WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

EQUIS ACCREDITED

Master Thesis

Machine Learning

A Practical Assessment of Classification Techniques

Sanja Jovanovic

Subject Area: Data Analytics

Studienkennzahl: J 066 925

Supervisors: ao. Univ. Prof. Dr. Johann Mitloehner,
Dr. Sabrina Kirrane

Date of Submission: 14 April 2018

Department of Information Systems and Operations, Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

DEPARTMENT FÜR INFORMATIONS-VERARBEITUNG UND PROZESS-MANAGEMENT DEPARTMENT OF INFORMATION SYSTEMS AND OPERATIONS

Contents

List of Figures

List of Tables

Abstract

Machine learning is already used by many business domains and for various applications, such as fraud detection, credit assessment, market segmentation or risk management. The purpose of this thesis is to investigate machine learning in general and analyze different classification methods in particular. Furthermore, to gain a better understanding of classification techniques several potentially suitable approaches are applied to a dataset from an insurance company. We perform three learning algorithms with several variations on the dataset. We observed that ensemble learning methods perform better than simple learning methods. Moreover, we discovered that feature selection influences the classification positively. Lastly, we analyzed different approaches to deal with unbalanced datasets and found that there was relative little difference in their predictive power, however, the processing time showed a significant difference. Based on our analysis we provide a summary of the findings so that they can be applied to classification problems in different contexts.

1 Introduction

Technological progress and innovation changes how businesses work and operate: banks, retailers, manufacturers, healthcare institutions and other enterprises across different industries - all have already changed in the way they operate throughout the years [44]. The evolving technology leads to an increase in available data and analyzing it can be beneficial for companies [70]. With that in mind, analyzing data helps companies to understand their customers better and respond to their changing needs. For instance, a company can analyze social media comments about its products, services or promotions and identify aspects that people like or dislike [44]. Marketing can identify people that are more likely to buy a product and address their product suggestions effectively to the corresponding customer segment [10]. Fraud detection can be improved through the combination of various internal and external data sources [44]. Using the potential of today's data has many possible applications, but without analyzing the available data no additional value can be created.

Machine learning is one of the fastest growing areas [33] and refers to a system that discovers patterns in data automatically and improves through experience [10]. It is a discipline of **Artificial Intelligence** (AI) which requires a system to perform activities that in general would require human intelligence [62]. By definition, intelligence requires the ability to learn and improve [5]. Machine learning can achieve that through continuously analyzing data. Once it discovers patterns or correlations in the data it can apply the knowledge on new, unseen data. The goal of machine learning, however, is to act autonomously based on its learnings that are achieved through the analysis of big amounts of data [5]. Machine learning can be used for many problem areas, such as the ones mentioned above. Therefore, machine learning has gained importance as different industries became aware of potential possibilities for improving their businesses. Using the examples from above we can see that machine learning is a relevant research area that has a lot of potential for various industries. The relevancy of machine learning and its ubiquitous applicability is the main motivation for this thesis. However, due to the fact that machine learning is a very broad topic, we want to particularly focus on **classification** which is one area of machine learning that is used to assign data instances to one class or another. Classification can be used for predictive analytics, where data is used to predict whether a certain outcome will occur or not.

1.1 Research Question

The aim of the thesis is to perform research into machine learning in general and look into classification tasks in particular. This is done by applying various techniques to a practical example borrowed from the insurance industry. The methods proposed in the thesis can be applied to most of the classification tasks as they are not domain specific. The research was guided by a publicly available competition, where an insurance company provided anonymized customer data to predict whether a customer will file a claim in the next year. More details on the scope of the competition can be found in Section 1.2.

With this in mind, the following research question arises:

- What is the effectiveness of various feature selection and classification techniques?

Based on this, the following subquestions need to be answered:

- How do you identify valuable features?
- Which algorithms are suitable for a classification problem?
- Which metrics are suitable for the comparison of classification algorithms?

1.2 Use Case

The insurance industry has been subject to many fraud activities and is one of the domains which requires a lot of speculation about its customers, as they generally never know when a claim will be filed. Therefore, insurance companies increase their prices to stabilize their losses from insurance claims [63]. Through machine learning, however, this has changed and has already made a severe impact in the insurance business environment. If insurance companies could identify the customers who are more likely to file a claim they could adapt their prices and minimize their risk of losses. It would also lead to an improved price strategy, where customers who are less likely to claim pay a reduced rate [1].

The practical part of the thesis is based on the case study of the Brazilian Insurance Company Porto Seguro which was published as a machine learning competition on Kaggle[1]. Kaggle is an online platform for various machine

[1]for more details see https://www.kaggle.com/c/porto-seguro-safe-driver-prediction

learning challenges, where companies can host a competition, and individuals and teams can participate in it. In the case of the Porto Segoro competition, the company provided historical data consisting of records that provide information about certain attributes from the past three years. These attributes are called features and represent stored information about the customer, such as age or gender, the car or the region. However, we do not know what these features are in a real-world scenario as they were anonymized. Understanding the features, their interaction and influence between each other is, therefore, considered to be a main challenge of the competition, and by extension this thesis.

1.3 Methodology

This section describes the methodology that we use in order to carry out our research and can be grouped into a theoretical and a practical part. The following explains each of the two parts more in detail.

1.3.1 Theory

The theoretical part of the thesis is necessary to build a common understanding of machine learning and identify some of the possible ways to approach a classification task. The literature review consists of academic papers, reviews, books, but also community based inputs (e.g. blogs).

To evaluate the importance of data for today's enterprises, we investigate the economical benefits that can be derived from it. Therefore, to set the scene, we analyze data science, analytics and their connection to machine learning in Chapter 2. Additionally, to identify the necessary steps in performing a machine learning task we analyze the three main parts of it which is presented in Chapter 3. According to Flach [25], these are features, tasks and models which can be seen as inputs, machine learning types and outputs correspondingly. Moreover, the literature review is essential for identifying possible classification algorithms which are explained in Chapter 4. To understand how these classification algorithms work, we additionally use online tutorials[2], blogs or wikis, as the community provides many valuable inputs.

1.3.2 Practice

The goal of the practical part is to apply the knowledge gained in the theoretical part via a concrete use case. As this part iteratively applies sev-

[2]https://udacity.com/course/intro-to-machine-learning–ud120

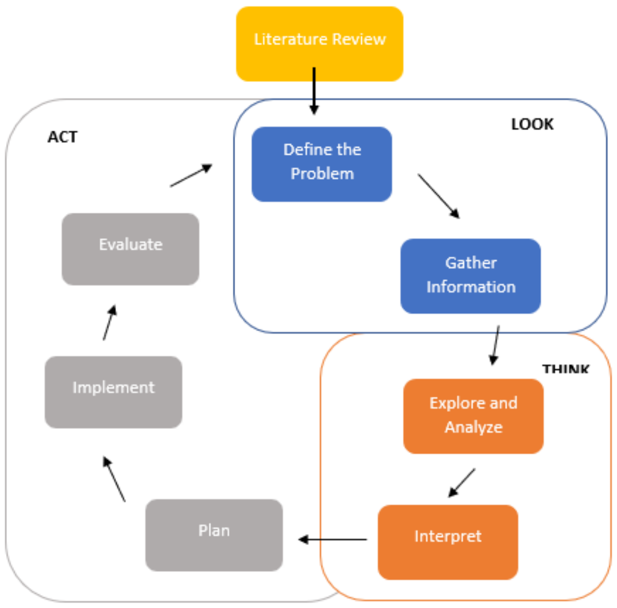


Figure 1: Action Research Cycle

eral approaches and evaluates them, we use Action Research based on the framework presented from Stringer [66] as our main methodology. Chapter 5 summarizes the practical part which is performed iteratively with each cycle repeating the steps depicted in Figure 1. A description of each step can be found below.

Define the Problem. The problem definition is an important initial step as it helps to narrow the scope of the research. The problem was already defined by the Kaggle competition which asked participants to predict which customer is more likely to make an insurance claim.

Gather Information. Since historical data of the insurance company has already been provided, we do not need to gather company relevant data. However, we need to explore different implementation possibilities. With that in mind, we analyze several libraries, such as scikit-learn [56], and evaluate different functions that could be relevant for the case study. Furthermore, we explore several approaches of other participants.

Explore and Analyze. Exploring and analyzing the data is necessary to understand what data we have. In particular, it is important to analyze the types of features, its ranges or statistics.

Interpret. By interpreting the results from the analysis we can recognize patterns and establish hypotheses that can be tested in the next steps.

Plan. Planning is necessary to prioritize tasks and make sure that all relevant tasks are considered. As we use several approaches that we present in the theoretical part, planning also deals with evaluating which approaches are suitable for the given dataset.

Implement. Implementation applies the techniques that were identified as suitable in the plan step.

Evaluate. Evaluation is important to assess the relevance of the techniques to the given problem. If we identify that the approach that was used does not work well with our dataset, we repeat the entire process.

The thesis is structured in two parts. The first part focuses on the theory and the second part on a practical implementation of the knowledge gained from the theoretical part. In Chapter 2 we introduce the relevancy of data analytics and set the scene for machine learning and its components in Chapter 3, which looks at features, models and tasks individually as they can be seen as main pillars of machine learning. Chapter 4 analyzes classification methods in detail and is consequently the last chapter of the theoretical part. The practical part is described in Chapter 5 where we apply the knowledge gained from the previous chapters on a dataset from the car insurance industry.

2 Data Analytics and the Potentials from an Economic Perspective

As previously mentioned in Chapter 1, we know that technological progress changes how companies operate. One reason for this change is the increase in available data. In the last two years more data has been created than in the entire human history and it is estimated that each person will create 1.7 megabytes of new data every second by 2020 [43]. Different sources of data, such as social media content (e.g. picture uploads or postings), voice recordings, video data, and messages are combined resulting in an important asset for companies [22]. It is not new that data is known to be "the new oil", which was initially coined by Clive Humby [2] in 2006, and that analyzing it, can benefit companies, research organizations and the people itself [69].

Davenport [21] argues that analyzing data can basically generate three types of value: Firstly, he says that the results derived from a data analysis can lead to improved processes which simultaneously result in cost savings. Secondly, he states that decisions can be improved through it. For instance, by adding new sources to predictive models, new perspectives can be derived. A model which is going to predict a recommended offer for customers based on their historical data can be improved by adding customers' sentiments or liking behaviors on social media platforms. Lastly, he sees an opportunity in improving products and services. LinkedIn's feature "People You May Know" is a good example illustrating the economic potential of data analytics, as the introduction of this feature created a lot of new customers on LinkedIn and increased the click-through rates by 30%.

Today's companies may have lots of data, however, according to Kitchin [38], this data is not self-explanatory and it needs to be further examined to extract meaning out of it. The overall goal of finding insights and deriving knowledge from it is called **Data science** [6]. Besides the available data, companies need people with certain skills that are able to work with the data. Those people are called data scientists and according to Davenport [21], they are responsible for producing models, identifying patterns or predicting a likelihood of an outcome. For this they require a diverse set of skills, such as technical understanding, curiosity, statistical knowledge and strong communication skills to be able to advice management and also domain knowledge. Latter is important to understand the problems and the identified solutions.

An important step in data analytics is the preparation of the available data

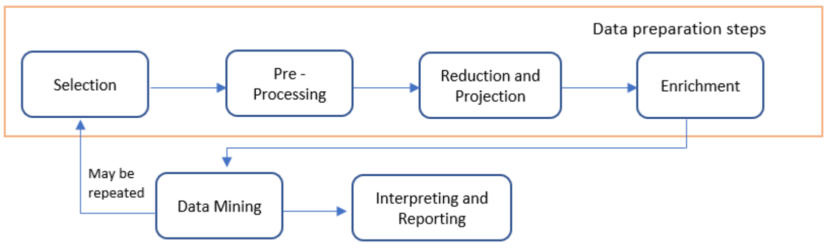


Figure 2: Data Mining Process

in order to derive insights on it. The steps in the process lane from Figure 2 are important pre-processing steps to reduce the complexity of the available data and increase its quality. The following description explains each of the steps seen in Figure 2 which was presented by Miller [46], who also indicated that some of the steps can be skipped or repeated.

Data selection. Not all the data available will be important for an analysis. This step requires to identify which data variables or which data points are necessary. Using all of them could result in inefficiency and redundancy.

Data pre-processing. This step includes cleaning the data. It makes sure that noise, errors or bias are removed. It also includes the handling of missing values and duplicates. Furthermore, this step can also include the data enrichment step as the data can be combined with external data for instance.

Data reduction and projection. Reducing the dimensionality of data through aggregation or normalization for example can lead to better representations of the data.

Data enrichment. In order to increase insights and knowledge the data can be combined with other data, such as combining customer data with their feedback on social media platforms.

Data mining. Data mining techniques are used to find hidden patterns, relations and trends in the data.

Interpreting and Reporting. Techniques to conclude patterns and trends do not help us much, unless we are able to evaluate them and understand what they say. Interpreting the results is essential and allows us to further communicate the gained information.

According to Minelli et al. [47], we can basically put data analytics into three basic categories: *description*, *prediction* and *prescription*. **Description** analyzes what and when something happened. **Prediction** looks at what could happen next or what would change if we used another approach. Lastly, **prescription** focuses on what is expected and how to achieve an optimal results. They further explain that the descriptive analysis is usually done at the beginning, as the value of the data is not known yet and needs to be identified by analyzing historical data. The predictive and prescriptive approach get more important once the data has been analyzed. For instance, we can predict the likelihood of an outcome if we use historical data that we identified as important in a prior descriptive analysis.

Fisher et al. [24] describe a new discipline of data analytics, which they see as a process to extract high-value data from a huge amount of low-value data. Therefore, data analytics consists of different analyses, which strongly depend on the type of available data. For instance, more structured data use statistical and *data mining* techniques to derive insights on it. Hand et al. [29] define **data mining** as the "analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner". Nilsson [54] further specifies data mining as extracting relationships and correlation through machine learning methods and Alpaydin [5] summarizes data mining as "the application of machine learning methods to large databases". All however, say that data mining concentrates on identifying the relationships in the data from various sources.

Machine learning can be applied in order to learn, improve and automate the analysis over time [9]. Machine learning is a branch of artificial intelligence, which refers to a system that is considered smart or intelligent. The definition of intelligence requires the ability to learn and adopt to changes [5]. Machine learning as a whole, can be further broken down into four groups: supervised, unsupervised, semi-supervised and reinforcement learning. The following will roughly describe how a classification algorithm, which is part of supervised learning, works. Classifying data means matching them to certain classes and in the case of binary classification we only have two classes, which are referred to either positive or negative [25].

Henke et al. [30] point out that usually software applications are composed of business logic that is hard coded by software developers. Over time, however, hard coded rules can become outdated due to change in the environment. Through machine learning, however, the program is trained to learn from the data and its diversity. They further mention that the more data a machine algorithm gets, the more it can learn from the inputs. It will analyze different criteria and based on its experience, the method can predict future outcomes and/or gain knowledge from the data. More concretely, training the algorithm means feeding it with data that consists of certain attributes, called **features**. This set of data, which helps the system learn is called a **training set** [9]. During this process, the data scientist selects and creates new features (e.g. combines available ones) that will make an impact on the result [64]. As data scientists understand the business they might have an initial idea which features could contribute to a better result [30].

Supervised learning, in contrast to the other disciplines of machine learning, provides correctly classified instances in the training data. Using the correct classes the algorithm can learn by analyzing the inputs associated with them. Therefore, the more data the algorithm has, the more correct results it can provide [32]. Often, the training set is divided into two parts: a training and *validation set* [5]. The **validation set** is used to evaluate the performance of a **classifier**, which is the machine learning algorithm used for classification. Usually data scientists try several algorithms to see how each one performs. The validation set is therefore created so that a comparison between the different classifiers is possible [5]. After the learning process is completed, the system is presented with a completely new data set, called **test set** [71]. Roughly speaking, based on the training and the conclusions it made, it will end up in a good or bad result [26].

Machine learning can be used for different problems and within different industries. A current McKinsey Global Institute analysis [30] looked at 2,000 work activities across roughly 800 occupations, such as those from a retail worker ("Greet customer", "Answer questions about products and services", "Demonstrate product features", etc.) and evaluated corresponding capabilities (e.g. "language understanding"). They identified that some of the capabilities needed for the job are suited for machine learning. Capabilities that stand out are for instance understanding and generating natural language, recognizing patterns, optimizing and planning, and sensory perception. On the other hand, capabilities which are not suited for machine learning included logical reasoning, creativity and coordination with multiple agents. Machine learning has already some potential use in a daily working environment and this example should highlight this. Furthermore, the study analyzed the potential impact of machine learning on eight problem types on twelve industries. The results can be seen in Figure 3. Interesting is that strategic optimization, predictive analytics, and forecasting seem to have the

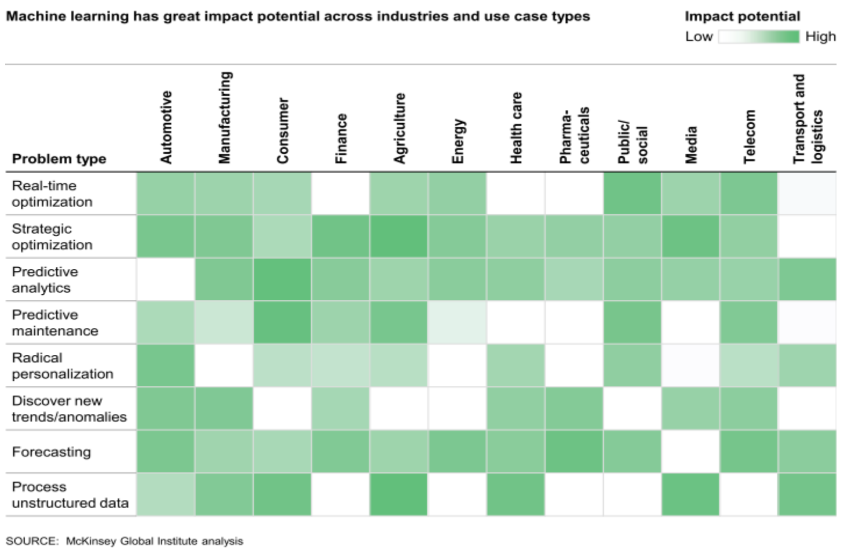


Figure 3: Machine learning potential across industries

highest impact in almost all industries and there is no analyzed industry which does not have any potential for machine learning. What we are missing in this matrix, however, is the potential impact of machine learning in the insurance industry. It is known that insurance companies use the available data about their customers for analysis and risk assessment [30]. Their analysis used inputs such as age, sex or driving records to classify the individual and calculate the premium [45]. However, it can be assumed that this approach includes a lot of bias as for instance all young people were put into one category, without really differentiating whether the person is a good or bad driver [71].

Machine learning, however, has become an important tool for this industry as now more accurate models can be developed through the increasing heterogeneous data. Insurance companies can use external data to better assess the risk of a customer [19], which consequentially allows them to optimize their prices and increase their customer satisfaction [41]. Furthermore, identifying fraud is more efficient through machine learning as more unstructured data, such as text (medical notes, reports, etc.) or social media information (e.g. relation between two parties) can be analyzed [41].

As risk depends on many factors and differs from person to person, it still presents a core challenge for insurance companies to calculate appropriate prices for each individual. Machine learning is already used by the insurance industry to overcome this challenge. The goal of this thesis is to better understand what machine learning is, what its components are and how to apply it on a dataset. Through experimentation an empirical analysis is conducted using a dataset provided by Porto Seguro, a car insurance company.

3 Machine Learning

"Things learn when they change their behavior in a way that makes them perform better in the future" [71]. Learning results in change and if done right, it always results in improved skills, sometimes more, sometimes less. This definition can be applied to humans but also systems, that are capable of changing their behavior by themselves through learning what is right and wrong. Flach [25] introduced a concept that he calls "the ingredients of machine learning" for explaining to his readers the three main parts of machine learning: Features, models and tasks. As they are principal elements of machine learning, this chapter is structured accordingly.

3.1 Features

"Features[3] determine much of the success of a machine learning application, because a model is only as good as its features" [25]. This statement says already a lot about the importance of features, or rather, the importance of selecting the right features amongst many. Flach [25] further indicates that at the beginning of each machine learning task it can be valuable to first of all analyze the features from a statistical perspective and to subsequently identify which of the features will be useful or can be discarded. For instance, if another feature incorporates the same information or if features can be combined into a new one. In a dataset different types of features may exist and each one requires a different analytical approach. Basically they can be divided into *quantitative*, *ordinal*, *categorical* and *boolean* values [25].

Hand et al. [29] explain the terms and describe that **quantitative features**, also called continuous features, refer to values that incorporate a numerical scale, such as age or income. If an order of values exists, then they are called **ordinal features**, for example rankings. **Categorical features**, also called

[3]Features are attributes of a data instance. A person might have the features gender, age or weight.

nominal features, are values that are neither numerical nor in a certain order, such as regions or cities. **Booleans**, in contrast, only work with two values, true and false, or 1 and 0. A boolean feature would for example be whether a customer filed a claim (=1) or not (=0).

3.1.1 Statistical Summary of Features

Statistics can be used in order to analyze the available data. However, depending on the type of feature some statistical calculations make sense, whereas others do not. Statistics can be grouped into *statistics of central tendency*, *statistics of dispersion* and *shape statistics* [25]. **Statistics of central tendency** are used to analyze and compare the data, **statistics of dispersion** look at the spread of the values, and **shape statistics** look at how the visualization of the values is formed, for example a normal distribution [60].

The *arithmetic mean*, *median* and *mode* are probably the most used statistics of central tendency. The **mean** is the average value of a set of observations. The mean cannot be calculated for each feature type. Moreover, it does not make sense for all quantitative features regardless of their domain [25]. To clarify, calculating the mean on the drivers age will make sense as we can identify the average age of a group of drivers. However, averaging the horse power of a car will most probably not give any insight. In order to make use of the **median**, it is required that the values can be ordered, as the median is the middle value of the ordered list [25]. The median could therefore also be calculated on a driver's age, as the different age values could be ordered from smallest to highest. Reimann et al. [60] explain that the median is resistant to extreme values, also called outliers, measurements that are way above or below the other ones. They further describe that the median stays the same even if 50% of the data falls into extreme values. In contrast, outliers are influencing the mean as each value is included in the calculation and weighted equally. Therefore, if dealing with outliers, the median is seen to be a better choice for evaluating the central value. Lastly, the **mode** is the value that exists the most often in a set of observations [25] and is coherently, the value that has the highest probability to occur in that set [60]. With that in mind, it can provide valuable input if enough data is available, especially in dealing with missing values, as one approach might be to use the mode for the replacement of missing values of categorical features within certain groups [7]. To demonstrate, the engine size of a car might be an important feature to identify whether a customer will file a claim in the next year. If for some observations the engine size is missing, one might calculate the mode of the engine size in the range of a certain gas consumption (miles per gallon).

According to Flach [25], *Variance* and *standard deviation* are often used in the statistics of dispersion. The **variance** is the squared deviation of the mean and the **standard deviation** is its square root. Both statistics measure the spread to the mean, but the standard deviation uses the same measuring unit as the feature. Furthermore, statistics such as *range* or *quantiles* can be used to analyze the features. The **range** is the difference between the maximum and minimum value and **quantiles** can be broken down to percentiles, deciles and quartiles. Percentile refers to the value where the p-th percent of the observations are below it. The 10th percentile is also the first decile, the 25th percentile is more often called the first quartile and similarly, the 50th percentile, second quartile and the fifth decile are the same as the median [64]. Percentiles in the upper and lower range (e.g. first or 95th percentile) are often used to identify outliers [60].

Reimann et al. [60] indicate that in a very balanced dataset the distribution of the data will result in a normal distribution, where, theoretically, the mean, mode and median are the same. As the data gets more diverse and includes higher spreads, the shape of the distribution changes. *Skewness* and *kurtosis* are two possibilities to measure the shape of data. **Skewness** measures the symmetry of the dataset. This means that if a dataset has a normal distribution, its skewness is zero. However, if the distribution of the data shows that one side includes more data than the other, it includes skewness. If a dataset includes more values on the left side than the right, it is called right-skewed and the skewness takes on positive values. In contrast, if the values are more on right side it is a left-skewed distribution, where the skewness takes on negative values. **Kurtosis** is another measure of shape and indicates how flat or peaked the distribution is. A normal distribution has a kurtosis of zero, it takes positive values for a higher peak than the normal distribution and negative values for a flatter peak. Depending on the data size, skewness and kurtosis can be standardized with a constant. Moreover, both standardized values of the shape measures should be within the range of $\pm$ 2, otherwise they are considered extreme.

Table 1 was adopted from Flach [25] and summarizes the possibilities of statistical calculations depending on the kind of feature. In general, quantitative features can be calculated on all of the above mentioned statistics, whereas others, such as categorical features, do not really allow a broad statistical analysis. In fact, all non numeric features (categorical and boolean) values, cannot be used to calculate the standard deviation or the variance.

Feature Kind	Tendency	`Used`	Spread	`Used`	Shape	`Used`
Categorical	mode	$\surd$	`n.a.`		`n.a.`	
Boolean	mode	$\surd$	`n.a.`		`n.a.`	
Ordinal	mode, median	$\surd$	quantiles	$\surd$	`n.a.`	
Quantitative	mode, median, mean	$\surd$	quantiles, range, variance, standard deviation	$\surd$	skewness, kurtosis	$\times$

Table 1: Features and statistics used in the Case Study

Nor can they use shape statistics to analyze the size of the peak. Therefore, those values are indicated with `n.a.` in Table 1. Furthermore, the `Used` column indicates whether it was used in the case study that can be found in Chapter 5.

3.1.2 Feature Transformation

Some machine learning methods cannot deal with certain features, for example categorical, which is why these features need to be changed in order to work with them. The process of improving features through changing, removing or adding information is called **feature transformation** and Flach [25] provides an excellent description of feature transformations which are summarized below.

The different types of feature transformations depend on the type of detail a feature has. Quantitative features include the most detailed information of all four feature types. Ordinal features include more information than categorical ones and boolean features contain the least information. **Binarization** is the simplest feature transformation, as it transforms categorical values into a corresponding set of boolean features. This means, that if a categorical feature includes three different categories, these three will each result in their own corresponding boolean feature. This is a necessary step if a model cannot deal with more than two values at once. If we assume that we have a feature "engine size" which specifies to which group a car belongs. This feature could have the values 1100, 2000 or 3000 cubic centimeter (cc). Then the set of booleans could look like the following:

$$x_{\mathrm{i1}} \begin{cases} 1, & \text{if } i\text{th car is an 1100 cc engine} \\ 0, & \text{otherwise} \end{cases}$$

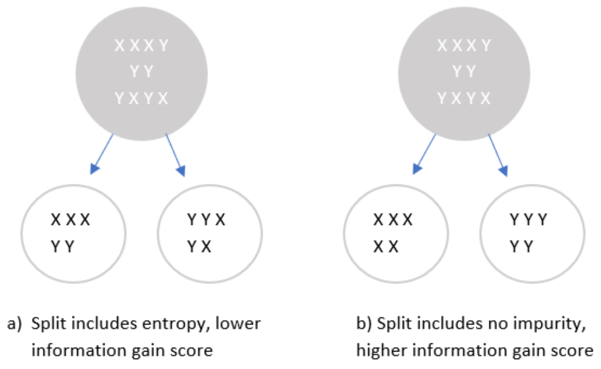

XXXY
YY
YXYX
XXX
YY
YYX
YX
a) Split includes entropy, lower information gain score
XXXY
YY
YXYX
XXX
XX
YYY
YY
b) Split includes no impurity, higher information gain score

Figure 4: Information gain

The second could be

$$x_{i2} \begin{cases} 1, & \text{if } i\text{th car is an 2000 cc engine} \\ 0, & \text{otherwise} \end{cases}$$

and so forth, until the set of booleans is created containing one for each value of the feature.

The process of transforming an ordinal feature into a categorical one is called **Unordering**, as the ordering is eliminated. Unordering is often necessary as many learning methods cannot deal with ordinal features.

Thresholding is another feature transformation and refers to the change of quantitative or ordinal features into a boolean feature. Decision trees, a supervised learning method which is discussed more in detail later on in this chapter, could use thresholding as a splitting criteria. It can be further reduced to unsupervised and supervised thresholding, where unsupervised thresholding usually uses statistics of tendency such as the mean to evaluate splits. In contrast, supervised thresholding can often do a more precise job, as it sorts the data values and tries to increase metrics such as the information gain. The information gain is a frequently used measurement in information theory and often finds its use in decision trees [57]. The amount by which the entropy decreases is the information gain. The entropy is a measure that evaluates how impure a given set is. If a split divides a set into two pure leafs then its entropy is 0 [28]. Figure 4 uses a trivial visual example to illustrate entropy and information gain.

Flach [25] further indicates that **Discretization** is a more commonly used way of feature transformation. The available data points are generalized with the help of so called "bins". In other words, each bin is a grouping of data and represents an interval of the original values. A country's gross salary could be an example where discretization makes sense. Instead of many different values, a model might work better by grouping them into bins (e.g. <25,000 €, 25,000 - 45,000 €, >45,000 €). Discretization can again be broken down into a supervised and unsupervised approach. Unsupervised discretization requires the amount of bins upfront, whereas the supervised approach uses other methods, such as a top-down or bottom-up approach, to identify an optimal number of splits progressively. A top-down approach splits the values continuously into bins, whereas bottom-down approaches start by putting values into its own bin and merging them one after another. However, both methods require a stopping criteria which decides whether a further split is done.

Normalization and *calibration* are two feature transformation methods which focus on the different scales of quantitative or ordinal variables. **Normalization** refers to a unsupervised approach and transforms quantitative features which use different measuring units on a 0-1 scale. This kind of feature transformation enables it to work with different units. An often used example to illustrate normalization is the comparison between persons using their height and their weight. Both values can be used to identify a certain outcome, for instance the gender of a person. However, as height values are usually bigger if the measure is in centimeters (or smaller if the measure is in meter), it gets more difficult to compare weight to height. Normalizing the features results in values between 0 and 1 and can consequently be compared to each other. On the other hand, **feature calibration** is a supervised method of feature transformation that adds a scale to categorical or ordinal features.

The above mentioned feature transformations require the availability of data. However, for various reasons it might be that some of the data points are missing. The process of finding a acceptable value for the missing one is called **imputation**. The quickest and easiest way to fill out a missing value is to use statistics, such as the mean, median or mode for a certain group [7]. Another possibility for analyzing in a more precise way is to analyze the correlations between features and build a predictive model to identify the missing values [25]. Table 2 summarizes the possible feature transformations from this chapter and indicates if they are needed in the case study which can be seen in the `Used` column. The table also includes the reason why the method was necessary or not.

Transformation	`Used`	Reason
Binarization	$\surd$	categorical features need to be transformed into booleans
Unordering	$\times$	learning methods that are used can work with ordinal values, therefore unordering is not needed
Thresholding	$\surd$	automatically done by decision trees when quantitative features are used
Discretization	$\times$	features with a high amount of values were discarded from the dataset due to irrelevance
Normalization	$\surd$	dataset includes features with different scales
Calibration	$\times$	the dataset does not include ordinal values with high variety of scales
Imputation	$\surd$	missing values are present in dataset

Table 2: Feature Transformation in the Case Study

3.1.3 Feature Creation and Selection

At the beginning of this chapter it was mentioned that features play a major role in machine learning. Besides feature transformation other feature-relevant tasks are important. The quality of the dataset sets the base for the success of the machine learning task [65]. Usually, the dataset consists of many features, whereby not all of them are useful to a model [35]. If a model uses features that are irrelevant, it might harm the model more than it contributes to a better result [35]. Furthermore, some of the available features might be redundant [50], highly correlate with each other (e.g. price and tax paid), or only have an impact on the model if they are combined with others, appearing as one single feature [65]. These problems are classified as *feature irrelevance* and *feature interaction* and can be addressed by *feature selection* and *feature creation* respectively [65, 42].

Feature selection

Motoda and Liu [50] explain feature selection as the process of choosing a subset of features in order to optimally reduce the original feature space according to a criterion. The information gain, correlation coefficient and the accuracy are often used to measure the utility of a feature [65, 35]. Furthermore, feature selection is seen as the process to eliminate irrelevant and redundant features from the original data [35]. For datasets which include thousands of features such as text processing or gene expression problems,

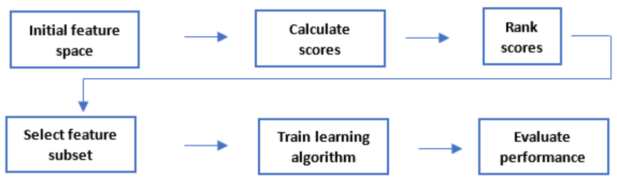


Figure 5: Filter approach

feature selection is especially of high importance. However, it is also a necessary step in smaller datasets, since fewer features allow the learning method to operate faster and more effectively [28]. With that in mind, reducing the original feature space can in some cases improve accuracy [28, 35] and in others, allow a better comprehensibility of the results [65, 28]. Depending on the way of how the features are chosen, feature selection methods can be categorized into *filters*, *wrappers*, and *embedded* methods. For a better understanding of the three feature selection methods we depicted each one visually by using a blog post [36] as inspiration. They are shown from Figure 5 through 7.

In the case of **filters**, features are already selected before the training of the learning algorithm starts and are seen as a data preprocessing step [28]. Filters have the advantage that they are faster compared to wrappers as irrelevant or redundant features are eliminated before the training of the algorithm has started [65]. Additionally, filter methods are generic in their nature; hence they can be used by all learning methods [65]. Figure 5 visually shows how a filter approach works; the corresponding description of each step can be found below.

Initial feature space. The initial features that are available, when starting a machine learning task.

Calculate scores. The calculation on the importance of features is necessary in order to rank the features. There exist various possibilities for that such as dependency, information, or distance measures [20]. Information gain is one of the calculations measuring the relevancy of a feature [65]. Further calculations on feature importance include the Pearson correlation which belongs to the group of dependency measures and calculates the correlation of the features to the response variable, and the chi squared, which can be associated with the group of distance measures as it looks at how close the expected values are from

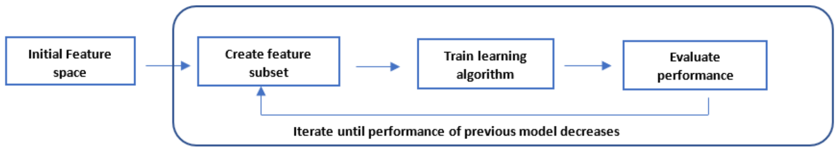


Figure 6: Wrapper approach

the actual ones [53].

Rank scores. Once the importance of each feature has been calculated, it is ranked according to their score.

Select feature subset. At this point, the new feature subset is selected. The ranking in the previous step is used to identify which of the features should be kept. However, the threshold for the selection of the features can be manually specified.

Train learning algorithm. Once the new set of features has been selected, training the algorithm is conducted with the new feature space.

Evaluate performance. As a last step, the results of the learning algorithm need to be evaluated.

Wrappers apply the learning method which subsequently creates a new feature subset [65]. Usually, this approach separates the training set into a training and validation set; the training set is used to train the predictor, the performance is then checked on the validation set [65]. Each of the subsets provides the accuracy as it can compare the actual results to the predicted ones and this accuracy is used as a score for the corresponding feature set [65]. The process of training and evaluating is done repeatedly and in the end, the feature set with the highest score is taken [65]. Wrappers have proved useful as an optimal subset is selected, however, due to the long iterative process they are computationally more intensive and consequently very slow [65, 27]. Figure 6 visually shows how a wrapper approach works; the corresponding description of each step can be found below.

Initial feature space. The initial feature space is the set of features that is available, when starting a machine learning task.

Create feature subset. Creating the new feature subset is done through a search strategy. The three most commonly known search strategies are forward selection, backward elimination and randomized feature selection [50].

Train learning algorithm. Once the new set of features has been selected, training the algorithm is conducted with the new feature space.

Evaluate performance. After each iteration the results of the current learning algorithm need to be evaluated in order to identify whether the process should still go on or be terminated.

As can be seen in Figure 6 creating the new set of features, training the algorithm and evaluating its performance is repeated until the performance decreases based on the current feature set. The three search strategies differ in the way how the feature subset is created.

Liu and Motoda [40] describe the different search strategies as follows: In the case of **forward selection** we start with an empty feature subset and iteratively add a feature. Usually the process stops once the performance, such as the accuracy, decreases. **Backward elimination** starts with all available features and removes one feature at each iteration. Both approaches are heuristic, which means that they do not provide the best solution but a good one, as the potential sub space of features grows exponentially. Lastly, a **randomized feature selection** is a good choice when dealing with a large set of possible features, where forward selection and backward elimination would take too long. A random feature selection algorithm decides randomly which features to use. Often, a randomized feature selection can be the simplest and fastest approach in selecting a subset of features [51].

Embedded methods combine the process of feature selection and model training [39]. More specifically, this means that the learning algorithm automatically selects a subset of features while training the model [39]. A decision tree is an example for an embedded feature selection method as it selects its feature subset through the splits in the tree [40]. Figure 7 visually shows how an embedded approach works; the corresponding description of each step can be found below.

Initial feature space. The initial feature space is the set of features that is available, when starting a machine learning task.

Create feature subset and train learning algorithm. Creating a feature subset and training the learning algorithm is done as one single step since this approach is in the nature of the learning algorithm [39].

Evaluate performance. At the end the performance is evaluated which can be done with various metrics.

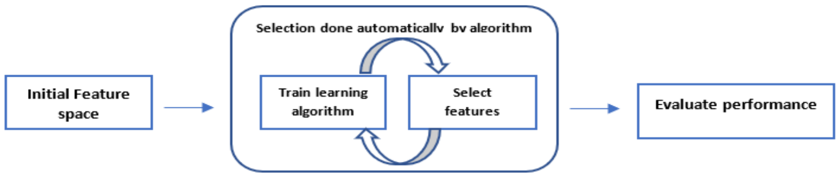


Figure 7: Embedded approach

Selection method	Used	Reason
Filter	√	We use a variance threshold to select features which is considered to be a filter approach. A filter approach is generic and can be used for various learning methods
Wrapper	√	We use random forest which can be defined as wrapper approach
Embedded	√	We use a decision trees and boosted trees which both can be defined as embedded approach
Automated Feature Selection	√	We use WEKA to automatically select features using forward and backward selection

Table 3: Feature selection methods used in the Case Study

Furthermore, several tools for an automated feature selection exist which for instance use statistical calculations to identify the importance of features. The Weka Datamining tool[4] is one of them that can be used for larger datasets. The use of such a tool might be especially useful, when domain knowledge is limited and consequently a feature subset selection is more challenging.

Table 3 summarizes the possible feature selection methods and provides information which of them are used in our case study.

Feature creation

Feature creation is also known as *feature generation*, *feature engineering* or *feature construction*. There exist various definitions for feature creation which slightly differ but in summary still explain the same, namely that new features are created. One definitions describes the transformation of input features into new, more powerful features [65]. Other describe feature creation as the process of generating features from raw data [34, 49]. Similarly,

[4]https://www.cs.waikato.ac.nz/ml/weka/downloading.html

further interpretations define the area as the process of creating new features from original ones [50, 42]. Feature engineering is done in order to achieve two goals: either to reduce dimensionality or improve performance, such as accuracy, visualization and understandability [59, 31, 50, 65]. Reducing the dimensionality will consequently lead to a lower amount of features, whereas performance improvement will lead to more features than in the original data set [65]. There exist various possibilities to create new features depending on the feature type and the goal someone has in mind. The following first of all explains the possibilities to reduce dimensionality and further on continues with different approaches to improve the performance of a model.

Dimensionality Reduction
Principal Component Analysis (PCA) is a often used method in order to reduce the dimensionality of datasets [29]. Alpaydin [5] describes that the original features are replaced by new attributes that are constructed from the linear combinations of the original features. For instance, a two dimensional space can be reduced to a one dimensional space by using a principal components analysis. The PCA identifies the center of the data and rotates the axes in such a way that a projection of the data points leads to one dimension. The principal component is drawn where the data points have the highest variance (spread) and the mapping of the data points have the least information loss. The information loss is the distance that is lost from turning a two dimensional space into a one dimensional space.

Figure 8 through 10 are used to illustrate how a PCA works. The x axis shows the income of a customer and the y axis represents a certain part of the city. We assumed that the more income customers have, the more likely they will live in a better neighborhood. Therefore, a PCA could reduce the dimensions, as it takes income and part of a city and creates a new feature neighborhood that includes the same information.

Performance Improvement
One approach to increase the performance of models is to combine two or more features to one single feature. This might be useful if, for instance, the features from the original feature space do not have any, or low, impact on the model but if combined, they result in a strong predictor. Taking an insurance company as an example, one might think that instead of treating the age of the driver and his/her driving experience in years as two individual features, it might be stronger if the two features are combined resulting in a new feature. With that in mind, drivers that are older could potentially be wrongly classified as not risky whereas young drivers might be classified

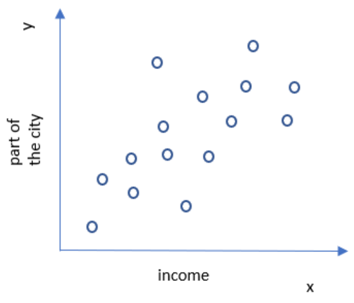


Figure 8: Scatter plot of income and part of the city

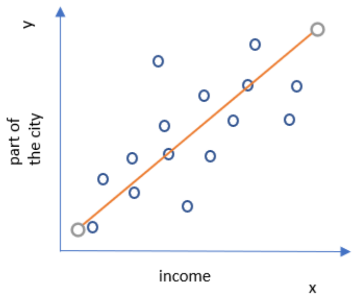


Figure 9: The orange diagonal is the identified principal component

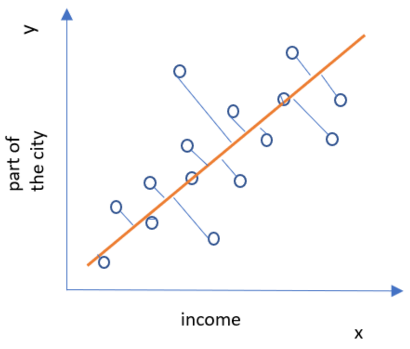


Figure 10: The connections between the principal component and the points is the information loss

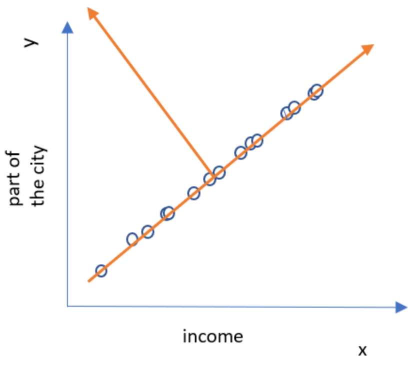


Figure 11: Initial data points in a one dimensional space

as risky. Although, this seems reasonable it still might improve the model if the years of driving experience are combined with the age, as some older drivers might actually not have much driving experience and, therefore, have a higher risk. However, it depends on the learning method whether this approach leads to an improvement or not [25]. Furthermore, quantitative features can be combined using common mathematical operators [65]. Table 4 shows a summary of some possible operators which is adopted from Sondhi [65].

Polynomial combinations are a powerful method in feature engineering. Often, this approach is used if a learning algorithm requires linear separability.

Feature type	Operator
Boolean	conjuction, disjunction, negations, etc.
Categorical	cartesian product
Quantitative	min, max, polynomial combinations, mathematical operators

Table 4: Feature Types and Operators

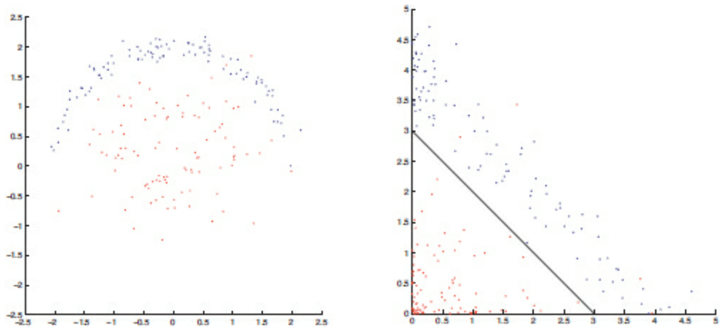


Figure 12: The original data (left figure) can be linearly divided through squaring the values and consequently adding an additional dimension (right figure) [25].

In real-world scenarios a linear separation is often not possible and therefore, needs to be modified. For example, support vector machines are a supervised learning method that use so called **kernels** to add dimensions, which achieve the desired linear separability. Figure 12 was adopted from Flach [25] and illustrates how kernels can change the distribution of the data. It includes samples from two different classes. The left side shows the initial data and demonstrates that a linear separability is not possible as the two classes are compound. He solves the problem by squaring the the original values. Through this action another dimensionality is added to the dataset which consequently makes a linear split possible. This is illustrated by the decision boundary on the right side of the figure.

Real-world datasets usually include a large amount of features, however, only a few of them may be relevant for the target variable. Therefore, choosing the right approach and operators is a major challenge in machine learning. Creating arbitrary features may lead to many irrelevant features which consequently lead to a downgrading in performance, both in speed and in

Feature Generation	Used	Why
PCA	×	no knowledge about features
Feature combination	×	no knowledge about features
Common operators	×	no knowledge about features
Polynomial combinations	×	no knowledge about features

Table 5: Feature generation methods used in the Case Study

predicting accuracy [65, 37]. Table 6 summarizes the possible feature generation approaches. Unfortunately, as the features in the dataset provided by Porto Seguro were anonymized we do not have any knowledge about the features, thus no feature generation methods are used in the case study.

3.2 Tasks

Machine learning can be applied to various tasks which have different goals in mind. Classification, as one of the tasks, was already mentioned earlier. Classification can be used for prediction, where new data is analyzed and according to its features classified into one of the available groups. The machine learning tasks especially differ in their goals; for instance, if a company wants to predict whether or not something will happen, it will rather use classification, whereas if they want to predict a number they will rather use regression. Both tasks are part of *supervised learning*, one of the four machine learning areas. The other three are *unsupervised*, *semi-supervised* and *reinforcement learning.* The following subsections explains each of the machine areas. As the thesis concentrates on a classification task, Chapter 4 takes a closer look at several learning algorithms.

3.2.1 Supervised Learning

James et al. [32] summarize supervised learning as the machine learning area where we have an observation of predictors and the associated response. For each data instance of the predictor variables x_i, i=1,…,n, there is an corresponding response y_i. Roughly speaking, this means that we have a set of data (training set) which includes the predictors and the corresponding response, for instance $\{(x_1,y_1), (x_2,y_2), \ldots,(x_n,y_n)\}$. The predictors are called input or independent variables, and the response is called outcome or dependent variable. The training data is used to teach the learning method how to estimate the outcome variable based on the inputs. Once new data from the testing set is presented to the learning method, it is then compared to the learned results and an estimate is placed. Supervised learning can basically be divided into regression and classification.

Regression problems deal with quantitative predictions, whereas classification deals with categorical features. If a real estate agency wants to predict house prices in the next year, it would be a regression problem. However, if a financial institution wants to predict which customer will default on the loan, it would be a classification problem [32]. Therefore, identifying whether a customer will file an insurance claim in the next year is also a classification problem, where we use predictors such as gender, age, income, or city to predict the response "yes" or "no". As we already mentioned in the introduction, this is a binary classification where we only have two possible classes. So each instance is either assigned to one class or the other. However, if multiple classes are available we call it *multiple classification* [5].

James et al. [32] further explain a well-known problem of supervised learning called the *variance-bias dilemma* which deals with the fact that some models either tend to have high variance or high bias. **High variance** means that the model focuses too much on details which can lead to poor generalization when confronted with new data. On the contrary, **high bias** means that the model misses a lot of instances and does, therefore, not accurately predict the target variable. High variance and high bias are also referred to as *over-* and *underfitting* respectively. Usually, the more flexible a learning method is the higher its variance and the lower its bias will be. It is quite simple to find a learning method that has high variance but low bias or high bias but low variance, but the main challenge is to find a trade-off between the two, such that the model has both, low bias and low variance. If both values are low, the predicting accuracy will increase.

3.2.2 Unsupervised Learning

James et al. [32] indicate that unsupervised learning, in contrast to supervised learning, does not include any information about the target variable. Hence, the training set only contains observations of the predictors x_i, i=1,. . . ,n, but no mapping of the corresponding response. Clustering as one example of unsupervised learning is used for different real-world problems. They provide the example of a market segmentation analysis which gives us information about different preferences, habits and attributes that customers have. Based on the gathered information, we can group each customer into a certain segment, such as customers who spend a lot and those who do not. Supervised learning could be done if we already have the information describing which characteristics of a customer lead to which spending behavior. However, as this information is not available in an unsupervised context we can instead cluster the customers with similar attributes into separate groups. Even though this sounds trivial, it is important to know that real-world problems usually include observations where an overlap between the clusters exist. This makes it challenging to accurately group instances into the right cluster.

3.2.3 Semi-supervised learning

Chapelle et al. [17] indicate that a distinction between supervised and unsupervised learning is usually possible. Nevertheless, a distinction between the two machine learning areas is not always guaranteed, as some instances may include a response variable and others might not. Sometimes gathering data about the response might not be available or it might be too costly to identify it. Therefore, it can also happen that datasets only partially include the target variable. This mix, containing observations with and without a response, is called semi-supervised learning. However, as the goal of semi-supervised learning is to decrease the classification error, it still belongs to the group of supervised learning. Speech recognition is one application where semi-supervised learning is quite often applicable, as it is easy and cheap to record a huge amount of speech. However, finding an associated label is an expensive progress as humans need to listen to all of the recordings and transcript it. In a semi-supervised setting, unlabeled data is needed in huge amounts, as it carries less information than labeled data. Therefore, semi-supervised learning methods need to be fast and efficient and be able to deal with a large number of observations.

3.2.4 Reinforcement Learning

The last category of machine learning is reinforcement learning. It differs from the other areas in the sense that there is no data available from which the algorithm could learn such as in a supervised settings. Nor is there a possibility to learn through identifying similarities between features and grouping them together, such as in an unsupervised setting. Sutton and Barto [67] explain reinforcement learning as the machine learning approach that learns how to do something through trial and error. It maps situations to actions. In a reinforcement learning setting we basically distinguish between two parties: The agent and the environment. If we apply this to the real-world we could say that learning how to walk or learning how to ride a bicycle is similar to reinforcement learning. When we learn to ride a bike, we

Machine Learning Task	Used	Reason
Supervised	√	The case study deals with a binary classification task where either one class or the other needs to be predicted
Unsupervised	×	labeled data is provided
Semi-supervised	×	labeled data is provided
Reinforcement	×	n.a. for this dataset

Table 6: Feature generation methods used in the Case Study

will probably fall down a couple of times and stand up again until we master it. The same can be applied for reinforcement learning.

Additionally, they describe that the agent is the computer system that learns through doing a given task in the environment. Specifically, it means that the agent learns through its own actions and experience. The environment reveals its current status to the agent, afterwards the agent performs an action which changes the status of the environment. The goal in reinforcement learning is to maximize the quantitative reward which the agent gathers along the way. The more it learns during the process, the more it will know about the environment. It can use this knowledge to remember what it has already explored in the environment to continuously collect rewards, but also needs to explore new possibilities to expand its knowledge. However, this results in a trade-off as exploring new areas can lead to a lower reward. Recently, reinforcement learning has been used in various fields, including game theory, robotics or multi-agent systems.

3.3 Models

Models in general can be grouped into three different categories, namely *geometric models*, *probabilistic models* and *logical models*. They are the results that are achieved through the learning process of a machine learning task [25]. This section describes each model in more detail.

Geometric Models

A geometric model is created in the instance space, which is the set of all possible instances, using common concepts available in geometry such as lines, planes and distances to eventually build the model. More concretely, a geometric model uses a coordinate system and plots features in a d-dimensional space. In order to create the final model lines, planes or distances can then be applied to solve a machine learning task. Geometric models deal with numerical data, hence other feature types need to be transformed into quantitative features [25].

One category of geometric models are linear models, which in general can be used for all predictive tasks in machine learning, such as classification and regression. To create a linear model, several learning methods are possible. However, one example are support vector machines which set a linear decision boundary between instances where the distance from the decision boundary to the closest data points is the largest [25]. An alternative to linear models are distance based models. Distance based models identify similarities based on the closeness of data points. It is assumed that the closer two instances are, the more similar they are and thus can be placed in the same segment. The nearest neighbor classifier is one of the most known distance based models, that compares new instances with already available ones and classifies them based on their closeness [25].Chapter 4 describes support vector machines and nearest neighbors more in detail.

Probabilistic Models

Probabilistic models identify the relation between independent and dependent variables and are used to predict a certain outcome with regard to its probability [25]. They learn the mapping of a set of predictor measurements x_i to the associated output y_i [29]. The goal is then to apply the knowledge learned from the mapping to a new set of data instances which consist of the predictor measurements. These are then mapped to a corresponding target response which is not known at the time of the mapping [29]. Probabilistic models use probabilities, such as posterior or prior probabilities and likelihoods to predict a certain outcome. Posterior probabilities are used if a feature X has already been observed. Based on this observation the target variable is predicted. The likelihood observes the probabilities of an event to occur, such as 'how likely is it that a 20 year old male customer will file a claim in the next year?'. Prior probability looks at the probability before a value has been observed, which means we do not have any evidence yet [25].

Logical Models

Logical models are those models that can be easily interpreted by humans as they use logical expressions to create segments of the instance space. The

Model	Used	Reason
Geometric	✓	We use the K-nearest neighbor classifier which is a distance based model.
Probabilistic	✓	We want to predict whether a customer will file a claim in the next year which is a probabilistic model.
Logical	✓	We use learning methods such as random forest and boosting which are based on several decision trees and then averaged. Decision trees are logical models.

Table 7: Model types in the Case Study

purer or more homogeneous these segments are, the better the results of the machine learning task will be. For the segmentation they use non-numerical features, however numerical values can be transformed into boolean ones by using thresholds [25].

One example of logical models are tree models, such as decision trees. They use splits to choose between possibilities and consequently divide the dataset into segments [25]. More concretely, this means that each split results in another segment and the deeper it goes the more homogeneous they will be [25]. For instance, the dataset can be divided into two splits, one grouping female customers and the other grouping male customers. If a new customer is male we can just follow the corresponding path. If such a tree contains all possible features it is called complete. However, such a complete tree can lead to overfitting (or high variance), which means that the model cannot generalize well and leads to misclassified results when confronted to new data, as the model only concentrates on the features it knows [32]. On the contrary, underfitting (or high bias) means that too less features are taken into consideration and therefore, even classifying the data in the training set is prone to errors [32]. Recall that it is important to find a good balance, such that neither overfitting nor underfitting occurs. It is worth noting that, tree models have the advantage that they can be used for a wide variety of tasks, such as for classification, regression and clustering [25].

We basically grouped the models into three categories. However, a model does not need to only be part of one such group. With that in mind, a model can be a geometric and probabilistic model as well, depending on the machine learning task. Table 7 summarizes the three types of models and indicates if it is used in the case study.

3.4 Summary

This chapter looked at features, the different types of them and the possible statistical analyses that can be conducted with each type. In addition, we identified that the feature subspace can often be minimized either through the right selection of the available features or the generation of new features. Both can lead to improved performance and comprehensibility. Moreover, we introduced the four different areas of machine learning which differ in the way how the learning process is conducted. Looking at supervised, unsupervised vs. semi-supervised learning we emphasized that the difference is the availability of an outcome variable. Reinforcement learning, on the other hand, is a completely different approach and requires and interaction of an agent with its environment which often is used in robotics. Finally, we also looked at models which are the outcomes that learning methods produce. They can be a combination of logical, geometric and probabilistic nature.

All three areas are very important for achieving good results in machine learning and according to Flach [25], they can be seen as the three pillars of machine learning. As the goal of the thesis is to perform practical experimentations in classification, the next chapter introduces several learning methods more in detail.

4 Classification Methods

Classification is part of supervised learning and its goal is to correctly classify data instances into one class. The dataset provided by Porto Seguro is one example of classification as it focuses on identifying potential claiming customers. Recall from Section 3.2.1 that supervised learning means that for each predictor x_i there is an associated response y_i. A good classification method needs to perform well on the training data, but more importantly also on new test data. This is a tough challenge, as one needs to find a good method for the data and make sure that training the model does not go too much into detail, as then the algorithm would perform poorly on new data. Wolpert [72] emphasized that "there is no such thing as the free lunch theorem", which means that there is no single learning method that performs best over all other methods for any given dataset. Therefore, the process of finding the right approach in dealing with a classification problem is often characterized by trial. This chapter looks closer into some of the classification methods, starting with simpler one and finalizing it with more complex algorithms. James et al. [32] provide an excellent overview of the classification algorithms, therefore, it is a main source of this chapter.

4.1 K Nearest Neighbor

According to Smola and Vishwanathan [64], the K Nearest Neighbor (KNN) uses distance measures between two observations (data instances). The label of an unknown data instance is identified by looking at its k-nearest neighbors. In its most basic form it uses only the nearest neighbor and assigns the corresponding class of this neighbor to the new instance. However, as this could include noisy data and wrong classifications, it is more practical to consider more than one neighbor and assign the majority class of the k neighbors. If *k=3* we would consider the three nearest neighbors and assign the class that occurred the most often to the new instance.

An often used measurement for evaluating the distance between the instances is the Euclidean distance that is the square root of the sum all squared distances on a coordinate systems [25]. If we have two data points in a two dimensional coordinate system with the coordinates q=(3,6) and p=(4,8) then the Euclidean distance would be $\sqrt{(3-4)^2+(6-8)^2}$. If the values are numeric it is as simple as above. However, categorical values, such as colors, need to be transformed into a numerical representation. Therefore, Witten and Frank [71] propose to treat equal categorical values with a distance of 0 and non equal values with 1. That means that the distance between 'green' and 'green' is 0, whereas the distance between 'green' and 'blue' would be 1. In a more precise way, one could treat closer colors, such as 'orange' and 'yellow', with a shorter distance than 'orange' and 'blue'.

Smola and Vishwanathan [64] further indicate that classification using KNN can become very computationally intensive if there is a high number of observations or high dimensionality, as the instances need to be stored in order to allow a distance based analysis of new data to old one. Therefore, in order to decrease the computational costs, not all observations may be used, or the dimensionality may be reduced. Considering only a randomized subset is a possible approach, as the high dimensional space can be reduced to a lower one through projection. This allows a mapping, such that the distance between two points only changes by an acceptable factor. Therefore, the projection will consequently result in an increased speed and decreased amount of storage. However, choosing which of the instances to consider in the model is a key challenge in instance-based[5] learning algorithms.

4.2 Support Vector Machines

James et al. [32] indicate that in a classification problem, support vector machines (SVM) are seen as one of the best "out-of-the-box" classifiers. In the SVM setting, there are, however, three main possibilities or options that can be used which differ depending on the data. The simplest option is the *maximal marginal classifier* that separates the classes through a hyperplane, which is a linear decision boundary. However, as linear separability is rarely the case in a real-world scenario, the *support vector classifier* has been introduced as an extension to the maximal margin classifier. It deals with cases where a clear linear decision boundary is not possible. Even though the support vector classifier already showed improved results compared to the maximal margin classifier, it still, however, leads to poor results if it deals with non-linearity. Therefore, the *support vector machines* were introduced as an improvement of the support vector classifier. SVMs usually perform well on non-linear classification problems. The following explains each option more in detail.

An essential tool of support vector machines are hyperplanes, which are used to divide the data into n classes. In a binary classification setting this means that a hyperplane is used to identify the decision boundary between two classes. In a d-dimensional space, a hyperplane represents a subspace of d-1 dimensions. More concretely, a hyperplane in a two-dimensional space is only a subspace with a single dimension. Hence, it is a line. This would further imply that in a three-dimensional space a hyperplane is a two-dimensional subspace, that we call a plane. Data points that lie above the hyperplane are labeled as instances of one class, whereas instances below the hyperplane are labeled as instances of the other class.

Maximal Marginal Classifier
According to James et al. [32], there exists an infinite number of possible hyperplanes for data that can be linearly separated as the decision boundary can be slightly moved and rotated arbitrary times. However, to find the best place of a hyperplane and consequently the optimal space between two classes, we can use the maximal margin hyperplane, which is the one hyperplane that is the farthest away from the training instances of each class. The

[5]Instance-based learning methods store only specific instances that are used for training the algorithm and compare new instances to the stored onces [3].

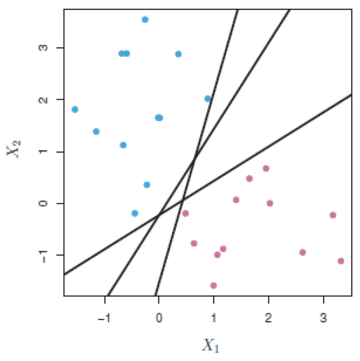


Figure 13: Three possible hyperplanes [32].

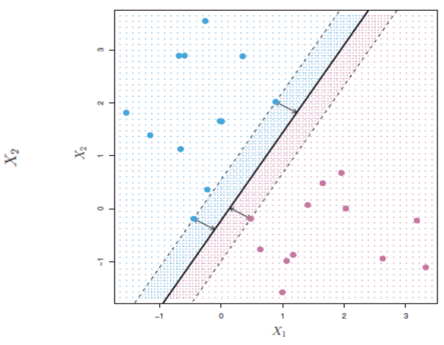


Figure 14: The hyperplane with the maximal margin between the instances [32].

margin is the minimal distance between the hyperplane and an observation. If we are looking for the maximal margin hyperplane, we are interested in the hyperplane that maximizes the margin, hence maximizes the distance between the observations of both classes and the hyperplane. The maximal distance between hyperplane and the instances is important, so that the model generalizes well [5]. If the distance is smaller, new instances could more likely be wrongly classified. With larger distances the risk of wrong classifications is minimized [5].

James et al [32] further explain the concept of hyperplanes and the maximal margin by introducing examples which can be seen in Figure 13 through 15. Figure 13 shows three possible hyperplanes of a dataset and we can already identify that not all of them satisfy the maximal margin between the instances. Figure 14, however, shows the hyperplane that has the maximal margin; which is the largest distance that is closest to the observations. In Figure 14 the margin is the distance between the dashed lines and the hyperplane. Furthermore, the two blue and one purple point are the *support vectors* that are the closest points to the hyperplane. Note that the support vectors are insofar important to the hyperplane as they support the separation. This means, if they are moved, the maximal marginal hyperplane changes as well [32].

Support Vector Classifier
We already mentioned, that the maximum marginal hyperplane requires a

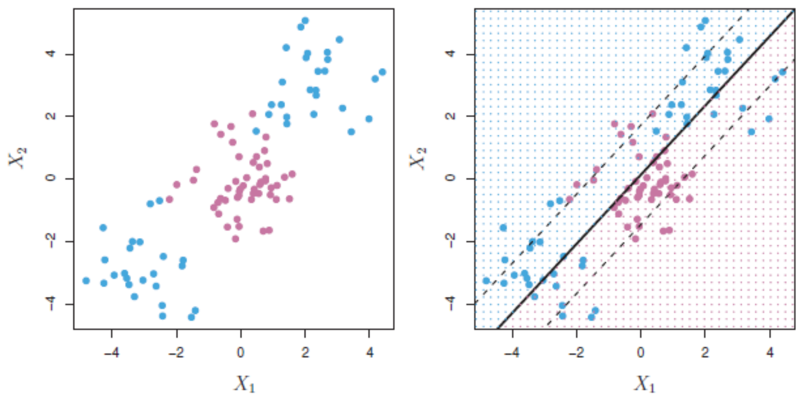


Figure 15: Separation of the data based on the support vector classifier [32].

linear separability between two classes, as shown in Figures 13 and 14. However, as linear separability is quite rare in real-world problems, the maximal margin classifier as a learning algorithm is not the best choice. Furthermore, as this method enables a perfect partition between two classes, it tends to overfit the data [5]. Recall that overfitting is not beneficial for a model as it performs poorly when it comes to the classification of new data as it can not generalize its learning. James et al. [32] indicate that the support vector classifier has been introduced as an extension to the maximal margin classifier. This classifier does not overfit the data as it does not perfectly classify the instances. This means it accepts that some of the instances are misclassified, so that a better robustness of the overall model is created. According to Alpaydin [5], most of the new instances can therefore be correctly classified which consequently leads to a better performance of the algorithm.

Figure 15 shows an example of a support vector classifier fit on non-linear data that is adopted from James et al. [32]. They classified the data using a support vector classifier. Looking at the right side of the figure, we can identify that it classified the instances poorly as no clear separation of the data instances is achieved. However, we could increase the feature space by using quadratic or polynomial functions in order to deal with non-linearity and improve the performance.

Support Vector Machines
James et al. [32] explain that improving the performance by increasing

the feature space through polynomial functions for example, could, however, quickly result in an immense increase in complexity as we do not know how many dimensions need to be added to make the data linearly separable. Consequently, this would lead to an increase in computational costs. Therefore, support vector machines have been introduced as an improvement to support vector classifiers. They are able to deal with non-linear relations by increasing the feature space automatically with the help of *kernels*. A **kernel** is a similarity function that takes two instances as input and outputs their similarities by using the inner product [32]. The inner product uses two values of a vector space as inputs and outputs a number, that could for instance be the dot product [61]. It is a similar approach to the one described above, where a support vector classifier enlarges the feature space by using quadratic or polynomial functions, except that using kernels is less computationally intensive. Various kernels can be used with support vector machines, such as linear, polynomial and radial kernels. Using an SVM with linear kernels leads to the same result as a support vector classifier [32]. The linear kernel evaluates the similarity between two observations using the Pearson correlation, which measures the linear correlation between an input and output variable [32].

James et al. [32] further explain that creating more flexible methods can be achieved through polynomial kernels with a degree of d. However, a degree of 1 leads to the same results as a support vector classifier or a linear kernel. Another possibility are radial kernels, that look at the distance between train and test observations. This means that training observations that are far away in an Euclidean distance do generally not effect test observations, as the radial kernel only considers nearby instances for its classification [32]. Figures 16 and 17 were adopted from James et al. [32] who improved the classification from Figure 15 by using polynomial and radial kernels respectively.

4.3 Decision Tree

The decision tree is a hierarchical learning method that uses the "divide and conquer" strategy and divides the region into n subspaces [71]. Roughly speaking, it divides the entire data into different sections, using some independent attributes as splitting criteria [71]. Each decision tree consists of *decision nodes* and *decision leafs*. The **nodes** are all splits within the decision tree that are not the resulting outcome [5]. With that in mind, classifying the correct class of a new instance means that the evaluation starts at the root of the decision tree and goes deeper down the nodes until the final

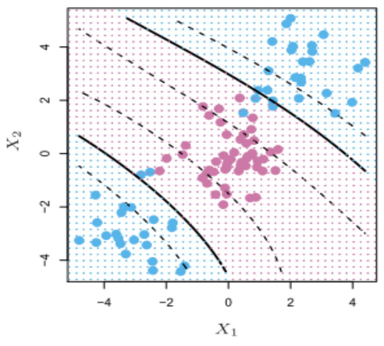


Figure 16: A polynomial kernel of degree three applied to a SVM [32].

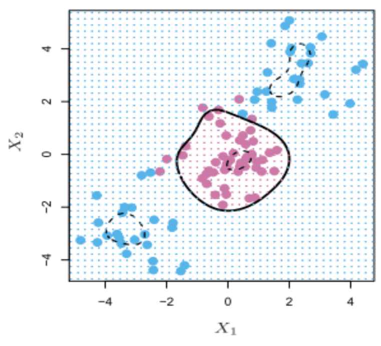


Figure 17: A radial kernel applied to a SVM [32].

outcome, which is the **leaf**, is met [5]. The goal of decision trees is to find the optimal split, with the goal of maximizing the purity of a node [71]. Recall that the maximal purity of a node is achieved if the node contains only one class. A well-known measure of identifying a good split is the cross-entropy [5, 32]. An alternative measure of the split impurity is the gini index. Both, gini index and cross-entropy, can range between 0 and 1. The purer the split is the smaller the number of the both will be [32]. These two measures can be used to prune the tree and increase the quality of the splits [32].

According to Witten and Eibe [71], one main advantage of decision trees is that it can deal with all feature values, as it transforms the values into splits having two or more directions. For instance, a quantitative feature can take on a binary split, such that all data instances can either walk down the *less than* or *greater than* way. Alternatively, an additional direction, such as *equal to*, could be introduced if a certain value is important. On the contrary, qualitative features could divide the observations into the possible categories. For example, if the gender is considered to be important, the decision tree could split the observations into *female* and *male*. A qualitative feature with more than two possible values could likewise use the available values as splits.

James et al. [32] indicate that decision trees have many advantages, such as that they are easy to read and interpret for humans as they are logical models. Furthermore they can be graphically visualized and often read from non-experts as well. Lastly, as was mentioned above, decision trees are able to work with all feature types, which means that we do not need to create dummy variables. However, sometimes other learning methods have a better predicting power and outperform decision trees. Furthermore, decision trees are very sensitive to the training data, which means that the results may differ if the training data is changed. To improve the prediction power of decision trees, we can create many trees and combine them. This approach is referred to as **ensemble methods** and the main goal is to combine weak learners so that strong learners are developed. Through ensemble methods such as *bagging*, *random forest* and *boosting*, decision trees can be aggregated resulting in a stronger performance.

4.4 Bagging

Decision trees are a quite flexible learning method and hence need to deal with high variance. Differences in the training data result in differences in the model [32]. Recall from Section 3.2.1 that high variance can result in overfitting which leads to poor classification on unseen data. Therefore, we can use *bagging*, also called *bootstrap aggregation*, in order to decrease the variance.

According to Flach [25], bagging is a powerful method in the context of decision trees as the predictive power of a single tree can be improved through combining many of them and averaging the results. However, bagging can be applied to many other supervised learning methods as well. Bagging creates n different training subset which are constructed by selecting different observations of the training set. The observations within each subset might be picked more than once, thus there might be duplicates but it ensures that each model is different. The individual decision trees are each trained on different subsets which means that if we have ten decision trees we also have ten different training sets. Consequently, we also receive ten different models. In the case of regression, the mean is taken to identify the corresponding label. However, for classification purposes the majority vote is taken in order to predict the label. That means if we query an observation x we look at the responses among the n subsets and take the class that occurred the most often. Additionally, bagged trees do not necessarily need to be pruned as using multiple trees with different training sets decreases the variance and hence the risk of overfitting. Although bagging leads to improved performance compared to a single decision tree, it also decreases the interpretability of the model in contrast to a single decision tree [32].

James et al [32] indicate that each of the n subset only include a fraction of the overall training data and some might be duplicates. The remaining observations are called *out-of-bag* (OOB) observations. These remaining OOB observations can be used to compute the test error of the model. If we want to predict the response for the ith observation, we can use all the bagged trees in which this ith observation was OOB. Afterwards, the majority vote is taken and then the classification error can be computed to measure the test error, which is a measure to evaluate how accurate the model classifies on new data (test data). This is a valid approach for calculating the test error, since we use the OOB observation that was not used to fit the model.

4.5 Random Forest

According to James et al. [32], a random forest leads to improved results over bagged trees. Although bagging decreases the variance through combining many decision trees, it still tends to repeatedly use strong predictors over weak ones in each subset, as it tries to minimize the error. Therefore, strong predictors are more frequently used at the beginning of each decision tree. Hence, the predictions tend to correlate. Random forests similarly take different data instances of the training dataset and create n different subsets, but when they train the model, only a certain number of randomly selected predictors is taken. In contrast, bagged trees use all of the predictors and strong predictors are usually among the first splits of each decision tree. Therefore, the trees tend to correlate with each other. The amount of random predictors for each decision tree can be specified, but a usual default value is $m = \sqrt{p}$, where m is the the number of the randomly selected features and p is the number of all features in the dataset. Moreover, a random forest which uses the maximal amount of features in each decision tree can be compared to bagged trees. They further point out that a small number of predictors is helpful when dealing with a lot of correlated predictors. Nevertheless, there are similarities between bagging and random forest. For instance, the out-of-bag error can likewise be used to evaluate the model performance. However, the main difference to bagging is the choice of the predictors, since random forests improve the accuracy of the model by using random, uncorrelated and weak predictors.

4.6 Boosting

Boosting, similarly to bagging, can be applied to any statistical learning method for regression and classification [32]. However, we look at boosting in the context of decision trees, to align it with bagged trees and random forest.

In Section 4.4 we discussed that bagging builds n subsets of data observations, which are taken out randomly and with replacement[6] from the original dataset. A decision tree is then fit to each subset and the predictions are aggregated, finally resulting in a single prediction. According to James et al. [32], boosting works similar to bagging as it also creates n numbers of subsets, where the first subset of observations is selected randomly. However, the further process differs as boosting additionally puts focus on wrongly classified instances. In detail, after the first subset is created, a decision tree is fit on the data. Then all observations from the initial training dataset are used to evaluate the performance of the first decision tree and often some of the observations will have a significant error. Afterwards, a second dataset using randomly selected observations is built, but in addition some of the misclassified instances are considered as well. The decision tree is again fit to the data and the performance is evaluated using the original dataset. This process is repeated until the number specified trees is reached. As we focus on instances that were misclassified at each iteration, boosting is considered to be a slow learner. However, evidence shows that slow learners also lead to better performance compared to other methods.

Even though bagging and random forest are not prone to overfitting, boosting can overfit if the number of trees is too high. Therefore, when working with boosting we need to consider three important parameters and the number of trees is one of them. In order to identify the number of subsets we can use *cross-validation*, a method that is typically used for the evaluation of model performance or as a resampling method. Secondly, boosting requires a learning rate which defines the speed of learning. Usually, it takes on values of 0.01 and 0.001. The slower the algorithm learns the less prone to overfitting it is. The last important parameter is the number of splits of each decision tree, as it defines the complexity. The more splits a tree has, the more likely it is to overfit. Often, only a single split is used in boosting which is then called a *stump* rather than a tree. Generally, it is sufficient to only use stumps instead of more complex trees as boosting considers wrongly classified instances in the next iteration. In addition, less splits can increase the comprehensibility of the learning method.

4.7 Cross-Validation

According to Alpaydin [5], a common challenge in machine learning is the trade-off between training and testing data. Recall from Chapter 2 that train-

[6]Data observations can be picked more than once.

ing data is used to train the model and the test data to apply the learned knowledge on unseen data. It is important to know that we can only analyze the performance of the model once it is applied to unseen data. However, the test dataset is usually not initially used to check the performance of a classifier, rather another technique is used. By applying *cross-validation* we can divide the training data into two sets - one for training the learning method and the other as a "representation" of the testing dataset to evaluate the performance. This is the validation set which we also introduced in Chapter 2. It is beneficial to evaluate the model on unseen data as this is a good indicator for future events. For instance, cross-validation could split the training dataset into a 70:30 proportion, where 70% are used for training and the remaining 30% for validating. However, this indicates that if an instance is in the training set in cannot be in the validation set and vice versa. Therefore, an extension to cross-validation is *k-fold cross-validation* that partitions the training data into k bins, which are equally sized sets of data observations. K can be any number greater than two but generally it is either ten or thirty. In k-fold cross validation we use one of the k bins as the testing set and the remaining *k-1* as the training data. This process is eventually repeated k times so that each bin was used as a validation dataset. With this in mind, a dataset containing 2,000 instances a 10-fold cross validations splits this dataset into ten equally sized bins, each containing 200 instances. The first 200 are used as a validation dataset and the remaining 1,800 are used as the training set. The model is fit on this data and validated on the 200 instances. This is repeated ten times and at the end the average of the ten test performances is taken, resulting in the final performance. K-fold cross validation requires more computational time because the learning process is repeated k times. However, it eventually increases the performance of the learning method.

He further indicates that an extreme case of k-fold cross-validation is *leave-one-out cross-validation* that uses, instead of k equally sized sets, only one instance as testing data and the remaining instances as training data. Similarly, splitting the data into training and testing set is repeated k times, where k=N and N is the total amount of instances in the dataset. However, k-fold cross-validation is usually used if labeled data is difficult to obtain. Again, assuming a dataset with 2,000 instances, we would run the learning algorithm 2,000 times so that each instances is used as a test set. At the end, the average of the error is calculated leading us to the final performance evaluation.

A classification task can be solved by many different algorithms, some of them have been introduced in this chapter. Depending on the dataset, one learning method can be better than another. Unfortunately, as this depends on the data it needs to be tried out which method leads to the best results. Nevertheless, ensemble methods have shown to be good approaches, as they combine multiple methods and average the outcomes, usually resulting in a better performance. The next chapter applies several of the above mentioned methods to the dataset provided by Porto Seguro.

5 Application of Classification Techniques

Since we use Action Research as our main research methodology, this chapter is structured accordingly. It includes three main steps: Look, Think and Act, the detailed steps can be reviewed in Figure 1 of Chapter 1.3. At the beginning of this chapter the different evaluation metrics that we use in the case study are introduced, while the further sections subsequently explain the approaches conducted in the practical assessment. They are grouped in iterations and follow the Action Research steps. From Section 1.2 we can recall that our dataset was provided by the Brazilian car insurance company Porto Seguro which published historical data on Kaggle and asked the community to predict which customer is more likely to file a claim in the next year.

5.1 Evaluation Metrics

Accuracy, precision, recall and *f-score* are well known and used evaluation criteria of classification problems [70]. This chapter provides a definition and an example for each one of them, as they are used for evaluating the performance of the classifiers of the case study in Chapter 6.

For a binary classification problem the results usually fall into four categories: *true positives (TP)*, *false positives (FP)*, *true negatives (TN)* and *false negatives (FN)*. Often, these four values are represented in a **confusion matrix**, an example of it can be seen in Table 8. The diagonals (TP and TN) in a confusion matrix are the values that are correctly predicted. With the help of these four categories accuracy, precision an recall can be calculated. **True positives** are the samples that are correctly predicted. **False positives** are data samples that are classified as true but in reality are false. **True negatives** are observations that are indeed false, whereas **false negatives** are data instances that are wrongly classified as false. Accuracy is defined in Definition 1 [11].

		predicted value	
		p	**n**
actual value	**p**	True Positive	False Negative
	n	False Positive	True Negative

Table 8: Confusion matrix

Definition 1. *Accuracy.*
The accuracy is the proportion of correctly classified results among all instances.
Accuracy is calculated by $\frac{TP+TN}{TP+TN++FP+FN}$.

In some classification tasks accuracy can give reliable information about the performance of a model. However, it can also lead to a false impression of the performance because in unbalanced datasets the accuracy can be high although no predictions on the minority class have been done. To be more clear we can look at Example 1.

Example 1. *Accuracy.*
We have 100 insurance holders and only 10 of them should be classified as filing a claim. If an algorithm classifies all of the customers as negative then its accuracy is 90% although none of the claiming customers was identified.
Accuracy is calculated by $\frac{90+0}{90+10+0+0} = 0.9$.

If the dataset is unbalanced as in Example 1, it gives a false impression of the classifier's performance. Here, no customer has been identified to file an insurance claim; however, we still achieve an accuracy of 90%. This phenomenon is called the *Accuracy-Paradox* [73]. Therefore, precision, recall and f-score are used to evaluate the correctly predicted observations. The terms are defined in Definitions 2, 3 and 4 [68].

Definition 2. *Precision.*
Precision is the fraction of the total number of samples that are relevant among the total number of retrieved samples . Precision is calculated by $\frac{TP}{TP+FP}$.

Definition 3. *Recall.*
Recall is the fraction of the total number that are relevant among the total number of relevant samples in the entire data set. Recall is calculated by $\frac{TP}{TP+FN}$.

To be more concrete, we can look at an example:

Example 2. *Precision and Recall.*
Amongst 100 insurance holders in the training data 30 customers should be classified as positive of filing a claim. The algorithm identifies 25 customers as positive, whereas only 10 of the 25 are indeed correct (TP) and the other 15 are wrong (FP). 20 relevant customers are not identified (FN) and the other remaining 55 are correctly classified as not relevant (TN).

Precision is calculated by: $\frac{10}{10+15} = 0.4$
Recall is calculated by: $\frac{10}{10+20} = 0.33$.

Based on the domain, we can focus on either improving precision or recall. We assume that an insurance company is probably more interested in increasing the recall, as they want to identify as many customers as possible of filing a claim. On the contrary they will more likely accept customers that are wrongly classified as risky rather than misclassifying customers as not risky. Often, improving recall means decreasing precision. Last but not least, we can calculate the f-score.

Definition 4. *F-score.*
The F-score is the average of precision and recall, to better compare the two learning methods against each other. F-score is calculated by
$2 * \frac{precision * recall}{precision + recall}$.

Using the same setting as in Example 2, we can calculate the f-score from the results that were received:

scale AUC	Performance
0.5 - 0.6	poor
0.6-0.7	fair
0.7-0.8	good
0.8-0.9	very good
0.9-1	excellent

Table 9: AUC value interpretation

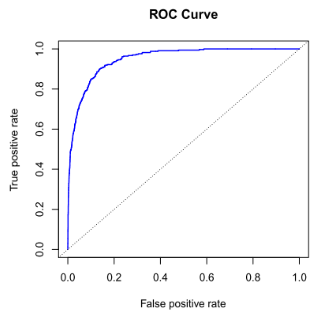


Figure 18: ROC curve with 0.95 AUC [32]

Example 3. *F-score.*
F-score is calculated by: $2 * \frac{0.4*0.33}{0.4+0.33} = 0.36$

According to James et al. [32], it is also common to use the *Area Under the Curve (AUC)*, which shows the overall performance of a classifier. It uses the true positive and the false positive rate and puts it against each other to define the performance. Furthermore, the AUC is often plotted with the help of a *Receiver Operating Characteristic curve (ROC curve)*, an example can be found in Figure 18, that shows the performance of a classifier which achieves a good AUC value of 0.95. The figure was borrowed from James et al. [32]. The true positive rate and the false negative rate are defined in Definition 5.

Definition 5. *Sensitivity and Specificity.*
The true positive rate is equivalent to the recall and also called sensitivity. The false positive rate is also called specificity and shows how many of the true negatives have been identified.
Specificity is calculated by: $\frac{TN}{TN+FP}$

According to Allaire [4], the AUC value can be interpreted according to the scale in Table 9. We can see that the value can range between 0.5 and 1 and the higher it is the better the classification algorithm performs. Using the same setting from Example 2 we can calculate the false positive rate, which is seen in Example 4.

Example 4. *Specificity.*
The specificity is calculated by: $\frac{55}{55+15} = 0.78$

5.2 Iteration 1

Iteration 1 includes a detailed analysis of the dataset and its features. It uses statistics and visualizations to explore the value ranges, frequencies and a potential impact on the outcome of the target variable.

5.2.1 Look & Think

The Look step of the Action Research process includes defining the problem and gathering information on it. However, the problem was already defined by the Kaggle competition and focuses on identifying which customer will place an insurance claim in the next year. We were provided with the historical data of customers and we were informed about the facts that the data was anonymized, categorical features were already transformed into a quantitative scale and missing values were categorized as "-1". Additionally, more data was gathered through analyzing discussions and implementations from other contributions on Kaggle or forums.

As part of the first iteration, we looked at many different classification algorithms and came to the conclusion that a random forest is a good choice for our analysis for the following reasons: Ensemble methods, which are methods that combine different models, are considered one of the most powerful methods in machine learning as they use a combination of models rather than a single model [25]. In addition, random forests include a certain amount of randomness which can benefit the overall model by reducing noise. Furthermore, in statistics it is also well known that averaging a set of measurements can lead to more stable results than only a single measurement [12]. Lastly, it is claimed that ensemble methods do often receive a higher accuracy and are therefore often outperforming single learning methods [8].

Initially, we planned to use a Jupyter notebook[7], which is an Integrated Developer Environment (IDE), locally on our machine. It provides you with different features, such as the integration of HTML. However, due to performance issues we decided to directly work on a Jupyter notebook in Kaggle as each participant has the opportunity to work online on one notebook per

[7]http://jupyter.org/

```
for f in train.columns:
        if 'bin' in f or f == 'target':
                level = 'binary'
        elif 'cat' in f:
                level = 'nominal'
        elif f == 'id':
                level='id'
        elif train[f].dtype == float:
                level = 'interval'
        elif train[f].dtype == int:
                level = 'ordinal'
```

Listing 1: Grouping of feature type

competition. The participants can decide whether they want to publish their notebook within the competition or keep it as private. We created a repository on Github[8] so that the work can be accessed. For our practical part we used python and mostly made use of the scikit-learn (short: sklearn)[9], pandas[10] and numPy[11] libraries. As we worked on the online notebook, we did not need to download the dataset and upload it into a locally stored notebook. The data was made available on the homepage, therefore, we could use it from there. Our first step was to read the dataset into our notebook by using the pandas read statement:

```
train = pd.read_csv('../input/train.csv')
```

We assigned the training data to the variable "train" and stored it as a pandas data frame[12] as can bee seen from the line of code above. This allowed us to access many methods for our analysis, which are described further down. We had fifty-nine features in the dataset, including the ID and the target variable. We already mentioned that Porto Seguro rephrased the names of the features to ensure the privacy of its customers. Therefore, we did not know what the features exactly were in the real word, but we knew that some features belonged to the same group through the prefix in the feature name. Table 10 shows a summary of the prefixes, which was clarified by Adriano Moala [48] within a discussion on Kaggle. By analyzing the table we can conclude that there were features which provided data about the drivers, cars, regions and some calculated features. It was further explained that the dataset included binary, categorical, continuous and ordinal features.

By using the `head(10)`[13] function from the pandas library we could view the

[8]https://github.com/SanjaJo/PortoSeg

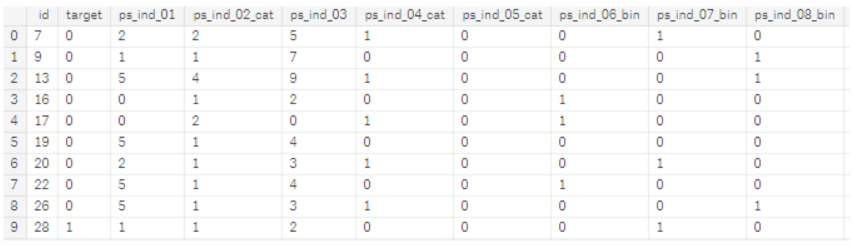
[9]http://scikit-learn.org/stable/

[10]https://pandas.pydata.org/

[11]http://www.numpy.org/

[12]https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html

[13]https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.head.html



	id	target	ps_ind_01	ps_ind_02_cat	ps_ind_03	ps_ind_04_cat	ps_ind_05_cat	ps_ind_06_bin	ps_ind_07_bin	ps_ind_08_bin
0	7	0	2	2	5	1	0	0	1	0
1	9	0	1	1	7	0	0	0	0	1
2	13	0	5	4	9	1	0	0	0	1
3	16	0	0	1	2	0	0	1	0	0
4	17	0	0	2	0	1	0	1	0	0
5	19	0	5	1	4	0	0	0	0	0
6	20	0	2	1	3	1	0	0	1	0
7	22	0	5	1	4	0	0	1	0	0
8	26	0	5	1	3	1	0	0	0	1
9	28	1	1	1	2	0	0	0	1	0

Figure 19: First ten rows of training data

Prefix	Meaning
ind	individual, driver
car	car
reg	region
calc	calculated features
bin	binary features
cat	categorical features
no prefix	continuous or ordinal features

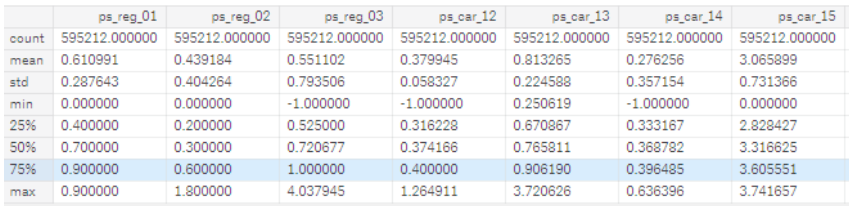
Table 10: Feature labeling

first ten rows of the training data. This allowed us to conduct a first analysis of the feature names, its values, and ranges and can be seen in Figure 19.

As a next step, we grouped features of the same type together to enable a better interpretability. More concretely, all binary, categorical, continuous and ordinal features were put into individual groups. The code for the grouping was adopted from Bert Carremans [15] and can be seen in Listing 1. After the grouping, we assigned each level to a new variable, which allowed us to analyze the individual feature types.

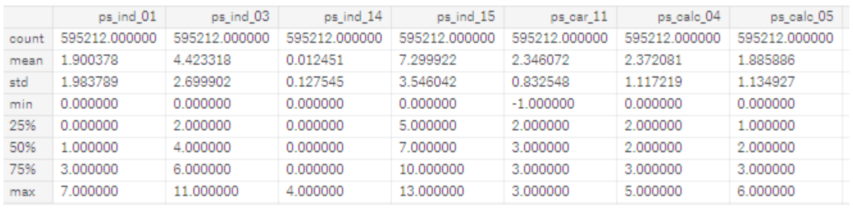
Statistical Analysis
In Section 3.1.1 we introduced some statistics, such as the mean or the mode, which are used to analyze data and which are an important step to get familiar with the data. Figures 20 and 21 show an excerpt of the `describe()`[14] function from the pandas library to get familiar with the distribution of the data. The results seen in Figures 20 and 21 give us already an idea about the data. More concretely, we can see that some of the features have a missing value as the minimal value is -1. From the maximal value we can derive the scale of each feature. For instance, we can see that ordinal values have higher

[14]https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.describe.html



	ps_reg_01	ps_reg_02	ps_reg_03	ps_car_12	ps_car_13	ps_car_14	ps_car_15
count	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000
mean	0.610991	0.439184	0.551102	0.379945	0.813265	0.276256	3.065899
std	0.287643	0.404264	0.793506	0.058327	0.224588	0.357154	0.731366
min	0.000000	0.000000	-1.000000	-1.000000	0.250619	-1.000000	0.000000
25%	0.400000	0.200000	0.525000	0.316228	0.670867	0.333167	2.828427
50%	0.700000	0.300000	0.720677	0.374166	0.765811	0.368782	3.316625
75%	0.900000	0.600000	1.000000	0.400000	0.906190	0.396485	3.605551
max	0.900000	1.800000	4.037945	1.264911	3.720626	0.636396	3.741657

Figure 20: Statistical analysis of continuous features



	ps_ind_01	ps_ind_03	ps_ind_14	ps_ind_15	ps_car_11	ps_calc_04	ps_calc_05
count	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000	595212.000000
mean	1.900378	4.423318	0.012451	7.299922	2.346072	2.372081	1.885886
std	1.983789	2.699902	0.127545	3.546042	0.832548	1.117219	1.134927
min	0.000000	0.000000	0.000000	0.000000	-1.000000	0.000000	0.000000
25%	0.000000	2.000000	0.000000	5.000000	2.000000	2.000000	1.000000
50%	1.000000	4.000000	0.000000	7.000000	3.000000	2.000000	2.000000
75%	3.000000	6.000000	0.000000	10.000000	3.000000	3.000000	3.000000
max	7.000000	11.000000	4.000000	13.000000	3.000000	5.000000	6.000000

Figure 21: Statistical analysis of ordinal features

ranges compared to the continuous features.

Visualization
Visualization is another important step in order to get familiar with the data, as for example histograms show the frequency of each value. Furthermore, we used box-whisker-plots to identify the median and possible outliers of each feature. We analyzed all feature groups in the same way, however, to ensure readability we included only the categorical features. The other plots can be found in Appendix A.

In Figure 22 we can see that there are values in each feature that occur more often than the others. The histograms also show us that *ps_ car_ 03 _ cat* and *ps_ car_ 05_ cat* have more missing values than any other value. These two features require a further analysis - if they do not influence the target variable, they could be discarded due to the fact that more as the half of the information is missing. Some of the other features have missing values as well but the amount of them is not significantly high. Furthermore, *ps_ car_ 10 _ cat* stands out, as category one seems to be present in almost all data samples. Similarly, a comparison of this feature to the target variable could give us more information about the importance of it.

Figure 23 shows us the first and third quartile, the median and identified

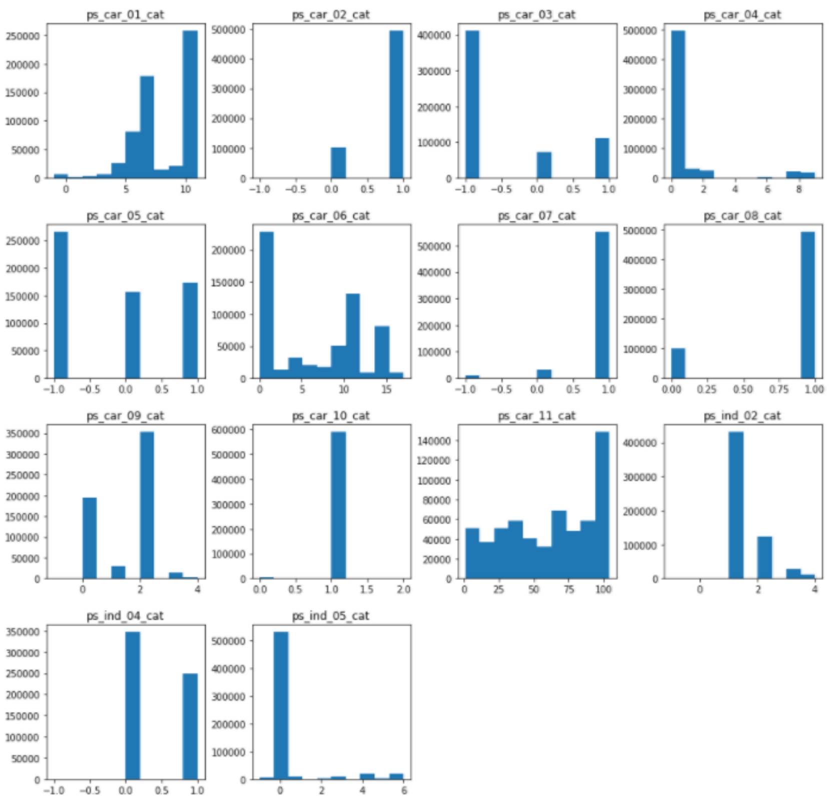


Figure 22: Histograms of categorical features

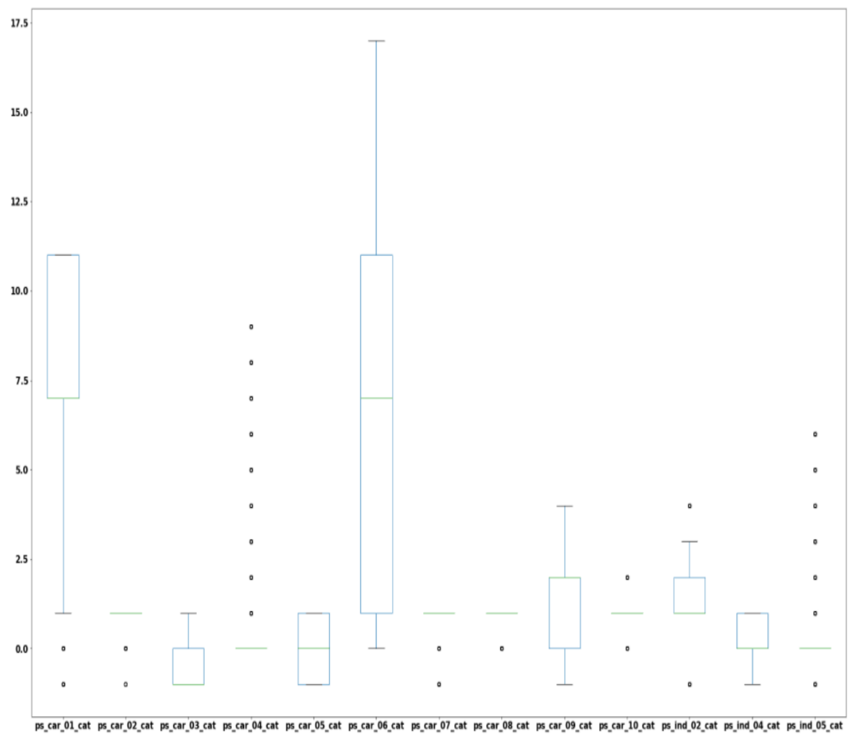


Figure 23: Boxplot of categorical features

outliers. Note that the middle line is the median, the line below is the first quartile, the line above is the third quartile and the top short line is the fourth quartile. The features which have a lot of missing values use -1 as either the median or first quartile, whereas the other features treat them as outliers. Furthermore, we can see that the value frequency of some of the features is more monotonous than others. The features which are plotted only by the median, such as *ps_ car_ 02_ cat* have the same values in the first, second, third and fourth quartile.

To analyze the influence of each feature on the target variable, we used histograms to identify values that are more likely to effect the output variable. For comprehensibility reasons, this part includes the results of all features which are shown from Figure 24 through Figure 27.

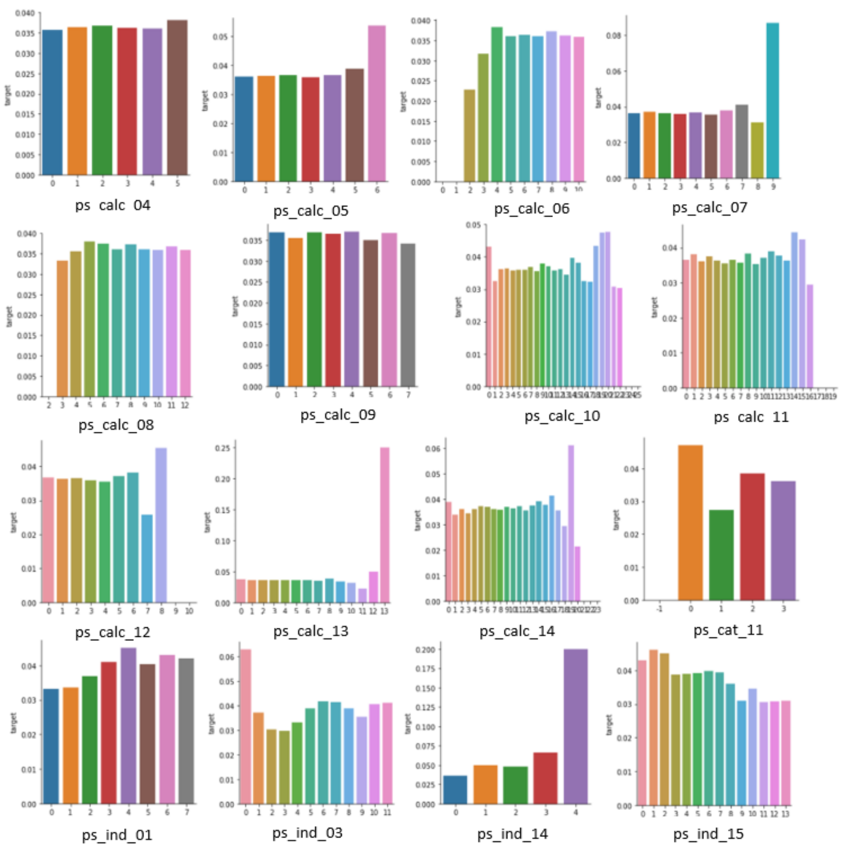


Figure 24: Influence of ordinal features on the target variable

Figure 24 shows all ordinal features that are evaluated based on their influence on the target variable. We can see that the feature *ps_ calc_ 04* has almost no influence on the target variable as the individual values of the feature are almost equal. The same can be said for *ps_ calc_ 08* and *ps_ calc_ 09*. Furthermore, the features *ps_ calc_ 07*, *ps_ calc_ 13*, *ps_ calc_ 14*, *ps_ ind_ 03* and *ps_ ind_ 14* appear to have strong influential values as the fraction to the target variable is comparably high.

Looking at the categorical features in Figure 25 we can see that the feature *ps_ car_ 01_ cat* has a very high impact on the target variable if the value is missing. This can be concluded from the peak at -1. The same can be observed for almost all other categorical features with missing values namely: *ps_ car_ 07_ cat*, *ps_ car_ 09_ cat*, *ps_ ind_ 02_ cat*, *ps_ ind_ 04_ cat* and *ps_ ind_ 0 5_ cat*. This led us to believe that if those five features are not recorded, the customer is more likely to file a claim. To further analyze the six features, we compared them to their frequency of occurrence which we can see in the histograms of Figure 22. All of the six features have a very low amount of missing values, yet they are of very high importance to the model, which is why we did not replace them and kept them as -1s. Furthermore, the feature *ps_ car_ 10_ cat*, which almost only had the value 1 in Figure 22, seems not to have any impact on the target as all possible values are almost equally influencing the outcome. This can be seen on the similar size of the bars.

Looking at the distributions of the binary features in Figure 26 we can see that a lot of them seem not to be relevant to the outcome of the target variable as most of them have equally sized bars. Especially all binary calculation features are almost equally distributed and hence do not have a strong influence whether a customer filed a claim. On the contrary, *ps_ ind_ 13_ bin* and *ps_ ind_ 17_ bin* seem to influence the outcome as the value 1 has a higher impact on the outcome.

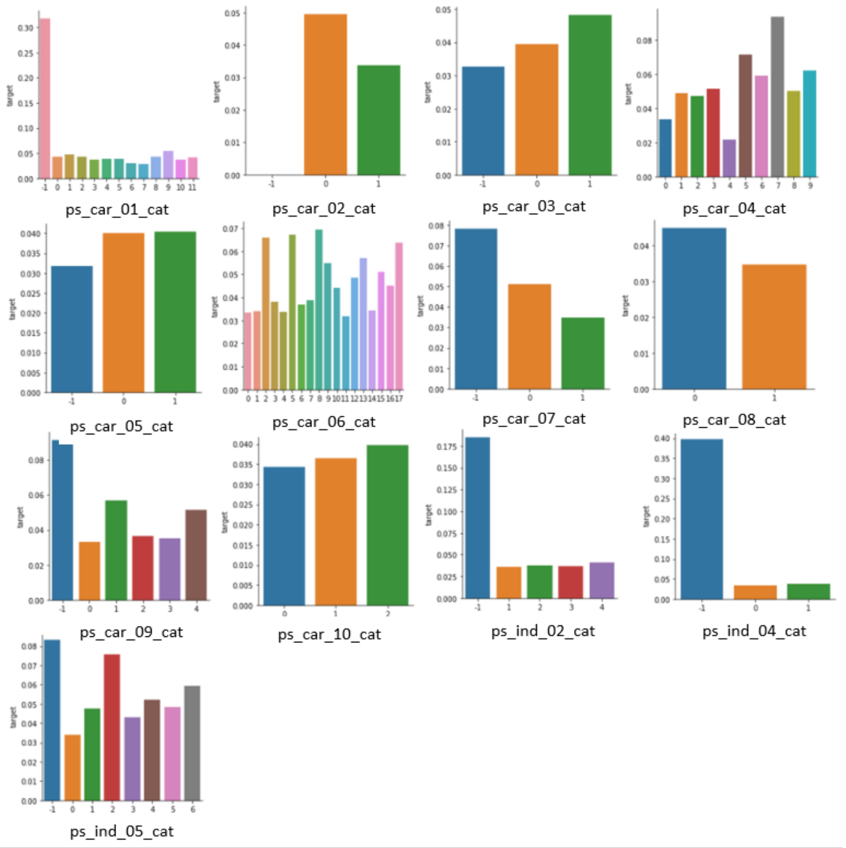


Figure 25: Influence of categorical features on the target variable

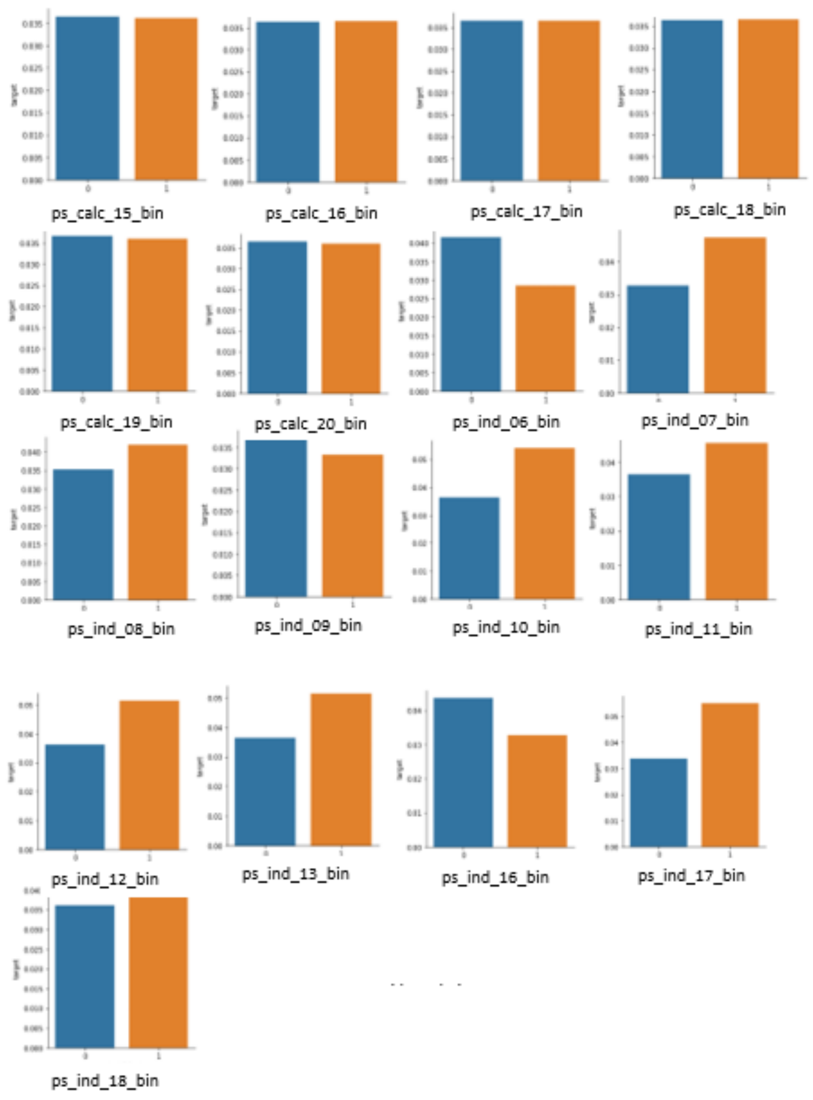


Figure 26: Influence of binary features on the target variable

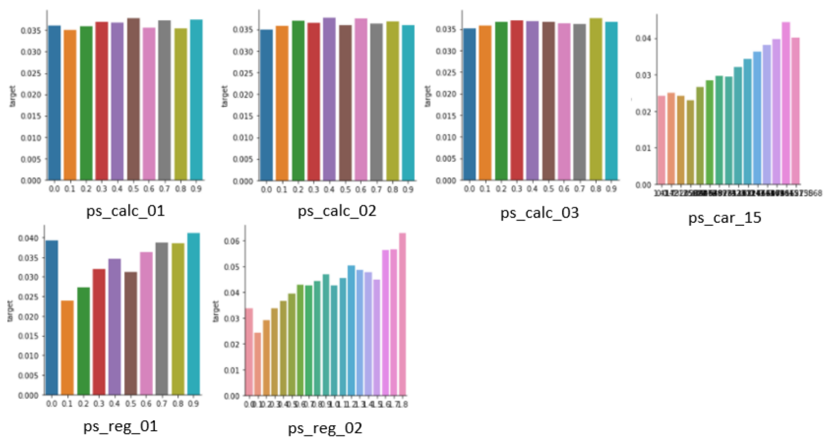


Figure 27: Influence of continuous features on the target variable

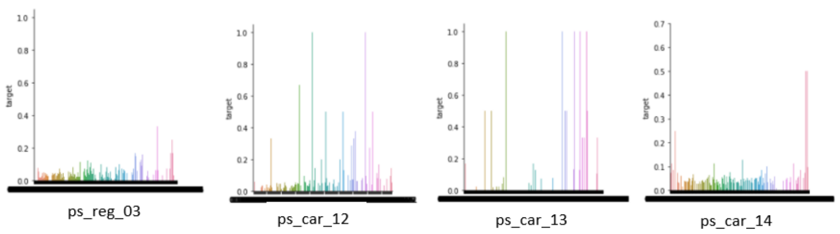


Figure 28: Influence of continuous features on the target variable

As the continuous features include high ranges, we separated the analysis into one group with features that could be analyzed easily and another one for those that are more difficult to analyze. Figure 27 shows the analysis of those features which have a manageable amount of values, whereas Figure 28 shows the features which require a more detailed analysis. Looking at Figure 27 we see that especially the calculation features seem not to have any impact on the outcome, as they are quite equally distributed. Customers that have higher values of the features *ps_ car_ 15* and *ps_ reg_ 02* were more likely to claim than those with lower values. The listing below explains the conclusions derived from a more detailed analysis of the features from Figure 28.

- *Ps_ reg_ 03*:
  78% of the customers filed a claim if the feature was above the mean. 46% of the customers filed a claim if the values fell above the mean but below the third quartile.

- *Ps_ car_ 12*:
  55% of the customers filed a claim if the value was above the mean. However, only 5% of the customers filed a claim if the value was between the mean and the third quartile, which leads us to the conclusion that high values are more prone for claiming.

- *Ps_ car_ 13*:
  50% of the customers filed a claim if the values were above the mean and 16% were positive if the value was between the mean and the upper quartile.

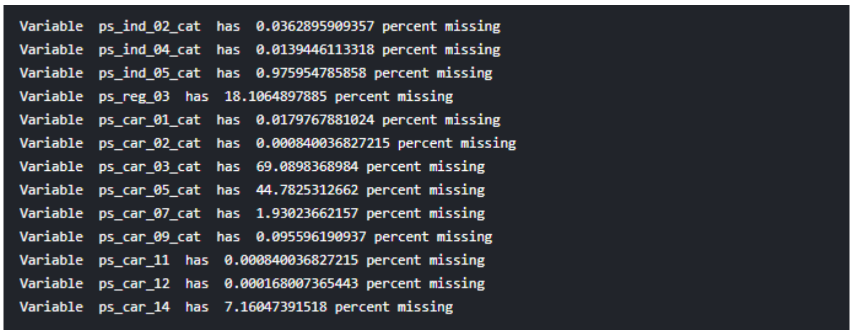
- *Ps_ car_ 14*:
  91% of the customers claimed if the feature had a value above the mean. Even if we limit the range and look at the distribution between the mean and the upper quartile, there were still 62% of the customers who filed a claim.

5.2.2 Act

After the analysis and interpretation of the features, it was important to conduct some data-preprocessing steps before creating the model. This step was important so that comprehensibility could be increased and dimensionality decreased. Furthermore, it allowed us to speed up the learning process as a feature subset was selected rather than the initial feature space.

In the previous chapter, we heard that some features have missing values. Figure 29 shows the percentage of the missing values per feature. The code for this analysis was adopted from Bert Carreman's [15] notebook on Kaggle. Furthermore, Table 11 summarizes our approaches of dealing with the missing values.

As we can see from Table 11 almost all categorical features that include missing values have a high impact on the target variable. Therefore, we did not replace them and kept the -1s. The two categorical features (*ps_ car_ 05_ cat* and *ps_ car_ 03_ cat*), that have a very high amount of missing values, do not influence the outcome variable. Therefore, we dropped them from our dataset. The other missing values were replaced by the mean or the mode,



```
Variable  ps_ind_02_cat  has  0.0362895909357 percent missing
Variable  ps_ind_04_cat  has  0.0139446113318 percent missing
Variable  ps_ind_05_cat  has  0.975954785858 percent missing
Variable  ps_reg_03  has  18.1064897885 percent missing
Variable  ps_car_01_cat  has  0.0179767881024 percent missing
Variable  ps_car_02_cat  has  0.000840036827215 percent missing
Variable  ps_car_03_cat  has  69.0898368984 percent missing
Variable  ps_car_05_cat  has  44.7825312662 percent missing
Variable  ps_car_07_cat  has  1.93023662157 percent missing
Variable  ps_car_09_cat  has  0.095596190937 percent missing
Variable  ps_car_11  has  0.000840036827215 percent missing
Variable  ps_car_12  has  0.000168007365443 percent missing
Variable  ps_car_14  has  7.16047391518 percent missing
```

Figure 29: Summary of percentage of missing values within each feature

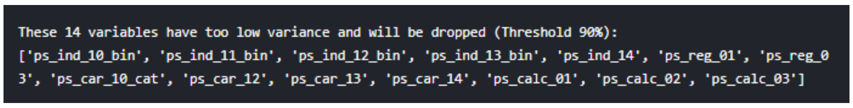
depending on whether the feature was ordinal or continuous. The sklearn library provides the function `Imputer()`[15], which allowed us to specify the type of replacement and the value that indicates a missing instance. Applied to our dataset we needed to specify the missing values as "-1" and the imputer strategy to "mean" or "most_frequent", depending on whether the feature was continuous or ordinal.

Recall that a reduced feature space can lead to a more accurate prediction, which is why data selection is an important step. Therefore, we needed to look at the feature importance and identify those features that were not valuable to the model. As we wanted to compare different approaches with each other, we decided to focus on a filter approach for feature selection first. The sklearn library provides the function `VarianceThreshold()`[16] that discards features that have low variance, as it is assumed that those values are more constant and have low predictive power. Using a variance threshold is one way of performing a filter approach [58]. We used the function to identify the features with too low variance and set the variance threshold to 90%. This means that if the same value is in 90% of the samples the feature is dropped. Before agreeing on the 90% we worked with different variance threshold values to identify differences in the results. The default threshold is 0.0, however, our dataset did not include any feature that had the same value in all the samples. When we set the threshold to 95% it only returned three features. A threshold of 85% returned twenty features. However, we wanted to select a feature subset that was smaller than the initial feature space, but at the same time it should keep enough features. Therefore, a

[15]http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Imputer.html
[16]http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html

Feature	Influence on target	Action
ps_ind_02_cat	high	keep missing values
ps_ind_04_cat	high	keep missing values
ps_ind_05_cat	high	keep missing values
ps_reg_03	low	replace by mean
ps_car_01_cat	high	keep missing values
ps_car_01_cat	medium	keep missing values
ps_car_03_cat	low	drop - due to high percent of missing values
ps_car_05_cat	low	drop - due to high percent of missing values
ps_car_07_cat	high	keep missing values
ps_car_09_cat	high	keep missing values
ps_car_11	low	replace by mode
ps_car_12	low	replace by mean
ps_car_14	low	replace by mean

Table 11: Summary of missing values and the action of dealing with them



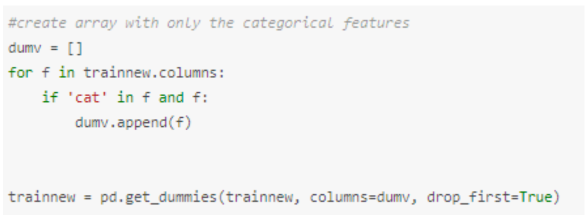
```
These 14 variables have too low variance and will be dropped (Threshold 90%):
['ps_ind_10_bin', 'ps_ind_11_bin', 'ps_ind_12_bin', 'ps_ind_13_bin', 'ps_ind_14', 'ps_reg_01', 'ps_reg_0
3', 'ps_car_10_cat', 'ps_car_12', 'ps_car_13', 'ps_car_14', 'ps_calc_01', 'ps_calc_02', 'ps_calc_03']
```

Figure 30: Features selection based on low variance as filter approach

threshold of 90% seemed reasonable as fourteen variables were identified to have a too low variance and hence were dropped from further analysis. Figure 30 shows which features were dropped. We continued our analysis with a total of forty-one features, excluding the *target* and *id*. Another important step in the data-preparation is the transformation of features that we introduced in Section 3.1.2. As our dataset included many categorical features it was important to create dummy variables for each of the possible values. The pandas library provides the function `get_dummies()`[17] which easily produces the additional variables. Figure 31 shows the code that is used to create these variables. First we created an array to save all categorical features that were in our dataset and then we applied the function.

After transformation, we received a total of one hundred ninety-four fea-

[17]https://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html



```
#create array with only the categorical features
dumv = []
for f in trainnew.columns:
    if 'cat' in f and f:
        dumv.append(f)


trainnew = pd.get_dummies(trainnew, columns=dumv, drop_first=True)
```

Figure 31: Code to create dummy variables for all categorical features

tures because *ps_ car_ 11 _cat* had one hundred four categories in total. Furthermore, we dropped the ID from the training data as it did not give us any insights. The target variable needed to be assigned to an individual variable that is not part of the training set. This allowed us to treat the remaining variables as predictors and the target variable as response which was necessary for training the model. For more details on the target and response variables recall Section 3.2.1. Table 12 summarizes the change history of various approaches that were conducted in Iteration 1. All five runs include the feature subset that was identified using the variance threshold.

In Section 4.6 we introduced cross-validation as a re-sampling method that is quite often used in machine learning. In the first run we split our training data into a train and validation set using a 70:30 ratio, where 70% of the data falls into the training and 30% into the validation test set. In the other runs we changed our approach to a 5-fold cross validation as it is claimed to achieve better performance. Furthermore, as an initial evaluation criterion we used the out-of-the bag (OOB) score to estimate the performance of the random forest. The random forest classifier[18] in sklearn provides the parameter `oob_score` which can be used to evaluate the score of the random forest. Recall that a random forest consists of several decision trees that use random subsets of the initial training data. The samples that are not used for each subset are validated against the results.

From Table 12 we can see that the initial approach using random forest with one hundred trees and a maximum depth gives us an OOB score of 97%. As we assumed that the score was too high, given the fact that almost no changes of hyper parameters were conducted, we chose a 5-fold cross validation in the second run to see the difference and achieved a score of 99.8% which led us to believe that our random forest was overfitting.

[18]http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

	CV	Method	Parameters	OOB_score
1.	70:30	RF	100 trees, max_depth=5	97%
2.	5-fold	RF	100 trees, max_depth=5	**99.8%**
3.	5-fold	RF	100 trees, max_depth=5, max_features=0.3	97%
4.	5-fold	RF	100 trees, delete max_depth	94%
5.	5-fold	KNN	default	93%

Table 12: Change history of Iteration 1

From the literature review we know that a random forest has the advantage that it is using random features for each decision tree, however, if we do not specify the number of maximum features upfront, it uses (by default) the square root of the number of features. In our case, there were one hundred ninety-four features and taking the square root results in approximately thirteen features. To increase the possible features we, therefore, set the parameter for maximal features to 0.3 and achieved an OOB score of 97%.

To be sure about the assumption that the model is overfitting, we deleted the maximal depth parameter such that the tree could grow as deep as possible. Consequently, we expected that the performance would increase as the subsets after each split were more homogeneous. The score decreased to 94% as can be seen in Table 12.

We decided to create another model using another classifier to compare the results. We suspected that if the new method reaches a similarly good performance then there needs to be another reason why the OOB score is so high. We wanted to use a learning method that usually performs well, however, it should not outperform a random forest classifier. The K-nearest neighbor classifier seemed to be a good other option for comparison as it is a simpler algorithm and uses the Euclidean distances to classify the observations. Hence, our assumption was that the method will give us worse results than the random forest. After applying the KNN to the dataset we achieved a score of 93%. Although the score is lower compared to the random forest it is still too high given that no changes on the hyper parameters were conducted. Looking at the confusion matrix of the KNN classifier, we recognized that the amount of true positives, which are values that are correctly predicted as 0, was very high and that no true negatives were identified. In the literature review we came across imbalanced datasets which could be the root of the problem. By analyzing the ratio between customers who claimed a file and those who did not, we recognized that only approximately 3% of the dataset included customers who claimed. This means, that even if the model predicted that all customers will not file a claim, the OOB score would still be around 97%.

5.3 Iteration 2

In Iteration 1 we conducted several classification approaches using random forest and K-nearest neighbor. We initially thought that the models were overfitting due to a high OOB score. However, after analyzing the distribution of the dataset we identified that the root cause of it was due to an imbalanced dataset.

5.3.1 Look & Think

Imbalanced datasets require a different way of handling compared to balanced datasets. Several possibilities exist to make prediction more valuable. One way is to use another performance metric, such as precision, recall or f-score [13]. Furthermore, we can resample our dataset, either *oversample* or *undersample* it. In the case of oversampling, we create more instances of the class that is too small. On the contrary, undersampling refers to working with a reduced sample set of the majority class [13].

5.3.2 Act

As a first approach we were oversampling our dataset using a well-known method called *SMOTE*, which is the abbreviation for Synthetic Minority Oversampling TEchnique [18]. SMOTE looks at the examples from the minority class and ignores those from the majority class. It looks at one instance, takes its k nearest neighbors and creates new instances halfway between the k selected neighbors and the initial instance. This is repeated for all the instances from the minority class. However, the drawback of SMOTE is that the inserted instances will always remain in the same area and cannot be inserted across the dataset [23].

We used the package of the imbalanced-learn API[19] which provides a function that performs the SMOTE method. Through the function in Listing 2

[19]http://contrib.scikit-learn.org/imbalanced-learn/stable/generated/imblearn.over_sampling.SMOTE.html#examples-using-imblearn-over-sampling-smote

	Feature set	CV	Method	Parameters	AUC
1.	Variance Threshold	70:30	RF	500 trees, default	**0.50**
2.	all 58 features	70:30	RF	500 trees, default	**0.50**

Table 13: Change history of Iteration 2

```
X_resampled, y_resampled = SMOTE().fit_sample(x_train, y_train)
```

Listing 2: Oversampling training data

we could assign the predictors and the label from the training set to a re-sampled set of predictors and labels. Then we run a random forest, initially using default parameters, with five hundred trees and fit it on the re-sampled predictors and labels from the training dataset. We predicted the outcomes using the `predict()`[20] function from sklearn and plotted the confusion matrix to analyze the predictions versus its actual results. Unfortunately, our model did not perform well even though the instances of the minority class were increased. If we look at the confusion matrix we can see that the random forest did not predict any true negatives at all, which means that no customer has been identified as filing a claim. As an additional metric to the confusion matrix we used the AUC and plotted the ROC curve that visualizes the overall performance. The code for plotting the ROC curve was adopted from a blogpost [52]. The ROC curve with the corresponding AUC value of the current model can be seen in Figure 32. The AUC is 0.5 as there is only the diagonal in the graph. Recall that an AUC value of 0.5 does not have much predictive power and can be compared to random guessing.

To deal with the low predictive power of our classifier, we decided to look for similar datasets that also used SMOTE as a oversampling method and compare the approaches. We found another Kaggle competition[21] that looked at fraud detection and used random forest for an imbalanced dataset. They achieved very good results using random forest without changing any parameters. As their random forest achieves good results without doing any pre-processing steps, we decided to try the same and compare the results that we got using a random forest with pre-processing steps and without. Keep in mind that the results from a random forest with pre-processing steps were already presented and can be retrieved from Table 12 and 13 (step 1). By

[20]http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[21]https://www.kaggle.com/chtaret/fraud-detection-with-smote-and-randomforest/notebook

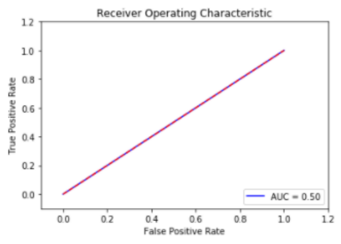


Figure 32: ROC curve of oversampled dataset using Random Forest

keeping the initial fifty-eight features, we received a slightly better AUC then the random forest from the first run. Nevertheless, the AUC was still approximately 0.5 which means that the classifier still performed poorly. As the difference of the AUC values of the two classifiers is small we do not include the ROC curve here.

5.4 Iteration 3

In Iteration 2 we used the initial dataset without any pre-processing in order to compare the results when pre-processing steps are conducted. Using all initial fifty-eight features did not give us any significant improvement.

5.4.1 Look & Think

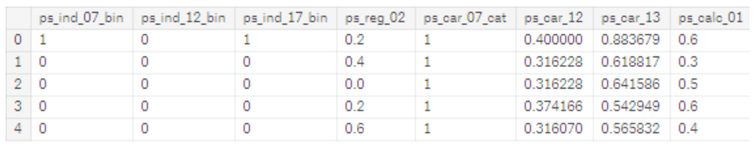
To deal with the low classification performance we needed to change our approach and consider other possibilities of performing the classification task. Although random forest often performs well, there are boosting methods that are known to achieve a better performance. Caruana and Niculescu-Mizil [16] used ten different classification methods on eleven datasets and compared their performance to each other. In brief, they came to the conclusion that boosting and random forest achieved the best results. Given the fact that boosting puts weight on wrongly classified instances, we conjectured that a boosting algorithm, such as AdaBoost, would achieve a better performance than random forest.

5.4.2 Act

As none of our previous approaches led to good results, we decided to change some settings of the random forest and also run AdaBoost on the data. We assumed that the feature subset was the main problem for the poor performance. Therefore, to identify the differences of feature selection, we used

	Feature Selection	CV	model	AUC	Precision	Recall	F-score
1.	WEKA Forward	70:30	RF	0.50	0.03	0.10	0.05
			AdaB	**0.55**	**0.05**	**0.27**	**0.09**
2.	WEKA Backward	70:30	RF	0.50	0.04	0.04	0.04
			AdaB	0.53	**0.05**	0.21	0.08

Table 14: Change history of Iteration 3



	ps_ind_07_bin	ps_ind_12_bin	ps_ind_17_bin	ps_reg_02	ps_car_07_cat	ps_car_12	ps_car_13	ps_calc_01
0	1	0	1	0.2	1	0.400000	0.883679	0.6
1	0	0	0	0.4	1	0.316228	0.618817	0.3
2	0	0	0	0.0	1	0.316228	0.641586	0.5
3	0	0	0	0.2	1	0.374166	0.542949	0.6
4	0	0	0	0.6	1	0.316070	0.565832	0.4

Figure 33: Feature subset using forward selection in Weka

Weka datamining tool to select a feature subset. Weka can analyze different approaches in feature selection depending on the dataset. For our dataset it suggested a wrapper method. In the first run we used forward selection to identify the best features that were highly correlated with the target variable and at same time did not correlate between each other. After running five hundred eighty-nine different subsets, Weka identified that the best one consists of the eight features seen in Figure 33. We can see that the subset included three features that belong to the drivers' group, one regional feature, three car relevant features and only one calculated feature. Nevertheless, all four feature categories were present.

In Table 14 we can see that the performance increased, using the features identified by Weka. Both methods ran on the oversampled data using the SMOTE method. The random forest had five hundred trees and used default parameters, whereas the AdaBoost only used the default parameters with fifty trees. AdaBoost achieved an AUC of 0.55 compared to the random forest of the previous iteration. Furthermore, we decided to use precision and recall as further evaluation metrics which allowed us to compare the results more in detail. Random forest achieved a recall of 0.10 and AdaBoost 0.27 which means that in the case of AdaBoost 27% of all customers were identified as placing a claim. For the further analysis we focused more on improving the recall rather than the precision, as we assumed that an insurance company is more interested in identifying more possibly claiming customers than identifying fewer but more accurate ones.

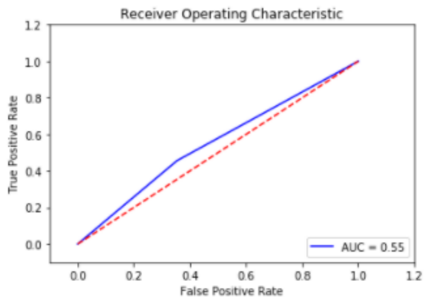
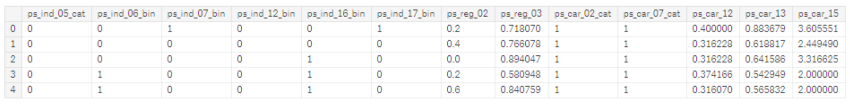


Figure 34: ROC curve of oversampled dataset using AdaBoost with 12 trees



	ps_ind_05_cat	ps_ind_06_bin	ps_ind_07_bin	ps_ind_12_bin	ps_ind_16_bin	ps_ind_17_bin	ps_reg_02	ps_reg_03	ps_car_02_cat	ps_car_07_cat	ps_car_12	ps_car_13	ps_car_15
0	0	0	1	0	0	1	0.2	0.718070	1	1	0.400000	0.883679	3.605551
1	0	0	0	0	0	0	0.4	0.766078	1	1	0.316228	0.618817	2.449490
2	0	0	0	0	1	0	0.0	0.894047	1	1	0.316228	0.641586	3.316625
3	0	1	0	0	1	0	0.2	0.580948	1	1	0.374166	0.542949	2.000000
4	0	1	0	0	1	0	0.6	0.840759	1	1	0.316070	0.565832	2.000000

Figure 35: Feature subset using backward selection in Weka

We improved our results in the previous run through changing the feature space. Therefore, we were interested how the results would change if yet another feature subset was used. The Weka Datamining tool provides a backward selection search strategy which evaluates the best subsets of features starting with all possible ones. Weka analyzed one thousand six hundred eighty different subsets and received the best score with thirteen features that are shown in Figure 35. By looking at the feature names, we can see that six features were identified as important that relate to the drivers group. Five features about the car were relevant and only two features about the region were important. Backward selection identified more features and analyzed more different subsets than forward selection. Another interesting aspect is that no calculated feature was identified as relevant in the subset.

We performed the same random forest and AdaBoost as in the first run, so that we could compare the results using the two search strategies. From Table 14 we can see that the backward selected feature subset leads to worse results than the forward selected set for both classifiers. Using this feature subset we predict less potential claiming customers as the recall decreased for both random forest and AdaBoost. Therefore, we decided to use the feature set of the forward selection for further analysis as it gave us the best results so far.

```
param_grid = {"n_estimators": [8, 10, 18, 22],
              "max_depth": [3, 5, 7],
              "min_samples_split": [10, 15, 20],
              "min_samples_leaf": [3, 5, 10, 20],
              "max_leaf_nodes": [10, 20, 40],
              "min_weight_fraction_leaf": [0.1]}
```

Listing 3: Tuning Parameters

5.5 Iteration 4

In Iteration 3, AdaBoost performed slightly better than random forest using the forward selected feature subset. However, as we want to increase the recall, this iteration focuses on tuning the parameters for the random forest and AdaBoost.

5.5.1 Look & Think

In the previous iteration we did not tweak our models at all. Therefore, to optimize the models, we used the knowledge gained from the literature review to analyze the current parameters and identify possible improvements. Sklearn provides a module called `gridsearchCV`[22] that is used to tune the hyper parameters, which are parameters that are passed as arguments instead of being used automatically with the classifier. Therefore, to skip the repetitive process of changing parameters, we applied this method to analyze the results of random forests using different parameters. We found a Kaggle notebook[23] that focuses on tuning random forest using grid search. As it is similar to our approach we used some of the input provided by it for our analysis.

5.5.2 Act

As gridsearchCV provides us with the possibility to try out several parameters at once without the need to repeatedly train the model, we used it for further analysis to tune the random forest. Listing 3 shows the possible parameters for the random forest that the method analyzed.

Applying gridsearchCV to our AdaBoost model requires a slightly different approach, as AdaBoost does not have many hyper parameters to change. However, it uses the parameter `base_estimator` which is the initial estimator from which the ensemble is built. By default, this parameter is a decision

[22]http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
[23]https://www.kaggle.com/hadend/tuning-random-forest-parameters

	Model	Parameters	AUC	Precision	Recall	F-score
1.	RF	8 trees,max_depth=3, max_leaf_nodes= 10, min_samples_leaf=3, min_samples_split=10, min_weight_fraction_leaf= 0.1	**0.57**	0.05	**0.59**	**0.09**
2.	RF	100 trees, remaining in row 1	0.56	0.05	0.55	0.08
3.	RF	200 trees, remaining as in row 1	0.56	0.05	0.55	0.08
4.	AdaB	10 trees, base_estimator= (criterion='gini', base_estimator_splitter='best')	0.51	**0.06**	0.07	0.06
5.	AdaB	100 trees, remaining as in row 4	0.50	0.04	0.11	0.05
6.	AdaB	200 trees, remaining as in row 4	0.50	0.04	0.11	0.05
7.	AdaB	50 trees, base_estimator=(decision tree with max_splits=1)	0.54	**0.06**	0.24	**0.09**
8.	AdaB	100 trees, remaining as in row 7	0.54	0.05	0.21	**0.09**
9.	AdaB	200 trees, remaining as in row 7	0.53	0.05	0.16	0.08
10.	AdaB	5 trees, base_estimator=RF from row 1	0.56	0.05	0.41	**0.09**

Table 15: Change history of Iteration 4

```
param_grid = {"base_estimator__criterion" : ["gini", "entropy"],
              "base_estimator__splitter" :   ["best", "random"],
              "n_estimators": [3,5,10]
             }
```

Listing 4: Tuning Parameters for decision tree

tree which selects weak learners and creates strong learners through combining them. This means that in order to perform gridsearchCV we needed to analyze the parameters used for the decision tree. The tuning parameters of the decision tree can be seen in Listing 4.

Table 15 shows the best evaluated tuning parameters and the corresponding results for both random forest and AdaBoost. From row 1 and 4, we can see that the tuned random forest performed much better than the tuned AdaBoost. Furthermore, training the data on the tuned AdaBoost algorithm achieved worse results than the AdaBoost model with default parameters from Iteration 3. Figure 36 and 37 show the plotted ROC of the tuned random forest and the AdaBoost with the tuned decision tree from Table 15 (row 1 and 4).

We further analyzed different variations of the two models by increasing the number of trees. As can be seen in Table 15 (row 1 to 3) random forest performed best with the initial eight trees, as one hundred and two hundred trees have a slightly worse performance. Eight trees achieved an AUC of 0.57 and a recall of 0.59. One hundred and two hundred trees both achieved an AUC of 0.56 and a recall of 0.55. Similarly, looking at Table 15 (row 4 to 6) AdaBoost showed a decreased performance in AUC the more trees were added. Ten trees achieved an AUC of 0.51 and the AUC for one hundred or two hundred trees remained the same at 0.50. However, looking at the recall we can see that more trees slightly improved it. Ten trees had a recall of 0.07, whereas one hundred and two hundred trees achieved a recall of 0.11.

As Caruana and Niculescu-Mizil [16] achieved the best results with a boosting algorithm that used decision stumps, we decided to try the same. Decision stumps are decision trees that only use one split. Table 15 (row 7 to 9) shows the results that we achieved using AdaBoost with decision stumps. The best result amongst the three variations is the AdaBoost with fifty trees as the AUC value was 0.54 and the recall was 0.24. Increasing the number of trees to one hundred gave us an AUC value of 0.54 and and recall of 0.16. Two hundred trees decrease the AUC value to 0.53 and the recall to 0.16. Looking at the results we see that the more trees were added the worse AdaBoost performed. However, AdaBoost using decision stumps performed better than AdaBoost using the tuned parameters for the decision tree as an estimator.

As a last approach we used our best performing model from the first run (see row 1 from Table 15) as a base estimator for AdaBoost. The AUC value was 0.56 and the recall was 0.41 which can be seen in the last row of Table 15. From all possible variations used of the AdaBoost algorithm the last run was the best as both AUC and recall were the highest with 0.56 and 0.41 respectively. In Iteration 3 AdaBoost achieved an AUC of 0.55 and a recall of 0.27. Nevertheless, the tuned random forest with eight trees from the first row of Table 15 achieved the best results, with an AUC of 0.57 and a recall of 0.59, so far. In this iteration we achieved a better performance of both methods, therefore we continue our analysis with the parameters that achieved the best results.

5.6 Iteration 5

Although, the performance of the random forest was slightly improved, it does still not have much predictive power. Therefore, further investigation is needed.

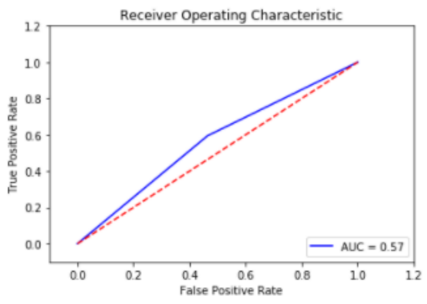


Figure 36: ROC curve of tuned Random Forest

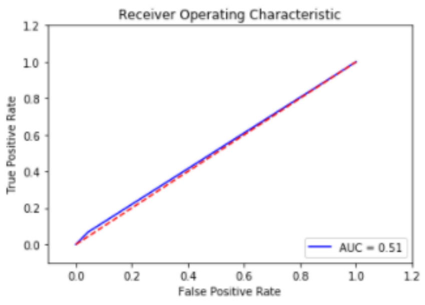


Figure 37: ROC curve of AdaBoost using tuned decision tree

	Feature Selection	Reason	model	AUC	Precision	Recall	F-score
1.	All except of calc	no correlation with target	RF	0.50	0.04	0.11	0.06
			AdaB	0.50	0.00	0.0	0.00
2.	all except the two with too high missing value	trial	RF	0.54	**0.05**	0.27	**0.08**
			AdaB	0.50	0.0	0.0	0.0
3.	Features without noise	to increase predictive power	RF	**0.55**	0.04	**0.50**	**0.08**
			AdaB	0.54	**0.05**	0.31	**0.08**

Table 16: Change history of Iteration 5

5.6.1 Look & Think

We identified that our random forest performed better by changing the hyper parameters. In this iteration we want to analyze whether we can further increase the recall by changing the feature space. During the literature review we identified that features play a major role in the success of a learning model. Therefore, we want to analyze various feature subspaces and the influence they have on the performance.

5.6.2 Act

In this iteration we split our training data using a cross validation of a 70:30 ratio for a train and validation test set. Furthermore, we kept the best parameters from Iteration 4. As can be seen in Table 16, in the first run the calculated features were deleted, as they have (almost) no correlation to the target variable. However, this subset did not improve the performance as the random forest achieved an AUC of 0.5 and a recall of 0.11. Similarly, AdaBoost achieved an AUC of 0.5 and did not predict any customer as claiming which is indicated by the zero values for precision and recall. In the second run we used all features except the two features that had too many missing values (more than 50%). Looking at Table 16 we can see that random forest improved its performance of this iteration as both AUC and recall increased to 0.54 and 0.27 respectively. On the contrary, we can see that AdaBoost performed poorly using this feature set as it did not identify any customer as potentially claiming. Lastly, we used the features that did not include any noise according to an analysis conducted by another participant of the competition[24]. This approach achieved the best result in this iteration as the random forest achieved an AUC of 0.55 with a recall of 0.50. The same can be observed for AdaBoost in this iteration as the AUC value is 0.54 and the recall 0.31. Although this feature subset achieved the best scores in this iteration, it still performed worse than in Iteration 4 as the random forest achieved a recall of 0.59.

5.7 Iteration 6

The last iterations showed improvements in the results. However, none of the changes that were conducted so far led to a good result. Therefore, this iteration focuses on undersampling the majority class instead of oversampling the minority class.

5.7.1 Look & Think

As we identified that our model improved by oversampling, we wanted to analyze the difference if we use an undersampling method instead. Tomek Links[25] is a good way to decrease the amount of the instances of the majority class. The method looks at instances that are close in distance to the ones of the majority class. It then removes the majority class leaving the instance of the minority class as it is. This process allows the classification algorithm to better differentiate amongst instances that are close to the majority class and hence can be misclassified easily.

	Model	Parameters	AUC	Precision	Recall	F-score
1.	RF	8 trees,max_depth=3, max leaf nodes= 10, min samples leaf=3, min samples split=10, min weight fraction leaf= 0.1, class weight="balanced"	0.56	**0.05**	0.56	**0.09**
2.	RF	100 trees,max depth=3, max leaf nodes= 10, min samples leaf=3, min samples split=10, min weight fraction leaf= 0.1, class weight="balanced"	0.56	**0.05**	**0.57**	**0.09**
3.	RF	200 trees,max depth=3, max leaf nodes= 10, min samples leaf=3, min samples split=10, min weight fraction leaf= 0.1, class weight="balanced"	0.56	**0.05**	0.55	**0.09**
4.	AdaB	50 trees, default	0.50	0.0	0.0	0.0
5.	AdaB	50 trees, base estimator= RF from run 2	0.57	**0.05**	**0.57**	**0.09**
6.	AdaB	100 trees, base estimator= RF from run 2	0.57	**0.05**	**0.57**	**0.09**
7.	AdaB	200 trees, base estimator= RF from run 2	0.57	**0.05**	0.56	**0.09**
8.	AdaB	100 trees, base estimator= decision stump (max features=3, max depth= 1, class weight="balanced")	0.57	**0.05**	0.53	**0.09**
9.	AdaB	50 trees, base estimator=decision stump (max features=3, max depth= 1, class weight="balanced", max leaf nodes=10)	**0.58**	**0.05**	**0.57**	**0.09**
10.	AdaB	100 trees, base estimator= decision stump (max features=3, max depth= 1, class weight="balanced", max leaf nodes=10)	**0.58**	**0.05**	**0.57**	**0.09**

Table 17: Change history of Iteration 6

5.7.2 Act

Table 17 shows the summary of approaches using Tomek Links as an undersampling technique. For all iterations, we use the features that were identified through WEKA datamining tool using forward selection as this feature set gave us the best results so far and a 70:30 cross validation to decrease computational costs. Furthermore, the variations of the random forest and AdaBoost used in this iteration are similar to the onces used in Iteration 4. Therefore, the table is structured similarly. Looking at Table 17 we can see that random forest achieved an AUC value and a recall of both 0.56 using the parameters that were identified as the best ones through gridsearchCV

[24]https://www.kaggle.com/ogrellier/noise-analysis-of-porto-seguro-s-features

[25]http://contrib.scikit-learn.org/imbalanced-learn/stable/under_sampling.html

from Iteration 4. However, compared to the random forest with the same parameters using SMOTE as an oversampling method we achieved a slightly lower performance as both AUC and recall were higher in Iteration 4. By increasing the number of trees from the initial eight to one hundred, we slightly improve our recall to 0.57, whereas the AUC remains the same with 0.56. Lastly, we used two hundred trees and identified that the AUC value remains at 0.56 whereas the recall slightly decreases to 0.55.

We applied AdaBoost with the default parameters on the undersampled data and surprisingly received bad results as can be seen in Table 17 (row 4). No customer was identified as placing a claim, which means that every prediction was "not claiming". We received an AUC value of 0.5 and both recall and precision were 0.0. To see the difference in the results we changed our parameters and set the base estimator to the random forest from the second run (see Table 17 row 2) , as this was the model that achieved the highest recall in this iteration so far. By changing the base estimator we improved our AUC value and recall both to 0.57. Thus using the undersampled dataset, we can conclude that AdaBoost performed better using the best performing random forest as base estimator. In Iteration 4 we did the same with oversampled data, however, random forest performed better than AdaBoost that used the random forest as base estimator. Lastly, we used AdaBoost with two hundred trees. However, it did not give us any improvement as the AUC remained at 0.56 and the recall slightly decreased to 0.56.

As a last approach we wanted to analyze AdaBoost's performance using decision stumps. As we assumed that Tomek Links did not provide us with a totally balanced dataset, we therefore, used an additional parameter `class_weight`[26] that was not used in the previous iterations. When specified as "balanced" the decision tree additionally puts weight on the minority class. In Table 17 we can see that AdaBoost received the best results in this iteration. AdaBoost with one hundred trees and a maximal depth of one achieved an AUC value of 0.57 and a recall of 0.53 (see Table 17 row 8). AdaBoost with fifty trees achieved an AUC of 0.58 and a recall of 0.57. Increasing the number of trees to one hundred did not lead to a improved results as both AUC and recall remain the same.

Undersampling created a similar result than oversampling. However, compared to the computational costs and hence the execution time of the resampling method, we can say that oversampling using SMOTE was much faster.

[26]http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Technique	Feature Set	Model	AUC	Recall
Iteration 1: Change of parameters	Variance Threshold	RF (100 trees)	-	-
		KNN	-	-
Iteration 2: Oversampling	Variance Threshold	RF (500 trees)	0.50	-
	all 58 features	RF (500 trees)	0.50	-
Iteration 3: Change of feature space	Weka Forward	RF	0.50	0.10
		AdaB	0.55	0.27
	Weka Backward	RF	0.50	0.40
		AdaB	0.53	0.21
Iteration 4: Change of parameters	Weka Forward	RF (8 trees)	0.57	**0.59**
		RF (100 trees)	0.56	0.55
		RF(200 trees)	0.56	0.55
		AdaB (10 trees)	0.51	0.07
		AdaB (100 trees)	0.50	0.11
		AdaB (200 trees)	0.50	0.11
		AdaB (Decision stump, 50 trees)	0.54	0.24
		AdaB (Decision stump, 100 trees)	0.54	0.21
		AdaB (Decision stump, 200 trees)	0.53	0.16
		AdaB (RF, 5 trees)	0.56	0.41
Iteration 5: Change of feature space	All except of calc	RF	0.50	0.11
		AdaB	0.50	0.0
	All except two with too high missing val.	RF	0.54	0.77
		AdaB	0.50	0.0
	Features without noise	RF	0.55	0.50
		AdaB	0.54	0.31
Iteration 6: Undersampling	Weka Forward	RF (8 trees)	0.56	0.56
		RF (100 trees)	0.56	0.57
		RF (200 trees)	0.56	0.55
		AdaB (50 trees)	0.50	0.0
		AdaB (RF, 50 trees)	0.57	0.57
		AdaB (RF, 100 trees)	0.57	0.56
		AdaB (RF, 200 trees)	0.57	0.53
		AdaB (Decision stump, 100)	0.57	0.53
		AdaB (Decision stump, 50, 10 max leafs)	**0.58**	0.57
		AdaB (Decision stump, 100, 10 max leafs)	**0.58**	0.57

Table 18: Summary of the results from all iterations

As the six iterations provided several approaches we want to summarize our findings. Table 18 shows a high-level summary of the six iterations, the details can be found in each of the iterations. Furthermore, the table does not provide all approaches of Iteration 1 as no AUC nor recall was calculated, thus the detailed summary is available in Table 12 of Section 5.2.

5.8 Comparison of Other Approaches

Looking at the approaches from other participants on Kaggle, we recognize that most of them use XGBoost as a learning method. XGBoost stands for "Extreme Gradient Boosting"[27]. In gradient boosting new models are created that predict the error of prior models. The models are then added up to create the final prediction [14]. XGBoost has the main advantage to be fast, memory efficient and achieves high accuracy [55].

There are a lot of published contributions on Kaggle, however, we particularly analyze the notebook from Head or Tails[28] as it is the highest ranked notebook based on "hotness" which looks at the popularity of the work among other participants. We identify that the features were analyzed from a descriptive point of view. Furthermore, the features were organized into the corresponding group to allow an easier visualization. The features were compared to the claim rates to identify patterns in the data. After the general feature analysis the binary features were examined more in detail. More concretely, the binary features were the starting point for the feature engineering step. Heads and Tails focused on the binary features and the corresponding claim rate. Two new features were created: one that looked at the sum of binary features and the other that looked at the difference between the binary features. Each binary feature was analyzed based on the influence it had on the claim rate and the count was added up resulting in a new feature. For the feature that looked at the differences, the median of each binary feature was used as the reference row. If there was a difference in the binary feature to the reference row, it was counted and added to the feature. For a more detailed description the notebook can be used for reference. Interesting is that Heads and Tails assumed that the calculated features have little predictive power. We can see that other participants assumed the same as the calculated features were dropped several times. Heads and Tales used XGBoost as a learning method, however since there are no results available we cannot compare it to our approach. Moreover, in this notebook feature se-

[27]http://xgboost.readthedocs.io/en/latest/model.html

[28]https://www.kaggle.com/headsortails/steering-wheel-of-fortune-porto-seguro-eda

lection was initially not considered as all features were combined (the initial features with the newly created once). We can also identify that this approach does not consider oversampling or undersampling. However, looking at other approaches we can see that some participants either used oversampling combined with XGBoost as a classifier or weighted approaches. Latter puts additional weight on the minority class so that the dataset is more balanced.

6 Conclusion

This chapter is split into two sections. The first section concludes the thesis by summarizing the scope of the thesis and deriving conclusions based on the findings. The second section looks at future works. It provides the reader with ideas how the analysis can be expanded.

6.1 Summary

In summary, this thesis analyzed machine learning in general and examined classification techniques in particular. With that in mind, the main research question of this thesis is, *'What is the effectiveness of various feature selection and classification techniques?'*. To answer the research question it was necessary to identify and examine possible feature selection techniques, classification algorithms and suitable evaluation metrics. Therefore, three subquestions needed to be answered: Firstly, we needed to evaluate how to identify valuable features. Secondly, we needed to assess which algorithms were suitable for a classification problem and lastly, we needed to look into possible evaluation metrics that were suitable for the comparison of the classification algorithms. The subquestions can be answered as follows:

We discovered that valuable features can be identified through feature selection methods, that can generally be grouped into filter, wrapper and embedded methods. Furthermore, we identified that there exist several classification algorithms, however, we only introduced distance-based and tree-based based algorithms because linear-based models require linear separability of the data which is rarely the case in real-world scenarios. Moreover, we identified several metrics as useful for a binary classification problem. In detail, the evaluation metrics were accuracy, precision, recall, f-score and the area under the curve. Initially, we started to compare the classifiers with their accuracy, however, when we identified that the dataset was imbalanced, we focused on the other metrics. The following summarizes our findings based on the implementation of various feature selection methods, classification techniques and evaluation metrics on the dataset that were used in the practical assessment.

We applied all three feature selection approaches to our dataset and identified that the wrapper approach, using forward selection, resulted in a feature set which contributed to the best results. As our dataset was imbalanced we used oversampling and undersampling to improve the predictions. Based on whether the dataset was oversampled or undersampled, we achieved different results with the classifiers. We applied K-nearest neighbor, random forest and AdaBoost to the dataset and discovered that AdaBoost performed best on oversampled data when no parameters were tuned. However, our analysis showed that random forest outperformed AdaBoost on oversampled data when the parameters were tuned, thus the tuned random forest provided us with the best results on oversampled data. Furthermore, we discovered that on undersampled data random forest performed slightly worse than on oversampled data. On the contrary, AdaBoost performed slightly better on undersampled data than on oversampled data. In addition, we observed that the best results on undersampled data were achieved by a tuned AdaBoost classifier.

With regard to the main research question that looked at the effectiveness of various feature selection and classification techniques, we can conclude that the most effective feature selection method was a wrapper approach with forward selection. The most effective classification technique differs based on the feature subset that was used. Furthermore, it also depends on whether the data was oversampled or undersampled. However, we can conclude that AdaBoost performed best when used out of the box. And random forest performed best when parameters were tuned. K-nearest neighbor performed the worst among these three classification methods.

6.2 Future Works

The feature subsets only take the available features into consideration. However, feature engineering is seen as an important step in data-preprocessing which can lead to improved results. As we have not considered any feature engineering steps, it would be interesting to see whether new generated features have any impact on the prediction of the classifier. In particular, it could be analyzed whether feature combinations from the same group, such as two regional features, or combinations from different groups, such as regional and car related features, can improve the prediction. Furthermore, it would be interesting to see how the results change when a XGBoost classifier is applied to our best feature subset. Lastly, it would be interesting to evaluate the impact of dimensionality reduction techniques, such as a principal component analysis. Therefore, in future research, it would be interesting to see the differences in the results when dimensionality reduction methods are performed on the dataset.

References

[1] Porto seguro's safe driver prediction. *Kaggle. https://www.kaggle.com/c/porto-seguro-safe-driver-prediction (accessed on April 3).*

[2] Tech giants may be huge, but nothing matches big data. *The Guardian. https://www.theguardian.com/technology/2013/aug/23/tech-giants-data (accessed 12 November 2017)*, 2013.

[3] David W Aha, Dennis Kibler, and Marc K Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.

[4] JF Allaire. Introduction Ã l'analyse roc receiver operating characteristic. *Centre de recherche Institut Phillipe-Pinel de MontrÂ ´eal, Â ´ecole d'Â ´etÂ ´e*, 2006.

[5] Ethem Alpaydin. *Introduction to Machine Learning.* MIT Press, 2004.

[6] Michele Baldassarre. Think big: learning contexts, algorithms and data science. *REM-Research on Education and Media*, 2016.

[7] Gustavo EAPA Batista and Maria Carolina Monard. An analysis of four missing data treatment methods for supervised learning. *Applied artificial intelligence*, 17(5-6):519–533, 2003.

[8] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1):105–139, 1999.

[9] Christopher M Bishop. *Pattern recognition and machine learning.* springer, 2006.

[10] Indranil Bose and Radha K Mahapatra. Business data mining-a machine learning perspective. *Information & management*, 39(3):211–225, 2001.

[11] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[12] Gavin Brown. Ensemble learning. In *Encyclopedia of Machine Learning*, pages 312–320. Springer, 2011.

[13] Jason Brownlee. Bagging and random forest ensemble methods for machine learning. *Machine Learning Mastery. https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/ (accessed on 7 January 2018)*, 2016.

[14] Jason Brownlee. A gentle introduction to xgboost for applied machine learning. *Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/ (accessed on 9 April 2018)*, 2016.

[15] Bert Carremans. Data preparation & exploration. *Kaggle. https://www.kaggle.com/bertcarremans/datapreparationexploration (accessed on February 03 2018)*, 2017.

[16] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

[17] Olivier Chapelle, Bernhard Schoelkopf, and Alexander Zien. *Semi-supervised learning.* The MIT Press, 2006.

[18] N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique.

[19] Richard Clarke and Ari Libarikian. Unleashing the value of advanced analytics in insurance. *McKinsey & Company. https://www.mckinsey.com/industries/financial-services/our-insights/unleashing-the-value-of-advanced-analytics-in-insurance (accessed on October 25 2017)*, 2014.

[20] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent data analysis*, 1(1-4):131–156, 1997.

[21] Thomas Davenport. *Big data at work: dispelling the myths, uncovering the opportunities.* harvard Business review Press, 2014.

[22] Thomas H Davenport, Paul Barth, and Randy Bean. How big data is different. *MIT Sloan Management Review*, 54(1):43, 2012.

[23] Tom Fawcett. learning from imbalanced datasets. *sillicon valley data science. https://svds.com/learning-imbalanced-classes/ (accessed on March 08 2018)*, 2016.

[24] Danyel Fisher, Rob DeLine, Mary Czerwinski, and Steven Drucker. Interactions with big data analytics. *interactions*, 19(3):50–59, 2012.

[25] Peter Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

[26] Martin Ford. *The rise of the robots: Technology and the threat of a jobless future*. Basic Books, 2015.

[27] Mark A Hall. Correlation-based feature selection of discrete and numeric class machine learning. 2000.

[28] Mark A Hall and Lloyd A Smith. Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper. In *FLAIRS conference*, volume 1999, pages 235–239, 1999.

[29] David J Hand, Heikki Mannila, and Padhraic Smyth. *Principles of data mining (adaptive computation and machine learning)*. MIT Press, 2001.

[30] Nicolaus Henke, Jacques Bughin, Michael Chui, James Manyika, Tamim Saleh, Bill Wiseman, and Guru Sethupathy. The age of analytics: competing in a data-driven world. *Global Institute*, 2016.

[31] Rezarta Islamaj, Lise Getoor, and W John Wilbur. A feature generation algorithm for sequences with application to splice-site prediction. *Lecture notes in computer science*, 4213:553, 2006.

[32] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[33] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[34] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE, 2015.

[35] Esra Mahsereci Karabulut, Selma Ayşe Özel, and Turgay Ibrikci. A comparative study on the effect of feature selection on classification accuracy. *Procedia Technology*, 1:323–327, 2012.

[36] Saurav Kaushik. Introduction to feature selection methods with an example (or how to select the right variables?). *Analytics*

Vydhja. https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/ (accessed on January 02 2018), 2016.

[37] Kenji Kira and Larry A Rendell. A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, pages 249–256, 1992.

[38] Rob Kitchin. *The data revolution: Big data, open data, data infrastructures and their consequences.* Sage, 2014.

[39] Thomas Lal, Olivier Chapelle, Jason Weston, and André Elisseeff. Embedded methods. *Feature extraction*, pages 137–165, 2006.

[40] Huan Liu and Hiroshi Motoda. *Computational methods of feature selection.* CRC Press, 2007.

[41] Cindy Maike. Machine learning & its impact on the future for insurance. *Hortonworks. https://de.hortonworks.com/blog/machine-learning-impact-future-insurance/ (accessed on October 25 2017)*, 2017.

[42] Shaul Markovitch and Dan Rosenstein. Feature generation using general constructor functions. *Machine Learning*, 49(1):59–98, 2002.

[43] Bernard Marr. Big data: 20 mind-boggling facts everybody must read. *Forbes. https://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read/ (accessed 10 November 2017)*, 2015.

[44] Bernard Marr. *Big Data in Practice: How 45 Successful Companies Used Big Data Analytics to Deliver Extraordinary Results.* John Wiley & Sons, 2016.

[45] Robert John McMillan, Alexander Dean Craig, and John Patrick Heinen. Motor vehicle monitoring system for determining a cost of insurance, 1998. US Patent 5,797,134.

[46] Harvey J Miller. The data avalanche is here. shouldn't we be digging? *Journal of Regional Science*, 50(1):181–201, 2010.

[47] Michael Minelli, Michele Chambers, and Ambiga Dhiraj. *Big data, big analytics: emerging business intelligence and analytic trends for today's businesses.* John Wiley & Sons, 2012.

[48] Adriano Moala. Discussions. *Kaggle. https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/discussion/40222 (accessed 30 November 2017)*, 2017.

[49] Muhammad'Arif Mohamad, Dewi Nasien, Haswadi Hassan, and Habibollah Haron. A review on feature extraction and feature selection for handwritten character recognition. *International Journal of Advanced Computer Science and Applications*, 6(2):204–212, 2015.

[50] Hiroshi Motoda and Huan Liu. Feature selection, extraction and construction. *Communication of IICM (Institute of Information and Computing Machinery, Taiwan) Vol*, 5:67–72, 2002.

[51] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

[52] moutai10. How to plot a roc curve in scikit learn? *Cloud, Data Processing and Machine Learning. https://datamize.wordpress.com/2015/01/24/how-to-plot-a-roc-curve-in-scikit-learn/ (accessed on March 08 2018*, 2015.

[53] n.a. Filter-based feature selection. *Microsoft Developer Network. https://msdn.microsoft.com/en-us/library/azure/dn905854.aspx (Accessed on 27 December 2017)*, n.a.

[54] Nils J Nilsson. Introduction to machine learning. an early draft of a proposed textbook. 1996.

[55] Szilard Pafka. Benchmarking random forest implementations. *DataScience.LA. http://datascience.la/benchmarking-random-forest-implementations/ (accessed on 9 April 2018)*, 2015.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[57] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[58] Sebastian Raschka. What is an example of a filter algorithm in feature selection in machine learning? *Quora. https://www.quora.com/What-is-an-example-of-a-filter-algorithm-in-feature-selection-in-machine-learning (accessed on February 15 2018)*, 2016.

[59] Michael L Raymer, William F. Punch, Erik D Goodman, Leslie A Kuhn, and Anil K Jain. Dimensionality reduction using genetic algorithms. *IEEE transactions on evolutionary computation*, 4(2):164–171, 2000.

[60] Clemens Reimann, Peter Filzmoser, Robert G Garrett, and Rudolf Dutter. *Statistical data analysis explained: applied environmental statistics with R.* Wiley Online Library, 2008.

[61] Cynthia Rudin. Kernels. course notes. *MITOpenCourseware. https://ocw.mit.edu/courses/sloan-school-of-management/15-097-prediction-machine-learning-and-statistics-spring-2012/lecture-notes/MIT15_097S12_lec13.pdf (accessed on 15 January 2018*, 2012.

[62] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach.* Malaysia; Pearson Education Limited,, 2016.

[63] Eric Siegel. *Predictive analytics: The power to predict who will click, buy, lie, or die.* John Wiley & Sons Incorporated, 2016.

[64] Alex Smola and S. V. N. Vishwanathan. *Introduction to machine learning.* Cambridge University Press, 2008.

[65] Parikshit Sondhi. Feature construction methods: a survey. *sifaka. cs. uiuc. edu*, 69:70–71, 2009.

[66] Ernest T. Stringer. *Action Research: A Handbook for Practitioners.* Sage Publications, Inc., third edition edition, 2007.

[67] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[68] Kai Ming Ting. *Precision and Recall*, pages 781–781. Springer US, Boston, MA, 2010.

[69] Joris Toonders. Data is the new oil of the digital economy. *Wired. https://www. wired. com/insights/2014/07/data-new-oil-digital-economy/ (accessed 19 October 2017)*, 2014.

[70] Chun-Wei Tsai, Chin-Feng Lai, Han-Chieh Chao, and Athanasios V Vasilakos. Big data analytics: a survey. *Journal of Big Data*, 2(1):21, 2015.

[71] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2005.

[72] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

[73] Xingquan Zhu. *Knowledge Discovery and Data Mining: Challenges and Realities: Challenges and Realities*. Igi Global, 2007.

Appendix A

Appendix A provides all remaining figures that have not been presented in Chapter 5 of the thesis.

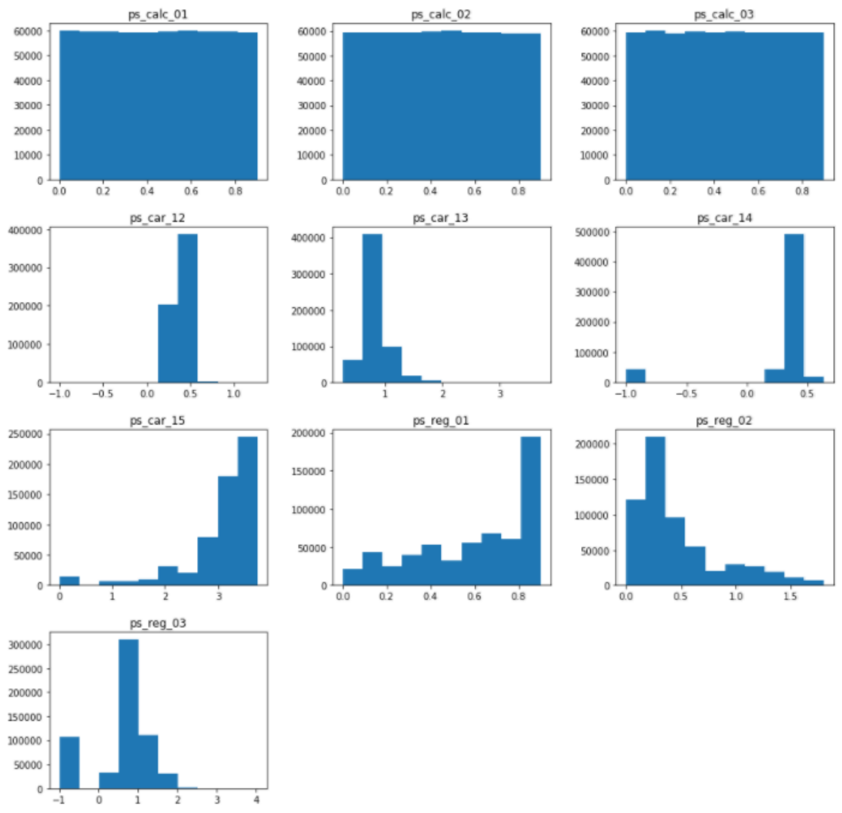


Figure 38: Histograms of continuous features

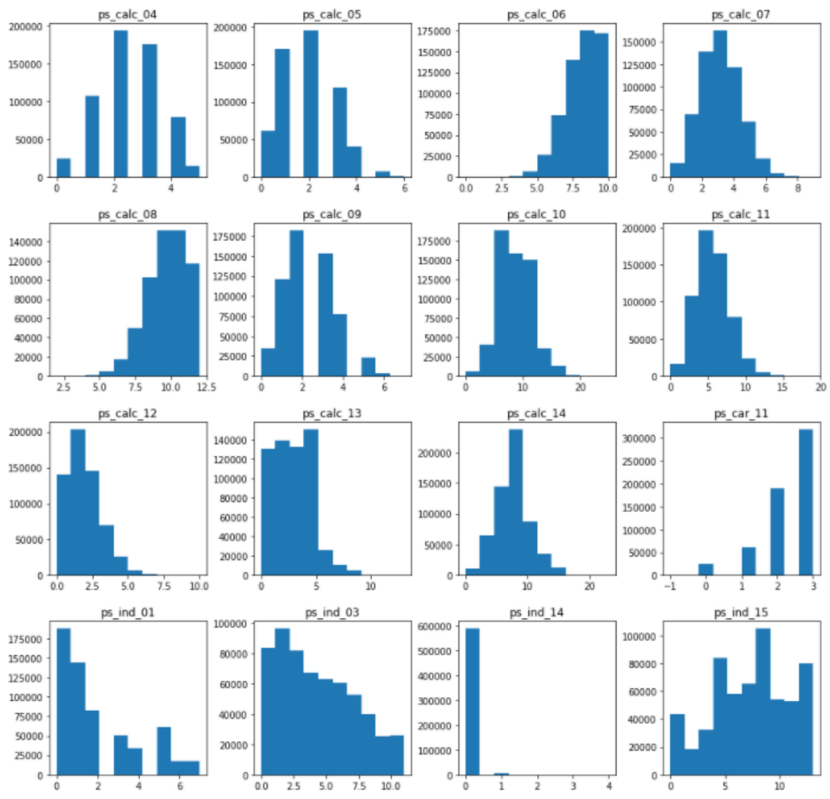


Figure 39: Histograms of ordinal features

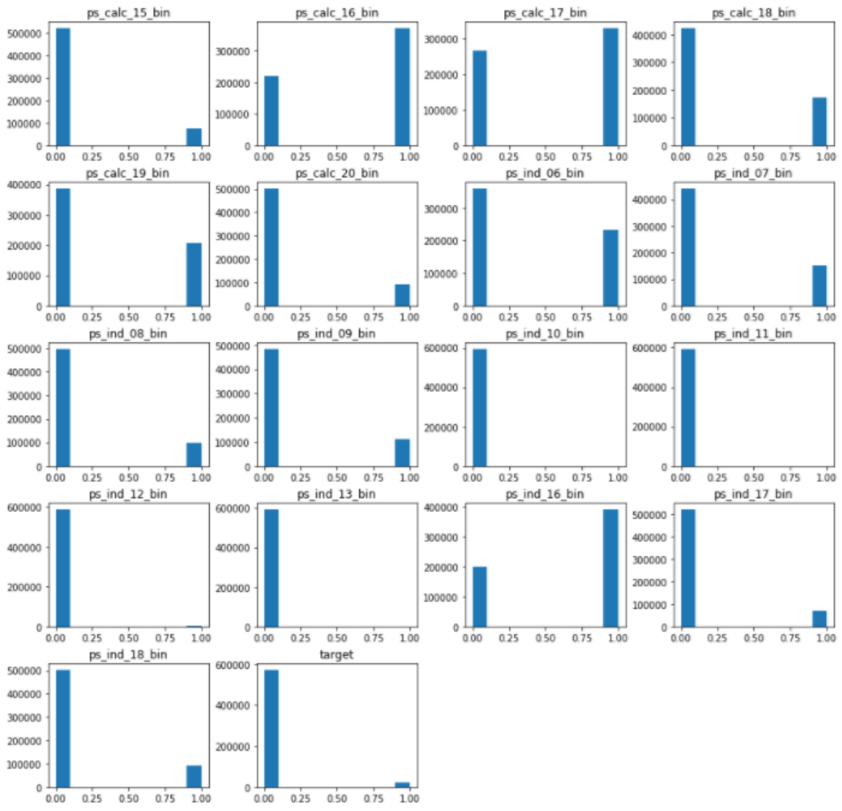


Figure 40: Histograms of binary features

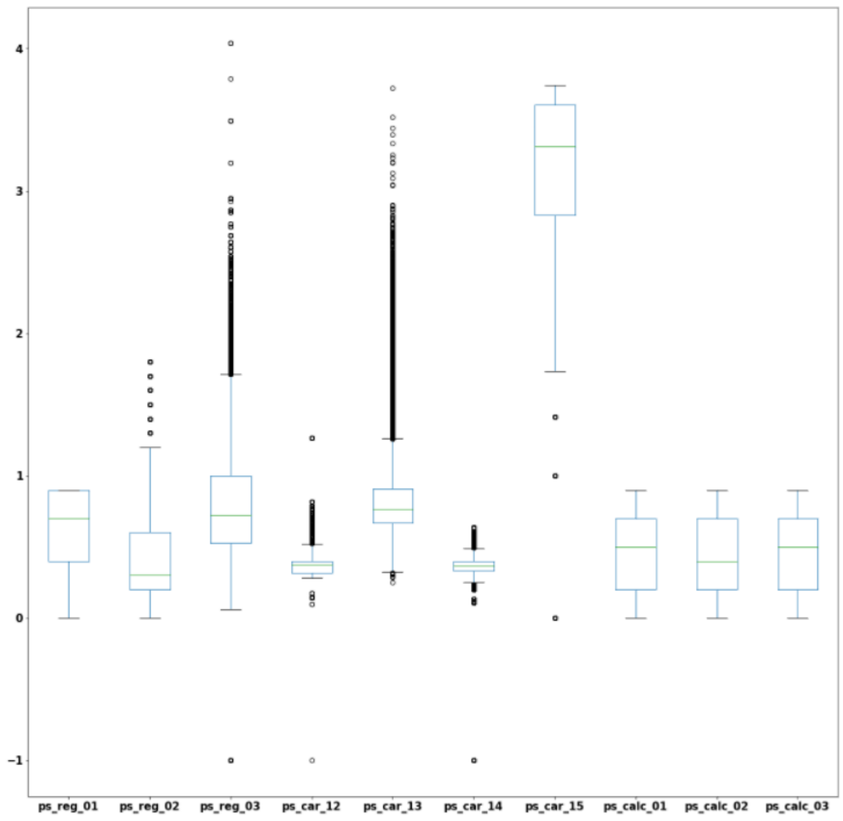


Figure 41: Boxplot of continuous features

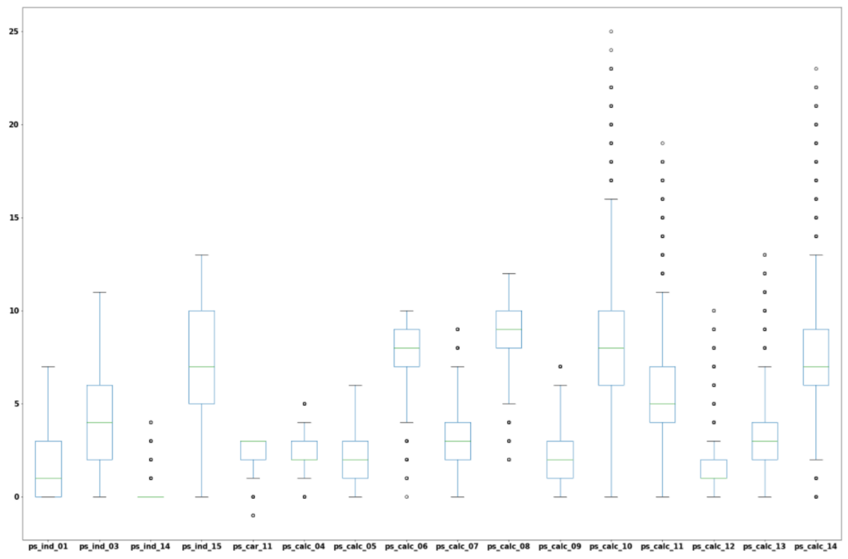


Figure 42: Boxplot of ordinal features