2020

An Examination of the Smote and Other Smote-based Techniques That Use Synthetic Data to Oversample the Minority Class in the Context of Credit-Card Fraud Classification

Eduardo Parkinson de Castro
Technological University Dublin

Recommended Citation

Parkinson de Castro, E. (2020). *An examination of the smote and other smote-based techniques that use synthetic data to oversample the minority class in the context of credit-card fraud classification*. Masters Dissertation. Technological University Dublin. DOI:10.21427/wj33-n221

AN EXAMINATION OF THE SMOTE AND OTHER SMOTE-BASED TECHNIQUES THAT USE SYNTHETIC DATA TO OVERSAMPLE THE MINORITY CLASS IN THE CONTEXT OF CREDIT-CARD FRAUD CLASSIFICATION.

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH

DUBLIN

TECHNOLOGICAL UNIVERSITY DUBLIN

Eduardo Parkinson de Castro

A dissertation submitted in partial fulfilment of the requirements of Technological University Dublin for the degree of M.Sc. in Computer Science (Data Analytics)

January 2020

DECLARATION

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Analytics) is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technological University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: Eduardo P. de Castro

Date: **27/January/2020**

ABSTRACT

This research project seeks to investigate some of the different sampling techniques that generate and use synthetic data to oversample the minority class as a means of handling the imbalanced distribution between non-fraudulent (majority class) and fraudulent (minority class) classes in a credit-card fraud dataset. The purpose of the research project is to assess the effectiveness of these techniques in the context of fraud detection which is a highly imbalanced and cost-sensitive dataset.

Machine learning tasks that require learning from datasets that are highly unbalanced have difficulty learning since many of the traditional learning algorithms are not designed to cope with large differentials between classes. For that reason, various different methods have been developed to help tackle this problem. Oversampling and undersampling are examples of techniques that help deal with the class imbalance problem through sampling.

This paper will evaluate oversampling techniques that use synthetic data to balance the minority class. The idea of using synthetic data to compensate for the minority class was first proposed by (Chawla et al., 2002). The technique is known as Synthetic Minority Over-Sampling Technique (SMOTE). Following the development of the technique, other techniques were developed from it. This paper will evaluate the SMOTE technique along with other also popular SMOTE-based extensions of the original technique.

Key words: *SMOTE, Class-Imbalance, Sampling, Oversampling, Machine Learning, Classification, Credit-card Fraud*

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere thanks to Dr. Aneel Rahim for his guidance and mentoring throughout the course of this dissertation.

I would also like to express my gratitude to all my professors who assisted me throughout my time in the university.

Lastly, I would like to thank my family and friends who always stood behind me, supporting and encouraging me.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

Synthetic Minority Oversampling Technique (SMOTE) is an oversampling technique that generates synthetic data examples of the minority class. It has been shown to be more effective as an oversampling technique than random oversampling due to its ability to resolve certain problematic aspects associated with random re-sampling. The technique was first introduced by Chawla (2003). Since its publication, more than 100 new variants have been introduced. Extensions and variants of the technique have been developed in order to help improve the performance of the algorithm under different circumstances.

In this research project, we review some of the extensions to the original SMOTE technique and evaluate how they compare to other oversampling techniques and whether they can be useful for credit-card fraud classification.

1.1 Background

In the 1990s, as data mining and machine learning technologies started to become more prevalent, the challenge became evident. How do we maximize the accuracy of a classifier when the dataset is particularly imbalanced, and the distribution of classes is skewed? (Sun et al., 2009; He & Garcia, 2009, López et al., 2013; Branco et al., 2016; Cieslak et al., 2012, Hoens & Chawla, 2013; Lemaitre et al., 2017)

It became noticeable that across several different disciplines, a similar trend became reoccurring for standard classification models that dealt with datasets that exhibited a class imbalance problem (Anand, Mehrotra, Mohan, & Ranka, 1993; Bruzzone & Serpico, 1997; Kubat, Holte, & Matwin, 1998). In many cases, the local accuracy of the majority class (i.e., the specificity) was significantly higher than the local accuracy of the minority class. This led to the creation of an active area of research in the machine learning community called "learning from imbalanced data" from whence the term 'class imbalance' was coined.

The first significant milestone for the 'Class-Imbalance Problem' is marked by the first workshop ever held in its name. This happened during the American Association for Article Intelligence Conference in the early 2000s (Japkowicz & Holtz, 2000). The next significant milestone for this problem happened in 2003 during the ICML-KDD Workshop on learning from imbalanced datasets, which presented the first-ever issue exclusively dedicated to the topic (Chawla, Japkowicz, & Kolcz, 2004).

This problem continues to be relevant since research is, in significant part, driven by the many challenges that arise from within the various different areas of application. For example, in face recognition, software engineering, social network, medical diagnosis, and more. (Krawczyk, 2016; Haixiang et al., 2017; Maua & Galinac Grbac, 2017; Zhang et al., 2017; Zuo et al., 2016; Lichtenwalter et al., 2010; Krawczyk et al., 2016; Bach et al., 2017; Cao et al., 2017a).

The fact that this problem is present across various different domains has served as a stimulus for research to expand further into understanding what affects the accuracies of the minority and the majority classes. In imbalanced domains, accuracy is highly dependent on a trade-off between false positives and false negatives. The solution space for these particular issues has ranged from the development of new sampling approaches to the development of new learning algorithms that are specifically designed to deal with the imbalanced problem. In the context of sampling, techniques that have since been developed can be described as belonging to one of three general categories: undersampling, oversampling, or a hybrid combination of both.

Undersampling techniques are described as able to produce a more compact training dataset that reduces the associated costs to the learning and processing time that classifiers face during the learning stage. However, undersampling also has its downsides. Firstly, these techniques tend to increase the variance of the classier and produce a distorted posterior probability (Dal Pozzolo, Caelen, & Bontempi, 2015). Secondly, undersampling techniques also potentiate the likelihood of discarding otherwise useful data for the classifier's learning. Furthermore, as the degree of imbalance becomes more severe, the amount of data that must be discarded to (appropriately) undersample the majority class also increases. This increase in the amount of scrapped data leads to a problem whereby the lack of data will limit the ability of a classifier model to make generalizations (Wasikowski & Chen, 2010).

As a solution to this problem, researchers developed oversampling methods that do not require a reduction of the majority class. Instead, these techniques deal with the class imbalance problem by replicating the minority class examples. Hence the name 'oversampling'; sampling techniques that fall under this category are known to make use of additional examples of the minority class by replicating instances from that class. Nonetheless, the application of random oversampling necessitates adjusting the weights of importance (often expressed as the associated costs) for the minority class. However, these weights can only be determined and calculated correctly if the learning algorithms are capable of distinguishing between class types, noise, and clusters competently.

The identification of clusters in the minority class is particularly hard when there are cases of overlapping data between the minority and majority class (García, Mollineda, & Sánchez, 2008; Cieslak & Chawla, 2008). Another aspect that further exacerbates the difficulty of identifying clusters of the minority class is when there is a presence of small disjuncts within the space of the minority class itself (Jo & Japkowicz, 2004). Moreover, the lower and more specific the decision region of the minority class, the higher the possibility of overfitting (when classifiers learn overly particular idiosyncrasies underlying in the data) since multiple copies of the same instance of data (from the minority class) can potentially be acquired by the classifier more than once. This is because the selection is made at random with replacement, such that the instances of data that are chosen to oversampled are chosen with replacement (meaning that once data is selected, it is re-inserted back into the training set so that probabilities of selection are always the same and independent of each other)

1.2 Research Project

Learning and mining from imbalanced datasets gained increased interest in recent years. One simple but efficient way to increase the performance of machine learning in imbalanced domains is to use synthetic data to increase the number of samples for the minority class. This technique is known as SMOTE.

This research project evaluates the effect of using SMOTE oversampling methodologies that rely on generating synthetic data to oversample the minority class. Different strategies were created using seven different learning classifiers in combination with seven different oversampling methods (most of which are variants of SMOTE algorithm).

The performance of these strategies will be assessed and ranked based on three different metrics. These metrics have been selected to measure the performance of strategies given that they are widely accepted as some of the more appropriate metrics for judging the performance of machine learning techniques in imbalanced domains.

1.3 Problem Definition: Credit Card Fraud Classification

Credit card fraud classification falls into one of two categories supervised (Bhattacharyya et al., 2011; Brause et al., 1999; Chan et al., 1999) or unsupervised (Bolton & Hand, 2001; Tasoulis et al., 2006). As a standard approach, supervised learning of credit card fraud typically involves the classification and not regression because the target variable is typically categorical or binary (i.e., fraud or non-fraud). Unlike regression, which concerns itself with predicting a continuous variable, classification involves the prediction of a category of a target variable.

The learning process is referred to as 'supervised' because the process occurs under the supervision of a pre-established target or output variable (which are assumed to have the class groups labeled accordingly). The target (i.e., dependent) variable is influenced by a set of independent variables (referred to as features) that can be used to explain and predict the respective class of data instances through a process known as 'classification.' Credit card fraud classification is then simply the classification of instances of credit card fraud in credit card data. And yet this particular classification task is notoriously challenging due to three main problems. First, fraud represents only a tiny fraction of all daily transactions (the percentage of fraudulent data is typically less than one percent). Secondly, the nature of fraud and its distributions overtime are never static and are instead always changing due to new strategies and seasonality. Thirdly, it takes time to certify that a transaction is fraudulent after a transaction takes place.

The first challenge is related to the class imbalance problem and is especially prominent in fraud data because the distribution of the transactions is particularly skewed towards the negative (non-fraudulent) class. Moreover, the distribution of fraud data also suffers from a problem of overlapping between class groups. Most learning algorithms are not suited to deal with both unbalanced and overlapping class distributions (Batista, Carvalho, and Monard, 2000). There is evidence that suggests that the class imbalance problem on its own can be solved. However, class imbalance, coupled with other problems caused by overlapping, small disjuncts, noise, and more, poses a much harder challenge for research altogether.

1.4 Research Objectives and Methodologies

The objective of this research is to present a detailed empirical comparison of six variants of the SMOTE technique for oversampling highly imbalanced data of credit-card fraud. This research seeks to evaluate how these techniques perform and compare in the context of credit-card fraud classification and its respective class-imbalance problems. Ultimately, the main objective of the experiments provided in this research is to expand our understanding of these techniques and evaluate how they are or are not useful for resolving some of the problems presented by the highly imbalanced distribution of fraud to non-fraud credit-card data.

This research will evaluate and compare the performance of sampling techniques in machine learning models based on three metrics on three different versions of the dataset (referred to as 'Sets') that have varying degrees of imbalance between class distributions (of fraud to non-fraud).

The First Set ('Set 1) uses all the data from the original dataset, and the distribution of fraud to non-fraud data is unmodified with 492 positive (i.e., fraudulent) cases and 284315 negatives (i.e., non-fraudulent) cases. The second Set, (' Set 2') uses all the positive class cases (492) and two-thirds (189543) of the negative class cases. The third set ('Set 3') uses all positive class cases (492) and one-third (94772) of the negative class cases.

For each Set, three metrics were calculated for each of the fourth-nine different strategies evaluated (the strategies combine seven different classifiers with seven different sampling techniques). Therefore, for every set 147 (7x7x3), different metrics were calculated. Moreover, for the three Sets created and tested, 441 (147x3), different metric values were calculated for evaluating the strategies. Using all 441 metric values, an aggregated rank and mean scores were produced. The objective of this project is to use these metrics to evaluate the relationship between the oversampling techniques and the performance obtained by the models.

The methodology used in this project is a mix of qualitative and quantitative methods. The metrics for performance is followed by an analytical evaluation of the results as well as the experimental design. Secondary research has been conducted by extrapolating data from an existing dataset on credit-card fraud and evaluating the summary of the findings. Moreover, secondary research is provided by producing a review of relevant literature and state-of-the-art techniques in the context of machine learning, credit-card fraud classification, and the 'Class-Imbalance Problem'.

The research in this project follows a Constructive form. This project uses seven different classifiers in combination with seven different sampling techniques to create models (referred to as strategies) that are built and tested through a pipeline that, in the end, compares the performance. The sampling techniques and learning classifiers that were used in this experiment are all relevant to the task of credit-card fraud classification and suited to imbalanced domains.

The quantitative data obtained from the performance of the various strategies are analytically assessed to determine whether alternative strategies outperform baseline strategies. Deductive reasoning is used to help decide and validate whether hypotheses should be accepted or rejected. The concluding remarks are based on the achieved performance of strategies in the experiments. Lastly, an analysis of whether the strategies are appropriate for credit-card fraud classification will be based on a general evaluation of their performance across all the experiments. Strategies will also be compared to strategies in the literature reviewed.

1.5 Scope and Limitations

The scope of this research is limited by the fact that only one dataset on credit-card fraud is currently publicly available. In order to enhance the research and the results, it would be essential to evaluate additional datasets on credit-card fraud. The size of the dataset itself is significantly large in comparison with 284,807 total credit-card transactions. The research is also limited by the lack of parameter optimization. The parameters that were set for the sampling techniques and learning classifiers were all default parameters given by that technique's respective library. Only two parameters were modified.

The dataset is significantly larger in comparison to the imbalanced datasets from the UCI Machine Learning Repository. A large part of existing research on class imbalance and sampling techniques have been done using these datasets. Therefore, because this project uses a different dataset, this is a limitation to its scope. However, this also presents an opportunity to produce an analysis that is original and relevant to the context of credit-card fraud due to the large and complex nature of the dataset.

This enabled the research to expand its scope (that is limited by the use of single dataset) and extend its investigation by also evaluating how each strategy and hence the sampling techniques and classifiers respond to changes in the distribution to the minority (fraud) class and the majority (non-fraud) class.

1.6 Document Outline

Chapter 2 - Review of Existing Literature

This chapter provides a look into the relevant literature and state-of-the-art technologies that are being used in the literature relevant to SMOTE and resolving the

'Class-Imbalance Problem.' This chapter will also describe some of the main advances that have been made in the context of credit card fraud and fraud classification.
This chapter will firstly provide a general historical view of the solutions to the imbalance problem with the goal of clarifying the subject and defining the key domain-specific concepts that are necessary for understanding the subject. Secondly, this chapter will also provide a more in-depth review of the literature and research on various different types of techniques that are currently available. Lastly, this chapter will outline some of the unresolved areas in literature and any gaps that have been identified in the research done for this project.

Chapter 3 - Experiment, Design and Methodology

This chapter will explain the design choices for this experiment. For this project, seven learning algorithms will be used to learn from the same dataset using seven different sampling methodologies. This project follows a design and experimental methodology that follows the methodology outlined by CRISP-DM (Cross Industry Standard Process for Data Mining) (Shearar, 2000) for data mining related tasks which will be outlined in this chapter. This chapter also gives a brief explanation and introduction to the sampling techniques and the learning classifiers that were used to create the various different strategies that were used to experiment on the dataset. Moreover, this chapter explains the procedures that were followed during the experimentation and how the dataset was prepared into three different sets ('Sets') that were used for evaluating the strategies and how the strategies were evaluated during the learning and training processes.

Chapter 4 – Experimental Results, Evaluation and Discussion

This chapter will present the results obtained from the experiments. A brief analysis will be given after the results. The results are divided into two parts. Results Part 1 are results that were used to evaluate the baseline hypothesis. Part 2 of results are results that were designed to evaluate the secondary hypothesis and expand on the evaluation and discussion.

This chapter will also present a summary of results followed by an evaluation and discussion of the findings. This chapter will then also evaluate the experiment by providing a brief discussion of experimental strengths and weaknesses and limitations.

Chapter 5 - Conclusion

This chapter will present a summary of the main findings. The chapter will then proceed and will include some discussion of findings. A brief discussion of the strengths and weaknesses of the experiment will be outlined. The chapter will then proceed to describe some of the viable avenues for future research and recommendations for future work. Lastly, the contributions and impact of this project will be outlined and briefly discussed.

2. REVIEW OF EXISTING LITERATURE

2.1 Early Research

In the research paper, by Weiss and Provost (2003), the authors conclude that naturally occurring distributions are not always optimal. For this reason, it became evident that it is necessary to modify the distribution of the training set based on an evaluation function. Therefore, re-sampling methodologies that function by either adding to the minority class or removing the majority class for a given dataset became the de facto standard for countering the class imbalance problem across several different domains. Over and under-sampling methodologies received significant attention in this context (Solberg and Solberg, 1996; Japkowicz, 2000a; Chawla et al., 2002; Weiss and Provost, 2003; Kubat and Matwin, 1997; Jo and Japkowicz, 2004; Batista et al., 2004; Phua and Alahakoon, 2004; Laurikkala, 2001; Ling and Li, 1998). A significant part of these studies explicitly addresses how the different variants of these techniques can counter the problem of imbalance and skewed class distributions. Sometimes conflicting, different viewpoints have since been presented on the appropriateness of oversampling versus undersampling (Chawla, 2003; Maloof, 2003; Drurnmond and Holte, 2003; Batista et al., 2004). However, one standard, long-lasting critique of this research remains: *how does one effectively identify the potentially optimal sampling techniques and parameters for a given data set?*

Moreover, an accompanying question is: *how can the techniques for imbalance generalize across cost-sensitive scenarios?* The challenge in establishing an appropriate trade-off between false positives and true positives can be of paramount importance. For example, particularly for classification tasks on cancer or fraud, the severity of the costs associated with type one and type two errors is particularly uneven. That is, the costs of misclassifying the negative class as positive (as cancer or fraud) are hugely disproportional to the inverse relation.

Previous research (Ling & Li, 1998; Japkowicz, 2000) has discussed over-sampling with replacement and has noted that it does not significantly improve minority class recognition. This investigation helped Chawla realise that at the root of the problem was the fact that the minority class was being overfitted from oversampling. Chawla's solution was to synthetically generate new instances of the minority class so that new (and not repeated) information is fed to the learning algorithm.

In Chawla, Bowyer, Hall, and Kegelmeyer (2002), the authors proposed an alternative that could potentially avoid the problem of overfitting that simple random oversampling technique. Instead of "weighting" existing data points, the main rationale behind the technique was to create new data points for the minority instances. The technique was called Synthetic Minority Oversampling Technique, and the term SMOTE (Chawla et al., 2002) became popularised. The basis of the SMOTE technique was to interpolate between neighboring instances of the minority to create new instances of that class. The number of instances, therefore, increases as new minority class examples are added to the 'neighbourhood' of the class. In essence, this allows the classifier model to train on more unique data, thereby decreasing the probability of overfitting and increasing its ability for generalization.

The technique quickly became popular and a frontrunner for the preprocessing techniques used in class imbalance research. Since its introduction in 2002, many extensions to the original technique have been developed with the objective of improving its suitability and performance depending on a specific context. The abundance of extensions and alternatives to the original SMOTE technique is a testament to how successful and impactful the original technique has been. The technique is arguably one of the most influential algorithms for preprocessing and sampling in machine learning and data mining (García, Luengo & Herrera, 2016).

2.2 Synthetic Minority Oversampling Technique (SMOTE)

The SMOTE algorithm relies on using an oversampling approach to rebalance the original training dataset. Instead of using simple replication on the minority class, the main idea of the technique is to introduce new data that is made up of synthetic examples of the minority class. These synthetic instances are created by interpolation between instances of the minority class that are located within a defined neighborhood space. This space is referred to as a "feature space" rather than "data space" (Chawla et al., 2002). The under-represented class is over-sampled by introducing synthetic examples along the line segment that joins any or all of the k minority class nearest neighbors. Depending on the amount of over-sampling that is required for the dataset, the number of k neighbors is randomly chosen (Chawla, 2005).

The procedure works as follows. Firstly, the total amount (an integer value) of oversampling *N* is established. This amount can be either set-up to obtain an approximate one to one class distribution or is discovered via a wrapper process (Chawla et al., 2008). Next, an iterative process composed of various steps is carried out. The first step is to select at random a minority class example from the training set. Then, its *K* nearest neighbors (5 by default) are obtained. Lastly, the *N* number of the *K* instances is randomly chosen for creating new instances by interpolation. In order to calculate the value of *N*, the difference between the feature vector (sample) and each selected nearest neighbor is computed. The difference is then multiplied by a number chosen at random between 0 and 1. Then the product of this multiplication is added to the previous feature vector. This causes the selection of a random point along the "line segment" that connects the minority class examples between the features. If the features are nominal, one of two values are randomly selected. The formal algorithm is outlined below. Given two minority class examples, such $x^i, x^j \in R^d$ new samples are generated by:

$$x^{new} = x^i + r \cdot |x^j - x^i|, \text{ where } r \in [0,1] \tag{1}$$

Such that 'r' denotes a random number ranging between zero and one. Hence, 'r' is also sometimes referred to as a random probability that dictates the proximity of the (newly) generated synthetic minority data instance to the original minority class instance on the interpolated line segment that joins the two points.

2.3 Extensions to the original SMOTE algorithm

The SMOTE technique has served as the underlying foundation for all other oversampling methods that use synthetic or artificial data. Therefore, in the context of class imbalance classification, any preprocessing methodology that uses synthetic examples by interpolation or some other process is, to some extent, derived from the original SMOTE algorithm by Chawla (2002).

Many extensions and variants to the original algorithm have since been developed. In the paper "SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary" by Garcia, Herrera, Chawla and Nitesh (2018), the authors compose a comprehensive list of categories to which extensions of the technique can be categorised. The most important categories are the following:

1. Selection Procedure: Extensions to the SMOTE technique that focus on the initial selection focus on determining the best candidates for oversampling before the synthesizing of new instances begins. Extensions that fall under this category will combine SMOTE with strategies that are meant to either reduce the amount of overlap or the amount of noise that is generated by the technique in the new dataset that contains synthetic examples. Many SMOTE extensions that focus on initial selection rely on either choosing not to generate synthetic data depending on the amount of minority class examples in the neighbourhood (Bunkhumpornpat, Sinapiromsaran, & Lursinsap, 2009) or how closely minority class examples are to the boundary class (Han et al., 2005). Nakamura (2013) presents another variant which relies on using a Learning Vector Quantisation (LVQ) (an algorithm that is similar to k-Nearest Neighbours but is optimised) to optimise the selection process. Other SMOTE variants will change the initial selection procedure based on borderline class data (Nguyen et al., 2009; Cervantes et al., 2017), for instance, by using a support vector machine (SVM) algorithm to help decide where borderline class examples are, to then generate synthetic data.

2. Type of Interpolation: Extensions of the algorithm that fall under this category include additional mechanisms to modify the way synthetic examples are to be interpolated. These interpolation mechanisms will ultimately define the way that new instances are synthesized. The interpolation mechanism can be *range-restricted* (Han et al., 2005; Bunkhumpornpat et al., 2009; Maciejewski & Stefanowski, 2011), whereby both the nearest neighbors for the minority class and the majority class are accounted. Alternatively,

weighting can be used to create synthetic instances that are closer to a specified instance than the nearest neighbor (Hukerikar, Tumma, Nikam, & Attar, 2011).

3. Integration with under-sampling: Extensions that fall under this category will rely on using undersampling to remove examples from the majority class. The under-sampling is done either randomly or by using an informed technique. The process of undersampling can occur either before or during (as an internal process) the SMOTE technique's application.

4. Dimensionality Modifications: Extensions to the SMOTE algorithm that fall under this category incorporate some mechanisms for either reducing or augmenting the dimensionality of the data. The adjustments to the dimensionality of the data takes place either before or during the generation of synthetic/artificial examples. Most commonly, the process of changing the dimensionality of the data occurs before the SMOTE technique is applied. In the context of reduction, the dimensionality can be reduced by using, for example, Principal Components Analysis (PCA) (Abdi & Hashemi, 2016). Bagging (Wang, Yun, li Huang, & ao Liu, 2013a) and various other nonlinear dimensionality reduction techniques (Bellinger, Drummond, & Japkowicz, 2016) can also be used.

5. Adaptive Generation: Extensions of the SMOTE that use adaptive generation for synthesizing new data, rely on using a weighted distribution depending on the degree of difficulty in learning each minority class example. In the paper (He, Bai, Garcia, & Li, 2008), the first adaptive SMOTE-based technique 'ADASYN' was introduced to help produce more synthetic data for minority class instances that proved harder to learn. This inspired other similar techniques to also incorporate mechanisms that help control the amount of synthetic data that is generated (Alejo, García, & Pacheco-Sánchez, 2015; Rivera, 2017).

6. Filtering of noise: These variants of the SMOTE algorithm were developed help reduce the number of synthetic data that exhibit overlap or noise within

the dataset. Two of the most renown techniques include SMOTE-Tomek and SMOTE+ENN (Batista et al., 2004).

2.4 The Class Imbalance Problem for Credit card Fraud

In the context of credit fraud classification, the nature of the data is particularly unbalanced, and most learning algorithms are not traditionally designed to cope with the degree of skewness that the class distributions exhibit. The degree of skewness is such that standard classification algorithms that seek to maximise over-all accuracy will often sacrifice learning instances of the minority class (lowering its recall score) to increase its local accuracy of the majority class and increase its specificity score. Nonetheless, the imbalance in the ratio between the minority and the majority class is not the only problem. Other prominent challenges for researchers include the amount of overlapping that often occurs between the classes (Holte, et al., 1989; Batista et al., 2005, García et al., 2007; García et al., 2008) and the presence of noise or noisy instances within the datasets (Anyfantis et al. 2007; Hulse & Khoshgoftaar, 2009). Furthermore, another existing challenge for this task is the existence of small sub-clusters (known as *small disjuncts)* within the minority class formation (Japkowicz & Stephen, 2002; Stefanowski, 2013). These are all factors that ultimately diminish the performance of classifiers. Different methodologies for handling some of these issues above have been developed.

2.5 Data-Level Solutions

At the data level, one way to resolve the issue is by using sampling to alter the size of training sets and rebalance the ratio and distribution of classes. The rebalancing of class distributions is often viewed as fundamental since research agrees that, in general, most standard classification models will almost always produce better results when trained on balanced datasets (Weiss & Provost, 2001; Laurikkala, 2001; Estabrooks et al., 2004). Contrary to this view, some classifiers have shown that there is no change in performance when learning from training sets that have been rebalanced through sampling (Japkowicz & Stephen, 2002; Batista et al., 2004). This means that some classifiers are just as capable of learning from imbalanced data and that rebalancing may not always be necessary. Fundamentally, however, the effect of resampling on performance can either be positive or null (as it has never been shown to be detrimental to performance). For now, the only way to know this is by running tests and evaluating a posteriori.

The next question then is: *what type of resampling is the most effective for fraud data?* Literature suggests that a superior methodology does not exist because the most optimal method will vary depending on the dataset and the classifier algorithm (Japkowicz & Stephen, 2002; Dal Pazzolo et al., 2013). Moreover, because fraud is an inherently evolving activity, this means that methodologies that worked well in the past can become obsolete. For all these reasons, the optimum sampling strategy for fraud classification is particularly hard to define as it depends on the distribution and nature of the data. In the research by Dal Pazzolo, the author proposes using a racing strategy so that various methodologies are tested in order to find the most optimal or superior method. Dal Pazzolo experimented on the following techniques: undersampling, oversampling, SMOTE, CNN, ENN, NCL, OSS, and Tomek Link. These techniques were tested using a ten-fold CV with the following different classification algorithms: SVM, Neural Networks (NNET), RF, LB, and Decision Trees. For every combination of a classification algorithm and a sampling technique, the average G-mean of the CV was taken and used to calculate the average accuracy over all the datasets. For the credit card dataset, the highest F-measure is achieved using RF with SMOTEnsemble.

In the research by (Chawla 2008), the author shows that there is no clear advantage in oversampling over undersampling or vice-versa. Instead, Chawla (2008) is a proponent of the view that the most effective rebalancing strategy is dependant on the nature and distribution of the dataset. In his research, Chawla's best performing model (that used SMOTE and undersampling as a rebalancing strategy) outperformed all other cost-based classifiers (that control the weight attributed to each class based on their estimated misclassification cost) of cost-sensitive classes for various real-life cases (Chawla, 2008).

2.6 Algorithm-Level Solutions

Algorithms can be modified or extended to serve unbalanced tasks better. At the algorithmic level, the algorithms that are used for resolving class imbalance will typically fall into one of two categories of methods: cost-sensitive learning or class-imbalance learning. For the latter category, the objective is to minimize costs by optimizing the trade-off between majority and minority classes based on misclassification costs. That is, by improving the accuracy of the class with the highest misclassification costs. For the former category, the objective is to improve the local accuracy of the minority class by modifying the original classifier algorithm to enhance its capacity to learn from the minority class.

For DT algorithms, the use of information gain (IG) as the splitting rule criterion has shown to return rules that are biassed towards the majority class (Quinlan, 1993). Therefore, traditional DT algorithms can be modified to better suit class-imbalance. For example, instead of us IG, the Hellinger Distance (HD) is more appropriate as the splitting rule criteria since HD is not sensitive to the skewness of classes. Standard C.45 DTs exhibit improved performance on imbalanced datasets when using HD Chawla & Cieslak, 2008) or the Class Confidence Proportion (CCP) as the splitting rule criteria (Liu et al., 2010). For the SVM algorithm, an F-measure optimization has been suggested as a way to improve the performance of the classifier on imbalanced datasets (Callut & Dupont, 2005). The SPARCCC algorithm (Verhein & Chawla, 2007) is another example of a learning algorithm designed explicitly for imbalanced learning.

These are all examples of tuning at the algorithmic level to help deal with the class imbalance problem. Some researchers propose that this type of solution should be considered more suitable for class-imbalance since it deals with the problem directly without biasing the classifier towards one class (Weiss, 2013). Many of the class-imbalanced learning methods use strategies that are common to ensemble learning methods. Bagging (Breiman, 1996) and Boosting (Freund, Schapire et al., 1996) are two standard techniques to aggregate classifiers for ensemble learning. The main idea behind classifier aggregation is that classifiers can be combined iteratively, permitting the ensemble model to combine different learning algorithms that are better suited for the minority and majority class distributions. EasyEnsemble and UnderBagging are examples of this.

In the research by Dal Pazzolo, the author showed that for the credit-card fraud dataset, a RandomForest (as the base learner) in combination with SMOTEnsemble was found to be the statistically superior strategy (Dal Pazzolo et al., 2013). SMOTEnsemble is a combination of the EasyEnsemble algorithm (Liu, Wu, and Zhou, 2008) and SMOTE that has shown to be particularly robust in imbalanced domains and for the classification of fraud.

2.7 SMOTEnsemble

SMOTEnsemble is an example of an ensemble learning method that combines SMOTE with the EasyEnsemble algorithm (Liu, Wu, and Zhou, 2009). EasyEnsemble samples several subsets of the majority class and trains a learner using each of them; a single output is then created by combining the outputs from these learners. The EasyEnsemble algorithm is an ensemble method that is designed to use the majority class examples, which are ignored by under-sampling.

Both EasyEnsemble and Balanced Random Forest use balanced bootstrap samples. In the research paper by the original authors of the algorithm, the authors remark that the algorithm is specifically designed to deal with the class-imbalance problem directly and when the class imbalance is "harmful" (i.e., severe enough to degrade the performance of classifiers) the algorithm outperformed all other methods (Liu, Wu, and Zhou, 2009).

A significant limitation of the EasyEnsemble method is the lack of comprehensibility that comes from combining the output from multiple classifiers, as ensemble methods are somewhat black-box methods. Moreover, the authors remark that class-imbalance learning methods like the EasyEnsemble can sometimes lead to a decrease in performance when the class-imbalance is not "harmful". However, there is no way to determine when the degree of imbalance is "harmful". Therefore, we cannot know without testing whether the EasyEnsemble method will be helpful or not. The authors describe that a potential indicator for a "harmful" imbalance is that when using AdaBoost and Bagging on DT will either decrease or have no effect on the performance. When the imbalance is not "harmful" AdaBoost and Bagging on DT will often significantly improve the performance of decision trees.

2.8 Cost-Sensitive Learning

Cost-sensitive learning can be used to avoid some of the potential problems that arise from using different sampling techniques, but it is limited by the fact that it requires the specific cost information to be known a priori. This information may sometimes be impossible to obtain. One potential solution to this is to take a cost-sensitive algorithm and to test it with different cost ratios in order to improve performance by finding the optimal trade-off between rare and regular classes. In comparison to using different sampling techniques, an advantage to this solution is that it allows all the data to be used. This means that there is no information loss and that the learning speed of algorithms is not reduced since no instances of duplicate data are re-inserted back into the training set (Drummond & Holte, 2000). When cost the information is known, the recommendation is to use cost-sensitive learning instead of sampling as these methods have been shown to outperform over-sampling and under-sampling (Japkowicz, Myeers, and Gluck, 1995). Conversely, it may be argued that sampling techniques have a higher universal value since their application is manifold and irrespective of costs. Sampling techniques do not require a context that includes considerations concerning profit-maximizing or loss-minimizing. Sampling methodologies can be used to reduce the class imbalance problems irrespective of the context or domain and do not require any extrinsic information to be known a priori. This greater flexibility is undoubtedly an advantage of sampling methodologies over cost-sensitive learning and should be an incentive to pursue advances in the field further.

Cost-Sensitive learning relies on tuning learning algorithms internally to help counteract the effects of class imbalance by adjusting the weights to classes. There have been many variants of the Adaboost algorithm (Freund & Schapire, 1996) that have been proposed as cost-sensitive learning algorithms (Ting, 2000). A common strategy of these learning algorithms is to increase the weights of class instances with higher misclassification costs during the Boosting process. The SMOTEBoost algorithm (Chawla et al., 2003) is an algorithm that uses boosting and SMOTE that is specifically designed for class-imbalance learning. It is similar to the AsymBoost algorithm, which is an algorithm that uses asymmetric boosting by minimizing the cost-sensitive loss function in the statistical interpretation of boosting.

What distinguishes AsymBoost from SMOTEBoost is the way that the algorithm adjusts the distribution of classes. While the former updates different weights of instances of the majority and minority class during each boosting iteration, the latter will first equally update the weights of instances of the majority and minority class and then use SMOTE to produce new synthetic instances of the minority class.

2.9 Open Issues and Existing Gaps in Class-Imbalance Research

2.9.1 Small Disjuncts, Noise and Insufficient Data

The problem of small disjuncts occurs when a dataset contains small clusters or groups of instances (independent of their class) that happen to be represented within small clusters in the feature space of data instances (Orriols-Puig et al., 2009; Weiss & Provost, 2003). When the degree of class-imbalance is high, this problem becomes more likely as the instances of the underrepresented class are usually located in small sub-sections of the datasets. Alternatively, when small disjuncts containing both the minority class and the majority class instances are found in the dataset, this increases the complexity of the problem and decreases the performance of the learning classifier. This problem is referred to as the "Subclus" problem and is, in part, created from the traditional procedural search for maximum generalization by standard learning classifiers (Napierala, Stefanowski, and Wilk, 2010). Moreover, sampling techniques like SMOTE that generate artificial samples can sometimes produce noisy examples, which hinders the ability of a classifier to identify the boundaries of a problem.

2.9.2 Noise and Small Disjuncts

Instances that overlap within disjuncts of the other classes are often interpreted as class-noise (i.e., noise specific to that particular class) by most learning classifiers because they are viewed as instances that are present within 'neighborhood' areas of the opposite class (Kubat & Matwin, 1997; Jo & Japkowicz, 2004). Moreover, in the research by Seiffert, Khoshgoftaar, Hulse & Folleco (2011) the authors show that standard classification algorithms are particularly sensitive to noise in imbalanced domains and as the degree of imbalance increases its effect on decreasing performance also increases. As a solution, the E-NN algorithm has been suggested as it has shown to be more robust in the presence of noise (Seiffert, Khoshgoftaar, Hulse, and Folleco, 2011). Moreover, because standard classifier algorithms are particularly sensitive to noise, this means that there is a higher chance of overfitting on training data when noise is present. The impact that this has on the classification accuracy of the positive (minority) class is much stronger than its effect on the accuracy of the negative (majority) class.

Therefore, in order to help prevent overfitting, additional techniques have been implemented to help alleviate the problem of noise and overfitting. For example, pruning of traditional C.45 Decision Trees has been shown to be somewhat effective at counteracting the effect of noise on classifier accuracy because of its ability to handle the existence of 'small disjuncts' in the dataset (Weiss, 2010). However, pruning comes at a cost since by increasing the generalization of the model, the likelihood of missing meaningful minority class examples increases. Given the already low number of minority class examples, a misclassification would mean a significant decline in performance. Therefore, other solutions to the problem have instead been advised.

2.9.3 Small Disjuncts and Insufficient Data

The problem of disjuncts is particularly detrimental for learning algorithms whose learning methodologies are based on a divide-and-conquer strategy (for example, with decision trees) (Weiss, 2004; Rokach, 2016). This occurs because as the problem is broken down into different sub-groups or components, each component is solved iteratively until all component-solutions are combined into one. This increase in the number of iterations can lead to data fragmentation (Friedman et al., 1996). Data fragmentation, in turn, leads to the degradation of classifier performance since it increases both the memory requirements and the computational processing time for the task. When there is insufficient data, these problems are further exacerbated. This is because as the number of data points decreases, the possibility of small disjuncts increases as learning classifiers will struggle to generate good generalizations when there is insufficient data to represent the boundaries of the problem (Jo & Japkowicz, 2004; Wasikowski & Chen, 2010).

The SMOTE technique implicitly contains a mechanism to help counteract the small disjuncts problems since the default technique relies on using a K-NN algorithm to define boundaries for separating classes and inter-class examples (Fernández, Garcia, Herrera, and Chawla, 2015). However, the authors also explain that its ability to successfully counteract small disjuncts problems is highly dependant on the number of elements that are contained within the small disjuncts and the selected value of K value for the K-NN algorithm that is used during oversampling. Moreover, the authors also conclude that if the small disjunct contains examples from both the minority and the majority class (i.e., if there is overlapping), then the SMOTE technique is incapable of rectifying the problems that are caused by the within-class imbalance.

GAP IN LITERATURE: There is no research on the specific threshold level or amount for the number of overlapping elements (that are contained within the small disjunct) for which K-NN begins to be inefficient for resolving the problem.

As a solution, the authors suggest SMOTE extensions that use a cluster-based interpolation that focuses on local densities because of two reasons. Firstly, cluster-based extensions allow the SMOTE algorithm to focus more on areas that lack representation and thus require additional data whereby synthetic instances are to be generated. Secondly, cluster-based extensions of SMOTE are better at reducing problems of overgeneralization for the minority class by synthesizing new instances more sparsely further away from the centroid of the minority class cluster (Fernández, Chawla, et al., 2018). An alternative solution to the problem is to account for the pairwise differences among data points (Pekalska & Duin, 2005). This is because, in the traditional feature space, different instances may have the same representation, whereas, in the dissimilarity space, only identical instances (that have the same class label) can have a dissimilarity distance equal to zero. Therefore, the possibility of class overlap is not possible in the dissimilarity space.

GAP IN LITERATURE: Furthermore, overall research shows that the analysis of clusters is an independent area of research in the class imbalance community that is still very much undeveloped. The analysis of clusters is a topic that still requires much more empirical experimentation and support since it currently relies on making assumptions that overly simplify systems that have a poor grounding in respect to its real-world application. This shortcoming is especially problematic when the task involves complex real-world data with varying intrinsic characteristics (overlapping, small disjuncts, noise, and more).

Furthermore, critical research by Kubat and Matwin (1997) has helped develop the distinction between noise, borderline, and safe examples. The latter refers to examples that are situated in the relatively homogenous areas of their own class label. Whereas the former is referred to as noise because it describes examples of a condition that occurs within safe areas of the opposite class. Meanwhile, *borderline examples* are examples that are situated in an area surrounding class boundaries, where examples of the minority and majority classes most overlap. The authors provide empirical evidence from a number of different experiments that show that borderline examples are particularly detrimental to the performance of a classifier.

The Borderline-SMOTE algorithm was developed in order to help resolve some of the problems caused by borderline data. This algorithm was first introduced by Han, Wen-Yuan, Bing-Huan (2005) as an extension of the SMOTE algorithm that focuses on generating synthetic data for minority class instances that are exclusively near the borderline. While Han, Wen-Yuan, Bing-Huan (2005) demonstrate that the borderline variant of SMOTE can increase the classification accuracy of the minority class, it is not clear how the overall performance of the models compare.

2.9.4 Dataset Shift Problem

The dataset shift problem refers to the problem that arises when the training and test data have different distributions. This problem is universal and is also present even in classification tasks that do not face class-imbalances since dataset shifts can happen with sample selection bias. Most real-world complex problems will inherently contain some minor degree of dataset shift. Most general classifiers are capable of handling mild dataset shifts without incurring a loss of performance. In highly imbalanced distributions, the minority class is particularly sensitive to classification errors, due to the limited number of minority class examples in the data. In cases of extreme imbalance, a single misclassification will produce a significant decline in the performance of the classifier.

GAP IN LITERATURE: In the context of class imbalance, most state-of-the-art research relies on using stratified cross-validation techniques since these techniques are useful for maintaining the distribution of classes in the test and train splits. This reliance is a natural source of dataset shift that is still unresolved. A more appropriate validation technique that avoids the dataset shift problem is yet to be developed.

2.9.5 Studies on the Effectiveness of Methods and Performance Metrics for Evaluating Binary Classification Tasks

When it comes to performance, the main questions on assessments that fall under the imbalanced class domain are:

1. What are the data characteristics that degrade the performance of classifiers in imbalanced tasks?
2. Is it possible to provide for approaches that, in general, are capable of providing the best improvements in performance?
3. Is the performance of the learning algorithms affected by different degrees of imbalance?
4. How do varying degrees of imbalance to the distribution of classes affect the performance of classifiers?

One of the first studies to ever address some of these questions can be found in Japkowicz and Stephen (2002), wherein the authors compare five different sampling strategies. The strategies involved using random undersampling and oversampling, focused random undersampling and focues oversampling (where the focus of the sampling was on parts of the input space that were either far or close to the decision boundary) and lastly modifying the misclassification costs associated with the classes. Although the study is an essential contribution to the class-imbalance research, a significant limitation is that the comparisons on performance have been assessed using the rate of error of the classifiers.

This measure has been shown to be unsuitable for class-imbalance domains. The main conclusion from the research by Japkowicz and Stephen (2002) is that when using DT, the impact of 'harm' caused by the imbalance increases as the degree of data separability decreases. Secondly, the increase in the training set size reduces the impact of 'harm' caused by the imbalance. Thirdly, the degree of imbalance is only a problem when disjuncts are present in the data. Fourthly, undersampling has been found to generally underperform in comparison to oversampling. Lastly, Japkowicz and Stephen (2002) also conclude that the modification of costs that are associated with the misclassification of different classes is a strategy that tends to outperform random or focused oversampling.

A different experimental approach was used in the research by (Batista et al. 2012; Prati et al. 2014) whereby the researchers used real datasets and for each dataset several training set distributions were generated using the same number of examples and varying degrees of imbalance. The effect of a change to the degrees of imbalance to the class distributions on a dataset was assessed by measuring the loss in performance (using the metric AUC) of an imbalanced distribution in comparison to perfectly balanced class distribution.

$$Loss = L = \frac{B-I}{B} \tag{2}$$

Where *B* represents the performance obtained on a perfectly balanced class distribution, and *I* represent the performance obtained on an imperfect (imbalanced) distribution.

Random oversampling, SMOTE, borderline-SMOTE, and ADASYN were some of the strategies tested. One of the main contributions from this research is that for highly imbalanced distributions (10/90, 5/95, 1/99) there is a general failure to improve performance for all the strategies tested. Moreover, Metacost proved to be the least favorable strategy for improving performance. Lastly, two extensions of the SMOTE algorithm (ADASYN and Borderline-SMOTE) did not prove to be significantly better than the standard algorithm itself.

Another vital contribution to the effectiveness of class-imbalance methods can be found in the research by López et al., (2013). In this study, the authors compare three different types of learning classifiers SVM, DT, and K-NN on sixty-six different datasets using the AUC metric. The study focused on using SMOTE and extensions of the SMOTE algorithm that fall into different categories. The first category involves extensions of the algorithm that include a pre-processing strategy (e.g., SMOTE+ENN, Borderline-SMOTE, safe-level-SMOTE, ADASYN…). The second category involves algorithm-level strategies that are either based on cost-sensitive learning or are ensemble-based strategies (e.g., the EasyEnsemble, RUSBoost, and SmoteBagging).

From the strategies that include a pre-processing method, SMOTE and SMOTE+ENN obtained the best results. Borderline-SMOTE and ADASYN also showed excellent performance on average. From the ensemble strategies, SmoteBagging showed the best results, followed by RUSBoost and EasyEnsemble. However, one notable limitation of these studies is that it assumes that a perfectly balanced distribution (between majority and minority class) is more favorable for performance. However, this has been shown not to be the case (Weiss and Provost 200); Khoshgoftaar et al., 2007). Albeit a lot more minor, another limitation of these studies is that they rely on using only one metric (AUC) to measure loss in performance.

In fact, there seems to be a lack of agreement on what is the best way to measure fraud detection performance. Many of the measures rely on costs to formulate measures of performance that are either transaction-dependent (Elkan, 2001; Bahsen et al., 2013; Bahsen et al., 2015) or class-dependent (Bolton & Hand, 2002; Hand et al., 2008). Alternatively, some literature avoids using cost-based measures by making an implicit assumption that predictive accuracy is more important for measuring performance (Bhattacharya et al., 2011; Dal Pozzolo et al., 2014). Moreover, it is often the case that cost matrices may not be producible either due to lack of information or confidentiality. For this reason, it would be important to formulate a measure of performance that is more objective.

The metric for determining the degree of class imbalance is given by the imbalance ratio, whereby:

The Imbalance Ratio (IR): $IR = \frac{N_+}{N_-}$ (3)

Where N_+ denotes to the number of positive class cases in the dataset and N_- denotes the number of negative class cases in the dataset.

GAP IN LITERATURE: However, an existing gap in the literature is that there is no metric for determining when the IR ratio represents a severity that is deemed 'harmful.' The IR may signal that class distributions are imbalanced, but this imbalance may not be what is referred to as 'harmful' (Liu, Wu, and Zhou, 2009).

So, there is no metric for gauging whether an imbalance is 'harmful' a priori. However, there are ways to determine whether an imbalance in ‘harmful’ after the fact. For example, it can be checked using class-imbalance learning methods (such as the EasyEnsemble). If the class-imbalance learning method has no effect (or has a decline) on the performance of the strategy, then the imbalance is not considered to be harmful. Unfortunately, there is no way to know this without testing, which ultimately comes at the cost of computational resources and time. As described in (Liu, Wu, and Zhou, 2009), some classification tasks suffer from a class-imbalance problem, but the severity of the imbalance is not significant enough to warrant the use of class-imbalance methods that are specifically designed to deal with the imbalance problem. For tasks that do not suffer from the class-imbalance problem, boosting and bagging techniques on DT can often significantly improve performance; but for tasks that do in fact suffer from class-imbalance, then AdaBoost and Bagging will have either no effect or deteriorate the performance of DT (Liu, Wu, and Zhou, 2009). This is one existing way the authors test for the presence of a ‘harmful’ imbalance. The empirical results of the research by Liu, Wu, and Zhou (2009) suggest that for tasks in which ordinary learning methods are able to achieve a high AUC score (for example, above 0.95), then the class-imbalance learning methods are not helpful. However, when class-imbalance learning methods improve performance, then BalanceCascade and, in particular, EasyEnsemble are both able to achieve a higher AUC, F-measure, and G-mean than almost all other class-imbalance learning methods.

The appropriateness (or lack of) to performance metrics for assessing imbalanced classification problems is a widely studied area of the Class Imbalance Problem. Nevertheless, there are many issues in this aspect that remain inconclusive. For example, the appropriateness of statistical tests or error estimation procedures is an essential area for the problem that is still largely unresolved due to a lack of research. These are significant issues that still require much more research and present an essential challenge to the Class Imbalance Problem (Japkowicz, 2013).

It is a well-known fact that traditional performance metrics in imbalanced domains can lead to sub-optimal classification models (He and Garcia 2009; Weiss, 2004; Kubat and Matwin 1997). Traditional performance metrics produce misleading results due to the fact that these measures are insensitive to skewness and imbalanced distributions (Ranawana and Palade 2006; Daskalaki et al. 2006). Therefore, the use of appropriate evaluation metrics is a critical aspect of classification tasks in imbalanced domains. An appropriate measure or metric should be used to both assess the performance of classifiers as well as help guide their learning processes during the learning phase.

For binary classification tasks with a negative and positive class, the results obtained by a classifier can be explained by a confusion matrix (see Table 1 below). For both the negative and the positive class, the confusion matrix provides:

1. True Positives (TP): The value for the number of positive class instances that were correctly classified;

2. True Negative (TN): The value for the number of negative class instances that were correctly classified;

3. False Positive (FP): The value for the number of positive class instances that were incorrectly classified;

4. False Negative (FN): The value for the number of negative class instances that were incorrectly classified.

Table 1: Confusion Matrix for a binary class problem

Actual	**Prediction**	**Prediction**	**Total**
N=sample Size	Predicted Positive $(Y = +)$	Predicted Negative $(Y = -)$	
Actual Positive $(Y = +)$	$TP = \sum_{i=1}^{N} I(y_i = +) I(y = +)$	$FN = N_{positive} - TP$	$N_{positive} = \sum_{i=1}^{N} I(y_i = +)$
Actual Negative $(Y = -)$	$FP = N_{negative} - TN$	$TN = \sum_{i=1}^{N} I(y_i = -) I(y = -)$	$N_{negative} = \sum_{i=1}^{N} I(y_i = -)$
Total	$\sum_{i=1}^{N} I(y_i = +)$	$\sum_{i=1}^{N} I(y = -)$	N

TABLE 1: CONFUSION MATRIX FOR A BINARY CLASS PROBLEM

Accuracy (see Equation 4 below in page 30) and its complement to the error rate are the most frequently used metrics for assessing the performance of classifiers in classification domains that do not suffer from the class imbalance problem.

$$Accuracy = \frac{(TP+TN)}{(TP+FN+TN+FP)} \qquad (4)$$

However, accuracy suffers from a preferential bias towards the majority class and is unsuitable for assessing imbalanced problems. For example, if only 1% of the total instances in the dataset belong to the minority class, high accuracy of 99% can be achieved by simply predicting all the majority class instances and none of the minority class instances. Consequently, when the objective is to predict rare class instances, this measure is not very useful.

We can derive other metrics from the confusion matrix that are more suitable for imbalanced problems. For example:

5. Recall or Sensitivity-True Positive Rate (TPR): $TP_{rate} = \frac{TP}{TP+FN}$

6. Specificity-True Negative Rate (TNR): $TN_{rate} = \frac{TN}{TN+FP}$

7. False Positive Rate (FPR): $FP_{rate} = \frac{FP}{TN+FP}$

8. False Negative Rate (FPR): $FN_{rate} = \frac{FN}{TP+FN}$

9. Precision-Positive Predictive Value (PPV): $PP_{value} = \frac{TP}{TP+FP}$

10. Negative Predictive Value (NPV): $NP_{value} = \frac{TN}{TN+FN}$

Instead, for evaluating classifiers in imbalanced domains, other classification metrics have been introduced, such as F measure (Rijsbergen, 1979), the geometric mean (Kubat et al., 1998), and the Receiver Operating Characteristic (ROC) curve (Egan, 1975). F1-Score**:** This metric is defined as the harmonic mean of precision and recall.

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5}$$

Where Precision and Recall are defined as follows:

$$Precision = \frac{TP}{TP+FP} \quad \text{and} \quad Recall = \frac{TP}{TP+FN} \tag{6}$$

The geometric mean (G-mean)**:** This metric was developed specifically for imbalanced domains. It calculates the accuracies of both classes by seeking to maximize their respective accuracies while maintaining a good balance between the two classes. However, equal weight importance is attributed to both classes under this formulation. There is another formulation of the G-mean that attributes higher importance to the positive class. In this alternative formulation, specificity is replaced by precision.

$$G_{mean} = \sqrt{\frac{TP}{(TP+FN)} \cdot \frac{TN}{(TN+FP)}} = \sqrt{sensitivity \cdot specificity} \tag{7}$$

The area under the operating receiver curve (AUROC) or the AUC in short is a metric that has become quite predominant for class Imbalance problems (Fawcett, 2003). For example, Dal Pozzolo, suggests an AUC estimate based on the Mann-Whitney (Wilcoxon) statistics (Dal Pozzolo, 2014).

$$AUC = \frac{1+TP_{rate}-FP_{rate}}{2} \tag{8}$$

The AUC-score characterizes the area under the curve of sensitivities of the classifier that is plotted against the corresponding false-positive rate at various different probability thresholds.

2.9.6 Additional Gaps

Listed in Table 2 below are some additional gaps that have been identified in the literature reviewed in this project of particularly important (i.e., highly cited) research.

Table 2: Additional Gaps Identified

Study	**Author(s) and Date**	**Relevant Findings**	**Gaps in the literature**
Learning When Training Data are Costly: The Effect of Class Distribution on TreeInduction. *Journal of Artificial Intelligence Research*,19:315-354.	Provost, F., & Weiss, G.M. (2003).	Valuable analysis on the relationship between training data class distribution and classifier performance (accuracy and AUC). If accuracy is performance metric: then best class distribution tends to approximate the naturally occurring class distribution If AUC is performance metric: then the best class distribution tends to be near the balanced class distribution.	1) Only use C4.5 as base learners; 2) Possible error arising from the problem of multiple comparisons (Jensen & Cohen, 2000) since the best distribution is evaluated between 13 different distributions. 3) There is an issue of statistical significance arising from (2) because classifiers are generated for 13 training distributions. 4) "…in 50 out of 52 cases the optimal range are contiguous, assuaging concerns that our conclusion are due to problems of multiple comparisons"

Survey of fraud detection techniques. ***IEEE International Conference on Networking, Sensing and Control (*2):749-754**	Yufeng Kou, Chang-Tien Lu, S.Sirwongwattana and Yo-Ping Huang.(2004).	The paper presents a survey of current techniques used in credit card fraud detection.	1) It could be beneficial to incorporate spatial information into detection systems (so that local of transaction to billing addresses may be considered) 2) For meta-learning classifiers (i.e., ensemble learning) It would be meaningful to define effective selection metrics for deciding the best base classifiers. 3) For simplicity reasons, all the base learners for credit card fraud detection use the same desired distribution. It would be interesting to implement and evaluate the credit card fraud detection system by using very large databases with skewed class distributions and non-uniform cost per errors." 4) Due to security concerns, very few approaches for credit card fraud detection are publicly available. Neural network is a popular approach, but it is difficult to implement due to lack of data availability.

Table 2: Additional Gaps Identified

Study	Author(s) and Date	Relevant Findings	Gaps in the literature
A Multiple Resampling Method for Learning from Imbalanced Data Sets. *Computational Intelligence*, 20: 18-36	Estabrooks, A., Jo, T. and Japkowicz, N. (2004)	In comparison to Adaboost and standard C4.5 a combination of different re-sampling based (C4.5) learners is more effective for imbalanced text classification	1) Only C4.5 learners are tested 2) There is no way to know which classifier is most valuable to the final classifier group. 3) The context is of naive over and under sampling scheme: it requires more testing on different and more complex domains
A Study of the Behavior of Several Methods for Balancing machine Learning Training Data. *SIGKDD Explorations*, (6): 20-29.	Batista, Gustavo & Prati, Ronaldo & Monard, Maria-Carolina. (2004)	An analysis of the behaviour of several over and under-sampling methods for the class imbalance problem, show that: 1) SMOTE + Tomek and SMOTE + ENN showed good results, especially when the dataset had very few positive (minority) examples. 2) For dataset with larger amounts of positive (minority) examples, random over-sampling method (which is computationally less demanding) produced meaningful results	1) Only C4.5 base learners are tested using the AUC measure; no ROC curve analysis is made 2) The datasets that were tested are fairly small: The 2 largest evaluated datasets contained only 20000 examples; the 2 smallest contained 90 and 194 examples. 3) Allocating half the training examples to the minority class does not provide better results

Table 2: Additional Gaps Identified

Study	Author(s) and Date	Relevant Findings	Gaps in the literature
Automatically countering imbalance and its empirical relationship to cost. *Data Mining and Knowledge Discovery,* 17(2): 225–252	Chawla, N.V., Cieslak, D.A., Hall, L.O. et al. (2008)	Wrapper-based paradigm approach for finding the optimal percentages for undersampling and SMOTE results in effective generalisation performance (outperforming many cost-sensitive learners in realistic cost scenarios).	1) Only C4.5 base learners and Ripper (rule-based learner) are tested 2) Wrapper method based on greedy search algorithm that may not be optimal for large (enterprise) level data or for dynamic learning.

Table 2: Additional Gaps Identified			
Study	**Author(s) and Date**	**Relevant Findings**	**Gaps in the literature**
A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. ***IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews*****), 42(4): 463 - 484**	Galar, Mikel & Fernández, Alberto & Barrenechea, Edurne & Sola, Humberto & Herrera, Francisco. (2012).	Provides a good holistic view and analysis of algorithms for solving class imbalance. Results show that: 1) SMOTEBagging, RUSBoost, and UnderBagging show most robust behaviour. The algorithm SMOTEBagging ranks best between the three. But RUSBoost excels if computational complexity (and comprehensibility) is considered. 2) The trade-off between performance and complexity of ensemble learning algorithms is positive as results significantly improve with higher complexity. 3) Empirical evidence of the positive synergy between sampling techniques and Bagging ensemble learning algorithms.	1) Only uses AUC as performance measure for evaluating different models. 2) No detailed ROC curve analysis is made 3) The highest imbalance ratio (IR) is IR= 128.87 (For which the best bagging algorithm was UB4 that keeps all minority cast instances) the rest are all under 40.

Table 2: Additional Gaps Identified

Study	**Author(s) and Date**	**Relevant Findings**	**Gaps in the literature**
Learning from Imbalanced Data. ***IEEE Transactions on Knowledge and Data Engineering*****, vol. 21, no.9, pp. 1263-1284, Sept. 2009.doi:10.1109/TKDE.2008.239**	H. He and E. A. Garcia (2009)	Created a review of imbalanced data research and formulated an assessment of possible opportunities and challenges.	The authors of this paper identify some of the fundamental problems that require further investigation and that must be addressed: 1. “What kind of assumptions will make imbalanced learning algorithms work better compared to learning from the original distributions?” 2. “To what degree should one balance the original data set? “ 3. “How do imbalanced data distributions affect the computational complexity of learning algorithms?” 4. “What is the general error bound given an imbalanced data distribution?” 5. “Is there a general theoretical methodology that can alleviate the impediment of learning from imbalanced data sets for specific algorithms and application domains?”

TABLE 2: ADDITIONAL GAPS IDENTIFIED IN IMPORTANT LITERATURE

3. DESIGN & METHODOLOGY

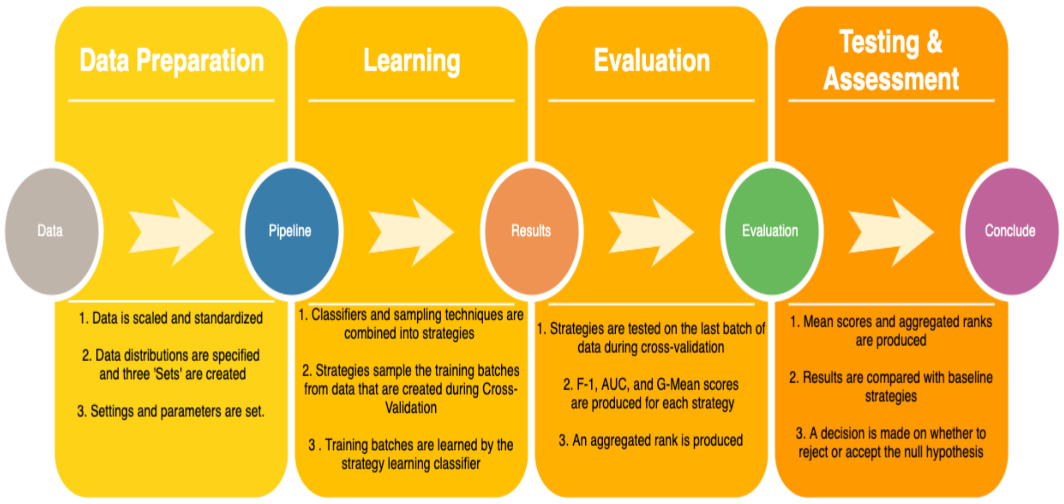
3.1 Introduction

This chapter will explain the design choices for this experiment. For this project, different learning algorithms will be used to learn from the same dataset using different sampling methodologies. This project follows a design and experimental methodology that follows the methodology outlined by CRISP-DM (Cross Industry Standard Process for Data Mining) (Shearar, 2000) for data mining related tasks.

This chapter starts by describing the experimental framework and providing a high-level view of the design. The hypotheses that will be tested are then described in greater detail in the next section. Further, the software, libraries and tools that were used in the experiment are defined. The chapter will then describe the dataset used in the experiments and any preparations that were subsequently done. The chapter will then explain the metrics that were chosen to be used for evaluating performance, followed by a brief description of the classifiers and the oversampling techniques tested in the experiments.

Lastly, the final sections in this chapter outline the settings and parameters of the techniques that were used followed by a brief description of the pipeline method that was created for the experimental testing of the strategies.

3.2 Experimental Framework



Data Preparation
Learning
Evaluation
Testing & Assessment
Data
Pipeline
Results
Evaluation
Conclude
1. Data is scaled and standardized
2. Data distributions are specified and three 'Sets' are created
3. Settings and parameters are set.
1. Classifiers and sampling techniques are combined into strategies
2. Strategies sample the training batches from data that are created during Cross-Validation
3 . Training batches are learned by the strategy learning classifier
1. Strategies are tested on the last batch of data during cross-validation
2. F-1, AUC, and G-Mean scores are produced for each strategy
3. An aggregated rank is produced
1. Mean scores and aggregated ranks are produced
2. Results are compared with baseline strategies
3. A decision is made on whether to reject or accept the null hypothesis

FIGURE 1: HIGH LEVEL DESCRIPTION OF EXPERIMENTS

A high-level description of the experimental framework is shown in Figure 1 above. The objective of the framework is to produce the results for each oversampling method in combination with each classifier for all three different sets of the data. The combination of the sampling method plus the classifier is referred to as a strategy. The results of each strategy are calculated for each of the three metrics F-1, ROC, and G-mean. The results of each strategy are obtained by using a tenfold stratified cross-validation, resulting in 147 strategy scores for each of the three sets (total of 441).

During cross-validation the Set is split into ten partitions, nine of which are used to train the model while the last remaining (hold-out) partition is used to test the model. A ranking score is applied to each metric to compare the performance of the oversampling methods and the classifiers. The ranking score for the best performing method is the max rank obtained (1 is the highest number obtained).

3.3 Hypotheses

3.3.1 Baseline Hypothesis

Null: A classification model that is trained using a learning algorithm that employs SMOTE or SMOTE-based over-sampling techniques will have a statistically significantly higher F-measure score, Receiver operating characteristics (ROC) score and G-mean score in comparison to the same learning classifier that uses random-over-sampling or no sampling technique, ceteris paribus.

Alternate: A classification model that is trained using a learning algorithm that employs SMOTE or SMOTE-based over-sampling techniques will not have a statistically significantly higher F-measure score, Receiver operating characteristics (ROC) score and G-mean score in comparison to the same learning classifier that uses random-over-sampling or no sampling technique, ceteris paribus.

Method of Evaluation: This hypothesis will be tested and evaluated by the Experimental Results Part I. It will be evaluated by calculating the difference in performance between strategies that use no sampling and random oversampling in comparison to strategies that use another sampling technique.

.

3.3.2 Secondary Hypothesis

Null: A strategy that uses a combination of SMOTE (for over sampling) and some additional extension for undersampling or data cleaning will not achieve a higher F-1 score in comparison to a strategy that uses only one type of sampling (either SMOTE or Random oversampling).

Alternate**:** A strategy that uses a combination of SMOTE (for over sampling) and some additional extension for undersampling or data cleaning will achieve a higher F-1 score in comparison to a strategy that uses only one type of sampling (either SMOTE or Random oversampling).

Method of Evaluation: This hypothesis will be tested and evaluated by the results described in Part II of Experimental Results of Chapter 4. It will be evaluated by calculating an aggregated rank (that combines the rank of a strategy based on its rank form all three performance metrics) to place the strategies. The mean performance of the strategies will also be evaluated.

3.4 Software

The experiments were carried out using Python scripts. The project is heavily reliant on the following libraries: Pandas, NumPy, Scikit-Learn (Pedregosa et al., 2012) and Imbalanced-learn (Lemaitre et al., 2016).

While Scitkit-Learn library is a library that is well-known for machine learning and data analytics, the latter is machine learning library that is specifically designed for dealing with imbalanced datasets. NumPy and Pandas are all also very popular libraries that are commonly used for machine learning and data analysis. The version of Scikit-Learn that is used is version 0.22.1 and the version of Imbalanced-Learn that is used is version 0.4.3.

For data visualisations, this project relied on using Ggplot2, Seaborn and Matplotlib. For statistical testing and analysis this experiment used SPSS Statistics software.

3.5 Metrics for Performance

Three different metrics will be used to compare the various different strategies. These metrics are the following:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{9}$$

$$AUC = \frac{1 + TP_{rate} - FP_{rate}}{2} \tag{10}$$

$$G - Mean = \sqrt{\frac{TP}{(TP+FN)} \cdot \frac{TN}{(TN+FP)}} \tag{11}$$

These metrics have been chosen because they have been widely accepted and deemed as the more appropriate metrics for measuring performance under imbalanced conditions.

Moreover, a rank was produced for each metric. Therefore, three different ranks (based on each metric) were calculated for every strategy. Each of these ranks placed strategies relative to the maximum score obtained for that metric. Using all three metric ranks, an aggregated rank was calculated. The aggregated rank of each strategy was calculated by adding the individual rank per strategy for each metric.

$$AR = F_{rank} + AUC_{rank} + G_{rank} \tag{12}$$

In order to provide a better understanding of the differences in performance between the strategies, the aggregated rank was used instead of a mean aggregated rank (by diving the AR by the number of metrics, that is, three). This was done because it is easier to visualize discrepancies that are somewhat minor or negligible and hence difficult to divulge when using a mean aggregated rank.

3.6 Dataset

The credit card fraud dataset contains a set of transactions occurring from 01/02/2012 to 20/05/2013. There are 284,807 total transactions out which 492 are positive (i.e., fraudulent) and 284315 are negative (i.e., non-fraudulent). The dataset is considered to be extremely unbalanced since the percentage of fraudulent transactions is less than one percent (0.172%) while non-fraudulent transactions represent an overwhelming majority of approximately 99.83%. The dataset includes 30 features out of which 28 have been transformed (and anonymised) through PCA. In order to implement PCA, it is necessary to scale features before. Therefore, it is reasonable to assume that all the features were previously scaled by the original authors of the dataset.

3.7 Data Preparation

All instances of the dataset have been used since there were no missing values for any of the class instances. The only features that had not been transformed are the features 'Time' and 'Amount'. Consequently, the features 'Time' and 'Amount' were also scaled in order to maintain a uniform scale. The features were scaled using robust scaling from the Scikit-Learn library. In order to maintain the heterogenous nature of the data, no outliers were removed, and all the data points were preserved.
In order to learn more about the relationship between the oversampling strategies and performance, the 49 different strategies were also tested in with two different sets (of the same dataset) with different skewness of class distributions. This was done to simulate the effect of including undersampling with the oversampling strategies that will be tested in this experiment. The first set will not use any undersampling of the majority class, the second set will apply undersampling by a factor two-thirds of the majority class, and lastly, the third set will undersample by a factor of one-thirds of the majority class.

Set 1: Set 1 contains all the data from the original dataset. The initial distribution and ratio between fraud to non-fraud is maintained so that there is a total of 492 positive (fraud) cases and 284315 negative (non-fraud) cases. The original ratio is not modified (0.17: 99.83) so that this case constitutes an example of unchanged distribution.

Set 2: Set 2 comprises of a generated split of 492 positive (fraud) cases to two-thirds of the negative (non-fraud) cases. This Set uses all 492 instances of all fraud cases and 189543 of non-fraud instances that are chosen at random. This set has chosen to under-sample the majority class by two-thirds to evaluate the effect of combining oversampling with undersampling (a factor of two-thirds) to alleviate class imbalance.

Set 3: Set 3 also comprises of a generated split of 492 to one-third of the negative (non-fraud) cases. This Set uses all 492 instances of all fraud cases and 94772 of non-fraud instances that are chosen at random. This set under-samples the majority class by one-third to evaluate the effect of combining oversampling with undersampling (by a factor of one-third) to alleviate the class imbalance.

3.8 Classifiers and Learning Algorithms

The classifiers that are tested in the experiments are the following:

1) **EasyEnsemble Classifier**: This popular ensemble technique was first proposed by Liu, Wu, and Zhou (2008). It is an ensemble technique that relies on under-sampling a subset of the majority class to create a more balanced dataset. The classifier is then trained on the under-sampled test set repeatedly until test predictions are aggregated. This learning classifier was obtained from the library Imbalanced Learn. This classifier is specifically designed to deal with imbalanced data.

2) **Random Forest Classifier**: This ensemble technique was first introduced in "Random Forests" by Breiman (2001). It combines a number of unpruned classification or regression trees into an ensemble, hence its name. Training data is bootstrapped, and random feature selection is used during the tree induction process. This ensemble method was obtained from the library Scikit-Learn. Contrary to the original publication, instead of letting each classifier tree vote for a single class, this version of the Random Forest combined classifiers by averaging its probabilistic predictions.

3) **Logistic Regression**: This is model that was originally designed as a general statistical model (used for regression rather than classification) that was originally developed and popularised by Joseph Berkson (1944) whereby "logit" term for the respective function was coined. This model was obtained from the Scikit-Learn library. In this version of the model, the probabilities of an outcome for a signed trail are modelled using a logistic function. Moreover, regularisation is applied to data by default.

4) **Balanced Random Forest Classifier**: This ensemble technique was proposed by Chen, Chao, Liaw and Breiman in "Using random forest to learn imbalanced data" (2004). This technique is similar to the Random Forest technique, but each tree instead is provided a balanced (the majority class is under-sampled) bootstrap sample of the data. Originally, the Random Forest model was built to help

minimise the overall error rate of the traditional single decision tree (such as CART or C4.5). For that reason, it tends to suffer in imbalanced domains, since it will often focus favour maximizing the prediction accuracy of the majority class at the expense of accurately predicting minority class. Consequently, to help alleviate this problem the Balanced Random Forest (BRF) was introduced. This model was obtained from the library Imbalanced-Learn.

5) **BalancedBagging Classifier**: This is an ensemble classifier that relies on using bagging methods to build several estimators on differently randomly selected subsets of the data. This is a bagging classifier with additional balancing that was introduced to counteract the original Bagging Classifier method's favouritism towards the majority class. This classifier was obtained from the Imbalaned-Learn library. This technique is similar to the BaggingClassifier technique from the Scikit-Learn implementation but instead included an additional step that helps balance the training data by randomly under-sampling the majority class during training. This technique is specifically designed for dealing with imbalanced datasets.

6) **RUSBoost Classifier**: This algorithm was introduced by Seiffert, Khoshgoftaar, Hulse and Napolitano in "RUSBoost: A Hybrid Approach to Alleviating Class Imbalance" (2010). The algorithm is meant to provide a simpler and faster alternative to the SMOTEBoost algorithm (that is based on the AdaBoost algorithm) by combining boosting and under-sampling (in contrast to the use of SMOTE sampling by the SMOTEBoost algorithm) to deal with imbalanced datasets. This classifier was obtained from the Imbalanced-Learn library.

7) **Decision Tree Classifier**: The version of tree used in this experiment is CART (Classification and Regression Trees). It is very similar to the C4.5 tree but it different in that it does not compute rule sets. CART constructs binary trees that use data features and thresholds that maximise the information gain at each node. Decision Trees are built using a heuristic method referred to as recursive partitioning, hence the algorithm relies on a divide-and-conquer approach since it splits data into subsets which are then repeatedly splits the data into smaller

subsets continuously until the algorithm decides that each subset is sufficiently homogenous or until it reaches a certain stopping criterion.

3.9 Oversampling Techniques and Algorithms

The oversampling techniques that are tested in this experiment are the following:

1) **Random Oversampling**: This sampling technique is referred to as the 'naive' form of over-sampling, since it generates new data samples from the minority class by randomly selecting instances of the minority class with replacement and adding them to a new training set. Two things are worth noting. Firstly, instances are randomly chosen from the original training set and not the newly generated training set. If the latter were true, this would mean that instances of the minority class would be randomly selected from the (new) generated training set, thus this would bias the randomness of the selection procedure. Secondly, the random oversampling is always done with replacement (meaning that we replace the data that we select back into the original training set) and that the probability of being selected is always independent previous selections. If we were to select without replacement, then this would mean that we could run out of members of the minority class before reaching the desired level of re-balancing to the distribution of each class. This technique was obtained from the Imbalanced-learn library.

2) **SMOTE**: This technique is the one used by as it was originally introduced in the work “SMOTE: synthetic minority over-sampling technique,” by Chawla, Bowyer, Hall and Kegelmeyer (2002). The technique that is used in the experiment was obtained from the Imbalanced-learn library.

3) **Borderline SMOTE**: This extension to the original SMOTE technique was introduced in the work “Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning,” by Han, Wen-Yuan, Bing-Huan (2005). The difference between SMOTE and this extension is that only borderline samples are used to generate new synthetic instances to the minority class. The technique that is used in the experiment was obtained from the Imbalanced-learn library.

4) **SVM-SMOTE**: This extension to the original SMOTE technique was introduced in the work “Borderline over-sampling for imbalanced data classification,” by Nguyen, Cooper and Kamei (2009). This technique uses an SVM algorithm to detect samples of the minority class that by generating new synthetic data around the borderline between the two data classes (Nguyen et. al, 2009). The technique that is used in the experiment was obtained from the Imbalanced-learn library.

5) **SMOTE-ENN**: This algorithm is an extension to the original SMOTE that was introduced in“ A study of the behavior of several methods for balancing machine learning training data,” by Batista, Pratti and Monard (2004). It is a variant of SMOTE that uses edited nearest-neighbours as a data cleaning method that is used after applying the SMOTE over-sampling to provide a cleaner data space. In this sense, it is also a combination of over and under-sampling for it removes any instances of data deemed to be redundant according to the edited nearest-neighbours algorithm. The technique that is used in the experiment was obtained from the Imbalanced-learn library.

6) **SMOTE-Tomek**: This algorithm is an extension to the original SMOTE That was introduced in "Balancing Training Data for Automated Annotation of Keywords: a Case Study," by Batista, Bazzan and Monard (2003). Similarly to the SMOTE-ENN technique, this technique is also a variant of SMOTE that utilises an additional technique for data cleaning, in this case that technique is Tomek’s Link. SMOTE is applied first to the dataset, creating new synthetic observations. Subsequently, Tomek’s link undersampling is applied to the new dataset (that contains the synthetic observations) to remove any pairs of examples that form a Tomek’s link. Tomek’s link is a link between two data points that are defined by a combination of two things: Firstly, they must be nearest neighbours; secondly, they must have different class labels (Tomek, 1976). Examples that are Tomek’s link are more likely to be either noise or points that are close to the optimal decision boundary (Kubat & Matwin, 1997).

7) **No sampling**: As a control group, there will be a sampling strategy that involves using no additional sampling technique in conjunction with a classifier.

3.10 Settings and Parameters

Only two parameters have been modified to differ from the default parameters of used by the Python libraries. The first parameter that has been modified from the default base estimator (AdaBoost) is that of the base estimator for the EasyEnsemble Classifier. The base estimator was set to use a Random Forest Classifier as suggested by Dal Pazzolo (2015). The second parameter that has been modified from default is for the sampling technique SVM-SMOTE. For this sampling technique, the support vector machine estimator was set use a sigmoid kernel. The default base estimator is Gaussian (RBF). However, due to the large size of the dataset, this change was done to help accelerate the long processing time that this sampling technique.

This experiment uses the default settings of all classifiers and sampling techniques in order to allow for greater reproducibility. For reproducibility, the experiment can be re-created by using a pipeline method[7] and setting the random state variable (random_state) to 42.

3.11 Pipeline

This experiment follows a pipeline that was constructed specifically for implementing this experiment. The pipeline was built by first taking the dataset and applying a ten-fold stratified cross-validation split on the data. It is important to note that sampling methods are applied during the stratified ten-fold validation process (i.e., during the stratified split loop). This way we avoid the possibility of leaking data into our testing set and hence overfitting the models.

The pipeline uses a value of 'K' cross-folds equal to ten. This means that the data is split into ten even batches whereby nine out of the ten batches are used for the training set and only one batch is used for testing and validating the model performance (i.e., verifying the model's predictions to determine its overall performance).

[7] Pipeline is available online (https://github.com/puzzle91/Model-Pipeline-Machine-Learning)

It is critical that the validation/test set not be influenced or manipulated to preserve authenticity of the original data (so as to ensure our measures of prediction are validated). Therefore, when creating random or synthetic data-points to over-sample the minority class, these data-points must be created "during" the cross-validation process and not before. To understand why, suppose that the SMOTE algorithm takes the minority class ("Fraud") and creates synthetic data instances of the class before the cross-validation phase takes place. In this case, then the validation/test set will be biased since we would have included new synthetic data points before cross-validation and hence before creating a validation dataset. This is referred to as a “data leakage” problem. For this reason, sampling must be done during cross-validation and never before.

A stratified K-fold split was used instead of regular cross-validation split because the former retains the initial distribution of classes from the dataset for every fold. This means that the uneven distribution of classes is maintained for every batch of the split, whereas during cross-validation the distribution is random and sometimes rebalanced to an even ratio. When dealing with imbalanced datasets, stratified k-fold can be used instead of traditional cross k-fold validation because it is important to preserve the heterogeneous distribution of the original dataset in order to better understand the behaviour of the strategies we apply to deal with the class imbalance.

4. EXPERIMENTAL RESULTS, EVALUATION & DISCUSSION

4.1 Experimental Results: Part 1 (Evaluating the baseline Hypothesis)

The objective of Part I of experimental results is to be able to evaluate and make a decision on whether to reject or accept the baseline hypothesis.

Baseline Hypothesis: A classification model that is trained using a learning algorithm that employs SMOTE or SMOTE-based over-sampling techniques will not have a statistically significantly higher F-measure score, Receiver operating characteristics (ROC) score and G-mean score in comparison to the same learning classifier that uses random-over-sampling or no sampling technique, ceteris paribus.

In order to test this hypothesis, the experiment should provide evidence that can be used to either support or reject the baseline hypothesis. Therefore, for each of the three Sets of data every strategy (composed of a classifier and a sampling technique) was tested and the F-1, ROC and G-mean scores were compared to, firstly, the same baseline model using no additional sampling technique) and, secondly, the same baseline model using random oversampling. Seven different classifiers were tested in conjunction with seven different sampling strategies and with three different metrics for performance, this gives a total of 147 (7x7x3) metric results that were calculated from the experiments. There are 42 (7x2x3) different metric values for the two baseline strategies and 105 (7x5x3) different metric values for each of the alternative strategies. There are two baseline versions of each strategy. Therefore, there are two comparisons based on 6 different measures of difference in scores. The difference is calculated simply by subtracting the baseline's strategy score from the alternate strategy's score.

$$Difference = Diff_{metric} = Score_{alternate} - Score_{baseline} \tag{13}$$

If the difference is found to be positive, then this would indicate that the alternate strategy achieved superior performance in comparison to the baseline strategy. If the difference is negative, then this indicates that the alternate strategy is inferior.
The count of superior strategies to inferior strategies was compared and used for deciding whether to accept or to reject the baseline hypothesis by majority voting. Therefore, if at least 53 out of the 105 metric scores are superior to the baseline scores achieved then this would indicate that there is evidence to support rejecting the null hypothesis and accepting the alternate hypothesis.

4.2 Experimental Results (Part 1)

4.2.1 Results 1: Comparison with No sampling (Set 1)

Table 3: Set 1 - Comparison with no sampling

Metric	Number of Superior Strategies	Number of Inferior Strategies
F-1	27	8
ROC	18	17
G-Mean	18	17
Total (N=105)	63	42
% of all Total (105)	60%	40%

TABLE 3: SET 1 - COMPARISON WITH NO SAMPLING

Number of strategies that show no percentage change in difference = 0
In terms of percentage difference from the total (105) this is: 0%

Majority Rule: There is sufficient evidence to support rejecting the null hypothesis and accepting the alternate hypothesis.

We can reject the null hypothesis since the number of alternative strategies that are superior strategies (63) is greater than the number of inferior strategies and 60% of alternate sampling strategies have superior performance in comparison to no sampling strategy in terms of F-1, ROC and G-mean scores.

4.2.2 Results 2: Comparison with Random Oversampling (Set 1)

Table 4: Set 1 - Comparison with Random Oversampling		
Metric	**Number of Superior Strategies**	**Number of Inferior Strategies**
F-1	10	24
ROC	27	6
G-Mean	27	7
Total (N=101)	64	37
% of all Total (105)	60.95%	35.24%

TABLE 4: SET 1 - COMPARISON WITH RANDOM OVERSAMPLING

Number of strategies that show no percentage change in difference = 4
In terms of percentage difference from the total (105) this is: 3.81%

Majority Rule: There is sufficient evidence to support rejecting the null hypothesis and accepting the alternate hypothesis.

We can reject the null hypothesis since the number of alternative strategies that are superior strategies (64) is greater than the number of inferior strategies and 60.95% of alternative sampling strategies have superior performance in comparison to random oversampling strategy in terms of F-1, ROC and G-mean scores.

4.2.3 Results 3: Comparison with No Sampling (Set 2)

Table 5: Set 2 - Comparison with no sampling		
Metric	**Number of Superior Strategies**	**Number of Inferior Strategies**
F-1	25	10
ROC	22	12
G-Mean	17	17
Total different (N=145)	64	39
% of all Total (105)	60.95%	37.14%

TABLE 5: SET 2 - COMPARISON WITH NO SAMPLING

Number of strategies that show no percentage change in difference = 2
In terms of percentage difference from total (105) this is: 1.90%

Majority Rule: There is sufficient evidence to support rejecting the null hypothesis and accepting the alternate hypothesis.

We can reject the null hypothesis since the number of alternative strategies that are superior strategies (64) is greater than the number of inferior strategies and 60.95% of alternative sampling strategies have superior performance in comparison to random oversampling strategy in terms of F-1, ROC and G-mean scores.

4.2.4 Results 4: Comparison with Random Oversampling (Set 2)

Table 6: Set 2 - Comparison with Random Oversampling		
Metric	**Number of Superior Strategies**	**Number of Inferior Strategies**
F-1	12	23
ROC	18	15
G-Mean	18	16
Total (N=145)	48	54
% of all Total (105)	45.71%	51.43%

TABLE 6: SET 2 - COMPARISON WITH RANDOM OVERSAMPLING

Number of strategies that show no percentage change in difference = 3
In terms of percentage difference from total (105) this is: 2.86%

Majority Rule: There is insufficient evidence to reject the null hypothesis.

We fail to reject the null hypothesis since 51.43% of alternative strategies are inferior to random oversampling. Therefore, following the majority rule argument, we fail to reject the null hypothesis since the number of strategies with superior performance in comparison to Random Oversampling is only 45.71% in terms of F-1, ROC and G-mean scores.

4.2.5 Results 5: Comparison with No Sampling (Set 3)

Table 7: Set 3 - Comparison with no sampling

Metric	Number of Superior Strategies	Number of Inferior Strategies
F-1	25	10
ROC	18	17
G-Mean	18	17
Total (N=105)	61	44
% of all Total (105)	58.10%	41.90%

TABLE 8: SET 3 - COMPARISON WITH NO SAMPLING

Number of strategies that show no percentage change in difference = 0
In terms of percentage difference from total (105) this is: 0%

Majority Rule: We accept that there is sufficient evidence to support rejecting the null hypothesis.
We can reject the null hypothesis since the number of strategies that are superior (61) is greater than the number of alternative strategies that are inferior as 58.10% of sampling strategies are superior to no sampling in terms of the metrics F-1, ROC and G-mean and 41.90% are inferior.

4.2.6 Results 6: Comparison with Random Oversampling (Set 3)

Table 8: Set 3 - Comparison with Random Oversampling		
Metric	**Number of Superior Strategies**	**Number of Inferior Strategies**
F-1	15	20
ROC	21	14
G-Mean	21	14
Total (N=105)	57	48
% of all Total (105)	54.29%	45.71%

TABLE 8: SET 3 - COMPARISON WITH RANDOM OVERSAMPLING

Number of strategies that show no percentage change in difference = 0
In terms of percentage difference from total (105) this is: 0%

Majority Rule: We accept that there is sufficient evidence to support rejecting the null hypothesis.
We can reject the null hypothesis since the number of superior strategies (57) is greater than half of the total amount of alternative strategies as 54.29% of alternative sampling strategies are superior to a Random Oversampling strategy in terms of F-1, ROC and G-mean scores and only 45.71% are inferior.

4.3 Evaluation of Baseline Hypothesis (Part I)

Total superior strategies=357
Total inferior strategies =264
Total metrics evaluated =630

The percentage of strategies superior to strategies that use Random oversampling and No sampling is 56.67% (357/630). The percentage of strategies that are inferior to strategies that use Random oversampling and No sampling is 41.90%. Following the majority rule, this suggests that there is sufficient evidence to support rejecting the null hypothesis in favour of the alternate hypothesis since 56.67% of the alternative strategies tested in this experiment were able to obtain a superior performance (in terms of F-1, ROC and G-mean) relative to the baseline strategies that used no sampling and random oversampling as the sampling techniques.

Since 357 of all metrics tested were superior to that of the baseline strategies and at least over 50% were superior and only 41.90% were inferior then we can confirm that there is evidence to reject the baseline hypothesis and accept the null hypothesis. Furthermore, from the six individual tests evaluated in Part 1 of this experiment, the only one baseline strategy produced performance scores that were superior to that of the alternate strategies. This can be seen by the results achieved by Random Oversampling in Set 2.

In this set, out of the105 different strategies tested, three strategies had no significant change (no increase or decrease) in performance and only 51.43% of strategies were inferior to Random Oversampling. This indicates that although Random oversampling achieved scores that were superior to more than half of the scores achieved by the alternative techniques, for this set of results, the difference is not particularly substantial. The fact that only 1.43% of scores are superior and that 2.86% of scores showed no significant change in performance may be indicative that, when found to be superior, Random Oversampling is only superior by a small margin and in certain cases this difference can be negligible.

4.4 Evaluation of Secondary Hypothesis (Part 2)

An aggregated rank was computed for each strategy by combining the rank of the strategy in terms of its F-1, its ROC and its G-mean performance. This aggregated rank was then used to compare the different strategies based on how well they performed across all three metrics. All three metrics are weighed with the same importance and individual ranks per metric are ranked relative to the maximum value obtained. This means that multiple strategies can therefore be placed in the same rank assuming they have scores that are evenly weighed.

The results in Part 2 of will be presented in the following order:

Results 1 (Set 1-Results 1): The results are the first collection of results from Set 1 (Set 1-Results 1) which describe the mean performance per sampling technique for Set 1. The sampling techniques will be ranked based on their mean aggregated rank score. A brief analysis of Results 1 will then be given at the end.

Results 2 (Set 1-Results 2): These are the second collection of results from Set 1 (Set 1-Results 2) which describe the mean performance per classifier for Set 1. The sampling techniques will be ranked based on their mean aggregated rank score. A brief analysis of Results 2 will then be given at the end.

Results 3 (Set 2-Results 1): The results are the first collection of results from Set 2 which describe the mean performance per sampling technique for Set 2. The sampling techniques will be ranked based on their mean aggregated rank score. A brief analysis of Results 3 will then be given at the end.

Results 4 (Set 2-Results 2): The results are the second collection of results from Set 2 which describe the mean performance per classifier for Set 2. The classifiers will be ranked based on their mean aggregated rank score. A brief analysis of Results 4 will then be given at the end.

Results 5 (Set 3-Results 1): The results are the first collection of results from Set 3which describe the mean performance per sampling technique for Set 3. The sampling techniques will be ranked based on their mean aggregated rank score. A brief analysis of Results 5 will then be given at the end.

Results 6 (Set 3-Results 2): The results are the second collection of results from Set 3 which describe the mean performance per classifier for Set 1. The classifiers will be ranked based on their mean aggregated rank score. A brief analysis of Results 61 will then be given at the end.

4.5 Experimental Results (Part 2)

4.5.1 Results 1: Sampling Technique Performance (Set 1)

Experiments on Set 1 were conducted without making any posterior modifications to the dataset size. The distribution is then 492 fraud to 284,315 non-fraud cases. Therefore, the imbalance ratio is given by: Imbalance Ratio: (N+/N-) = 492/284315 = 0.17%This value constitutes a severe imbalance ratio. Using stratified K-fold with 10 splits the training dataset size is 256327 and the testing set is 28480.

Table 9: Set 1 - Training and Processing time per sampling strategy

Sampling technique	**Approximate training and Processing time per sampling technique**
No sampling	1.5 mins
SMOTE	28 mis
Random Over Sampling	13 mins
Borderline SMOTE	34 mins
SVM-SMOTE	31 mins
SMOTE-ENN	122 mins
SMOTE-Tomek	86 mins

TABLE 9: SET 1 - TRAINING AND PROCESSING TIME PER SAMPLING STRATEGY

Results for Set 1 are shown in the next page (page 60) in Table 10.

Table 10: Set 1 - Mean Performance per Sampling techniques (relative rank in parenthesis)				
Sampling Technique	Mean Aggregated Rank	F1	ROC AUC	G mean
SMOTEENN	1 (92.571429)	0.620686 (20.142857)	**0.9423 (36.285714)**	**0.940729 (36.142857)**
SVMSMOTE	2 (85.571429)	0.725329 (28.714286)	0.928929 (28.428571)	0.926271 (28.428571)
SMOTE	3 (83.428571)	0.650971 (26.714286)	0.930814 (28.285714)	0.928371 (28.285714)
SMOTETomek	3 (83.428571)	0.650971 (26.714286)	0.930814 (28.428571)	0.928371 (28.285714)
BorderlineSMOTE	4 (66.142857)	0.725971 (30.857143)	0.915886 (17.714286)	0.912014 (17.571429)
No sampling	5 (62.285714)	0.415071 (12.571429)	0.9041 (24.571429)	0.897043 (25.142857)
Random OverSampling	6 (61.428571)	**0.7315 (31.00)**	0.913729 (15.142857)	0.909657 (15.285714)

TABLE 10: MEAN PERFORMANCE PER SAMPLING TECHNIQUES (RELATIVE RANK IN PARENTHESIS)

4.5.2 Analysis Results 1: Sampling Technique Performance (Set 1)

The results are in line with the predicted hypothesis that strategies that use either no sampling or use random oversampling as sampling techniques exhibit lower performance scores in comparison to the other strategies that were tested. SMOTE and SMOTE-Tomek exhibit the same mean results. This may indicate that the number of nearest neighbours used by SMOTE and the number of Tomek's links used by SMOTE-Tomek is the same. So, none of the nearest neighbours of minority class examples satisfy the Tomek's Link condition whereby links are created for instances of data that are both nearest neighbours and also members of the opposite classes.

Although the aggregated rank of random oversampling and no sampling are quite similar, the mean F-1 score of random oversampling is significantly higher than that of no sampling. Nonetheless, random oversampling came last in terms of aggregated rank. This may indicate that certain classifiers are made worse (relative to when using no sampling) by the inclusion of random oversampling to its pipeline when learning from the full dataset and that this effect is predominantly detrimental to the ROC and G-mean scores of the strategies.

4.5.3 Experimental Results 2: Classifier Performance (Set 1)

Table 11: Set 1 - Mean Performance per Classifier (relative rank in parenthesis)

Classifier	Mean Aggregated Rank	F1	ROC AUC	G mean
Easy Ensemble Classifier	1 (102.428571)	0.752271 (36.428571)	**0.933671 (33.142857)**	**0.931314 (32.857143)**
Balanced Random ForestClassifier	2 (95.857143)	0.747457 (35.714286)	0.9313 (29.857143)	0.9288 (30.285714)
Balanced Bagging Classifier	3 (82.142857)	0.695114 (24.714286)	0.929814 (28.714286)	0.927271 (28.714286)
Logistic Regresssion	4 (78.428571)	0.221 (7.00)	0.924329 (35.571429)	0.919614 (35.857143)
Random Forest Classifier	5 (77.00)	**0.847057 (37.00)**	0.915329 (20.142857)	0.911157 (19.857143)
RUS Boost Classifier	6 (53.571429)	0.6032 (17.00)	0.920257 (18.285714)	0.916786 (18.285714)
Decision Tree Classifier	7 (45.428571)	0.6544 (18.857143)	0.911871 (13.285714)	0.907514 (13.285714)

TABLE 11: SET 1 - MEAN PERFORMANCE PER CLASSIFIER (RELATIVE RANK IN PARENTHESIS)

4.5.4 Analysis of Experimental Results 2: Classifier Performance (Set 1)

The results agree with previous research that encourage the use of the EasyEnsemble Classifier due to its general robust behaviour and ability to perform well in highly imbalanced domains. The results also show that on the full dataset the simpler classifiers (such as, Logistic Regression, Random Forest Classifier and Decision Tree Classifier) all seem to suffer more in terms of performance in comparison to the ensemble methods that carry some form of re-balancing through either undersampling or bagging and boosting. This is in line with research that suggests that using a combination of undersampling and oversampling can be superior to using only one or the other. The Decision Tree classifier is also the worse classifier method, this is also in line with research that suggests that classifiers that rely on divide-and-conquer algorithms will typically exhibit low performance in highly imbalanced domains.

Although the logistic regression classifier came forth in terms of general performance based on its aggregated rank, this model also shows the highest degree of difference between its F-1 score/rank and its ROC and G-mean scores. This may be due to the fact that the logistic function will often tend to seek maximising the overall accuracy of the its predictions by favouring the majority class. This favouritism towards the majority class could be more predominant for the logistic regression classifier in comparison to the other classifiers. The low mean precision (0.188686) relative to the recall (0.865914) achieved by this classifier is much more disproportionate in comparison to other classifiers evaluated.

4.5.5 Experimental Results 3: Sampling Technique Performance (Set 2)

The distribution used in Set 2 is 492 fraud (all cases) to 189543 (i.e., two-thirds) non-fraud cases. The imbalance ratio is therefore: Imbalance Ratio: (N+/N-) = (492/164089) ·100 = 0.2998%

This value constitutes a severe imbalance ratio. Using stratified K-fold with 10 splits the training dataset size is 171032 and the testing set is 19003.

Table 12: Set 2 - Training and Processing time per sampling technique	
Sampling technique	**Approximate training and Processing time**
No sampling	52 secs
SMOTE	17 mins
Random Over Sampling	7 mins 30 seconds
Borderline SMOTE	17 mins 20 seconds
SVM-SMOTE	19 mins 25 seconds
SMOTE-ENN	83 mins 33 seconds
SMOTE-Tomek	61 mins 49 seconds

TABLE 12: SET 2 - TRAINING AND PROCESSING TIME PER SAMPLING TECHNIQUE

Table 13: Set 2 - Mean Sampling technique Performance				
Sampling Technique	Mean Aggregated Rank	F1	ROC AUC	G mean
SMOTE	1 (86.714286)	0.615743	**0.875443**	**0.866971**
SMOTETomek	1 (86.714286)	0.615743	**0.875443**	**0.866971**
SVMSMOTE	2 (85.285714)	**0.675814**	0.872014	0.862814
Random OverSampling	3 (76.857143)	0.672886	0.867100	0.857071
BorderlineSMOTE	4 (76.571429)	0.640900	0.857314	0.845114
SMOTEENN	5 (73.428571)	0.590043	0.872371	0.863400
No sampling	7 (61.571429)	0.402757	0.827371	0.797586

TABLE 13: SET 2 - MEAN SAMPLING TECHNIQUE PERFORMANCE

4.5.6 Analysis of Experimental Results 3: Sampling Technique Performance (Set 2)

Although the results between Random Oversampling, BorderlineSMOTE and SMOTE-ENN are comparably close in terms of mean aggregated rank, Random Oversampling ranks higher than both BorderlineSMOTE and SMOTE-ENN. This may suggest that when applied to a dataset that is re-balanced by undersampling the majority class, random oversampling can be a robust sampling strategy.
In comparison to results obtained from Set 1, SMOTE-ENN exhibits a significant decline to its overall performance. This may indicate that its performance declines as the degree of imbalance decreases. SMOTE, SMOTE-Tomek and SVM-SMOTE have similar mean aggregated ranks. SMOTE and SMOTE-Tomek exhibit the same mean results. This again may indicate that the number of nearest neighbours used by SMOTE and the number of Tomek's links used by SMOTE-Tomek is the same.

4.5.7 Experimental Results 5: Classifier Performance (Set 2)

Table 14: Set 2 - Mean Classifier Performance

Classifier	Mean aggregated Rank	F1	ROC AUC	G mean
Balanced Random ForestClassifier	1 (108.857143)	0.691471	0.877429	0.869114
Easy Ensemble Classifier	2 (106.285714)	0.687557	0.874043	0.865457
Random Forest Classifier	3 (97.428571)	**0.787857**	0.867157	0.856929
Logistic Regression	4 (87.714286)	0.260314	**0.883971**	**0.877429**
Balanced Bagging Classifier	5 (69.428571)	0.635886	0.865600	0.855286
Decision Tree Classifier	6 (41.00)	0.609514	0.854829	0.842429
RUS Boost Classifier	7 (36.428571)	0.541286	0.824029	0.793286

TABLE 14: SET 2 - MEAN CLASSIFIER PERFORMANCE

4.5.8 Analysis of Experimental Results 5: Classifier Performance (Set 2)

The EasyEnsemble Classifier and the Balanced RandomForestClassifier are again top performing classifiers. This agrees with existing literature that these classifiers are particularly robust for this kind of task and will maintain a good performance despite the reduction in the amount of data to the majority class and the change to the class distributions.

RUSBoost Classifier is significantly worse than all other classifiers. For this particular set it the worse classifier in the group. These results contradict the findings by López (2013) where RUSBoost outperformed EasyEnsemble.

4.5.9 Experimental Results 6: Sampling Technique Performance (Set 3)

Experiments on Set 3 were conducted by altering the majority class size from dataset. The distribution used is 492 fraud to 94772 (= 1/3 * 284315) non-fraud cases. The imbalance ratio is therefore: Imbalance Ratio: (N+/N-) = (492/94772) *100 = 0.5191%

This value constitutes a severe imbalance ratio. Using stratified K-fold with 10 splits the training dataset size is 85738 and the testing set is 9226.

Table 15: Set 3 - Training/Processing Time per sampling technique

Sampling technique	**Approximate training and Processing time**
No sampling	24 seconds
SMOTE	6 mins 58 seconds
Random Over Sampling	3 mins 10 seconds
Borderline SMOTE	8 mins 54 seconds
SVM-SMOTE	8 mins 2 seconds
SMOTE-ENN	32 mins 45 seconds
SMOTE-Tomek	23 mins 18 seconds

TABLE 15: SET 3 - TRAINING/PROCESSING TIME PER SAMPLING TECHNIQUE

Table 16: Set 3 - Mean Sampling Technique Performance				
Sampling Technique	Mean Aggregated Rank	F1	ROC AUC	G mean
SMOTE	1 (87.142857)	0.718129 (27.142857)	**0.924800 (30.00)**	**0.921743 (30.00)**
SMOTETomek	1 (87.142857)	0.718129 (27.142857)	**0.924800 (30.00)**	**0.921743 (30.00)**
SVMSMOTE	2 (85.857143)	**0.789843 (29.857143)**	0.918314 (27.857143)	0.914857 (28.142857)
SMOTEENN	3 (78.142857)	0.725014 (24.571429)	0.921757 (26.857143)	0.918486 (26.714286)
BorderlineSMOTE	4 (69.571429)	0.774214 (26.857143)	0.909457 (21.285714)	0.904943 (21.428571)
RandomOverSampler	5 (68.142857)	0.787714 (27.285714)	0.909586 (20.428571)	0.904871 (20.428571)
No sampling	6 (58.571429)	0.568986 (14.571429)	0.911814 (22.00)	0.907343 (22.00)

TABLE 16: SET 3 - MEAN SAMPLING TECHNIQUE PERFORMANCE

4.5.9 Analysis of Experimental Results 6: Sampling Performance (Set 3)

SMOTE and SMOTE-Tomek are again the highest performing sampling techniques for Set 3 again (after being the highest performing for Set 2).
SVM-SMOTE maintained a consistent rank relative to the previous set (Set 2). The same is true for the SMOTE-ENN technique, we can see from the minimal change in the aggregated rank (which is approximately the same) between Set 2 and Set 1. The sampling techniques that were most impacted by the decrease in the amount of negative class instances (non-fraud data) are BorderlineSMOTE and Random Oversampling. While Random Oversampling's mean aggregated rank dropped from 76.85 to 68.14 (approximately); BorderlineSMOTE's mean aggregated rank score dropped from 76.57 to 69.56 (approximately).

4.5.10 Experimental Results 7: Classifier Performance (Set 3)

Table 17: Set 3 - Mean Classifier Performance

Classifier	Mean Aggregated Rank	F1	ROC AUC	G mean
Easy Ensemble Classifier	1 (105.857143)	0.821757 (36.857143)	0.928429 (34.428571)	0.925614 (34.571429)
Balanced Random ForestClassifier	2 (105.142857)	0.822671 (36.857143)	0.928471 (34.00)	0.925657 (34.285714)
Random Forest Classifier	3 (85.428571)	**0.893700 (37.571429)**	0.911071 (24.142857)	0.906543 (23.714286)
Logistic Regresssion	4 (84.285714)	0.385343 (8.857143)	**0.935771 (37.714286)**	**0.933757 (37.714286)**
Balanced Bagging Classifier	5 (65.00)	0.748243 (21.285714)	0.913486 (21.714286)	0.909414 (22.00)
Decision Tree Classifier	6 (51.571429)	0.717471 (19.857143)	0.905329 (15.857143)	0.900571 (15.857143)
RUS Boost Classifier	7 (37.285714)	0.692843 (16.142857)	0.897971 (10.571429)	0.892429 (10.571429)

TABLE 17: SET 3 - MEAN CLASSIFIER PERFORMANCE

4.5.11 Analysis of Experimental Results 7: Classifier Performance (Set 3)

The classifiers maintained the same mean aggregated rank score that they had obtained Set 2. This could indicate that the change in the amount of non-fraud data does not have a strong influence on the average performance of these classifiers. Moreover, this could signal that the changes in performance between the different experimental Sets is predominantly the due to changes in the performance of the sampling techniques. This is a possibility since the performance of strategies (classifiers plus sampling techniques) and sampling techniques very from Set 1 to Set 3, while on average the classifiers maintain the same approximate level of performance.

Therefore, it is possible that such changes in performance are also due to the effect of changing the distribution of classes and the ratio between the majority class (non-fraud) relative to the minority class (fraud) and how certain sampling techniques are affected by these changes. More research is required to ascertain this possibility.

4.6 Evaluation of Performance based on Results across all Sets

4.6.1 Top Performers based on F-1 score

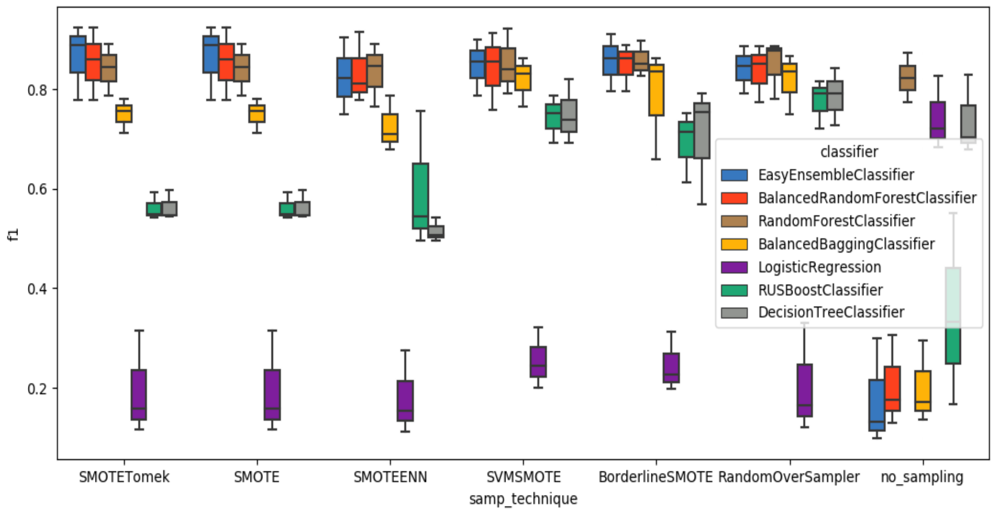


Figure 2: Boxplot for F-1 scores for all strategies across all three Sets

The results for the top performers based on F-1 score on shown in page 69 (see Table 18 on the next page).

Table 18: Top Performers for F-1 across all sets					
Rank	**Top Strategy/Strategies**	**Dataset**	**F-1 rank: F-1 score**	**ROC rank: ROC score**	**G-mean rank: G-mean score**
1 (147/147)	Group A [8, 9]	Set 3	147/147 0.9247	125/147 0.9387	125/147 0.9367
2 (143/147)	RandomForestClassifier + SVMSMOTE	Set 3	143/147 0.9231	113/147 0.9286	113/147 0.9258
3 (142/147)	BalancedRandomForestClassifier + SMOTE-ENN	Set 3	142/147 0.9149	125/147 0.9387	125/147 0.9367
4 (141/147)	BalancedRandomForestClassifier + SVM-SMOTE	Set 3	141/147 0.9130	112/147 0.9285	113/147 0.9285
5 (140/147)	EasyEnsembleClassifier + BorderlineSMOTE	Set 3	140/147 0.9111	104/147 0.9184	104/147 0.9147

TABLE 18: TOP PERFORMERS F-1 ACROSS ALL SETS

[8] Group A: 1) EasyEnsembleClassifier + SMOTE; 2) EasyEnsembleClassifier + SMOTE-Tomek; 3) BalancedRandomForest + SMOTE; and 4) BalancedRandomForest + SMOTE-Tomek. These Four different strategies ranked first in terms of F-1.

[9] These strategies obtained the same F-1. They are all strategies used on Set 3. If strategies are grouped in tables this means that they obtained the same F-1, ROC and/or G-mean scores and are all from the same ‘Set’.

4.6.2 Top Performers based on ROC score

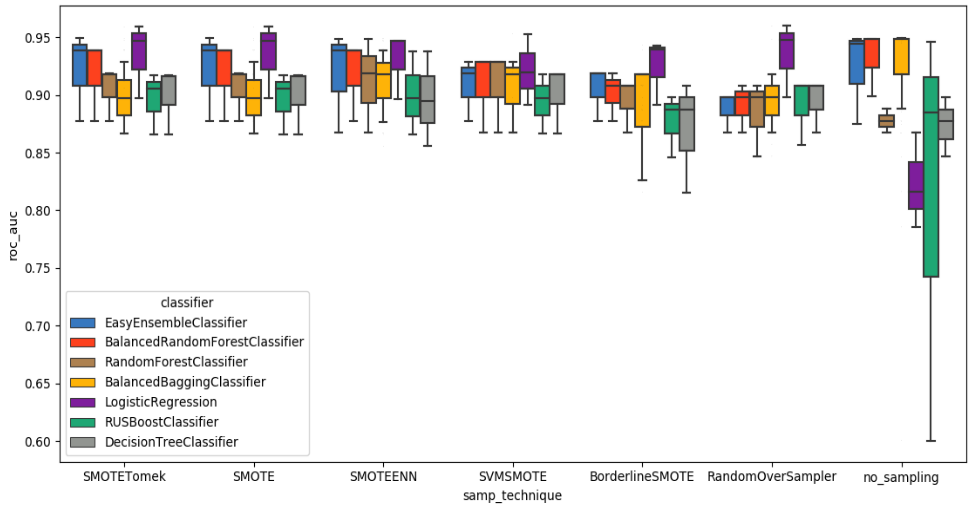


Figure 3: Boxplot for ROC scores for all strategies across all three Sets

The results for the top performers based on ROC score on shown in page 71 (see Table 19 on the next page).

Table 19: Top Performers ROC across all sets						
Rank	**Top Strategy/Strategies**	**Dataset**	**F-1 rank: F-1 score**	**ROC rank: ROC score**	**G-mean rank: G-mean score**	**Combined rank**
1 (147/147)	**LogisticRegression + RandomOverSampling**	**Set 3**	28/147 0.3297	147/147 0.9597	147/147 0.9595	322/413
2 (146/147)	**Group A**[10]	**Set 3**	26/147 0.3151	146/147 0.9286	146/147 0.9258	318/413
3 (144/147)	**LogisticRegression + SVM-SMOTE**	**Set 1**	17/147 0.2004	144/147 0.9592	144/147 0.9523	305/413
4 (143/147)	**BalancedBaggingClassifier + No sampling**	**Set 1**	8/147 0.1364	143/147 0.9492	143/147 0.9487	294/413
5 (142/147)	**Group B**[11]	**Set 1**	140/147 0.8889	142/147 0.9489	138/147 0.9475	413/413

TABLE 19: TOP PERFORMERS ROC ACROSS ALL SETS

[10] Group A: 1) LogisticRegression+SMOTE; and 2) LogisticRegression + SMOTE-Tomek.
[11] Group B: 1) EasyEnsembleClassifier + SMOTE-Tomek; and 2) EasyEnsembleClassifier + SMOTE

4.6.3 Top Performers based on G-Mean Score

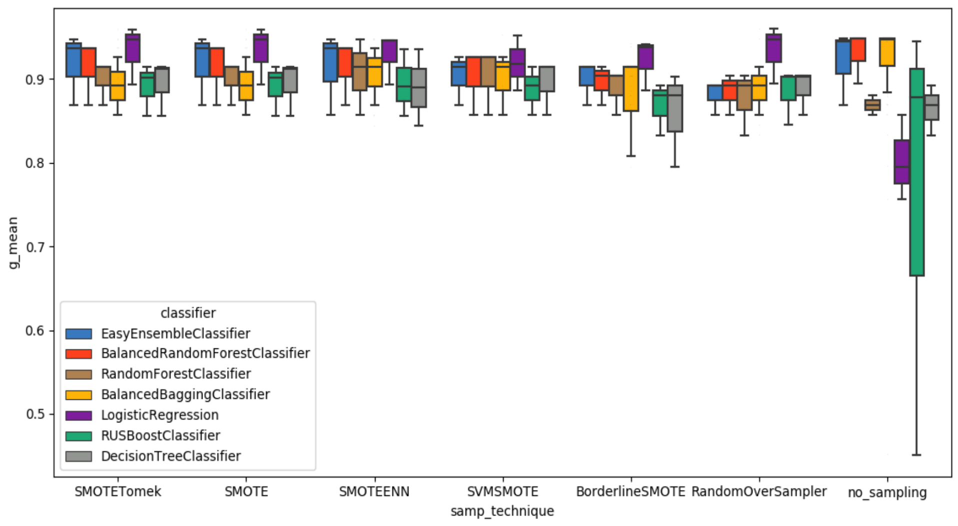


Figure 4: Boxplot for G-mean Scores for all strategies across all three Sets

The results for the top performers based on G-mean score on shown in page 73 (see Table 20 on the next page).

Table 20: Top Performers G-mean across all sets

Rank	Top Strategy/Strategies	Dataset	F-1 rank: F-1 score	ROC rank: ROC score	G-mean rank: G-mean score	Combined rank
1 (147/147)	**LogisticRegression + RandomOverSampling**	**Set 3**	28/147 0.3297	147/147 0.9597	147/147 0.9595	322/413
2 (146/147)	**Group A**[12]	**Set 3**	26/147 0.3151	146/147 0.9286	146/147 0.9258	318/413
3 (144/147)	**LogisticRegression + SVM-SMOTE**	**Set 1**	17/147 0.2004	144/147 0.9592	144/147 0.9523	305/413
4 (143/147)	**BalancedBaggingClassifier + No sampling**	**Set 1**	8/147 0.1364	143/147 0.9492	143/147 0.9487	294/413
5 (142/147)	**BalancedRandomForestClassifier + No sampling**	**Set 3**	23/147 0.3061	138/147 0.9492	142/147 0.9481	303/413

TABLE 20: TOP PERFORMERS G-MEAN ACROSS ALL SETS

4.6.4 Top Performing Strategies per aggregated rank

Results for the top performing strategies is shown on page 74 (in Table 21 below).

[12] Group A: 1) LogisticRegression+SMOTE; and 2) LogisticRegression + SMOTE-Tomek.

Table 21: Top Performers based on aggregated rank across all sets					
Combined Rank	**Strategy**	**Dataset**	**F1**	**ROC**	**G-mean**
1 (413/413)	Group A[13]	Set 1	133/147 0.8889	142/147 0.9489	138/147 0.9475
2 (397/413)	Group B[14]	Set 3	147/147 0.9247	125/147 0.9387	125/147 0.9367
3 (392/413)	BalancedRandomForestClassifier+SMOTE-ENN	Set 3	142/147 0.9149	125/147 0.9387	125/147 0.9367
4 (389/413)	RandomForest Classifier +SMOTE-ENN	Set 1	113/147 0.8462	140/147 0.9488	136/147 0.9474
5 (379/413)	EasyEnsemble Classifier+ SMOTE-ENN	Set 3	139/147 0.9053	120/147 0.9386	120/147 0.9366
6 (377/413)	EasyEnsemble Classifier+ SMOTE-ENN	Set 1	102/147 0.8224	139/147 0.9487	136/147 0.9474
7 (369/413)	RandomForest Classifier+SVM-SMOTE	Set 3	143/147 0.9231	113/147 0.9286	113/147 0.9258
8 (366/413)	BalancedRandomForestClassifer+SVM-SMOTE	Set 3	141/147 0.913	112/147 0.9285	113/147 0.9258
9 (360/413)	Group C [15]	Set 1	120/147 0.86	120/147 0.9386	120/147 0.9366
10 (348/413)	EasyEnsemble Classifier + BorderlineSMOTE	Set 3	140/147 0.9111	104/147 0.9184	120/147 0.9147

TABLE 21: TOP PERFORMERS BASED ON AGGREGATED RANK ACROSS ALL SETS

[13] Group A: 1) EasyEnsembleClassifier + SMOTE-Tomek; 2) EasyEnsembleClassifier+SMOTE

[14] Group B: 1) BalancedRandomForestClassifiers + SMOTE-Tomek; 2) EasyEnsembleClassifier + SMOTE-Tomek; 3) BalancedRandomForest + SMOTE; and 4) EasyEnsembleClassifier + SMOTE

[15] Group C: 1) BalancedRandomForestClassifier + SMOTE; and 2) BalancedRandomForestClassifier+SMOTE-Tomek

4.6.5 Top Performing Sampling Techniques per aggregated rank

Table 22: Top Mean Sampling Technique ranked by mean aggregated rank across all sets

Sampling Technique	Aggregated Rank	F1	ROC AUC	G mean
SVM-SMOTE	1 (244.142857)	0.730329 (86.571429)	0.906419 (78.904762)	0.901314 (78.666667)
SMOTE	2 (237.714286)	0.661614 (73.857143)	0.910352 (81.714286)	0.905695 (82.142857)
SMOTE-Tomek	2 (237.714286)	0.661614 (73.857143)	0.910352 (81.714286)	0.905695 (82.142857)
SMOTE-ENN	3 (233.904762)	0.645248 (67.428571)	**0.912143 (83.380952)**	**0.907538 (83.095238)**
Borderline-SMOTE	4 (219.571429)	0.713695 (85.476190)	0.894219 (67.095238)	0.887357 (67.00)
Random-OverSampling	5 (217.523810)	**0.730700 (89.00)**	0.896805 (64.476190)	0.890533 (64.047619)
No sampling	6 (179.714286)	0.462271 (44.619048)	0.881095 (66.904762)	0.867324 (68.190476)

TABLE 22: TOP MEAN SAMPLING TECHNIQUES RANKED BY AGGREGATED RANK ACROSS ALL SETS

4.6.6 Top Performing Classifiers per aggregated rank

Table 23: Top mean Classifier ranked by aggregated rank across all sets

Sampling Technique	Aggregated Rank	F1	ROC AUC	G mean
EasyEnsembleClassifier	1 (278.285714)	0.753862 (100.095238)	0.912048 (89.047619)	0.907462 (89.142857)
BalancedRandomForestClassifier	2 (275.714286)	0.753867 (98.714286)	0.9124 (88.333333)	0.907857 (88.666667)
RandomForestClassifier	3 (251.666667)	**0.842871 (111.571429**	0.897852 (70.333333)	0.891543 (69.761905)
BalancedBaggingClassifier	4 (221.571429)	0.693081 (74.047619)	0.902967 (73.809524)	0.897324 (73.714286)
Logistic Regression	5 (213.238095)	0.288886 (23.666667)	**0.91469 (93.809524)**	**0.910267 (95.761905)**
DecisionTreeClassifier	6 (168.142857)	0.660462 (59.714286)	0.890676 (54.285714)	0.883505 (54.142857)
RUSBoostClassifier	7 (161.666667)	0.612443 (53.00)	0.880752 (54.571429	0.8675 (54.095238)

TABLE 23: TOP MEAN CLASSIFIER RANKED BY AGGREGATED RANK ACROSS ALL SETS

4.7 Summary of Experimental Results, Discussion and Evaluation.

1. Across all three sets of data (with the three different distribution of fraud to non-fraud), most strategies obtained results that were superior to that of its baseline counterpart (same classifier with no sampling or random oversampling). The only experiment where the alternative strategies did not outperform the baseline strategy counterpart was in comparison to random oversampling in Set 2. It would

be interesting to investigate further why naive oversampling was able to outperform some of the other sampling techniques for this Set and not in others.

2. From Experimental Results Part 2, we can see that the mean aggregated rank of SVM-SMOTE achieved the highest score. This could suggest that this sampling technique is more robust than the other sampling techniques evaluated in this experiment. In second place came both SMOTE and SMOTE-Tomek while SMOTE-ENN came third. These findings are in line with previous research obtained (Dal Pazzolo et al., 2013; Dal Pazzolo, 2015).

3. The results show that the highest performing technique (SVM-SMOTE) is a variant of the algorithm that generates synthetic data around the borderline that surrounds the two classes. Since SVM-SMOTE outperformed the other sampling techniques, this may suggest that, for this particular dataset, an oversampling technique that has a different initial selection procedure and can account for the borderline cases of class groups may improve the performance of a strategy. Future research could develop on this possibility.

4. Furthermore, for the classifiers, the RUSBoost Classifier was significantly worse than all other classifiers. This was the case for every Set except Set 1, where it achieved the penultimate rank (Decision Tree came last) in terms of mean performance. Overall, it was the worse classifier in the group. These results contradict the findings in López (2013), where RUSBoost was robust and outperformed EasyEnsemble.This may indicate that naive undersampling on data (that has already undergone under-sampling) in combination with boosting (a technique that seeks to reduce bias) may not be a good strategy for this particular dataset. More research is needed to ascertain this by, for example, testing how the RUSBoost Classifier's performance compares relative to its SMOTE counterpart, the SMOTEBoost algorithm

5. The EasyEnsemble classifier was the best performing classifier in the group. BalancedRandomForest Classifier and the RandomForest Classifier also proved to be robust classifiers. These results are in line with the results obtained in previous research (Liu, wu, and Zhou, 2009; López 2013; and Dal Pazzolo et al., 2013).

6. The three sampling techniques that are ranked best across all three metrics are SVM-SMOTE, SMOTE, and SMOTE-Tomek, in that particular order. This may signal that for this specific dataset, SMOTE variants that are able to modify the initial selection to space for which synthetic examples are generated may be the superior strategy. Future research could investigate whether SMOTE variants that modify the initial selection procedure are, in fact, consistently superior to variants that include some data cleaning process (e.g., SMOTE-ENN, and SMOTE-Tomek).

7. The highest-ranked strategies include strategies that involve BalancedRandomForest Classifier and EasyEnsemble Classifier and some version of the SMOTE technique. This indicates that as far as classifiers are concerned, these classifiers are most robust and that when coupled with SMOTE, they are able to achieve a significantly higher performance in comparison to the other strategies. Similarly, SMOTE-ENN also showed to be a robust strategy when paired with these classifiers.

8. Across all three sets, none of the top ten best performing strategies were found from testing using Set 2. This may signal that the performance of certain techniques is influenced by class distributions. Therefore, it is possible that performance can be optimised if a more appropriate class distribution (from undersampling the majority class) is identified. More research could be done to find the optimum distribution for the maximising the performance of the sampling techniques.

9. Assuming both SMOTE and SMOTE-Tomek were implemented correctly from the Imbalanced-Learn library, then it is possible that this is due to the algorithm's inability to identify any Tomek's link in the data. Perhaps it is for this reason that the results of the two sampling techniques are exactly the same. Notwithstanding, this is a limitation of the experiment, and it would be essential to understand the source of this error. This could be a valuable avenue for future research to investigate.

4.8 Evaluation of Experimental Design

4.8.1 Experimental Strengths

1. Ease of Replication: The experiment was designed through a pipeline method that allows this experiment to be replicated and for results to be reproduced. Most settings used for techniques are default settings (except for two settings). The experiment also does not remove any outliers in order to help facilitate the replicating the experiment. Some research was done on including a method to detect and remove outliers using an Isolation Forest algorithm. However, since no significant improvement in performance was observed (on the contrary, performance declined), the use of Isolation Forest as a data cleaning method was not included in this experiment.

2. The use of three metrics allows for a more extensive analysis: The fact that this experiment uses three metrics for assessing performance allows the experiment to gain more insight into what influences performance. This also means that the aggregated rank metric (that combines ranks from all three metrics) is able to provide a more well-rounded evaluation of performance.

3. The sampling techniques and classifiers used are well-known and easy to implement and use: The sampling techniques and the classifiers that were evaluated in this experiment are popular techniques that can be easily obtained and implemented from the popular Python library (Imbalanced-Learn).

4. The experiment includes an additional evaluation of classifier performance: By also including data on the performance of classifiers (as well as the performance of sampling technique), more information on the relationship(s) between classifiers, sampling techniques, and data is produced. This allows for more insight into what is more or less critical for influencing the performance of the strategies for this dataset.

4.8.2 Experimental Weaknesses/Limitations:

1. Lack of hyper-parameters and optimization: Due to time constraints, the optimal hyper-parameters for each strategy has not yet been found. This could be an area for future research.

2. Possible bias in the Aggregated rank: An aggregated rank score was composed for each strategy by combining the F-1, ROC, and G-mean ranks of each strategy. Ranks are based on a score that is relative to the maximum score obtained for that metric. This means that ranks can be influenced by the presence of outliers like abnormally high or low scores. This may be a source of bias to the ranking measure. A mechanism should be included in the measure to reduce or prevent this bias.

3. No optimal distribution: The experiment could be enhanced by extending research that identifies the optimal distribution between the majority and the minority classes and for the strategies evaluated.

4. Pre-transformed data: Because the data is already transformed by PCA, we are not able to know what the features originally were before the transformation was applied. Moreover, we must assume that the data has been standardized and normalized (since this is an essential step to performing PCA). However, since we do not know how the procedure was done, we must assume it was performed correctly.

5. No additional repeats during Cross-Validation: The experiment could be enhanced by including additional repeats during the stratified cross-validation process. This would give the experiments greater certainty regarding the validity of the results obtained.

6. Lack of control during the oversampling process: This experiment chose to use mostly all the default parameters of classifiers and sampling techniques. This comes with a drawback as it means that there is no control over the

sampling process and the proportion/amount of synthetic data that is generated is unknown.

5. CONCLUSION

5.1 Summary of Findings

The first investigation was conducted to assess the baseline hypothesis that tests whether the strategies that use SMOTE or one the variants tested in this experiment will achieve higher performance in comparison to techniques that use random oversampling or no sampling. This research found that: The first and second sets of experimental results indicate that SMOTE and extensions to the SMOTE technique are, on average, more effective as strategies in terms of the three performance metrics in comparison to strategies that use random oversampling or no sampling. Therefore, there is sufficient evidence to reject the baseline null hypothesis.

A second investigation was conducted and in order to evaluate the secondary hypothesis that the SMOTE-variants that include a mechanism for data cleaning (SMOTE-Tomek and SMOTE-ENN) would achieve a better performance relative to the other sampling techniques tested in this experiment. This research found that: The results indicate that SVM-SMOTE, SMOTE, SMOTE-Tomek, and SMOTE-ENN are all robust oversampling techniques that can be used for this dataset. The best classifiers were found to be the EasyEnsemble Classifier and the BalancedRandomForest Classifier, in particular. The RandomForest Classifier also showed good results. Strategies that include these classifiers or sampling techniques are therefore advised for this particular dataset.

Since both SMOTE and SMOTE-Tomek exhibit the exact same results, this could mean that SMOTE-ENN was the only data cleaning variant of the algorithm that worked (as intended) in this experiment. Whether this is due to an error or intrinsic data characteristics is an avenue for future research. As a result, this experiment was only able to, ultimately, produce one set of results for data-cleaning variants of the technique. Therefore, its ability to evaluate whether these techniques are on average superior for this specific dataset is limited. Moreover, since the average performance (across all three sets) of SMOTE-SVM and the original SMOTE technique was higher than that of the SMOTE-ENN, this also suggests that for this specific dataset, SMOTE variants that include data cleaning methods are albeit robust not the best sampling techniques.

5.2 Future Work and Recommendations

1. Future research could be done to identify the optimum parameters that help reduce the training time while maintaining similar performance.

2. Additional research should be included to understand better why SMOTE and SMOTE-Tomek achieved the same result.

3. Future research should also develop on understanding why the SVM-SMOTE technique was superior to other strategies.

4. Given a large number of SMOTE and SMOTE extensions, it would be essential to create a more systematic framework to help aid research (by providing some guidance) on what techniques are more appropriate depending on the dataset and underlying characteristics of the data. A current limitation of studies on oversampling is that strategies are formulated only a posteriori, after experimentation. Future research should look into developing methodologies that can be used before experimentation, for example, during exploratory data analysis (EDA), and that can be used to help guide researchers on what algorithms are most appropriate.

5.3 Contributions and Impact

This research project has presented a comprehensive empirical investigation to the performance of several oversampling techniques that are based on the SMOTE algorithm in the task of credit-card fraud classification. Seven different classifiers, together with seven different sampling techniques (amongst which most are explicitly designed to help deal with imbalances), were assessed and compared using the credit-card fraud dataset. The experiment considered 49 different strategies for the credit-card dataset by evaluating the performance of all the different combinations of classifiers and sampling techniques. Furthermore, all strategies were also examined under different conditions of varying degrees of skewness to the class-distribution of the data resulting in 441 different metrics for performance. These metrics were compared, and the more robust strategies were identified.

The information produced by this research adds to the vast body of work in the fields of Machine Learning, Class-Imbalance, and credit-card fraud. The conclusions obtained by this research reiterate the importance of techniques (such as SMOTE) that are specifically designed to help deal with highly imbalanced data. Particularly in the context of credit card fraud, where the data is very complex and highly imbalanced in nature, these techniques can be used to significantly improve the performance of classifiers in accurately distinguishing fraudulent and non-fraudulent transactions.

Fundamentally, the objective of research on fraud should be to help researchers not only identify fraudulent behavior but also to anticipate and predict it. Consequently, a significant progression for research on fraud would be to develop a framework that helps identify strategies that are most consistently robust so that these strategies can be implemented into fraud-detection-systems that operate in real-time with non-static streams of data. With this in mind, we hope that this research project will be used to bring insights and knowledge that can bring us closer towards this goal, ultimately, in the fight against fraud.

BIBLIOGRAPHY

Dal Pozzolo A., Boracchi G., Caelen O., Alippi C., Bontempi G., (2018). Credit Card fraud detection: A realistic modelling and novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, Volume 29 (8), 2018, pp.3784-3797.

Dal Pozzolo A., O Caelen, Le Borgne, YA., Waterschoot S., Bontempi G., (2014). Learned Lessons in credit card fraud detection from a practitioner perspective. *Expert systems with applications*, Volume 41 (10), 2014, pp.4915-4928.

Dal Pozzolo, A., Caelen, O., Johnson, RA., Bontempi, G., (2015). Calibrating probability with undersampling for unbalanced classification. *IEEE Symposium Series on Computational Intelligence*, 2015. pp.159-166.

Aisha A., Mohd A.M, Anazida Z, (2016). Fraud detection system: A survey. *Journal of Network and Computer Applications*, Volume 68, 2016, pp. 90-113, ISSN 1084-8045

Baader, Galina and Krcmar, Helmut, (2018). “Reducing false positives in fraud detection: Combining the red flag approach with process mining,”. *International Journal of Accounting Information Systems,* Elsevier, vol. 31(C), pp.1-16.

Barandela, R. Valdovinos R.M., J.S. Sanchez, F.J. Ferri, (2004). The imbalanced training sample problem: under or over sampling? in: Joint IAPR International. Workshops on Structural, Syntactic, and Statistical Pattern Recognition (SSPR/SPR’04), Lecture Notes in Computer Science, vol. 3138, 3138/2004, 2004, pp. 806–814.

Batista, G.E., Prati, R.C., & Monard, M.C. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations, 6*, 20-29.

Bentley, P., Kim, J., Jung., G. & Choi, J. (2000). Fuzzy Darwinian Detection of Credit Card Fraud. Proc. of 14th Annual Fall Symposium of the Korean Information Processing Society.

Bolton, R. & Hand, D. (2002). Statistical Fraud Detection: A Review (With Discussion). Statistical Science 17(3): 235-255.

Bolton, R. & Hand, D. (2001). Unsupervised Profiling Methods for Fraud Detection. Credit Scoring and Credit Control VII.

Brennan, P. (2012). A comprehensive survey of methods for overcoming the class imbalance problem in fraud detection. 10.13140/RG.2.1.3375.5284.

Breiman. L. (2001) Random forests, *Machine Learning* 45 (1). pp. 5–32. doi.org/10.1023/A:1010933404324

Breiman. L (1996) Bagging predictors*, Machine Learning* 26 (2). pp.123–140. doi.org/10.1007/BF00058655

Bonchi, F., Giannotti, F., Mainetto, G., Pedreschi, D. (1999). A Classification-based Methodology for Planning Auditing Strategies in Fraud Detection. *Proc. of SIGKDD99*, pp.175-184.

Bhattacharyya, S., Jha, S.K., Tharakunnel, K.K., and Westland, J.C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50, pp.602-613.

Bhargava, B., Zhong, Y., & Lu, Y. (2003). Fraud Formalisation and Detection. *Proc. of DaWaK*. 2003, 330-339.

C. Drummond, R.C. Holte, (2003). C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling, in: *Workshop on Learning from Imbalanced Data Sets II*, International Conference on Machine Learning, 2003.

C. Elkan, The foundations of cost-sensitive learning, in: *Proceedings of the Seventeenth International Conference on Machine Learning*, 2001, pp. 239–246.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over–sampling technique. *Journal of Artificial Intelligent Research*, 16, pp. 321– 357.

Chawla, N. V., Japkowicz, N., & Kolcz, A. (2004). Editorial: Special issue on class imbalances. *SIGKDD Explorations*, 6(1), pp. 1–6.

Chawla, N. V., Japkowicz, N., & Kołcz, A. (2004). Editorial: Special Issue on Learning from Imbalanced Data Sets. *SIGKDD Explorations*. 6. 1-6. 10.1145/1007730.1007733.

Chawla, N. V., Cieslak, D. A., Hall, L. O., & Joshi, A. (2008). Automatically countering imbalance and its empirical relationship to cost*. Data Mining and Knowledge Discovery,* 17(2), pp. 225–252.

Chawla, N. V., Lazarevic, A., Hall, L. O., & Bowyer, K. W. (2003). SMOTEBoost: Improving prediction of the minority class in boosting. *Proceedings of 7th European Conference on Principles and Practice of Knowledge Discovery in Databases* (PKDD'03), pp.107–119.

Chan, Philip and Stolfo, Salvatore, (1998) "Toward Scalable Learning with Non-Uniform Class and Cost Distributions. *Proc. Int'l Conf. Knowledge Discovery and Data Mining,* 1998. pp.164-168.

Chen, R., Chiu, M., Huang, Y. & Chen, L. (2004). Detecting Credit Card Fraud by Using Questionaire-Responded Transaction Model Based on Support Vector Machines*. Proc. of IDEAL.*2004, 800-806.

D. Randall Wilson and Tony R. Martinez. (1997). Improved heterogeneous distance functions*. Journal of Artificial Intelligence Res*. 6, 1 (January 1997) pp. 1-34.

Dong, Yanjie & Wang, Xuehua. (2011). A New Over-Sampling Approach: Random-SMOTE for Learning from Imbalanced Data Sets. 7091. 343-352. 10.1007/978-3-642-25975-3_30.

E.W.T. Ngai, Yong Hu, Y.H. Wong, Yijun Chen, and Xin Sun (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems* (v. 50, no.3), pp. 559-569.

Elkan, C. (2001). Magical Thinking in Data Mining: Lessons from CoIL Challenge 2000. *Proc. of SIGKDD01*, 426-431.

Elkan. C. (2001). The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973-978. Lawrence Erlbaum Associates Ltd, 2001.

Estabrooks, A., Jo, T., & Japkowicz, N. (2004). A Multiple Resampling Method for Learning from Imbalanced Data Sets. *Computational Intelligence, 20*, 18-36.

Efstathios K., Charalambos S. and Yannis M. (2007). Data Mining techniques for the detection of fraudulent financial statements. *Expert Systems with Applications*. 2007, pp. 995-1003.

Fernández, A., García, S., Herrera, F., & Chawla, N.V. (2018). SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. J. *Artificial Intelligence Res*., 61, pp. 863-905.

Galar, Mikel & Fernández, Alberto & Barrenechea, Edurne & Sola, Humberto & Herrera, Francisco. (2012). A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based *Approaches. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions*. 42. 463 - 484. 10.1109/TSMCC.2011.2161285.

Gary M Weiss. (2010). The impact of small disjuncts on classifier learning. In Stahlbock R., Crone S., Lessmann S. (eds) Data Mining. *Annals of Information Systems*, vol 8. Springer, Boston, MA, 2010, pp.193–226.

García, Vicente & Sánchez, Josep & Alberto Mollineda, Ramón. (2007). An Empirical Study of the Behavior of Classifiers on Imbalanced and Overlapped Data Sets. 397-406. 10.1007/978-3-540-76725-1_42.

Gong, C., & Gu, L. (2016). A novel SMOTE-based classification approach to online data imbalance problem. *Mathematical Problems in Engineering*, Article ID 5685970, 14.

H. He and E. A. Garcia, (2009). "Learning from Imbalanced Data," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263-1284, Sept. 2009. doi: 10.1109/TKDE.2008.239

Han, W.Y. Wang, B.H. Mao, (2005). Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: *International Conference on Intelligent Computing*

(ICIC'05), Lecture Notes in Computer Science, vol. 3644, Springer-Verlag, 2005, pp. 878–887.

Han, J., & Kamber, M. (2000). Data mining: concepts and techniques. *Morgan Kaufmann Publishers* Inc., San Francisco, CA, USA.

Haibo He, Yang Bai, E. A. Garcia and Shutao Li, (2008) "ADASYN: Adaptive synthetic sampling approach for imbalanced learning,". *IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligen*ce), Hong Kong, pp.1322-1328. doi: 10.1109/IJCNN.2008.4633969

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intell. Data Anal., 6*, 429-449.

Japkowicz, N. (2000). The Class Imbalance Problem: Significance and Strategies. *Proceedings of the 2000 International Conference on Artificial Intelligence ICAI.* Vol.1,. pp.111-117.

Japkowicz, N. Catherine Myers, Mark Gluck, et al. (1995). A novelty detection approach to classification. In *IJCAI*, volume 1, pages 518-523.

Jo, T., & Japkowicz, N. (2004). Class imbalances versus small disjuncts. *SIGKDD Explorations, 6*, 40-49.

Jeatrakul P., Wong K.W., Fung C.C. (2010) Classification of Imbalanced Data by Combining the Complementary Neural Network and SMOTE Algorithm. In: Wong K.W., Mendis B.S.U., Bouzerdoum A. (eds) *Neural Information Processing. Models and Applications. ICONIP 2010. Lecture Notes in Computer Science*, vol 6444. Springer, Berlin, Heidelberg

J. T. S. Quah and M. Sriganesh (2008), Real-Time Credit Card Fraud Detection Using Computational Intelligence*, Expert Systems with Applications*. Vol. 35, No. 4 pp. 1721-1732.

Kantardzic M., (2011). Data mining: concepts, models, methods, and algorithms. *Institute of Electrical and Electronics Engineers*, published by John Wiley and Sons Inc., New Jersey. 2nd Edition, pp. 447-449. ISBN:978-0-470-89045-5.

Kou, Y., Lu, C., Sirwongwattana, S., & Huang, Y. (2004). Survey of fraud detection techniques. *IEEE International Conference on Networking, Sensing and Control, 2004, 2*, 749-754 Vol.2.

Kubat, M., and Matwin, S. (1997). Addressing the curse of imbalanced data sets: One-sided sampling. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 179-186. Morgan Kauffmann.

López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Inf. Sci., 250*, 113-141.

Ling C., C.X. Ling, Li. (1998) Data Mining for Direct Marketing: Problems and Solutions (1998). In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* (KDD-98).

Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. (2009). Exploratory undersampling for classimbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics*), 39(2):539-550

Maes, S., Tuyls, K., Vanschoenwinkel, B. & Manderick, B. (2002). Credit Card Fraud Detection using Bayesian and Neural Networks. *Proc. of the 1st International NAISO Congress on Neuro Fuzzy Technologies.*

Major, J. & Riedinger, D. (2002). EFD: A Hybrid Knowledge/Statistical-based system for the Detection of Fraud. Journal of Risk and Insurance 69(3): 309-324.

Misha Denil and Thomas Trappenberg. (2010). Overlap versus imbalance. In *Proceedings of the 23rd Canadian conference on Advances in Artificial Intelligence* (AI'10), Atefeh Farzindar and Vlado Kešelj (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 220-231.

Quinlan, J.R, (1993). C4.5: programs for machine learning. *Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.*

Pazzani, M.; Merz, C.; Murphy, P.; All, K.; Hume T.; and Brunk, C. (1994). Reducing misclassification costs. In *Proceedings of the Eleventh International Conference on Machine Learning*, 217-225. Morgan Kaufmann.

Prati, R.C., Batista, G.E., & Silva, D.F. (2014). Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. *Knowledge and Information Systems, 45*, 247-270.

Prati, R.C., Batista, G.E., & Monard, M.C. (2004). Class Imbalances versus Class Overlapping: *An Analysis of a Learning System Behavior. MICAI.*

Provost, F., & Weiss, G.M. (2003). Learning When Training Data are Costly: The Effect of Class Distribution on Tree Induction. *J. Artif. Intell. Res., 19*, 315-354.

Suryanarayana, S and Gn, Bala ji and Venkateswara Rao, G. (2018). Machine Learning Approaches for Credit Card Fraud Detection. *International Journal of Engineering and Technology (UAE).* 10.14419/ijet. v7i2.9356.

Sorournejad, S., Zojaji, Z., Atani, R.E., and Monadjemi, A.H. (2016). A Survey of Credit Card Fraud Detection Techniques: *Data and Technique Oriented Perspective*. pp.405-411. abs/1611.06439.

Varsha S Babar and Roshani Ade. (2015). A Review on Imbalanced Learning Methods. *IJCA Proceedings on National Conference on Advances in Computing NCAC* 2015(2):23-27, December 2015.

Jarrod W. and Maumita B. (2016). Some Experimental Issues in Financial Fraud Mining. *Procedia Computer Science* (vol. 80) pp.1734-1744.

Zhu X., Wu X., (2004) Class noise vs attribute noise: a quantitative study of their impacts, *Artificial Intelligence Review*. Vol.22 (3–4). pp.177–210.

Zhi-Hua Zhou and Xu-Ying Liu. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):63.