MASTER THESIS IN BUSINESS ANALYTICS & QUANTITATIVE MARKETING

A COMPREHENSIVE RESEARCH ON SMOTE: HOW TO DEAL WITH NON-CONTINUOUS VARIABLES IN IMBALANCED DATA

April 30, 2021

Erasmus University Rotterdam

Erasmus School of Economics

Author: L.M. van der Put

Supervisor: dr. M. Zhelonkin

co-reader: N.W. Koning, MSc.

Abstract

This research presents a method to properly incorporate discrete variables into the SMOTE algorithm, in order to contribute to solving the imbalanced data problem. Therefore, the method to calculate the distances between observation and the parameter value allocation has to be adjusted. The knn, bagging and random forest classifiers are considered to train the models for both simulated data and real-world datasets. The results clearly show that the adjusted SMOTE algorithm improves the classification accuracies in comparison with the situation without sampling techniques. When the differences in predictions with the existing SMOTE algorithm are looked into, especially the more comprehensive classifiers seem to improve with the new modification. This result is especially visible for the data simulation, and in a lesser extent for the real-world datasets. The adjusted SMOTE algorithm also provides better classification accuracies when the imbalance is higher, which is a very convenient conclusion.

Contents

1 Introduction

Since the previous century, deeper analytics of all sorts of data have gained more interest from a broader audience. This has several causes, for example the enlarged computational power and provided solution in areas such as machine learning. Due to this shift towards more data driven solutions of complex problems, there is a new demand for analysis of business processes [Hwang, 2018] [Biggio and Roli, 2018]. A lot of datasets that are used to solve these problems contain some kind of imbalance in the data. In this research, the main focus is to review and extend the current methodology in treating this imbalance problem properly.

Business Analytics can be used to answer several kinds of research questions. The focus with imbalanced data can for example be descriptive, prescriptive, predictive, or a combination of those [Greasley, 2019]. In this research, a prescriptive analysis will be performed. This means that the aim is to describe what should be done to achieve some set of goals. The goal is not only to predict more accurately, but also to investigate how these results can be found. In order to make a good prescriptive analysis on imbalanced data sets, first some basic concepts should be determined. The goal of supervised learning is to maximise the predictive accuracy and to draw a distribution for a classifier that is the same as that from the training data [Provost, 2000]. In case the response variable is not continuous but discrete, this is called a classification problem [He and Garcia, 2009]. Thus, the dependent variable has a finite number of values. So, in classification problems, one attempts to determine this value based on some explanatory variables. The imbalanced data problem appears in a dataset when the ratio between the classes is not evenly distributed: one class occurs much more frequent than the other [Chawla et al., 2004]. The class that occurs the most is the majority sample. The other class is the minority sample and therefore has less instances. In this paper, the majority samples will also be referred to as the *negative* samples and the minority samples as the *positive* samples. These terms will be used interchangeably.

The problem with classification in imbalanced data is that it is biased towards the sample that is more strongly represented. Assume a dataset for which one of the classes contains 99% of all the observations (e.g., negative). One can define a trivial classifier which assigns all instances to the majority, negative case. This classifier correctly predicts the new instances with an accuracy of 99%. These are the actual negative cases after all. It is very challenging to achieve a higher accuracy than this trivial classifier. However, in many cases we are specifically interested in the minority (or positive) case. In the medical sector for example, to predict the risk on some disease based on personal data and diagnosis in the past [Khalilia et al., 2011]. It is also applicable in marketing and sales datasets. One can try to predict whether employees or customers changes companies [Burez and Van den Poel, 2009] [Saradhi and Palshikar, 2011]. Imbalanced datasets are also common in fraud detection, because the number of fraudulent transactions is fortunately very low [Hassan and Abraham, 2016] [Wei et al., 2013]. Finally, research on imbalanced data can also contribute to the current COVID-19 pandemic. It can be useful to predict what pharmaceuticals actually affect the course of the disease or which people need to be hospitalized [Blanco-Melo et al., 2020]. Therefore, the imbalanced data problem is a relevant problem in data analytics that occurs across a lot of research areas [Visa and Ralescu, 2005].

The most commonly used solution for the imbalanced data problem is sampling. One can increase the minority sample (*oversampling*), decrease the majority sample (*undersampling*), or use a combination of both. In the literature, there are multiple techniques for these sampling methods. One that is widely used and supported, is the *Synthetic Minority Oversampling TEchnique* (SMOTE) [Chawla et al., 2002]. This oversampling technique uses the nearest neighbours of the minority samples to synthetically create new observations. This method will be explained into more detail later in the paper. However, an important aspect of SMOTE is that it uses distances between observations that belong to the minority class for oversampling. A problem arises in the current literature when discrete explanatory variables are considered. After all, the distance between ages can be calculated, that between different sexes cannot. Therefore, this method does not properly deal with discrete variables. In this research, the focus is to adjust SMOTE in a way that it can also handle these discrete variables. The main research question for this paper is therefore defined as followed:

How does a SMOTE algorithm, adjusted for continuous and discrete variables, affect the classification performance for several classifiers?

In order to answer this main research question, a number of sub-questions have been formulated:

1. How should the existing SMOTE algorithm be adjusted, in order to consider non-continuous variables properly in synthetically creating new minority samples?
2. How well does a for discrete variables adjusted SMOTE algorithm theoretically performs for different methods in a simulation study?
3. How well does a for discrete variables adjusted SMOTE algorithm performs in real-world datasets?
4. Are the conclusions from the theoretical framework obtained from the simulation study confirmed by the research on real-world data?

First, in Section 2, a research on the current literature of this subject will be discussed in the literature review. Thereafter, Section 3 explains the used methods to solve this imbalanced data problem. In Section 4, the used datasets and data simulation are elaborated. Subsequently, the results from the methods on these datasets and data simulation will be shown in Section 5. Finally, in Section 6, the conclusion and discussion, the influence on the current literature will be discussed.

2 Literature review

In the introduction above, the relevance of this research was already pointed out. In this section, the current literature on imbalanced data, sampling techniques and especially SMOTE will be discussed into more detail.

2.1 Imbalanced Data

At the end of the previous century, researchers already have run into problems with data that is not balanced. Due to the imbalance of the data, the positive observations have no or little representation in the sample [Mani and Zhang, 2003]. The problem especially arises when the interest of the researcher lies with the minority class, because the model parameters tend to bias towards the majority sample. For example, in medical research, where the number of cancerous cells or the number of persons that are diagnosed with some disease are duly outnumbered against the negative cases. Due to this imbalance, the minority sample cannot be accurately predicted [Sun et al., 2009].

The traditional approach to acquire balance in the data at that time, was to delete random observations from the majority class. It is justified to say that this is statically correct, however the loss of information caused by deleting observations is undesirable. This causes the predictions to lose their accuracy and the hypothesis tests to lose their power [Aitkin, 1978]. One of the solutions that is proposed is to use methods that are not affected by the imbalance of the data, for example tests that are specifically based on means or medians. After all, neither the mean nor the median is directly influenced by the fact that one sample contains more observations than the other [Shaw and Mitchell-Olds, 1993].

Another method to deal with the biased estimators caused by imbalanced data, is the *Jackknifing* procedure. The name of this method originates from the jack-knife of a boy scout that is *'rough and ready'*. This is because the jackknifing method can be used to reduce the bias in some specific situations where other standard statistical techniques do not apply

[Miller, 1964] [Tukey, 1962]. This method is adjusted by Shaw and Mitchell-Olds [1993] for unbalanced data. They find that robustness property is fulfilled, however the standard errors that are estimated by the method are not satisfactory. This technique is actually a precursor of other sampling techniques like *Bootstrap*. The jackknife resampling technique leaves out one observation every iterate and estimates model parameters over the sample. It uses the average of all these model parameters to approach the true parameters estimates [Efron, 1982].

2.2 Sampling

The techniques mentioned above works in some cases, however there was a call for a method that can truly fix the imbalanced data problem. For this reason, researchers started to use sampling techniques. As mentioned before, there are three main categories of sampling techniques that help to alter the class distribution of datasets. Firstly, one can reduce the observations in the majority sample. This is undersampling. Furthermore, the minority sample can be enlarged. This is called oversampling. The third sampling technique is a combination of both undersampling and undersampling [Shelke et al., 2017]. There are various techniques for these sampling methods, which will be discussed below.

For starters, the oversampling and undersampling techniques both have their benefits and drawbacks. Rationally, the major benefit of oversampling is that it ensures that the dataset is more balanced. However, oversampling can lead to overfitting the data. This is due to the fact that the new positive samples are created from the existing ones. It can therefore affect the accuracy of predictions on new data negatively when it is too specifically trained on the current set. The advantage of undersampling is that it decreases the amount of data. This can have a beneficial effect on the computation time and decrease the imbalance in the data. The downside of undersampling is that it can potentially eliminate important observations and thereby information [Chawla, 2009].

According to [Japkowicz et al., 2000], both undersampling and oversampling are effective methods to cope with imbalanced datasets. In their research, they compare random oversampling, random undersampling and a recognition-based oversampling technique. With random oversampling, random observations are selected and duplicated from the minority sample. With random undersampling, random observation from the majority sample are selected and deleted. The recognition-based technique that they use, is one of the earliest sampling techniques which uses a learner to construct new observations. When the two random sampling methods are compared, undersampling seems to be more effective. Their results show that the recognition-based technique has a lot of potential, however it still needs a lot of improvement. In their research, the performance metric that is used for each sampling method is the *Discrimination-based Multi-Layer Perceptron* (DMLP) which calculates the classification error levels for various settings. The disadvantage is that this technique assumes that the number of positive and negative samples are balanced [Japkowicz et al., 1995] [Eavis and Japkowicz, 2000]. Therefore, this performance measure is not widely used in the literature. The vast majority of the researches that focuses on the subject of imbalanced data use the *Confusion Matrix*, *Precision*, *Recall*, *F-measure* or the *ROC-curves* to evaluate the performance of the impending methods. These metrics evaluate the predictions of models by labelling them as *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) and *False Negative* (FN). These metrics therefore evaluate the ability of the methods to correctly classify a sample as positive or negative [Davis and Goadrich, 2006]. This is also the aim of this research. Therefore, these performance measures will also be used here. In Section 4, when the methodology is discussed, there will be a comprehensive description of these metrics.

Random oversampling is the most basic oversampling technique. However, the earlier mentioned drawbacks of oversampling are strongly present in this method: randomly duplicating positive samples increases the likelihood of overfitting [Batista et al., 2004]. For this reason, several other ways of oversampling have been devised.

Random OverSampling Examples (ROSE) is an extension of the random oversampling technique. It generates new synthetic samples of both the majority and minority classes with a bootstrap method, which is essentially a random sampling technique with replacement [Efron and Tibshirani, 1994]. The difference with ROSE is that these new samples are adjusted by the corresponding covariance matrix. This ensures that this method improves the accuracy of the classifier, particularly when the level of imbalance is high in the original dataset [Menardi and Torelli, 2014]. The ROSE method basically boils down to the following: one randomly picks one observation from the training set, regardless of which class it belongs to. The neighbourhood of this observation is defined by the probability distribution and covariance matrix of the sample. In this neighbourhood, a new sample of the concerning sample is generated. A fairly distributed dataset is formed by randomly selecting original and synthetically created samples [Lunardon et al., 2014].

Jo and Japkowicz [2004] came up with another way of oversampling. They argue that class imbalance itself does not directly cause the estimations of classifiers to degrade, but some small disjuncts in the dataset do. In their research, they try to define and locate these small clusters of the data to focus on, rather than the class imbalance itself. First, both classes of the dataset are clustered, after which these clusters are oversampled. This way, a dissimilitude is made between the within-class imbalance and the between-class imbalance. The within-class imbalance is the imbalance caused by the difference between clusters in a class and the between-class imbalance is the imbalance which occurs between the two classes. They claim that standard oversampling techniques only focus on the latter, while one should take into account both types of imbalance. Their method of oversampling is called *Cluster-Based Oversampling*. By oversampling the clusters, they want to deal with both the between-class imbalance and the within-class imbalance simultaneously.

Finally, in the beginning of this century, *Synthetic Minority Oversampling TEchnique* (SMOTE) was introduced by Chawla et al. [2002]. This method synthetically creates minority samples by choosing a random point on the line between two existing positive observations. Since Chawla et al. came up with this oversampling technique, it is the most broadly used method to cope with the imbalanced data problem [Fernandez et al., 2018] [Castellanos et al., 2018]. A lot of in dept studies have been performed and several extensions on SMOTE have been devised over the years. In the next subsection, these studies and extensions will be discussed into further detail.

Just like oversampling, multiple alternatives for random undersampling are conceived, e.g.: *One-Sided Selection* [Kubat et al., 1997], the *Near-Miss algorithms* [Zhang et al., 2016] and *Wilson's Editing* [Barandela et al., 2004]. However, in this research the focus is on an extension of SMOTE. Therefore, only the baseline random undersampling technique will be used to omit any potential increase of the performance measures by the undersampling technique. Thus, any improved results can be fully attribute to this adapted oversampling technique.

2.3 SMOTE

In the research of Chawla et al. [2002], an extensive research into the combination of undersampling and their new oversampling method is performed. They figured out a way to synthetically create new samples along line segments of some nearest neighbours of the positive sample. The selected number of observations near a positive sample is set on five, whereby the maximum percentage of oversampling is 500%. A new synthetic minority observation is created on one of the line segments of these nearest neighbours. When the oversampling is 100%, one of the neighbours is selected to realise a new positive sample. In case the oversampling is 200%, 300%, 400%, or 500% respectively two, three, four, and five nearest neighbours are selected. Thus, it is not possible to use one line segment twice. The performance metrics that they use in their research are precision, recall and the *Area*

Under the Curve (AUC), where the ROC-curve is defined as the curve [Provost and Fawcett, 2001]. The ROC-curve plots the performance of classification and the cost of the information distribution against each other, which basically means that the percentage of true positives is plotted against the percentage of false positives. Chawla et al. compare different levels and combinations of SMOTE and undersampling against random undersampling. The two base classifiers that they use are *C4.5* [Quinlan, 1992] and *Ripper* [Cohen, 1995]. In addition, they use a *Naive Bayes Classifier*, which they compare against a combination of C4,5 SMOTE and undersampling. For the Naive Bayes Classifier, a varying range of priors is used [Rish et al., 2001]. These methods are performed on 9 datasets with different levels of imbalance, varying from 1,88% to 34,90%. At the end of their research, they find that SMOTE in combination with any classifier performs superior in 44 of the 48 experiments. Therefore, SMOTE improves the classification performance in comparison with random oversampling with replacement. This result is attributed to the fact that SMOTE increases the decision region, while random oversampling could actually decrease it.

Since the introduction of SMOTE, a lot of researches on imbalanced data have used this technique with different classifiers and results. One research on this oversampling technique specifically mentions a major advantage, namely its simplicity: SMOTE ensures that standard regression models can be used, because it only manipulates the available training data [Torgo et al., 2013]. In their results, various variants of *Multivariate Adaptive Regression Splines* (MARS) [Friedman, 1991], *Support Vector Machines* (SVM) [Suykens and Vandewalle, 1999], and *Random Forest* (RF) [Pal, 2005] are applied for regression problems with SMOTE. They find that combining SMOTE with regular regression methods improves the accuracy of the predictions against simple undersampling or using the given data. Bhagat and Patil [2015] endorse these findings with their research on the performance of SMOTE with random forest on bigdata. Their results prove that the random forest classifier gets better results than any other classification technique in association with SMOTE. An extensive research on SMOTE is performed by Elreedy and Atiya [2019]. They use various settings and methods, for example the following three classifiers: *Decision Trees* (DT), Support Vector Machines and *K Nearest Neighbours* (KNN) [Song et al., 2007]. From their results, they conclude that for every classifier the improved prediction accuracy varies across settings. The results show that the gain in classification performance caused by SMOTE increases when the level of imbalance is higher. This result is convenient, because the data with a little minority sample needs the oversampling technique the most. Finally, another research that focuses on some adjusted SMOTE techniques is Wang and Pineau [2016]. They find out that methods with *Bagging* predict more accurately than methods with boosting according to the AUC's.

Because SMOTE is not infallible as oversampling technique, some extensions and modifications are contrived in the recent past. Modifications of SMOTE focus on a methodological alteration, while extensions use some other technique compared with SMOTE.

Borderline-SMOTE is an example of such a modification. This oversampling technique focuses more on the instances around the border of the sample, because these would have a greater impact on the classification accuracy rather than the samples farther away from the border. For this reason, only the positive instances around the border are used for oversampling. Thus, borderline-SMOTE differs from normal SMOTE due to the fact that it uses a subset of the minority sample, while SMOTE uses the complete set [Han et al., 2005]. In their research, they find that borderline-SMOTE produces better accuracy estimates than regular SMOTE.

Another modification of SMOTE is *Safe-level-SMOTE*. This method samples new positive samples across the same line segment as SMOTE, however it uses different weights. These weights focus on the instances of the minority's nearest neighbours. The safe-level is defined as the number of minority samples among the k nearest neighbours. The closer the safe-level is to k, the safer the sample is considered. When the safe-level is close to zero, one could see the sample as noise. Bunkhumpornpat et al. [2009], who came up with the method, retrieve better results for this method in comparison with both SMOTE and Borderline-SMOTE.

G-SMOTE is also a modification of SMOTE [Douzas and Bacao, 2017]. The focus of the research of Douzas and Bacao is to amplify the line segment that SMOTE uses to create new minority samples. This line segment is replaced by a geometric region. G-Smote allows the form to differ among settings, however a hyper-sphere is the basic figure. Just like the earlier mentioned modifications, also Douzasa and Bacao find significant better classification results in comparison with ordinary SMOTE.

The *SMOTE-IPF* (Iterative-Partitioning-Filter) is an extension of SMOTE where an existing technique is used to overcome some of the disadvantages [Saez et al., 2015]. In their opinion, SMOTE and the current extensions and modifications focus too much on blind oversampling. This means that oversampling is performed, while only the closeness of the samples is considered, without taking into account other characteristics of the data. By adding IPF after SMOTE is performed, they find superior results than earlier methods.

There exist a lot more extensions and modifications on SMOTE, for example SMOTEBoost ([Chawla et al., 2003], SMOTE-ENN [Batista et al., 2004], SMOTE-TomekLinks [Batista et al., 2004], ADASYN [He et al., 2008], MSMOTE [Hu et al., 2009], SMOTE-RS B* ([Ramentol et al., 2012], MWMOTE [Barua et al., 2012], SMOTE-D [Torres et al., 2016], et cetera. Chawla et al. [2002] indicate one important subject for further research which has not been extensively researched yet. They point out that synthetic samples can be created for continuous variables, because the distance between two observations is measurable. However, this is a lot harder for a dataset that also contains discrete variables. The SMOTE method, and all the extensions and modifications mentioned above, could potentially benefit from a modification which can take these nominal variables also properly into account. Therefore, this subject will be discussed in this study.

3 Methodology

In this section, the methods and techniques used are presented. First, the new SMOTE algorithm is explained, and a pseudocode is given. Then, the used classifiers are discussed. Hereafter, the performances measures are reviewed. Finally, the methods to deal with missing values in the datasets are elucidated.

3.1 SMOTE for continuous and discrete variables

Chawla et al. [2002] came up with an oversampling technique which uses line segments between an observation and their nearest neighbours to create synthetic positive samples. First, they find some k nearest neighbours of all the minority samples. Subsequently, lines are drawn between these nearest neighbours and the original observation. Depending on the amount of oversampling needed, a number of these line segments are randomly selected to create new samples. Each of these new observations is generated by taking a random point on this line. This procedure is visualized below in Figure 1 [Luengo et al., 2011]. In this figure, x_i is the positive sample for which new minority observations are created. x_{i1}, x_{i2}, x_{i3}, and x_{i4} are the nearest neighbours of x_i. The lines between x_i and its nearest neighbours are the line segments on which these new samples are randomly created. These potential new minority observations are respectively illustrated by r_1, r_2, r_3, and r_4. Which of these points actually becomes the synthetic sample is randomly determined. The number of points that is chosen depends on the amount of oversampling. For 100% oversampling, one of these points is considered. For 400%, all four points are included and therefore this is the maximum amount of oversampling possible in this example.

As mentioned before, a problem with this oversampling technique arises when discrete variables are present in the data as well. As can be seen from Figure 1 above, SMOTE is mainly based on the distances between observations. It is easy to calculate the differences between continuous observations. For example, one could measure the distance between

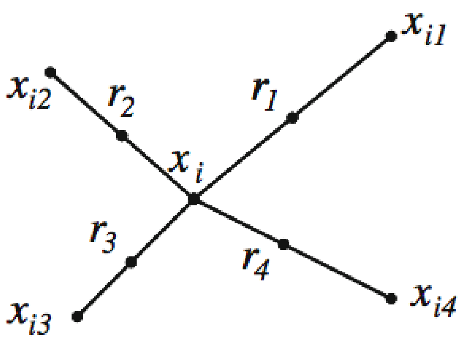

Figure 1: Example of the SMOTE procedure

ages, heights, incomes by subtracting one from the other. This gets trickier for nominal values. After all, how does one measure the distances between different sexes, colours, or nationalities? As mentioned in the previous section, Chawla et al. have already made some remarks on their oversampling method. They propose to extend this method for discrete variables in future research. In these remarks, they suggest to add the median of the standard deviations of the continuous variables to the total distance when the values of two discrete parameters differ.

Two methods to cope with distances between observation with both continuous and discrete variables are *HEOM* and *SHEOM*, where the latter is a scaled version of the former. The *Heterogeneous Euclidean Overlap Metric* (HEOM) is especially useful when distances need to be measured with both continuous and nominal values [Wilson and Martinez, 1997]. The method is described as followed: imagine there is a dataset with N observations and R attributes. The distance between every two observations x_i and x_j can be found with the Function 1 below:

$$HEOM(x_i, x_j) = \sum_{r=1}^{R} d_r(x_{i,r}, x_{j,r}), \tag{1}$$

The distance function d_r is described as followed:

$$d_r(x_{i,r}, x_{j,r}) = \begin{cases} \frac{|x_{i,r} - x_{j,r}|}{range_r} & \text{, when } r \text{ is a continuous variable,} \\ \delta_{i,j} & \text{, when } r \text{ is a discrete variable,} \end{cases} \tag{2}$$

where,

$$\delta_{i,j} = \begin{cases} 1 & \text{, when } x_{i,r} \neq x_{i,r}, \\ 0 & \text{, when } x_{i,r} = x_{i,r}. \end{cases} \tag{3}$$

Issues that can arise with the HEOM method are caused by scaling. It is important to note that the contribution of the continuous variables is bounded by zero and one. As shown above, when two observations have the same values for some attribute, both for continuous and discrete variables the distance added to the HEOM-distance is zero. When these values differ, for nominal attributes, one is added to the total distance. However, for continuous variables, the added value to this HEOM-distance is a ratio of the difference between the concerned points and the maximum distance. Therefore, only when the distance between two points is the maximum distance, the same value is added to the total distance for continuous attributes as for discrete attributes. In all other cases, the contribution to the total distance from continuous attributes is less than from discrete attributes. The HEOM distance could therefore be dominated by the nominal attributes when the proposed method of Chawla et al. is applied, which uses the median of the standard deviations for dissimilar values of discrete variables. It would depend too much on the unit of measurement. These problems are related to the scales of the variables and the scale of the contribution of the variables to this distance metric. Therefore, a scaled version of HEOM is introduced by Spencer et al. [2010]: the *Scaled Heterogeneous Euclidean Overlap Metric* (SHEOM). First, they propose to standardise the continuous variables before using the Manhattan distance. All continuous attributes after scaling have a zero mean and unit variance. It is good to emphasise that the median of the standard deviations of the continuous variables also becomes one after scaling. The first step of SHEOM, scaling the continuous attributes, can be presented as followed:

$$x^*_{i,r} = \frac{x_{i,r} - \overline{x}_r}{sd_r}. \tag{4}$$

In this Equation 4, i indicates one of the N indices. $\overline{x}_r$ and sd_r respectively present the mean and standard deviation of continuous attribute r. Then, the SHEOM distance between two observations x_i and x_j can be found with Function 5:

$$SHEOM(x_i, x_j) = \sum_{r=1}^{R} d^*_r(x_{i,r}, x_{j,r}). \tag{5}$$

The distance function d_r^* is described as followed:

$$d_r^*(x_{i,r}, x_{j,r}) = \begin{cases} \left| x_{i,r}^* - x_{j,r}^* \right| & \text{, when r is a continuous variable,} \\ \delta_{i,j} & \text{, when r is a discrete variable.} \end{cases} \tag{6}$$

In this Function 6, $\delta_{i,j}$ is defined in the same way as with HEOM. Note that SHEOM is actually the equivalent of HEOM, where the standard deviation is used for scaling rather than the range. All the standard deviations of the continuous attributes are equal to one, which is the assigned value that is added to the SHEOM distance when an inequality is perceived. Therefore, one could state that the idea of Chawla et al. is used, to add the median of these standard deviations to the total distance when two values of a discrete variable do not match. Finally, SHEOM fits the requested properties for measuring the total distances in datasets with both discrete and continuous attributes. Therefore, this method will be used to complement to distance calculations in this research.

Based on the obtained distances from SHEOM as described above, the k nearest neighbours from the minority sample are determined by selecting the samples with the k smallest distances. Now, it is time to create the synthetic minority samples from the observations and their neighbours. Assume that the five nearest neighbours are considered. The oversampling cannot exceed the number of nearest neighbours as indicated before, because every line segment can only be used once. First of all, a number of these line segments at hand are randomly selected. When the oversampling is 100%, one line segment is selected. When the oversampling is 500%, all five line segments are selected. For each selected line segment, one minority sample is synthetically created. The values of the continuous attributes of the new minority sample can be obtained in the same way as with regular SMOTE. The values of the discrete attributes are based on the values of these attributes for the original observation and its nearest neighbours. The ratio in which certain outcomes occur across these observations determines the assigned parameter value of a synthetic sample.

For example, there is a dataset with two continuous and three discrete variables. A particular minority observation S_1 can be represented as followed: $S_1 = [5, 10, A, B, C]$. There are three nearest neighbours selected, i.e., $S_2 = [5, 20, X, B, C]$, $S_3 = [10, 10, X, B, Y]$, and $S_4 = [10, 20, A, B, Z]$. Assume that the level of oversampling is 300%, which means that each nearest neighbour is used to create a synthetic observations. The values of the continuous attributes for these three new minority samples are a random point on the line between the standardized values of the original sample S_1 and the selected nearest neighbours S_2, S_3, or S_4. The values of the discrete attributes are based on the ratio of occurrence across these selected minority samples. The first discrete variable has two potential outcomes which occur equally often in this example. So, the assigned value has a fifty-fifty chance to be A or X. The second one has the same value for all four observation, whereby the value B is plainly attributed. Finally, the last discrete variable has three potential outcomes which occur respectively two times (C), one time (Y), and one time (Z). Therefore, the odds to assign these values to this attribute for the synthetic sample are respectively 50%, 25%, and 25%. Mathematically, this can be shown as followed:

1. $P(S_{syn,3} = A) = 0.5$, $P(S_{syn,3} = X) = 0.5$
2. $P(S_{syn,4} = B) = 1$,
3. $P(S_{syn,5} = C) = 0.5$, $P(S_{syn,5} = Y) = 0.25$, $P(S_{syn,5} = Z) = 0.25$

The pseudo code of this described algorithm to synthetically create new minority samples can be found below in Algorithm 1. The actual R code is included in appendix A. The input variables of the algorithm are *Data, Oversampling, Undersampling* and *k.* The output is obviously a more balanced dataset. The input variables can be described as followed:

- *Data:* This is the unbalanced dataset with both the minority and majority samples. Important is that the dependent variable is in the first column of this dataset.
- *Oversampling:* This is the requested amount of oversampling. This variable is a factor, which means that the number of initial minority observations is multiplied with this value to find the number of synthetically created minority observations in the output dataset. The default value is two.
- *Undersampling:* This is the proportion of undersampling that is considered. This variable is a ratio, which means that the value indicates the selected number of majority observations in proportion to the minority sample in the output dataset. When this value is one, the majority and minority samples are the same size. When this value is two, the majority sample is twice the size of the minority sample. The default value is one.
- *k:* This is the number of considered nearest neighbours. Note that this limits the oversampling, because every line segment can only be used once. The default value is five.

Algorithm 1 SMOTE Pseudocode

```
Input: Data, Oversampling = 2, Undersampling = 1, k = 5
Output: Minority Sample with synthetic created observations
/* The variables Oversampling, Undersampling and k have a default */
N = number of observations in Data
p = number of attributes
Nma = number of majority observations in Data
Nmi = number of minority observations in Data
Split the data in the Minority and Majority sample
/* Undersampling the Majority Sample */
Determine the number of observations for the Majority Sample, based on the current imbalance and given variable undersampling
Sample the observations from the Majority Sample random and without replacement
/* Oversampling the Minority Sample */
Function: Distance        // This function calculates the distances between observations
for All Continuous Variables do
    Standardize the values of the continuous variables to mean zero and unit variance
distance[ ][ ] <- N x N - Matrix with the distances between the observations
 for i in 1:N do
    for j in 2:p do
        if Attribute j is nominal then
            Compare observation i with all other observations for attribute j. If the values of the observations do not match, the value
              1 is added to the distance: $distance[i][j] = distance[i][j] + 1$
        else if Attribute is categorical then
            Compare observation i with all other observations for attribute j. $distance[i][.] = distance[i][.] + |data[.][j] - data[i][j]|$
Function: matrixnn  // This function determines the k nearest neighbours of the observations
matrixnn[ ][ ] <- N x k - Matrix with the k nearest neighbours of the observations in the columns
 for i in 1:N do
    for i in 1:k do
        Find the minimum distance to observation i in matrix distance. Save this observation in matrix matrixnn and delete this one
          from matrix distance.
Function: synthetic        // This function creates synthetic minority samples
$m = oversampling - 1$ = Number of line segments, or nearest neighbours, to use
$Nsyn = Nmi * m$ = Number of synthetic minority samples
synthetic[ ][ ] <- Nsyn x p - Matrix for the new synthetic observations
for i in 1:Nsyn do
    Choose random m of the nearest neighbours of observation i from matrixnn
    for j in 2:p do
        if Attribute is nominal then
            values = Save all the values of this attribute for the original sample and its nearest neighbours.
            synthetic[newsample][j] = Select random one of the values from values
        else if Attribute is categorical then
            difference = difference of this attribute between the minority sample and its nearest neighbour
            random = choose random number between 0 and 1
            synthetic[newsample][j] = minority[i][j] + ( difference * random )
minority = rowbind ( minority, synthetic )
newdata = rowbind ( minority, majority )
```

3.2 Methods

Three classification methods are considered in this research: *KNN*, *Bagging* and *Random Forest*. These methods are used because they are somewhat similar. Random forest is a decision tree method just like bagging; however, it builds on the methodology of bagging. Both bagging and random forest thereby use the technique from the knn classifier. This way, it can be determined whether a more extensive classifier of some sort also returns better results. These methods are described below into further detail. All three methods are included in the R function *train* of the *CARET* package, which will be used to train the models [Kuhn et al., 2008]. First, the datasets are split in 75% train set and 25% test set, where the classifiers are trained on the train set. Important to note: the models are trained using 5-fold cross validation.

3.2.1 KNN

The first classification method that will be discussed in this section, which is also the most basic, is the *K Nearest Neighbours* (KNN) classification method. This classification method is a popular method in data analytics, because it obtains significant classification results whereas it is very easy in its implementation [Guo et al., 2003]. Simply put, the knn classification method classifies a sample based on the majority vote of the nearest neighbours. For this reason, k should always be on odd number. Otherwise, one could end up in an indifferent scenario. For the binary depending variables considered in this research, this looks as followed:

$$y_i = \begin{cases} 1 & \text{, when } \sum_{k=1}^{K} I(y_k = 1) > \sum_{k=1}^{K} I(y_k = 0), \\ 0 & \text{, when } \sum_{k=1}^{K} I(y_k = 1) < \sum_{k=1}^{K} I(y_k = 0), \end{cases} \tag{7}$$

where K is the number of nearest neighbours considered and y_i is the binary dependent variable for observation i.

As seen before, the nearest neighbours of a sample are determined by taking the observations with the smallest distances to this specific sample. The performance of this classification method is quite dependent on the value of k. Therefore, it is important to not take a random value for k. One should compare the classification results for several values [Zhang et al., 2017b]. In this research, different values of k are analysed based on the *Root Mean Squared Error* (RMSE). This measure indicates the deviation from some estimated model to the values that are observed. The RMSE is defined as the Equation 8 below. Logically, the value of k with the lowest RMSE, and therefore the closest to the estimated model, is the preferred value [Zhang et al., 2017a].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (\hat{y}_i - y_i)^2}{N}} \tag{8}$$

According to Guo et al. [2003], the biggest drawback of the knn classification method is that it is a lazy learner. This means that the method does not learn generalizations from training values but classifies the samples one by one. In other words: a lazy trainer does not spend a lot of time on training. However, the classification time is longer. Consequently, the efficiency of other classifiers is better in general. This knn classification method will therefore be a good benchmark for the next two methods.

3.2.2 Bagging

The next classification method that will be discussed is *Bagging*, which is an acronym of Bootstrap Aggregation [Breiman, 1996]. This method is, just like the next classification method *Random Forest*, a decision tree learning method. This implies that the value of a target variable has been deducted by a sequence of nodes: the decision tree. When the target value is discrete, it is referred to as a classification tree. In decision trees, there are branches where a decision about a specific item is made. Such a decision is based on the maximum possible split in the data: one chooses the split out of all the possible splits that gives the biggest gain in information [Breiman et al., 1984]:

$$G(Q,\theta) = \frac{N_{left}}{N} H(Q_{left}(\theta)) + \frac{N_{right}}{N} H(Q_{right}(\theta)) \tag{9}$$

In this Equation 9, G is the information gain, N is the total number of observations, N_{left} and N_{right} are the number of observations in both nodes, Q is the total data, Q_{left} and Q_{right} are the data in both leaves, θ is the considered set of attributes in the model and H is the chosen splitting criterion. This actually means that the observations in both leaves of the node are most homogeneous among each other, but heterogeneous in comparison with those in the other leaf. The most common selection criterion for classification is the Gini index, which is also considered in the used R package [Laber and Murtinho, 2019]:

$$H(X_m) = \sum_{k} p_{mk}(1 - p_{mk}) \tag{10}$$

When these branches of the decision tree are followed, one finds the final assignment of the target attribute at the end of the last note. Bagging is special in the way that it uses an approach with some sort of model averaging [Sutton, 2005]. Bagging uses bootstrap as sample technique. This bootstrap technique randomly samples observations from the original dataset with replacement. Therefore, some observations will be unique, and some will be duplicates. The observations that are left out are collected in the out-of-bag dataset. This out-of-bag dataset can be used to test the performance of the constructed decision tree. The advantage of bootstrap sampling is that, because of the methodology of this technique, all the samples are independent of each other. For the classification method bagging, a model is fitted for each bootstrap sample. At the end, when a specified number of bootstrap samples is generated, all these models are combined by taking the average of the output. For classification trees, this means that the final classification is based on the majority vote of all the constructed trees [Lemmens and Croux, 2006].

The biggest strength of bagging as classification method is that it effectively reduces the variance of the prediction. Therefore, it is very useful for learners that are prone to overfitting. The predictive power and stability are improved for classification. Bagging is thereby widely applicable for a lot of different settings. This makes it very user friendly and an easy method to boost the performance of a classification method. The disadvantage of bagging comes from the fact that it takes the average of a lot of decision trees and their models. Therefore, the biggest advantage of decision trees, the interpretability of the model, evaporates with bagging [Prasad et al., 2006].

3.2.3 Random Forest

As mentioned in the section above, random forest is just like bagging a decision tree learning method. It is actually quite similar to bagging, with one important tweak. With bagging, all the attributes included in the dataset can be allocated to perform a split. Moreover, the decision split rule can be the same for every separate tree. This causes a potential high correlation between the trees and their predictions, because they possibly show a great number of similarities. Random forest responds to this potential flaw of bagging. In the algorithm of random forest, one has to specify the number of attributes that are used to perform the tree splits. The attributes that will be used are randomly selected from to whole set of variables when the decision tree is build. This also explains the name of the method [Breiman, 1999]. The various decision trees are slightly correlated at most, because the randomly selected attributes are different in each tree. As a result, the bias from the classification model is low. Random forest therefore provides models that are more suitable for prediction and classification in comparison with bagging. Thus, random forest is indeed very similar to bagging, apart from the randomly selected attributes that are used [Strobl et al., 2009]. The algorithm of a random forest classifier can therefore be illustrated by the following steps [Belgiu and Drăguţ, 2016]:

1. Create T samples by Randomly selecting N observations from the original datasets with replacement
2. Randomly select p attributes from the original set of variables for every sample t
3. Create a decision tree based on every sample t, with nodes only determined by the corresponding p attributes
4. Merge all T decision trees into one final decision tree, which is the random forest classifier.

The number of trees selected in both bagging and random forest are important parameters for the prediction accuracy. There is a trade-off between the accuracy and the computational cost. When more trees are considered, the AUC, and therefore the prediction accuracy, increases. However, the computational cost also increases with these parameters. The increase of the AUC slowly decreases and converges therefore asymptotically, while the computational cost increases fast. The balance between these factors is subjective and case depending [Oshiro et al., 2012]. The considered R package uses a default of 500 trees, which is quite standard for these classification methods. Therefore, the number of trees for both random forest and bagging is set to be the default of 500. The number of attributes that will be selected in every tree for random forest is random, but consequent over all settings and iterations.

As mentioned before, the biggest strength of random forest is that the selection bias is kept low by randomly selecting attributes for every tree. This causes the classification to become more accurate. Just like bagging, it also decreases the variance of the predictions. The biggest limitation of this method is that it is even less interpretable than bagging. Furthermore, this classification method can be very computational costly [Prasad et al., 2006].

3.3 Performance Measures

In this research, the focus is on the prediction accuracy of the methods. When the results of the classification methods described above are obtained, it is obviously important to review what method and which settings induce good predictions. Therefore, the performance metrics are discussed in this subsection.

When a classification method is tested, the dataset is first randomly divided in a train set and a test set. The model is trained based on the data from the training set. Thereafter, these trained models are used to classify the observations of the test set. This allows us to compare the classification results with the actual labels. An important concept in these classification studies is therefore the confusion matrix. This performance metric is the basis from all the other measures that will be discussed later. The confusion matrix is shown below in table 1:

	Actual Positive	Actual Negative
Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative

Table 1: Confusion Matrix

The abbreviations of these categories are widely used in the literature: TP (True Positive), FP (False Positive), FN (False Negative) and TN (True Negative). From this confusion matrix, three important performance measures can be derived [Johnson, 2019]:

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$

Therefore, accuracy gives the percentage of correctly predicted classifications. This is however not a great performance metric with imbalanced data, because it will be dominated by the negative class. For this reason, precision and recall are considered to be more appropriate for this kind of data. Precision indicates which part of the samples that was classified as positive is actually correctly classified as positive. On the other hand, recall shows what percentage of the actual positive samples is actually correctly labelled as positive. Therefore, this is also referred to as the *True Positive Rate* (TPR) or the *sensitivity* of the model. Furthermore, another definition to measure the prediction accuracy of these classification methods is the *specificity*. This is the ratio between the false positives and the actual negatives, which is therefore also known as the *False Positive Fraction* (FPR) and can mathematically be described as followed: $FPR = \frac{FP}{FP+TN}$. It would be interesting to set this TPR/recall/sensitivity and the FPR/specificity against each other in a graph to visualize the classification performance. In the literature, there is such a curve which can be referred to as the *Receiver Operating Characteristics curve* (ROC-curve) [Provost and Fawcett, 1997]. The Y-axis is the sensitivity, and the X-axis shows the specificity. When the classification is performed by a method, it provides a number between 0 and 1 (or 0% and 100%). Based on some threshold, these samples are attributed to the positive and negative classes. The ROC-curve gives an overview of the TPR and the FPR for various thresholds. When this threshold is higher, less items are classified as positive and therefore the number of false positives and true positives decreases. Thus, these ROC-curves can be used to study the classification performance of different methods. However, it is not always convenient to just compare these graphs. An appurtenant metric of this curve, that would allow us to compute the absolute performance of the methods, would be pragmatic. The *Area Under the Curve* (AUC) is such a metric. This calculates the area under the ROC-curve, whereby the value indicates the classification performance [Bradley, 1997]. An illustrative example of a ROC-curve and the AUC can be found in figure 2 below:

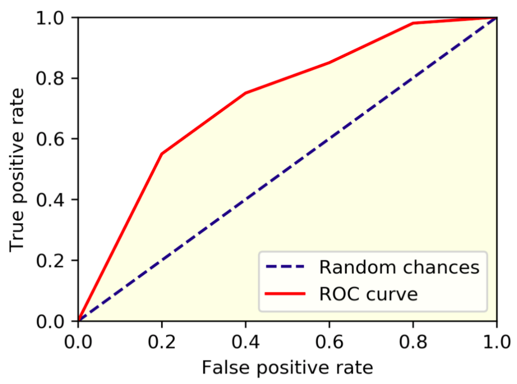

Figure 2: illustrative example of a ROC-curve and the AUC

The ROC-curve always starts in the origin. Here, all samples are classified as negative. Therefore, no true positive and false positive samples are present. Needless to say, the coordinate (1,1) naturally indicates the case that all samples are labelled as positive. The point (0,1) is quite interesting: this marks to situation of perfect classification. Therefore, the closer the ROC-curve is to this point, the better the performance will be. The striped blue line indicates the strategy where the class of a sample is randomly guessed. Thus, the ROC-curves should be at least above this line. As mentioned above, the area under the curve measures the classification performances of the model. For perfect classification, this area is exactly one. When the classifier performs is less than perfect, the ROC-curve moves towards the random line. Thus, the AUC moves towards to the corresponding area of 0.5. In summary, the AUC lies somewhere between 0.5 and 1. The closer this metric is to one, the better the classification performance of the corresponding method [Fawcett, 2006].

3.4 Missing Values

In this research real-world datasets are used, and a data simulation is performed. It happens very often that missing values are observed in a dataset. Therefore, the method to cope with this problem is discussed here.

A distinction can be made between three different types of missing values: *Missing Not At Random* (MNAR), *Missing At Random* (MAR) and *Missing Completely At Random* (MCAR) [Little and Rubin, 2019]. When the missing values are MNAR, a problem arises. These missing values are not randomly distributed over the attribute itself. Therefore, they can be denoted as followed: $P(X_{miss}|X) = P(X_{miss}|X_{obs}, X_{miss})$. An example of these kind of missing values is when people who own a more expensive residence do not want to report the value of their property. In that case; the higher the value of the realty, the higher the probability of a missing value. The other types of missing values, MAR and MCAR, can respectively be denoted as followed: $P(X_{miss}|X) = P(X_{miss}|X_{obs})$ and $P(X_{miss}|X) = P(X_{miss})$. This indicates that MAR observations are conditionally random: they are random in the variable where the missing values appear, when these are controlled for the other attributes. An example of these kind of missing values is when, for instance, elderly people are more sceptical against providing personal information. In that case, there is a higher probability of missing values in personal details for samples with a higher age. However, when we control for this mechanism, the missing values are random in the variable that reflects this personal information. For both MNAR and MAR the observations with missing values cannot be deleted from the dataset, because this will initiate a bias due to the unrepresented segments of the population. Both should therefore be handled with care. On the other hand, observations that contain MCAR values could be left out of the research. These kinds of missing values are completely random distributed over the data. However, deleting these observations may lead to an increase of the standard errors and a reduction of the confidence intervals. Therefore, these observation can only be discarded when the sample is large enough [Acock, 2005].

Accordingly, it is important to properly deal with missing values when they are not completely random distributed over the dataset. There are multiple methods contrived in the literature to approximate these missing values. These methods are referred to as imputation methods. The imputation method that will be used in this research is *KNN-imputation* [Troyanskaya et al., 2001], because this method is appropriate for MCAR as well as for MNAR and MAR values. This is a convenient method that assumes that a missing value can be approximated with the help of some complete observations close to this incomplete sample. The big advantage of this imputation method is that it is non-parametric. Therefore, there is no need to make any assumptions about the distribution of the data. It uses a weighted average of the considered attribute's k nearest neighbours. For this reason, the SHEOM method can be used in combination with this method. The weights of the observations are based on the similarity of the observation in comparison with its nearest neighbours. The R-function *kknImp* of the *performanceEstimation* package is used to conduct the knn-imputation [Torgo, 2014].

4 Description of data

In this research, an adjusted form of SMOTE is extensively tested. First, the theoretically expected performance is tested through a data simulation study. Thereafter, the oversampling technique for both continuous and discrete variables is tested on real-world datasets with different levels of imbalance. First, insight is gained into the results which could be expected on ideally constructed datasets, after which it is attempted to reproduce these outcomes in real-world datasets. Every method described below in both the study on simulated data and real datasets is repeated 50 times to avoid coincidence in the results. The averages of the 50 repetitions are used to answer the research questions.

4.1 Data Simulation

For the simulation study, first, the data must be simulated. The number of explanatory variables and the appurtenant signal-to-noise ratio could possibly affect the prediction accuracy of the models. Thereby it could be interesting to use different ratios of continuous and nominal attributes. Therefore, three models will be considered. The first one will contain four independent variables: two continuous attributes, one binominal attribute and one discrete attribute with five levels. The second model contains 10 independent variables, where the first four are defined exactly the same as before. The six attributes that are added in comparison with earlier model are equally distributed between binominal and discrete variables. The final model contains twenty explanatory variables, where the first ten are defined as before. Again, the added attributes are equally distributed between binominal and discrete variables.

For imbalanced data, logically, the dependent variable follows a binomial distribution $Bin(n, p)$ with $n = 1$. In this distribution, n is the number of successes and p is the probability of this success. For this data simulation, it is assumed that this probability follows a logistic model. Therefore, the three models can be written as followed [Peng et al., 2002]:

$$Logit(\pi_4) = \beta_0 + \sum_{p=1}^{4} \beta_p x_p \tag{11}$$

$$Logit(\pi_{10}) = \beta_0 + \sum_{p=1}^{10} \beta_p x_p \tag{12}$$

$$Logit(\pi_{20}) = \beta_0 + \sum_{p=1}^{20} \beta_p x_p \tag{13}$$

For all three models, x_1 and x_2 are the continuous attributes and therefore assumes to be normally distributed with respectively distributions $x_1 \sim N(0,1)$ and $x_2 \sim N(2,1)$. The binominal variables follow a binomial distribution where $p = 0.5$, thus $Bin(1, \frac{1}{2})$. Finally, the discrete variables follow a discrete uniform distribution, because we assume that the finite numbers of outcomes all have the same probability. Therefore, they are distributed conform $Unif(1,5)$. B_0 is the intercept of each model. This intercept and the other coefficient are arbitrary chosen, considering a signal-to-noise ratio: $B_0 = 0.5$, $B_1 = 1$, $B_2 = -0.5$, $B_3 = 1$, $B_4 = -0.5$, $B_5 = 1$, $B_6 = 0.25$, $B_7 = -0.75$, $B_8 = -1$, $B_9 = -0.25$, $B_10 = 0.75$, $B_11 = -1$, $B_12 = -0.5$, $B_13 = 1.25$, $B_14 = -0.25$, $B_15 = 0.75$, $B_16 = 1$, $B_17 = 0.5$, $B_18 = -1.25$, $B_19 = 0.25$, $B_20 = -0.75$.

As mentioned before, the dependent variable follows a binomial distribution where the probability of success is indicated by π. This can be derived from the logistic models of equations 11, 12 and 13 above as followed:

$$\pi = \frac{e^{\beta_0 + \sum_{p=1}^{P} \beta_p x_p}}{1 + e^{\beta_0 + \sum_{p=1}^{P} \beta_p x_p}} \tag{14}$$

This simulation as described above gives 2000 observations for each model where, on average, 25% of the observations belong to the minority sample and 75% belong to the majority sample. To test the classification performance of the adjusted SMOTE technique proposed in this research, different amounts of imbalance are selected for each model. The levels of imbalanced that will be used are: 1%, 5%, 10%, 15%, 20%, 25% and 30%. In other words, 21 different datasets are considered in the simulation study. For each level, a total number of 1000 observations with the corresponding ratio between the majority and minority samples is used. These are randomly selected from the simulated dataset. For example, for the 15%-level randomly 150 observations are sampled from the minority set and 850 from the majority set.

The three classifiers discussed earlier are used to perform the classification on these simulated datasets. First, the concerned dataset is split into a 75% train set and a 25% test set. Thereafter, the model is trained on the train set, using 5-fold cross-validation. The main focus of this simulation study is on the classification performance of different (combinations of) sampling techniques. First, the models are trained on the dataset without oversampling or undersampling. Thereafter, the current and the adjusted SMOTE techniques are executed to test the improvement in classification without undersampling. With these oversampling techniques, three different number of nearest neighbours (k = 1, 3 and 5) and several levels of oversampling (100%, 200%, 300%, 400% and 500%) are implemented. Finally, both SMOTE techniques are used in combination with random undersampling, again with the same number of nearest neighbours and oversampling levels as before. When both oversampling and undersampling are combined, the focus is on completely balanced datasets after sampling, because the most interesting case is when the imbalance is completely resolved. Sun et al. [2009] argue that the best classification results can be found when the majority sample is undersampled until the number of negative observations is equal to the number of positive observations. In summary, the only values which are tested for undersampling are zero (no oversampling) and one (equal number of minority and majority samples).

Then a final remark: it will be prevented that the majority sample will be oversampled. This can be the case when the number of minority samples outgrows the majority observations. These oversampling levels are discarded from the research.

4.2 Real-world datasets

In this subsection, the real-world datasets are briefly discussed. Three datasets are obtained from the UCI Machine Learning Repository, while three other datasets are retrieved from Kaggle. The different datasets represent the relevance of the imbalance data problem across different research areas. Furthermore, the level of imbalance is neatly distributed over a range from 0,5% to 25%. These levels are shown in table 2 below. Tables with statistics of the continuous variables and barplots of the discrete variables can be found in the Appendix B.

First, these datasets will be split into a test and train set in the same way as the simulated data. This split will therefore also be 75% against 25%. Thereafter, the models are trained on

Dataset	Total observations	Minority samples	Percentage Imbalance
Adult	32561	7841	24,08%
Churn	10000	2037	20,37%
Shopping	12330	1908	15,47%
Fraud detection	307511	24825	8,07%
Hypothyroid	3163	151	4,77%
Kickstarter	378657	1844	0,49%

Table 2: Overview of the used datasets from the UCI database and Kaggle with total observations, number of minority samples and the percentage of imbalance in the relevant dataset.

the train set using 5-fold cross-validation. The same settings as for the data simulation are tested on the real-world datasets to compare these results. When missing values are present in these datasets, the knn-imputation method is used as mentioned in the previous section.

4.2.1 Adult

The *Adult* dataset is obtained from the UCI Machine Learning Repository. The goal in this dataset is to predict whether an individual earns more or less than $50,000 per year, hence the dependent variable is *Income*. This dataset contains 32561 observations of which 7841 actually earn more than this $50.000. This is therefore the minority sample. The dataset contains 14 explanatory variables, which especially contain information about education, demographics and socioeconomic status. The variables *education*, *relationship* and *native-country* are excluded from the research. The information of *education* and *relationship* are respectively already summarized in the categorical variables with the highest number of education (*educationnum*) and the *maritalstatus*. The attribute *nativecountry* has been omitted, because the distribution of the observations across the 41 levels is very unbalanced. The value *United-States*, which occurs 27491 times, is too dominantly present. The second largest value only contains 2%. Of the remaining eleven variables, five are discrete and six are continuous. There is one binomial variable and there are four nominal variables. Two variables contain missing values, namely *workclass* and *occupation*. The number of missing values are respectively 1857 and 1843 observations, where all the missing values for *occupation* are also missing for *workclass*.

4.2.2 Churn

The *Churn* dataset is retrieved from Kaggle and has exactly 10000 observations. In this dataset, the variable *Excited* is the dependent variable. This variable indicates whether a current employee is interested to stay with the value one, and whether this individual is not convinced (yet) with value zero. Only 20,37% of the individuals are willing to stay at the firm, which indicates the imbalance in target variable. It is interesting for a company to know which factors influence this decision. There are 14 variables, where the first three are left out. These variables contain row numbers, ID-numbers and surnames, which do not add any information to the model. Furthermore, the variables contain demographics and some financial information. These ten independent variables try to explain to dependent variable. There are three binominal variables, one nominal variable and six continuous variables. Finally, there are no missing observations in the Churn dataset.

4.2.3 Shopping

The *Shopping* dataset is obtained from the UCI Machine Learning Repository. This dataset has 12330 observations and 18 variables. The variable of interest is the variable *shopping*, which indicates whether a session at the company's website led to a purchase. When they did not buy anything, the dependent variable is assigned as zero. When they did actually buy something, the variable *shopping* has value one. Only in 15,47% of all the sessions a purchase decision has been made. The dataset is formed in a way that each observation represents a different user. It is interesting for a company to know which factors drove this customer to buy an item on their website. Therefore, this dataset considers 17 explanatory variables. All the variables are focused on data of the visit to the website. For example, it is known what kind of pages people have seen and for how long they visited those. Thereby, it is also known in what season an individual visited the website and whether there was a sale. There are ten continuous, one binomial and six categorical variables. Only the level names of the variables *month*, *visitortype* and *weekend* are known. For the other categorical variables, the difference between the assigned numbers to the levels suffice for fitting the model and classification. Finally, this dataset contains no missing observations.

4.2.4 Fraud Detection

The dataset *Fraud Detection* has 122 attributes and is retrieved from Kaggle. The goal for this dataset is to predict which loans could be fraudulent. The large number of attributes is mainly due to the fact that a lot of categorical variables are converted into dummies. These dummies are already represented or summarized in other variables and are therefore discarded. That leaves 16 explanatory variables to explain the target variable *fraud*. The value of this variable is based on a loan of an individual. When there is a reason to believe that this person is a credit card defaultees, *fraud* returns the value one. 24825 of the observations are marked as fraudulent, which comes down to 8,07%. A lot of financial companies (banks, insurance companies, loan providers), but also government agencies try to deal with potential fraudulent observations. Often only a small part of the observations is malicious. Therefore, these kinds of companies are highly interested in research on imbalanced data. This dataset contains four binomial variables, six categorical variables and six continuous variables, which describe the demographic characteristics and financial data of a particular individual. The total number of observations of this dataset is 307511. When the amount of data gives computational difficulties, one could consider to only use a share of the total data. It is important that this is interference is done randomly to maintain the correct proportion within the data. The variables *annuity, priceGoods, familystatus* and *familiymembers* have respectively 12, 278, 2 and 2 missing values. This is such a low percentage of the total number of observations, that these could be discarded. However, *suite* and *occupation* have respectively 1292 and 96391 missing values. Therefore, these values have to be imputed using knn-imputation.

4.2.5 Hypothyroid

The dataset *Hypothyroid* is one from the medical research field, which is obtained from the UCI Machine Learning Repository. This dataset focuses on predicting whether a patient could possibly suffer from hyperthyroidism, which indicates a condition where the thyroid glance has become too active [Streeten et al., 1988]. The target variable indicates with the value one that a patient actually has this disease, while zero rejects this diagnosis. This dataset has 3163 observations and 25 independent variables. Only 151 samples of the entire set came back positive and are therefore indicated with value one. This is only 4,77% of all the observations. Due to this imbalance, it is hard to correctly predict the positive cases, while this is of the utmost importance. The explanatory variables contain mostly medical indicators and some demographic characteristics. There are seven continuous variables and eighteen binomial variables. In this dataset, there are 5329 missing observations. These are distributed over the variables *age, sex, TSH, T3, TT4, T4U, FTI,* and *TBG,* which has respectively 446, 73, 468, 695, 249, 248, 247, and 2903 missing values. The values of *TBG* are for nearly 92% missing values. This attribute contains that little information, that it is left out in the analysis. The other missing values will also be imputed by the knn-imputation method.

4.2.6 Kickstarter

The last real-world dataset is the *Kickstarter* dataset from Kaggle. In this dataset, the focus is to classify the dependent variable *state* correctly as suspended. This means that the team behind the platform Kickstarter has chosen to suspend a fundraiser. Reasons for a suspension are violations of the company's rules regarding the project, such as misrepresentation or self-pledging. The dataset has 378657 observations, of which only 1844 are labelled as suspended. The imbalance is therefore piccaninny, which also makes this very interesting. By classification, while taking into account the imbalance in this dataset, one could make a severe impact on the results. There are 16 variables in this dataset, where 9 are left out. First, the *ID* and *Name* variables are left out, because they do not provide any additional information for classification. The *Category* and *Currency* variables are deleted considering their information is already (summarized) in respectively *Main Category* and *Country*. Then, *Deadline* and *launched* are combined in a new variable *Term* by taking the difference between them. Therefore, this new variable indicates how long the fundraiser should have been open. The information from both *Goal* and *Pledged* are represented in other variables that are also adjusted for the dollar. Finally, the last variable in the dataset is also left out, because it only contains missing values. The remaining 6 explanatory variables contain specific information about the project, the initiator and the supporters. There are two categorical and four continuous variables. Due to the large amount of data, one could encounter computational difficulties. Just like with the Fraud Detection dataset, one could choose to only use a share of the total data. It is important that this is interference is done randomly to maintain the correct proportion within the data. However, the number of factors is quite different between the two large datasets. It could be interesting to take a look at the computational speed between these two datasets to research the influence on the numbers of factors in the model. There are 7271 missing observations in this dataset, of which 4658 in *country*, 870 in *usd_pledged_real*, 870 in *usd_goal_real* and 873 in *term*. The latter three variables share the same observations with missing values as attribute *country*. Therefore, these observations are discarded from the dataset.

5 Results

In this section, the results of the previously discussed methods will be presented and discussed. First, the classification performance of the simulated data is shown, because this will indicate what can be expected from the methods with real-world datasets. Logically, these results will be presented thereafter. The focus lies on the differences between the performances of the adjusted SMOTE algorithm and the existing one of Chawla et al. However, it is also important to show that sampling actually improves the classification results. Therefore, the prediction accuracies without sampling and the corresponding differences in comparison with sampling techniques are looked into first. All the classification results of the simulated data and the real-world datasets can be found in Appendix C. In this section, however, mostly the differences between the classification accuracies of the various sampling methods will be shown.

5.1 Simulated datasets

5.1.1 Reference model: no sampling

First of all, the classification accuracies without oversampling and undersampling are investigated to take a look at the benchmark for the results on the simulated data. These are shown in table 3 below.

% Minority	KNN	Bagging	RF
1%	0,3462	0,5558	0,6259
5%	0,6124	0,5950	0,5623
10%	0,7178	0,7340	0,7182
15%	0,7106	0,7018	0,6987
20%	0,7274	0,7264	0,7069
25%	0,7565	0,7495	0,7478
30%	0,7680	0,7560	0,7314

(a) Model with 4 attributes

% Minority	KNN	Bagging	RF
1%	0,5940	0,7122	0,7846
5%	0,7146	0,6918	0,7283
10%	0,7821	0,7663	0,7889
15%	0,8061	0,8094	0,8238
20%	0,7846	0,7787	0,8060
25%	0,8151	0,8129	0,8252
30%	0,8291	0,8196	0,8320

(b) Model with 10 attributes

% Minority	KNN	Bagging	RF
1%	0,7306	0,7338	0,8065
5%	0,7541	0,7302	0,7754
10%	0,8313	0,8072	0,8409
15%	0,8227	0,8039	0,8296
20%	0,8513	0,8272	0,8609
25%	0,8686	0,8569	0,8853
30%	0,8624	0,8419	0,8749

(c) Model with 20 attributes

Table 3: Classification accuracy for simulated datasets with different levels of imbalance and attributes, *and* without sampling techniques

The first thing that stands out is that the accuracy clearly increases with the number of minority samples. Especially for the model with 4 attributes, the results are very poor with high levels of imbalance. When knn is used as classifier and only 1% minority samples are present in the dataset, the accuracy is even far worse than random allocation of the dependent variable which would approximately result in 50% accuracy. This already confirms the statement that regular classifiers have a hard time classifying observations when the data is strongly imbalanced. The biggest improvements in classification accuracies are made until roughly 10% of the observations are minority samples. Thereafter, a rise is still observable. However, the accuracy increases with a smaller rate when more positive samples exist in the data. This could indicate that especially the datasets with less than 10% minority sample can benefit from sampling methods. The predictions clearly tend to increase when more attributes are considered. Random forest clearly classifies the most accurate of the three classifiers for the models with 10 and 20 independent variables. For the models with 4 attributes, such a general conclusion is more difficult. First, Random Forrest returns the best results, thereafter bagging. However, knn seems to be the most accurate method when the percentage of minority samples is 15% or higher,

5.1.2 Adjusted SMOTE algorithm against reference model

The classification accuracies of the adjusted SMOTE algorithm for every setting can be found in the appendix. The focus here will be on the differences in comparison with the results from the reference model. This comparison is made for both oversampling with and without undersampling. First a small reminder: there are 450 different settings simulated, which are repeated 50 times to avoid coincidence in the results. When only oversampling is performed, the adjusted SMOTE algorithm provides a better classification accuracy in 396 of the 450 settings than the reference model. For the other 54 settings, the reference model actually predicts more accurate. When oversampling and undersampling are combined, the adjusted SMOTE algorithm even outperforms the reference model 428 out of 450 times. Therefore, there is enough evidence to state that the adjusted SMOTE algorithm significantly outperforms the tested classifiers without sampling. It is however more interesting to elaborate on these findings by investigating the various settings.

First of all, one could take a look at the used classifiers and the number of attributes included in the model. There are 54 settings for which the new SMOTE algorithm is outperformed by the reference model when only oversampling is used as sampling technique: 44 occur with the knn classifier, 3 with bagging, and 7 with random forest. In addition, 31 of these 54 inferior cases appear for the model with four attributes, 8 for the model with ten attributes, and 15 for the model with twenty attributes. When undersampling is added to the sampling process, knn, bagging, and random forest take on respectively 19, 3, and 10 of the 32 outperformed settings. The models with four, ten, and twenty attributes contribute respectively with 21, 6, and 5 cases. This could indicate that especially bagging and random forest are outstanding classifiers when a model without sampling is compared to models with sampling methods. In addition, one could also cautiously conclude that more attributes give the classification method the opportunity to classify more accurate when sampling is implemented.

In table 4 below, the average differences in classification accuracy between the reference model and the adjusted SMOTE algorithm for knn, bagging and random forest are shown for the models with four, ten and twenty attributes. Between brackets, behind the average accuracy, the percentage of datasets for which the reference model is outperformed by the adjusted SMOTE algorithm is given. The result that stands out the most is that each displayed average is greater than zero, regardless of one exception ($k = 1$, $oversampling = 1$, $undersampling = 0$, KNN, 20 $variables$). This confirms the previous conclusion that he adjusted SMOTE algorithm significantly outperforms the tested classifiers without sampling.

This table also allows us to take a better look at the changes in results when undersampling is added. Only for knn both the percentages and average accuracies improve most of the time. For both bagging and random forest, the average accuracy does not seem to decrease or increase systematically. The percentages mostly stay the same. Possibly due to the fact that it is 100% already most of the time. Thus, there is no unambiguous conclusion about these percentages. There is also no clear statement possible from this table on the increase of k or oversampling with the same value of k. When one takes a look at the results in the appendix, a potential explanation appears. The absolute classification accuracy increases in general when undersampling is added, more oversampling is used, and bagging or random forest are used instead of knn. This especially holds for the models with 1%, 5%, and 10% minority samples. However, this increase disappears when the averages of the datasets with different number of minority samples are shown. In addition, an increase in classification accuracy could be minor for the settings that are already superior in the reference model.

K = 1	**Undersampling = 0**			**Undersampling = 1**		
Oversampling = 1	KNN	Bagging	RF	KNN	Bagging	RF
4 variables	0,0530 (100%)	0,0452 (100%)	0,0410 (100%)	0,0762 (86%)	0,0490 (100%)	0,0342 (71%)
10 variables	0,0151 (57%)	0,0412 (100%)	0,0380 (100%)	0,0487 (86%)	0,0349 (86%)	0,0325 (100%)
20 variables	-0,0146 (43%)	0,0367 (100%)	0,0314 (100%)	0,0231 (100%)	0,0267 (86%)	0,0241 (86%)
K = 3	**Undersampling = 0**			**Undersampling = 1**		
Oversampling = 1	KNN	Bagging	RF	KNN	Bagging	RF
4 variables	0,0118 (14%)	0,0464 (100%)	0,0444 (100%)	0,0544 (71%)	0,0440 (100%)	0,0367 (100%)
10 variables	0,0210 (71%)	0,0374 (86%)	0,0360 (86%)	0,0590 (86%)	0,0401 (100%)	0,0339 (100%)
20 variables	0,0113 (71%)	0,0365 (100%)	0,0318 (100%)	0,0229 (86%)	0,0230 (100%)	0,0158 (86%)
Oversampling = 2	KNN	Bagging	RF	KNN	Bagging	RF
4 variables	0,0136 (33%)	0,0402 (100%)	0,0335 (100%)	0,0312 (50%)	0,0393 (100%)	0,0268 (83%)
10 variables	0,0256 (100%)	0,0378 (100%)	0,0376 (100%)	0,0549 (100%)	0,0435 (100%)	0,0310 (100%)
20 variables	0,0075 (83%)	0,0303 (100%)	0,0275 (100%)	0,0294 (100%)	0,0314 (100%)	0,0249 (100%)
Oversampling = 3	KNN	Bagging	RF	KNN	Bagging	RF
4 variables	0,0064 (40%)	0,0358 (100%)	0,0257 (80%)	0,0344 (60%)	0,0436 (100%)	0,0340 (80%)
10 variables	0,0229 (100%)	0,0319 (100%)	0,0326 (100%)	0,0481 (100%)	0,0400 (100%)	0,0245 (100%)
20 variables	0,0091 (60%)	0,0296 (100%)	0,0291 (100%)	0,0306 (100%)	0,0299 (100%)	0,0228 (100%)
K = 5	**Undersampling = 0**			**Undersampling = 1**		
Oversampling = 1	KNN	Bagging	RF	KNN	Bagging	RF
4 variables	0,0285 (43%)	0,0417 (100%)	0,0360 (86%)	0,0584 (71%)	0,0503 (100%)	0,0428 (100%)
10 variables	0,0408 (100%)	0,0561 (100%)	0,0492 (100%)	0,0547 (100%)	0,0393 (100%)	0,0265 (86%)
20 variables	0,0253 (100%)	0,0377 (86%)	0,0284 (86%)	0,0305 (100%)	0,0339 (100%)	0,0196 (100%)
Oversampling = 2	KNN	Bagging	RF	KNN	Bagging	RF
4 variables	0,0270 (33%)	0,0384 (100%)	0,0286 (83%)	0,0525 (50%)	0,0444 (100%)	0,0305 (100%)
10 variables	0,0418 (100%)	0,0468 (100%)	0,0443 (100%)	0,0619 (100%)	0,0443 (100%)	0,0280 (100%)
20 variables	0,0230 (100%)	0,0362 (100%)	0,0297 (100%)	0,0307 (100%)	0,0255 (83%)	0,0177 (100%)
Oversampling = 3	KNN	Bagging	RF	KNN	Bagging	RF
4 variables	0,0236 (40%)	0,0355 (100%)	0,0285 (80%)	0,0459 (60%)	0,0457 (100%)	0,0282 (80%)
10 variables	0,0289 (100%)	0,0413 (100%)	0,0349 (100%)	0,0577 (100%)	0,0410 (100%)	0,0231 (80%)
20 variables	0,0257 (80%)	0,0316 (100%)	0,0250 (100%)	0,0402 (100%)	0,0451 (100%)	0,0275 (100%)
Oversampling = 4	KNN	Bagging	RF	KNN	Bagging	RF
4 variables	0,0380 (50%)	0,0374 (100%)	0,0262 (100%)	0,0549 (75%)	0,0382 (100%)	0,0307 (100%)
10 variables	0,0263 (100%)	0,0305 (100%)	0,0183 (75%)	0,0573 (100%)	0,0329 (100%)	0,0197 (100%)
20 variables	0,0145 (75%)	0,0319 (100%)	0,0228 (100%)	0,0320 (100%)	0,0326 (100%)	0,0210 (100%)
Oversampling = 5	KNN	Bagging	RF	KNN	Bagging	RF
4 variables	0,0313 (67%)	0,0307 (100%)	0,0347 (100%)	0,0498 (100%)	0,0316 (100%)	0,0212 (100%)
10 variables	0,0345 (100%)	0,0295 (100%)	0,0287 (100%)	0,0472 (100%)	0,0216 (100%)	0,0139 (67%)
20 variables	0,0126 (67%)	0,0072 (67%)	0,0097 (100%)	0,0242 (100%)	0,0314 (100%)	0,0147 (100%)

Table 4: Average classification accuracy for knn, bagging and random forest for models with 4, 10 and 20 independent variables. Between brackets is the percentage of cases shown that the adjusted SMOTE algorithm returns better results than the reference model

5.1.3 Adjusted SMOTE algorithm against regular SMOTE

Now, the differences in classification accuracies between the Adjusted SMOTE algorithm and regular SMOTE are investigated. First, the focus is on all the data, after which the individual settings will be highlighted.

When only oversampling is used as sampling method, only 220 of the 450 settings show an increase in classification accuracy for the adjusted SMOTE algorithm in comparison with the regular one. When undersampling is added however, and the datasets are thus completely balanced, the results improve: 262 settings show an improvement over the current SMOTE method. This is already somewhat more promising. Therefore, an elaboration on the different settings by taking a look at the different levels of *k* will be presented below.

K = 1, Oversampling = 1							
	Undersampling = 0				**Undersampling = 1**		
4 variables	KNN	Bagging	RF	4 variables	KNN	Bagging	RF
1% minority	0,0400	0,0440	0,0592	1% minority	0,0232	0,0464	-0,0275
5% minority	-0,0451	-0,0175	-0,0194	5% minority	0,0312	0,0159	0,0262
10% minority	0,0288	0,0321	0,0273	10% minority	-0,0173	-0,0024	-0,0040
15% minority	0,0068	0,0129	0,0104	15% minority	0,0223	0,0048	0,0048
20% minority	-0,0001	0,0109	-0,0073	20% minority	0,0016	-0,0008	0,0002
25% minority	0,0052	0,0072	0,0079	25% minority	0,0003	0,0023	0,0032
30% minority	-0,0056	0,0011	-0,0023	30% minority	-0,0177	-0,0124	-0,0160
10 variables	KNN	Bagging	RF	10 variables	KNN	Bagging	RF
1% minority	0,0265	-0,0041	0,0013	1% minority	-0,0149	0,0182	0,0078
5% minority	-0,0780	-0,0015	-0,0135	5% minority	0,0008	-0,0098	0,0127
10% minority	-0,0291	-0,0166	-0,0068	10% minority	-0,0181	-0,0188	-0,0222
15% minority	-0,0131	0,0020	0,0008	15% minority	-0,0220	0,0017	-0,0026
20% minority	-0,0203	0,0141	0,0170	20% minority	-0,0106	0,0065	0,0116
25% minority	-0,0224	-0,0113	-0,0078	25% minority	-0,0029	0,0038	0,0105
30% minority	-0,0082	0,0168	0,0123	30% minority	-0,0168	-0,0091	0,0011
20 variables	KNN	Bagging	RF	20 variables	KNN	Bagging	RF
1% minority	0,0125	0,0588	0,0174	1% minority	0,0201	-0,0482	0,0112
5% minority	-0,0386	-0,0250	-0,0285	5% minority	0,0222	0,0109	0,0060
10% minority	-0,0235	0,0034	0,0028	10% minority	-0,0025	0,0138	0,0121
15% minority	-0,0286	0,0035	-0,0075	15% minority	-0,0231	0,0075	0,0039
20% minority	-0,0235	0,0034	0,0028	20% minority	-0,0025	0,0138	0,0121
25% minority	-0,0133	0,0001	0,0030	25% minority	-0,0038	-0,0005	0,0039
30% minority	-0,0166	-0,0002	-0,0024	30% minority	-0,0073	-0,0092	-0,0096

Table 5: Difference in classification accuracy between adjusted SMOTE and regular SMOTE algorithms for K = 1 and corresponding oversampling

The differences in classification accuracies between the adjusted and current SMOTE algorithms, for all settings considering one nearest neighbour, are shown in table 5 above. The green highlighted cells shown the settings for which the adjusted SMOTE method returns better results. The first thing that stands out is that knn shows a lot of non-green cells in comparison with bagging and random forest, for both sampling with and without undersampling. Especially for the dataset with more explanatory variables and lower imbalance, knn seems to prefer the current SMOTE algorithm. On the contrary, both bagging and random forest actually seem to improve with the adjusted method. Another important observation is that these differences, for all three classifiers, tend to decrease when the number of minority samples increases. This could indicate that the adjusted SMOTE algorithm performs especially better than regular SMOTE when a higher level of imbalance is present in the data.

K = 3, Oversampling = 1							
	Undersampling = 0				**Undersampling = 1**		
4 variables	KNN	Bagging	RF	4 variables	KNN	Bagging	RF
1% minority	0,0170	0,0279	0,0485	1% minority	0,0411	0,1052	0,0663
5% minority	-0,0452	-0,0066	0,0289	5% minority	0,0030	-0,0060	-0,0046
10% minority	-0,0439	0,0046	0,0019	10% minority	-0,0144	-0,0068	-0,0078
15% minority	-0,0333	-0,0034	0,0010	15% minority	-0,0017	0,0064	0,0090
20% minority	-0,0323	-0,0075	-0,0057	20% minority	-0,0004	0,0064	0,0061
25% minority	-0,0170	-0,0011	0,0111	25% minority	-0,0341	0,0019	0,0089
30% minority	-0,0329	0,0048	-0,0006	30% minority	-0,0243	-0,0008	-0,0035
10 variables	KNN	Bagging	RF	10 variables	KNN	Bagging	RF
1% minority	-0,0770	0,0026	-0,0056	1% minority	0,0172	-0,0757	-0,0083
5% minority	0,0249	0,0105	0,0191	5% minority	-0,0368	-0,0011	0,0207
10% minority	-0,0092	-0,0140	-0,0035	10% minority	0,0103	0,0154	0,0262
15% minority	-0,0174	-0,0098	-0,0040	15% minority	-0,0066	-0,0032	0,0028
20% minority	-0,0148	0,0016	0,0172	20% minority	0,0107	0,0082	0,0293
25% minority	-0,0147	-0,0058	-0,0025	25% minority	-0,0120	0,0081	0,0066
30% minority	-0,0188	-0,0056	0,0033	30% minority	-0,0120	-0,0022	0,0004
20 variables	KNN	Bagging	RF	20 variables	KNN	Bagging	RF
1% minority	-0,0453	-0,0372	0,0147	1% minority	-0,0532	-0,0034	0,0008
5% minority	-0,0193	0,0012	0,0035	5% minority	-0,0176	-0,0441	-0,0139
10% minority	-0,0200	-0,0018	0,0000	10% minority	0,0370	-0,0280	0,0120
15% minority	0,0126	0,0125	0,0128	15% minority	-0,0188	0,0112	0,0107
20% minority	-0,0200	-0,0018	0,0000	20% minority	-0,0041	0,0090	0,0033
25% minority	-0,0082	-0,0049	-0,0051	25% minority	-0,0056	0,0117	0,0044
30% minority	-0,0134	0,0052	-0,0004	30% minority	-0,0193	0,0055	0,0012
K = 3, Oversampling = 2							
	Undersampling = 0				**Undersampling = 1**		
4 variables	KNN	Bagging	RF	4 variables	KNN	Bagging	RF
1% minority	0,0057	-0,0300	-0,0064	1% minority	0,0028	0,0101	0,0286
5% minority	-0,1135	0,0061	-0,0014	5% minority	-0,0383	0,0009	-0,0081
10% minority	0,0027	0,0036	0,0116	10% minority	0,0163	0,0231	0,0264
15% minority	-0,0317	0,0025	0,0104	15% minority	-0,0363	0,0017	-0,0037
20% minority	-0,0752	0,0042	0,0162	20% minority	-0,0701	-0,0024	0,0006
25% minority	-0,0247	-0,0012	0,0075	25% minority	-0,0359	-0,0086	0,0107
10 variables	KNN	Bagging	RF	10 variables	KNN	Bagging	RF
1% minority	0,0483	-0,0189	-0,0018	1% minority	0,0001	0,0145	0,0252
5% minority	-0,0444	0,0218	0,0396	5% minority	-0,0055	-0,0074	0,0295
10% minority	-0,0170	0,0012	0,0119	10% minority	-0,0013	-0,0030	0,0040
15% minority	-0,0181	-0,0101	-0,0001	15% minority	0,0055	0,0087	0,0168
20% minority	-0,0307	-0,0037	0,0036	20% minority	-0,0260	-0,0001	0,0194
25% minority	-0,0096	0,0251	0,0190	25% minority	-0,0110	0,0104	0,0095
20 variables	KNN	Bagging	RF	20 variables	KNN	Bagging	RF
1% minority	-0,0014	0,0232	0,0337	1% minority	-0,0280	0,0128	-0,0002
5% minority	-0,0155	-0,0146	-0,0122	5% minority	0,0048	0,0142	0,0254
10% minority	0,0008	-0,0053	0,0019	10% minority	-0,0034	0,0080	0,0108
15% minority	-0,0113	0,0091	0,0150	15% minority	-0,0068	0,0062	0,0068
20% minority	0,0008	-0,0053	0,0019	20% minority	-0,0034	0,0080	0,0108
25% minority	-0,0011	0,0082	0,0093	25% minority	-0,0011	0,0061	0,0070

Table 6: Difference in classification accuracy between adjusted SMOTE and regular SMOTE algorithms for K = 3 and oversampling = 1, 2

K = 3, Oversampling = 3							
	Undersampling = 0				**Undersampling = 1**		
4 variables	KNN	Bagging	RF	4 variables	KNN	Bagging	RF
1% minority	-0,0262	0,0069	0,0092	1% minority	0,0175	0,0180	0,0642
5% minority	-0,0518	0,0121	0,0067	5% minority	-0,0197	0,0017	0,0050
10% minority	-0,0065	0,0195	0,0187	10% minority	-0,0358	-0,0217	-0,0217
15% minority	-0,0466	-0,0117	-0,0186	15% minority	-0,0365	0,0089	0,0154
20% minority	-0,0741	0,0097	0,0164	20% minority	-0,0697	0,0043	0,0026
10 variables	KNN	Bagging	RF	10 variables	KNN	Bagging	RF
1% minority	-0,0243	-0,1008	-0,0370	1% minority	-0,0652	-0,0365	-0,0182
5% minority	-0,0318	-0,0035	-0,0072	5% minority	-0,0068	-0,0132	0,0015
10% minority	-0,0290	0,0239	0,0244	10% minority	-0,0014	0,0423	0,0317
15% minority	-0,0163	0,0016	-0,0036	15% minority	-0,0271	-0,0073	-0,0111
20% minority	-0,0279	0,0090	0,0243	20% minority	-0,0252	0,0128	0,0322
20 variables	KNN	Bagging	RF	20 variables	KNN	Bagging	RF
1% minority	-0,0187	0,0617	0,0105	1% minority	0,0488	0,0477	0,0520
5% minority	-0,0408	-0,0150	-0,0129	5% minority	0,0088	0,0157	0,0082
10% minority	-0,0047	-0,0036	0,0035	10% minority	-0,0040	0,0077	0,0135
15% minority	-0,0081	0,0052	0,0102	15% minority	-0,0113	-0,0026	-0,0048
20% minority	-0,0047	-0,0036	0,0035	20% minority	-0,0040	0,0077	0,0135

Table 7: Difference in classification accuracy between adjusted SMOTE and regular SMOTE algorithms for K = 3 and oversampling = 3

Tables 6 and 7 above show the differences between the two considered SMOTE algorithms when three nearest neighbours are considered. The datasets with a high percentage of minority samples are left out when the oversampling causes the minority sample to outgrow the majority sample, otherwise the majority sample should also be oversampled. As mentioned in the methodology section, it is not considered relevant for this research to also oversample the majority sample. It is therefore omitted from this research.

The adjusted SMOTE algorithm again does not seems to be very effective for knn. However, it is effective for both bagging and random forest. These improvements are even stronger when the number of attributes increase and undersampling is added to the sampling techniques. There is no unambiguous picture on these results when the oversampling or the number of considered nearest neighbours grow. In addition, the differences seem to decrease when the amount of imbalance decreases, especially with random forest. The conclusion from earlier, that the adjusted SMOTE algorithm seems to perform better than regular SMOTE when a higher level of imbalance is present in the data, is accordingly confirmed

Finally, the tables for $K = 5$ and the corresponding oversampling values are shown in the tables 8 and 9 on the subsequent pages. A lot of conclusions that were stated before can be confirmed from these results. knn seems to be outperformed by both bagging and random forest, while adding undersampling has a frequently positive influence on the differences. Regular SMOTE actually seems to outperform the adjusted algorithm in case of knn, where the opposite holds for both bagging and random forest. The models with four attributes tend to return inferior results in comparison with the models that include more independent variables, while there is no clear picture on a discrepancy between the two latter ones. In addition, there does not seem to be any relationship between the increase in oversampling and the number of considered nearest neighbours to the performance comparison of both algorithms.

K = 5, Oversampling = 1							
	Undersampling = 0				**Undersampling = 1**		
4 variables	KNN	Bagging	RF	4 variables	KNN	Bagging	RF
1% minority	0,0042	0,0311	0,0447	1% minority	0,0257	0,0126	0,0346
5% minority	-0,0199	-0,0244	-0,0193	5% minority	-0,0350	-0,0049	-0,0155
10% minority	-0,0292	-0,0147	-0,0181	10% minority	0,0020	0,0097	0,0215
15% minority	-0,0289	0,0082	0,0022	15% minority	0,0008	0,0037	0,0084
20% minority	-0,0327	-0,0001	-0,0023	20% minority	0,0041	0,0052	0,0090
25% minority	-0,0132	-0,0033	0,0010	25% minority	-0,0095	0,0019	0,0035
30% minority	-0,0241	0,0011	0,0072	30% minority	-0,0067	0,0063	0,0118
10 variables	KNN	Bagging	RF	10 variables	KNN	Bagging	RF
1% minority	0,0645	0,0337	0,0402	1% minority	-0,0270	-0,0265	0,0145
5% minority	-0,0354	0,0354	0,0325	5% minority	-0,0214	0,0232	0,0187
10% minority	0,0009	0,0057	0,0092	10% minority	0,0089	0,0020	0,0071
15% minority	0,0044	0,0258	0,0202	15% minority	0,0182	0,0187	0,0246
20% minority	-0,0037	0,0177	0,0285	20% minority	-0,0192	-0,0031	0,0033
25% minority	0,0087	0,0138	0,0156	25% minority	-0,0152	0,0062	0,0042
30% minority	-0,0052	0,0071	0,0045	30% minority	-0,0089	0,0058	0,0049
20 variables	KNN	Bagging	RF	20 variables	KNN	Bagging	RF
1% minority	0,0823	0,1045	0,0398	1% minority	0,0146	0,0451	-0,0095
5% minority	0,0163	0,0276	0,0094	5% minority	-0,0073	-0,0109	-0,0060
10% minority	-0,0154	-0,0066	-0,0063	10% minority	-0,0077	-0,0051	0,0084
15% minority	-0,0118	-0,0074	-0,0119	15% minority	-0,0122	-0,0073	-0,0018
20% minority	-0,0154	-0,0066	-0,0063	20% minority	-0,0077	-0,0051	0,0084
25% minority	-0,0103	0,0149	0,0113	25% minority	-0,0067	0,0054	0,0122
30% minority	0,0006	-0,0210	-0,0149	30% minority	0,0055	0,0127	0,0110
K = 5, Oversampling = 2							
	Undersampling = 0				**Undersampling = 1**		
4 variables	KNN	Bagging	RF	4 variables	KNN	Bagging	RF
1% minority	0,0680	-0,0101	-0,0208	1% minority	0,0738	0,0053	0,0137
5% minority	-0,0436	-0,0004	-0,0010	5% minority	-0,0135	-0,0013	-0,0089
10% minority	-0,0327	-0,0079	-0,0059	10% minority	-0,0244	0,0149	0,0186
15% minority	-0,0483	0,0040	-0,0069	15% minority	-0,7267	-0,0045	0,0131
20% minority	-0,0441	-0,0072	0,0008	20% minority	-0,7411	0,0011	0,0125
25% minority	-0,0261	0,0088	0,0157	25% minority	-0,0349	0,0091	0,0130
10 variables	KNN	Bagging	RF	10 variables	KNN	Bagging	RF
1% minority	0,0657	0,0496	0,0246	1% minority	0,0287	0,0214	0,0106
5% minority	0,0039	0,0281	0,0321	5% minority	0,0105	-0,0005	-0,0064
10% minority	-0,0030	0,0109	0,0069	10% minority	-0,0044	0,0135	0,0101
15% minority	0,0075	0,0213	0,0187	15% minority	0,0113	0,0094	0,0138
20% minority	-0,0253	-0,0047	0,0148	20% minority	-0,0192	0,0043	0,0202
25% minority	-0,0035	0,0144	0,0220	25% minority	-0,0114	0,0017	0,0046
20 variables	KNN	Bagging	RF	20 variables	KNN	Bagging	RF
1% minority	-0,0168	0,0211	-0,0346	1% minority	0,0599	0,0176	0,0143
5% minority	-0,0324	-0,0049	0,0087	5% minority	-0,0308	0,0038	0,0067
10% minority	-0,0066	0,0132	0,0071	10% minority	-0,0079	-0,0011	0,0034
15% minority	-0,0094	-0,0010	0,0050	15% minority	-0,0046	-0,0119	-0,0061
20% minority	-0,0066	0,0132	0,0071	20% minority	-0,0079	-0,0011	0,0034
25% minority	-0,0003	0,0133	0,0096	25% minority	-0,0019	-0,0079	-0,0015

Table 8: Difference in classification accuracy between adjusted SMOTE and regular SMOTE algorithms for K = 5 and oversampling = 2

K = 5, Oversampling = 3							
	Undersampling = 0				**Undersampling = 1**		
4 variables	KNN	Bagging	RF	4 variables	KNN	Bagging	RF
1% minority	-0,0323	0,0170	0,0063	1% minority	0,1125	0,0360	0,0593
5% minority	-0,0390	-0,0180	-0,0397	5% minority	-0,0030	0,0133	0,0075
10% minority	-0,0309	-0,0271	-0,0190	10% minority	-0,0138	0,0117	0,0103
15% minority	-0,0430	-0,0013	-0,0007	15% minority	-0,0513	-0,0075	-0,0086
20% minority	-0,0657	-0,0059	0,0033	20% minority	-0,0534	0,0237	0,0222
10 variables	KNN	Bagging	RF	10 variables	KNN	Bagging	RF
1% minority	-0,0928	-0,0441	-0,0324	1% minority	-0,0293	-0,0709	-0,0804
5% minority	0,0131	0,0494	0,0385	5% minority	-0,0022	-0,0052	0,0091
10% minority	-0,0110	0,0100	0,0139	10% minority	0,0066	0,0014	0,0101
15% minority	-0,0076	0,0070	0,0022	15% minority	-0,0044	0,0125	0,0152
20% minority	-0,0212	0,0088	0,0269	20% minority	-0,0166	0,0138	0,0300
20 variables	KNN	Bagging	RF	20 variables	KNN	Bagging	RF
1% minority	0,0227	0,0126	-0,0320	1% minority	0,0513	0,0907	0,0347
5% minority	0,0026	0,0254	0,0317	5% minority	-0,0048	0,0498	0,0500
10% minority	0,0099	0,0017	0,0124	10% minority	-0,0056	0,0153	0,0143
15% minority	-0,0112	-0,0068	-0,0018	15% minority	0,0000	-0,0019	0,0045
20% minority	0,0099	0,0017	0,0124	20% minority	-0,0056	0,0153	0,0143
K = 5, Oversampling = 4							
	Undersampling = 0				**Undersampling = 1**		
4 variables	KNN	Bagging	RF	4 variables	KNN	Bagging	RF
1% minority	0,0132	0,0718	0,0070	1% minority	0,0515	0,0393	0,0367
5% minority	-0,0425	-0,0245	-0,0264	5% minority	-0,0395	-0,0205	-0,0072
10% minority	-0,0130	-0,0114	-0,0044	10% minority	0,0059	0,0168	0,0125
15% minority	-0,0417	0,0064	-0,0038	15% minority	-0,0310	-0,0095	-0,0021
10 variables	KNN	Bagging	RF	10 variables	KNN	Bagging	RF
1% minority	-0,0253	0,0026	-0,0452	1% minority	0,0374	-0,0009	0,0411
5% minority	-0,0054	0,0236	0,0301	5% minority	0,0059	0,0302	0,0403
10% minority	-0,0276	-0,0061	-0,0028	10% minority	0,0012	0,0101	0,0035
15% minority	0,0061	0,0148	0,0221	15% minority	-0,0018	0,0160	0,0147
20 variables	KNN	Bagging	RF	20 variables	KNN	Bagging	RF
1% minority	0,0373	-0,0183	0,0040	1% minority	0,0045	0,0067	0,0008
5% minority	-0,0079	-0,0202	-0,0021	5% minority	0,0043	0,0391	0,0436
10% minority	0,0099	0,0136	0,0088	10% minority	0,0090	0,0122	0,0121
15% minority	-0,0132	0,0117	0,0084	15% minority	0,0031	0,0077	0,0171
K = 5, Oversampling = 5							
	Undersampling = 0				**Undersampling = 1**		
4 variables	KNN	Bagging	RF	4 variables	KNN	Bagging	RF
1% minority	-0,0524	-0,0279	0,0343	1% minority	-0,0274	-0,0546	-0,0792
5% minority	-0,0757	0,0201	0,0157	5% minority	-0,0610	-0,0209	-0,0173
10% minority	0,0012	-0,0099	-0,0088	10% minority	-0,0289	-0,0118	-0,0106
10 variables	KNN	Bagging	RF	10 variables	KNN	Bagging	RF
1% minority	-0,0859	0,0144	0,0002	1% minority	-0,0341	0,0106	0,0196
5% minority	0,0349	0,0533	0,0478	5% minority	0,0121	0,0224	0,0209
10% minority	0,0039	-0,0029	0,0238	10% minority	0,0006	-0,0057	0,0211
20 variables	KNN	Bagging	RF	20 variables	KNN	Bagging	RF
1% minority	-0,0334	-0,0675	-0,0472	1% minority	-0,0134	0,0445	0,0178
5% minority	-0,0299	-0,0108	0,0017	5% minority	-0,0016	0,0202	0,0030
10% minority	-0,0131	-0,0360	0,0065	10% minority	-0,0010	0,0190	0,0026

Table 9: Difference in classification accuracy between adjusted SMOTE and regular SMOTE algorithms for K = 5 and oversampling = 5

Clearly, the classifiers knn, bagging, and random forest all perform differently. The results in this section already showed that the results for knn are quite bad: when there is no undersampling used, in only 25,33% of all cases the adjusted algorithm outperforms the existing one. If oversampling is added, this increases to 32,67%, which still indicates a significant underperformance of the new algorithm. These results improve when the bagging classifier is used. because then respectively 56,67% and 63,33% of all settings show an improvement in the results for the adjusted SMOTE algorithm in comparison with regular SMOTE algorithm, with and without undersampling. Through a simple proportion test one can find the respective p-values of 0,0516 and 0,0006. Therefore, it has been shown that the adjusted SMOTE algorithm with a bagging classifier significantly outperforms the SMOTE methods from Chawla et al. in case the datasets are completely balanced after sampling. This however does not hold when there is no undersampling. The results are than only significantly better for significant levels higher than 5%. When random forest is considered as classifier, the results become even better. In case only oversampling is used, already 97 of the total 150 settings show an improvement in classification accuracy for the adjusted SMOTE algorithm. For a combination of oversampling and undersampling, this even becomes 116 of the 150 cases. Therefore, it is easy to conclude that for random forest the adjusted SMOTE algorithm significantly performs better than the current SMOTE method, with or without undersampling! There are no remarkable differences notable when the available nearest neighbours or the oversampling increases. However, it has been mentioned before that the number of variables could play their part in this research. When undersampling is left out of the equation, the models with four, ten, and twenty attributes show respectively 43,44%, 54,67%, and 48,67% settings for which the adjusted SMOTE algorithm performance better than the existing one. In case undersampling is added to the sampling techniques, these percentages become respectively 55,33%, 60,00%, and 59,33%. This does not give any hard evidence that the adjusted model outperforms the regular one when more attributes are added, however it neither rules it out.

5.2 Real-world datasets

5.2.1 Reference model: no sampling

Now, it is time to take a look at real-world datasets to test the findings from the simulated data. Below, in table 10, an overview of the real-world datasets and the corresponding classification accuracies is presented. The first things that catches the eye is that there is no longer a clear improvement in accuracy visible when the number of minority samples increases. Especially the datasets Kickstarter and Fraud return dramatic results for all the classifiers. This could indicate that these datasets do not follow a clear underlying model, causing the signal-to-noise ratio to be disturbed strongly. In addition, the accuracies for all three classifiers are extremely high for the Hyper dataset. It will be interesting to see whether sampling could still improve these results. Finally, knn is eminently the worst classifier of the three considered, just as with the simulated data.

Dataset	% Minority	Attributes	KNN	Bagging	Random Forest
Kickstarter	0,49%	6	0,4759	0,5852	0,5934
Hyper	4,77%	25	0,9117	0,9196	0,9305
Fraud	8,07%	16	0,4995	0,5365	0,5333
Shopping	15,47%	17	0,6377	0,8516	0,8466
Churn	20,37%	10	0,5260	0,7877	0,8232
Adult	24,08%	11	0,5579	0,8641	0,8841

Table 10: Classification accuracy for real-world datasets with different levels of imbalance, attributes ànd *without* sampling techniques

5.2.2 Adjusted SMOTE algorithm against reference model

In this section, the focus will be on the difference between the reference model and the adjusted SMOTE algorithm to check whether sampling indeed improves classification results. In the tables 11 and 12 on the next pages, these differences are shown for all the tested settings. Just like with the simulated data, a green cell indicates that the classification result with sampling is higher than for the reference model. Based on those tables, it can be easily concluded that the adjusted SMOTE technique classifies more accurate than the reference model without sampling. This holds for sampling with and without undersampling. When undersampling is not being used, a whopping 95,56% of all cases return a higher accuracy for the adjusted SMOTE algorithm. In case undersampling is included, this percentage even becomes 97,04%. Again, the improvement in classification accuracy seems to be higher with knn. This is probably due to the fact that the initial classification of knn is by far the lowest.

In addition, there is nothing remarkable notable for the datasets Kickstarter and Fraud that seemed to lack a proper underlying model. The accuracy improves due to sampling. However, it is neither significantly bigger nor smaller than for other datasets. Finally, no unequivocal conclusion can be drawn from the results when oversampling or the number of available nearest neighbours available increases.

K = 1, Oversampling = 1							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0988	0,0470	0,0226	Kickstarter	0,1020	-0,0060	0,0014
Hyper	0,0651	0,0591	0,0624	Hyper	0,0687	0,0696	0,0633
Fraud	0,0505	0,0265	0,0270	Fraud	0,0474	0,0266	0,0320
Shopping	0,0800	0,0239	0,0382	Shopping	0,0346	0,0387	0,0423
Churn	0,0429	0,0369	0,0283	Churn	0,0416	0,0405	0,0231
Adult	0,0247	0,0167	0,0177	Adult	0,0083	0,0094	0,0138
K = 3, Oversampling = 1							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0090	0,0074	0,0065	Kickstarter	0,0813	0,0270	0,0162
Hyper	0,0516	0,0614	0,0599	Hyper	0,0689	0,0704	0,0611
Fraud	0,0615	0,0448	0,0469	Fraud	0,0534	0,0341	0,0344
Shopping	0,0798	0,0220	0,0249	Shopping	0,0544	0,0335	0,0368
Churn	0,0307	0,0396	0,0283	Churn	0,0376	0,0452	0,0269
Adult	0,0228	0,0114	0,0087	Adult	0,0088	0,0038	0,0057
K = 3, Oversampling = 2							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0307	0,0035	0,0011	Kickstarter	0,0940	0,0111	0,0036
Hyper	0,0575	0,0645	0,0628	Hyper	0,0693	0,0697	0,0617
Fraud	0,0567	0,0492	0,0296	Fraud	0,0564	0,0252	0,0320
Shopping	0,1007	0,0337	0,0387	Shopping	0,0683	0,0428	0,0449
Churn	0,0375	0,0382	0,0255	Churn	0,0394	0,0412	0,0241
Adult	0,0237	0,0070	0,0051	Adult	0,0180	0,0042	0,0066
K = 3, Oversampling = 3							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0495	-0,0107	0,0037	Kickstarter	0,1399	-0,0136	0,0053
Hyper	0,0609	0,0692	0,0624	Hyper	0,0720	0,0670	0,0605
Fraud	0,0431	0,0394	0,0415	Fraud	0,0627	0,0286	0,0377
Shopping	0,0940	0,0406	0,0441	Shopping	0,0844	0,0387	0,0416

Table 11: Difference in classification accuracy between reference model and adjusted SMOTE for K = 1 and 3 and adequate oversampling levels

K = 5, Oversampling = 1							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0823	0,0504	0,0032	Kickstarter	0,1037	0,0247	0,0131
Hyper	0,0603	0,0577	0,0617	Hyper	0,0711	0,0713	0,0618
Fraud	0,0492	0,0345	0,0265	Fraud	0,0434	0,0214	0,0363
Shopping	0,0774	0,0267	0,0345	Shopping	0,0410	0,0329	0,0366
Churn	0,0382	0,0369	0,0263	Churn	0,0371	0,0436	0,0227
Adult	0,0201	0,0115	0,0070	Adult	0,0114	0,0173	0,0161
K = 5, Oversampling = 2							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0730	0,0422	-0,0044	Kickstarter	0,1120	0,0367	0,0622
Hyper	0,0617	0,0628	0,0601	Hyper	0,0720	0,0712	0,0619
Fraud	0,0437	0,0425	0,0511	Fraud	0,0637	0,0333	0,0380
Shopping	0,0993	0,0339	0,0387	Shopping	0,0611	0,0389	0,0414
Churn	0,0371	0,0416	0,0260	Churn	0,0337	0,0409	0,0250
Adult	0,0243	0,0050	0,0057	Adult	0,0259	0,0010	0,0015
K = 5, Oversampling = 3							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0398	0,0206	-0,0025	Kickstarter	0,1392	0,0326	0,0488
Hyper	0,0531	0,0568	0,0603	Hyper	0,0725	0,0708	0,0623
Fraud	0,0500	0,0415	0,0535	Fraud	0,0609	0,0279	0,0380
Shopping	0,0927	-0,0652	0,0380	Shopping	0,0845	0,0401	0,0453
K = 5, Oversampling = 4							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0250	-0,0134	-0,0158	Kickstarter	0,0742	-0,0108	-0,0285
Hyper	0,0622	0,0679	0,0643	Hyper	0,0738	0,0683	0,0639
Fraud	0,0524	0,0144	0,0326	Fraud	0,0462	0,0173	0,0302
Shopping	0,0886	0,0204	0,0297	Shopping	0,1055	0,0411	0,0413
K = 5, Oversampling = 5							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0923	0,0343	0,0458	Kickstarter	0,1268	0,0099	0,0376
Hyper	0,0592	0,0620	0,0614	Hyper	0,0716	0,0691	0,0602
Fraud	0,0507	0,0021	0,0265	Fraud	0,0422	0,0417	0,0385

Table 12: Difference in classification accuracy between reference model and adjusted SMOTE for K = 5 and adequate oversampling levels

5.2.3 Adjusted SMOTE algorithm against regular SMOTE

In the tables 13 and 14 below, the differences between the adjusted SMOTE algorithm and the existing one are presented. First, the focus is on the general picture for the different classifiers. For both sampling with and without undersampling, all three classifiers are used for 45 different settings. Without undersampling, the knn, bagging, and random forest classifiers return a higher accuracy in respectively 77,78%, 62,22%, and 53,33% of all settings for the adjusted SMOTE algorithm. Only the performance of knn is significant when a simple proportion test is carried out with a significance level of 5%. This is quite the opposite in comparison with the findings from the simulation study. When undersampling is added as sampling technique, these percentages change to respectively 53,33%, 66,67% and 57,77%. It is remarkable that there is such a big drop for knn, which makes these results more similar to those from the data simulation. However, it is significant for bagging only, again through a simple proportion test with 5% as significance level.

K = 1, Oversampling = 1							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0319	0,0314	-0,0308	Kickstarter	-0,0243	-0,0284	0,0210
Hyper	0,0027	-0,0004	0,0016	Hyper	-0,0067	-0,0024	0,0011
Fraud	0,0037	0,0128	-0,0048	Fraud	-0,0086	0,0148	0,0144
Shopping	-0,0022	-0,0028	0,0021	Shopping	0,0052	0,0039	0,0061
Churn	0,0045	0,0034	0,0052	Churn	-0,0053	-0,0024	-0,0029
Adult	0,0054	0,0061	0,0064	Adult	0,0054	0,0038	0,0039
K = 3, Oversampling = 1							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	-0,0120	0,0034	0,0238	Kickstarter	-0,0155	0,0199	-0,0245
Hyper	0,0068	0,0020	-0,0003	Hyper	-0,0019	-0,0020	-0,0022
Fraud	0,0129	0,0070	0,0095	Fraud	0,0034	0,0119	0,0086
Shopping	0,0089	-0,0128	-0,0132	Shopping	-0,0107	-0,0057	-0,0041
Churn	0,0256	0,0056	-0,0011	Churn	0,0271	0,0033	-0,0010
Adult	0,0014	-0,0011	0,0001	Adult	-0,0234	-0,0044	-0,0105
K = 3, Oversampling = 2							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0042	-0,0044	-0,0151	Kickstarter	-0,0068	-0,0057	-0,0283
Hyper	0,0017	-0,0044	-0,0010	Hyper	-0,0025	-0,0027	-0,0001
Fraud	0,0109	0,0307	0,0058	Fraud	0,0223	0,0245	0,0110
Shopping	0,0119	0,0024	0,0003	Shopping	-0,0009	0,0093	0,0049
Churn	0,0341	0,0089	0,0027	Churn	0,0362	0,0084	-0,0014
Adult	-0,0064	0,0052	-0,0003	Adult	-0,0114	0,0010	0,0018
K = 3, Oversampling = 3							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	-0,0037	-0,0574	-0,0207	Kickstarter	0,0405	-0,0336	-0,0224
Hyper	-0,0063	0,0001	-0,0016	Hyper	0,0003	-0,0025	-0,0016
Fraud	-0,0052	0,0203	0,0072	Fraud	0,0139	0,0247	0,0236
Shopping	0,0046	0,0069	0,0008	Shopping	0,0142	0,0109	0,0039

Table 13: Difference in classification accuracy between adjusted SMOTE and regular SMOTE algorithms for K = 1 and 3 and adequate oversampling levels

K = 5, Oversampling = 1							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0421	0,0227	-0,0165	Kickstarter	-0,0139	-0,0304	-0,0212
Hyper	0,0073	-0,0039	-0,0004	Hyper	-0,0026	-0,0011	-0,0006
Fraud	0,0072	0,0000	-0,0109	Fraud	-0,0128	-0,0085	0,0018
Shopping	0,0205	-0,0001	0,0023	Shopping	-0,0094	-0,0040	-0,0022
Churn	0,0274	-0,0001	-0,0018	Churn	0,0215	0,0024	-0,0057
Adult	-0,0037	-0,0007	-0,0091	Adult	-0,0087	0,0124	0,0035
K = 5, Oversampling = 2							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	-0,0383	0,0513	-0,0448	Kickstarter	-0,0061	0,0571	0,0696
Hyper	0,0051	-0,0021	-0,0034	Hyper	0,0009	0,0011	0,0019
Fraud	-0,0085	0,0329	0,0156	Fraud	0,0394	0,0171	0,0215
Shopping	0,0220	0,0099	0,0028	Shopping	0,0055	0,0110	0,0044
Churn	0,0312	0,0075	0,0016	Churn	0,0287	0,0090	-0,0009
Adult	-0,0032	0,0029	0,0026	Adult	0,0029	0,0019	-0,0007
K = 5, Oversampling = 3							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0147	-0,0095	-0,0355	Kickstarter	0,0197	0,0095	0,0503
Hyper	0,0004	-0,0018	-0,0012	Hyper	0,0026	0,0012	0,0010
Fraud	0,0083	0,0358	0,0172	Fraud	0,0179	0,0192	0,0171
Shopping	0,0054	-0,0898	0,0065	Shopping	0,0052	0,0102	0,0101
K = 5, Oversampling = 4							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0132	0,0050	0,0010	Kickstarter	-0,0093	-0,0293	-0,0630
Hyper	0,0033	-0,0005	0,0002	Hyper	0,0039	0,0005	0,0028
Fraud	0,0068	0,0162	0,0058	Fraud	-0,0013	0,0199	0,0268
Shopping	0,0010	0,0013	-0,0051	Shopping	0,0244	0,0147	0,0061
K = 5, Oversampling = 5							
	Undersampling = 0				**Undersampling = 1**		
	KNN	Bagging	RF		KNN	Bagging	RF
Kickstarter	0,0153	0,0237	0,0221	Kickstarter	0,0018	0,0085	0,0176
Hyper	0,0061	0,0057	0,0012	Hyper	0,0017	0,0032	0,0023
Fraud	0,0002	0,0045	-0,0044	Fraud	-0,0066	0,0562	-0,0014

Table 14: Difference in classification accuracy between adjusted SMOTE and regular SMOTE algorithms for K = 5 and adequate oversampling levels

The reason for these outcomes could possibly be found in the tables above. Here, the first thing that stands out is that the results for $K = 1$, $K = 3$ are contradictory with the observations from the data simulation. For these values of k, the classification accuracy seems to decrease when undersampling is added as sampling technique in comparison with merely oversampling. This is an unexpected observation, since one would expect better results when the datasets are more balanced. However, this result diminishes when the level of oversampling increases, or when five nearest neighbours are available. This could actually be the conclusion that could not be drawn before from the results of the data simulation: it seems that an increase in the available number of neighbours, or oversampling, returns a higher classification accuracy for the adjusted SMOTE algorithm in comparison with the existing one. The closest neighbours in real-world datasets perhaps contain more information about the near points than with data simulation, from which the new adjusted SMOTE algorithm could benefit. This conclusion however must be drawn with caution, because it cannot be shown to be significant from these results. However, it could be interesting for future research.

6 Conclusion and discussion

Imbalanced data is a troublesome problem that occur across a lot of research areas. Various sampling methods are contrived through the years, where SMOTE is a widely appreciated technique. It uses nearest neighbours to synthetically create new minority samples, however only continuous variables are used to determine this. Distances between discrete variables are hard to include, because this is often not measurable. An appropriate method to cope with this problem could increase the prediction accuracy of classifiers in imbalanced data and thereby contribute to this issue in the appurtenant literature. The main question for this research was therefore: How does a SMOTE algorithm, adjusted for continuous and discrete variables, affect the classification performance for several classifiers?

To answer this question, the existing SMOTE algorithm had to be adjusted. Chawla et al. already proposed a possible direction in their research. They proposed to add the median of the standard deviations of the continuous variables to the total distance when the values of two observations differ for a particular discrete variable. This is not exactly the proper method, because the median of the standard deviations could dominate the total distance when it is relatively large. Therefore, this research proposes to use the SHEOM distance measure. This method actually also uses the median of the standard deviations of the continuous variables, but it scales them first. Then, the nearest neighbours of the minority observations can be determined based on the new total distances. The parameter values of the discrete attributes for the synthetic minority samples depend on the ratio of the possible outcomes among the nearest neighbours. This new approach of oversampling, combined with undersampling, should give a more accurate classification. Both an analysis on simulated data and real-world data were performed to test this hypothesis, using knn, bagging and random forest as classifiers.

The data simulation has shown that without sampling techniques the classification accuracy increases when the imbalance in the data decreases, more attributes are incorporated in the model, and bagging or random forest are used as classifiers. The latter indicates that a more refined classification technique improves the results for imbalanced data. The adjusted SMOTE algorithm clearly improves these results significantly, both with solely oversampling and with a combination of oversampling and undersampling. This holds for all three classifiers, but in a lesser extent for knn. However, knn appears to benefit the most from the addition of undersampling as sampling technique. Simpler classifiers may therefore need stronger sampling techniques to undergo the same improvements. There is also a positive relation visible between the number of explanatory variables and gain in classification accuracy, which indicates that classification accuracy improves more when extensively specified models are regarded. In addition, according to the results from the data simulation, the adjusted SMOTE algorithm outperforms regular smote. This especially holds for the classifiers bagging and random forest, models with more attributes, and a combination of oversampling and undersampling. The knn classifier, however, seems to prefer the regular SMOTE. The difference between these two SMOTE methods decreases when the imbalance in the data reduces. Finally, there is no clear relationship between the number of considered nearest neighbours or the amount of oversampling with an improvement of the classification accuracy.

The classification results from the real-world datasets without sampling techniques did not confirm the hypothesis from the data simulation that less imbalance ensures higher classification accuracy. This is presumably caused by the unequal number of attributes among the datasets, or just poor defined models. However, bagging and random forest still return better results than knn. The adjusted SMOTE technique significantly improves these results, both with and without undersampling. For the real-world datasets however, the largest gain is noted by the knn classifier. This indicates that especially poor classifiers could really benefit from sampling techniques. The improved results are neither better nor worse for datasets which initially returned a lower accuracy. When the adjusted SMOTE algorithm is compared with the regular SMOTE algorithm, the most settings show improved results. Without undersampling however, this is only significant for knn. This result is unexpected, because it is against the expectations from the results from the data simulation. It changes when undersampling is added. Then, only the bagging classifier shows significant improvement. When one or three nearest neighbours are considered, the results are counterintuitive. This is because undersampling seems to decrease the classification accuracy. However, it declines when oversampling or the number of considered nearest neighbours increases. This could therefore indicate that the results improve for the adjusted SMOTE technique when more nearest neighbours or higher levels of oversampling are considered.

In short, some of the results from the data simulation are confirmed by the results from the real-world datasets. However, it is important to note that this is not true for all conclusions. This is mainly because the data simulation is based on a prescribed model. It is however evident that the SMOTE algorithm, adjusted for discrete variables, improves the classification accuracy in comparison with the results from methods without sampling for both simulated data and real-world datasets. In addition, it also returns higher classification accuracies than regular SMOTE. From the data simulation, it appears that this especially holds for the more comprehensive classifiers bagging and random forest, models with more attributes, and for datasets that are more balanced after sampling. This improvement fades away when the datasets initially have less imbalance. It seems that strongly imbalanced datasets prefer the adjusted SMOTE technique. For real-world datasets, these relations were less clear. However, it appeared that they emerged again when the number of considered nearest neighbours and the levels of oversampling increase.

This research mainly focused on adjusting sampling techniques for discrete variables. It proved that better results can be retrieved by the newly proposed techniques. However, the addition is to the basic technique that is already developed further in various directions. It would therefore by interesting to research this extension on other sampling modifications, like Borderline-SMOTE, Safe-level-SMOTE, or G-SMOTE. In addition, this technique uses the initial ratios of the values from the nearest neighbours to assign a value to the discrete parameter. An attribution based on the majority vote of the values of the considered parameter could possibly improve the results even more. Thereby, one could also choose to focus on the nearest neighbours in ascending order of distance. Finally, the classifiers in this research were similar to each other. This way, it was possible to draw conclusions based on the complexity of the classifier. In some future research, one could try to confirm the found results on less decision tree driven techniques to test if it also works for a wider range of classifiers.

References

Alan C Acock. Working with missing values. *Journal of Marriage and Family*, 67(4):1012–1028, 2005.

Murray Aitkin. The analysis of unbalanced cross-classifications. *Journal of the Royal Statistical Society: Series A (General)*, 141(2):195–211, 1978.

Ricardo Barandela, Rosa M Valdovinos, J Salvador Sánchez, and Francesc J Ferri. The imbalanced training sample problem: under or over sampling? In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 806–814. Springer, 2004.

Sukarna Barua, Md Monirul Islam, Xin Yao, and Kazuyuki Murase. Mwmote–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):405–425, 2012.

Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1):20–29, 2004.

Mariana Belgiu and Lucian Drăguţ. Random forest in remote sensing: a review of applications and future directions. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114:24–31, 2016.

Reshma C Bhagat and Sachin S Patil. Enhanced smote algorithm for classification of imbalanced big-data using random forest. In *2015 IEEE International Advance Computing Conference (IACC)*, pages 403–408. IEEE, 2015.

Battista Biggio and Fabio Roli. Wild patterns: ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

Daniel Blanco-Melo, Benjamin E Nilsson-Payant, Wen-Chun Liu, Skyler Uhl, Daisy Hoagland, Rasmus Moller, Tristan X Jordan, Kohei Oishi, Maryline Panis, David Sachs, et al. Imbalanced host response to sars-cov-2 drives development of covid-19. *Cell*, 181(5):1036–1045, 2020.

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.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

Leo Breiman. Random forests. *UC Berkeley TR567*, 1999.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC Press, 1984.

Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 475–482. Springer, 2009.

Jonathan Burez and Dirk Van den Poel. Handling class imbalance in customer churn prediction. *Expert Systems with Applications*, 36(3):4626–4636, 2009.

Francisco J Castellanos, Jose J Valero-Mas, Jorge Calvo-Zaragoza, and Juan R Rico-Juan. Oversampling imbalanced data in the string space. *Pattern Recognition Letters*, 103:32–38, 2018.

Nitesh V Chawla. Data mining for imbalanced datasets: an overview. *Data Mining and Knowledge Discovery Handbook*, pages 875–886, 2009.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16: 321–357, 2002.

Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 107–119. Springer, 2003.

Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations Newsletter*, 6(1):1–6, 2004.

William W Cohen. Fast effective rule induction. In *Machine Learning Proceedings 1995*, pages 115–123. Elsevier, 1995.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine learning*, pages 233–240, 2006.

Georgios Douzas and Fernando Bacao. Geometric smote: rffective oversampling for imbalanced learning through a geometric extension of smote. *ArXiv Preprint ArXiv:1709.07377*, 2017.

Todd Eavis and Nathalie Japkowicz. A recognition-based alternative to discrimination-based multi-layer perceptrons. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 280–292. Springer, 2000.

Bradley Efron. *The jackknife, the bootstrap and other resampling plans.* SIAM, 1982.

Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap.* CRC Press, 1994.

Dina Elreedy and Amir F Atiya. A comprehensive analysis of synthetic minority oversampling technique (smote) for handling class imbalance. *Information Sciences*, 505:32–64, 2019.

Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

Alberto Fernandez, Salvador Garcia, Francisco Herrera, and Nitesh V Chawla. Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of Artificial Intelligence Research*, 61:863–905, 2018.

Jerome H Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, pages 1–67, 1991.

Andrew Greasley. *Simulating business processes for descriptive, predictive, and prescriptive analytics.* Walter de Gruyter GmbH & Co KG, 2019.

Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 986–996. Springer, 2003.

Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing*, pages 878–887. Springer, 2005.

Amira Kamil Ibrahim Hassan and Ajith Abraham. Modeling insurance fraud detection using imbalanced data classification. In *Advances in Nature and Biologically Inspired Computing*, pages 117–127. Springer, 2016.

Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328. IEEE, 2008.

Shengguo Hu, Yanfeng Liang, Lintao Ma, and Ying He. Msmote: Improving classification performance when training data is imbalanced. In *2009 Second International Workshop on Computer Science and Engineering*, volume 2, pages 13–17. IEEE, 2009.

Tim Hwang. Computational power and the social impact of artificial intelligence. *Available at SSRN 3147971*, 2018.

Nathalie Japkowicz, Catherine Myers, Mark Gluck, et al. A novelty detection approach to classification. In *IJCAI*, volume 1, pages 518–523. Citeseer, 1995.

Nathalie Japkowicz et al. Learning from imbalanced data sets: a comparison of various strategies. In *AAAI workshop on learning from imbalanced data sets*, volume 68, pages 10–15. AAAI Press Menlo Park, CA, 2000.

Taeho Jo and Nathalie Japkowicz. Class imbalances versus small disjuncts. *ACM Sigkdd Explorations Newsletter*, 6(1):40–49, 2004.

Justin Matthew Johnson. An evaluation of deep learning with class imbalanced big data. *Monthly Weather Review*, 78(1):1–3, 2019.

Mohammed Khalilia, Sounak Chakraborty, and Mihail Popescu. Predicting disease risks from highly imbalanced data using random forest. *BMC Medical Informatics and Decision Making*, 11(1):1–13, 2011.

Miroslav Kubat, Stan Matwin, et al. Addressing the curse of imbalanced training sets: one-sided selection. In *ICML*, volume 97, pages 179–186. Citeseer, 1997.

Max Kuhn et al. Building predictive models in r using the caret package. *J Stat Softw*, 28(5): 1–26, 2008.

Eduardo Laber and Lucas Murtinho. Minimization of gini impurity: Np-completeness and approximation algorithm via connections with the k-means problem. *Electronic Notes in Theoretical Computer Science*, 346:567–576, 2019.

Aurélie Lemmens and Christophe Croux. Bagging and boosting classification trees to predict churn. *Journal of Marketing Research*, 43(2):276–286, 2006.

Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

Julian Luengo, Alberto Fernandez, Salvador Garcia, and Francisco Herrera. Addressing data complexity for imbalanced data sets: analysis of smote-based oversampling and evolutionary undersampling. *Soft Computing*, 15(10):1909–1936, 2011.

Nicola Lunardon, Giovanna Menardi, and Nicola Torelli. Rose: a package for binary imbalanced learning. *R Journal*, 6(1), 2014.

Inderjeet Mani and I Zhang. knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of Workshop on Learning from Imbalanced Datasets*, volume 126. ICML United States, 2003.

Giovanna Menardi and Nicola Torelli. Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery*, 28(1):92–122, 2014.

Rupert G Miller. A trustworthy jackknife. *The Annals of Mathematical Statistics*, 35(4):1594–1605, 1964.

Thais Mayumi Oshiro, Pedro Santoro Perez, and José Augusto Baranauskas. How many trees in a random forest? In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 154–168. Springer, 2012.

Mahesh Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, 2005.

Chao-Ying Joanne Peng, Kuk Lida Lee, and Gary M Ingersoll. An introduction to logistic regression analysis and reporting. *The Journal of Educational Research*, 96(1):3–14, 2002.

Anantha M Prasad, Louis R Iverson, and Andy Liaw. Newer classification and regression tree techniques: bagging and random forests for ecological prediction. *Ecosystems*, 9(2): 181–199, 2006.

Foster Provost. Machine learning from imbalanced data sets 101. In *Proceedings of the AAAI'2000 Workshop on Imbalanced Data Sets*, volume 68, pages 1–3. AAAI Press, 2000.

Foster Provost and Tom Fawcett. Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. 1997.

Foster Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.

J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 1992.

Enislay Ramentol, Yailé Caballero, Rafael Bello, and Francisco Herrera. Smote-rs b*: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using smote and rough sets theory. *Knowledge and Information Systems*, 33(2): 245–265, 2012.

Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, volume 3, pages 41–46, 2001.

Jose A Saez, Julian Luengo, Jerzy Stefanowski, and Francisco Herrera. Smote-ipf: addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Information Sciences*, 291:184–203, 2015.

V Vijaya Saradhi and Girish Keshav Palshikar. Employee churn prediction. *Expert Systems with Applications*, 38(3):1999–2006, 2011.

Ruth G Shaw and Thomas Mitchell-Olds. Anova for unbalanced data: an overview. *Ecology*, 74(6):1638–1645, 1993.

Mayuri S Shelke, Prashant R Deshmukh, and Vijaya K Shandilya. A review on imbalanced data handling using undersampling and oversampling technique. *International Journal of Recent Trends in Engineering and Research*, 3(4):444–449, 2017.

Yang Song, Jian Huang, Ding Zhou, Hongyuan Zha, and C Lee Giles. Iknn: informative k-nearest neighbor pattern classification. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 248–264. Springer, 2007.

Matthew S Spencer, Samantha C Bates Prins, Margaret S Beckom, et al. Heterogeneous distance measures and nearest-neighbor classification in an ecological setting. *Missouri Journal of Mathematical Sciences*, 22(2):108–123, 2010.

DH Streeten, GUNNAR H Anderson Jr, TIMOTHY Howland, RICHARD Chiang, and HAROLD Smulyan. Effects of thyroid function on blood pressure. recognition of hypothyroid hypertension. *Hypertension*, 11(1):78–83, 1988.

Carolin Strobl, James Malley, and Gerhard Tutz. An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological Methods*, 14(4):323, 2009.

Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. Classification of imbalanced data: a review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04): 687–719, 2009.

Clifton D Sutton. Classification and regression trees, bagging, and boosting. *Handbook of Statistics*, 24:303–329, 2005.

Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

Luis Torgo. An infra-structure for performance estimation and experimental comparison of predictive models in r. *ArXiv Preprint ArXiv:1412.0436*, 2014.

Luis Torgo, Rita P Ribeiro, Bernhard Pfahringer, and Paula Branco. Smote for regression. In *Portuguese Conference on Artificial Intelligence*, pages 378–389. Springer, 2013.

Fredy Rodriguez Torres, Jesus A Carrasco-Ochoa, and Jose Fco Martinez-Trinidad. Smote-d a deterministic version of smote. In *Mexican Conference on Pattern Recognition*, pages 177–188. Springer, 2016.

Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

John W Tukey. Data analysis and behavioral science. *Unpublished manuscript*, 1962.

Sofia Visa and Anca Ralescu. Issues in mining imbalanced data sets-a review paper. In *Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference*, volume 2005, pages 67–73. SN, 2005.

Boyu Wang and Joelle Pineau. Online bagging and boosting for imbalanced data streams. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3353–3366, 2016.

Wei Wei, Jinjiu Li, Longbing Cao, Yuming Ou, and Jiahang Chen. Effective detection of sophisticated online banking fraud on extremely imbalanced data. *World Wide Web*, 16(4): 449–475, 2013.

D Randall Wilson and Tony R Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.

Chong Zhang, Kay Chen Tan, and Ruoxu Ren. Training cost-sensitive deep belief networks on imbalance data problems. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 4362–4367. IEEE, 2016.

Shichao Zhang, Xuelong Li, Ming Zong, Xiaofeng Zhu, and Debo Cheng. Learning k for knn classification. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(3):1–19, 2017a.

Shichao Zhang, Xuelong Li, Ming Zong, Xiaofeng Zhu, and Ruili Wang. Efficient knn classification with different numbers of nearest neighbors. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1774–1785, 2017b.

Appendices

A Adjusted SMOTE algorithm

```
#### Base code SMOTE
#### Author: Merijn van der Put, 376686
#### Erasmus School of Economics
#### Erasmus University Rotterdam
### Input:
## data: Dataset with the minority class samples, where the first column
contains the target variable (NO DEFAULT)
## oversampling: How many times has the minority sample to be oversampled
(DEFAULT is 2)
## undersampling: What ratio of the minority sample has the majority sample
to be? (DEFAULT is 1) (1 means the same number of observations, 0,5 means
half the observations, 2 means twice the observations)
## k: number of nearest neighbours to be used (DEFAULT is 5)
### output:
## Synthetic generated minority samples

SmotePut <- function(data, oversampling = 1, undersampling = 1, k = 5) {
  #Call the necessarily packages and define some starting variables
  set.seed(376686)
  library(MASS)
  library(dplyr)
  library(caret)
  library(pROC)
  library(performanceEstimation)
  names = names(data)
  split = split(data, data[,1], drop = FALSE)
  if (nrow(split[[1]]) > nrow(split[[2]])) {
    majority = as.data.frame(split[1])
    minority = as.data.frame(split[2])
  } else {
    majority = as.data.frame(split[2])
    minority = as.data.frame(split[1])
  }
```

```
colnames(minority) = names
colnames(majority) = names
T = nrow(minority); R = nrow(majority); N = oversampling;
us = undersampling
attributes = ncol(minority)

##### UNDERSAMPLING MAJORITY SET
if (undersampling == 0) {
  print("Warning:␣The␣majority␣sample␣is␣not␣undersampled")
} else {
undersampling = T / R * us
R = undersampling * R * (oversampling + 1)
majority = majority[sample(nrow(majority), size = R, replace = FALSE),]
}

##### OVERSAMPLING MINORITY SET
##If oversampling is lower than 1 (undersampling the minority sample)
if (N < 1){
  T = (N/1)*T; N = 1
  minority = minority[sample(nrow(minority), T),]
  print("Warning:␣You␣undersampled␣the␣minority␣sample")
} else {
##Oversampling is higher than 1 (oversampling the minority case)

#Synthetic minority samples are created on one of the line segments,
#wherefore oversampling can't be larger than k
  if (k < oversampling) {
    print("Error:␣number␣is␣nearest␣neighbours␣is␣not␣sufficient
␣␣␣␣␣␣for␣this␣amount␣of␣oversampling")
  } else {

  #Calculate distances between all samples
  N = (T * N)
  distance <- function(minority, T, attributes) {
    distance = matrix(0, nrow = T, ncol = T)
    stdevs = vector(length = 0); j = 1
    #Standarizing the continuous values
    for (i in 2:attributes) {
      if (!is.integer(minority[,i])) {
        mean <- mean(minority[,i]); sd <- sd(minority[,i])
        minority[,i] <- (minority[,i] - mean) / sd
        j = j + 1
        }
    }

    #Calculate the distances
    for (i in 1:T) {
      for (j in 2:attributes) {
        if (is.integer(minority[,j])) {
          for(l in 1:T){
            if (minority[i,j] != minority[l,j]) {
              distance[i,l] <- distance[i,l] + 1 #SD is one!
            }
          }
        } else if (is.numeric(minority[,j])) {
          distance[i,] <- distance[i,] + abs(minority[,j] - minority[i,j])
        }
        j <- j + 1
      }
      i <- i + 1
    }
    return(distance)
  }

  #Chose the k nearest neighbours for matrixnn and save in matrixnn
  matrixnn <- function(distance, minority, k, T) {
    matrixnn = matrix(0, nrow = nrow(minority), ncol = k)
    for (i in 1:T) {
      dis <- distance[-i,i]
      for (kk in 1:k) {
        if (matrixnn[i,kk] == 0) {
          min_index <- which(distance[i,] == min(dis))
          if (length(min_index) == 1) {
```

```
          matrixnn[i,kk] <- min_index
          dis <- dis[-which.min(dis)]
        } else {
          for (l in 1:length(min_index)) {
            if (min_index[l] == i) {
              min_index = min_index[-l]
              break
            }
          }
          j <- 1
          while (j + kk - 1 <= k && j <= length(min_index)) {
            matrixnn[i,kk+j-1] <- min_index[j]
            dis <- dis[-which.min(dis)]
            j <- j + 1
          }
        }
      }
    }
  }
  return(matrixnn)
}

#Create the synthetic samples based on the sample and their
#nearest neighbours
## - for continuous variables: Use original SMOTE procedure
## - for nominal and binominal variables: Assign randomly value based on
## the probability the different values occur in the nearest neighbours
## Save new synthetic minority samples in new table
synthetic <- function(minority, matrixnn, T, N, attributes) {
  synthetic = data.frame(matrix(0, nrow = N, ncol = attributes))
  i = 1
  for (index in 1:T) {
    nns = matrixnn[index,]
    nns <- as.numeric(sample(nns, oversampling))
    for(os in 1:oversampling) {
      nn <- nns[os]
      difference <- abs(minority[index,] - minority[nn,])
```

```
          pmin <- pmin(minority[index,], minority[nn,])
          for (j in 1:attributes) {
            if (is.integer(minority[,j])) {
              value = vector("integer", k+1)
              value[1] = minority[index,j]
              for (l in 1:k) {
                nr <- matrixnn[index,l]
                value[l+1] = minority[nr,j]
              }
              synthetic[i,j] <- sample(value, 1)
            } else if (is.numeric(minority[,j])){
              random <- runif(1)
              synthetic[i,j] <- pmin[j] + (random * difference[j])
            }
          }
          i = i + 1
        }
      }
      colnames(synthetic) = names
      synthetic = rbind(minority, synthetic)
      return(synthetic)
    }
    distance <- distance(minority, T, attributes)
    matrixnn <- matrixnn(distance, minority, k, T)
    minority <- synthetic(minority, matrixnn, T, N, attributes)
  }
 }

  synthetic = rbind(minority, majority)
  return(synthetic)
}
```

B Datasets

B.1 Adult

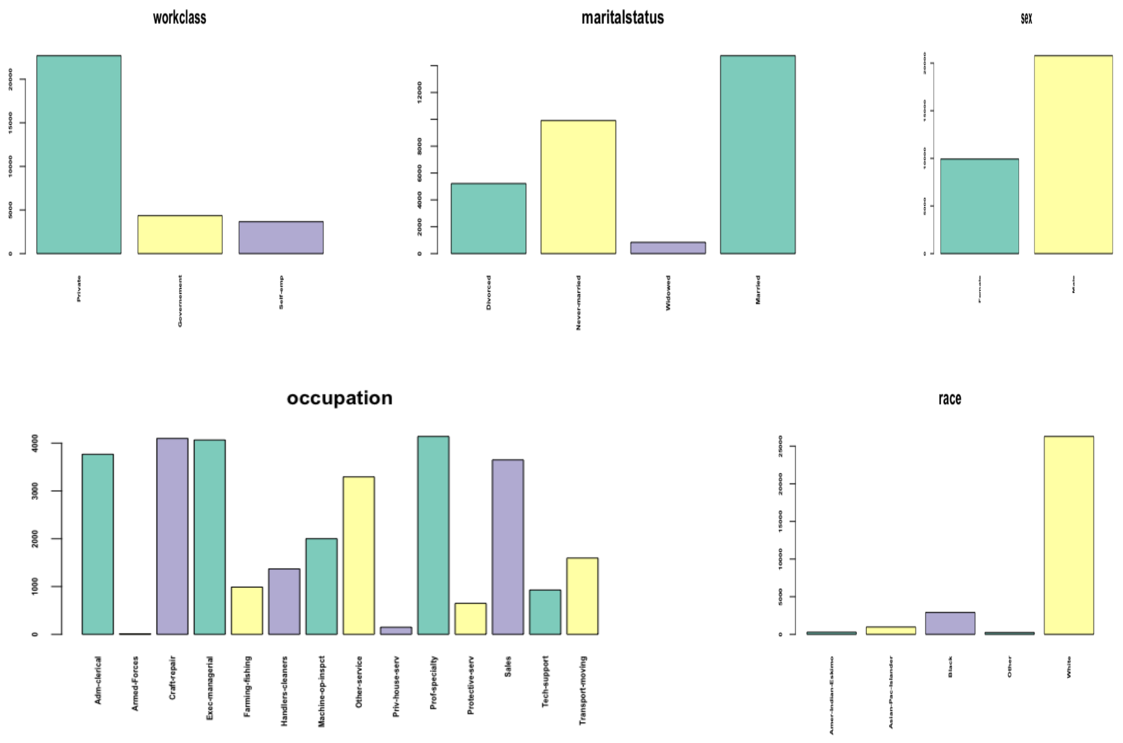

Figure 3: Discrete variables of the Adult dataset with the frequency of their levels

From left to right, the levels of the variables are:

1. Workclass: Private, Governement, Self employed
2. Maritalstatus: Divorced, Never married, Widowed, Married
3. Sex: Female, Male
4. Occupation: Administrative clerical, Armed Forces, Craft Repair, Executive Managerial, Farming - Fishing, Handlers - Cleaners, Machine Inspection, Other Services, Private House Service, Professional Specialty, Protective Services, Sales. Technical Support, Transport - Moving
5. race: American-Indian-Eskimo, Asian-Pacific,Islander, Black, Other, White

	Age	FNL weight	Education	Capital Gain	Capital Loss	Hours per week
Average	38.44	189853	10.13	1106.32	88.95	40.95
Minimum	17	13769	1	0	0	1
Median	37	178534	10	0	0	40
Maximum	90	1484705	16	99999	4356	99
St. dev.	13.11	105467.14	2.56	7499.51	405.75	11.98

Table 15: Overview of the continuous variables of the Adult dataset

B.2 Churn

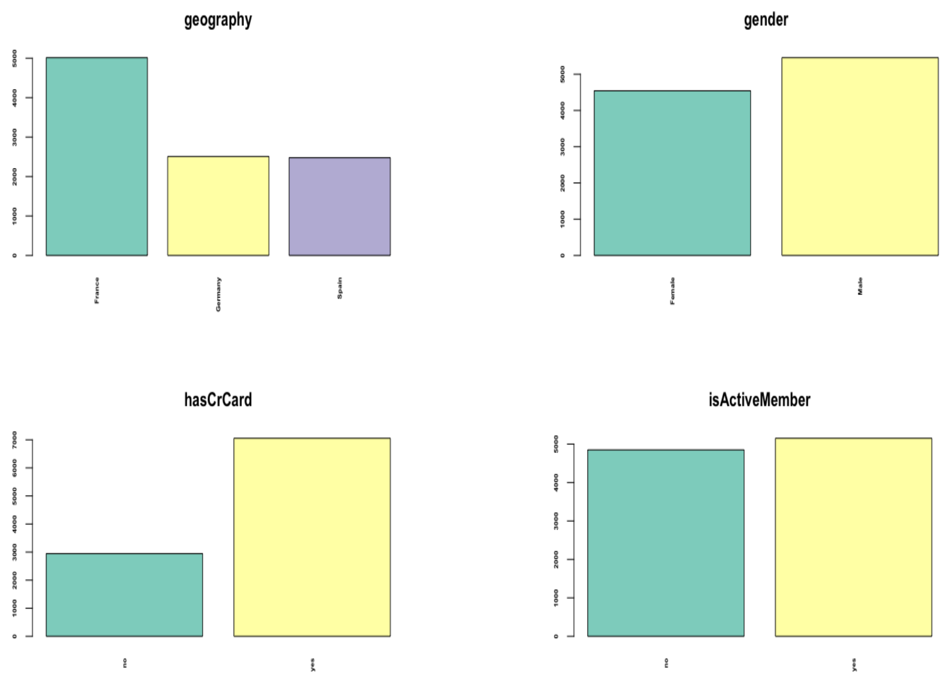

Figure 4: Discrete variables of the Churn dataset with the frequency of their levels

From left to right, the levels of the variables are:

1. Geography: France, Germany, Spain
2. Gender: Female, Male
3. HasCrCard: no, yes
4. IsActiveMember: no, yes

	CreditScore	Age	Tenure	Balance	NumOfProducts	EstimatedSalary
Average	650.53	38.92	5.01	76485.89	1.53	100090.24
Minimum	350	18	0	0	1	11.58
Median	652	37	5	97198.54	1	100193.91
Maximum	850	92	10	250898.09	4	199992.48
St. dev.	96.65	10.49	2.89	62397.41	0.58	57510.49

Table 16: Overview of the continuous variables of the Churn dataset

B.3 Shopping

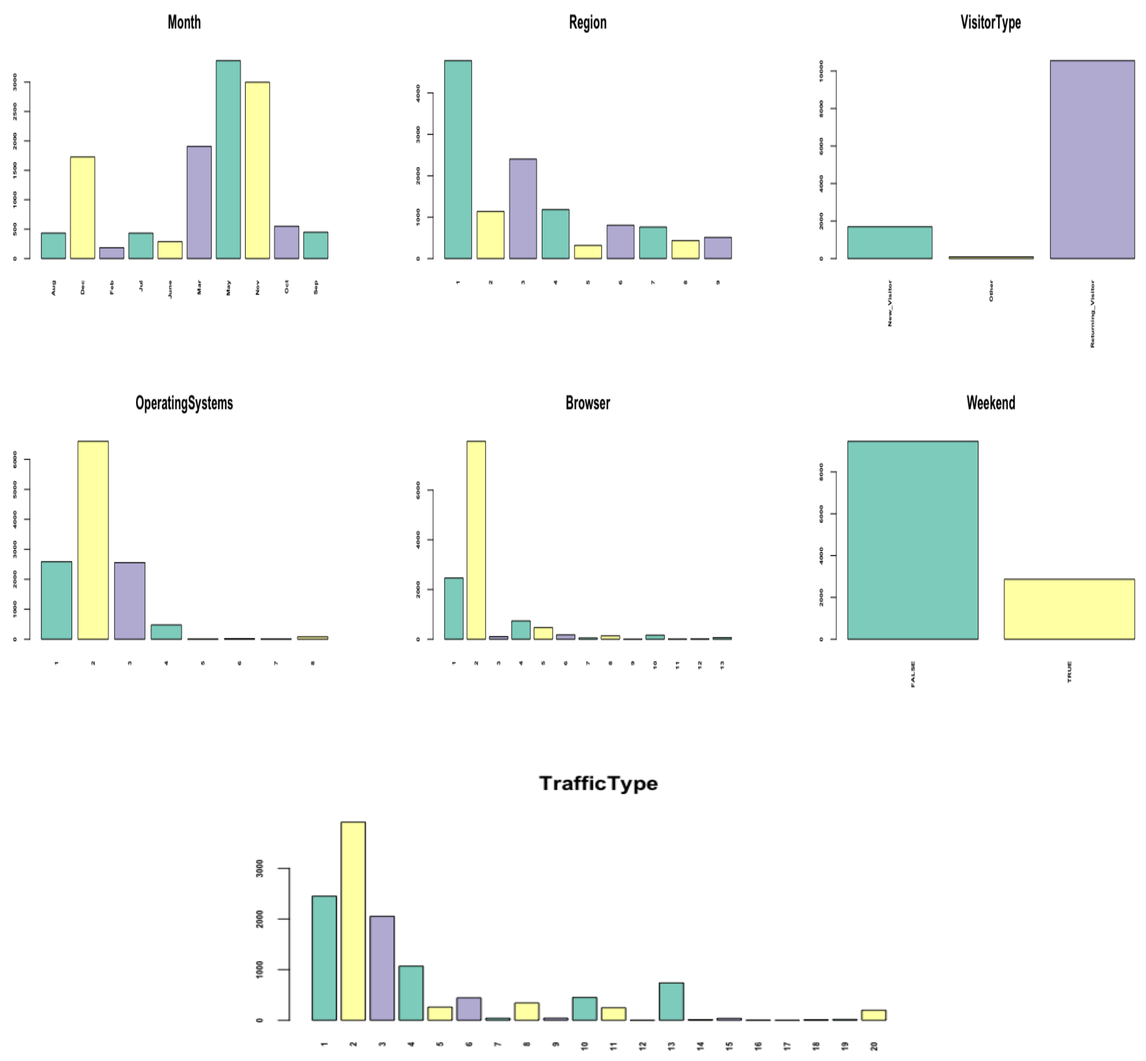

Figure 5: Discrete variables of the Shopping dataset with the frequency of their levels

Only the level names of the variables 'Month', 'Visitor type' and 'Weekend' are know. For the other category variables the difference between the numbers suffice for fitting the model and classification. From left to right, the levels of the variables are:

1. Month: Augustus, December, February, July, June, March, May, November, October, September
2. Visitor type: New visitor, Other, Returning visitor
3. Weekend: False, True

	Admin.	**Admin. duration**	**Info**	**Info duration**	**Prod. related**
Average	2.32	80.82	0.5	34.47	31.73
Minimum	0	0	0	0	0
Median	1	7.5	0	0	18
Maximum	27	3398.75	24	2549.38	705
St. Dev.	3.32	176.78	1.27	140.75	44.48
	Prod. related duration	**Bounce rates**	**Exit rates**	**Page values**	**Special day**
Average	1194.75	0.02	0.04	5.89	0.06
Minimum	0	0	0	0	0
Median	598.94	0	0.03	0	0
Maximum	63973.52	0.2	0.2	361.76	1
St. Dev.	1913.67	0.05	0.05	18.57	0.2

Table 17: Overview of the continuous variables of the Shopping dataset

B.4 Fraud Detection

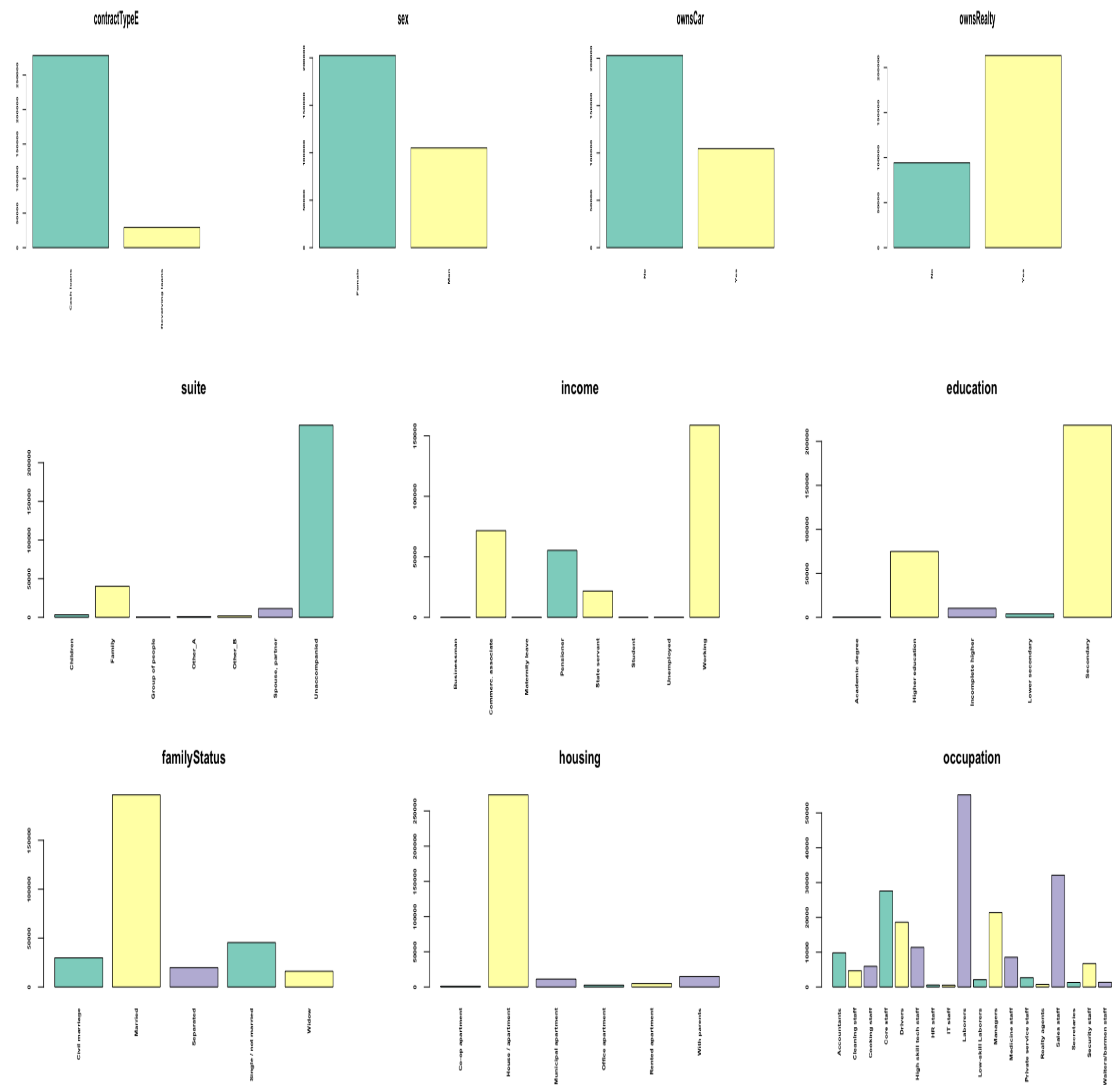

Figure 6: Discrete variables of the Fraud dataset with the frequency of their levels

From left to right, the levels of the variables are:

1. ContractType: Cash Loans, Revolving Loans
2. Sex: Female, Male
3. OwnsCar: No, Yes
4. OwnsRealty: No, Yes
5. Suite: Children, Family, Group of People, Other A, Other B, Spouse Partner, Unaccompanied

6. Income: Businessman, Commercial Associate, Maternity Leave, Pensioner, State Servant, Student, Unemployed, Working
7. Education: Academic Degree, Higher Education, Incomplete Higher, Lower Secondary, Secondary
8. FamilyStatus: Civil Marriage, Married, Separated, Single or Not Married, Widow
9. Housing: Co-op apartment, House or Apartment, Municipal Apartment, Office Apartment, Rented Apartment, With parents
10. Occupation: Accountants, Cleaning staff, Cooking staff, Core staff, Drivers, High skill tech staff, HR staff, IT staff, Laborers, Low-skill Laborers, Managers, Medicine staff, Private service staff, Realty agents, Sales staff, Secretaries, Security staff, Waiters/barmen staff

	Children	**Total Income**	**Credit**	**Annuity**	**Price Goods**	**Family Members**
Average	0.42	1107.31	599026	27108.57	538396.21	2.15
Minimum	0	1	45000	1615.5	40500	1
Median	0	957	513531	24903	450000	2
Maximum	19	2548	4050000	258025.5	4050000	20
St. Dev.	0.72	720.82	402490.78	14493.45	369279.43	0.91

Table 18: Overview of the continuous variables of the Fraud dataset

B.5 Hypothyroid

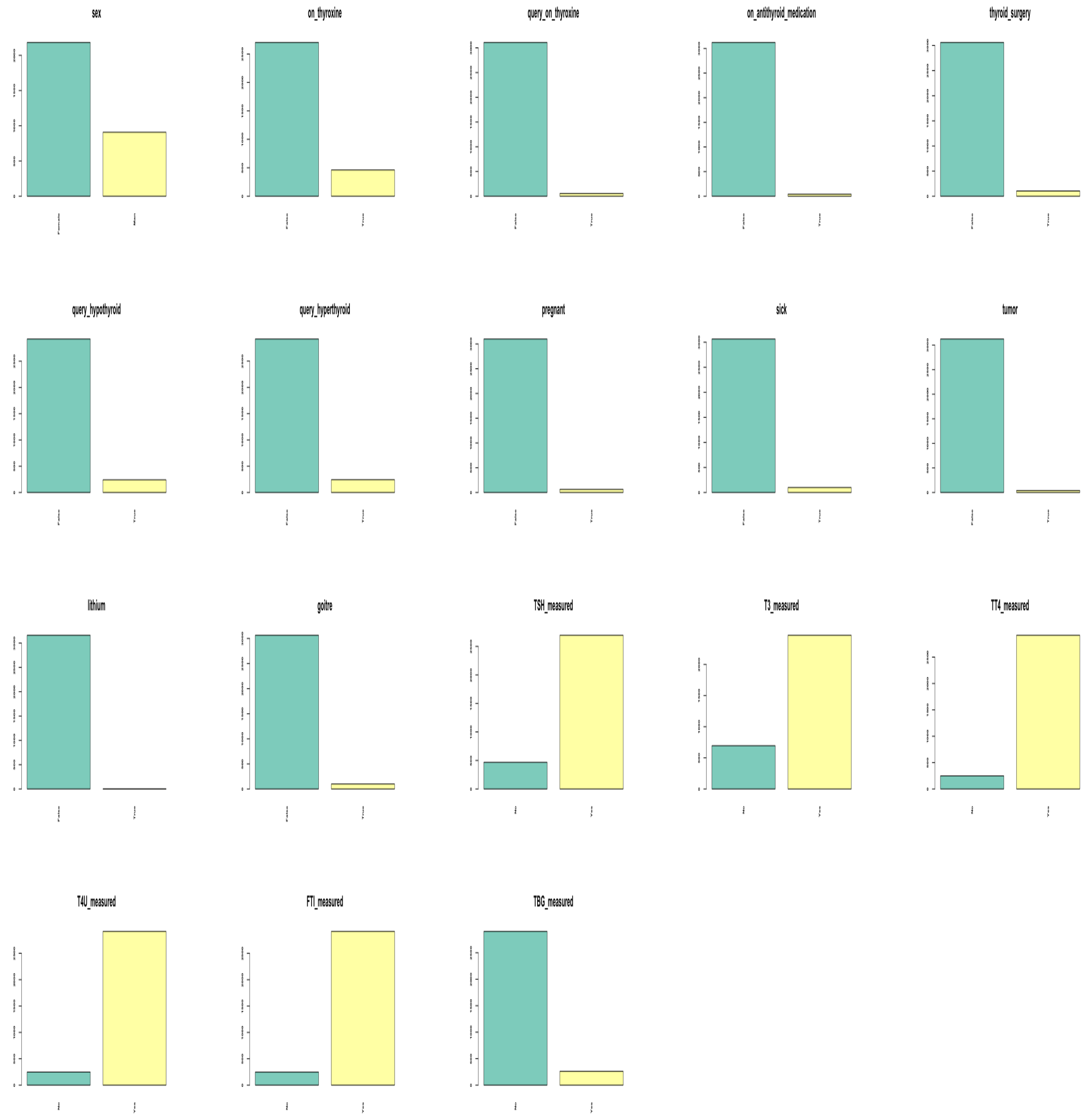

Figure 7: Discrete variables of the Hyperthyroid dataset with the frequency of their levels

From left to right, the levels of the variables are:

1. Sex: Female, Male
2. On thyroxine: False, True
3. Query thyroxine: False, True
4. On antithyroid medication: False, True
5. Thyroid surgery: False, True

6. Query hypothyroid 1: False, True
7. Query hypothyroid 2: False, True
8. Pregnant: False, True
9. Sick: False, True
10. Tumor: False, True
11. Lithium: False, True
12. Goitre: False, True
13. TSH: No, Yes
14. T3: No, Yes
15. TT4: No, Yes
16. T4U: No, Yes
17. FTI: No, Yes
18. TBG: No, Yes

	Age	**TSH**	**T3**	**TT4**	**T4U**	**FTI**	**TBG**
Average	39.87	45.27	17.93	120.1	62.61	112.04	2.81
Minimum	1	1	1	1	1	1	1
Median	42	23	19	71	64	52	1
Maximum	93	240	70	269	159	281	53
St. Dev.	24.84	58.1	12.35	106.54	27.77	109.7	6.72

Table 19: Overview of the continuous variables of the Hyperthyroid dataset

B.6 Kickstarter

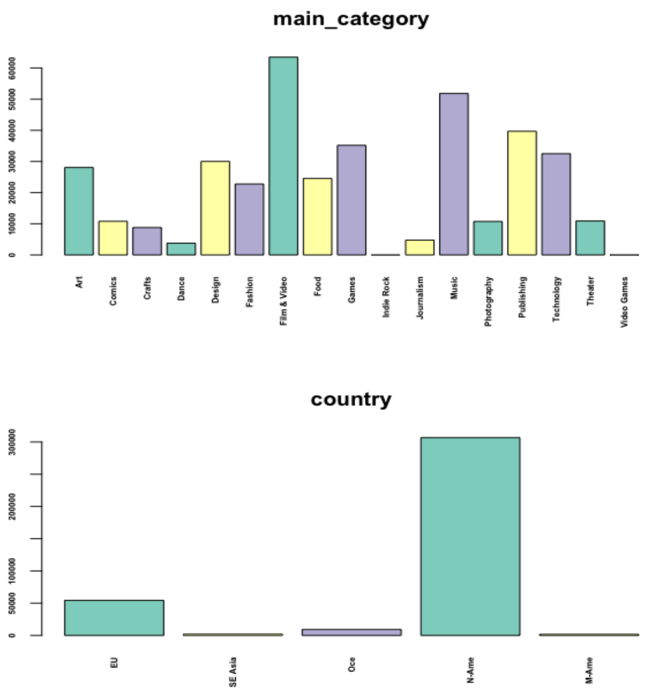

Figure 8: Discrete variables of the Kickstarter dataset with the frequency of their levels

From left to right, the levels of the variables are:

1. Category: Art, Comics, Crafts, Dance, Design, Fashion, Film & Video, Food, Games, Indie Rock, Journalism, Music, Photography, Publishing, Technology, Theater, Video Games

2. Country: EU, SE Asia, Oce, N-Ame, M-Ame

	backers	USD Pledged Real	USD Goal Real	Term
Average	1537.22	9071.62	45499.49	34.48
minimum	2	0	0.01	1
Median	1301	625	5500	30
Maximum	3964	20338986.27	166361390.71	16739
St. Dev.	1307.4	91076.17	1154249.58	65.98

Table 20: Overview of the continuous variables of the Kickstarter dataset

C Classification accuracy results

Ref - Chawla - Put, Undersampling = 0, Oversampling = 1, k = 1									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,4851	0,5252	0,5558	0,6010	0,6450	0,6259	0,6061	0,6653
5% minority	0,6124	0,6945	0,6494	0,5950	0,6982	0,6807	0,5623	0,6860	0,6666
10% minority	0,7178	0,7622	0,7911	0,7340	0,7704	0,8025	0,7182	0,7720	0,7993
15% minority	0,7106	0,7452	0,7521	0,7018	0,7228	0,7358	0,6987	0,7227	0,7331
20% minority	0,7274	0,7551	0,7551	0,7264	0,7348	0,7457	0,7069	0,7270	0,7197
25% minority	0,7565	0,7615	0,7667	0,7495	0,7462	0,7535	0,7478	0,7427	0,7506
30% minority	0,7680	0,7763	0,7707	0,7560	0,7709	0,7719	0,7314	0,7455	0,7432
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,6482	0,6747	0,7122	0,7904	0,7863	0,7846	0,8227	0,8240
5% minority	0,7146	0,7606	0,6826	0,6918	0,7631	0,7616	0,7283	0,7972	0,7837
10% minority	0,7821	0,8367	0,8076	0,7663	0,8267	0,8101	0,7889	0,8519	0,8451
15% minority	0,8061	0,8351	0,8220	0,8094	0,8465	0,8486	0,8238	0,8593	0,8600
20% minority	0,7846	0,8240	0,8037	0,7787	0,8041	0,8182	0,8060	0,8343	0,8512
25% minority	0,8151	0,8358	0,8133	0,8129	0,8349	0,8236	0,8252	0,8509	0,8432
30% minority	0,8291	0,8358	0,8276	0,8196	0,8137	0,8305	0,8320	0,8354	0,8477
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,6684	0,6808	0,7338	0,7691	0,8279	0,8065	0,8401	0,8576
5% minority	0,7541	0,7450	0,7064	0,7302	0,8110	0,7859	0,7754	0,8493	0,8208
10% minority	0,8313	0,8630	0,8395	0,8072	0,8437	0,8471	0,8409	0,8790	0,8818
15% minority	0,8227	0,8560	0,8274	0,8039	0,8362	0,8398	0,8296	0,8762	0,8687
20% minority	0,8513	0,8630	0,8395	0,8272	0,8437	0,8471	0,8609	0,8790	0,8818
25% minority	0,8686	0,8821	0,8688	0,8569	0,8604	0,8605	0,8853	0,8957	0,8987
30% minority	0,8624	0,8729	0,8563	0,8419	0,8499	0,8497	0,8749	0,8862	0,8838
Ref - Chawla - Put, Undersampling = 1, Oversampling = 1, k = 1									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,6555	0,6788	0,5558	0,6413	0,6877	0,6259	0,6492	0,6217
5% minority	0,6124	0,6572	0,6884	0,5950	0,6902	0,7061	0,5623	0,6734	0,6996
10% minority	0,7178	0,7815	0,7642	0,7340	0,7829	0,7804	0,7182	0,7736	0,7696
15% minority	0,7106	0,7414	0,7638	0,7018	0,7286	0,7334	0,6987	0,7230	0,7277
20% minority	0,7274	0,7458	0,7474	0,7264	0,7355	0,7346	0,7069	0,7236	0,7238
25% minority	0,7565	0,7667	0,7671	0,7495	0,7586	0,7609	0,7478	0,7539	0,7571
30% minority	0,7680	0,7808	0,7631	0,7560	0,7707	0,7583	0,7314	0,7473	0,7313
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,8125	0,7976	0,7122	0,7857	0,8039	0,7846	0,8094	0,8172
5% minority	0,7146	0,7730	0,7738	0,6918	0,7389	0,7292	0,7283	0,7684	0,7811
10% minority	0,7821	0,8391	0,8209	0,7663	0,8210	0,8022	0,7889	0,8518	0,8296
15% minority	0,8061	0,8391	0,8171	0,8094	0,8382	0,8399	0,8238	0,8547	0,8520
20% minority	0,7846	0,8306	0,8201	0,7787	0,8161	0,8226	0,8060	0,8390	0,8507
25% minority	0,8151	0,8232	0,8203	0,8129	0,8228	0,8266	0,8252	0,8381	0,8486
30% minority	0,8291	0,8339	0,8171	0,8196	0,8197	0,8107	0,8320	0,8360	0,8371
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,7773	0,7974	0,7338	0,8356	0,7874	0,8065	0,8233	0,8345
5% minority	0,7541	0,7755	0,7978	0,7302	0,7484	0,7594	0,7754	0,7951	0,8011
10% minority	0,8313	0,8574	0,8549	0,8072	0,8387	0,8525	0,8409	0,8749	0,8870
15% minority	0,8227	0,8614	0,8383	0,8039	0,8277	0,8352	0,8296	0,8602	0,8642
20% minority	0,8513	0,8574	0,8549	0,8272	0,8387	0,8525	0,8609	0,8749	0,8870
25% minority	0,8686	0,8786	0,8749	0,8569	0,8626	0,8621	0,8853	0,8921	0,8959
30% minority	0,8624	0,8721	0,8648	0,8419	0,8482	0,8390	0,8749	0,8824	0,8727

Table 21: Classification accuracy of data simulation with k = 1 and oversampling = 1

Ref - Chawla - Put, Undersampling = 0, Oversampling = 1, k = 3									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,5014	0,5184	0,5558	0,6269	0,6549	0,6259	0,6579	0,7064
5% minority	0,6124	0,6537	0,6085	0,5950	0,6980	0,6914	0,5623	0,6444	0,6733
10% minority	0,7178	0,7546	0,7107	0,7340	0,7754	0,7800	0,7182	0,7703	0,7721
15% minority	0,7106	0,7431	0,7098	0,7018	0,7492	0,7457	0,6987	0,7452	0,7462
20% minority	0,7274	0,7457	0,7134	0,7264	0,7485	0,7410	0,7069	0,7172	0,7115
25% minority	0,7565	0,7467	0,7298	0,7495	0,7564	0,7553	0,7478	0,7403	0,7515
30% minority	0,7680	0,7639	0,7311	0,7560	0,7705	0,7753	0,7314	0,7417	0,7411
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,7078	0,6308	0,7122	0,7869	0,7895	0,7846	0,8493	0,8437
5% minority	0,7146	0,7333	0,7582	0,6918	0,7576	0,7681	0,7283	0,7844	0,8035
10% minority	0,7821	0,8233	0,8141	0,7663	0,8150	0,8010	0,7889	0,8356	0,8321
15% minority	0,8061	0,8351	0,8177	0,8094	0,8432	0,8334	0,8238	0,8507	0,8467
20% minority	0,7846	0,8375	0,8227	0,7787	0,8204	0,8220	0,8060	0,8312	0,8484
25% minority	0,8151	0,8276	0,8129	0,8129	0,8300	0,8242	0,8252	0,8385	0,8361
30% minority	0,8291	0,8349	0,8160	0,8196	0,8199	0,8142	0,8320	0,8273	0,8306
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,7528	0,7075	0,7338	0,8562	0,8190	0,8065	0,8445	0,8591
5% minority	0,7541	0,7667	0,7475	0,7302	0,7775	0,7787	0,7754	0,8125	0,8160
10% minority	0,8313	0,8841	0,8641	0,8072	0,8510	0,8492	0,8409	0,8859	0,8859
15% minority	0,8227	0,8628	0,8754	0,8039	0,8332	0,8457	0,8296	0,8636	0,8763
20% minority	0,8513	0,8841	0,8641	0,8272	0,8510	0,8492	0,8609	0,8859	0,8859
25% minority	0,8686	0,8861	0,8778	0,8569	0,8670	0,8621	0,8853	0,8984	0,8933
30% minority	0,8624	0,8773	0,8639	0,8419	0,8476	0,8527	0,8749	0,8798	0,8794
Ref - Chawla - Put, Undersampling = 1, Oversampling = 1, k = 3									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,6197	0,6608	0,5558	0,5634	0,6686	0,6259	0,6169	0,6832
5% minority	0,6124	0,6751	0,6781	0,5950	0,6979	0,6920	0,5623	0,6760	0,6713
10% minority	0,7178	0,7540	0,7396	0,7340	0,7873	0,7805	0,7182	0,7765	0,7687
15% minority	0,7106	0,7338	0,7321	0,7018	0,7290	0,7354	0,6987	0,7167	0,7257
20% minority	0,7274	0,7377	0,7373	0,7264	0,7266	0,7330	0,7069	0,7081	0,7143
25% minority	0,7565	0,7673	0,7332	0,7495	0,7561	0,7580	0,7478	0,7409	0,7498
30% minority	0,7680	0,7628	0,7385	0,7560	0,7600	0,7592	0,7314	0,7385	0,7350
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,8092	0,8264	0,7122	0,8814	0,8056	0,7846	0,8273	0,8189
5% minority	0,7146	0,8026	0,7657	0,6918	0,7583	0,7572	0,7283	0,7748	0,7955
10% minority	0,7821	0,8285	0,8388	0,7663	0,7918	0,8073	0,7889	0,8168	0,8429
15% minority	0,8061	0,8356	0,8290	0,8094	0,8301	0,8269	0,8238	0,8413	0,8441
20% minority	0,7846	0,8279	0,8386	0,7787	0,8186	0,8267	0,8060	0,8277	0,8570
25% minority	0,8151	0,8287	0,8167	0,8129	0,8198	0,8279	0,8252	0,8281	0,8347
30% minority	0,8291	0,8353	0,8233	0,8196	0,8219	0,8197	0,8320	0,8326	0,8330
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,8081	0,7549	0,7338	0,7817	0,7783	0,8065	0,7995	0,8003
5% minority	0,7541	0,8232	0,8056	0,7302	0,7986	0,7545	0,7754	0,8153	0,8014
10% minority	0,8313	0,8281	0,8651	0,8072	0,8651	0,8371	0,8409	0,8564	0,8684
15% minority	0,8227	0,8627	0,8439	0,8039	0,8201	0,8313	0,8296	0,8537	0,8644
20% minority	0,8513	0,8693	0,8651	0,8272	0,8281	0,8371	0,8609	0,8651	0,8684
25% minority	0,8686	0,8913	0,8857	0,8569	0,8577	0,8694	0,8853	0,8945	0,8989
30% minority	0,8624	0,8804	0,8610	0,8419	0,8487	0,8542	0,8749	0,8811	0,8823

Table 22: Classification accuracy of data simulation with k = 3 and oversampling = 1

Ref - Chawla - Put, Undersampling = 0, Oversampling = 2, k = 3									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,6008	0,6065	0,5558	0,6910	0,6610	0,6259	0,6970	0,6906
5% minority	0,6124	0,6759	0,5624	0,5950	0,6531	0,6592	0,5623	0,6476	0,6462
10% minority	0,7178	0,7233	0,7259	0,7340	0,7757	0,7793	0,7182	0,7596	0,7713
15% minority	0,7106	0,7127	0,6810	0,7018	0,7394	0,7419	0,6987	0,7183	0,7288
20% minority	0,7274	0,7416	0,6663	0,7264	0,7406	0,7448	0,7069	0,6930	0,7092
25% minority	0,7565	0,7488	0,7242	0,7495	0,7585	0,7574	0,7478	0,7404	0,7479
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,6571	0,7054	0,7122	0,7805	0,7616	0,7846	0,8394	0,8376
5% minority	0,7146	0,7605	0,7161	0,6918	0,7522	0,7740	0,7283	0,7617	0,8013
10% minority	0,7821	0,8252	0,8082	0,7663	0,8198	0,8210	0,7889	0,8411	0,8529
15% minority	0,8061	0,8404	0,8223	0,8094	0,8452	0,8351	0,8238	0,8540	0,8539
20% minority	0,7846	0,8361	0,8055	0,7787	0,8090	0,8053	0,8060	0,8220	0,8256
25% minority	0,8151	0,8282	0,8186	0,8129	0,8136	0,8387	0,8252	0,8299	0,8489
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,6652	0,6638	0,7338	0,7879	0,8111	0,8065	0,8155	0,8492
5% minority	0,7541	0,7846	0,7691	0,7302	0,7946	0,7800	0,7754	0,8301	0,8179
10% minority	0,8313	0,8647	0,8656	0,8072	0,8389	0,8336	0,8409	0,8706	0,8725
15% minority	0,8227	0,8664	0,8551	0,8039	0,8411	0,8502	0,8296	0,8665	0,8815
20% minority	0,8513	0,8647	0,8656	0,8272	0,8389	0,8336	0,8609	0,8706	0,8725
25% minority	0,8686	0,8934	0,8923	0,8569	0,8547	0,8629	0,8853	0,8882	0,8975
Ref - Chawla - Put, Undersampling = 1, Oversampling = 2, k = 3									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,6322	0,6350	0,5558	0,6592	0,6693	0,6259	0,6213	0,6499
5% minority	0,6124	0,6581	0,6198	0,5950	0,6632	0,6640	0,5623	0,6540	0,6459
10% minority	0,7178	0,7366	0,7528	0,7340	0,7611	0,7842	0,7182	0,7503	0,7767
15% minority	0,7106	0,7204	0,6841	0,7018	0,7292	0,7309	0,6987	0,7255	0,7218
20% minority	0,7274	0,7487	0,6786	0,7264	0,7339	0,7315	0,7069	0,7032	0,7038
25% minority	0,7565	0,7550	0,7192	0,7495	0,7663	0,7577	0,7478	0,7382	0,7488
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,8126	0,8126	0,7122	0,8153	0,8298	0,7846	0,7814	0,8066
5% minority	0,7146	0,7851	0,7796	0,6918	0,7621	0,7546	0,7283	0,7627	0,7922
10% minority	0,7821	0,8362	0,8349	0,7663	0,8112	0,8081	0,7889	0,8346	0,8386
15% minority	0,8061	0,8261	0,8316	0,8094	0,8384	0,8472	0,8238	0,8442	0,8610
20% minority	0,7846	0,8320	0,8060	0,7787	0,8107	0,8106	0,8060	0,8185	0,8378
25% minority	0,8151	0,8270	0,8160	0,8129	0,8150	0,8254	0,8252	0,8280	0,8375
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,8395	0,8115	0,7338	0,8142	0,8270	0,8065	0,8469	0,8467
5% minority	0,7541	0,7938	0,7985	0,7302	0,7479	0,7622	0,7754	0,7726	0,7981
10% minority	0,8313	0,8605	0,8571	0,8072	0,8361	0,8441	0,8409	0,8702	0,8810
15% minority	0,8227	0,8622	0,8554	0,8039	0,8289	0,8351	0,8296	0,8602	0,8670
20% minority	0,8513	0,8605	0,8571	0,8272	0,8361	0,8441	0,8609	0,8702	0,8810
25% minority	0,8686	0,8858	0,8846	0,8569	0,8603	0,8664	0,8853	0,8923	0,8993

Table 23: Classification accuracy of data simulation with k = 3 and oversampling = 2

Ref - Chawla - Put, Undersampling = 0, Oversampling = 3, k = 3									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,5800	0,5538	0,5558	0,6301	0,6371	0,6259	0,6348	0,6439
5% minority	0,6124	0,6307	0,5789	0,5950	0,6727	0,6849	0,5623	0,6643	0,6709
10% minority	0,7178	0,7255	0,7191	0,7340	0,7626	0,7820	0,7182	0,7533	0,7719
15% minority	0,7106	0,7101	0,6636	0,7018	0,7351	0,7233	0,6987	0,7249	0,7063
20% minority	0,7274	0,7179	0,6438	0,7264	0,7268	0,7365	0,7069	0,6824	0,6989
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,7193	0,6950	0,7122	0,8250	0,7242	0,7846	0,8706	0,8335
5% minority	0,7146	0,7516	0,7198	0,6918	0,7693	0,7658	0,7283	0,7882	0,7810
10% minority	0,7821	0,8256	0,7966	0,7663	0,8055	0,8294	0,7889	0,8296	0,8541
15% minority	0,8061	0,8411	0,8248	0,8094	0,8393	0,8409	0,8238	0,8512	0,8476
20% minority	0,7846	0,8334	0,8055	0,7787	0,8128	0,8218	0,8060	0,8194	0,8437
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,7359	0,7171	0,7338	0,7704	0,8321	0,8065	0,8604	0,8708
5% minority	0,7541	0,7880	0,7472	0,7302	0,7969	0,7819	0,7754	0,8328	0,8199
10% minority	0,8313	0,8688	0,8642	0,8072	0,8322	0,8286	0,8409	0,8704	0,8739
15% minority	0,8227	0,8691	0,8609	0,8039	0,8332	0,8385	0,8296	0,8685	0,8787
20% minority	0,8513	0,8688	0,8642	0,8272	0,8322	0,8286	0,8609	0,8704	0,8739
Ref - Chawla - Put, Undersampling = 1, Oversampling = 3, k = 3									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,6350	0,6525	0,5558	0,6816	0,6997	0,6259	0,6397	0,7039
5% minority	0,6124	0,6357	0,6160	0,5950	0,6707	0,6724	0,5623	0,6524	0,6574
10% minority	0,7178	0,7822	0,7464	0,7340	0,8005	0,7787	0,7182	0,7872	0,7654
15% minority	0,7106	0,7234	0,6869	0,7018	0,7271	0,7360	0,6987	0,7095	0,7249
20% minority	0,7274	0,7228	0,6532	0,7264	0,7271	0,7313	0,7069	0,6960	0,6986
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,8709	0,8057	0,7122	0,8431	0,8066	0,7846	0,8074	0,7892
5% minority	0,7146	0,7817	0,7749	0,6918	0,7637	0,7506	0,7283	0,7761	0,7776
10% minority	0,7821	0,8194	0,8181	0,7663	0,7909	0,8332	0,7889	0,8187	0,8505
15% minority	0,8061	0,8505	0,8234	0,8094	0,8430	0,8357	0,8238	0,8576	0,8466
20% minority	0,7846	0,8213	0,7961	0,7787	0,7992	0,8120	0,8060	0,8072	0,8393
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,7500	0,7988	0,7338	0,7934	0,8411	0,8065	0,8005	0,8525
5% minority	0,7541	0,7994	0,8082	0,7302	0,7623	0,7780	0,7754	0,7962	0,8044
10% minority	0,8313	0,8718	0,8679	0,8072	0,8214	0,8291	0,8409	0,8628	0,8764
15% minority	0,8227	0,8725	0,8612	0,8039	0,8370	0,8344	0,8296	0,8678	0,8630
20% minority	0,8513	0,8718	0,8679	0,8272	0,8214	0,8291	0,8609	0,8628	0,8764

Table 24: Classification accuracy of data simulation with k = 3 and oversampling = 3

Ref - Chawla - Put, Undersampling = 0, Oversampling = 1, k = 5									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,5658	0,5700	0,5558	0,6660	0,6971	0,6259	0,6819	0,7266
5% minority	0,6124	0,6366	0,6167	0,5950	0,6583	0,6338	0,5623	0,6358	0,6165
10% minority	0,7178	0,7627	0,7336	0,7340	0,7932	0,7785	0,7182	0,7828	0,7647
15% minority	0,7106	0,7367	0,7078	0,7018	0,7215	0,7297	0,6987	0,7273	0,7295
20% minority	0,7274	0,7597	0,7270	0,7264	0,7502	0,7501	0,7069	0,7244	0,7221
25% minority	0,7565	0,7609	0,7477	0,7495	0,7611	0,7578	0,7478	0,7467	0,7477
30% minority	0,7680	0,7601	0,7360	0,7560	0,7621	0,7632	0,7314	0,7285	0,7357
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,6536	0,7182	0,7122	0,7854	0,8190	0,7846	0,8151	0,8554
5% minority	0,7146	0,7619	0,7265	0,6918	0,7462	0,7816	0,7283	0,7805	0,8130
10% minority	0,7821	0,8342	0,8350	0,7663	0,8312	0,8369	0,7889	0,8506	0,8598
15% minority	0,8061	0,8309	0,8353	0,8094	0,8306	0,8564	0,8238	0,8446	0,8648
20% minority	0,7846	0,8307	0,8270	0,7787	0,8039	0,8215	0,8060	0,8193	0,8478
25% minority	0,8151	0,8242	0,8329	0,8129	0,8257	0,8396	0,8252	0,8356	0,8512
30% minority	0,8291	0,8419	0,8367	0,8196	0,8212	0,8283	0,8320	0,8370	0,8415
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,7123	0,7946	0,7338	0,7719	0,8764	0,8065	0,8414	0,8811
5% minority	0,7541	0,7574	0,7737	0,7302	0,7600	0,7876	0,7754	0,8046	0,8141
10% minority	0,8313	0,8735	0,8581	0,8072	0,8346	0,8280	0,8409	0,8751	0,8688
15% minority	0,8227	0,8742	0,8624	0,8039	0,8533	0,8459	0,8296	0,8837	0,8718
20% minority	0,8513	0,8735	0,8581	0,8272	0,8346	0,8280	0,8609	0,8751	0,8688
25% minority	0,8686	0,8885	0,8781	0,8569	0,8513	0,8662	0,8853	0,8862	0,8975
30% minority	0,8624	0,8729	0,8735	0,8419	0,8537	0,8328	0,8749	0,8847	0,8698
Ref - Chawla - Put, Undersampling = 1, Oversampling = 1, k = 5									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,6548	0,6805	0,5558	0,6886	0,7012	0,6259	0,6591	0,6937
5% minority	0,6124	0,6660	0,6309	0,5950	0,6997	0,6948	0,5623	0,6898	0,6743
10% minority	0,7178	0,7603	0,7623	0,7340	0,7788	0,7884	0,7182	0,7666	0,7880
15% minority	0,7106	0,7264	0,7272	0,7018	0,7199	0,7236	0,6987	0,7107	0,7191
20% minority	0,7274	0,7344	0,7385	0,7264	0,7273	0,7325	0,7069	0,7061	0,7151
25% minority	0,7565	0,7612	0,7517	0,7495	0,7593	0,7612	0,7478	0,7534	0,7569
30% minority	0,7680	0,7634	0,7566	0,7560	0,7624	0,7687	0,7314	0,7319	0,7438
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,8117	0,7846	0,7122	0,8081	0,7815	0,7846	0,7681	0,7826
5% minority	0,7146	0,7914	0,7700	0,6918	0,7358	0,7590	0,7283	0,7641	0,7828
10% minority	0,7821	0,8353	0,8442	0,7663	0,8102	0,8122	0,7889	0,8340	0,8410
15% minority	0,8061	0,8248	0,8430	0,8094	0,8241	0,8428	0,8238	0,8344	0,8591
20% minority	0,7846	0,8371	0,8179	0,7787	0,8185	0,8154	0,8060	0,8299	0,8331
25% minority	0,8151	0,8330	0,8179	0,8129	0,8227	0,8289	0,8252	0,8319	0,8361
30% minority	0,8291	0,8398	0,8309	0,8196	0,8204	0,8262	0,8320	0,8346	0,8396
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,7750	0,7897	0,7338	0,8292	0,8743	0,8065	0,8412	0,8317
5% minority	0,7541	0,8095	0,8022	0,7302	0,7596	0,7487	0,7754	0,7853	0,7793
10% minority	0,8313	0,8727	0,8649	0,8072	0,8334	0,8283	0,8409	0,8632	0,8715
15% minority	0,8227	0,8643	0,8521	0,8039	0,8390	0,8317	0,8296	0,8689	0,8671
20% minority	0,8513	0,8727	0,8649	0,8272	0,8334	0,8283	0,8609	0,8632	0,8715
25% minority	0,8686	0,8899	0,8832	0,8569	0,8606	0,8660	0,8853	0,8863	0,8985
30% minority	0,8624	0,8719	0,8774	0,8419	0,8483	0,8610	0,8749	0,8802	0,8911

Table 25: Classification accuracy of data simulation with k = 5 and oversampling = 1

Ref - Chawla - Put, Undersampling = 0, Oversampling = 2, k = 5									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,5358	0,6038	0,5558	0,6506	0,6405	0,6259	0,6807	0,6599
5% minority	0,6124	0,6505	0,6069	0,5950	0,6748	0,6744	0,5623	0,6522	0,6513
10% minority	0,7178	0,7699	0,7372	0,7340	0,7971	0,7892	0,7182	0,7817	0,7758
15% minority	0,7106	0,7327	0,6845	0,7018	0,7220	0,7259	0,6987	0,7242	0,7173
20% minority	0,7274	0,7390	0,6949	0,7264	0,7469	0,7397	0,7069	0,7033	0,7041
25% minority	0,7565	0,7588	0,7327	0,7495	0,7528	0,7616	0,7478	0,7359	0,7516
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,6315	0,6972	0,7122	0,7417	0,7913	0,7846	0,8217	0,8463
5% minority	0,7146	0,7611	0,7651	0,6918	0,7466	0,7747	0,7283	0,7746	0,8067
10% minority	0,7821	0,8479	0,8449	0,7663	0,8346	0,8455	0,7889	0,8577	0,8646
15% minority	0,8061	0,8335	0,8410	0,8094	0,8301	0,8514	0,8238	0,8434	0,8621
20% minority	0,7846	0,8425	0,8172	0,7787	0,8153	0,8106	0,8060	0,8235	0,8383
25% minority	0,8151	0,8270	0,8235	0,8129	0,8112	0,8257	0,8252	0,8270	0,8490
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,7833	0,7665	0,7338	0,8210	0,8421	0,8065	0,8859	0,8513
5% minority	0,7541	0,7925	0,7600	0,7302	0,7878	0,7830	0,7754	0,8158	0,8245
10% minority	0,8313	0,8748	0,8682	0,8072	0,8313	0,8445	0,8409	0,8721	0,8792
15% minority	0,8227	0,8746	0,8652	0,8039	0,8388	0,8377	0,8296	0,8694	0,8744
20% minority	0,8513	0,8748	0,8682	0,8272	0,8313	0,8445	0,8609	0,8721	0,8792
25% minority	0,8686	0,8919	0,8917	0,8569	0,8476	0,8609	0,8853	0,8888	0,8984
Ref - Chawla - Put, Undersampling = 1, Oversampling = 2, k = 5									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,6370	0,7108	0,5558	0,6649	0,6702	0,6259	0,6233	0,6369
5% minority	0,6124	0,6601	0,6466	0,5950	0,6837	0,6824	0,5623	0,6686	0,6597
10% minority	0,7178	0,7759	0,7514	0,7340	0,7768	0,7917	0,7182	0,7668	0,7855
15% minority	0,7106	0,7267	0,7075	0,7018	0,7330	0,7285	0,6987	0,7112	0,7242
20% minority	0,7274	0,7411	0,7019	0,7264	0,7315	0,7326	0,7069	0,7034	0,7159
25% minority	0,7565	0,7550	0,7201	0,7495	0,7587	0,7678	0,7478	0,7382	0,7511
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,7984	0,8271	0,7122	0,8039	0,8253	0,7846	0,7891	0,7997
5% minority	0,7146	0,7596	0,7701	0,6918	0,7392	0,7388	0,7283	0,7676	0,7611
10% minority	0,7821	0,8473	0,8429	0,7663	0,8118	0,8253	0,7889	0,8343	0,8443
15% minority	0,8061	0,8432	0,8545	0,8094	0,8409	0,8503	0,8238	0,8466	0,8604
20% minority	0,7846	0,8350	0,8158	0,7787	0,8123	0,8166	0,8060	0,8253	0,8455
25% minority	0,8151	0,8311	0,8197	0,8129	0,8235	0,8252	0,8252	0,8371	0,8417
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,7224	0,7823	0,7338	0,7858	0,8035	0,8065	0,7949	0,8092
5% minority	0,7541	0,8429	0,8121	0,7302	0,7704	0,7742	0,7754	0,7989	0,8056
10% minority	0,8313	0,8689	0,8610	0,8072	0,8415	0,8404	0,8409	0,8738	0,8773
15% minority	0,8227	0,8698	0,8651	0,8039	0,8384	0,8265	0,8296	0,8651	0,8590
20% minority	0,8513	0,8689	0,8610	0,8272	0,8415	0,8404	0,8609	0,8738	0,8773
25% minority	0,8686	0,8937	0,8918	0,8569	0,8605	0,8527	0,8853	0,8958	0,8943

Table 26: Classification accuracy of data simulation with k = 5 and oversampling = 2

Ref - Chawla - Put, Undersampling = 0, Oversampling = 3, k = 5									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,6177	0,5855	0,5558	0,6333	0,6503	0,6259	0,6730	0,6794
5% minority	0,6124	0,6367	0,5977	0,5950	0,6863	0,6683	0,5623	0,6850	0,6453
10% minority	0,7178	0,7669	0,7360	0,7340	0,8014	0,7744	0,7182	0,7868	0,7677
15% minority	0,7106	0,7321	0,6891	0,7018	0,7332	0,7320	0,6987	0,7199	0,7192
20% minority	0,7274	0,7372	0,6715	0,7264	0,7421	0,7362	0,7069	0,6969	0,7002
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,7421	0,6493	0,7122	0,8157	0,7716	0,7846	0,8579	0,8256
5% minority	0,7146	0,7448	0,7579	0,6918	0,7529	0,8023	0,7283	0,7747	0,8131
10% minority	0,7821	0,8410	0,8300	0,7663	0,8096	0,8196	0,7889	0,8344	0,8483
15% minority	0,8061	0,8404	0,8328	0,8094	0,8305	0,8375	0,8238	0,8410	0,8432
20% minority	0,7846	0,8349	0,8137	0,7787	0,8081	0,8169	0,8060	0,8185	0,8454
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,7044	0,7271	0,7338	0,8156	0,8282	0,8065	0,8588	0,8268
5% minority	0,7541	0,8122	0,8149	0,7302	0,7840	0,8094	0,7754	0,8151	0,8469
10% minority	0,8313	0,8673	0,8772	0,8072	0,8257	0,8275	0,8409	0,8612	0,8736
15% minority	0,8227	0,8850	0,8738	0,8039	0,8377	0,8310	0,8296	0,8690	0,8672
20% minority	0,8513	0,8673	0,8772	0,8272	0,8257	0,8275	0,8609	0,8612	0,8736
Ref - Chawla - Put, Undersampling = 1, Oversampling = 3, k = 5									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,5838	0,6962	0,5558	0,6724	0,7084	0,6259	0,6063	0,6656
5% minority	0,6124	0,6382	0,6352	0,5950	0,6487	0,6620	0,5623	0,6390	0,6465
10% minority	0,7178	0,7618	0,7481	0,7340	0,7742	0,7859	0,7182	0,7637	0,7740
15% minority	0,7106	0,7347	0,6835	0,7018	0,7383	0,7308	0,6987	0,7258	0,7172
20% minority	0,7274	0,7258	0,6724	0,7264	0,7224	0,7461	0,7069	0,6840	0,7062
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,8285	0,7992	0,7122	0,8751	0,8042	0,7846	0,8347	0,7544
5% minority	0,7146	0,7869	0,7847	0,6918	0,7581	0,7529	0,7283	0,7748	0,7840
10% minority	0,7821	0,8424	0,8491	0,7663	0,8253	0,8267	0,7889	0,8378	0,8479
15% minority	0,8061	0,8387	0,8343	0,8094	0,8325	0,8450	0,8238	0,8425	0,8576
20% minority	0,7846	0,8346	0,8180	0,7787	0,8027	0,8165	0,8060	0,8198	0,8498
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,7906	0,8419	0,7338	0,8202	0,9110	0,8065	0,8308	0,8656
5% minority	0,7541	0,8188	0,8140	0,7302	0,7399	0,7896	0,7754	0,7616	0,8116
10% minority	0,8313	0,8785	0,8729	0,8072	0,8238	0,8391	0,8409	0,8640	0,8783
15% minority	0,8227	0,8700	0,8700	0,8039	0,8411	0,8392	0,8296	0,8674	0,8719
20% minority	0,8513	0,8785	0,8729	0,8272	0,8238	0,8391	0,8609	0,8640	0,8783

Table 27: Classification accuracy of data simulation with k = 5 and oversampling = 3

Ref - Chawla - Put, Undersampling = 0, Oversampling = 4, k = 5									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,6239	0,6371	0,5558	0,5997	0,6715	0,6259	0,6558	0,6628
5% minority	0,6124	0,6528	0,6103	0,5950	0,6802	0,6557	0,5623	0,6666	0,6402
10% minority	0,7178	0,7441	0,7311	0,7340	0,7895	0,7782	0,7182	0,7702	0,7658
15% minority	0,7106	0,7167	0,6750	0,7018	0,7365	0,7429	0,6987	0,7234	0,7196
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,6939	0,6686	0,7122	0,7708	0,7735	0,7846	0,8118	0,7666
5% minority	0,7146	0,7645	0,7592	0,6918	0,7442	0,7678	0,7283	0,7659	0,7961
10% minority	0,7821	0,8360	0,8084	0,7663	0,8172	0,8111	0,7889	0,8359	0,8331
15% minority	0,8061	0,8385	0,8446	0,8094	0,8261	0,8409	0,8238	0,8359	0,8580
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,6911	0,7284	0,7338	0,8759	0,8576	0,8065	0,8716	0,8756
5% minority	0,7541	0,8059	0,7980	0,7302	0,7970	0,7768	0,7754	0,8179	0,8158
10% minority	0,8313	0,8360	0,8459	0,8072	0,8157	0,8294	0,8409	0,8425	0,8513
15% minority	0,8227	0,8811	0,8679	0,8039	0,8229	0,8346	0,8296	0,8611	0,8695
Ref - Chawla - Put, Undersampling = 1, Oversampling = 4, k = 5									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,6399	0,6914	0,5558	0,6284	0,6677	0,6259	0,6194	0,6562
5% minority	0,6124	0,6780	0,6385	0,5950	0,6864	0,6659	0,5623	0,6731	0,6659
10% minority	0,7178	0,7447	0,7507	0,7340	0,7695	0,7863	0,7182	0,7609	0,7734
15% minority	0,7106	0,7217	0,6907	0,7018	0,7439	0,7344	0,6987	0,7266	0,7245
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,7790	0,8164	0,7122	0,7950	0,7941	0,7846	0,7509	0,7921
5% minority	0,7146	0,7992	0,8052	0,6918	0,7215	0,7517	0,7283	0,7389	0,7793
10% minority	0,7821	0,8292	0,8305	0,7663	0,8115	0,8216	0,7889	0,8318	0,8353
15% minority	0,8061	0,8475	0,8458	0,8094	0,8269	0,8429	0,8238	0,8424	0,8570
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,7838	0,7883	0,7338	0,8269	0,8337	0,8065	0,8459	0,8468
5% minority	0,7541	0,8224	0,8267	0,7302	0,7513	0,7904	0,7754	0,7791	0,8227
10% minority	0,8313	0,8616	0,8705	0,8072	0,8395	0,8517	0,8409	0,8481	0,8602
15% minority	0,8227	0,8742	0,8773	0,8039	0,8197	0,8273	0,8296	0,8527	0,8698

Table 28: Classification accuracy of data simulation with k = 5 and oversampling = 4

Ref - Chawla - Put, Undersampling = 0, Oversampling = 5, k = 5									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,6021	0,5497	0,5558	0,6678	0,6400	0,6259	0,6809	0,7152
5% minority	0,6124	0,6762	0,6006	0,5950	0,6645	0,6845	0,5623	0,6527	0,6684
10% minority	0,7178	0,7441	0,7453	0,7340	0,7849	0,7750	0,7182	0,7746	0,7658
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,7965	0,7106	0,7122	0,7651	0,7796	0,7846	0,8448	0,8450
5% minority	0,7146	0,7563	0,7912	0,6918	0,7365	0,7898	0,7283	0,7650	0,8129
10% minority	0,7821	0,8263	0,8302	0,7663	0,8105	0,8077	0,7889	0,8209	0,8447
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,8276	0,7942	0,7338	0,8174	0,7499	0,8065	0,8558	0,8086
5% minority	0,7541	0,8108	0,7809	0,7302	0,7785	0,7678	0,7754	0,8221	0,8238
10% minority	0,8313	0,8420	0,8289	0,8072	0,8398	0,8038	0,8409	0,8514	0,8579
Ref - Chawla - Put, Undersampling = 1, Oversampling = 5, k = 5									
	KNN			Bagging			Random Forest		
p = 4	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,3462	0,6765	0,6491	0,5558	0,7058	0,6512	0,6259	0,7063	0,6271
5% minority	0,6124	0,6921	0,6311	0,5950	0,7018	0,6808	0,5623	0,6816	0,6642
10% minority	0,7178	0,7734	0,7445	0,7340	0,7860	0,7742	0,7182	0,7741	0,7635
p = 10	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,5940	0,8180	0,7839	0,7122	0,7383	0,7489	0,7846	0,7442	0,7638
5% minority	0,7146	0,7879	0,8000	0,6918	0,7416	0,7640	0,7283	0,7684	0,7893
10% minority	0,7821	0,8365	0,8371	0,7663	0,8146	0,8089	0,7889	0,8250	0,8460
p = 20	Ref	Chawla	Put	Ref	Chawla	Put	Ref	Chawla	Put
1% minority	0,7306	0,8007	0,7872	0,7338	0,7807	0,8252	0,8065	0,8128	0,8306
5% minority	0,7541	0,8504	0,8487	0,7302	0,7858	0,8061	0,7754	0,8195	0,8225
10% minority	0,8313	0,8503	0,8493	0,8072	0,8408	0,8599	0,8409	0,8703	0,8729

Table 29: Classification accuracy of data simulation with k = 5 and oversampling = 5

Ref - Chawla - Put, Undersampling = 0, Oversampling = 1, k = 1									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5427	0,5747	0,5852	0,6008	0,6322	0,5934	0,6468	0,6160
Hyper	0,9117	0,9741	0,9767	0,9196	0,9792	0,9787	0,9305	0,9914	0,9930
Fraud	0,4995	0,5462	0,5499	0,5365	0,5502	0,5630	0,5333	0,5651	0,5603
Shopping	0,6377	0,7200	0,7177	0,8516	0,8783	0,8755	0,8466	0,8828	0,8848
Churn	0,5260	0,5644	0,5689	0,7877	0,8211	0,8246	0,8232	0,8463	0,8515
Adult	0,5579	0,5772	0,5826	0,8641	0,8747	0,8808	0,8841	0,8954	0,9017
Ref - Chawla - Put, Undersampling = 1, Oversampling = 1, k = 1									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,6021	0,5779	0,5852	0,6077	0,5793	0,5934	0,5739	0,5949
Hyper	0,9117	0,9871	0,9804	0,9196	0,9916	0,9892	0,9305	0,9927	0,9938
Fraud	0,4995	0,5554	0,5469	0,5365	0,5483	0,5631	0,5333	0,5509	0,5653
Shopping	0,6377	0,6671	0,6723	0,8516	0,8864	0,8903	0,8466	0,8828	0,8889
Churn	0,5260	0,5729	0,5676	0,7877	0,8305	0,8281	0,8232	0,8493	0,8463
Adult	0,5579	0,5608	0,5662	0,8641	0,8697	0,8735	0,8841	0,8939	0,8979
Ref - Chawla - Put, Undersampling = 0, Oversampling = 1, k = 3									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,4969	0,4849	0,5852	0,5892	0,5926	0,5934	0,5761	0,5999
Hyper	0,9117	0,9565	0,9633	0,9196	0,9791	0,9810	0,9305	0,9907	0,9904
Fraud	0,4995	0,5480	0,5609	0,5365	0,5742	0,5812	0,5333	0,5708	0,5802
Shopping	0,6377	0,7086	0,7175	0,8516	0,8864	0,8736	0,8466	0,8847	0,8715
Churn	0,5260	0,5311	0,5566	0,7877	0,8217	0,8273	0,8232	0,8526	0,8515
Adult	0,5579	0,5793	0,5807	0,8641	0,8765	0,8755	0,8841	0,8927	0,8928
Ref - Chawla - Put, Undersampling = 1, Oversampling = 1, K = 3									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5727	0,5572	0,5852	0,5923	0,6122	0,5934	0,6342	0,6097
Hyper	0,9117	0,9825	0,9806	0,9196	0,9920	0,9900	0,9305	0,9938	0,9916
Fraud	0,4995	0,5494	0,5529	0,5365	0,5587	0,5706	0,5333	0,5592	0,5677
Shopping	0,6377	0,7028	0,6921	0,8516	0,8907	0,8850	0,8466	0,8875	0,8833
Churn	0,5260	0,5364	0,5636	0,7877	0,8296	0,8328	0,8232	0,8511	0,8501
Adult	0,5579	0,5900	0,5667	0,8641	0,8723	0,8679	0,8841	0,9002	0,8897
Ref - Chawla - Put, Undersampling = 0, Oversampling = 2, k = 3									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5023	0,5065	0,5852	0,5932	0,5888	0,5934	0,6096	0,5945
Hyper	0,9117	0,9675	0,9692	0,9196	0,9886	0,9841	0,9305	0,9943	0,9933
Fraud	0,4995	0,5452	0,5561	0,5365	0,5550	0,5857	0,5333	0,5570	0,5629
Shopping	0,6377	0,7266	0,7385	0,8516	0,8828	0,8852	0,8466	0,8850	0,8853
Churn	0,5260	0,5293	0,5634	0,7877	0,8169	0,8258	0,8232	0,8461	0,8488
Adult	0,5579	0,5880	0,5816	0,8641	0,8659	0,8711	0,8841	0,8895	0,8892
Ref - Chawla - Put, Undersampling = 1, Oversampling = 2, K = 3									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5767	0,5699	0,5852	0,6021	0,5964	0,5934	0,6254	0,5970
Hyper	0,9117	0,9835	0,9810	0,9196	0,9919	0,9893	0,9305	0,9923	0,9922
Fraud	0,4995	0,5335	0,5558	0,5365	0,5373	0,5617	0,5333	0,5544	0,5653
Shopping	0,6377	0,7070	0,7061	0,8516	0,8851	0,8944	0,8466	0,8865	0,8915
Churn	0,5260	0,5292	0,5654	0,7877	0,8204	0,8289	0,8232	0,8487	0,8473
Adult	0,5579	0,5873	0,5759	0,8641	0,8672	0,8682	0,8841	0,8888	0,8906
Ref - Chawla - Put, Undersampling = 0, Oversampling = 3, k = 3									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5291	0,5254	0,5852	0,6319	0,5745	0,5934	0,6179	0,5971
Hyper	0,9117	0,9789	0,9726	0,9196	0,9887	0,9888	0,9305	0,9946	0,9929
Fraud	0,4995	0,5478	0,5426	0,5365	0,5556	0,5759	0,5333	0,5676	0,5748
Shopping	0,6377	0,7272	0,7318	0,8516	0,8852	0,8921	0,8466	0,8899	0,8907
Ref - Chawla - Put, Undersampling = 1, Oversampling = 3, K = 3									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5752	0,6157	0,5852	0,6052	0,5716	0,5934	0,6211	0,5988
Hyper	0,9117	0,9834	0,9837	0,9196	0,9891	0,9866	0,9305	0,9926	0,9910
Fraud	0,4995	0,5482	0,5621	0,5365	0,5404	0,5651	0,5333	0,5473	0,5710
Shopping	0,6377	0,7079	0,7221	0,8516	0,8793	0,8903	0,8466	0,8843	0,8882

Table 30: Classification accuracy of real-world datasets with k = 1, 3

Ref - Chawla - Put, Undersampling = 0, Oversampling = 1, k = 5									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5161	0,5582	0,5852	0,6129	0,6356	0,5934	0,6131	0,5967
Hyper	0,9117	0,9647	0,9720	0,9196	0,9812	0,9773	0,9305	0,9927	0,9922
Fraud	0,4995	0,5415	0,5487	0,5365	0,5710	0,5709	0,5333	0,5707	0,5598
Shopping	0,6377	0,6947	0,7152	0,8516	0,8783	0,8782	0,8466	0,8789	0,8811
Churn	0,5260	0,5368	0,5642	0,7877	0,8247	0,8246	0,8232	0,8513	0,8496
Adult	0,5579	0,5817	0,5780	0,8641	0,8762	0,8756	0,8841	0,9002	0,8911
Ref - Chawla - Put, Undersampling = 1, Oversampling = 1, k = 5									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5935	0,5796	0,5852	0,6403	0,6099	0,5934	0,6278	0,6066
Hyper	0,9117	0,9854	0,9828	0,9196	0,9920	0,9909	0,9305	0,9930	0,9923
Fraud	0,4995	0,5556	0,5428	0,5365	0,5664	0,5579	0,5333	0,5678	0,5696
Shopping	0,6377	0,6881	0,6787	0,8516	0,8885	0,8844	0,8466	0,8853	0,8832
Churn	0,5260	0,5416	0,5630	0,7877	0,8288	0,8312	0,8232	0,8516	0,8459
Adult	0,5579	0,5780	0,5693	0,8641	0,8690	0,8814	0,8841	0,8966	0,9001
Ref - Chawla - Put, Undersampling = 0, Oversampling = 2, k = 5									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5872	0,5489	0,5852	0,5761	0,6274	0,5934	0,6339	0,5891
Hyper	0,9117	0,9683	0,9734	0,9196	0,9845	0,9824	0,9305	0,9940	0,9906
Fraud	0,4995	0,5517	0,5431	0,5365	0,5461	0,5790	0,5333	0,5688	0,5844
Shopping	0,6377	0,7151	0,7371	0,8516	0,8756	0,8855	0,8466	0,8825	0,8853
Churn	0,5260	0,5318	0,5630	0,7877	0,8218	0,8293	0,8232	0,8476	0,8492
Adult	0,5579	0,5853	0,5822	0,8641	0,8662	0,8691	0,8841	0,8872	0,8898
Ref - Chawla - Put, Undersampling = 1, Oversampling = 2, K = 5									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5939	0,5878	0,5852	0,5649	0,6219	0,5934	0,5861	0,6557
Hyper	0,9117	0,9828	0,9837	0,9196	0,9897	0,9908	0,9305	0,9906	0,9924
Fraud	0,4995	0,5238	0,5632	0,5365	0,5527	0,5698	0,5333	0,5498	0,5713
Shopping	0,6377	0,6933	0,6988	0,8516	0,8795	0,8905	0,8466	0,8836	0,8880
Churn	0,5260	0,5309	0,5597	0,7877	0,8196	0,8286	0,8232	0,8491	0,8482
Adult	0,5579	0,5808	0,5838	0,8641	0,8632	0,8651	0,8841	0,8863	0,8855
Ref - Chawla - Put, Undersampling = 0, Oversampling = 3, k = 5									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5010	0,5157	0,5852	0,6154	0,6058	0,5934	0,6264	0,5910
Hyper	0,9117	0,9644	0,9648	0,9196	0,9782	0,9764	0,9305	0,9920	0,9908
Fraud	0,4995	0,5411	0,5494	0,5365	0,5422	0,5780	0,5333	0,5696	0,5868
Shopping	0,6377	0,7251	0,7305	0,8516	0,8762	0,7864	0,8466	0,8781	0,8846
Ref - Chawla - Put, Undersampling = 1, Oversampling = 3, K = 5									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5954	0,6150	0,5852	0,6083	0,6178	0,5934	0,5919	0,6422
Hyper	0,9117	0,9816	0,9842	0,9196	0,9892	0,9904	0,9305	0,9918	0,9928
Fraud	0,4995	0,5425	0,5604	0,5365	0,5452	0,5644	0,5333	0,5541	0,5713
Shopping	0,6377	0,7171	0,7223	0,8516	0,8815	0,8916	0,8466	0,8818	0,8919
Ref - Chawla - Put, Undersampling = 0, Oversampling = 4, k = 5									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,4877	0,5008	0,5852	0,5668	0,5718	0,5934	0,5766	0,5776
Hyper	0,9117	0,9706	0,9739	0,9196	0,9879	0,9875	0,9305	0,9946	0,9948
Fraud	0,4995	0,5450	0,5518	0,5365	0,5348	0,5509	0,5333	0,5601	0,5659
Shopping	0,6377	0,7254	0,7264	0,8516	0,8706	0,8719	0,8466	0,8813	0,8763
Ref - Chawla - Put, Undersampling = 1, Oversampling = 4, K = 5									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5594	0,5501	0,5852	0,6038	0,5745	0,5934	0,6280	0,5650
Hyper	0,9117	0,9816	0,9855	0,9196	0,9873	0,9879	0,9305	0,9916	0,9945
Fraud	0,4995	0,5470	0,5457	0,5365	0,5339	0,5537	0,5333	0,5367	0,5635
Shopping	0,6377	0,7189	0,7432	0,8516	0,8779	0,8927	0,8466	0,8818	0,8879
Ref - Chawla - Put, Undersampling = 0, Oversampling = 5, k = 5									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,5529	0,5682	0,5852	0,5958	0,6195	0,5934	0,6171	0,6392
Hyper	0,9117	0,9648	0,9709	0,9196	0,9759	0,9816	0,9305	0,9908	0,9920
Fraud	0,4995	0,5500	0,5502	0,5365	0,5341	0,5386	0,5333	0,5642	0,5598
Ref - Chawla - Put, Undersampling = 1, Oversampling = 5, K = 5									
	KNN			Bagging			Random Forest		
Kickstarter	0,4759	0,6008	0,6026	0,5852	0,5867	0,5951	0,5934	0,6134	0,6310
Hyper	0,9117	0,9816	0,9833	0,9196	0,9855	0,9887	0,9305	0,9885	0,9907
Fraud	0,4995	0,5483	0,5417	0,5365	0,5220	0,5782	0,5333	0,5732	0,5718

Table 31: Classification accuracy of real-world datasets with k = 5