A Comparative Review of SMOTE and ADASYN in Imbalanced Data Classification

By Jakob Brandt & Emil Lanzén

Department of Statistics
Uppsala University

Supervisor: Patrik Andersson

2020

ABSTRACT

In this thesis, the performance of two over-sampling techniques, SMOTE and ADASYN, is compared. The comparison is done on three imbalanced data sets using three different classification models and evaluation metrics, while varying the way the data is pre-processed. The results show that both SMOTE and ADASYN improve the performance of the classifiers in most cases. It is also found that SVM in conjunction with SMOTE performs better than with ADASYN as the degree of class imbalance increases. Furthermore, both SMOTE and ADASYN increase the relative performance of the Random forest as the degree of class imbalance grows. However, no pre-processing method consistently outperforms the other in its contribution to better performance as the degree of class imbalance varies.

Keywords: Machine learning, supervised learning, classification, class imbalance, over-sampling, SMOTE, ADASYN, Sensitivity, F-measure, Matthews correlation coefficient

Contents

1 Introduction

1.1 Background

Classification is a domain within supervised statistical learning that deals with predicting qualitative responses and is used in a wide array of areas from predicting defaulting payments in banking to classifying individuals with illnesses. In this setting a learning model is trained on a set of training observations, $(x_i, y_i),\ \ i = 1, \dots, n,\ \ x_i \in \mathrm{R}^d,\ \ y_i \in \{-1, 1\}$, where x_i denotes the explanatory variables and y_i denotes the response variables. In the training data, the values of the response variables are used to learn a statistical model and to predict the classes of new observations. Different statistical learning models possess varying degrees of flexibility and are thus different in their prediction performance, although this may come with a trade-off with model interpretability (James et al., 2017, p.24-26). The statistical learning models in classification perform worse when the training data is imbalanced, and in many real-world applications the proportion of one class is much larger than the proportion of other classes. This results in the trained model being biased towards the majority group (Krawczyk, 2016). This is not necessarily due to an error in the sampling procedure, but rather an intrinsic property of the population of interest. For example, some illnesses occur for only a small minority of the population and oftentimes only a minority of individuals with debts default on their payments. As in these examples, a common setting is that of binary classification where one class, the majority class, is much larger than the minority class (Herrera et al., 2016). In these settings, the interest often lies in predicting the minority observations correctly, as misclassifications of such observations may bring large costs or other negative effects.

The class imbalance problem is present in a wide array of areas, and various approaches have been proposed to overcome the issue in the past decade, see Kaur et al. (2019) for a review. The methods of dealing with this issue can be partitioned into three subgroups: data level-methods, algorithm level-methods and hybrid methods (Krawczyk, 2016). Data level-methods, also known as pre-processing approaches, focus on the data space by reducing the imbalance between the majority and the minority class. Algorithm level-methods are based around the idea of upgrading existing learning algorithms or creating new ones that address imbalanced data. Hybrid methods combine data level-methods and algorithm level-methods. Generally, data level-methods balance the data by either removing observations from the majority class, known as under-sampling, or adding observations to the minority class, known as over-sampling. The simplest data level-methods are random over-sampling, in which the minority class is augmented by making exact copies of minority class observations, and random under-sampling in which observations of the majority class are removed at random (Han et al., 2005). While random under-sampling results in data that is computationally easier for learning compared to random over-sampling, its main drawback lies in the risk of losing informative observations of the majority class. Random over-sampling on the other hand introduces the issue of over-fitting, since this method replicates existing observations of the minority class. The data-level methods that have been proposed over the last two decades focus on addressing these issues. Kaur et al. (2019) and Ganganwar (2012) present reviews of several variations of these methods.

One of the earliest and most popular algorithms in over-sampling is the Synthetic Minority Over-sampling Technique (SMOTE) algorithm. It was proposed by Chawla et al. (2002) and adds synthetic minority class observations based on the k-nearest neighbors' algorithm for the minority class observations. Thus, more observations of the minority class are added to the data to even out the imbalance between the classes. One of the main problems of SMOTE is that it arbitrarily generates synthetic observations based on all the minority class observations. Therefore, the class boundaries between the majority class and the minority class after applying SMOTE can look very different from the original data and may not reflect the underlying distribution of the minority class. As a solution, He et al. (2008) addressed this problem by proposing the Adaptive Synthetic (ADASYN) sampling approach. The ADASYN algorithm generates synthetic observations according to the level of difficulty in learning particular minority class observations; more synthetic observations are generated for observations of the minority class that are relatively harder to learn.

Although ADASYN was designed to tackle some of the problems of SMOTE, the literature of comparisons between them does not unanimously favor either of the two. For example, Taneja et al. (2019) compared SMOTE and ADASYN, among other pre-processing methods, in conjunction with Random forest, Light Gradient Boosting and Extreme Gradient boosting on a single data set with a high degree of imbalance. Their results showed that the metrics from all models in conjunction with SMOTE outperformed ADASYN. Another analysis on a single data set with a high degree of imbalance was done by Barros et al. (2019) where SMOTE and ADASYN were used in conjunction with Decision trees and Multilayer perceptron to improve the performance of the classification. The performance of SMOTE and ADASYN were close to equal, although SMOTE outperformed ADASYN in more cases. In another study by Davagdorj et al. (2020) where SMOTE and ADASYN were used with seven different learning models on a single data set, the results were once again varied with SMOTE sometimes outperforming ADASYN and vice versa.

Although the authors in the above cited papers have been able to conclude that the particular data is best suited with a particular model and pre-processing method, there have been more mixed results in studies that have included more than one data set. For example, when He et al. (2008) compared the performance between SMOTE and

ADASYN on five data sets with varying degrees of imbalance, it was found that ADASYN outperformed SMOTE in almost all performance metrics for all sets of data. Although this pointed towards ADASYN performing better than SMOTE, the only learning model that was used was Decision trees. A more general result may have been obtained if more learning models would have been added to the analysis. Furthermore, the results did not speak for either ADASYN or SMOTE being better with respect to varying degrees of imbalance. In another comparison between SMOTE and ADASYN, among other pre-processing methods, Gosain & Sardana (2017) used more than one learning model when comparing the performance between the pre-processing methods. The models were the Support vector machine, the Naïve Bayes classifier and the Nearest Neighbors classifier, and the comparison included six different data sets with varying degrees of imbalance. The results showed that SMOTE outperformed ADASYN in a majority of the performance metrics for all models and all data sets. However, the differences were minor in many cases. Although the data sets had different degrees of imbalance, there was no clear pattern to be found in the performance of the pre-processing methods with respect to the data sets. As with the former study, Amin et al. (2016) also included several sets of data with varying degrees of imbalance as well as four different learning models. Their results did not show either SMOTE or ADASYN consistently outperforming the other in most evaluation metrics, as the results varied.

1.2 Purpose and research question

There is a lack of comparative studies with focus on SMOTE and ADASYN, and the existing experimental results does not consistently favor one method against another in all instances. In the cited studies where ADASYN and SMOTE have been compared, some authors have included a single data set while others have included multiple data sets. For example, Gosain & Sardana (2017) included a variety of learning models and data sets with varying degrees of imbalance, but the difference in degrees of imbalance between the data sets is not particularly large (ranging from 44 % to 30 % of the observations belonging to the minority class). He et al. (2008) on the other hand included data sets where the degrees of imbalance had a greater difference (ranging from 35 % to 5 % of the observations belonging to the minority class), but only a single learning model was included. Lastly, Amin et al. (2016) included a variety of learning models and multiple data sets with varying degrees of imbalance (ranging from around 27 % to 7 % of the observations belonging to the minority class), and the results did not point towards either SMOTE or ADASYN being consistently better than the other. However, in order to determine if SMOTE or ADASYN works relatively better when the degree of imbalance is varied, using data sets with larger differences in degrees of imbalance could help shed light on this issue. None of the authors have focused on the performance of SMOTE and ADASYN with respect to the degree of imbalance. Thus, a comparison between the pre-processing methods on data sets with very different degrees of imbalance together with more than a single model could be a relevant addition to the current literature on this issue. This could help gain a better understanding of whether SMOTE or ADASYN works relatively better than the other the higher the degree of imbalance there is. Furthermore, none of the cited articles have focused on if specific learning models benefit more from SMOTE or ADASYN than other models. It could be that some learning models benefit more from using SMOTE compared to ADASYN. As a result, a focus on the results of the pre-processing methods in conjunction with some canonical and often-used statistical learning models could shed light on this issue as well.

Thus, the purpose of this thesis is to compare suitable evaluation metrics for models of imbalanced data that have been pre-processed using SMOTE and ADASYN. Three different data sets have been selected that reflect different degrees of class imbalance.

The processed data sets will be classified by a set of canonical and common statistical learning models: the SVM, the random forest classifier, and the logistic regression classifier. The research question is as follows:

Are there any differences in performance of learning models after the data has been pre-processed by SMOTE or ADASYN? If so, are they mainly reflected by the choice of classifier and/or differing prior class imbalances?

The rest of the paper is organized as follows. Section 2 introduces and explains the data used in the paper. In Section 3, a theoretical background of the pre-processing methods and the classifiers are presented. Section 4 explains the method and chosen evaluation metrics. The results are presented in Section 5. Section 6 discusses the results, and the conclusion is given in Section 7.

2 Data

In this section, the data sets and the removal of observations and variables are described. The description of the data sets includes the number of observations, number of variables and the degree of class imbalance.

2.1 Description of data

The data consists of three data sets, all with binary response variables. They were retrieved from Kaggle.com and have different degrees of class imbalance. The first data set is called *Predicting Churn for Bank Customers* (Kumar, 2018), with the response variable being whether customers leave a bank. In this data set the minority class makes up 20.37 percent of all the observations. The second data set is called *Home Credit Default Risk* (Home Credit Group, 2018), with the response variable being whether a customer defaults on payments. In this data set the minority class makes up 8.07 percent of all the observations. The third data set is called *Credit Card Fraud Detection* (Machine Learning Group, 2018), with the response variable being whether a credit card transaction is fraudulent. In this data set the minority class makes up 0.17 percent of all the observations.

2.2 Data washing and dimensioning

The *Home Credit Default Risk* and the *Credit Card Fraud Detection* data sets have observations with missing values which were deleted prior to the pre-processing and the modeling. No analysis of missing values in the data has been done since interest lies in comparing models with unprocessed and processed data and not necessarily in making inference about the population. Thus, the comparison between them will be valid without an analysis of missing values. Furthermore, the number of variables of the *Home Credit Default Risk* and the *Predicting Churn for Bank Customers* were shrunk from 122 to 29 variables and from 14 to 11 variables, respectively. The rationale behind the deletion of variables was, apart from easing the computational burden, to remove inexplicit and vague variables. The descriptions and metadata of these variables gave reasons to believe they lacked meaning in predicting the minority class. For example, variables such as *surname*, *customer id* and *row number* were deemed to have this property. For the *Credit Card Fraud Detection* data set, all variables had names that lacked descriptions (*Var1*, *Var2*, etc.). Furthermore, no metadata of the variables were available. Thus, there was no way of determining their meaning in prediction, resulting in none of them being removed. Table 2.2 summarizes the data sets after removing missing values and non-relevant variables; a description of the response variables, the number of observations, the number of variables included and their class imbalance ratio. The class imbalance ratio reflects the proportion of minority class observations in relation to all observations.

Data set	Description of response variable	Number of observations	Number of variables	Class imbalance ratio
Predicting Churn for Bank Customers	Customers leaving a bank	10 000	11	20.37 %
Home Credit Default Risk	Defaulting payments	307 220	29	8.07 %
Credit Card Fraud Detection	Fraudulent credit card transactions	284 807	31	0.17 %

Table 2.2. — Description of data.

3 Theory

In this section the theory behind the thesis is presented. This includes the over-sampling techniques and the learning models used.

3.1 Over-sampling techniques

In this subsection, the theory behind the over-sampling techniques SMOTE and ADASYN are described. This includes short descriptions of how they work as well as corresponding pseudocodes in order to gain an understanding of how they generate new observations for the minority class.

3.1.1 Synthetic Minority Over-sampling Technique

SMOTE was proposed by Chawla et al. (2002) and generates synthetic observations for the minority class. For a given minority class observation, synthetic observations are generated in a random range between the observation and its k-nearest minority class neighbors. This procedure is done for every minority class observation. For SMOTE, the number of k-nearest neighbors is set to 5. The number of generated synthetic observations is specified prior to the procedure and should reflect the degree of imbalance. Figure 3.1.1 illustrates the generation of synthetic observations according to SMOTE on imbalanced data. The SMOTE algorithm in pseudocode is given below.

SMOTE Algorithm

Choose k and N, which denotes the number of nearest neighbors and the number of synthetic observations, respectively. Then

```
1. Let $x_i$, $i = 1, \dots, n_s$, denote the observations belonging to the minority class and
   let $A$ denote the set of all $x_i$, such that $A \ni x_i$. For every $x_i$:
2. Calculate the Euclidean distance between $x_i$ and all other elements of $A$ to
   obtain the k-nearest neighbors of $x_i$.
```

3. Let S_{ik} denote the set of the k-nearest neighbors of x_i.
4. Randomly sample N synthetic observations denoted $x_{ij}, (j = 1, \ldots, N)$ from S_{ik} with replacement.
5. Let λ denote a number in the range [0,1]. For a given x_{ij}, draw a λ uniformly and then generate a synthetic observation by the formula $x_k = x_i + \lambda\,(x_i - x_{ij})$.
6. Execute Step 5 for every x_{ij}.
7. Stop algorithm.

Here, $(x_i - x_{ij})$ is the difference vector in p-dimensional space, where p is the number of variables in the data.

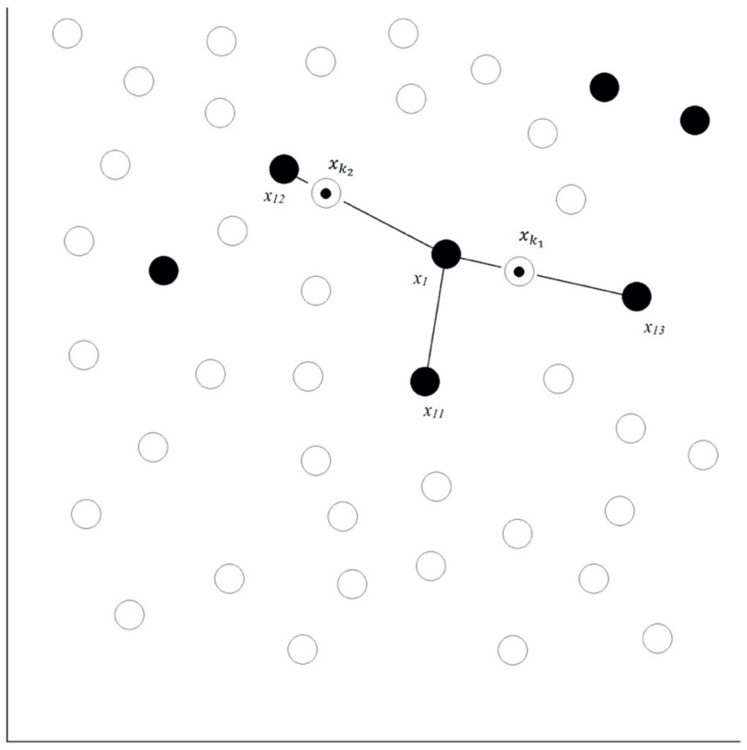

Figure 3.1.1. — Illustration of the SMOTE procedure on an imbalanced data set in two-dimensional space. For the minority observation x_1 with k = 3 and N = 2, the synthetic observations x_{k_1} and x_{k_2} are at a random distance along the straight line between the nearest neighbors.

3.1.2 Adaptive Synthetic sampling approach

ADASYN was proposed by He et al. (2008) and works similar to SMOTE in that it generates synthetic observations for the minority class. It is however based on generating more synthetic data for observations that are harder to learn than those that are easier to learn for a given model. As with SMOTE, ADASYN generates synthetic observations along a straight line between a minority class observation and its k-nearest minority class neighbors. As with SMOTE, the number of k-nearest neighbors is set to 5. However, ADASYN generates more synthetic observations for minority class observations which have more majority class observations inside the k-nearest neighbors' region. On the other hand, if a minority observation has no majority observations inside its k-nearest neighbor range, then no synthetic observations will be generated for this observation. The rationale lies in that these observations are harder for learning than minority observations that lie far from the majority observations. The ADASYN algorithm in pseudocode is given below.

ADASYN Algorithm

Choose k and β, which denote the number of nearest neighbors and the desired level of class balance after generating the synthetic data, respectively. Then

1. Let n_l denote the number of observations of the majority class and let n_s denote the number of observations of the minority class. Calculate $G = (n_l - n_s) \times \beta$.
2. Let x_i, $i = 1, \ldots, n_s$, denote the observations belonging to the minority class and let A denote the set of all x_i, such that $A \ni x_i$. For every x_i:
3. Calculate the Euclidean distance between x_i and all other elements of A to obtain the k-nearest neighbors of x_i.
4. Let S_{ik} denote the set of the k-nearest neighbors of x_i.

5. Define Δ_i as the number of observations in the k-nearest neighbors' region of x_i that belong to the majority class. Calculate the ratio r_i defined as $r_i = \Delta_i / k, i = 1, \ldots, n_s$.
6. Normalize r_i according to $\widehat{r_\imath} = r_i / \sum_{i=1}^{n_s} r_i$, so that $\widehat{r_\imath}$ is a probability $\left(\sum_{\mathrm{i}} \widehat{r_\imath} = 1\right)$.
7. Calculate $g_i = \widehat{r_\imath} \times G$, which is the number of synthetic observations that need to be generated for each x_i.
8. Randomly sample g_i synthetic observations denoted x_{ij},$(j = 1, \ldots, g_i)$ from S_{ik} with replacement.
9. Let λ denote a number in the range [0,1]. For a given x_{ij}, generate a synthetic observation according to $x_k = x_i + \lambda\ (x_i - x_{ij})$, where λ is uniformly drawn for each x_k.
10. Stop algorithm.

Here, $(x_i - x_{ij})$ is the difference vector in p-dimensional space, where p is the number of variables in the data. If $\beta = 1$ then the data set will be fully balanced after the procedure.

3.2 Learning models

In this subsection, the theoretical background of the learning models is presented.

3.2.1 Logistic regression

A suitable model in the classification setting is the Logistic regression model. According to Hair et al. (2010, p.341), it is widely preferred among researchers because of its simplicity, and in the often-present case where there is more than one explanatory variable, the model is called multiple logistic regression. This model is given by

$$p(Y_i|X_i, ..., X_p) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}. \tag{3.1}$$

Where $X = (X_1, \ldots, X_p)$ are p explanatory variables used to predict the response variable, Y. The above expression can be rewritten as

$$log\left(\frac{p(Y_i|X_i, ..., X_p)}{1-p(Y_i|X_i, ..., X_p)}\right) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p. \tag{3.2}$$

Where the left-hand side is the *logit function* of *p*. It can thus be seen that the multiple logistic regression model has a *logit* that is linear in the explanatory variables. An estimated multiple logistic regression model can thus be used to predict the probability of a given observation to belong to either one of the classes. The question is then to decide at what probability an observation should be classified as positive or negative in order to correctly classify new observations which the model has not been trained on. James et al. (2013, p.37-39) explain that the number of misclassifications on test data, called the *test error rate*, is minimized on average by assigning each observation to its most likely class, conditioned on the values of the explanatory variables. This classifier is called *Bayes classifier*. In other words, a test observation with explanatory vectors $x_1, \ldots, x_p$ should be assigned to the class j for which

$$p(Y = j|X_1 = x_1, ..., X_p = x_p) \tag{3.3}$$

is largest. In the binary setting, this corresponds to assigning a test observation to class 1 if

$$\tag{3.4}$$

$$p(Y = 1|X_1 = x_1, ..., X_p = x_p) > 0.5,$$

and to class -1 otherwise.

3.2.2 Random forest classifier

The random forest classifier is based on classification trees, that is decision trees used to predict qualitative responses. According to James et al. (2013, p.303-321), decision trees are made by dividing the predictor space $X_1, \ldots, X_p$, into J distinct and non-overlapping regions, $R_1, \ldots, R_j$. Then, the same prediction is made for every observation that falls into the region R_j, which in the classification setting is the majority group that occupies that specific region, which again can be regarded as Bayes classifier. The rule by which the predictor space is partitioned is called *recursive binary splitting*, in which the predictor space is split iteratively based on the highest reduction of some measure of classification error. More formally, consider the predictor X_j and the cutpoint *s*. Then *recursive binary splitting* is done by splitting the predictor space into the regions

$$\{X|X_j < s\} \;\; and \;\; \{X|X_j \geq s\}. \tag{3.5}$$

In the classification setting one measure that is often used for splitting is the *Gini index*, which is defined as

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}). \tag{3.6}$$

Here, $\hat{p}_{mk}$ is the proportion of training observations in the *m*th region that belong to the *k*th class.

Applying decision trees in the learning setting will likely lead to overfitting the data. An approach that addresses this issue is called *bagging*, which is a method that utilizes the bootstrap. In *bagging*, bootstrap samples from the training set are generated. Then, the model is trained on the individual bootstrapped training sets in order to get B classification functions/models

$$\hat{f}^{*1}(x), ..., \hat{f}^{*B}(x). \tag{3.7}$$

Finally, the predictions of all the models are averaged to obtain

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x). \tag{3.8}$$

Each individual tree has high variance but low bias. However, since the B trees are averaged the variance is reduced. In the classification setting, the simplest approach of predicting an observation is to record the class prediction by each of the B trees and take a majority vote, that is the overall class prediction is the most commonly occurring class among the B predictions.

The random forest is an extension of bagged trees that aims at further reducing the variance of the model. Averaging a set of classification models does not lead to a large reduction of variance if the models are highly correlated. Random forest produces less correlated models by only considering a subset of the predictors at each split. At each split a random sample of the m predictors is chosen as split candidates from the full set of p predictors. Typically, the number of predictors considered at each split is approximately equal to the square root of the total numbers of predictors, such that

$$m \approx \sqrt{p}. \tag{3.9}$$

This process decorrelates the trees, and thus the total variance of the averaged models will be reduced substantially for a trade with a small increase in bias.

3.2.3 Support vector machines

According to James et al. (2013, p.337-353), support vector machines (SVM) consist of a group of similar classifiers that have been shown to perform well in a wide variety of settings and have grown in popularity since their introduction in the 1990s. They are a generalization of the maximal margin classifier, which utilizes a separating hyperplane to classify observations. In a p-dimensional space, a hyperplane is a subspace of p-1 dimensions and is defined by the equation

$$\beta_0 + \sum_{i=1}^{p} \beta_i x_i = 0. \tag{3.10}$$

For example, if the feature space is of dimension 2 then the hyperplane is simply a straight line. Which side of the hyperplane an observation $x = (x_1, \dots, x_p)$ lies is determined by if

$$\beta_0 + \sum_{i=1}^{p} \beta_i x_i > 0, \quad or\ if \quad \beta_0 + \sum_{i=1}^{p} \beta_i x_i < 0. \tag{3.11}$$

If the hyperplane perfectly classifies the observations, then the hyperplane can be used as a classifier. However, if a data set can be separated using a hyperplane then there are in fact an infinite number of such hyperplanes. Of all the possible hyperplanes the maximal margin classifier chooses the hyperplane that lies the farthest from all the training observations. Hence, of all the training observations, there are a subset of these that lie on the margin of the hyperplane that determine its shape. These observations are called *support vectors.*

The maximal margin classifier works only if a separating hyperplane exists in the first place. In many settings, this is not true, where classes are non-separable by linear class boundaries. However, the concept of a separating hyperplane can be extended to correctly classify *most* observations using a so-called *soft margin.* The generalization of the maximal margin classifier to the non-separable case is known as the support vector classifier. This classifier has greater robustness to individual training observations and better classification of most of the training observations. The support vector classifier classifies training observations depending on which side of the hyperplane they lie and do so in a manner such that most of the observations are correctly classified at the cost of a few misclassifications. It is the solution to the following optimization problem:

$$\begin{gathered}\max_{\beta_0,\beta_1,\ldots,\beta_p,\ \epsilon_1,\ldots,\epsilon_n,\ M} M \\ \textit{subject to} \sum_{j=1}^{p} \beta_j^2 = 1, \\ y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} \geq M(1-\epsilon_i), \\ \epsilon_i \geq 0, \ \ \sum_{i=1}^{n} \epsilon_i \leq C,\end{gathered} \tag{3.12}$$

where C is a non-negative tuning parameter that controls the magnitude of violations of the hyperplane and the margins that are allowed. In practice, C is chosen via cross-validation and controls the variance-bias trade-off of the model. A higher value of this parameter allows for more violations of the hyperplane and makes it less likely to overfit the data — some added bias is traded for a smaller amount of variance of the model.

The support vector machine (SVM) is an extension of the support vector classifier. Although it is a linear classifier, it can accommodate non-linear class boundaries by expanding the feature space using functions known as *kernels.* The bridge to the SVM from the support vector classifier starts in the above optimization problem. It turns out that the solution to the problem involves only the *inner products* of the observations. The inner product of two observations $x_i, x_{i'}$ is given by

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}. \tag{3.13}$$

Thus, the linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle, \tag{3.14}$$

where α_i, $(i = 1, \ldots, n)$ are parameters, one per observation. In order to estimate the parameters of this model, one needs to compute the inner product of each new point x and each of the training points x_i. It turns out that the parameters are nonzero only for the support vectors. Now, suppose each time an inner product appears in the representation of the above expression, it is replaced with a generalization of the inner product of the following form

$$K(x_i, x_{i'}), \tag{3.15}$$

where K is a function that is referred to as a *kernel.* A kernel is a function that measures the similarity of two observations. The expression then becomes

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i). \tag{3.16}$$

Where S denotes the set of all observations that are support vectors. A popular choice of kernel is the radial basis kernel which is given by

$$K(x_i, x_{i'}) = \exp\left(-\gamma + \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right). \tag{3.17}$$

Where γ is a positive constant and is determined by cross validation.

4 Method

In this section, the method is presented. This includes the split of training and test sets, chosen over-sampling rates, model selection, choice of evaluation metrics and estimation of standard deviations and confidence intervals with bootstrap.

4.1 Training and test sets

The observations in the data sets were split into training sets and test sets. The split of observations was done randomly while keeping the class imbalances in the training and test sets the same as in the data sets prior to the split. The split in the *Predicting Churn for Bank Customers* data set was done such that 70 percent of the observations were included in the training set and the remaining 30 percent of the observations were included in the test set. This is equivalent to approximately 7000 observations in the training set and 3000 observations in the test set. The size of the training sets for the *Home Credit Default Risk* and the *Credit Card Fraud Detection* data sets were set to be roughly equal to the training set of the *Predicting Churn for Bank Customers* data set. The reason for this was to ease the computational burden of learning the models from these data sets, as they consist of a very large number of observations to begin with. The remaining observations were assigned to the test sets, resulting in approximately 300 000 and 278 000 observations for the *Home Credit Default Risk* data set and the *Credit Card Fraud Detection* data set, respectively. The classification models were then trained on the unprocessed training sets, training sets pre-processed using SMOTE and training sets pre-processed with ADASYN.

4.2 Over-sampling rates

The over-sampling rates were set so that the number of observations in the minority classes in each data set were as close as possible to equal to the number of observations in the majority classes. Although not all of the existing literature explicitly states the over-sampling rates used, the ones who do (Taneja et al., 2019; He et al., 2008 and Barros et al., 2019) all aim at perfectly balancing the classes. Moreover, since the reason for using the pre-processing methods is to shift the bias of the learning model (He & Ma, 2013, p.30), aiming at balanced classes in order to completely reduce the bias of the learning model against a particular class seems like a reasonable method from a heuristic perspective.

4.3 Model selection

For the choice of kernel of the SVM, a radial basis kernel was selected. Moreover, to determine the value of the hyperparameters in the model, a 5-fold cross validation was conducted in order to calculate the error for different values of the hyperparameters. In 5-fold cross validation, a given training set is randomly split into five sets that are as close as equal as possible. Then, a model is trained on four of these sets and then the classification error is calculated on the fifth set. The procedure is then repeated for all the sets until five in-sample errors have been calculated. The weighted in-sample error is then a good estimate of the out-of-sample error. Since the SVM contains two hyperparameters that cannot be determined by in-sample error minimization, the cross-validation procedure was repeated for different values of the parameters. The hyperparameters resulting in the lowest cross-validation error were then selected for the SVM. An illustration of the procedure is given in Table 4.3.

Both the Logistic regression and the Random forest classifier contain no hyperparameters, and thus cross-validation was not used for these models. For the Random forest classifier however, the choice of the number of bagged trees and number of allowed variables at each split had to be set. The number of trees were set to 500, which seemed sufficiently large in order for the error rate to decrease. According to James et al. (2013, p.321) the Random forest does not overfit when the number of trees is increased. The allowed number of randomly selected variables was set to the square root of the total number of explanatory variables, which according to James et al. (2013, p.319) is a typical choice.

Train	Train	Train	Train	Validation
Train	Train	Train	Validation	Train
Train	Train	Validation	Train	Train
Train	Validation	Train	Train	Train
Validation	Train	Train	Train	Train

Table 4.3. — Illustration of 5-fold cross-validation.

4.4 Evaluation metrics

In this section, the tools and metrics used in order to evaluate the model outputs are presented. The used evaluation metrics have been selected to reflect the ability of a classifier to classify the minority observations correctly as well as the classifiers overall classification performance.

4.4.1 Confusion Matrix

A confusion matrix is a table that presents the output of a classifier in terms of predicted and actual values, where values refer to class labels. On one axis lies the actual values and on the other axis lies the predicted values. In the binary classification setting where the response variable belongs to class -1 or 1, i.e., $y_i \in \{-1,1\}$ the confusion matrix is a two-by-two table. Moreover, an observation that belongs to class -1 is referred to as a *negative* and an observation that belongs to class 1 is referred to as a *positive.* An observation that truly belongs to class -1 and that the model correctly predicted as belonging to this class is referred to as a *true negative.* In the opposite case, where an observation belongs to class 1 and the model correctly predicted as belonging to this class is referred to as a *true positive*. If the model predicts that an observation is a positive but in fact is a negative, then that observation is called a *false positive*. In the opposite case, where the model predicts that an observation is a negative but in fact is a positive, that observation is called a *false negative* (Burkov, 2019). For the rest of this paper, observations referred to as positive are synonymous to observations belonging to the minority class. A confusion matrix is depicted in Table 4.4.1.

		Actual values	
‘Positive’ class: 1		**-1**	**1**
Predicted values	**-1**	True negative	False negative
	1	False positive	True positive

Table 4.4.1. — Confusion matrix presenting the output of a classifier in terms of actual and predicted values, where -1 and 1 refer to class labels. True/false represent whether the classifier predicted an observation correctly/incorrectly. Positive/negative refers to the class label, -1 or 1, of the predicted observation.

4.4.2 Sensitivity

Sensitivity is a measurement of the degree to which positive observations are correctly classified. It is defined as the fraction of true positives in relation to the total number of observations belonging to the positive class. The formula for sensitivity is given by

$$Sensitivity = \frac{True\ positives}{True\ positives + False\ negatives}. \tag{4.1}$$

This evaluation metric is suitable when dealing with imbalanced data, since it measures how well the model can correctly classify observations belonging to the minority group (Kaur et al., 2019).

4.4.3 F-measure

The *F-measure* is commonly used to evaluate model performance when data is imbalanced, and combines *precision* and *sensitivity* in its measurement,

$$F - measure = \frac{2 \times Precision \times Sensitivity}{Precision + Sensitivity}. \tag{4.2}$$

Precision measures the ratio between positive observations being correctly classified and all observations being classified as positive,

$$Precision = \frac{True\ positives}{True\ positives + False\ positives}. \tag{4.3}$$

It is easy to see the similarity between precision and sensitivity. While sensitivity measures the degree to which a model correctly classifies all truly positives correctly, precision measures the degree to which a model correctly classifies positives in relation to all observations it classifies as positives. They thus account for different misclassifications of a model, and the F-measure is the harmonic mean of sensitivity and precision (Kaur et al., 2019). The rationale of the choice of this metric lies in its widespread usage when comparing models and its intuitiveness when comparing it to sensitivity: If the F-measure for a model is higher than its sensitivity then the precision is higher than the sensitivity, and vice versa. It is thus, along with sensitivity, a useful measure when evaluating the outputs of different classification models.

4.4.4 Matthews correlation coefficient

The *Matthews correlation coefficient* (MCC) is a measure used to evaluate the output of classification models, often in the binary classification case. It is a method for calculating the *Pearson correlation coefficient* between actual and predicted values in a contingency table, such as the confusion matrix. Although it is not as widely used as the F-measure, its usefulness has been shown in different scientific fields when it comes to evaluation of predictive models (Liu et al., 2015; The MAQC Consortium, 2010). Chicco & Jurman (2020) argue that the MCC should be preferred to the F-measure by all scientific communities when evaluating binary classification models. The reason is that the MCC generates results that reflect the overall predictions made by a model, which is not the case for the F-measure. It can be shown that when a model predicts only one of the two classes well (i.e., displaying many true positives but few true negatives or vice versa), the F-measure can give a misleading result while the MCC will not. The MCC is given by the following formula:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}. \tag{4.4}$$

As the Pearson's correlation coefficient, the MCC returns a value in the range [-1, +1]. A value of -1 or 1 is returned in the case of perfect misclassification or perfect classification, respectively. A value equal to zero is the expected value of a classifier based on coin tossing. The MCC will be used in conjunction with the F-measure and sensitivity when evaluating the models. The reason for not only using the MCC is to include metrics that give an idea at how the model performs in terms of predicting the positive class only. The MCC will give an idea of the overall classification performance of the model.

4.5 Bootstrap estimation of standard deviation

After training the models on the training sets, they will be used to classify the observations in the test sets. Afterwards, the predicted values will be bootstrapped into sets of bootstrap replicates. The replicates will then be compared to the actual values in order to extract the evaluation metrics from each comparison. The reason for this procedure is to obtain a spread of metrics in order to calculate the standard deviations and 95% confidence intervals of the metrics. The confidence intervals will be used in order to determine whether the results are statistically significant, where non-overlapping confidence intervals indicate a significant difference. Although bootstrapping the training sets would make it possible to calculate the standard deviation of the estimated models, this option is not viable since the computation time is too long. Instead of being able to calculate the standard deviation and confidence interval of the estimated models, the procedure implemented here will arrive at an estimated standard deviation and confidence interval of the evaluation metrics for each set of predicted values holding the trained models fixed.

5 Results

In this section, the results of using the pre-processing methods on the minority class proportions of the training sets are presented. Furthermore, the sensitivity, F-measures and MCC's are presented using combinations of pre-processing methods and models for the different data sets, whilst the confidence intervals are presented in the Appendix. The class imbalance for each data set without any pre-processing, after implementing SMOTE and after implementing ADASYN can be seen in Table 5.1.

5.1 Churn Modeling Data Set

The sensitivity, F-measure and MCC for each combination of pre-processing method and classification model is presented in Table 5.1.1. The bootstrap standard deviations of the metrics are within parentheses. SVM in conjunction with SMOTE or ADASYN performs better in terms of sensitivity compared to SVM without any pre-processing. Although the highest sensitivity is achieved using SVM in conjunction with SMOTE, the value is not significantly different than the value achieved using SVM in conjunction with ADASYN. Regarding the F-measure and MCC, there are no significant differences amongst the values for the different pre-processing methods when applying the SVM.

Name	**No pre-processing**	**SMOTE**	**ADASYN**
Predicting Churn for Bank Customers	20.37%	50.27%	50.71%
Home Credit Default Risk	8.08%	48.40%	49.76%
Credit Card Fraud Detection	0.19%	50.03%	49.98%

Table 5.1. — Class imbalance ratio of the original training sets and after the implementation of over-sampling techniques.

Logistic regression together with SMOTE or ADASYN gave higher sensitivity and F-measure compared to the Logistic regression with no pre-processing method, whilst the values when using SMOTE and ADASYN are not significantly different from each other. There are no significant differences amongst the MCC's when using the Logistic regression. When using the Random forest in conjunction with the different pre-processing methods, none of the differences between the evaluation metrics are statistically significant.

Churn Modeling			
Method	**Sensitivity** (SD)	**F-measure** (SD)	**MCC** (SD)
SVM	0.3944 (0.0121)	0.5373 (0.0156)	0.5150 (0.0188)
SMOTE + SVM	**0.5552 (0.0154)**	0.5932 (0.0152)	0.4970 (0.0183)
ADASYN + SVM	**0.5280 (0.0155)**	0.5670 (0.0153)	0.4655 (0.0183)
LR	0.2079 (0.0105)	0.3064 (0.0147)	0.2633 (0.0184)
SMOTE + LR	**0.6912 (0.0198)**	**0.4884 (0.0125)**	0.3273 (0.0182)
ADASYN + LR	**0.6992 (0.0193)**	**0.4913 (0.0122)**	0.3316 (0.0180)
RF	0.4714 (0.0132)	0.5944 (0.0157)	0.5491 (0.0188)
SMOTE + RF	0.4944 (0.0137)	0.5836 (0.0151)	0.5100 (0.0182)
ADASYN + RF	0.4896 (0.0140)	0.5812 (0.0154)	0.5089 (0.0183)

Table 5.1.1. — Sensitivity, F-measure, and Matthews correlation coefficient of the models using different pre-processing methods on the Churn Modeling data set. Highest values are marked in bold. If two values are marked in bold, no statistically significant difference exists between them. No bold values indicate that there is no significant difference amongst the values.

5.2 Home Credit Group Data Set

The sensitivity, F-measure and MCC for each combination of pre-processing method and classification model is presented in Table 5.2.1. The bootstrap standard deviations of the metrics are within parentheses. As shown, Random forest has the overall worst performance as a model compared to SVM and Logistic regression for almost all metrics. Furthermore, there are no significant differences for the evaluation metrics comparing SMOTE and ADASYN when using the SVM. The implementation of ADASYN shows the largest increase in sensitivity when applying the Logistic regression, whilst SMOTE performs better with regards to the F-measure when applying the Logistic regression which also applies to the sensitivity and F-measure when applying the Random forest. The MCC for the Random forest is the only case where neither SMOTE nor ADASYN shows a significant improvement.

Home Credit Group			
Method	**Sensitivity** (SD)	**F-measure** (SD)	**MCC** (SD)
SVM	0.0000	0.0000	0.0000
SMOTE + SVM	**0.1072 (0.0016)**	**0.1130 (0.0017)**	**0.0396 (0.0018)**
ADASYN + SVM	**0.1062 (0.0017)**	**0.1124 (0.0017)**	**0.0394 (0.0018)**
LR	0.0000	0.0000	0.0000
SMOTE + LR	0.5891 (0.0030)	**0.2057 (0.0010)**	**0.1264 (0.0018)**
ADASYN + LR	**0.6238 (0.0031)**	0.2009 (0.0010)	**0.1218 (0.0018)**
RF	0.0000	0.0000	-0.0009 (0.0017)
SMOTE + RF	**0.0009 (0.0001)**	**0.0018 (0.0002)**	0.0095 (0.0018)
ADASYN + RF	0.0003 (0.0001)	0.0006 (0.0001)	0.0047 (0.0019)

Table 5.2.1. — Sensitivity, F-measure, and Matthews Correlation Coefficient of the models using different pre-processing methods on the Home Credit Group data set. Highest values are marked in bold. If two values are marked in bold, no statistically significant difference exists between them. No bold values indicate that there is no significant difference amongst the values.

5.3 Credit Card Fraud Data Set

The sensitivity, F-measure and MCC for each combination of pre-processing method and classification model is presented in Table 5.3.1. The bootstrap standard deviations of the metrics are within parentheses. SVM in conjunction with SMOTE or ADASYN improves all metrics, whilst the largest improvement is achieved using SMOTE. However, both SMOTE and ADASYN worsen the performance when applying the Logistic regression. When applying the Random forest, the sensitivity is not significantly different when using SMOTE or ADASYN. The highest F-measure is achieved using ADASYN. Although Logistic regression with no pre-processing gave the highest sensitivity amongst all models and pre-processing methods, the highest F-measure was obtained using the Random forest in conjunction with ADASYN. Lastly, there are no significant differences amongst the MCC's for the Random forest.

Credit Card Fraud			
Method	**Sensitivity** (SD)	**F-measure** (SD)	**MCC** (SD)
SVM	0.1587 (0.0008)	0.2705 (0.0010)	0.3808 (0.0019)
SMOTE + SVM	**0.5708 (0.0016)**	**0.6716 (0.0014)**	**0.6818 (0.0019)**
ADASYN + SVM	0.5417 (0.0016)	0.6468 (0.0014)	0.6588 (0.0019)
LR	**0.7704 (0.0022)**	**0.6514 (0.0015)**	**0.6586 (0.0019)**
SMOTE + LR	0.6688 (0.0034)	0.3228 (0.0015)	0.3755 (0.0019)
ADASYN + LR	0.5063 (0.0023)	0.4057 (0.0016)	0.4127 (0.0019)
RF	0.6889 (0.0017)	0.7719 (0.0014)	0.7773 (0.0019)
SMOTE + RF	**0.7188 (0.0018)**	0.7735 (0.0015)	0.7754 (0.0019)
ADASYN + RF	**0.7146 (0.0018)**	**0.7795 (0.0015**)	0.7824 (0.0019)

Table 5.3.1. — Sensitivity, F-measure, and Matthews Correlation Coefficient of the models using different pre-processing methods on the Credit Card Fraud data set. Highest values are marked in bold. If two values are marked in bold, no statistically significant difference exists between them. No bold values indicate that there is no significant difference amongst the values.

6 Discussion

When applying SVM to the data sets, all metrics are improved after implementing SMOTE or ADASYN, except for the F-measure and the MCC on the Churn Modeling data set where there are no significant differences. The increase in performance is larger when using SMOTE compared to using ADASYN on the Credit Card Fraud data set, whilst there are no significant differences in performance in any of the other cases. Thus, both pre-processing methods result in better performance in most cases. Comparing the two pre-processing methods, there does seem to be a pattern that speaks in favor of SMOTE when applying the SVM, especially as the degree of imbalance increases. However, in the majority of scenarios the differences are non-significant.

When applying the Logistic regression to the Churn Modeling data set, both the sensitivity and F-measure are improved after implementing SMOTE or ADASYN. When applying the Logistic regression to the Home Credit Group data set, both ADASYN and SMOTE improve the performance of the classification models, although the improvements from ADASYN are greater with regards to sensitivity, whilst the implementation of SMOTE results in a larger increase in performance with regards to the F-measure. There is no significant difference in performance increase of the MCC between the two. When applying the Logistic regression to the Credit Card Fraud data set, both SMOTE and ADASYN worsen the performance of the metrics. Thus, the results point to that both ADASYN and SMOTE in conjunction with the Logistic regression seem to have an equally large potential of improving the metrics. However, this does not hold for all cases as the metrics from the Credit Card Fraud data set speak against this claim. Lastly, the results from using the Logistic regression does not speak for ADASYN and/or SMOTE being relatively better the more imbalance there is.

When applying the Random forest to the Churn Modeling data set, there are no significant differences between the metrics for the pre-processing methods, whilst an increase of the sensitivity and F-measure can be identified when implementing SMOTE on the Home Credit Group data set. For the Credit Card Fraud data set, both SMOTE and ADASYN improve the performance of the classifier with regards to sensitivity, whilst the F-measure is improved only by using Random forest in conjunction with ADASYN. The results speak for ADASYN and SMOTE being relatively better the more imbalance there is for the Random forest. However, there does not seem to be a pattern that speaks in favor of either SMOTE or ADASYN being consistently better than the other.

The confidence intervals calculated using the bootstrap are used in order to determine whether the results are statistically significant. However, a limitation of the standard deviations and confidence intervals is that they are derived from bootstrap replicates of the predicted values of the test sets. A more informative procedure would have been to bootstrap the training sets and train the models on every replicate of the training sets. The reason for not choosing this method was the computational time required to implement this. One of the limitations of this paper lies in in this matter.

Another matter that is subject to discussion are the over-sampling rates. This paper aimed at setting the classes of the processed training sets to as close as equal class balances. The rationale was that the models were assumed to benefit the most if the classes were perfectly balanced. However, there might be a risk associated with this procedure since this may introduce substantial overlapping between the classes. Even though SMOTE and ADASYN aims at making the class boundaries more present and easier for the learning models to distinguish between, there may be cases where a substantial over-sampling of the minority class leads to synthetic observations in the data space that are placed in ill-fitted locations. The minority class observations may not always be located in clusters more or less close to each other but may be more scattered in the data space. The implementation of an over-sampling technique such as SMOTE or ADASYN will then create synthetic observations in locations between observations where majority observations are located. Then, the learning models may have an even harder time to correctly distinguish between the classes than without any pre-processing, leading to a deterioration of the metrics as seen in some cases in the results. Following this hypothetical example, ADASYN which is meant to create synthetic observations for the observations that are harder to learn from, may generate synthetic observations for outlier minority observations that would have been 'ignored' by the learning model. SMOTE would not have generated the same number of observations for the outlier observation, leading to better results from this pre-processing method. Thus, lower levels of over-sampling rates might be more adequate than an aim of full balance, in some cases. In other cases, implementing other over-sampling techniques such as Borderline-SMOTE I or Borderline-SMOTE II may reduce the risk of over-sampling when outliers or scattered minority observations are present (Han et al. 2005).

7 Conclusion

Of all the models, the results suggest that there is no pre-processing method that consistently improves the performance of all the models with regards to sensitivity, the F-measure and the MCC. The results do however point towards that using SMOTE consistently improves the performance of the SVM in most cases, especially as the degree of imbalance increases. Furthermore, the results point towards that both pre-processing methods improve the relative performance of the Random forest the more imbalance there is in the original data set. However, there does not seem to be a pattern that speaks in favor of either SMOTE or ADASYN being consistently better than the other. That is, no pre-processing method consistently outperforms the other in its contribution to better performance as the degree of class imbalance varies. Both SMOTE and ADASYN improve the overall performance of the models. However, this does not apply to every case, as they sometimes give similar results as the models of the unprocessed data or may even worsen the performance.

Reference list

Amin, Adnan; Anwar, Sajid; Adnan, Awais; Nawaz, Muhammad; Howard, Newton; Qadir, Junaid; Hawalah, Ahmad; Hussain, Amir. 2016. Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study. *IEEE Access* 4. 7940-7957.

Barros, Thiago M.; Souza Neto, Plácido A.; Silva, Ivanovitch; Guedes, Luiz Affonso. 2019. Predictive Models for Imbalanced Data: A School Dropout Perspective. *Education sciences* 9(4). 275-292.

Burkov, Andriy. 2019. *The Hundred-Page Machine Learning Book.* Hardcover edn. Andriy Burkov, 2019.

Chawla, Nitesh V.; Bowyer, Kevin W.; Hall, Lawrance O.; Kegelmeyer, W. Philip. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artifical Intelligence Research* 16. 321-357.

Chicco, Davide; Jurman, Giuseppe. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* 21(6). 6-19.

Davagdorj, Khishigsuren; Lee, Jong Seol; Pham, Van Huy; Ryu, Keun Ho. 2020. A Comparative Analysis of Machine Learning Methods for Class Imbalance in a Smoking Cessation Intervention. *Applied sciences* 10(9). 3307-3327.

Ganganwar, Vaishali. 2012. An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering* 2. 42-47.

Gosain, Anjana; Sardana, Saanchi. 2017. Handling class imbalance problem using oversampling techniques: A review. *ICACCI: 2017 International Conference on Advances in Computing, Communications and Informatics.* Udupi, India. 79-85.

Hair, Joseph F.; Black, William C.; Babin, Barry J; Anderson, Rolph E. 2010. *Multivariate Data Analysis: A Global Perspective.* 7th edn. New Jersey: Pearson Education, Inc.

Han, Hui; Wang, Wen-Yuan; Mao, Bing-Huan. 2005. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In: Huang, De-Shuang;

Zhang, Xiao-Ping; Huang, Guang-Bin. (eds) *Advances in Intelligent Computing. ICIC 2005. Lecture Notes in Computer Science* 3644(5). 878-887.

He, Haibo; Bai, Yang, Garcia, Edwardo A.; Li, Shutao. 2008. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. *IEEE World Congress on Computational Intelligence: 2008 IEEE International Joint Conference on Neural Networks*. Hong Kong, China, 1322-1328.

He, Haibo; Ma, Yunqian. 2013. *Imbalanced Learning: Foundations, Algorithms, and Applications*. 1st edn. New Jersey: John Wiley & Sons, Inc.

Herrera, Francisco; Ventura, Sebastián; Bello, Rafael; Cornelis, Chris; Zafra, Amelia; Sánchez-Tarrago, Dánel; Vluymans, Sarah. 2016. *Multiple Instance Learning: Foundations and Algorithms*. 1st edn. New York: Springer.

Home Credit Group. 2018. *Home Credit Default Risk*. Retrieved November 17, 2020 from https://www.kaggle.com/c/home-credit-default-risk.

James, Gareth; Witten, Daniela; Hastie, Trevor; Tibshirani, Robert. 2013. *An Introduction to Statistical Learning*. 8th edn. New York: Springer.

Kaur, Harsurinder; Husanbir, Pannu; Avleen, Malhi. 2019. A Systematic Review on Imbalanced Data Challenges in Machine Learning: Applications and Solutions. *ACM Computing Surveys* 52(4). 1-36.

Krawczyk, Bartosz. 2016. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence* 5(4). 221-232.

Kumar, Santosh. 2018. *Bank Customers Churn*, Version 1. Retrieved November 17, 2020 from https://www.kaggle.com/santoshd3/bank-customers.

Liu, Yingbo; Cheng, Jiujun; Yan, Chendan, Wu, Xiao; Chen, Fuzchen. 2015. Research on the Matthews correlation coefficients metrics of personalized recommendation algorithm evaluation. *International Journal of Hybrid Information Technology* 8(1). 163-172.

Machine Learning Group. 2018. *Credit Card Fraud Detection*, Version 3. Retrieved November 17, 2020 from https://www.kaggle.com/mlg-ulb/creditcardfraud.

Taneja, Schweta; Suri, Bhawna; Kothari, Chirag. 2019. Application of Balancing Techniques with Ensemble Approach for Credit Card Fraud Detection. *GUCON: 2019*

International Conference on Computing, Power and Communication Technologies. New Delhi, India. 753-758.

The MicroArray Quality Control (MAQC) Consortium. 2010. The MAQC-II Project: a comprehensive study of common practices for the development and validation of microarray-based predictive models. *Nature Biotechnology* 28(8). 827-838.

Appendix

Churn Modeling			
Method	**Sensitivity** (CI)	**F-measure** (CI)	**MCC** (CI)
SVM	(0.3729, 0.4188)	(0.5085, 0.5692)	(0.4807, 0.5530)
SMOTE + SVM	(0.5256, 0.5848)	(0.5636, 0.6229)	(0.4609, 0.5328)
ADASYN + SVM	(0.4971, 0.5595)	(0.5361, 0.5976)	(0.4297, 0.5017)
LR	(0.1875, 0.2299)	(0.2782, 0.3364)	(0.2281, 0.3002)
SMOTE + LR	(0.6538, 0.7290)	(0.4642, 0.5124)	(0.2918, 0.3624)
ADASYN + LR	(0.6617, 0.7369)	(0.4670, 0.5149)	(0.2963, 0.3667)
RF	(0.4454, 0.4978)	(0.5639, 0.6251)	(0.5137, 0.5858)
SMOTE + RF	(0.4684, 0.5228)	(0.5539, 0.6147)	(0.4748, 0.5471)
ADASYN + RF	(0.4621, 0.5181)	(0.5513, 0.6120)	(0.4736, 0.5453)

Table 1.1. — 95% confidence intervals of the sensitivity, F-measure, and Matthews Correlation Coefficient of the models using different pre-processing methods on the Churn Modeling data set.

Home Credit Group			
Method	**Sensitivity** (CI)	**F-measure** (CI)	**MCC** (CI)
SVM	0	0	0
SMOTE + SVM	(0.1038, 0.1104)	(0.1095, 0.1162)	(0.0361, 0.0431)
ADASYN + SVM	(0.1030, 0.1095)	(0.1092, 0.1158)	(0.0359, 0.0429)
LR	0	0	0
SMOTE + LR	(0.5832, 0.5948)	(0.2038, 0.2077)	(0.1229, 0.1299)
ADASYN + LR	(0.6178, 0.6295)	(0.1990, 0.2027)	(0.1183, 0.1253)
RF	0	0	(-0.0005, 0.0049)
SMOTE + RF	(0.0007, 0.0012)	(0.0013, 0.0022)	(0.0062, 0.0132)
ADASYN + RF	(0.0001, 0.0006)	(0.0004, 0.0009)	(0.0017, 0.0091)

Table 1.2. — 95% confidence intervals of the sensitivity, F-measure, and Matthews Correlation Coefficient of the models using different pre-processing methods on the Home Credit Group data set.

Credit Card Fraud			
Method	**Sensitivity** (CI)	**F-measure** (CI)	**MCC** (CI)
SVM	(0.1568, 0.1599)	(0.2700, 0.2739)	(0.3800, 0.3855)
SMOTE + SVM	(0.5686, 0.5738)	(0.6707, 0.6757)	(0.6804, 0.6856)
ADASYN + SVM	(0.5397, 0.5447)	(0.6459, 0.6509)	(0.6573, 0.6626)
LR	(0.7679, 0.7763)	(0.6501, 0.6555)	(0.6564, 0.6635)
SMOTE + LR	(0.6633, 0.6758)	(0.3209, 0.3260)	(0.3723, 0.3795)
ADASYN + LR	(0.5022, 0.5075)	(0.4013, 0.4066)	(0.4112, 0.4165)
RF	(0.6875, 0.6938)	(0.7710, 0.7758)	(0.7756, 0.7826)
SMOTE + RF	(0.7163, 0.7236)	(0.7726, 0.7772)	(0.7738, 0.7806)
ADASYN + RF	(0.7121, 0.7195)	(0.7785, 0.7832)	(0.7809, 0.7877)

Table 1.3. — 95% confidence intervals of the sensitivity, F-measure, and Matthews Correlation Coefficient of the models using different pre-processing methods on the Credit Card Fraud data set.