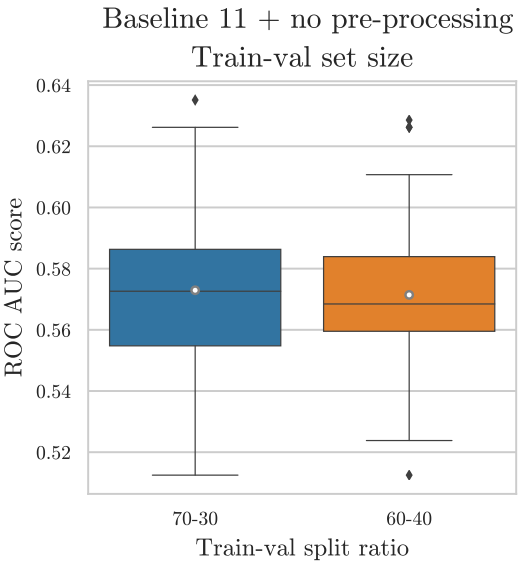


Figure 5.4: The Baseline_11 + no pre-processing variant for different train-val split ratios.

As illustrated in Figure 5.4, the mean value of ROC_AUC score is higher for the 70-30 split. In addition, the mean value in this split deviates slightly more from the median (compared with the 60-40 split), which can be caused by the low value of the outlier. Similarly, the width of boxes for both splits seems to be alike, yet a little shifted down for the 60-40 split. Finally, the whiskers range is wider for the 60-40 split.

In conclusion, having less data to train the model weakens the performance (the mean of ROC_AUC score dropped by 2 percentage points) and makes it slightly less stable (the whiskers range increased by 2.5 percentage points).

5.1.3 Testing

Finally, testing data was predicted on Baseline_11 + no pre-processing model for the chosen parameters. The mean accuracy for different train-val splits was 82% and the whiskers range was within 76%-90%. The mean ROC_AUC score was 57%.

5.2 The pseudo-labelling model

The pseudo-labelling model was based on extensive data pre-processing which firstly transformed the multivariate time series into the engineered features and, subsequently, into the variance-free components.

5.2.1 Parameters

The Gradient Boosting (GB) classifier was applied to the components of the pseudo-labelling model. It is an ensemble method combining several weak learning algorithms to create a single better learning one.
The GB algorithm has multiple parameters to tune. As stated in the Methods chapter, they were examined by the Automated Machine Learning Tool *TPOT* package. The parameters chosen are shown in Table 5.1.

Parameter	**Value**
learning_rate	0.1
max_depth	4
max_features	0.05
min_samples_leaf	14
min_samples_split	16
n_estimators	100
subsample	0.95

Table 5.1: The parameters chosen for the GB algorithm.

5.2.2 Iterations

Firstly, the algorithm was implemented only on the labelled data. For each setup, 30% of data was left to validate the classifier performance (the *base* iteration).
Secondly, the classifier predicted the class of the unlabelled data[3]. Based on the score of the certainty measure, some pseudo-labelled instances were appended to the training set and, then, the model was refitted. Accordingly, the performance of the classification was evaluated on the val set (the 1st iteration).
Finally, the class of the remaining (i.e. not added to the training set in the 1st iteration) unlabelled instances was again predicted using the model from the 1st iteration. Similarly, some instances were added to the training set, the model was

[3]To which the same pre-processing techniques were applied to beforehand.

again refitted and the correctness of predictability was examined on the val set (the 2nd iteration).
The results of the pseudo-labelling model for the validation sets are depicted in Figure 5.5. The ROC_AUC scores are used to compare the model efficiency.

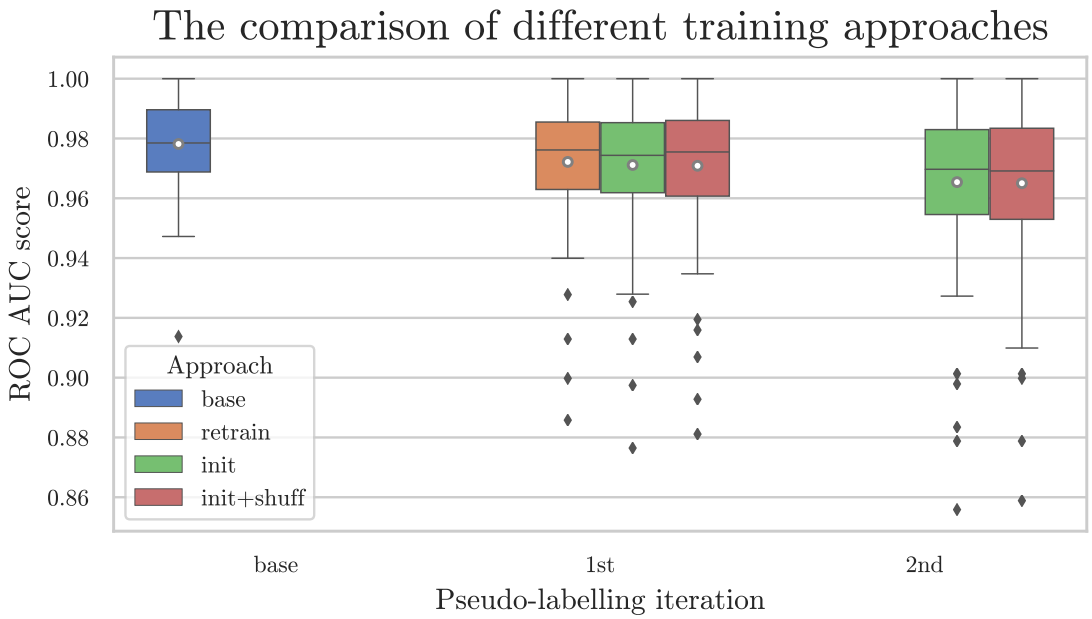


Figure 5.5: A comparison of different training approaches evaluated by ROC_AUC score on the val set. Train-val split ratio: 70-30, threshold value: 99%. The box-plots are grouped by the number of pseudo-labelling iterations. The *base* approach refers to the standard training only on labelled instances. Once the initial (base) iteration is applied, new and pseudo-labelled instances are appended to the training set and the model is retrained. If the same weights as in the base iteration are used, then it is a *retrain* approach. In case the model weights are initialised (set to 0), it is named an *init* approach. The *init+shuff* approach refers to the training where the model weights are initialised and instances are shuffled.

The evaluation of the base iteration
As illustrated in Figure 5.5, the pseudo-labelling model achieves high classification precision on the validation data. The mean value of ROC_AUC scores totals to 98%, while the lower whisker of the box-plot is slightly higher than 94.5%.

The evaluation of the iterations
The classification accuracy of the pseudo-labelling model does not improve after addition of the pseudo-labelled observations, as the mean (or median) value of 1st or 2nd iteration does not exceed the mean (or median) value of the base iteration. However, the drop in the ROC_AUC score between the base and the 1st iteration (for all learning approaches) is smaller than 0.5 percentage point. Correspondingly, the drop between the base and the 2nd iteration is around 1 percentage point. Although there are more outliers in the 1st and 2nd pseudo-labelling iterations than in the base iteration, the modelling seems to be relatively stable.

The evaluation of the training approaches
There were different manners of adding new instances to the model and retraining it. However, the differences in accuracy between these approaches are relatively diminutive.

5.2.3 Thresholds and train-val split ratio

Furthermore, the pseudo-labelling model was tested against two different train-val split ratios (70-30 and 60-40) and threshold values (99% and 95%).
As shown in Figure 5.6, the more rigorous the threshold, the more precise and stable classification of the validation instances. The box-plot width, both for *train70val30-thres99* and *train60val40-thres99* is tighter than for the corresponding ones for the *thres-95*. Moreover, the larger the training sample size (*train70val30*), the higher the mean (or median) value. For instance, the mean value of *train70val30-thres99* is around 1 percentage point higher than the mean value of *train60val40-thres99*.

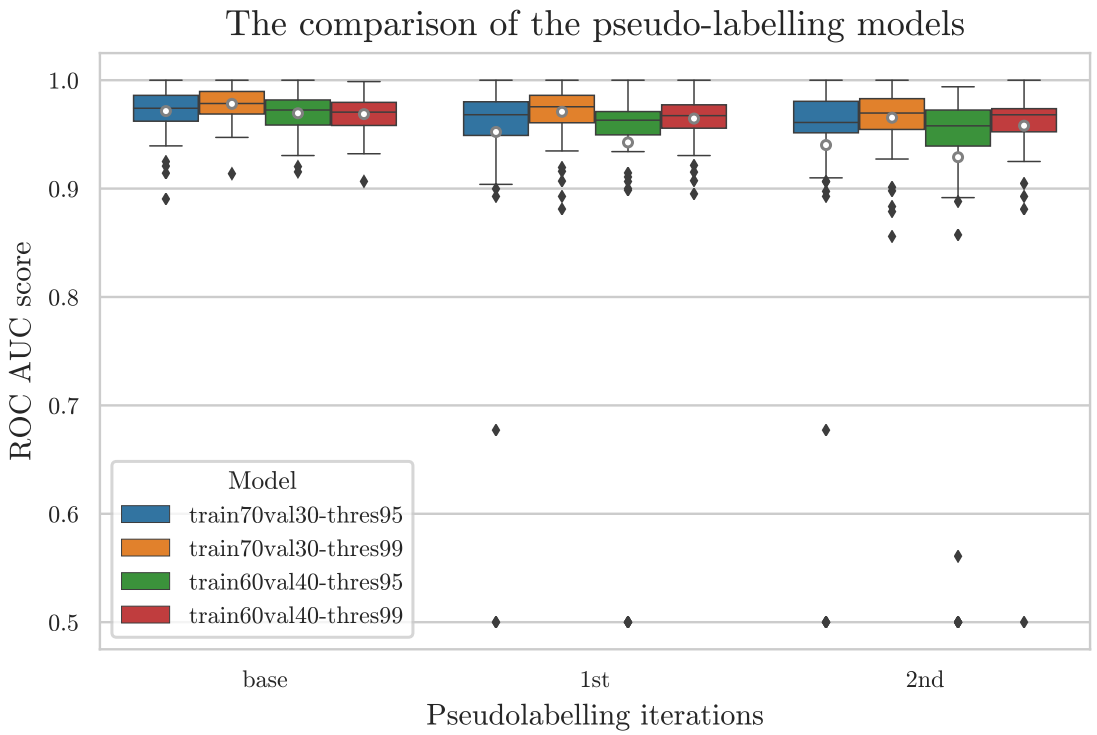


Figure 5.6: The comparison of the pseudo-labelling models in terms of train-val split ratio and threshold of pseudo-labelling. The *init+shuffle* approach was used for the 1st iteration, while *init* approach for the 2nd one.

5.2.4 Important components, features and signals

The secondary goal of this thesis was to determine which signals are most associated with false activations. In order to obtain this information, the components and extracted features had to be post-processed. This post-processing was applied only to the base iteration of the pseudo-labelling model.

During training, the model evaluated how well each component separated classes of the target vector. This information was represented by the importance score. The higher the importance score, the better the separation of the feature and the target vector. Figure 5.7 shows the importance score per component.

Due to the fact that each component can be a combination of tens or hundreds of features, only one component per each modelling setup of the highest importance score was chosen for further analysis.

Subsequently, the chosen component was decomposed to the features that it was based on, using the decomposition matrix from the SPCA step. Depending on the setup, the number of features that one component was constructed from varied between 30 and 240, with the mean value of 90 features. Afterwards, since each feature was a mathematical transformation of an initial signal, each feature was assessed in terms of its base. For example, a feature *signal1_entropy* was post-processed to the form *signal1*, or *signal2_max* to *signal2*. In this step, the mathematical transformations were ignored to enable easier interpretation for the domain expert. The sample of results is stored in Table 5.2

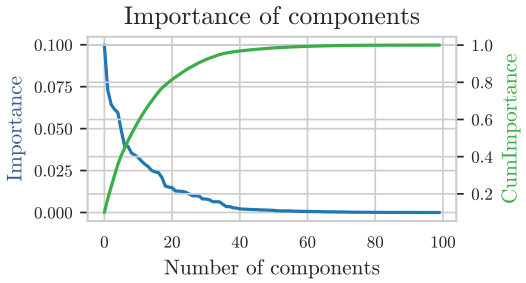


Figure 5.7: Importance of components. The left axis depicts the importance score per component. The right axis shows the cumulative value of the importance scores. Around 60 components influence the classification.

Signal	**Sum**	**Count**	**Setup0**	**Setup1**	...	**Setup99**
signal1	410	48	32	16	...	40
signal2	65	30	7	10	...	13
signal3	0	0	0	0	...	0
...	...	...	...	...	...	...
signal132	529	20	25	35	...	45
		sum	102	59	...	158

Table 5.2: Post-processing.

Finally, the classification performance was not excellent for some of the setups, thus considering all setups in detecting dominant signals would lead to a bias in post-processing mechanism[4]. To avoid this, the setups were filtered based on the ROC_AUC score. Hence, only the best performing setups were included in further analysis, resulting in shrinking the results shown in Table 5.2. As a consequence, the list of signals, sums of features and count of features was communicated to the domain expert.

[4]The post-processing mechanism would have been misled by the features that did not achieve high accuracy, therefore they were not able to successfully separate the classes of the target vector during modelling.

5.2.5 Testing

Finally, the base iteration of the model was examined on test data.
71 testing events were pre-processed and predicted on the trained base model. The classifier returned two values: (1) the probability of belonging to the false class, and (2) the probability of belonging to the true class. A sample of results is shown in Table 5.3. The events were sorted by the difference in probability.

No	Event	Prob0	Prob1	Abs_Diff	Annotation	Corr
1	event1	1.00	0.00	0.9990	FALSE	yes
2	event2	0.00	1.00	0.9989	TRUE	yes
3	event3	1.00	0.00	0.9986	FALSE	yes
4	event4	1.00	0.00	0.9980	FALSE	yes
5	event5	1.00	0.00	0.9970	FALSE	yes
...	...	...	...	...	...	...
39	event39	0.10	0.90	0.8031	TRUE	yes
40	event40	0.10	0.90	0.8004	TRUE	yes
41	event41	0.90	0.10	0.7967	FALSE	no
...	...	...	...	...	...	...
69	event69	0.46	0.54	0.0704	FALSE	no
70	event70	0.52	0.48	0.0430	TRUE	no
71	event71	0.52	0.48	0.0319	TRUE	no

Table 5.3: Results of testing. The predicted class of an event versus the annotated class. *Prob0*: probability of belonging to false class (rounded to 2 decimal places), *Prob1*: probability of belonging to true class (rounded to 2 decimal places), *Abs_Diff*: absolute difference between Prob0 and Prob1 (rounded to 4 decimal places), *Annotation*: an annotated class of an event, *Corr*: if "yes", then prediction was correct, if "no", then prediction was incorrect

Out of 71 additional annotations, 58 of them were correctly classified (corresponding to 82%). However, most of the misclassifications were noted in the lower part of Table 5.3, indicating the events about which the classifier was the most uncertain about. If to calculate the correctness only for the events for which the *Abs_Diff* was higher than 0.70, then 47 out of 50 annotations were correct (accuracy 94%). Moreover, the ROC_AUC score for these test data reached 91%. This extra verification gives promising results.

5.3 Comparison of models

Last but not least, an annotator model and two implemented models were inspected in terms of correctness of prediction, time complexity, advantages, limitations and applicability. The results are shown in Table 5.4.
The biggest advantage of producing the ML model to label the City Safety events is the annotation time reduction as well as the possibility of not involving human

Category	**Annotator**	**Baseline**[a]	**Pseudo-labelling**[b]
Accuracy on test	Approx. 100%	82%	82% (94%)[c]
ROC_AUC on test	-	57%	91%
Time complexity[d]	0.5 - 2 h	0.00192 s	0.013 s
Limitations	human resource involvement	no important feature scoring	extensive data pre-processing
Advantages	deep understanding of the environment, clear reasons why City Safety was activated	fast and simple, sufficient for benchmarking of complex methods	features scoring, high correctness, features interactions taken into account, fast
Applicability	low	low	high

[a]Baseline_11 + no pre-processing with the City-block metric, the distance weight function and the parameter k=5, [b]base interaction of the pseudo-labelling model, [c]94% accuracy more certain predictions, [d]the time needed for one event labelling, excluding data pre-processing

Table 5.4: The models' comparison in terms of correctness of prediction, time complexity, limitations, advantages and applicability.

resources in this process. As there are ten thousands of events stored in the company database, a manual annotation seems unfeasible. Additionally, even having thousands of events labelled, there would still be a need for a process that would determine what affects the false activations of the City Safety system.
Furthermore, the pseudo-labelling model outperforms the baseline model due to careful pre-processing and an application of a strong learning algorithm, which enabled the detection of features that improved the classification. This ability to rank the importance of features is the most advantageous because the further investigation of the false activations can account for potential improvements of the City Safety system in the future.
At last, the pseudo-labelling model is the most applicable, meaning that this model is more dynamic and shows a more data-driven approach in comparison to the other two.

5.4 Review of the research questions

1. Does machine learning, with all its semi-automatic procedures, achieve better performance than a simple model constructed with the help of expert knowledge within this field?
If we define that performance is a measure of correct predictions, then a machine learning model (the pseudo-labelling model) achieves better results than a simple model constructed on the reduced set of signals indicated with the help of expert knowledge (the baseline model). The average precision of the model constructed

with the help of expert knowledge achieves 90% measured by ROC_AUC score, while the average ROC_AUC score of the machine learning model achieves 98%. The second model is slightly slower in labelling one new event (0.013s vs. 0.002s).

2. Does a semi-supervised approach reach better results than a supervised approach? What are the limitations and assumptions of each of them? What is a confidence level of each of approaches?

The semi-supervised approach, which is referred to as *1st* and *2nd iteration* of the pseudo-labelling model does not reach better results than the supervised approach (*base iteration*). However, the results are not significantly worse either, as the decline in the ROC_AUC score is around 0.5 - 1 percentage points. The pseudo-labelling model is limited as it is prone to escalate prediction errors if the underlying distribution of data is assumed wrongly.

3. What are the most significant signals for a true/false activation City Safety system?

Since modelling involved extensive data post-processing, indicating the most significant signals depends on the method of how we transform back the PCA components as well as the extracted features. If we apply the method which was presented in subsection 5.2.4, then the most significant signals are identified by the majority voting of features that they are compound of.

Interestingly, several of the most relevant signals according to the model are exactly the same as the 11 signals used for the *Baseline_11* model (selected by the domain expert). In this setting, we can claim that that classifier performed well since it discovered the important signals by itself. However, among the top 10 high-scoring signals, there were some which were not previously considered. Thus, it is a good start for another research why these particular signals were chosen. This will be further elaborated in section 5.5.

4. To what extent can we believe that labelling of unlabelled events using a semi-supervised approach is adequate? Is the distribution of labelled events a good indicator of inference about the whole population? Which method is best?

Section 5.2.5 describes the test which evaluates how well the pseudo-labelling model predicts the new and unseen data. The prediction of test instances achieved quite a high score, namely, 82% of test events were labelled correctly. Those events that were misclassified were usually different from the observations in the training set, meaning that the classifier did not have enough information about the population in the training. There was only one learning algorithm applied in this model (Gradient Boosting Classifier), however, Naive Bayes was also scoring high in some data setups.

5. What dimensionality reduction method performs the best and to what extent it provides meaningful and interpretable results?

It is difficult to select solely the most critical one dimensionality reduction method since all of them contributed towards the common goal of constructing a high quality model. Since the K-S test, mRMR and SPCA were not the only methods tested, it is worth mentioning that the combination of all three assisted the modelling. The K-S provided a reduced feature set based on univariate filtering, while mRMR on multivariate filtering. Then, SPCA found a linear combination of an input data to its reduced form, thus the collinearity of the features was eliminated. Additionally,

MI and F-ANOVA filtering methods were also tested. However, none of these was of the multivariate approach.
Out of the three-dimensionality reduction methods, SPCA tends to be the most difficult one for a straightforward interpretation. The components computed by this method are created from the original features given in the input and they represent the variability of them.

5.5 Future work directions

The major limitation we faced was the small number of samples. As a consequence, more advanced techniques, such as Vanilla Recurrent Neural Networks or Long Short-Term Memory, could not have been applied. These models were tested initially, but this approach was rejected due to too few training instances that resulted in highly unstable outcomes.
Moreover, bootstrapping methods can be applied in future work to increase the number of samples. Having more observations would allow for less strict filtering.
In addition, as illustrated in Figure 5.7, there are around 40 components that scored approximately zero in the importance score for one example data setup. In future work, these components could be removed from the feature space and the model should be trained again. Afterwards, the ROC_AUC score of the modified model should be compared with the ROC_AUC score of the initial model to evaluate if the application of this procedure improved the classification performance. Moreover, there exists an automatic manner of doing it, namely Recursive Feature Elimination, which was described in the Methods chapter.
Likewise, the number of computed components totalled to 100 to facilitate and speed up computations, since searching for an optimal number of components is relatively time-consuming in SPCA. In future work, it should be examined what would be the optimal number of components.
Furthermore, several more learning algorithms could be inspected. In this thesis, due to the result of the TPOT package that searched for an optimal learning algorithm, Gradient Boosting Classifier was used. However, Naive Bayes Classifier was the second best. In future work, a few more learning algorithms could be tested and compared.
Finally, this thesis aimed to determine what influences true and false activations of the City Safety system. Such an algorithm, which classifies instances rapidly and with relatively high precision, allows the company to apply this technique to thousands of unlabelled events. As a continuation of this research, false activations, being the most critical, should be examined further. For instance, they could be clustered based on some similarity measure. This procedure would enhance the understanding of why the false activations occur (e.g. is it a severe weather condition that distorts the sensor signals).

5.6 Discussion

This section discusses the issues raised during the thesis defence.

Feature engineering approach suggested in this thesis has a significant disadvantage of producing a high dimensionality problem. Nevertheless, this approach was chosen to transform the time series features (called signals) into univariate features, thus a bigger amount of common learning algorithms could be applied to such data. Although such a procedure produced 100 000 features, thousands of these features were irrelevant or redundant. The features were filtered by statistical tests and information-based algorithms, hence after filtering step, the feature space was reduced to around 10 000 features. Given that initially there were 132 signals of length 40, which created around 5 200 features, the feature engineering step increased the feature space only twice.
Additionally, LASSO regularisation was suggested in the discussion as another algorithm for feature selection. Linear modelling is one of the approaches that could be analysed further in the future work. In this thesis, the decision tree based algorithms were used for classification (Gradient Boosting Classifier).
According to [6], the Receiver Operating Characteristic (ROC) Curve can be a biased method to compare models' performance. The advantage of the ROC Curve over a simple accuracy measure is that the ROC Curve evaluates the model's efficiency for all cut-off thresholds, while the simple accuracy for only one threshold. However, to assure the proper use of the ROC Curve, the cut-off threshold should be chosen on train data and then it should be evaluated on test data. It may happen that the best threshold on train and test set differs, as, for instance, data may come from the same distribution but of the fixed size. To overcome the disadvantage of this evaluation method, the authors propose a new evaluation method called The Expected Curve, which aims to provide unbiased estimates for different cut-off thresholds. In future work, this method should be used for models comparison to avoid such potential bias.

6
Conclusion

City Safety is an auto-braking mechanism that intends to detect a possible road collision and, as a consequence, to prevent it. Activation of this system occurs if an object appears in front of a vehicle and the driver does not respond in order to avoid it. On rare occasions, it may happen that according to the driver's perception, the City Safety activation is not needed. Thus, this is a false activation of City Safety.
The purpose of this thesis was to develop a machine learning algorithm that will classify activations and determine what influences this classification. Data used for the analysis was represented as a multivariate time series. A length of a time series was 40, constituting to 8 seconds and the activation of City Safety was at the time equalled to 4 seconds. The variables represented the sensor fusion signals, such as a speed of the host vehicle or a distance to the target object. There were 223 events with a ground truth label. 71 events were used for testing, as they were annotated after the start of the research. The rest (152 events), where therefore used for training and validating. Due to the small sample size, 100 random train-val splits were applied, creating 100 different data setups which were used for modelling.
Two machine learning models were proposed: (1) Baseline, and (2) Pseudo-labelling model. Both models reached quite high performance (82% accuracy on tests sets) for the optimal set of parameters.
The Baseline model was constructed on three-dimensional data, meaning that a multivariate time series were given as an input. The classification was performed using a modified k-Nearest Neighbours. The City-block metric was used a measure to calculate distances between data points, the parameter k equalled to 5 and the class of a test instance was determined as a majority vote of the k nearest neighbours, adjusted by a distance weight.
The pseudo-labelling model relied on highly pre-processed data. The pre-processing transformed a time series signals into the set of variables. As a consequence, two-dimensional data could be applied to the learning algorithm. However, such a procedure resulted in the enormous dimensionality of the data (1000 times more variables than observations). Therefore, three-step dimensionality reduction method was employed. Firstly, the sample was filtered by a univariate statistical test Kolgomorov-Smirnov (K-S). Secondly, the output of the K-S test was filtered again by the Minimum Redundancy Maximum Relevance algorithm (mRMR). In comparison to the K-S test, mRMR not only evaluated the relevance between X variables and a target variable but also reduced collinearity of the X variables. Finally, the Sparse Principal Component Analysis method transformed the variables into independent components. Next, Gradient Boosting Classifier was selected as a learning algorithm. Although the accuracy of the pseudo-labelling was same as for the Baseline

model, but the ROC_AUC score outperformed the Baseline model by 34 percentage point.
In addition, the pseudo-labelling model predicted the class of unlabelled data. Depending on the certainty of a prediction, several instances were added to the training set. Afterwards, the model was retrained and tested. Two iterations as described above was applied to the basic pseudo-labelling model. The accuracy of testing did not improve, yet also did not significantly worsen (drop by 0.5-1 percentage point in ROC_AUC score). This indicates that the instances in the training set were not adequately representing the underlying distribution of all events.
Finally, two models were scrutinised and compared to the procedure existing nowadays in the company. Firstly, the machine learning models were much quicker in annotating one event. Although the pseudo-labelling model was slightly slower (13 milliseconds vs. 2 milliseconds), the time reduction in comparison to the annotator was drastic (0.5 - 2 h need for a manual annotation). By using a proposed classifier, human resource involvement can be shifted from a tedious and manual work towards the analysis of what factors are associated with false activations. Albeit the correctness of prediction is lower for machine learning approaches, it is still sufficiently high to offset the time complexity. When comparing the baseline and the pseudo-labelling model, the pseudo-labelling model tends to outperform the other one due to the higher accuracy of prediction and higher applicability. Since this approach involved the learning algorithm that could rank the importance of the features, it gives more insights for the domain experts than the baseline model which only guesses a class of a test instance based on distances measures.
To conclude, this thesis contributed to a deeper understanding of which factors (signals) determined the true and false activations of the City Safety system. The proposed machine learning approach (the pseudo-labelling model) reached good classification performance, examined by additional testing instances. The pseudo-labelling model can be used as a base for more advanced algorithms, such as Long Short-Term Memory, that operate on sequence modelling, thus they do not require time series extraction. An application of the extensive data pre-preprocessing was needed in this thesis, due to the small amount of data, which constrained the selection of modelling algorithms.

Bibliography

[1] Encyclopedia of mathematics. kolmogorov-smirnov test. `http://www.encyclopediaofmath.org/index.php?title=Kolmogorov-Smirnov_test&oldid=22659`. [Online; accessed 09-may-2019].

[2] Volvo cars corporation. city safety. https://www.volvocars.com/uk/about/our-innovations/city-safety, 2018. [Online; accessed 09-Dec-2018].

[3] A. Agrawala. Learning with a probabilistic teacher. *IEEE Transactions on Information Theory*, 16(4):373–379, 1970.

[4] E. A. M. Association. The automobile industry pocket guide 2018-19.

[5] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks*, 5(4):537–550, 1994.

[6] S. Bengio, J. Mariéthoz, and M. Keller. The expected performance curve. In *International Conference on Machine Learning, ICML, Workshop on ROC Analysis in Machine Learning*, number CONF, 2005.

[7] Y. Benjamini, D. Yekutieli, et al. The control of the false discovery rate in multiple testing under dependency. *The annals of statistics*, 29(4):1165–1188, 2001.

[8] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[9] G. J. Bowden, H. R. Maier, and G. C. Dandy. Optimal division of data for neural network models in water resources applications. *Water Resources Research*, 38(2):2–1, 2002.

[10] N. Brundin and G. Samuelsson. The future of active safety. the next steps towards autonomous driving. *Automotive Viewpoint*, 2014. [Online; accessed 10-Feb-2019].

[11] M. Campbell, M. Egerstedt, J. P. How, and R. M. Murray. Autonomous driving in urban environments: approaches, lessons and challenges. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4649–4672, 2010.

[12] O. Chapelle, B. Scholkopf, and A. Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[13] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr. Extracted features by tsfresh package. `https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html`, 2018. [Online; accessed 04-Apr-2019].

[14] 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*, 307:72–77, 2018.

[15] P. Davidson and A. Spinoulas. Autonomous vehicles: what could this mean for the future of transport. In *Australian Institute of Traffic Planning and Management (AITPM) National Conference, Brisbane, Queensland*, 2015.

[16] L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[17] E. D. Dickmanns. The development of machine vision for road vehicles in the last decade. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 1, pages 268–281. IEEE, 2002.

[18] L. Floridi and M. Taddeo. What is data ethics? *The Royal Society*, 2016.

[19] S. Fralick. Learning to recognize patterns without a teacher. *IEEE Transactions on Information Theory*, 13(1):57–64, 1967.

[20] B. Franke, J.-F. Plante, R. Roscher, E.-s. A. Lee, C. Smyth, A. Hatefi, F. Chen, E. Gil, A. Schwing, A. Selvitella, et al. Statistical inference, learning and models in big data. *International Statistical Review*, 84(3):371–389, 2016.

[21] S. García, J. Luengo, and F. Herrera. *Data preprocessing in data mining*. Springer, 2015.

[22] L. Gautier. rpy2: A simple and efficient access to r from python. *URL http://rpy. sourceforge. net/rpy2. html*, 2008.

[23] M. González, C. Bergmeir, I. Triguero, Y. Rodríguez, and J. M. Benítez. Self-labeling techniques for semi-supervised time series classification: an empirical study. *Knowledge and Information Systems*, 55(2):493–528, 2018.

[24] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008.

[25] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90, 2007.

[26] I. M. Johnstone and A. Y. Lu. On consistency and sparsity for principal components analysis in high dimensions. *Journal of the American Statistical Association*, 104(486):682–693, 2009.

[27] I. Jolliffe. *Principal component analysis*. Springer, 2011.

[28] N. Kaempchen, B. Schiele, and K. Dietmayer. Situation assessment of an autonomous emergency brake for arbitrary vehicle-to-vehicle collision scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 10(4):678–687, 2009.

[29] M. B. Kursa. *praznik: Collection of Information-Based Feature Selection Filters*, 2018. R package version 5.0.0.

[30] T. N. Lal, O. Chapelle, J. Weston, and A. Elisseeff. *Embedded Methods*, pages 137–165. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[31] P. Langley. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall symposium on relevance*, pages 1–5, 1994.

[32] D.-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2, 2013.

[33] C. Lundquist. *Sensor fusion for automotive applications*. PhD thesis, Linköping University Electronic Press, 2011.

[34] C. Marzban. The roc curve and the area under it as performance measures. *Weather and Forecasting*, 19(6):1106–1114, 2004.

[35] W. McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython.* " O'Reilly Media, Inc.", 2012.

[36] F. Mörchen. Time series feature extraction for data mining using dwt and dft, 2003.

[37] N. J. Nilsson. Introduction to machine learning: An early draft of a proposed textbook, 1996.

[38] OECD. Road accidents (indicator). `https://data.oecd.org/transport/road-accidents.html`, 2018. [Online; accessed 09-Dec-2018].

[39] R. S. Olson, R. J. Urbanowicz, P. C. Andrews, N. A. Lavender, L. C. Kidd, and J. H. Moore. *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I*, chapter Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, pages 123–137. Springer International Publishing, 2016.

[40] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[42] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8):1226–1238, 2005.

[43] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[44] A. Rakhshan, H. Pishro-Nik, D. L. Fisher, and M. Nekoui. Tuning collision warning algorithms to individual drivers for design of active safety systems. In *2013 IEEE Globecom Workshops (GC Wkshps)*, pages 1333–1337. IEEE, 2013.

[45] Z. Reitermanova. Data splitting. In *WDS*, volume 10, pages 31–36, 2010.

[46] A. Runhäll. Active safety overview [not published]. Technical report, Volvo Cars Corporation, 2018.

[47] Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 08 2007.

[48] B. Saghafi, M. Hettinger, and N. Patel. Safety critical event detection in dashcam videos. 2019.

[49] A. Samuel. Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort. *IBM J Res Dev*, 3:210, 1959.

[50] J. S. Sangorrin, J. Sparbert, U. Ahlrichs, O. Schwindt, and R. Bosch. Sensor data fusion for active safety systems 2010-01-2332. 2010.

[51] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[52] H. Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

[53] T. Sentinel. Phantom auto'will tour city. *Google News Archive, ed*, 1926.

[54] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[55] C. E. Shannon and W. Weaver. University of illinois press. *Urbana*, 104107, 1949.

[56] M. Sodenkamp, K. Hopf, and T. Staake. Using supervised machine learning to explore energy consumption data in private sector housing. In *Handbook of research on organizational transformations through big data analytics*, pages 320–333. IGI Global, 2015.

[57] S. M. Sulaman, A. Beer, M. Felderer, and M. Höst. Comparison of the fmea and stpa safety analysis methods–a case study. *Software Quality Journal*, 27(1):349–387, 2019.

[58] R. S. Sutton, A. G. Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[59] J. Tang, S. Alelyani, and H. Liu. Feature selection for classification: A review. *Data classification: algorithms and applications*, page 37, 2014.

[60] R. Tavenard. tslearn: A machine learning toolkit dedicated to time-series data, 2017.

[61] C. Thorpe, M. H. Hebert, T. Kanade, and S. A. Shafer. Vision and navigation for the carnegie-mellon navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, 1988.

[62] UN. Transforming our world: the 2030 agenda for sustainable development. `https://sustainabledevelopment.un.org/post2015/transformingourworld`, 2015. [Online; accessed 07-Apr-2019].

[63] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.

[64] M. Verleysen and D. François. The curse of dimensionality in data mining and time series prediction. In *International Work-Conference on Artificial Neural Networks*, pages 758–770. Springer, 2005.

[65] J. Wannenburg and R. Malekian. Physical activity recognition from smartphone accelerometer data for user context awareness sensing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(12):3142–3149, 2017.

[66] M. Waskom et al. mwaskom/seaborn: v0.8.1 (september 2017), 2017.

[67] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping. Use of the zero-norm with linear models and kernel methods. *Journal of machine learning research*, 3(Mar):1439–1461, 2003.

[68] WHO. Global status report on road safety 2018. key facts. `https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries`, 2018. [Online; accessed 09-Dec-2018].

[69] M. Yuan. High dimensional inverse covariance matrix estimation via linear programming. *Journal of Machine Learning Research*, 11(Aug):2261–2286, 2010.

[70] S. Zainudin, D. S. Jasim, and A. Abu Bakar. Comparative analysis of data mining techniques for malaysian rainfall prediction. *International Journal on Advanced Science, Engineering and Information Technology*, 6(6):1148–1153, 2016.

[71] M. Zook, S. Barocas, K. Crawford, E. Keller, S. P. Gangadharan, A. Goodman, R. Hollander, B. A. Koenig, J. Metcalf, A. Narayanan, et al. Ten simple rules for responsible big data research, 2017.

[72] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.