CYPRUS INTERNATIONAL UNIVERSITY
INSTITUTE OF GRADUATE STUDIES AND RESEARCH
MANAGEMENT INFORMATION SYSTEMS DEPARTMENT

HCAB-SMOTE: A HYBRID CLUSTERED AFFINITIVE BORDERLINE SMOTE APPROACH FOR IMBALANCED DATA BINARY CLASSIFICATION

(Ph.D. Thesis)

Hisham AL MAJZOUB

Nicosia – 2020

CYPRUS INTERNATIONAL UNIVERSITY
INSTITUTE OF GRADUATE STUDIES AND RESEARCH
MANAGEMENT INFORMATION SYSTEMS DEPARTMENT

HCAB-SMOTE: A HYBRID CLUSTERED AFFINITIVE BORDERLINE SMOTE APPROACH FOR IMBALANCED DATA BINARY CLASSIFICATION

(Ph.D. Thesis)

Hisham AL MAJZOUB

Supervisor
Asst. Prof. Dr. Öykü AKAYDIN

Co-supervisors
Assoc. Prof. Dr. Islam ELGEDAWY
and
Asst. Prof. Dr. Mehtap KÖSE ULUKÖK

Nicosia – 2020

CYPRUS INTERNATIONAL UNIVERSITY
INSTITUTE OF GRADUATE STUDIES AND RESEARCH
MANAGEMENT INFORMATION SYSTEM DEPARTMENT

THESIS APPROVAL CERTIFICATE

This Thesis study of Management Information System Department Ph.D. student Hisham AL MAJZOUB student number 20153624 entitled "HCAB-SMOTE: A Hybrid Clustered Affinitive Borderline SMOTE Approach for Imbalanced Data Binary Classification" has been approved with the unanimity / majority of votes by the jury and has been accepted as a Ph.D. of Management Information System thesis.

Thesis Defense Date: 14 /8/2020

Jury Members	**Signature**
1) Asst. Prof. Dr. Öykü AKAYDIN Thesis Supervisor	……..…….....
2) Prof. Dr. Derviş Zihni DENİZ Member	……..…….....
3) Prof. Dr. Hasan DEMİREL Member	……..…….....
4) Assoc. Prof. Dr. Önsen TOYGAR Member	……..…….....
5) Asst. Prof. Dr. Kamil YURTKAN Member	……..…….....

Prof. Dr. Tahir ÇELİK

Director of the Institute

DECLARATION

Name and Surname: Hisham AL MAJZOUB

Title of the thesis : HCAB-SMOTE: A Hybrid Clustered Affinitive Borderline SMOTE Approach for Imbalanced Data Binary Classification

Supervisor : Asst. Prof. Dr. Öykü AKAYDIN

Co-supervisors : Assoc. Prof. Dr. Islam ELGEDAWY

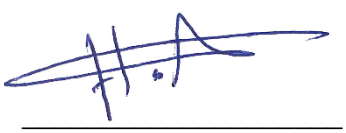
Asst. Prof. Dr. Mehtap KÖSE ULUKÖK

Year: 2020

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work

I hereby declare that the Cyprus International University, Institute of Graduate Studies and Research is allowed to store and make available electronically the present thesis.

Date: 02-Sep-2020

Signature: 

ACKNOWLEDGEMENT

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I want to thank my family which made it possible for me to study the Ph.D.in Cyprus international university. Special thanks to my supervisors Asst. Prof. Dr. Öykü Akaydın, Assoc. Prof. Dr. Islam Elgedawy and Asst. Prof. Dr. Mehtap Köse Ulukök who helped me with all the stages of the research by, stimulating suggestions and encouragement in all the time of research for and writing of this thesis.

My colleagues from the Institute of Graduate Studies and Research supported me during my research work. I want to thank them all for their help, support, interest, and valuable hints.

ABSTRACT

In this thesis, three algorithms are developed and implemented to optimize the performance of machine learning oversampling algorithms that increase the accuracy of the classification tasks over datasets having an imbalanced class problem. Machine learning uses historical data to reveal hidden patterns and improve the decision-making process in different business, medical, or other fields. However, it faces lots of obstacles and challenges, one of them is the dataset structure issue called imbalanced class dataset problem. In imbalanced class datasets, the distribution of the instances is imbalanced between the classes, leading the classification algorithm to act in a biased manner toward the class having the most instances and obtaining low classification accuracy for instances falling in the minority class. Most often, the goal of using machine learning is to get the patterns of the minority class instances so that the model can predict the class of the new unlabeled instances, but this process acquires low accuracy if the dataset has an imbalanced class problem. Different methods are available to reduce the effect of the imbalanced class problem on the generated models bias, but to increase the classification accuracy with those methods it has to deeply modify the original data through removing a high number of majority instances through undersampling methods or generating a huge number of new instances within the minority class through oversampling methods. The main focus of this thesis is to optimize the oversampling algorithm SMOTE to increase the classification accuracy of the intended class with minimal data altering. This thesis proposes the Affinitive Borderline – SMOTE (AB-SMOTE) that outperforms the classification accuracy of the former Borderline - SMOTE due to oversampling new instances within the borderline area instead of oversampling the instances around it. Then, the thesis develops Clustered Affinitive Borderline – SMOTE (CAB-SMOTE) which clusters the borderline area into different smaller clusters and oversamples within these clusters, delivering higher classification accuracy than AB-SMOTE in classifying the minority instances. Finally, the thesis proposes the Hybrid Clustered Affinitive Borderline - SMOTE which combines the undersampling method for removing noisy borderline instances from the majority and minority classes with oversampling CAB-SMOTE. Thus, obtaining the highest classification accuracy among other oversampling techniques. Therefore, these methods can be used to improve the accuracy of some machine learning applications to make them more reliable for the decision-making process that leads to decrease cost, and increase profit.

Keywords Imbalanced Data · Borderline SMOTE · Oversampling · SMOTE · AB-SMOTE ·CAB-SMOTE · HCAB-SMOTE· K-Means Clustering

TABLE OF CONTENTS

Contents

LIST OF TABLES

Tables **Page**

LIST OF FIGURES

Figures **Page**

ABBREVIATIONS

Acronyms	Definitions
SMOTE	Synthetic Minority Oversampling Technique
B-SMOTE	Borderline SMOTE
AB-SMOTE	Affinitive Borderline SMOTE
CAB-SMOTE	Clustered Affinitive Borderline SMOTE
HCAB-SMOTE	Hybird Clustered Affinitive Borderline SMOTE
IR	Imbalanced Ratio: $\frac{\text{Number of Majoriy instances}}{\text{Number of Minority instances}}$
Nn	Nearesst Neighboers
Sample; Pi	Minority instance: P1, P2…Pnum
Ni	Majority instance: N1,N2…Nnum
S%	Intput:oversampling percentage value to generate
S	New generated instance
Ns	Number of samples to generate
Danger; P'i	Borderline Instance: P'1,P'2…P'num
KNN	K nearest neighbors: used in the oversampling stage to generate new instances
Mnn	M Nearest Neighbors: used to compute the borderline instances
M'	Number of majority instances found within the Mnn from the minority instance at different itterations
Dif	Difference between two instances
k-means	Number of clusters which the data will be clustered to

CHAPTER 1
INTRODUCTION

The information revolution leads to new changes in the daily lives where it surrounds us with a huge number of data created from different sources, such as microphones, sensors, cameras, applications, and other inputs. The combination of these data is used for monitoring, tracking, and controlling purposes in various applications. It even reached a point where the machines, by creating models, can learn from these data how to predict or classify new entities without human intervention. This evolution which increased the capability of using or mining the data is called machine learning. It creates models that are implemented in different applications to forecast or classify unlabeled new entities. Forecasting or classification delivers higher profitability and cost minimization in application areas such as marketing campaigns focusing on a specific customer segment, fraud detection, or medical abnormalities identification. Machine learning can be separated into different categories; two of them are supervised and unsupervised learning techniques. Supervised machine learning uses datasets that have historical samples with known class labels, whereas unsupervised machine learning uses datasets that contain historical samples with unknown class labels. For supervised machine learning to work properly, it needs to use datasets with balanced classes to learn and generate models that deliver high classification accuracy for these dataset classes.

Some binary class datasets have an imbalanced class problem, which occurs when one of the two classes within the dataset has more instances than the other, dividing the dataset into majority and minority classes. The majority class, which is also called the negative class has the most instances of the dataset (more than 60% of instances within the dataset), whereas the minority class is called the positive class and it represents the class with least number of instances (less than 40% of the instances within the dataset). Classification models created from imbalanced class datasets are usually biased toward the largest class, where the model misclassifies most of the new unlabeled instances to be in the majority class. The imbalanced binary class problem is found in

different fields such as categorization (Sun, Lim and Liu, 2009), medical field (Tek, Dempster and Kale, 2010), customer churn prediction (Qureshi et al., 2013), and wine quality (Keel Datasets, Wine Quality, 2009). In general, the minority class is the intended class to be classified and it is important to accurately classify new instances that belong to it. There are a lot of solutions to rebalance the dataset and minimize the bias of the imbalance class issue, but these solutions have to add (oversampling) or remove (undersampling) a lot of instances to reach this goal.

Different approaches and methods have been proposed to minimize the negative impact on the performance of the imbalanced class problem such as (Sun, Lim and Liu, 2009), (Bekkar et al., 2013), (Chawla et al., 2002), (Bunkhumpornpat, Sinapiromsaran and Lursinsap, 2009), (Chawla et al., 2003), (Han, Wang and Mao, 2005), and (Bach et al., 2017). These methods are classified as undersampling and oversampling methods. The undersampling methods such as in (Sun, Lim and Liu, 2009) and (Bach et al., 2017), randomly eliminates instances within the majority class to balance the dataset, but doing so might remove essential instances from the majority class, thus decreasing the accuracy of classifying the majority instances against increasing the accuracy of classifying the minority instances. On the other hand, the oversampling methods proposed by (Chawla et al., 2002), (Bunkhumpornpat, Sinapiromsaran and Lursinsap, 2009), (Chawla et al., 2003), and (Han, Wang and Mao, 2005), create new instances within the minority class using systematic algorithms to balance the dataset, thus improving the accuracy of minority instances classification without jeopardizing the accuracy of majority instances classification. But they have some drawbacks like instance duplication or overfitting that they might not improve the classification accuracy more than the original dataset (Bekkar et al., 2013).

SMOTE (Chawla et al., 2002) is the most used oversampling technique to minimize the effect of imbalanced class datasets. It creates new scattered synthetic instances within the minority class without duplication. SMOTE needs to generate a large number of new instances (more than 100% of the minority instances) (Bach et al., 2017) to effectively increase the minority class classification accuracy. This is because SMOTE calculates distances between the widely scattered minority instances to create

new instances. While Borderline SMOTE (B-SMOTE) that is proposed by (Han, Wang and Mao, 2005) improved SMOTE to generates new instances based on minority borderline instances and other scattered minority instances to calculate the position and values of the new instances, resulting in a narrower oversampling area than SMOTE. B-SMOTE achieves better classification results with the same or fewer newly generated synthetic instances than SMOTE. However, the number of newly generated instances of B-SMOTE is slightly less than that of SMOTE, because this number is calculated by multiplying the number of minority instances found in the borderline region (80% or more of the minority class instances) by the oversampling percentage, leading to 20% or less in the reduction of the number of generated instances. The newly generated instances by B-SMOTE are generated around the borderline area, not within the borderline itself, which has the same effect of SMOTE. Therefore, this thesis aims to increase the efficiency of the oversampling technique SMOTE to get higher classification accuracy while generating fewer new instances than the original method. It used binary datasets suffering from the imbalanced class problem to evaluate the different oversampling techniques. The thesis improved B-SMOTE to AB-SMOTE where the new instances are only generated within the borderline area which delivered higher classification accuracy than the former oversampling techniques. And then the thesis proposed Clustered AB-SMOTE which focused the generation of new instances into a smaller area within the borderline area and delivered higher classification accuracy. Finally, the thesis combined undersampling with CAB-SMOTE and proposed HCAB-SMOTE which removed majority borderline instances from the majority class, the noise of the minority borderline instances, and oversampled new instances within clusters of the minority borderline area, which lead to a higher classification accuracy in classifying minority instances. This thesis used the F-measure value (harmonic mean of recall and precision) in comparing the different algorithms, where it showed a value increase in each newly proposed method. For example, in the Sales Win-Loss dataset, the F-measure value when classifying the original dataset with decision tree was 0.695, while oversampling the dataset with SMOTE, the classifier obtains a value of 0.838, whereas when the dataset got oversampled with CAB-SMOTE the classifier F-measure value

occurred was 0.841, Finally, when applying HCAB-SMOTE, the classifier obtains 0.864 F-measure value with a fewer number of oversampled new instances.

1.1. Problem Statement

For the machine learning to learn from a dataset for classifying new samples or entries, the dataset should have balanced classes, where both classes should have a somehow equal number of samples inside them for a model to be created and deliver good classification accuracy for all the classes, especially for the smaller one. In case the data had imbalanced classes, machine learning will not be able to learn from it and create a model that can correctly classify new instances to belong to the classes with equal accuracy, but it will be biased toward the class that has more instances. This research uses binary datasets that have two classes with imbalanced class distribution problem, where the intended class to be classified have a small portion of the dataset, less than 40% from the other one (Chawla, Japkowicz and Drive, 2004)(Prati, Batista and Monard, 2004). Thus, dividing the dataset into majority and minority classes, the majority class which holds the more than 60% of the instances within the dataset and its instances are referred to as negative instances, whereas the minority class that holds less than 40% of the instances within the dataset and its instances are called positive instances. The imbalanced class dataset problem leads to a bias in the classification model generated toward the majority class, where any new unclassified instance could be classified as a negative instance belonging to the majority class. Some datasets have imbalanced class problems where they have limited number of observations in one class and huge number of instances within the other, thus limiting the functionality of machine learning model building process and generates a model that deliver low classification accuracy in classifying new instances into the smaller class. This issue makes the model unreliable to obtain good output with high accuracy for assisting in the decision-making process. Even with available solutions to this problem, these solutions do not work in a sufficient way and they have to oversample and create big

number of new instances to get a higher and acceptable classification accuracy. That is why the thesis optimizes the oversampling methods.

1.2. Aims and Objectives

The thesis focuses on optimizing oversampling techniques that improve the learning process in machine learning. when dealing with data sets having an imbalanced class issue. This will be done by proposing new oversampling methods developed from modifications or in addition to the original oversampling methods. The thesis focuses on this region to increase the reliability of the machine learning process when using imbalanced class datasets. Because the main goal of machine learning is to ‘get it right’, through developing algorithms that predict and classify unlabeled new samples with a high level of accuracy to assist in the decision-making process with high reliability and least cost. But when using imbalanced class datasets, the classification accuracy for predicting such samples is usually low, which makes the machine learning process unreliable to be used for decision making. Even with different solutions already proposed and used to increase the prediction accuracy, but they are imperfect where they have to generate a lot of new instances to increase the accuracy, which motivated the thesis to develop SMOTE to make it function more efficiently when dealing with the imbalanced class problem.

1.3. Significance of The Study

Machine learning is a method of data analysis that automates analytical model building, it consists of systems that have many algorithms built inside to learn from data, identify patterns, and assist in decisions. With the machine learning process, it is possible to quickly and automatically produce models that can analyze bigger, more complex data and deliver faster, and more accurate results, even on a very large scale. Building precise reliable models to classify or predict new entities deliver a better

chance for the organizations to identify profitable opportunities, and avoid unknown risks. Data-driven decisions increasingly make the difference between keeping up with the competition or falling further behind. Machine learning assists in unlocking the value of corporate and customer data and enacting decisions that keep a company ahead of the competition. But machine learning cannot deliver accurate and reliable classification models if it was using the imbalanced class dataset, which is the main problem where most of the time the main class to be predicted, is the minority class which has few numbers of instances to learn from, and it generates a model with low and unreliable prediction accuracy. These kinds of datasets are found in many areas that deal with categorization or classification applications (Sun, Lim and Liu, 2009). They are found in the medical field (Tek, Dempster and Kale, 2010), customer churn prediction (Qureshi *et al.*, 2013), wine quality assessment (*Keel Datasets*, *Wine Quality*, 2009) credit card fraud prediction and other classification problems. An example of such datasets, a one that has information about diagnosing the existence of cancer cells, where there are few numbers of patients that have cancerous cells to form a good model to classify them. If machine learning was able to create a reliable model from this data it could classify which person has cancerous cells with fewer tests, therefore, minimizing expenses. Another example is about a dataset that solves credit card fraud problems where the banks have a very limited amount of historically recorded data of actual fraud cases for the machine learning to learn from them to generate a good classification algorithm to predict new potential frauds which would save a lot of money if classified directly. To solve this weakness lot of solutions were proposed, such as oversampling techniques, but they had to generate and add a large number of new samples within the minority class in the training data to increase the accuracy of the model. Therefore, this research aims to optimize and increase the efficiency of the oversampling techniques used to increase the classification accuracy in predicting instances falling within the minority class with fewer newly generated instances than the former ones. Thus, delivering better and more reliable knowledge to assist in the decision-making process that will decrease the cost of a given application, such as classifying the new person as a healthy or sick patient without conducting the costly exhausting tests or to other real-world problems. In a way it is the same as teaching a small child about the world, and as the children grow older, he

gets more information about the world around him, which will allow him can work and be productive.

1.4. Contributions of This Thesis

The exponential growth of digital data has significantly increased the need to access the right data to organize and categorize it into useable information. Therefore, the classification process where the algorithm generates a model that assigns unlabeled instances to predefined categories based on their content is very important in machine learning. But the machine learning process has a data weakness when using a dataset with an imbalanced class problem, due to the lack of instances within the minority class. The imbalanced class problem in datasets leads to classification model bias toward classifying majority instances and deliver low accuracy in classifying instances that belong to the minority class. Even there are a lot of solutions for this problem, they do not increase the classification accuracy unless they oversample and add a lot of new instances, such as 5 more times the number of instances found in the minority class. This thesis addresses the oversampling solutions for class imbalance problems, focus, and optimizes SMOTE in such a way that it will give better classification accuracy while generating fewer new samples than the original SMOTE or its derivatives. The thesis develops and enhances B-SMOTE to AB-SMOTE, and then intelligently combines it with K-means SMOTE to create CAB-SMOTE. After that, the thesis combines an undersampling process with CAB-SMOTE SII and proposes HCAB-SMOTE to increase the classification accuracy of the minority class with fewer oversampled instances. This contribution has other effects that will be reflected in the business and marketing in a wide scope. Because the thesis is optimizing an oversampling technique to solve imbalanced class dataset problem and increase the classification accuracy of the minority class within these data, this effect leads the management to use the machine learning more within its tasks because now they can rely more on the generated model and put it in use in finding hidden patterns, classifying issue or problems automatically without any human intervention. As it is

noticeable when organizations integrate machine learning into its process, its productivity will be increased with less time and effort than before.

1.5. Organization of This Thesis

The rest of this thesis is organized as follows: Chapter 2 presents the background of machine learning, the application that machine learning is used in, an overview of the methods that treat the imbalanced class dataset problem. Chapter 3 explains the existing related work within oversampling methods such as SMOTE and B-SMOTE, and discuss undersampling methods in general. Chapter 4 contains the methods which are proposed by this thesis starting from AB-SMOTE that oversamples new instances within the borderline area, CAB-SMOTE which clusters the borderline region and oversamples within the clusters independently, to finally presents HCAB-SMOTE which combines undersampling and oversampling techniques to reach higher classification accuracy with minimal generated instances. Chapter 5 presents the methodology that states which datasets are used in the thesis, how the thesis applies the oversampling approaches against the former original ones, and which metrics to use for evaluating the methods. Chapter 6 displays the results which are attained after applying the classification algorithm over the datasets in their original state and after applying the oversampling techniques, and display comparison between the proposed methods and other works doing in this field such as Safe-SMOTE, Density-Based SMOTE, and Local neighborhood SMOTE. Finally, chapter 7 contains the conclusion of this thesis and its contribution to the decision-making process.

CHAPTER 2
BACKGROUND

Stored data is considered raw, have no context, made up from numbers such as quantitative type, and text belonging to the qualitative type. It is created from facts, observations, statistics, and other sources of inputs which makes it independent and can stand by itself, unlike the information which only exists because it relies on data points. Data consist of features that help in creating outcomes or information which is considered as a group of data with context, or can be formed from processed data, or as an outcome from summarized, organized and analyzed data. Both data and information can be structured or unstructured and can be presented in tables, graphs, polygons, and other forms of presentations, but do not have any significance unless they are analyzed to meet the user's needs. In other words, information is formed up from analyzed and processed data to become more meaningful and useful. Information can be presented as a visualization of data, reports, or dashboards, to fulfill user requirements and deliver logical meaning. Organizations use Information Systems to collect and process data using collaboration between technologies, procedures, and tools to gather and assemble the data in a structured way to help in creating the information needed to create a certain knowledge. In other words, data is formed from raw material, when they are combined, they form building blocks which are referred to as information, therefore combining the building blocks (Information) results in creating a building which is referred to as knowledge as shown in Figure 2.1. These are the main cornerstones for having intelligent decisions for businesses or organizations.

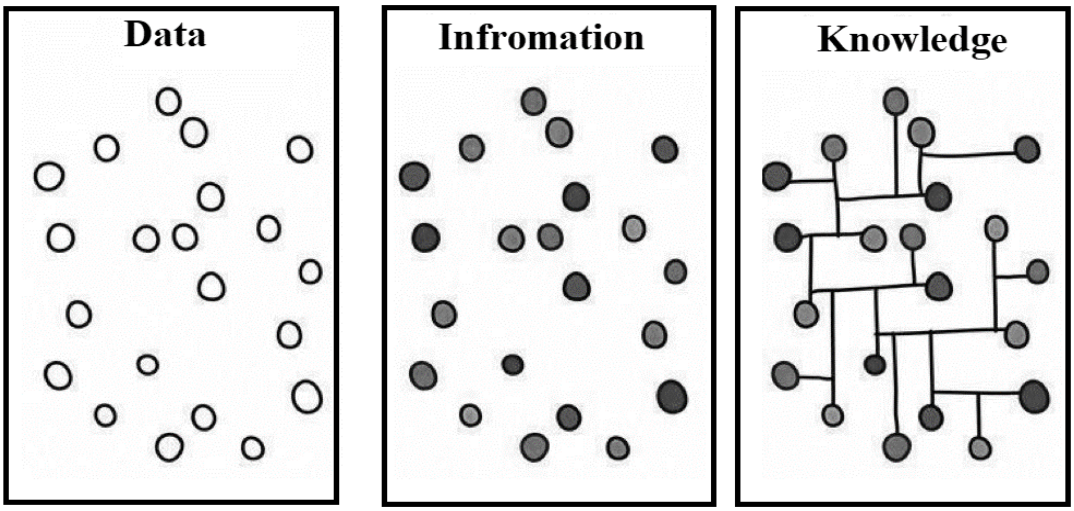


Figure 2.1: Difference Between Data, Information, and Knowledge

Machine learning is divided into supervised and unsupervised learning depending on the application and dataset type and properties as displayed in Table 2.1. Supervised learning, is the leaning method that combines and uses systems and algorithms to generate a model from the set of instances with the known class label to predict the class label of new classless instances having the same properties of the former instances. This model performance and accuracy can be tested by predicting the class label of the instances that have a known class and comparing them (predicted labels) with the actual values, where different measurements can be obtained to measure model performance. Supervised learning is somehow a straightforward simple method that delivers high prediction or classification accuracy because of the known class values. It uses algorithms such as Support vector machine, neural network, linear and logistics regression, random forest, and Classification trees and others. Whereas the unsupervised learning is a machine learning method that use systems and algorithms over a set of instances with an unknown class label to discover their class label or other information. Its processes are more complex than that of supervised learning. Unsupervised learning use cluster algorithms, dimensionality reduction, association, and other complex algorithms delivering lower accuracy than supervised learning due to the unidentified class values. The data can be formed from two kinds of numerical values, discrete or continuous. Discrete data are the data that can be counted and limited to certain values, where they cannot be divided into smaller values, whereas

continuous data are uncounted and have an infinite number of values within a certain range, where it is possible to have tiny differences between their values. Specific machine learning tasks can be applied only over specific data types. Classification, categorization, clustering, or others can only be applied over discrete data types, however, regression, dimensionality reduction, or other forms of machine learning, can only be applied over continuous data types. Continuous data can be converted into discrete ones by dividing them into ranges of values and assigning discrete values to all the continuous variables that fall within a certain range. In other words, different weights between 40.6 and 41.5 kg are converted to 41 kg discrete value, because dealing with discrete values is easier than dealing with continuous ones, and each algorithm has a certain way of functioning over the data.

Table 2.1: Machine Learning Types

	Supervised Learning	**Unsupervised Learning**
Discrete	Classification; Categorization	Clustering
Continuous	Regression	Dimensionality Reduction

Supervised machine learning is defined by searching for hidden patterns within large datasets and generate models to label new samples into a specific class. Those datasets have a large number of samples, more than 100 samples, and each sample has its features values and its defined known class. Machine learning aims to generate a concise algorithm that reflects the distribution of class labels in terms of sample features. This algorithm is then used to assign class labels to new instances or samples that have defined values for its features, but undefined class label (Konieczny and Idczak, 2016). The state of the art machine learning techniques read the different feature values and their corresponding class label from historical data, find the relation

between them within all the samples to get a model, or function which can suites the largest number of instances, and use this model to label class value to new unlabeled samples. In other words, it creates a model that generalizes the available instances and classify new unlabeled instances. These applications are used to optimize and solve different problems in a fast and reliable manner. Data mining is one of the most used applications of machine learning, where every sample within the datasets have the same set of features. These features can be from any nature, it might be numerical, or nominal. In supervised learning, the instances will be labeled, meaning that they are already categorized within a certain class, unlike the unsupervised learning where their class is unknown. Machine learning has to follow some steps to make its process, in the beginning, the dataset had to be collected, and a data expert might be needed to choose which features have to be included in each sample. In case the data expert is not available the dataset can be created with a brute-force method that all the data and information is gathered for every sample, but using the brute force method is not an efficient method of creating a dataset because it will take a lot of time, memory, and there will be a large number of unused noisy features, missing or impossible values due to input errors, which will lead to increasing the time of pre-processing the dataset.

Dataset might have different problems which are visible in the preparation and preprocessing stages. Some of these problems are that the datasets might have samples with impossible features values, such as negative value for cost or price of the thing being measured. This issue can be corrected in considering them as missing values and refill them using missing values imputation techniques or deleting the instances having such values. Other issues might arise such as inputting illegal values, where the feature values are only two, such as male, and female, but the input was something else, even it might have miss inputted values outside the scope of the feature values, such as outside the logical minimum and maximum values, where ratios should be between 0 and 1, a miss inputted value might be 1.5 or 2. Even missing values and misspelling can be critical issues within the dataset, where it should undergo a preprocessing stage to clean and organize the samples and their values. Datasets that have incomplete values within its samples is a major problem in machine learning. Different factors have to be looked after when checking the datasets, such as checking the source of the

“unknown-ness”, whether the missing value was forgotten or lost, the property of an instance do not accept a certain feature, or the expert user added a feature with minimum or no importance and marked it as “don’t care value”. Deferent methods were proposed to solve these issues, such as ignoring any sample that has one missing value within its features, filling up the missing value with the most common feature value found in other samples within the dataset, filling up the missing value with the most common feature value found in other samples within the same class of that sample having the missing value, filling the missing value with the mean features values throughout all the classes or within the same class, use the missing value as a class and create a model that classifies these value and fill it up with the new classified one (this method is time and memory consuming), hot-deck imputing where the software check similar sample to the sample that has the missing value and copy the feature value from that value to the missing one, imputing the word Unknown into the feature value and treating it as a new value for the following feature, or other methods for dealing with missing values. Another method for optimizing the dataset is called feature selection that identifies and removes irrelevant or meaningless features, thus reducing the number of dimensions within the samples that optimize the machine learning process by making it function more efficiently. It categorizes the features as relevant when it has an influence or effect over the output class, and as irrelevant when the feature could be randomly generated without affecting the output class, and as a redundant feature if it influences another class value. Feature selection consists of two algorithms, a selection algorithm that generates and find optimal features to be used, and an evaluation algorithm that determines how “good” a proposed feature subset is. Most of the features are interdepending on each other and that affects the accuracy of the classification modeling, but this can be resolved by creating new features from the basic ones by feature construction transformation. Where they will make the dataset more comprehensible, better understood, and will increase the classifier accuracy as well. Other forms of dataset optimization are the instance selection where the preprocessing algorithm removes noisy instances, therefore allowing the classification algorithm to function more efficiently. This can be done using undersampling techniques where it removes samples from the dataset. The process can be applied to

remove instances randomly or in a systematic focused way to balance the dataset in case it had an imbalanced class problem (Kotsiantis, Zaharakis and Pintelas, 2006).

To achieve goals efficiently and competitively for optimizing the performance, decisions have to be made by businesses, with the assistant of decision support systems and strategic information systems planning. Creating new services and products, with sustaining or developing them are very important objectives for the businesses' progress, and it is challenging to align these new objectives to the customers' needs in the ever-changing environment. These issues are impossible to monitor and follow when having large businesses, therefore Information technology is found to assist them effectively and instantly for increasing the level of certainty while making decisions about products or services. Information technology can work on many levels in a firm's hierarchy, where it aids the tasks for the lowest to the highest levels. At the managerial level, it aids in assisting the managers while planning, decision making, and problem-solving stages, which are some of the major phases and concepts within the strategic management tasks where they have a long-term effect on the business direction, mission, and its competitiveness among its rivals. Using the information by itself cannot assist the managers in boosting the business, it must always be maintained and aligned between the business strategies and the information systems nods. Reduction of cost, increasing the income, and improving the efficiency of the departments aligned with supply chain and service improvements are very important goals that businesses strive to achieve. All of these goals should be affiliated and integrated with the IT department which constantly updates and follow the tasks to support the objectives and attain higher competitive advantages within the market. The Information application should have known objectives, which might be problems, quarries or other issues, then the systems have to gather data and transfer it to information to make the readings or feature more understandable, clear and comprehendible, after this step, the user checks for possible and suggested solutions and outcomes for the issue on hand, and make their decision. That is how the system supports the business by showing them a lot of information in one dashboard to have an overall look about what is happening around and be able to give a good decision based on that information. (Kitsios and Kamariotou, 2016a).

To be at the top between the competitors, the business has to own successful services that show a good impression that the business is professional and successful. Services having an innovation factor that optimizes the performance delivers a strong positive effect over the business position. Innovation refers to an added value to a product, service, or idea, which in other words called creative development of an idea or product. This innovation rate can be increased and developed with the aid of information technology, where it requires radical skills and good alignment with goals needed to be covered. Another factor that supports the business to be competitive, is having an ongoing strategic development for the services that it owns, which have to be also aligned with the business objectives. Information technology is considered as a very important factor as well because it affects the process throughout the firm, where it supports the services and improve them by offering high speed and accurate monitoring for what is happening all around the clock. Well defined business strategy, organizational hierarchy, information technology department, staff experience, skills, knowledge, and good customer services, are all required to have the innovation within the services. This shows that technology is a very important role to be found in the business to develop a creative role. Technology is referred to as the use of datasets and automating business tasks and processes. The rate of business innovation increases when the rate of using technology within that business increases (Kitsios and Kamariotou, 2016b).

To protect endangered species, it is very curtailed for the park rangers to know the number of animals found in large wildlife reserves. Counting the animal in the old manual methods are very expensive, time-consuming and dangerous, regardless of the nature of the animals, if they are cattle or wild, a lot of new technologies had been arisen in this area and nowadays, unmanned aerial vehicles with high-quality consumer cameras such as drones are used to count the number of animals within a designated area. Machine learning serves a lot in detecting animals found in pictures taken by drones, and it is used in different scientific researches as well, using neural networks through deep learning algorithms that have a high level of accuracy in classifying special objects within large datasets. But a large number of wild animal counting, relay on small datasets which could not give high accuracy in detecting the animals.

(Kellenberger, Marcos and Tuia, 2018) proposed a method to manually detect animals using three times smaller data size. This allows the rangers to scan all the data in an accurate and fast way to classify nearly all the animals within the search scope automatically. Animal counting is very vital in monitoring the decreasing number of endangered mammal animals. Former methods of counting animals in remote areas were done manually by surveying using helicopters having more than one person to fly over designated areas or using camera traps on special areas and other manual methods which were very expensive and time-consuming. And these methods were the only way to get information about the number of exact animals within a chosen area. These methods had a lot of weaknesses, such as possible dangers over the human operators that have to be within the premises of the wild animals or armed poachers, very expensive and time-consuming, and very limited to the area that can be monitored within a controlled budget. For example, the price of camera traps decreased too much down to 30 $ per unit, but they still have to be installed and maintained in the wilderness inside the wild animals' regions, which is a risky operation. This issue could be addressed by using expensive manned aircraft with a special team which requires renting the airplanes, pilot, runways, and other expenses, which remove the risk factor but dramatically increase the costs. For these reasons, these methods limit animal counting in remote areas such as African savanna due to their high cost and risk. To overcome these issues unmanned aerials vehicles such as drones are used and controlled remotely to monitor these areas and count the chosen animals within a specific region. These drones are cheaper than manned aircraft and do not need special expensive pilot licenses to fly them. They are equipped with high-quality cameras and sensors which can be infrared, or any other options, minimizing the risk and financial issues for doing the needed tasks. Especially because of their smaller size and battery life raging between 30 min to 12 hours, they can reach inaccessible areas from safe distances. After scanning the area and taking the photos, they can be used through machine learning software to train the model over a data which was manually detected the animals, and then use that model to detect and count the animal in the newly taken photos. Then the software can count the animals automatically with a high level of accuracy. Researchers that use these techniques to count, and monitor animals such as (Omatu *et al.*, 2014), (Bouché, Lejeune and Vermeulen, 2012), (Andrew, Greatwood

and Burghardt, 2017) and (Ofli *et al.*, 2016). As it is noticed machine learning is widely used in different applications where it optimizes the organization activity by decreasing its costs and increasing its productivity, where here it increases the chosen area for counting the animals with less cost.

Preceding Human activity recognition systems were using balanced datasets to learn and generate a model from. But for predicting and classifying special cases in human activity recognition systems, such as falling over while walking the human activity recognition systems mostly use imbalanced class datasets. Imbalanced class data sets are datasets that have samples belonging to more than one class, and one of the classes have very few samples in relevance to other classes such as displayed in Figure 2.2. These type of imbalanced class datasets are widely found in different real-world problems. Human activity recognition researches were well spread within those past years. These days researchers are focusing on daily long-term tracking for human activities so that to aid and assist the medical field for classifying and identifying major diseases for delivering better health care for sick people who require treatment. These applications expand and develop the healthcare quality for old people as they can recognize certain diseases without making costly operation or testing. It works by deploying different types of sensors around the body that relay and send information of specific body parts to the system which will later classify or detect any abnormality with high accuracy. Researchers suggested diverse methods for detecting and recognizing human activities. They did not suggest one method for classification, because each method can deliver its results with different accuracy, and there is no uniform method that works overall data. It was expected that the datasets storing data for human activities to be balanced, but it is the other way around for certain activities where the datasets are highly imbalanced. Some activities happen more than others, such as sleeping is a daily activity and cover a long time through the day, whereas falling rarely occurs, and when it does it would be for a very short time. Some of the activity classifications applications can be used to monitor the physical activities for humans to be used in 3d animation, spying, physical therapy progress, athletics, or others (Wu *et al.*, 2016).

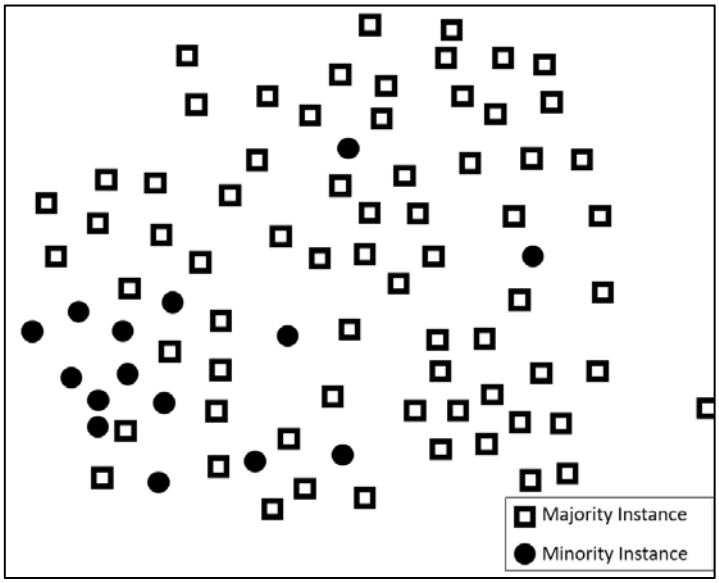


Figure 2.2: Example of The Imbalanced Dataset

Datasets have different types of weaknesses other than the imbalanced class issue, such as disjuncts, noise, lack of data, or other issues. The dataset is said to have small disjuncts when some of its classes are represented with small clusters of instances distributed in different areas. Especially in imbalanced class datasets, small disjuncts take place frequently because the minority class has few instances, and they may be distributed in different regions in the datasets as it is represented in Figure 2.3. In this case, it is hard for the classifier to uncover patterns or learn from these samples because of their random locations, therefore leading to the low classification accuracy of the intended class. Oversampling these disjuncts by generating new instances between them increase their density and enables the classifier to identify their pattern, thus create a model capable of predicting or classifying new unlabeled samples to which class it might belong to accurately.

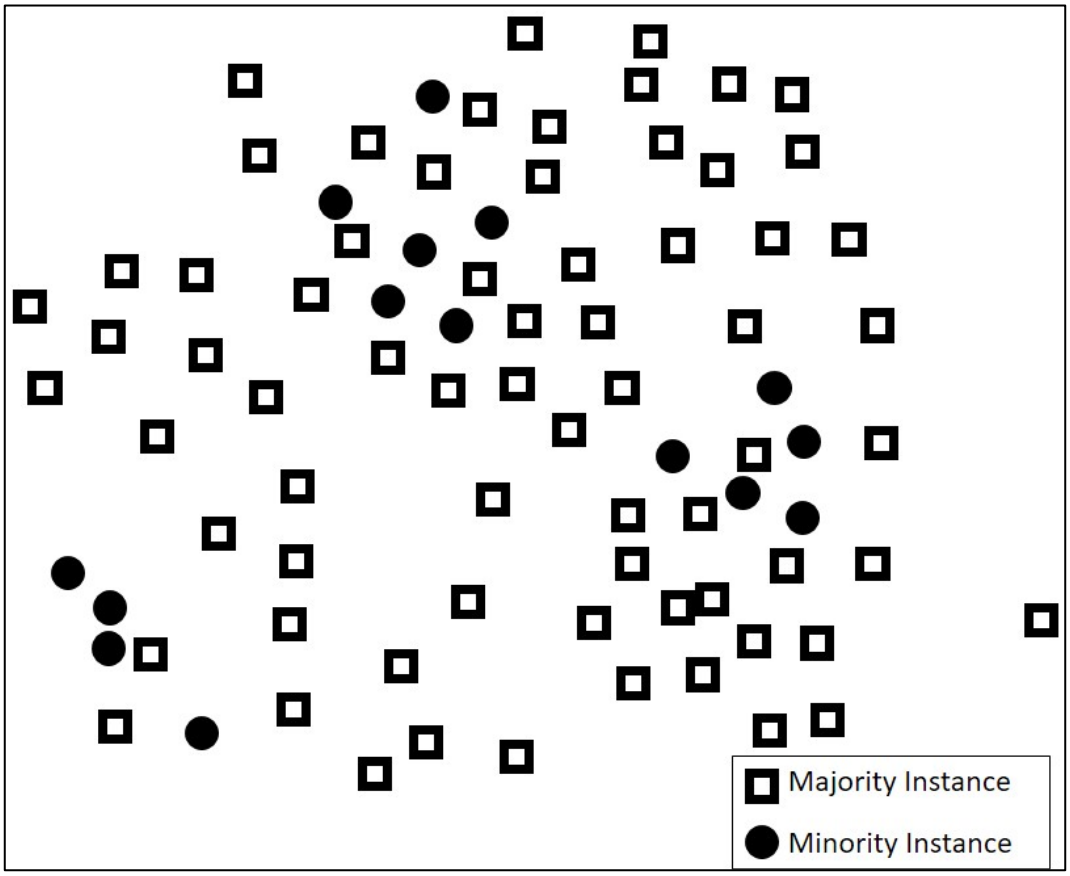


Figure 2.3: Example of small disjuncts on imbalanced data

Another topic that uses machine learning and prediction applications is the puzzling bankruptcy prediction where it combines economic and computer science. It is a very important task in banks, governments, lenders, investors, and others because they are working with a huge amount of money and they are in desperate need of reducing any risk of losing it due to bad decisions or investment. As any other prediction application bankruptcy need to have datasets to form the prediction model from, but these datasets mostly have an imbalanced class problem. They have this issue because the bankruptcy companies are way fewer than the normal functioning companies leading to bias within the classifier in building and predicting the companies that might be bankrupt. In the last years, lots of data being stored online and offline, these data can be pictures, audio files, spreadsheets, texts, and other forms of data. To make the most use out of these data, they have to be changed into useful information, to allow people to work in a smarter efficient way. Machine learning facilitated that by teaching the machine to

understand and create models that can classify and predict entities while doing very fast calculations and delivering a high level of classification, or prediction accuracy, some of these applications are, weather, and stock forecasting (Le *et al.*, 2018).

Worldwide, diabetes is considered a challenging health issue especially because it's spreading more among people every day throughout all countries. Diabetes is divided into three different types, Type 1, 2, and gestational diabetes. The most abundant one in the world is type 2 diabetes, but diabetes of whatever type increases the risk of having different health complications to the patient and might endanger his life if the patient was not conscious or had not properly treated himself. Blindness, kidney failure, heart diseases, and stokes can be caused by diabetes, even sometimes body part amputations have to be done to stop the spreading of extreme diabetes. For these reasons, it is categorized as the fourth leading disease that causes death worldwide. In 2015 the most country that had people suffering from diabetes was china, with 110 million diabetic people. Most of them were type 2 because of chubbiness and central obesity, no or lack of exercise, and harmful dietary habits such as eating fast junk food. Most of the people have diabetes of type 2. Around eighty percent of Type 2 diabetes patients can be cured or their complications postponed if they discovered that they are diabetic at the early stages. Diabetes is a long-lasting continuing disease, where the glucose sugar level in the blood becomes very high, caused by malfunctioning of the pancreas leading to low secretion of insulin, or malfunctioning of the cells where it cannot react to the incoming insulin. More than half of the people that have diabetes are unaware that they have it until having major health complications, if those people have known that they have diabetes at an early stage, a large number of them would avoid or delayed having major health complications, and lived healthier lives. The identification of diabetes criteria was unified in 1985 by the world health organization, where they used an oral glucose tolerance test known for its high sensitivity and specificity. But due to its inconvenience where people had to not drink or eat 8 to 12 hours before the test, drink glucose and wait two hours after that to test their blood. The world health organization relied on a simpler test that diagnoses if the person has diabetes, which is called fasting plasma glucose, but it can only detect 70% of cases of undiagnosed people. But even with that if the medical staff wants to manually

analyze people to check who might have diabetes it will be very time consuming and expensive, so for these reasons, an intelligent data analysis came in handy. Data mining and machine learning will assist by giving a tremendous amount of value in automatically classifying who would have diabetes with higher accuracy. Most of the datasets related to diabetes are consisting of the imbalanced class datasets where less than 10 percent of the samples inside these datasets are labeled diabetic, and the other majority samples are labeled as healthy ones. Thus, directly using these datasets without data preprocessing might give around ninety-five percent overall accuracy in predicting healthy people, whereas the people having diabetes are misclassified. For that oversampling should be applied over these datasets to somehow balance the classes and get higher detection accuracy for the diabetic patients (Wei *et al.*, 2017).

Pollution is spreading out all over the world, and the increase in the number of vehicles in cities is one of its causes. Therefore, it is very important to optimize transportation within the country, inside and outside the cities, therefore the researchers have to monitor and classify the vehicles already to optimize the maintenance of the transportation infrastructure, road safety management, and traffic flow. From the beginning of the 1980s, automatic vehicle classification systems were under research where researchers established that distortion of an invisible magnetic field can be used to detect moving vehicles in a given place. Where they wanted to classify the different types of vehicles and count them passing through chosen roads to optimize the transportation system, decrease private vehicles, and increase public vehicles. The intelligent transport system automatically classifies the type and speed of the vehicles. But to function properly and deliver high vehicle classification accuracy, it needed balanced datasets that were not found. Most of the datasets contained a high number of small cars and very few numbers of buses, which weaken the classifier while classifying the busses. Thus they needed methods to solve these problems, such as preprocessing the data through oversampling, undersampling, or other methods to balance the datasets (Xu *et al.*, 2018).

At the beginning of the 1990s as more data and applications of machine learning and data mining started to spread and to be used. The machine learning and data mining faced a lot of challenges in achieving high classification accuracy, especially when

using imbalanced data sets that were widely spread due to the lack of data and samples for the intended problem. It was at the beginning of the 2000s when the fundamentals of the issue “learning from imbalanced data” were established during the first workshop on class imbalanced learning during the American Association for Artificial Intelligence Conference (Provost, 2000). Imbalanced class data sets have common characteristics among themselves, such as the presence of small disjuncts when the data sets have small clusters containing few instances disbursed in the dataset which are unnoticeable when applying most classification algorithms as displayed before in Figure 2.3. Lack of training data density is another characteristic of an imbalanced class dataset where the algorithms do not have sufficient samples to generalize their distribution, which is found in high dimensional and imbalanced datasets. Noisy instances have a greater impact on the classification accuracy of the minority classes within the imbalanced class data sets than in usual cases. Subsequently, because the minority class has fewer samples, it will take fewer noisy instances to impact the performance of learning algorithms. And the dataset shift problem found in these datasets, which is defined as when training and test are divided in a way where the training data are selected from instances from majority class more than the instances of the minority class, in other words when the training and testing data sets have different instances distribution ratio than the whole dataset, therefore leading to sample selection bias issues (Ramyachitra and Manikandan, 2014).

Many techniques were proposed to improve the imbalanced class datasets problem, this thesis will mention some of them and present their limitation and weakness. One of the methods is PSO SMOTE by (Cervantes *et al.*, 2017) that apply SVM classification over the dataset and use the support vector points from different classes for the oversampling, where it chooses one support vector point from the minority class instance and connects it to one of its KNN from the support vector points from the majority class instances and then generate a new instance using particle swan optimization with values between 0.001 and 0.1 so that the new instance will be created in an area very near to the minority class region and then it check again for new support vector points, calculate the fitness and classification accuracy and then oversample again, this technique shows that it takes a lot of time because it runs SVM whenever a

new instance is generated. Another method uses KNN for undersampling which was proposed by (Zhang and Mani, 2003) where it removes borderline majority class instances from the data set where it badly affects the classification accuracy of the majority instances in favor of increasing the classification accuracy of classifying minority class instances, which is a very bad side effect because for machine learning to be successful it has to accurately classify instances to the minority class as well to the majority class. Other methods use Probabilistic Approach proposed by (Gupta, Gupta and Khobragade, 2016) where it applies KNN clustering over all the dataset and oversamples the clusters that have minority class instances in them, because these clusters have higher possibility of being misclassified, but doing so might increase noisy instances and would not necessarily improve the classification of the minority class as it should. As the thesis noticed from other methods, each method has its strength and weakness, and it is up to the user to choose which one to pick and use to rebalance the dataset intended to be classified. This is motivated the thesis work so that it will generate a new method with a lot of strength and least weaknesses points, which will later give the user an optimum balancing technique that will increase the classification accuracy of the imbalanced class data set with the least changing within the original data. Which is done by removing noisy borderline instances belonging to the majority class and small clusters within the minority class which disrupts the classifier, after that it oversample and create new instances inside clusters that are found in the borderline area which belong to minority class, increasing number of instances inside each cluster independently than the other clusters, therefore leading to a huge increase in the classification accuracy for the minority class without jeopardizing the majority classification accuracy.

CHAPTER 3
RELATED WORK

For optimizing datasets, preprocessing techniques should be applied to facilitate the classification algorithms' functions. These techniques are formed of data preparation, integration, cleansing, normalization, and transformation. There are additional tasks that can be added such as data reduction which includes a feature or instance selection, discretization, or other techniques. These methods are applied over the raw dataset to make it more organized, comprehensive, and useful for the algorithms to function on. They make the dataset more adaptable for the algorithms, minimizing the dataset size and decreasing its time complexity which lowers its computational costs, increasing the effectivity, efficiency, and accuracy of classifiers applied on these datasets by enabling the algorithms to handle large datasets more easily and precisely. These tasks improve different machine learning tasks such as classification, regression, and unsupervised learning. Different authors consider machine learning as knowledge discovery in database process and divide them into 6 different consecutive steps. The first step is specifying the problem where the application domain is arranged and aligned with experts' knowledge and the end-user objectives. Secondly understanding the problem by analyzing it to get the most understandable and usable data and the experts' knowledge aligned to achieve higher reliability. Third, the data undergo the preprocessing stage which includes data cleaning from noise and inconsistency, integration from different data sources, transforming to be merged into appropriate forms for the algorithms to work with (some algorithms only handle numeric data and cannot handle nominal data), and data reduction by removing some instances or features from the datasets. After the dataset had been cleaned and processed it undergo the data mining stage where the algorithm for machine learning is chosen to solve the task on hand (classification, regression, clustering or association) and implemented to extract the hidden patterns inside the data to create a model to accurately predict or classify new instances. After applying the data mining technique over the dataset, it undergoes the evaluation process using different metrics and measures to estimate and evaluate the accuracy of the model created in term of how accurate it can predict

known instances to being in the same class that is actually in so that it can use this model over other unknown instances after that. And finally, the last stage is using this evaluated model to predict or classify the class label for new unlabeled instances or samples.

This research focuses on prediction and classification techniques within the supervised learning, which attempt to discover the correlations between instances features. These correlations are called model, where it defines the unseen patterns within the datasets to be used for predicting the label class value of instances that have known features values. In other words, the supervised learning learns from the training set, create a generalization model to predict any unknown values. Most often the class values in supervised learning are finite, and categorical, where they have a specific number of known class values to classify new instances into. The new instances should have features values similar to the instances found in the training set to enable the classifier to predict their class values. Many real-world applications have imbalanced datasets, where samples belonging to the class of interest are fewer than that of the other classes, leading to a bias in the classifiers. Imbalanced datasets can be optimized to get higher classification accuracy to the class of interest by using specific methods more than the standard ones in the data preparation stage. In every machine learning task, there should be some data preparation applied over the data to convert it from raw to useable one to fit the mining process, otherwise, the algorithm might not function as it meant to and its outcome would not be accurate or meaningful. Some of the standard data preparation stages can be grouped as data cleaning, transformation, integration, normalization, missing value imputation, and others. The data cleaning stage is when the user interferes to correct or remove wrong inputted data, such as missing, illegal, or noisy values. Another stage of data preparation is called data transformation where the values within the data are converted into a specific form which allows the mining process to function more efficiently. It includes data smoothing, feature construction or selection, summarization of the values, normalization, discretization, generalization, and other tasks. Each task of the above is applied independently than the other and after one another, and these takes require human intervention especially when changing the type of the features such as changing the full dates within the data

sets to be the shorter ones like having only the number of the year within that feature, or categorizing a different range of numbers into specific categories. These transformations ease the work of the classifiers in building up the prediction models. Sometimes the dataset can be created from different sources, so it should be combined and gathered by data integration techniques to avoid duplications or inconsistencies. Data normalization is an important stage in data preparation where the values within the samples are transformed to be expressed in the same measurement unite or within a specific unified scale. Thus, equalizing the weight of the feature values, and this method is mostly used in a statistical type of data. As mentioned earlier lot of mistakes can be done while constructing the datasets, and one of them is having missing data, where it might occur by human error or other reasons, this problem can be addressed by data imputation techniques where they fill the missing values but intuitive, estimated data, or removing them. Another important form of data preparation is reducing the data which consists of various techniques that apply over the data set and transform it into another smaller one maintaining its essential structure and integrity, but with fewer samplers or features. Data reduction is an optional preparation step and not a mandatory one because some algorithms can be able to handle complex data but will need high computational costs. But some algorithms can work with the limited size of datasets, when it's larger than that limit, data reduction is a must to decrease the complexity and data size to improve the output model generated by the classifier. Different data reduction techniques displayed in Figure 3.1 are used to perform a specific task, such as Feature selection will be used to minimize data dimensionality by deleting features that have a minimum impact over the class value or irrelevant ones, to minimize the number of attributes found in the dataset with maintaining the original distribution to facilitate the pattern recognition and increase the model building stage. Whereas instance selection will remove conflictive or duplicate instances through undersampling techniques that can remove instances randomly or in a focused way to minimize the effect of overfitting and noisy data found within the classes that have the greatest number of instances. However, discretization will be used to simplify the features or domains by transforming the feature value from quantitative numerical value into qualitative nominal data or transforming continuous numbers into a set of ranges which will dramatically minimize the feature values from infinite

number into finite countable numbers or nominal value, making it much easier for the classifier to work on that data and generate higher quality models. Finally, feature extraction or instance generations will fill up gaps within the dataset by merging two or more features into one or to create a new artificial feature or adding new instance through oversampling techniques to balance the instance distribution within the classes of the dataset to optimize the work of the classifiers in supervised machine learning tasks.

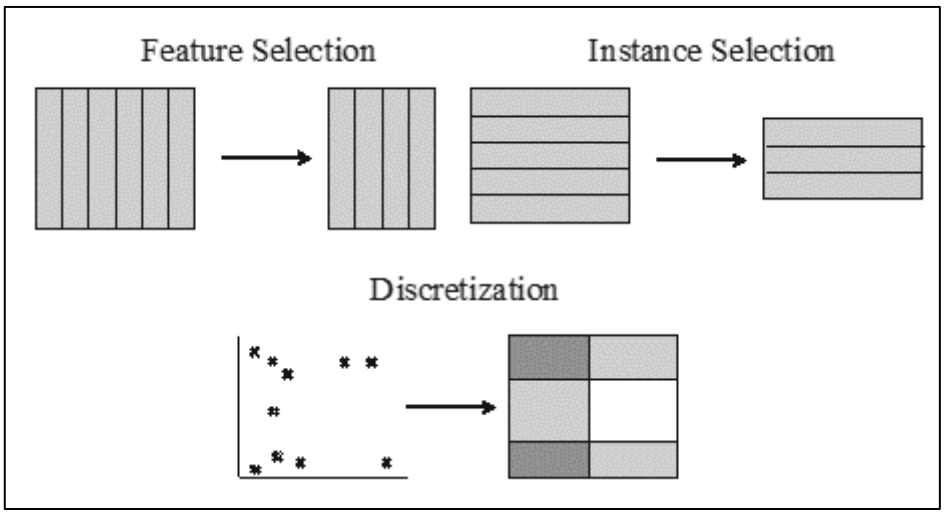


Figure 3.1: Forms of Data Reduction

Source:(García *et al.*, 2015)

For minimizing the bad effects of the imbalanced class dataset problem different approaches and methods have been proposed such as (Sun, Lim and Liu, 2009), (Bekkar *et al.*, 2013), (Chawla *et al.*, 2002), (Bunkhumpornpat, Sinapiromsaran and Lursinsap, 2009), (Chawla *et al.*, 2003), (Han, Wang and Mao, 2005), and (Bach *et al.*, 2017). These methods can be categorized as sampling, ensemble learning, cost-sensitive learning, feature selection, or algorithms modification methods. Feature selection can is used before any other preprocessing method, because it impacts the number of features found within the samples, where it checks which features have more influence over the class value, and disregards features that have a minimum or no influence over the class values. Removing one feature or more from the dataset

decrease the multidimensionality and allow other algorithms to function faster and delivering higher prediction accuracy. Because most of the real-world applications for solving a specific concept have huge datasets that when applying machine learning to them, might not work properly unless some of the features are removed (Dash and Liu, 1997). Because misclassification costs for minority class are very high, especially when having high dimensionality in features, feature selection is widely used to optimize the dataset to be a starting point to address this problem, by removing weak features that do not contribute in predicting the minority class. SYMON is one of the methods that use symmetrical uncertainty and harmony search to give different weights and select the important features depending on their influence over the class labels. Thus, marking the powerful ones that reflect the class labels with the least number of features. Feature selection can be divided into three sub approaches, filter, wrapper, and hybrid one. Filter approach will find a good but not the optimal subset for a classifier, wrapper, and embedded chose more focused feature subsets using feature ranking but does not always yield higher classification accuracy. Whereas combing the ranking feature with their dependency on a class and using them to select the best features may optimize the classification accuracy more. Feature selection is like any other algorithm, is always on progressing, and researchers are finding new methods of optimizing it so that the chosen features are selected to be more relevant to the class labels even when having high dimensionality. (Moayedikia *et al.*, 2017)

Using different classifiers to increase the classification accuracy is called an ensemble learning. It is widely used in different applications such as object, face or text detection and recognition, medical diagnosis, and others. Regular methods use one classifier to train and generate a model, while the ensemble uses more than one classifier to obtain higher classification accuracy. It trains using different base learners over the training dataset and classifies using each one of them, then syndicate them to give final classification, by doing so it optimizes these weak classifiers to give a better overall and minority class classification accuracy. Ensembles methods can be separated into two categorize, parallel and sequential. Parallel ensembles such as bagging, create base models in parallel, while sequential ensembles such as AdaBoost, generate first models and then create a next model-based over the first one and so on, where the model

generated influenced with their former models. For an ensemble to get higher prediction accuracy it should combine the most accurate and diverse classifiers as possible. Ensembles also optimized oversampling and undersampling methods to get higher classification accuracy when dealing with imbalanced datasets (Liu and Zhou, 2013). Most classification algorithms tend to minimize classification error rate, which is the ratio of the misclassified instances, ignoring the difference between the types of misclassification errors as they assume that all misclassification errors have the same cost. Whereas cost-sensitive algorithms consider the misclassification costs and minimize it to increaser the classification accuracy. Because in the real-world applications, there is a huge difference between the misclassification errors. Such as when diagnosing specific cancer, a patient having this cancer is labeled an instance in the positive class, where a healthy person is labeled as an instance belonging to the negative class, then if the classifier classified an actual patient (positive) as a healthy one (negative) this scenario is called false positive error, is more expensive than classifying a healthy person (negative) as a patient (positive) having a false-positive error, where in the first scenario the actual patient may die because of this false prediction that might prevent or delay applying the treatment over the patient. And in case of searching the airplane passengers for having explosive substances (as a positive class), it is more costly to miss someone that has any, than searching a person who does not have these kinds of substances or items. Cost-sensitive learning is an important type of machine learning method used in a lot of real-world applications. It has different methods to measure the costs, such as measuring misclassification, data acquisition (instances and attributes), active learning, computation, user-computer interaction, and other kinds of costs. The most important one of them where all the studied focused on, is the misclassification cost (Ling and Sheng, 2008).

This thesis focuses on the sampling methods that alter the class distribution ratios in the training dataset to reduce its imbalance class problem. These methods are applied over the training datasets before the classification algorithm stage, for that they are called preprocessing techniques, which can be categorized as undersampling or oversampling techniques that take place before model building stage in the data mining

process to adjust the data distribution, support the classifier algorithm to act in a balanced manner and increase the classification accuracy.

The undersampling methods such as in (Sun, Lim and Liu, 2009) and (Bach *et al.*, 2017), can be separated into random undersampling where it randomly eliminates instances within the majority class to balance the dataset and focused undersampling which removes specific instances that fall under a certain criterion (Tomek, 1976). Both methods increase the classification of the minority class, but they remove essential instances from the majority class which leads to a decrease in the accuracy of classifying an instance to be in the majority class against increasing the accuracy of classifying an instance in the minority class.

Whereas the oversampling methods such as (Chawla *et al.*, 2002), (Bunkhumpornpat, Sinapiromsaran and Lursinsap, 2009), (Chawla *et al.*, 2003), and (Han, Wang and Mao, 2005), create new instances within the minority class using systematic algorithms to increase the minority class classification accuracy. This improves the accuracy of minority instances classification without jeopardizing the accuracy of majority instances classification. But it also has some drawbacks such as duplication, or overfitting where they create new instances without improving the classification accuracy (Bekkar *et al.*, 2013).

SMOTE oversample within all the regions in the data set, which will slightly increase the classification accuracy for the minority class instances. That is the reason for having different derivatives for SMOTE, where there are more than 85 extensions have been proposed in the specialized literature which was gathered in the work of (Alberto Fernandez *et al.*, 2018). Their work mentioned Safe-level-SMOTE (Bunkhumpornpat, Sinapiromsaran and Lursinsap, 2009), SMOTEBOOST (Chawla *et al.*, 2003), Borderline SMOTE (Han, Wang and Mao, 2005), K-means SMOTE (Last, Douzas and Bacao, 2017) and others. Where they oversample instances in more focused regions with different tactics to increase the classification accuracy.

Safe-level SMOTE assigns a safe level for each minority instance before generating new instances. Where the newly generated instances will be placed closer to the largest

safe level using gap number to control its position, thus creating new instances only in safe regions. This safe level is referred to by the ratio between the numbers of minority instances within their neighborhood.

SMOTEBOOST combines SMOTE and boosting algorithms to address small disjuncts problem where they apply different weights to instances dynamically while the process is functioning. It increases the weights of misclassified instances to focus on them more. Due to the difficulty of predicting small clusters or disjuncts, it is logical that boosting improves classification performance.

Borderline SMOTE divided the minority class instances into noisy, safe, and danger instances depending on the neighboring instances around a chosen minority instance, where the method neglects the safe and noisy instances and uses the danger instance for the oversampling process. It only oversamples new instances using danger instance and their neighboring instances belonging to the minority class, in other words, it oversampling around danger area where the instances contribute to increaser the classification accuracy more than the original SMOTE more does.

AHC (Cohen *et al.*, 2006) was the first method that applies the clustering algorithm in the oversampling process. It uses the k-means clustering algorithm to remove samples from the majority class when undersampling from the dataset and uses agglomerative hierarchical clustering over all the minority class to oversample and generate new instances to increaser the minority classification accuracy.

DBSMOTE (Bunkhumpornpat, Sinapiromsaran and Lursinsap, 2012) uses density-based clustering called DBSCAN over the minority. DBSMOTE was inspired by Borderline-SMOTE in the sense it operates in an overlapping region, but unlike Borderline-SMOTE, it also tries to maintain both the minority and majority class accuracies. The thesis compared the proposed techniques CAB-SMOTE and HCAB-SMOTE with this oversampling method (DBSMOTE) where it showed that HCAB-SMOTE attained better results than DBSMOTE.

3.1. Synthetic Minority Oversampling Technique (SMOTE)

SMOTE (Chawla *et al.*, 2002) is the most used oversampling technique in treating imbalanced datasets. It systematically generates new synthetic instances within the minority region without creating duplicates. It has some shortcomings, such as how it identifies which minority examples to act as seeds for the oversampling process, where it chooses all the instances from the minority class, without knowing that these instances are not equally important for the learning classification. Instances that are found in the regions far away from the other class and surrounded only by minority instances do not optimize the classification model as the instances falling inside or around the borderline area between the two classes. This leads to oversample instances that do not affect the classification accuracy, and it is the reason of why SMOTE needs to generate a large number of new instances, sometimes more than 500% of the minority instances depending on the dataset instances distribution (Bach *et al.*, 2017) to effectively increase the minority class classification accuracy. Where it unseeingly generalizes the minority class area without considering the majority one, which makes it problematic in the case of a highly imbalanced class dataset (Bunkhumpornpat, Sinapiromsaran and Lursinsap, 2009).

SMOTE, has input parameters that the user can modify to tweak its process. These parameters are as follows: K which is the number of nearest neighbors around the selected instance, with a default value of 5, the oversampling percentage with a default value of 100%, where the algorithm generate 100% more of minority instances, hence doubling the number of minority instances, a random seed number with a default value of 1 to be used for experimental purposes to get the same oversampling instances in each run so for comparing the results, and a Class value parameter with a default value of zero, where the algorithm automatically check the smallest class and apply SMOTE on that class, the user can change it if they want to apply SMOTE on a different class. When applying SMOTE, it loads the dataset to the memory, and checks which class is the minority one by comparing the number of instances in each class, after that it holds one positive instance (seed) and calculates its KNN between the selected seed and other minority instances, later it randomly picks one of the KNN instances and calculates the distance between the seed instance and the randomly picked one. Later

the distance value will be multiplied by a random number (gap) that has a random value between 0 and 1, to mark a point between the chosen instances where it will fill up the feature values of the newly created instance. After generating the new instance, the algorithm fills the instance feature's values using Equation 3.1, where P_i is the initial seed instance from the minority class, and P_j is the randomly picked positive instance from the KNN around the initial instance. SMOTE repeat this process with other positive seed instance by incrementing the seed id number by one, till the oversampling percentage is fulfilled. Eight loops are displayed in Figure 3.2 where 8 new instances are created and marked in green. The new instances are possible to be created in any location in the data set without being limited to a special area. If a positive seed or KNN sample falls within the majority class, it will create a new positive sample in the majority class as well. The thing which will cause an increase of noise within the majority class area as displayed in Figure 3.3, or overfitting the safe minority region far from the borderline area if the seed and KNN were deep inside the minority class region.

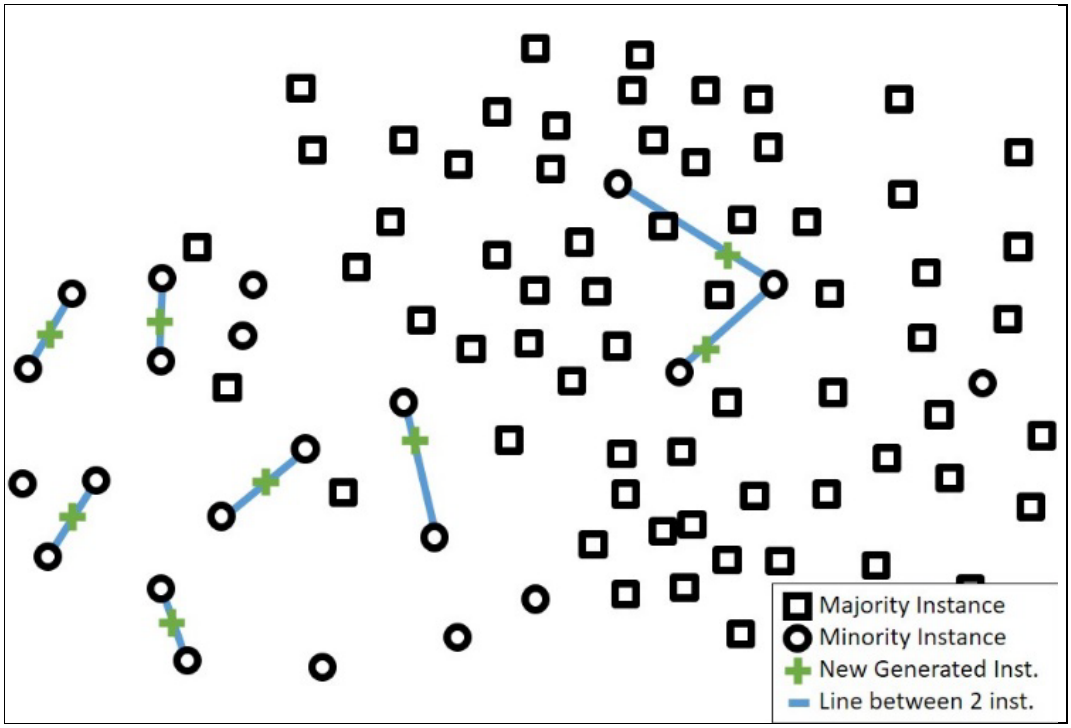


Figure 3.2: SMOTE Diagram

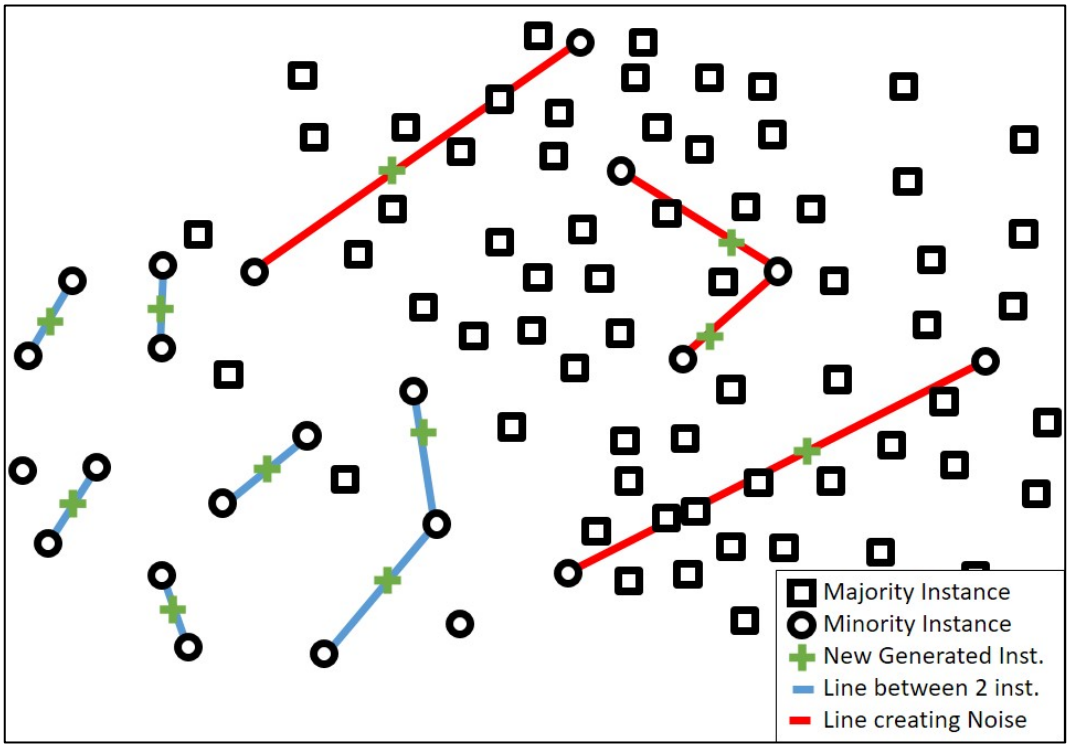


Figure 3.3: SMOTE Generating Noisy Instances

$$\text{New Instance} = \text{Pi} + \text{gap} * (\text{distance}(\text{Pi}, \text{Pj})) \quad (3.1)$$

The Synthetic Minority Oversampling Technique is considered an actual standard when dealing with an imbalanced class dataset, where it is first used before other oversampling techniques. Its design simplicity and good performance made it well spread into a wide range of machine learning software. SMOTE is a powerful technique that can reduce the imbalanced class problem over different types of datasets, where it successfully proved that in a large diversity of applications within different domains, whether they are commercial or open-source software. The wide spreading of SMOTE motivated a lot of researchers to create different oversampling approaches to address the imbalanced class problem, overall are data types. These data types ranged from binary class, multi-class, regression, incremental, supervised, semi-supervised, and other learning environments (Alberto Fernandez *et al.*, 2018).

Figure 3.4 displays SMOTE pseudo-code that explains how the algorithm works. For each process inside the SMOTE algorithm, the thesis creates a function to be used in other oversampling pseudo-codes to minimize the length and increase the comprehensiveness of the later codes, and these functions are gathered in Table 3.1.

Algorithm 1:SMOTE

Input: P number of minority class sample ; S% amount of synthetic to be generated; k Number of nearest neighbors

Output: Ns= (S/100)* P synthetic samples

1. ***Create function Compute KNN (i←1 to P, P_i,P_j)***
   { For i ← 1 to P
   Compute k nearest neighbors of each minority instance P_i
   and other minority instances P_j.
   Save the indices in the nnarray.
   Populate (Ns, i, nnarray) to generate new instance.
   End for }
2. Ns= (S/100)*P
   While Ns ≠ 0
3. ***Create function GenerateS (***P_i, P_j**)**
   {Choose a random number between 1 and k, call it nn.
   For attr ← 1 to numattrs
   dif=P_i [nnarray[nn]][attr] – P_j [i][attr]
   gap = random number between 0 and 1
   Synthetic [newindex][attr] = P_i[i][attr] + gap ∗ dif
   End for
   newindex= newindex +1}
   Ns = Ns – 1
   End while
4. Return (∗ End of Populate. ∗)

End of Pseudo-Code.

Figure 3.4: SMOTE Pseudo-Code

Table 3.1: List of Functions and Their Definitions

Function	Definitions
ComputKNN	(i←1 to P, P_i,P_j) Compute the KNN between P & P.
	(i←1 to P', P_i',P_j) Compute the KNN between P' & P.
	(i←1 to P', P_i',P_j') Compute the KNN between P' & P'.
GenerateS	(P,P) Create instance & fill its values between P&P
	(P_i',P_j) Create instance & fill its values between P'&P
	(P_i',P_j') Create instance & fill its values between P'&P'
MinDanger	Creates the minority border P' subset.

3.2. Borderline-SMOTE (B-SMOTE)

Because of SMOTE short comes, a lot of derivatives were proposed to optimize it to get better classification accuracy. B-SMOTE was proposed by (Han, Wang and Mao, 2005) which generates new instances based on positive borderline instances (seeds)

and other positive instances within the dataset to calculate the position and values of the new instances, which results in a narrower oversampling area than SMOTE and somehow more focused around the borderline region. These changes in oversampling made B-SMOTE achieve better classification results with the same or fewer newly generated synthetic instances than SMOTE. However, the number of newly generated instances of B-SMOTE is slightly fewer than that of SMOTE, because the algorithm calculates the number of new instances to be created by multiplying the number of minority instances found in the borderline region (which is around 80% or more of the minority class instances) by the oversampling percentage, leading to 20% or less reduction of the number of generated instances. And another reason is that the new instances are generated around the borderline area, and not within the borderline itself, which have the same effect of SMOTE.

B-SMOTE separates the minority instances into danger, safe, and noise instances. This categorization is done by counting the number of majority instances M' within the M nearest neighbors (Mnn) around the seed minority instance. It checks within the Mnn of P_i and its surrounding instances. P_i is considered to be safe if it is surrounded by minority class instances, which is computed by the number of M' to be between 0 and M/2 within the Mnn. P_i is considered as noise if it was surrounded M number of majority instances within the Mnn, that is when M'= M. Whereas P_i is considered as danger P' when more than half of its Mnn instances belong to the majority class (between M/2 and M) and it is then added to the new subset called danger. After the danger subset is created, the B-SMOTE measures KNN between the selected danger instance and other positive instances within all the dataset – including safe, noise, and danger instances-, compute the distance, generates the new instance and assign its feature values using Equation 3.2.

$$\text{New instance} = P'i + \text{gap} * (\text{distance}(P'i, Pj)) \quad (3.2)$$

Where P'_i is the seed borderline minority instance, P_j is the random positive instance chosen within KNN, and a gap is a random number having a value between 0 and 1. The danger instances are displayed as the blue-colored circles in the region between the two classes in Figure 3.5 The B-SMOTE algorithm is demonstrated in Figure 3.6.

(Han, Wang and Mao, 2005) also proposed another version of B-SMOTE and named it B-SMOTE 2 where the algorithm uses instances from the majority class in the KNN computation and gap values between 0 and 0.5 so that the newly created instances will be created closer to the borderline instance. B-SMOTE 1 and 2 oversamples around the borderline region, thus they both serve in spreading and widening the borderline region, by doing so they achieve better classification accuracy than the original SMOTE. This proves that focused oversampling and generating new instances around the borderline area is better than random oversampling within all the dataset regions for increasing the classification accuracy of imbalanced class datasets. B-SMOTE is one of the main derivates of SMOTE where other scholars adopted its idea and improved it for creating new oversampling methods that would deliver higher classification accuracy.

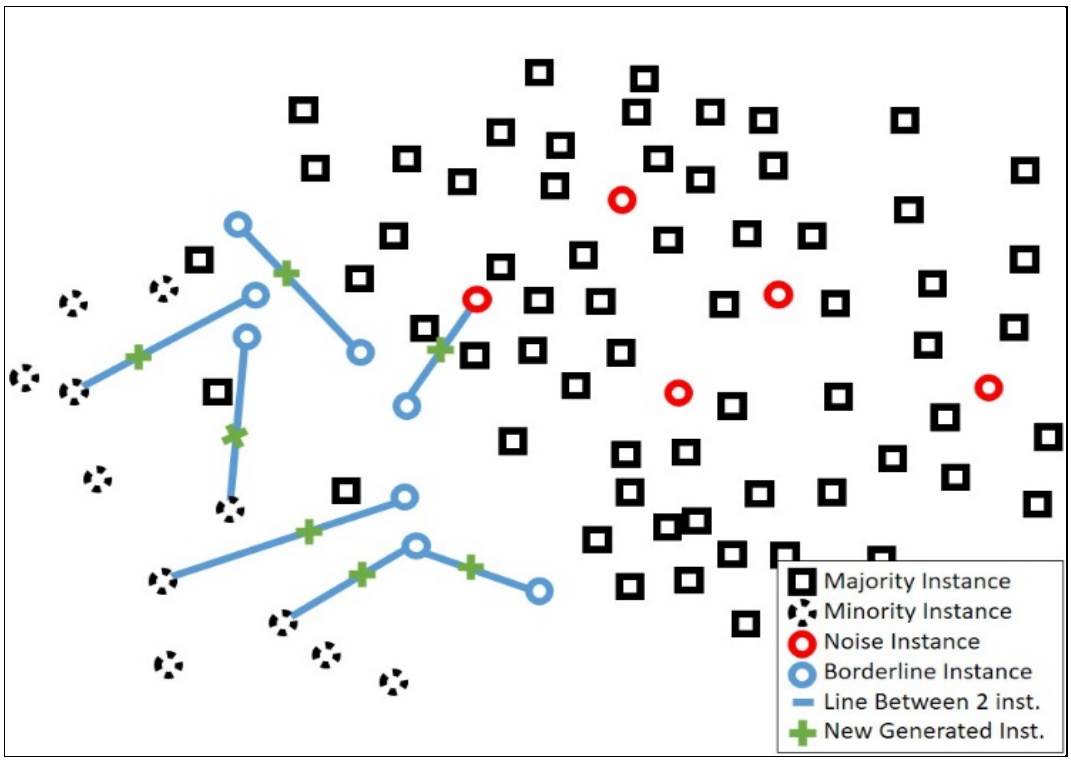


Figure 3.5: B-SMOTE Diagram

Algorithm 2:B-SMOTE

Input: P number of minority class sample; S% amount of synthetic to be generated; M number of nearest neighbors to create the borderline subset; k Number of nearest neighbors
Output: (S/100)* P' synthetic samples

1. ***Create function MinDanger ( )***
   { For i ← 1 to P
   Compute M nearest neighbors of each minority instance
   and other instances from the dataset.
   Check number of Majority instance M' within the Mnn
   IF M/2< M' < M
   Add instance P to borderline subset P'
   End if
   End for }
2. **Compute KNN (i←1 to P', P_i',P_j)**
3. Ns= (S/100)*P'
   While Ns ≠ 0
4. ***GenerateS (***P'_i, P_j**)**
   Ns = Ns – 1
   End while
5. Return (∗ End of Populate. ∗)

End of Pseudo-Code.

Figure 3.6: B-SMOTE Pseudo-Code

3.3. Undersampling Methods

Another solution for imbalanced class problems is by rebalancing the dataset with the undersampling process. Where instances from the majority class are removed from the dataset (Bach *et al.*, 2017)(T and M, 2016) and (Oskouei and Bigham, 2017). The undersampling methods can be divided into two types: Random Undersampling where it randomly removes majority instances without using any selective criteria; and Focused Undersampling that removes majority instances located on the borders between the two classes (Japkowicz, 2000).

These methods have their pros and cons. On one hand, they eliminate a large number of majority instances that reduce the size of the datasets and decreases the model building run time. On the other hand, the data will lose important instances that contribute to the learning process leading to a decrease in the majority instance classification accuracy. (Stefanowski and Wilk, 2008) Discusses different methods used in the undersampling process such as Tomek Link and Nearest neighbors (Laurikkala, 2001) to identify which majority instances to be eliminated from the

dataset. The Tomek link method is defined by comparing distances between instances. It chooses two instances X that belong to class A, and Y belongs to class B, and compute the distance between them and denoted by d (X, Y). The points X and Y are considered a Tomek Link if the distance from any instances Z, to any of points X or Y (d (Z, X) or d (Z, Y)) is larger than the distance between the two instances d (X, Y). If any of the two instances X or Y are identified as Tomek Link, then the instance that belongs to the majority class will be removed. Figure 3.7 illustrates the Tomek Link method, where it shows two adjacent instances from different classes. Whereas the Nearest neighbors cleaning method works in a different way, where they use Wilson's edited nearest-neighbor rule to identify noisy instances in the majority class. The Wilson rule function in a way that it chooses instances from the majority class and checks its three nearest neighbors, if two or more of the nearest neighboring instances around it (instance belonging to the majority class) belong to the minority class then this instance will be deleted and removed from the training data.

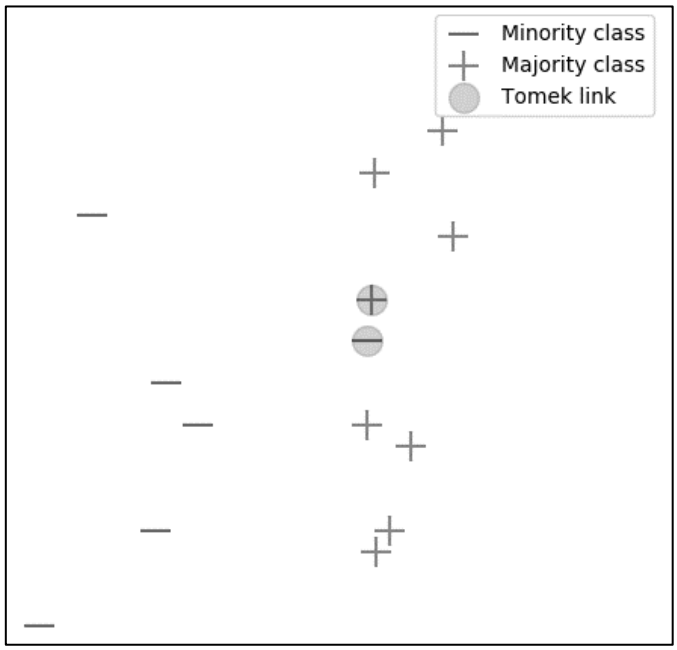


Figure 3.7: Illustration of Tomek Link

Borderline SMOTE uses the nearest neighbor's process to identify the border minority instances using 5 and 8 nearest neighbors. The thesis also used the nearest neighbor's process for removing border instances belonging to the majority class with a value of 5 and 8 nearest neighbors. The process of removing the majority borderline instance was very similar to the process of creating the danger subsite from the minority class, but instead, it was implemented within the majority class and when an instance was flagged to be at the borderline, it was deleted and removed from the dataset. The thesis chose this process for its simplicity and good outcome in differentiating between safe and danger instances within the majority class. The removal of noisy instances from the majority class also improves the classification accuracy of classifying the minority class instances which give the proposed method a higher classification accuracy as will be experimented and explained after in this thesis. This classification accuracy increase was accrued because noisy instances were removed from the majority class, which allowed the classifier to better understand and uncover the pattern of the minority class. Later on, in the experiment, it is possible to notice the advantages of the undersampling done when applying HCAB-SMOTE by comparing its classification results with the classification results when applying CAB-SMOTE SII, where both algorithms function in the same manner, but HCAB-SMOTE remove instances from the majority class and noisy minority border. It is noticeable that the undersampling increase the classification accuracy, that is why the thesis chose to combine it with oversampling in CAB-SMOTE, to make it easier for the user to apply one preprocessing algorithm to obtain the highest classification accuracy over an imbalanced class dataset.

CHAPTER 4
PROPOSED METHODS

This research extends B-SMOTE to AB-SMOTE by focusing on the creation of newly generated instances within the borderline area. Then the thesis develops AB-SMOTE by combining it with K-mean SMOTE to propose CAB-SMOTE where it induces the clustering process over the borderline area and oversampling new instances within each cluster independently, which focuses the oversampling process within those clusters. Later after implementing CAB-SMOTE, the research develops HCAB-SMOTE where it combines undersampling technique to remove noisy borderline instances from the majority and minority class and oversamples within clustered minority borderline area to get the best positive classification accuracy results. The thesis uses WEKA (*Weka*, 2020), which is an open-source JavaScript software for data mining that enables the user to read and modify its codes. The research investigates in its oversampling filter, SMOTE, and develops new derivatives from it. The source codes of the new techniques are uploaded to the GITHUB website under the URL github.com/Hishammajzoub/HCAB-SMOTE so that other users or researchers can use it or continue the work and develop it more.

4.1. Affinitive Borderline SMOTE (AB-SMOTE)

Because WEKA does not have a B-SMOTE filter. The thesis accesses and modifies the source code of the SMOTE filter to function as B-SMOTE and adds it to the oversampling filter window inside WEKA. While doing so, the thesis notices some gaps in B-SMOTE, one of them is that the oversampled instances are generated around the borderline area and not inside it, the thing that motivated the thesis to develop it to oversample new instances within the borderline area. The thesis names this new approach, Affinitive Borderline SMOTE (AB-SMOTE) which means closely connecting new instances within the borderline. This method creates the danger subset

in the same way as B-SMOTE creates it, and then it changes the function of KNN computation, where it locks it within the instances inside the border region, thus limiting the sample diversity used for generating new instances to the border region. AB-SMOTE is displayed in Figure 4.1 where it creates 6 new instances within the border and can be compared with Figure 3.6 that oversamples 7 instances around the border region to show the differences between B-SMOTE and AB-SMOTE. Figure 4.2 displays the AB-SMOTE pseudo-code that explain every step of how the algorithm works, where it checks the KNN between each danger instance as seed and other instances within the same danger subset, randomly choose one of the KNN instances, calculates the distance between the seed and selected instances and applies Equation 4.1 for computing the feature values of the new instance.

$$\text{New instance} = \text{P'i} + \text{gap} * (\text{distance}(\text{P'i}, \text{P'j})) \quad (4.1)$$

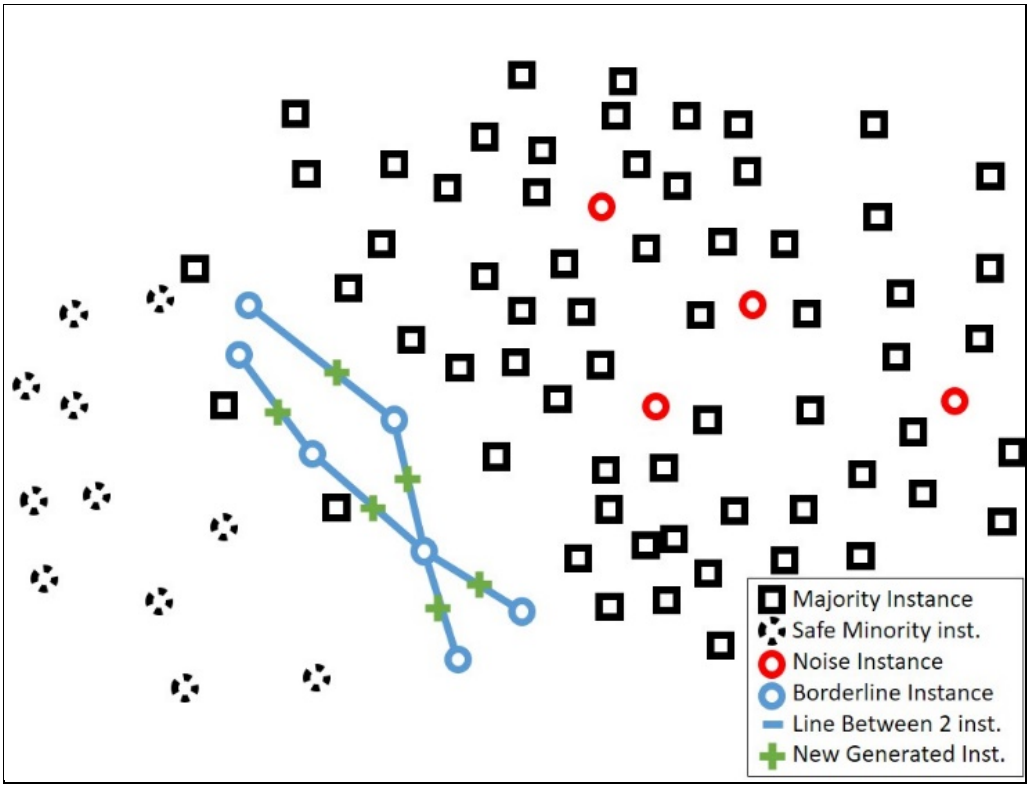


Figure 4.1: AB-SMOTE Diagram

Algorithm 3:AB-SMOTE

Input: P number of minority class sample; S% amount of synthetic to be generated; M number of nearest neighbors to create the borderline subset; k Number of nearest neighbors
Output: (S/100)* P' synthetic samples
1. **MinDanger ( )** to create danger subset
2. **ComputKNN (i←1 to P', P_i',P_j')**
3. Ns= (S/100)*P'
   While Ns ≠ 0
4. ***GenerateS (***P'$_i$, P$_j$'**)**
   Ns = Ns – 1
   End while
5. Return (∗ End of Populate. ∗)

End of Pseudo-Code.

Figure 4.2: AB-SMOTE Pseudo-Code

4.2. Clustered AB-SMOTE (CAB-SMOTE)

After creating the AB-SMOTE and records its results, the thesis notices the opportunity to upgrade it more, when it stumbles with the k-means approach. This approach is proposed by (Douzas, Bacao and Last, 2018) which cluster all the minority class instances and oversample within the clusters. The thesis proposes CAB-SMOTE that only cluster the minority borderline instances. Dividing the borderline into different segments that have their instances, allowing the algorithm to read and compare these clusters in terms of the number of instances to be used for the computational process. The oversampling process is then applied inside each cluster independently of the others, therefore narrowing the oversampling scope within the clusters inside the borderline.

To test the algorithm from its different sides, the thesis applies the oversampling techniques in different ways depending on different criteria. Then applies classification on the dataset to check which one gave better results. The thesis separated these different strategies of CAB-SMOTE to SI, SII, and SIII. All of them have the same process of building up the danger subset. After that, the k-means clustering is applied over the danger subset to divide it into different clusters. The thesis applies the algorithm more than once independently, each run with different k-

mean values from 2, 3, 5, and 7 for narrowing the experiment scope. The minimum value for k-means clustering to function is 2, where it divides the data into two clusters. So the thesis starts from the k-means value of 2 and then increase that value to 3, 5, and 7. Increasing the k-means value to more than 7 would increase the computation cost and will not be useful for increasing the classification accuracy of the minority class, that is why the research stopped at the value of 7. After the clustering process is applied, the algorithm oversamples within the clusters using the seed instances within each cluster and KNN instances around that see instances inside the same cluster as shown in Figure 4.3, Figure 4.5, and Figure 4.7 which display how CAB-SMOTE SI, SII, and SIII works.

The different oversampling strategies work as follows:

4.2.1. CAB-SMOTE Strategy I

The algorithm creates the danger subset using Mnn in the same way as B-SMOTE creates it, then it applies k-means clustering over danger subset, counts the instances within each cluster, and holds the largest one, after that it oversamples all the other clusters till they reach the size of the largest cluster. Thus, equalizing the number of instances inside the borderline clusters.

Figure 4.3 shows how this oversampling strategy works by clustering the danger area into three different clusters, each having 7, 10, and 6 instances respectively, oversample 3 new instances in the first cluster, holds the second cluster unchanged, and oversample 4 new instances in the third cluster. Hence making all the clusters have the same number of instances Figure 4.4 shows the pseudo-code this algorithm. This algorithm is then tested over different datasets and the F-measure values were compared with other oversampling results.

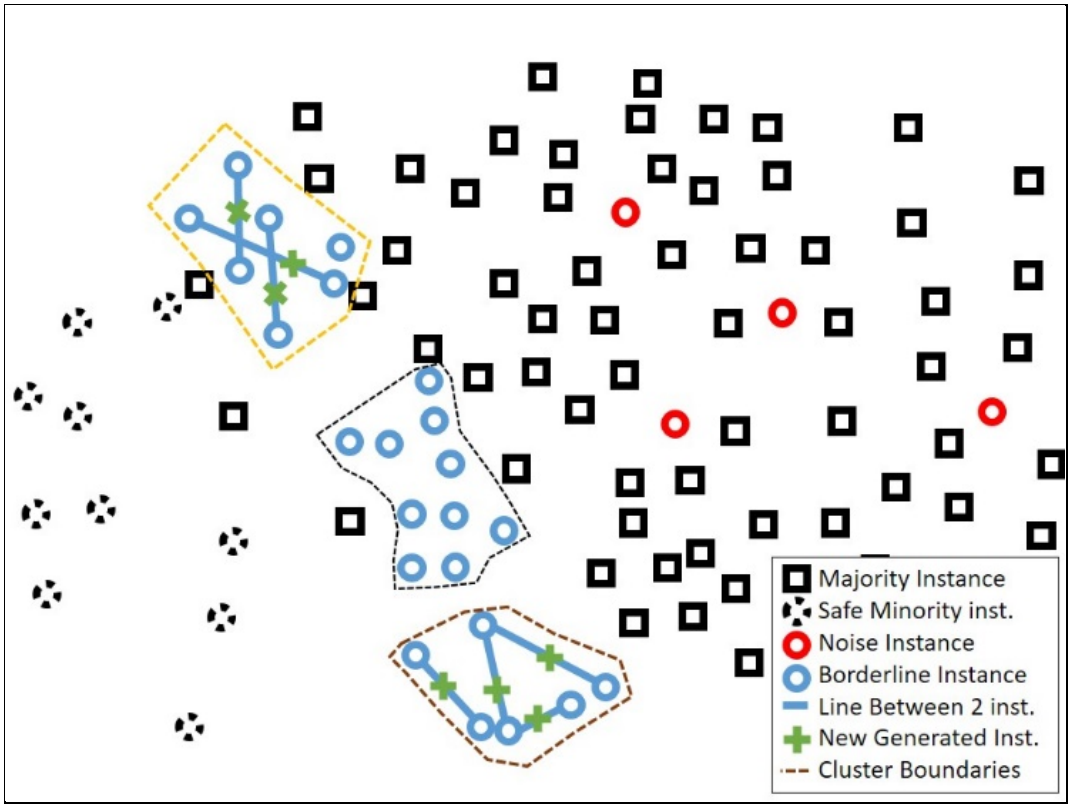


Figure 4.3: CAB-SMOTE SI Diagram

Algorithm 4:CAB-SMOTE Strategy1

Input: P number of minority class sample; S% amount of synthetic to be generated; M number of nearest neighbors to create the borderline subset; k Number of nearest neighbors
Output: (S/100)* P' synthetic samples

1. **MinDanger ( )** to create danger subset
   Apply k-means clustering on P' subset
   Nmax: number of instances found in the largest cluster
2. **For** km ← 0 to k-means
   Ns=0
3. Ns= Nmax - number of instances in cluster km
   While Ns ≠ 0
4. **ComputKNN (i←1 to P', P_i',P_j')**
5. ***GenerateS (***P'_i, P_j'**)**
   Ns = Ns – 1
   End while
   End for
6. Return (∗ End of Populate. ∗)

End of Pseudo-Code.

Figure 4.4: CAB-SMOTE SI Algorithm

4.2.2. CAB-SMOTE Strategy II

This algorithm creates the borderlines subset, cluster it with k-mean clustering, counts the instances found inside each cluster, and only oversamples the clusters which are larger than half of the largest cluster including the largest one. Figure 4.5 shows how this algorithm generates the new instances and Figure 4.6 display its pseudo code. The algorithm stops the oversampling process when it reaches the number attained by multiplying the oversampling percentage by the number of instances in each cluster that is larger than half of the largest cluster. Because not all the datasets have the same properties, some of them get higher classification accuracy if clusters which are larger than 0.3 of the largest one got oversampled, while others might get highest if the algorithm oversampled clusters larger than 0.7 of the largest cluster. The thesis experimented with different criteria values of which clusters to oversample, between 0.3 to 0.7 of the largest cluster, it was found that 0.5 times the largest cluster delivered the highest classification accuracy among the tested datasets.

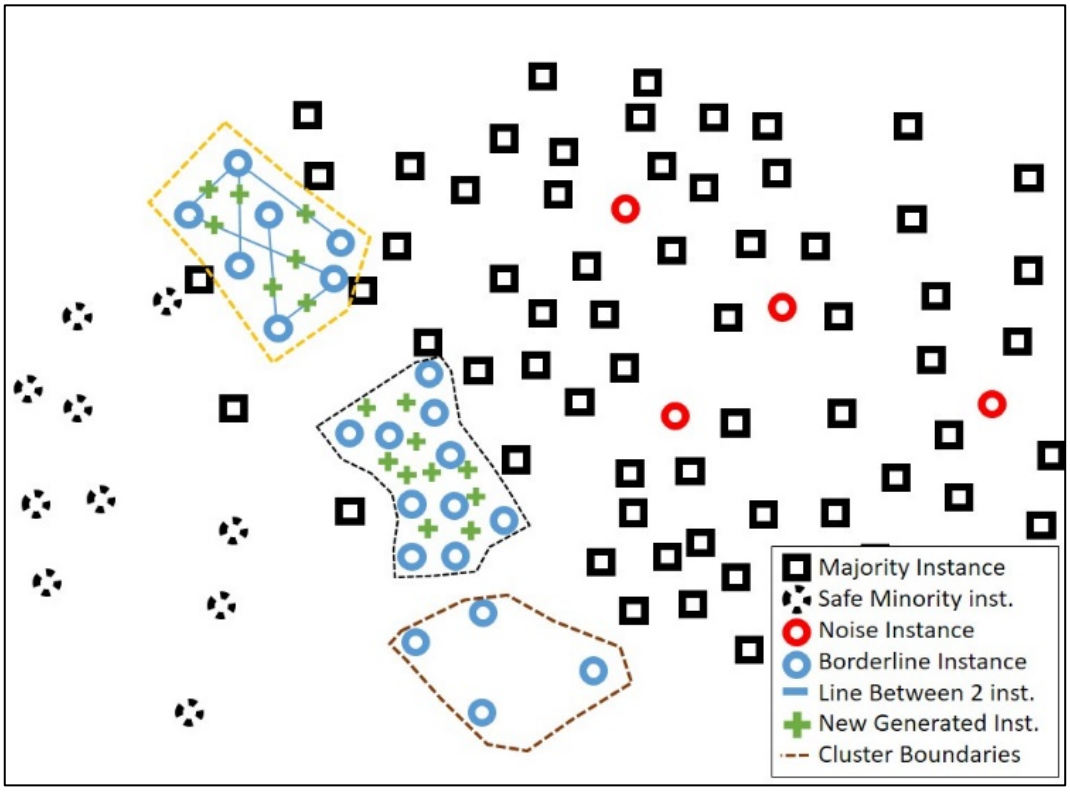


Figure 4.5: CAB-SMOTE SII Diagram

Algorithm 5:CAB-SMOTE Strategy 2

Input: P number of minority class sample; S% amount of synthetic to be generated; M number of nearest neighbors to create the borderline subset; k Number of nearest neighbors

Output: (S/100)* P' synthetic samples

1. **MinDanger ( )** to create danger subset
   Apply k-means clustering on P' subset
   Nmax: number of instances found in the largest cluster
2. **For** km ← 0 to k-means
   Ns=0
3. **If** number of instances in cluster km >= 0.5* Nmax
   Ns= (S/100)*P'
   While Ns ≠ 0
4. **ComputKNN (i←1 to P', P_i',P_j')**
5. ***GenerateS (***P'_i, P_j'**)**
   Ns = Ns – 1
   End while
   End if
   End for
6. Return (∗ End of Populate. ∗)

End of Pseudo-Code.

Figure 4.6: CAB-SMOTE SII Algorithm

4.2.3. CAB-SMOTE Strategy III

Creates the borderline subset, cluster it with k-mean clustering, and oversamples all clusters without any criteria till it reaches the given oversampling limit, which is referred by the oversampling percentage multiplied by the number of instances within each cluster. Figure 4.7 displays how the algorithm oversamples new instances when the oversampling percentage is at 100 percent, thus doubling the number of instances found inside each cluster. The thesis develops this strategy as a benchmark, and to compare between oversampling all clusters and oversampling just the clusters that are larger than half of the largest cluster. this is done to check the importance of small clusters in the danger area over classification accuracy.

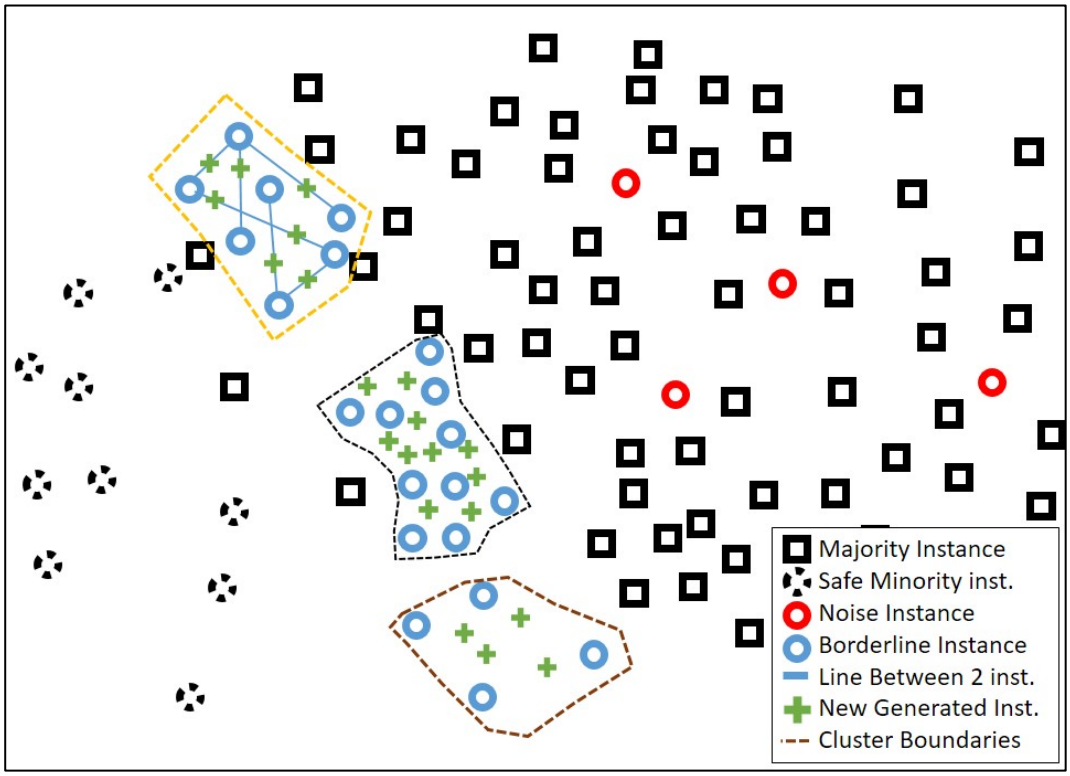


Figure 4.7: CAB-SMOTE SIII Diagram

4.3. Hybrid CAB-SMOTE (HCAB-SMOTE)

This method combines between undersampling and oversampling techniques, it uses the undersampling method to remove noisy majority class borderline instances and small minority class clusters within the borderline area of the dataset. The way it works is that it collects all the needed instances (without the noisy ones) from the initial dataset, add to them the newly generated instances and create and extract a new dataset to be used for model building and classification. Figure 4.8 displays what happens inside the dataset, showing the newly generated instances using CAB-SMOTE SII and the instances that are removed. Figure 4.9 present HCAB-SMOTE pseudo-code that explain the steps that the algorithm follows. HCAB-SMOTE create different subsets to function, these subsets are listed below:

-*Total*: holds all the instances, and is used to compute the nearest neighbors.

-*Minority:* holds all minority instances P.

-*MinDanger*: consist of minority border instances P'.

-*MinNonDanger*: holds minority non-border instances, categorized as safe instances.

-*Majority:* holds all majority instances N.

-*MajorityNonDanger:* hold non-border majority instances, which are categorized as safe instances.

-*Generated:* holds oversampled new synthetic instances S.

-*Newtotal*: hold non-noisy minority danger (clusters that have instances more than half of the largest clusters), MinNonDanger, MajNonDanger, and generated instances.

HCAB-SMOTE uses the same process found in the original SMOTE that count the number of instances within each class and then compare them to differentiate the majority and minority class and create the *Minority* and *Majority* subsets. The *MinDanger* subset is created by checking the M nearest neighbor (Mnn) between each minority instance and all instances in the dataset (Total), the algorithm checks within 8 nearest instances around the seed minority instance and counts the number of majority instances (M'), if the number of M' is between M/2 and M then this instance is considered a minority borderline instance and is added to *MinDanger* subset, which is the same process of creating the borderline area as of B-SMOTE. In contrast for the *MajNonDanger* subset which is formed from majority class instances found in a place away from the minority instances, the algorithm first copies all the majority instances into *MajNonDanger* subset, checks the Mnn between each majority instance N_i and all other instances, counts the number of minority instances found within the Mnn, if the number of minority instances falls between M/2 and M, the algorithm deletes that instance N_i from the *MajNonDanger* subset. After that, the algorithm clusters the *MinDanger* subset, separating it into *k* clusters, counts the number of instances within each cluster, and holds the number of instances found in the largest one. Then it compares the number of instances in each cluster to the number of instances inside the largest one, if a cluster has more than half of the instances of the largest cluster, the algorithm oversample that cluster and put the newly generated instance inside the *Generated* Subset. After the oversampling is occurred the algorithm copy the instances within the minority borderline clusters larger than half of the largest one, instances from the *MajNonDanger* subset, and the *Generated* subsets are copied to the *Newtotal*

subset, which is exported to a new arff dataset file named HCA-BSMOTE. This new dataset is then loaded into WEKA to apply the classification algorithm for model building and evaluation. It extracts a new dataset because it was not possible to remove the negative borderline instances from the main dataset while the program was running.

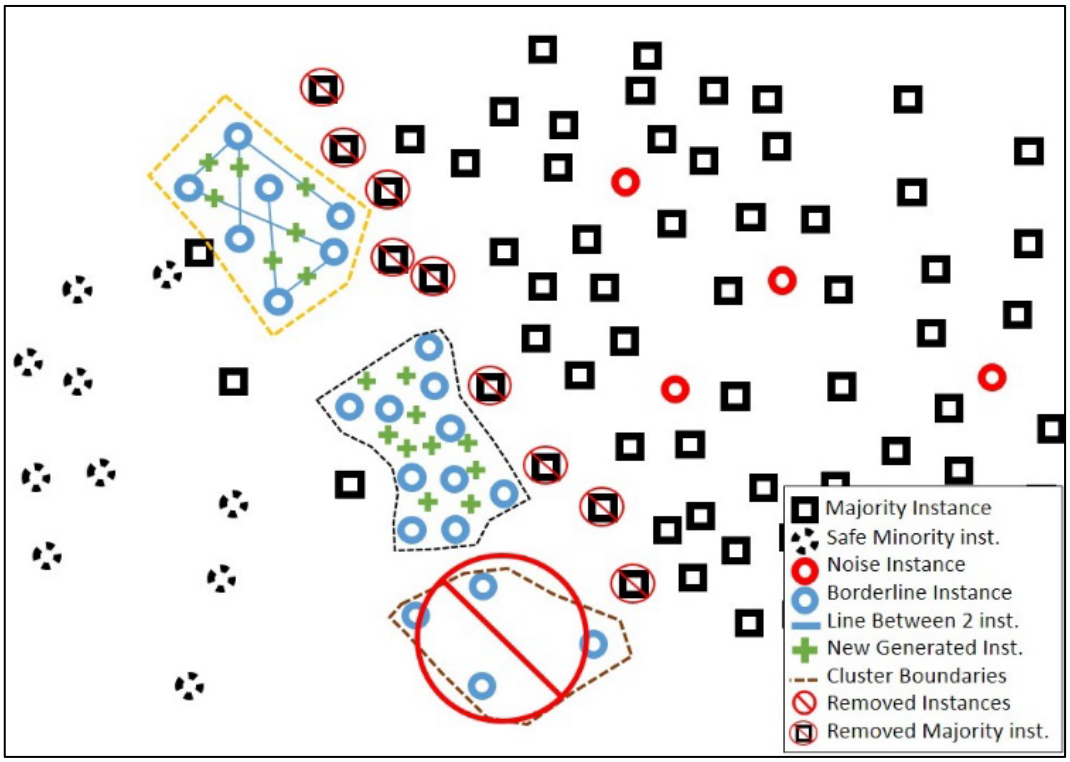


Figure 4.8: HCAB-SMOTE Diagram

Algorithm 6:HCAB-SMOTE

Input: P number of minority class sample; S% amount of synthetic to be generated; M number of nearest neighbors to create the borderline subset; k Number of nearest neighbors; k-means value

Output: (S/100)* P' synthetic samples

1. **Call MinDanger** to create danger subset
   Add all majority instances to MajNonDanger arrey
2. **For** i ← 1 to N
   Compute M nearest neighbors for each majority instance
   Ni and other instances from the dataset.
   Check number of Minority instance P within the Mnn
   If M/2< P < M
   remove instance Ni from MajNonDanger
   End if
   End for
   Apply k-means clustering on MinDanger P' subset
   Nmax: number of instances found in the largest cluster
3. **For** km ← 0 to k-means
   Ns=0
4. **If** number of instances in cluster km >= 0.5* Nmax
   Copy km cluster to Newtotal subset
   Ns= (S/100)*P'
   While Ns ≠ 0
5. **Call ComputKNN (i←1 to P', P_i',P_j')**
6. ***GenerateS (***P'_i, P_j'**)**
   Ns = Ns – 1
   End while
   End if
   End for
7. Copy Majoritynondanger, Minoritynondanger, and Generated subsets to Newtotalsubset.
8. Export Newtotalsubset

End of Pseudo-Code

Figure 4.9: HCAB-SMOTE Approach

CHAPTER 5
METHODOLOGY AND EVALUATION

To evaluate the oversampling approaches, the thesis uses WEKA (*Weka*, 2020), an open-source JavaScript software for data mining that enables anyone to access, read, and modify its code. This thesis focuses on the oversampling filter SMOTE, which is available in WEKA. Because WEKA does not have the B-SMOTE build in its filters, B-SMOTE is developed and implemented inside WEKA by the thesis via modifying SMOTE. The thesis also creates AB-SMOTE, CAB-SMOTE, and HCAB-SMOTE as they are displayed in Figure 5.1 that presents WEKAs window showing the oversampling techniques, SMOTE, Borderline SMOTE, AB-SMOTE, CAB-SMOTE SI, SII, SII, and HCAB-SMOTE.

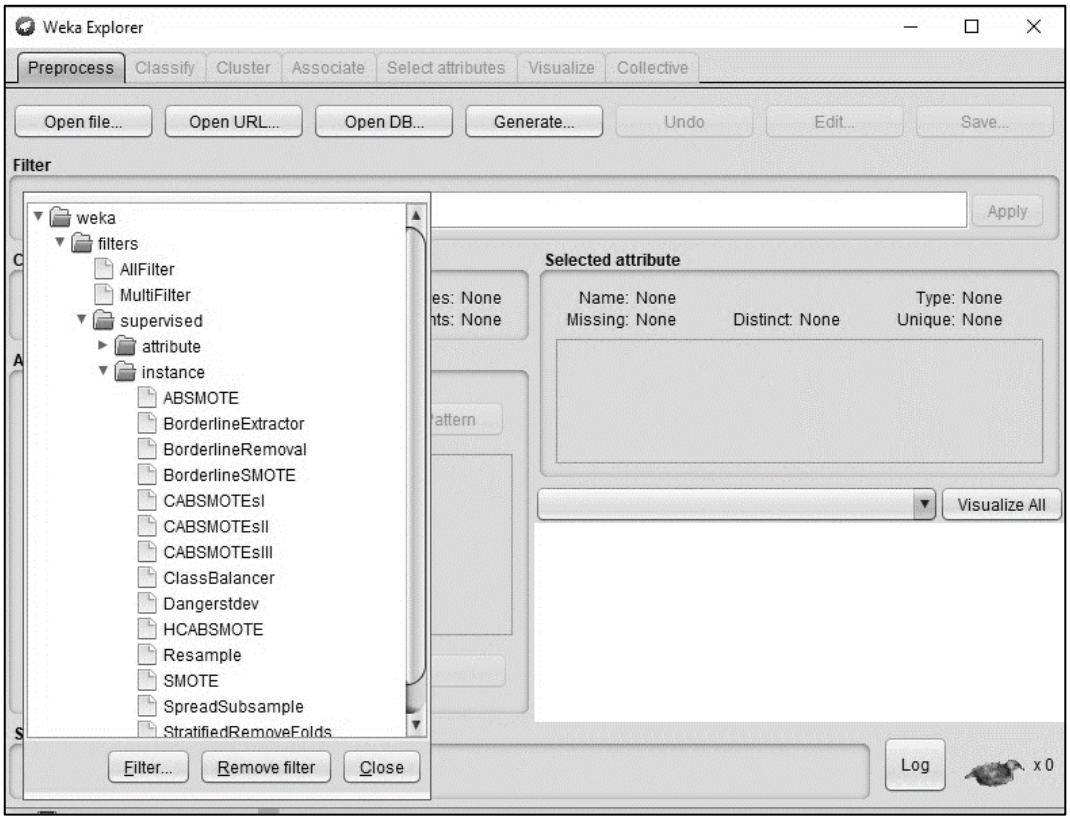


Figure 5.1: WEKA Windows Interface

Different supervised binary real datasets were used to evaluate and test the classification accuracy after applying different oversampling techniques. These datasets are displayed in Table 5.1 and obtained from different online repositories widely used in machine learning scientific researches. These datasets were downloaded from Crowd Analytics, UC Irvine Machine Learning Repository, Keel, and IBM. This research used the same criteria considered in (Fernández *et al.*, 2013) for choosing the imbalanced datasets, where the number of minority instances should be less than 40% of the number of instances of the majority class to be considered as an imbalanced dataset. The Imbalanced ratios (IR) used in Table 5.1 is computed by dividing the number of instances of the majority class over the number of instance of the minority class, and if the value is more than 1.5 then the data set is set to be imbalanced one. A class distribution measure can also be used to show the percentage of minority instances to the majority ones in the dataset, which is computed by dividing the number of minority instances over the number of majority instances $\frac{Pi}{Ni} \times 100$, that is when its value is less than 40%, the dataset is considered to have imbalance class problem. The datasets varied between abalone seashells age prediction dataset (*Keel Datasets, Abalone9-18*, 1995), two Customer churn datasets (*Crowd Analytix*, 2012), and (*IBM Analytics Telco Customer Churn Dataset*, 2018) that recorded historical data about customers and flagged the churned ones that unsubscribed from the service, to create a model to predict the customers who might churn in the future to develop customer retention programs to keep those customers from churning. Haberman's Survival dataset (Dua, Dheeru and Graff, 2017) which consists of recoded data from a study conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on patients who had undergone surgery for breast cancer and recorded if they died or survived within 5 years after the surgery (Haberman, 1976), and Employee Attrition dataset (*IBM & Kaggle Employee Attrition*, 2018) for tracking and analyzing employee satisfaction for retaining valuable employees to maintain good productivity. Solar Flare dataset (*Keel Datasets, Solar Flare*, 1989), Wine quality (*Keel Datasets, Wine Quality*, 2009) which rely on different features such as wine acidity, sugar, PH, alcohol levels and other measurements to predict the good wine quality from the bad ones, Sales dataset to understand the sales pipeline and uncover what can lead to

successful sales opportunities to better anticipate and address performance gaps using different features (*IBM Analytic, Win Loss*, 2017). Table 5.1 display the total number of instances in Total column, number of instances within the minority class within the Minority column, number of instances within the majority class within the Majority column, Imbalanced Ratio within I.R. columns, number of features found within each dataset without counting the Class as a feature within Number of features column, and the source of where the thesis got the dataset from withing the Source column.

Table 5.1: Datasets Used in The Research

Datasets	Total	Minority	Majority	I.R.	Number of features	Source
Abalone 9-18	731	42	689	16.4	8	(*Keel Datasets, Abalone9-18*, 1995)
Customer Churn Wireless Telecom	5000	707	4293	6.07	20	(*Crowd Analytix*, 2012)
Telco Customer Churn	7043	1869	5174	2.76	20	(*IBM Analytics Telco Customer Churn Dataset*, 2018)
Haberman	306	81	225	2.77	3	(Dua, Dheeru and Graff, 2017)
Employe Attrition	1470	237	1233	5.2	34	(*IBM & Kaggle Employee Attrition*, 2018)
Flare	1109	43	1066	24.79	11	(*Keel Datasets, Solar Flare*, 1989)
Wine Quality	1493	53	1546	29.16	11	(*Keel Datasets, Wine Quality*, 2009)
Sales: Win Loss	78025	17627	60398	3.42	18	(*IBM Analytic, Win Loss*, 2017)
Bank	45211	5289	39922	7.54	16	(Moro, Laureano and Cortez, 2011)
Adult Census income	32561	7841	24720	3.15	14	(Becker, 1996)
Amazon Employee access	32769	1897	30872	16.27	9	(Challenge, 2013)
Click Prediction	39948	6728	33220	4.93	11	(Cup, 2012)

Bank (Moro, Laureano and Cortez, 2011) which was related to banking direct phone marketing campaigns of a Portuguese banking institution to ask clients to subscribe for bank term deposit which was related to banking direct phone marketing campaigns of a Portuguese banking institution to ask clients to subscribe for a bank term deposit.

An Adult Census income (Becker, 1996) for determining if the person makes over $50k per year. Amazon Employee access dataset (Challenge, 2013) which used to classify which new employees would have denied access which stops them from fulfilling their role. And a Click Prediction dataset (Cup, 2012) to predict if a user will click on a given advertisement that pops up in specific websites.

After obtaining the datasets, the research first applied the decision tree classification algorithm, which is a predictive model created by dividing and conquer iterations of hierarchical decisions, where these iterations are applied over the dataset to split it into homogenous subgroups using the inter independent features. Thus, creating a tree form shaped like IF, Then, Else rules from the beginning till the end of its leaf nodes as displayed in Figure 5.2. The decision tree algorithm is related to rule learning methods and might suffer from some disadvantages as well, such as having a bias of the model due to noisy examples and outliers (García *et al.*, 2015). The thesis chose decision trees because it is known to be sensitive to class imbalance and was often used in studies of SMOTE and their extensions (Batista, Prati and Monard, 2004),(Bunkhumpornpat, Sinapiromsaran and Lursinsap, 2009), and (Chawla *et al.*, 2002). The decision tree is referred to as J48 in WEKA and will be applied over each dataset without applying any oversampling.

Figure 5.2: Decision Tree Demonstration

K-folds cross-validation with k=5 was used while performing the classification algorithm, which divides the dataset into 5 folds with the same class distribution

proportion. Which means that if the dataset had 1000 instances consisting of 800 majority instances and 200 minority instance, and undergo through 5 fold cross-validation, this dataset will be divided into 5 different subsets, each one of them will have 200 instances consisting of 160 majority instances and 40 minority instances, thus keeping the same class distribution. While several authors recommend using k larger than 10 for cross-validation, this thesis could not apply it because the thesis is dealing with imbalanced data sets, where the use of k=10 is unreasonable. After all, the minority class would have very few numbers of instances inside each fold (Cervantes *et al.*, 2017). After the 5-fold cross-validation dividing process occurs, the classification algorithm will be applied five times. In each iteration, the cross-validation will use one subset and use 4 folds of that subset for training and holds the remaining fold for testing and record the results in the memory. For the other iterations, it will use different subset where it will hold a different testing fold and train on the other folds, and record the results of each iteration in the memory. After the last fold cross-validation is done, it prints out the average of those results into WEKA results window displayed in Figure 5.3 (Cervantes *et al.*, 2017) (López, Fernández and Herrera, 2014).

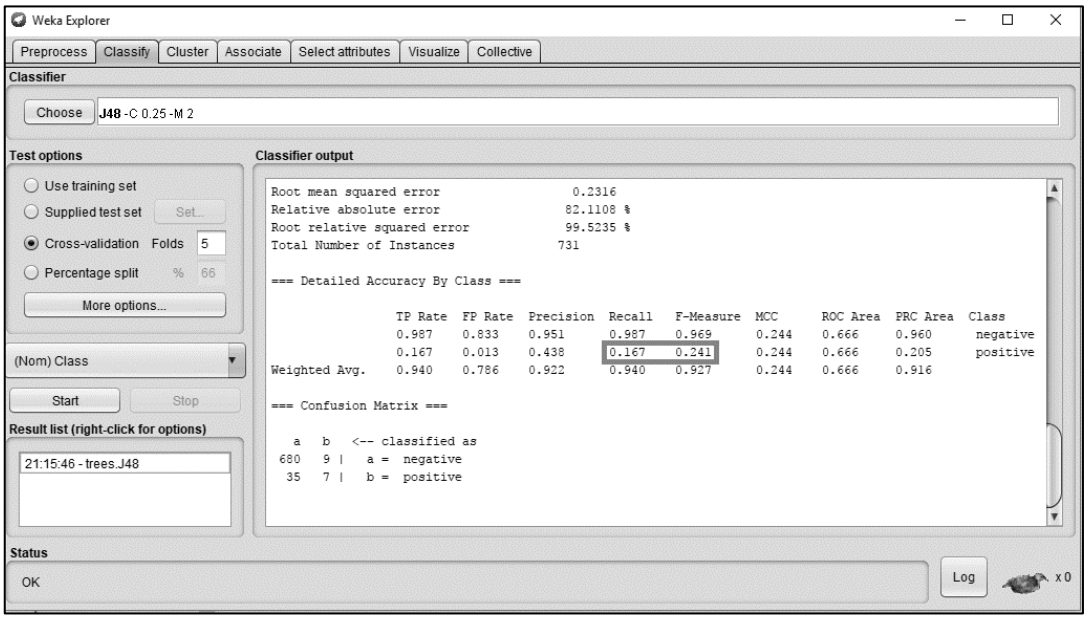


Figure 5.3: WEKA Result Output

After recording the results of using the classification algorithm without oversampling the data, the research applied different oversampling techniques with different tuning parameters using 5-fold cross-validation to compare the results as shown in Figure 5.4 Table 5.2. Figure 5.4 displays the thesis framework that applied Decision Tree, Naïve Bayes, and Random Forest classifications over the original data, then applied one oversampling technique at each run, build up the classification models using the three classifiers independently, and recorded their classification results. After that the thesis recorded the results in separate tables grouped by the same classification technique, to compare the results within each classification technique bit itself (Decision Tree, Naïve Bayes, and Random Forest classifiers).

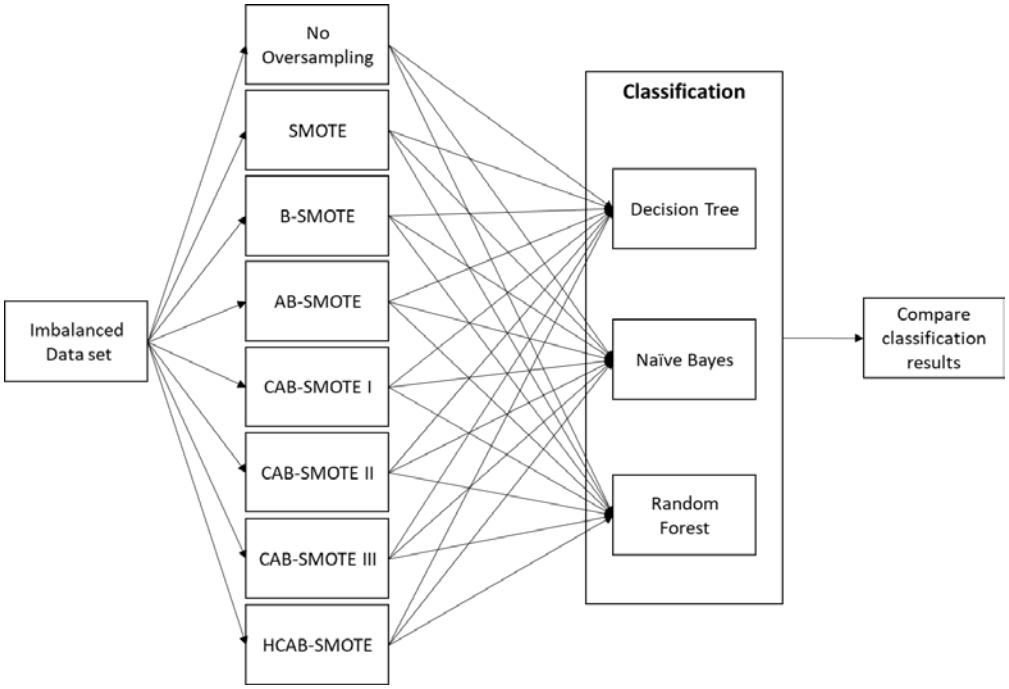


Figure 5.4: Research Methodology

Table 5.2 displays in the algorithms' column a number followed by 'x' before the algorithm name. Number 1.0 stands for the original dataset without any oversampling, whereas the numbers from 2 to 8, each one of them stands for a specific oversampling method followed by 'x' for referring to the different versions of the oversampling method when using different parameters. These parameters are the KNN with values equal to 5 and 8 independently, and the values of K-mean clustering ranging between

2,3,5, and 7. ‘x’ values are limited to the number of versions V (2 to 8). SI, SII, SIII stands for different Strategies used for CAB-SMOTE, and V. is the number of versions of the oversampling technique, while each version has different KNN and Mnn values. The first version uses KNN and Mnn with a value of 5 for each, while the second version uses KNN and Mnn with a value of 8 for each. These two versions were applied for all the techniques, but regarding CAB-SMOTE and HCAB-SMOTE they were repeated for each k-means value (2, 3, 5, and 7). P denotes to the positive minority instances, P’ denotes to the minority borderline instances, N denotes to the majority instance, Gap is the number which is multiplied by the distance between the seed instance and the random instance from its KNN to pinpoint the location of where the new instance would be located within the line connecting both instances and assign the feature values to the new instance. Mnn value to be used between minority instances and the whole dataset for defining the borderline subset, KNN to be used between the seed minority instances in the final stage of oversampling, k-means is the number of clusters that the danger area will be divided into, and Per. stands for oversampling percentage which will be multiplied with specific minority instances to calculate the number of instances to be generated. After getting the results, the Recall and F-measure values for classifying the minority class (explained in the next section) for each dataset were recorded, and the version that delivered the highest F-measure value within each oversampling approach was chosen to compare them. At a later stage, the research applied to oversample over those versions with different oversampling percentages ranged from 50, 100, 200, 300, up to 400 to observe the progress of increasing the oversampling rate using the different methods, the results will be displayed in the results section. However, the algorithm B-SMOTE SII 3.3 and 3.4 that use minority borderline instances and majority instances for the oversampling process did not deliver better results than B-SMOTE 3.1 and 3.2, therefore these oversampling methods were not included in the oversampling with increasing oversampling rates.

Table 5.2: Oversampling Techniques Used in The Experiment

Algorithm	Mnn	KNN	GAP	k-means	Per.
1.0: Original	-	-	-	-	-
2.x: SMOTE 2V	-	5,8	0-1	-	100
3.x: B-SMOTE P'.P 2V	5,8	5,8	0-1	-	100
3.3: B-SMOTE SII P'.N 2V	5,8	5,8	0-0.5	-	100
4.x: AB-SMOTE P'.P' 2V	5,8	5,8	0-1	-	100
5.x: CAB-SMOTE SI 8V	5,8	5,8	0-1	2,3,5,7	-
6.x: CAB-SMOTE SII 8V	5,8	5,8	0-1	2,3,5,7	100
7.x: CAB-SMOTE SIII 8V	5,8	5,8	0-1	2,3,5,7	100
8.x: HCAB-SMOTE 8V	5,8	5,8	0-1	2,3,5,7	100

The classification accuracy measurement is not sufficient to get a precise evaluation of any classifier applied over an imbalanced dataset because it does not reflect the true classification accuracy in predicting instances for the minority class. For that reason, a confusion matrix such as the one displayed in Table 5.3 is used for monitoring different metrics to get a detailed image of the results which deliver a better evaluation of the classifier predicting power for instances having minority class (Saito and Rehmsmeier, 2015). As mentioned before for the imbalanced datasets the majority class instances are referred to as Negative instances, whereas the minority class instances are referred to as Positive instances. For the classifier to have a perfect prediction accuracy regarding the minority instances in the confusion matrix, the true positive (TP) number should be equal to the total number of the minority instances in the dataset, which implies that the model is correctly predicting all actual positive instances to be in a positive class. Since the research is using supervised datasets that have a defined class for each instance, the prediction accuracy calculation is possible by comparing the predicted class for instances with their actual one. Below are the Confusion Matrix definitions:

- -TP/ True Positive: Instances that are predicted to be in the minority class and its actual class is a minority
- -FP/ False Positive: Instances that are predicted to be in the minority class but its actual class is majority (type 1 error).
- -FN/ False Negative: Instances that are predicted to be in the majority class but its actual class is a minority (type 2 error).

-TN/ True Negative: Instances that are predicted to be in the majority class and its actual class is also the majority class.

-Recall: The ratio of correctly classified minority positive instances from all actual minority positive instances.

-P.P.V. / Positive Prediction Value (Precision): The ratio of correctly classified minority instances from all predicted minority instances.

The classification accuracy of the datasets without and with oversampling was evaluated with recall and F-measure are displayed in the WEKA window after applying the decision tree with 5-folds cross-validation as shown in Figure 5.3.

F-measure is a measurement that displays the harmonic mean of recall and precision (Saito and Rehmsmeier, 2015), calculated with Equation 5.1.

$$\text{F\_Measure} = \frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{ Recall}} \tag{5.1}$$

Table 5.3: Confusion Matrix

		Actual Values			
		Positive	Negative		
Predicted Values	Positive	TP	FP	P.P.V.	$\frac{TP}{(TP+FP)}$
	Negative	FN	TN	N.P.V	$\frac{TN}{(FN+TN)}$
		Recall $\frac{TP}{(TP+FN)}$	**Specificity** $\frac{TN}{(TN+TP)}$	**Accuracy** $\frac{(TP+TN)}{(TP+FP+TN+FN)}$	

For achieving the maximum accuracy, recall and F-measure values should be very close to 1. If their value is equal to 1, that implies that the model is correctly predicting all the instances that have actual minority class instances to be in the minority class and all the instances that have actual negative class instances to be in majority class. The F-measure value reflects precision and recall, if F-measure is high it means that precision and recall values are also high. That is why the thesis chose to focus on the

F-measure value in the experiment.

The Thesis first conducted research and proposed Affinitive Borderline SMOTE (AB-SMOTE) where it was evaluated against Borderline SMOTE (B-SMOTE). The thesis used 12 datasets for the research, these datasets will be further explained in section five. Both methods generate the same number of instances because the number of new instances to be generated is computed by multiplying the oversampling rate by the number of instances found in the danger area, which is the same by the two methods. After applying both methods separately over the datasets, the minority recall and F-measures values were extracted for performance comparison and evaluation. The results show that AB-SMOTE got higher recall and F-measure than B-SMOTE within most of the datasets as shown in Table 5.4. Table 5.4 displays the names of the datasets used, number of instances within each dataset, recall, and F-measure values of decision tree classification after the B-SMOTE and AB-SMOTE were applied to the datasets. This gives the thesis more motivation to develop AB-SMOTE to CAB-SMOTE for further optimization of the oversampling effect over the classification accuracy with generating the fewer or the same number of instances.

Table 5.4: B-SMOTE and AB-SMOTE Accuracy Comparison

DATASET	Number of inst.	Recall		F-Measure	
		B-SMOTE	AB-SMOTE	B-SMOTE	AB-SMOTE
Abalone	731	**0.395**	0.37	**0.496**	0.438
Wireless Customer Churn	5000	**0.595**	0.585	**0.718**	0.711
Telco Customer churn	7043	0.895	**0.897**	0.677	**0.678**
Haberman	306	0.541	**0.57**	**0.583**	0.581
Employee Attrition	1470	0.501	**0.516**	0.59	**0.594**
Flare	1109	**0.369**	0.293	**0.434**	0.421
Wine Quality	1493	**0.151**	**0.151**	**0.215**	**0.215**
Sales, win loss	78025	0.799	**0.804**	0.817	**0.826**
Bank Subscription	45211	0.684	**0.685**	0.693	**0.703**
Adult income	32561	0.772	**0.788**	0.803	**0.81**
Amazon Employee	32769	0.486	**0.488**	0.577	**0.587**
Click Prediction	39948	0.406	**0.565**	0.397	**0.555**
TOTAL		**4**	**9**	**5**	**8**

CHAPTER 6
RESULTS AND DISCUSSION

Tables 6.1 to 6.12 display the minority classification recall and F-measure results obtained from classifying the datasets before and after applying 100% oversampling percentage, the number of majority instances (MAJ.), number of minority instances (MIN.), number of instances within the borderline area at Mnn= 5 and Mnn=8 fallowed by their percentages from the minority samples which are used with B-SMOTE and the techniques that follow, the imbalanced ratio (I.R.) referred to the number of majority instance over the number of minority instances, the oversampling technique number x (x.0) when x values starts from the value of '1' where no oversampling was applied, to the value of '8' when HCAB-SMOTE is applied, followed with the oversampling version number that got the highest F-measure result y (0.y) that belong to interval between 1 and 8, where x.y are written on the left side before the algorithm name, K nearest neighbors, M nearest neighbors and k-means values used while applying these techniques (x.y: method name KNN, Mnn, K-means). The B% column represents the percentage of generated instances over the borderline size to display how much the algorithm generated fewer instances than the former ones, where the values are always 0% when there was no oversampling or when using SMOTE because there was no borderline in those processes, and having a value of 100% while using B-SMOTE and AB-SMOTE because they oversample all of the borderline. The table also records the number of generated new instances GEN, Recall, and F-measure values, and the highest and best results of recall and F-measure are bolded.

After applying all the techniques using the same 100% oversampling rate. HCAB-SMOTE and other techniques are compared by the difference in generated instances (Dif.GEN) with Equation 6.1 and F-measure difference (Dif. F measure) with Equation 6.2 and then the results for each dataset are analyzed.

$$\text{Dif. GEN} = \left|\left(\frac{\text{GEN.HCAB}-\text{SMOTE}}{\text{GEN.SMOTE}}\right) - 1\right| \tag{6.1}$$

$$\text{Dif. F measure} = \left|\left(\frac{\text{F.HCAB}-\text{SMOTE}}{\text{F.SMOTE}}\right) - 1\right| \tag{6.2}$$

Table 6.1, displays the results obtained from the Abalone dataset, where HCAB-SMOTE generates 7.14% fewer instances and attains the highest F-measure with 43.8% more than that of SMOTE, and generates the same number of instances as B-SMOTE and AB-SMOTE but improves 40.9% and 59% in the F-measure values respectively. Followed by CAB-SMOTE SII and SIII that delivers the same results because all borderline clusters were larger than half of the largest cluster.

Table 6.1: Results from Abalone Dataset

MAJ.= 689; MIN. = 42	**BORDER M5= 40 – 95.2% BORDER M8= 39 – 92.8%**			**I.R. = 16.4**
OVERSAMPLING TECHNIQUE	**B%**	**GEN**	**RECALL**	**F-MEASURE**
1.0: ORIGINAL	0	0	0.167	0.241
2.1: SMOTE: 8	0	42	0.417	0.486
3.1: B-SMOTE P' & P: 8, 8	100	39	0.395	0.496
4.1: AB-SMOTE P' & P': 8,8	100	39	0.37	0.438
5.4: CAB-SMOTE SI: 5,5,5	100	40	0.427	0.493
6.0: CAB-SMOTE SII: 5, 5, 2	100	40	0.549	0.629
7.0: CAB-SMOTE SIII: 5, 5, 2	100	40	0.549	0.629
8.1: HCAB-SMOTE: 8, 8, 2	100	39	**0.617**	**0.699**

Table 6.2 shows results obtained from the Customer Churn dataset. The results obtained without any oversampling had the highest recall and F-measure compared to the results achieved after applying SMOTE, B-SMOTE, and AB-SMOTE. Whereas after applying the CAB-SMOTE the results outperform the former ones. The highest F-measure value is achieved by HCAB-SMOTE that generates 38.8% less than SMOTE and 14% increase in the F-measure value and generated 19% less than CAB-SMOTE SIII gaining 1.3% more in f measure value. CAB-SMOTE SIII followed it for the second-highest F-measure values for this dataset outperforming SII because it oversamples all clusters.

Table 6.2: Results from Customer Churn Dataset

MAJ.= 4293; MIN.= 707	**BORDER M5= 534 - 75% BORDER M8= 517 - 73%**			**I.R. = 6.07**
OVERSAMPLING TECHNIQUE	**B%**	**GEN**	**RECALL**	**F-MEASURE**
1.0: Original	0	0	0.639	0.748
2.0: SMOTE 8	0	707	0.631	0.755
3.1: B-SMOTE P' & P:8,8	100	517	0.595	0.718
4.1: AB-SMOTE P' & P':8,8	100	517	0.585	0.711
5.3: CAB-SMOTE SI:,8,8,3	67	362	0.77	0.838
6.0: CAB-SMOTE SII:5,5,2	55	298	0.787	0.855
7.0: CAB-SMOTE SIII: 5,5,2	100	534	0.816	0.867
8.2: HCAB-SMOTE:5,5,3	80	432	**0.822**	**0.879**

Table 6.3 displays the results obtained from classifying the Haberman dataset with the decision tree classifier before and after applying different oversampling methods on the dataset. The HCAB-SMOTE algorithm outperforms other techniques and achieves the highest F-measure value. It generates 37% fewer instances than SMOTE and 18.35% increase in F-measure value. In comparison with CAB-SMOTE SII, they both generate the same number of instances, but HCAB-SMOTE delivers 16% more F-measure value which translated to the undersampling gain. CAB-SMOTE SIII achieves the second-highest recall and F-measure values.

Table 6.3: Results from Haberman Dataset

MAJ. = 225; MIN. = 81	**BORDER M5=65 - 80.2% BORDER M8=61 - 75.3%**			**I.R. = 2.76**
OVERSAMPLING TECHNIQUE	**B%**	**GEN**	**RECALL**	**F-MEASURE**
1.0: Original	0	0	0.272	0.331
2.0: SMOTE 5	0	81	0.617	0.643
3.0: B-SMOTE P' & P:5,5	100	65	0.541	0.583
4.1: AB-SMOTE P' & P': 8,8	100	61	0.57	0.581
5.5 CAB-SMOTE SI: 8,8,5	121	74	0.69	0.656
6.6: CAB-SMOTE SII: 5,5,7	78	51	0.712	0.651
7.5: CAB-SMOTE SIII: 5,5,3	100	61	**0.775**	0.673
8.6: HCAB-SMOTE: 5,5,7	78	51	0.729	**0.761**

Table 6.4 displays result obtained from the employee attrition dataset. The highest values of recall and F-measure are obtained after applying HCAB-SMOTE which oversamples 42.79% less than SMOTE and attains 17.96 % more classification accuracy. CAB-SMOTE SII and SIII achieve the second-highest recall and F-measure values where they generate 74.8% more and achieve 3.3% less in F-measure value.

Table 6.4: Results from Employee Attrition Dataset

MAJ.= 1233; MIN. = 237	**BORDER M5= 236 - 99% BORDER M8=232 – 97.8%**			**I.R.= 2.77**
OVERSAMPLING TECHNIQUE	**B%**	**GEN**	**RECALL**	**F-MEASURE**
1.0: ORIGINAL	0	0	0.181	0.265
2.0: SMOTE 8	0	236	0.501	0.59
3.0: B-SMOTE P' & P: 5,5	100	236	0.501	0.59
4.0: AB-SMOTE P' & P': 5,5	100	236	0.516	0.594
5.6: CAB-SMOTE SI: 5,5,7	89	212	0.528	0.605
6.0: CAB-SMOTE SII: 5,5,2	100	236	0.588	0.673
7.0: CAB-SMOTE SIII: 5,5,2	100	236	0.588	0.673
8.6: HCAB-SMOTE: 5,5,7	57	135	**0.605**	**0.696**

Table 6.5 displays the classification accuracy results from using the Telco Customer Churn dataset. HCAB-SMOTE outperforms the other oversampling methods by giving the highest F-measure value with 49.5 % less generated instances than of SMOTE and 10.3% more in the classification accuracy. SMOTE achieves the second-highest F-measures values. CAB-SMOTE SI reaches the highest recall value even while oversampling less than SMOTE. But it had a lower F-measure value, meaning that it was misclassifying actual negative instances as positive ones more than SMOTE, which would be costly misclassification such as giving more advertisements to a loyal client which was not going to churn. HCAB-SMOTE generates the same number of instances as CAB-SMOTE SII but achieved 12.5% more F-measure value due to the undersampling process.

Table 6.5: Results from Telco Customer Churn Dataset

MAJ.= 5174; MIN. = 1869	**BORDER M5=1101 - 58.9% BORDER M8=949 – 50.7%**			**I.R. = 5.2**
Oversampling Technique	**B%**	**GEN**	**RECALL**	**F-MEASURE**
1.0: Original	0	0	0.489	0.543
2.0: SMOTE 8	0	1869	0.923	0.732
3.0: B-SMOTE P' & P: 5,5	100	1101	0.895	0.677
4.0: AB-SMOTE P' & P': 5,5	100	1101	0.897	0.678
5.4: CAB-SMOTE SI: 5,5,5	114	1259	**0.932**	0.705
6.6: CAB-SMOTE SII: 5,5,7	99	943	0.811	0.718
7.6: CAB-SMOTE SIII: 5,5,7	100	1101	0.918	0.699
8.6: HCAB-SMOTE: 5,5,7	99	943	0.876	**0.808**

Table 6.6 displays the classification accuracy results from the Flare dataset where HCAB-SMOTE gets the highest results followed by CAB-SMOTE SIII. HCAB-SMOTE generates 9.3% fewer instances than SMOTE and delivers a 50.8% increase in classification accuracy, and generates the same number of instances as CAB-SMOTE SII but 15% less in F-measure value due to the undersampling process.

Table 6.6: Results from Flare Dataset

MAJ.= 1066; MIN.= 43	**BORDER M5= 39 – 90.6% BORDER M8=41 – 95.3%**			**I.R. = 24.79**
OVERSAMPLING TECHNIQUE	**B%**	**GEN**	**RECALL**	**F-MEASURE**
1.0: ORIGINAL	0	0	0	0
2.0: SMOTE 5	0	43	0.36	0.47
3.1: B-SMOTE P' & P: 8,8	100	41	0.369	0.434
4.0: AB-SMOTE P' & P': 5,5	100	39	0.293	0.421
5.1: CAB-SMOTE SI: 8,8,2	51	21	0.375	0.5
6.0: CAB-SMOTE SII: 5,5,2	100	39	0.488	0.615
7.1: CAB-SMOTE SIII: 8,8,2	100	41	0.56	0.618
8.0: HCAB-SMOTE: 5,5,2	100	39	**0.61**	**0.709**

Table 6.7 shows results from using the Wine Quality dataset where HCAB-SMOTE outperformed the other techniques followed by CAB-SMOTE SII. HCAB-SMOTE managed to generate 33.9% less than SMOTE and delivering 199% more F-measure value, and the same number of generated instances of CAB-SMOTE SII but with a 40% increase in F-measure values.

Table 6.7: Results from Wine Quality Dataset

MAJ.= 1546; MIN. = 53	**BORDER M5=53 – 100% BORDER M8=53 – 100%**			**I.R. = 29.169**
OVERSAMPLING TECHNIQUE	**B%**	**GEN**	**RECALL**	**F-MEASURE**
1.0: ORIGINAL	0	0	0	0
2.0: SMOTE 5	0	53	0.151	0.215
3.0: B-SMOTE P’ & P: 5,5	100	53	0.151	0.215
4.0: AB-SMOTE P’ & P’: 5,5	100	53	0.151	0.215
5.2: CAB-SMOTE SI: 5,5,3	98	52	0.286	0.38
6.3: CAB-SMOTE SII: 5,5,3	66	35	0.341	0.458
7.0: CAB-SMOTE SIII: 5,5,2	100	53	0.415	0.442
8.3: HCAB-SMOTE: 8,8,3	66	35	**0.529**	**0.643**

Table 6.8 displays result from the Sales Win-Loss dataset where HCAB-SMOTE gets the highest values followed by CAB-SMOTE SII. HCAB-SMOTE oversamples 13.9% less than SMOTE and 3.1% more F-measure value and generates the same number of instances as CAB-SMOTE SII but with a 2.7% increase in F-measure value.

Table 6.8: Results from Sales Win-Loss Dataset

MAJ.= 60398; MIN. = 17627	**BORDER M5=15176 – 86% BORDER M8=14620 – 82.9%**			**I.R. = 3.426**
OVERSAMPLING TECHNIQUE	**B%**	**GEN**	**RECALL**	**F-MEASURE**
1.0: ORIGINAL	0	0	0.644	0.695
2.0: SMOTE 5	0	17627	0.816	0.838
3.0: B-SMOTE P' & P: 5,5	100	15176	0.799	0.817
4.0: AB-SMOTE P' & P': 5,5	100	15176	0.804	0.826
5.7: CAB-SMOTE SI: 8,8,7	99	14618	0.806	0.839
6.2: CAB-SMOTE SII: 5,5,3	100	15176	0.808	0.841
7.0: CAB-SMOTE SIII: 5,5,2	100	15176	0.807	0.841
8.0: HCAB-SMOTE: 5,5,2	100	15176	**0.844**	**0.864**

Table 6.9 displays classification accuracy from the Bank Subscription dataset, where HCAB-SMOTE gets higher results followed by CAB-SMOTE SII & III. HCAB-SMOTE generated 18.1% less than SMOTE and 11.39% more F-measure, and it generates the same number of instances as CAB-SMOTE SII and SIII but delivered 8.5 increase in F-measure value.

Table 6.9: Results from Bank Subscription Dataset

MAJ.= 39922; MIN. = 5289	**BORDER M5= 4331 – 81.8% BORDER M8= 3896 – 73.6%**			**I.R. = 7.54**
OVERSAMPLING TECHNIQUE	**B%**	**GEN**	**RECALL**	**F-MEASURE**
1.0: ORIGINAL	0	0	0.49	0.546
2.1: SMOTE 8	0	5289	0.759	0.737
3.0: B-SMOTE P' & P: 5,5	100	4331	0.684	0.693
4.0: AB-SMOTE P' & P': 5,5	100	4331	0.685	0.703
5.6: CAB-SMOTE SI: 5,5,7	89	3873	0.696	0.743
6.0: CAB-SMOTE SII: 5,5,2	100	4331	0.708	0.756
7.0: CAB-SMOTE SIII: 5,5,2	100	4331	0.708	0.756
8.0: HCAB-SMOTE: 5,5,2	100	4331	**0.796**	**0.821**

Table 6.10 displays results achieved from the Adult dataset, where HCAB-SMOTE gets the highest accuracy results while generating 38.2 % less than SMOTE and 1.67% increase in F-measure values and generating 13.2% less than CAB-SMOTE SII and delivering 3.9% F-measure increase. CAB-SMOTE SI oversamples more than SMOTE because the borderline is clustered into 7 clusters and divided into small clusters and a large one. So, it oversamples the small cluster to reach the largest one, leading to this high number of generated instances and obtaining a higher F-measure value.

Table 6.10: Results from Adult Census Income Dataset

MAJ.= 24720; MIN. = 7841	**BORDER M5=5640 – 71.9% BORDER M8=5499 –**			**I.R. = 3.152**
OVERSAMPLING TECHNIQUE	**B%**	**GEN**	**RECALL**	**F-MEASURE**
1.0: ORIGINAL	0	0	0.625	0.684
2.1: SMOTE 8	0	7841	0.81	0.838
3.1: B-SMOTE P' & P: 8,8	100	5499	0.772	0.803
4.1: AB-SMOTE P' & P': 8,8	100	5499	0.788	0.81
5.6: CAB-SMOTE SI: 5,5,7	140	7926	0.81	0.842
6.0: CAB-SMOTE SII: 5,5,2	100	5640	0.774	0.82
7.2: CAB-SMOTE SIII: 5,5,3	100	5640	0.779	0.821
8.2: HCAB-SMOTE: 5,5,3	86	4893	**0.827**	**0.852**

Table 6.11 displays results from the oversampling Amazon Employees access the dataset. Where HCAB-SMOTE generated 8% fewer instances than SMOTE and gained 17.9% more F-measure value. HCAB-SMOTE outperformed the other techniques, followed by CAB-SMOTE SI because it oversampled more instances than the original SMOTE.

Table 6.11: Results from Amazon Employee Access Dataset

MAJ.= 30872; MIN. = 1897	**BORDER M5=1744 – 91.9% BORDER M8=1752 – 92.3%**			**I.R. = 16.274**
OVERSAMPLING TECHNIQUE	**B%**	**GEN**	**RECALL**	**F-MEASURE**
1.0: ORIGINAL	0	0	0.17	0.265
2.1: SMOTE K=8	0	1897	0.523	0.617
3.0: B-SMOTE P' & P: 5,5	100	1744	0.486	0.577
4.1: AB-SMOTE P' & P': 8,8	100	1752	0.488	0.587
5.7: CAB-SMOTE SI: 8,8,7	110	1937	0.602	0.71
6.1: CAB-SMOTE SII: 8,8,2	100	1752	0.587	0.704
7.1: CAB-SMOTE SIII: 8,8,2	100	1752	0.587	0.704
8.2: HCAB-SMOTE: 5,5,3	100	1744	**0.621**	**0.728**

Table **6.12** displays the results from using the Click Prediction dataset. Where HCAB-SMOTE generates 14% less than SMOTE and obtains a 17% increase in F-measure value. CAB-SMOTE SI gets the highest F-measure because it generates 16% instances more than SMOTE, and more than other techniques which made it logical to get the highest F-measure value.

Table 6.12: Results from Click Prediction Dataset

MAJ.= 33220; MIN. = 6728	**BORDER M5=6300 – 93.6% BORDER M8=6132 – 91.1%**			**I.R. = 4.93**
OVERSAMPLING TECHNIQUE	**B%**	**GEN**	**RECALL**	**F-MEASURE**
1.0: ORIGINAL	0	123456	0.041	0.077
2.1: SMOTE K=8	0	6728	0.419	0.575
3.3: B-SMOTE P' &N: 5,5	100	6132	0.406	0.565
4.0: AB-SMOTE P’ & P’: 5,5	100	6300	0.397	0.555
5.7: CAB-SMOTE SI: 8,8,7	127	7812	**0.563**	**0.705**
6.0: CAB-SMOTE SII: 5,5,2	100	6300	0.507	0.667
7.0: CAB-SMOTE SIII: 5,5,2	100	6300	0.507	0.667
8.4: HCAB-SMOTE: 8,8,3	94	5765	0.532	0.676

After observing the results, the thesis summed up the following summary:

- CAB-SMOTE SI oversamples all the borderline clusters to reach the size of the largest one, without oversampling within the largest cluster. Therefore, it might oversample fewer or more instances than B-SMOTE and AB-SMOTE. Its classification results outperform those techniques even while generating fewer instances. Indicating that oversampling within clusters is more efficient than oversampling within the whole danger region.
- CAB-SMOTE SII oversamples all the clusters which are larger than half of the largest cluster. It achieves better classification accuracy than CAB-SMOTE SI when generating a slightly fewer number of instances and achieves better results while generating the same number of instances. Indicating that the oversampling large clusters are more important than small clusters within the borderline region.
- CAB-SMOTE SIII oversamples all the borderline clusters, offers the same results as CAB-SMOTE SII when they generate the same number of instances, and better results when it generates more instances. Therefore, SII delivers good results

without sampling clusters smaller than half of the largest one. Which lead the thesis to use SII within HCAB-SMOTE to minimize generating instances.

- HCAB-SMOTE outperforms the other techniques due to the combination of under and oversampling processes which removes small clusters of minority borderline instances and negative borderline instances, improving the classification results from two sides at the same time.

The thesis also applied the same experiment to test the oversampling techniques using Naïve Bayes classifier which functions differently than decision tree thus delivering different F-measure values (Bunkhumpornpat, Sinapiromsaran and Lursinsap, 2012). The Naive Bayes classifier is a set of algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. The results are displayed in Table 6.13 showing the 12 datasets in the first column and each other column displays the F-measure value of the oversampling technique used for oversampling the dataset. CAB-SMOTE and HCAB-SMOTE got the highest F-measure values in most of the datasets.

Table 6.13: Naïve Bayes F-Measure Values After Applying Different Oversampling

	Orig.	**SMOTE**	**B SMOTE**	**AB SMOTE**	**CAB SMOTE SI**	**CAB SMOTE SII**	**CAB SMOTE SIII**	**HCAB SMOTE**
Abalone	0.258	0.34	**0.344**	0.339	0.258	0.307	0.307	0.313
Customer churn	0.522	0.713	0.672	**0.705**	0.61	0.621	0.673	0.675
Haberman	0.309	0.439	0.321	0.312	0.407	0.277	0.343	**0.545**
Employee Attrition	0.348	0.558	0.551	0.554	0.365	0.655	0.655	**0.656**
Teclo Customer Churn	0.606	0.758	0.706	0.71	0.752	0.704	0.71	**0.78**
Flare	0.348	0.52	0.521	0.527	0.481	0.595	0.576	**0.598**
Wine Quality	0.168	0.251	0.251	0.251	**0.378**	0.34	0.336	0.302
Sales Win-Loss	0.597	0.724	0.71	0.714	0.659	0.646	**0.851**	0.653
Bank Subscription	0.505	0.623	0.583	0.588	0.691	0.712	0.712	**0.769**
Adult	0.603	0.757	0.693	0.738	0.626	0.794	0.797	**0.808**
Amazon Emp. access	0.151	0.433	0.344	0.417	0.426	0.512	**0.6**	0.24

Click Prediction	0.109	0.601	0.505	0.579	**0.682**	0.653	0.653	0.078

For additional testing of the proposed oversampling techniques, the thesis used eleven datasets and compared F-measure values before and after applying the oversampling techniques while classifying with Random forest classifier and presented them in Table 6.14. The random forest classifier consists of a combination of tree classifiers, where each classifier is generated using a random vector sampled independently from the input vector, and each tree casts a unit vote for the most popular class to classify an input vector (Pal, 2005). The results show that the proposed oversampling method, HCAB-SMOTEE got the highest F-measures values against other techniques, while CAB-SMOTE SI got the highest F-measure values due to oversampling more than 100% of the minority intended class, therefore the results were not taken into consideration. The high F-measure value attained after classifying the dataset with random forest after being oversampled with HCAB-SMOTE implies that the technique successfully optimizes random forest classifier for imbalanced class datasets.

Table 6.14: Random Forest F-measure values after applying different oversampling

	Orig.	**SMOTE**	**B SMOTE**	**AB SMOTE**	**CAB SMOTE SI**	**CAB SMOTE SII**	**CAB SMOTE SIII**	**HCAB SMOTE**
Abalone	0.456	0.561	0.615	0.615	0.496	0.76	0.76	**0.76**
Haberman	0.306	0.679	0.648	0.582	0.604	0.539	0.577	**0.779**
Employee Attrition	0.207	0.649	0.661	0.664	0.681	0.704	0.704	**0.712**
Flare	0.036	0.451	0.504	0.532	0.495	0.671	0.648	**0.736**
Wine Quality	0	0.154	0.154	0.154	0.489	0.541	0.56	**0.604**
Adult	0.666	0.833	0.794	0.8	0.835	0.808	0.807	**0.838**
Bank	0.509	0.756	0.733	0.727	0.731	0.751	0.751	**0.819**
Amazon Employee access	0.455	0.66	0.675	0.638	0.758	0.747	0.747	**0.778**
Click Prediction	0.186	0.644	0.677	0.63	**0.718**	0.677	0.677	**0.686**
customer churn	0.036	**0.794**	0.793	0.682	0.521	0.519	0.613	0.61
Telco Customer churn	0.036	0.821	0.809	0.762	**0.831**	0.689	0.724	**0.826**

For obtaining more credibility about the proposed methods, the thesis compared the

classification F-measure values attained from the proposed oversampling methods with other researches that applied the same oversampling experiment with their own proposed techniques and displayed them in Table 6.15. The thesis compared the work with Safe SMOTE (Bunkhumpornpat, Sinapiromsaran and Lursinsap, 2009), Density-Based SMOTE (Bunkhumpornpat, Sinapiromsaran and Lursinsap, 2012), Local neighborhood extension of SMOTE (MacIejewski and Stefanowski, 2011), and CURE-SMOTE (Ma and Fan, 2017). Safe SMOTE cautiously generates new positive instances inside the safe level area of the minority class, which is called a safe level. This safe level is computed by using nearest neighbor minority instances found around a given positive instance. It counts how many positive instances are found around a given one and categorize the given instance as noise if it did not have any positive instance around it (safe level equal to zero) and it categorizes it as safe if the safe level is equal to k nearest neighbors (all the KNN are from minority class). Safe SMOTE uses a safe level ratio to select the location of the generated instance. Safe SMOTE used decision tree classifiers over the Haberman dataset in their experiment, but they did not post the F-measure values so the thesis used results found in Density-Based SMOTE where they also applied the same decision tree classifier and compared their work with Safe SMOTE. Density-Based SMOTE that uses a density-based concept in creating clusters and oversample new instances in a randomly shaped cluster that is computed by DB-SCAN. This approach creates new instances over the shortest path from a positive instance to a pseudo centroid of the minority class cluster. Subsequently, the newly generated instances are gathered and compressed near the pseudo centroid and are fewer whenever they are further away from this centroid by combining DBCAN and SMOTE. Density-Based SMOTE is also a modified version of Borderline SMOTE, where it oversamples in specific regions around the borderline area, but extends over Borderline SMOTE in that it oversamples in safe area inside the minority class to increase the oversampling rate for the whole region, and manage to precisely over samples this region to maintain the majority class detection rate. Density-Based SMOTE used seven datasets to evaluate their work, these datasets are Pima Indians Diabetes, Haberman's survival, Glass Identification, Image Segmentation, Land Satellite image, E. coli, and Yeast datasets. The thesis only used Pima and Haberman datasets, because Density-Based SMOTE work modified the

other dataset in an unknown system to transform the multi class dataset to a binary class one. By doing so the thesis could not compare the proposed HCAB-SMOTE with other oversampling methods on different datasets, the datasets should be identical to get good comparison results. Density-Based SMOTE used decision tree classifier and other classifiers while applying cross-validation and evaluated their work with F-measure values to compare it with other methods, this F-measure value has to be closer to one which reflects that both recall and precision are also closer to one, which is of relevance to minority classes. HCAB-SMOTE results overcame Density-Based ones with fewer generated instances. Local Neighborhood SMOTE (LN-SMOTE) exploits more information about the local neighborhood of the minority class to generate new instances within the class. It is directly related to Safe SMOTE where they both take into consideration the presence of majority examples within the KNN by the safe level special coefficient as explained before. Where two instances are picked from the minority class and their safe levels are calculated SL(p1) and SL(p2), after that, the algorithm LN-SMOTE compute their safe level ratio which is SL(p1)/SL(p2). This algorithm also categorizes the minority instances as Noisy or safe under five different criteria. If the safe level of both instances is equal to zero, then they are treated as a noisy outlier and no oversampling will be made with these instances, whereas if the instance p1 safe level is higher than zero and the other instance level is equal to zero, then the algorithm will duplicate p1 instance. In the case of the safe level ratio is equal to one, which means that p1 and p2 have the same safe level value, the new oversampled instance will be created within the line joining both instances in the same way as in SMOTE. In the case of the safe level ratio is larger than one, p1 safe level higher than p2 level, the new instance will be created closer to p1 by modifying the gap number in SMOTE which allows the change of the distance of the newly generated instance. Whereas if the safe level ratio is less than one, p2 value is larger than p2, the newly generated instance will be created closer to p2. This algorithm oversamples new instances in the same number as SMOTE do minus when both minority instances having a safe level equal to zero when there will be no oversampling. LN-SMOTE used the decision tree algorithm and Naïve Bayes to classify the datasets that were used to compare their results.

Because each work used different software and method to evaluate their proposed

technique, the thesis had to come up with a system to evaluate its work against others in an unbiased manner, so it calculated the percentage of how much the oversampling results increased the F-measure values from the original F-measure value the dataset attained without oversampling and compared it with the other methods. As an example, if the original F-measure value is 0.331 without any oversampling, and SMOTE attained 0.643, the thesis divided SMOTE result over original results and subtract that number by one, it gets the percentage increase in the value (94.26). The thesis used 8 datasets for the comparison. Some of the datasets had more than two classes, but the researchers kept the smallest class and combined the other classes into one class, converted the dataset from multi-class to a binary class one. The thesis uses Haberman and Flare datasets which already conducted the formerly experiment on, and then the thesis added Pima Diabetes dataset, that diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset where all the samples are females are at least 21 years old of Pima Indian heritage. German credit data set that consists of entries representing the individuals who take credit from the bank, and each one of them is classified as good or bad credit risks according to the set of attributes. Blood transfusion dataset that has information about 748 donors at random from the donor database to classify whether the donor donated blood in March 2007. The breast cancer dataset consisted of a digitized image of a fine needle aspirate of a breast mass. A balance scale dataset was generated to model psychological experimental results to classify the balance scale tip to the right, tip to the left, or be balanced. And the Cleveland dataset was used to classify the presence of heart disease in the patient. The table shows that HCAB-SMOTE obtained the highest F-measure percentage increase against other results, but for the balance scale dataset, CAB-SMOTE got the highest F-measure value followed by HCAB-SMOTE and CURE-SMOTE. The dashes '– 'are included in the table because the results were not found in the literature on hands as the thesis collected from the different articles and each article made an experiment on a specific number of datasets.

Table 6.15: F-Measure Comparison Between the Proposed and Other Oversampling Methods Applied in Literature

	Haberman	Flare	Pima Diabetes	German Credit	Blood Transfusion	Breast cancer	Balance Scale	Cleveland
Original	0.331	0	0.594	0.506	0.444	0.917	0	0.179
SMOTE	94.26%	47.00%	31.99%	47.23%	38.00%	4.23%	0.00%	189.94%
B-SMOTE	76.13%	43.40%	21.04%	40.12%	19.26%	4.25%	0.00%	189.94%
AB-SMOTE	75.53%	42.10%	22.39%	42.89%	31.31%	3.16%	0.00%	189.94%
SAFE SMOTE (Bunkhumpornpat, Sinapiromsaran and Lursinsap, 2009)	45.23%	3.02%	47.25%	9.91%	28.07%	0.24%	28.07%	34.94%
DBSMOTE (Bunkhumpornpat, Sinapiromsaran and Lursinsap, 2012)	126.82%	-	45.92%	-	-	-	-	-
C-SMOTE (Ma and Fan, 2017)	38.09%	-	-	-	14.16%	0.05%	14.16%	-
K-mean SMOTE	42.88%	-	-	-	18.47%	-0.39%	18.47%	-
Cure SMOTE (Ma and Fan, 2017)	52.49%	-	-	-	53.69%	0.26%	53.69%	-
Random sampling	6.86%	-	-	-	11.23%	-0.37%	11.23%	-
LNSMOTE I (MacIejewski and Stefanowski, 2011)	46.63%	13.51%	4.96%	11.87%	-	4.04%	-	51.74%
LNSMOTE II (MacIejewski and Stefanowski, 2011)	40.14%	18.82%	4.85%	10.88%	-	4.58%	-	53.97%
CAB-SMOTE	103.32%	60.10%	25.42%	43.68%	52.03%	3.71%	**56.30%**	264.25%
HCAB-SMOTE	**129.91%**	**70.90%**	**47.64%**	**55.34%**	**83.78%**	**4.80%**	54.80%	**204.47%**

The thesis tried to use more datasets which the other articles worked on, but after attaining the datasets online it appeared that the other articles added or modified the dataset in an unknown way to transform them from multi-class datasets into binary class datasets (data sets containing two classes), so it was better not to use them because they had a different number of instances within minority and majority classes than the original one. Even after trying to analyze how they transformed the datasets; the thesis was successful in transforming two datasets from multi-class into binary class while having the same number of samples within the minority and majority classes of the other research. When the original dataset classification F-measure

obtained a zero value, the thesis substituted the value of the increased classification percentage by the F-measure value 0.47 to 47%.

Figure 6.1 and Figure 6.2 displays the recall and F-measure values against the percentage of Imbalanced ratio found in the tested datasets. Imbalanced ratios which are calculated by dividing the number of majority instances over the number of minority instances. The graph displays different I.R values of each original dataset within the experiment on the horizontal axis and then displays on the vertical axis the Recall and F-measure values after applying the decision tree algorithm on the original data, and then after oversampling with different oversampling techniques. The dataset that has the most of the imbalanced ratio problem is the one that has the highest I.R. with a value of 29.17 related to the wine quality dataset. These graphs show that when the I.R is large, the recall and F-measure values for classifying minority class are small, and these values increase after applying oversampling techniques over the dataset. HCAB-SMOTE outperformed other oversampling techniques through most of the datasets.

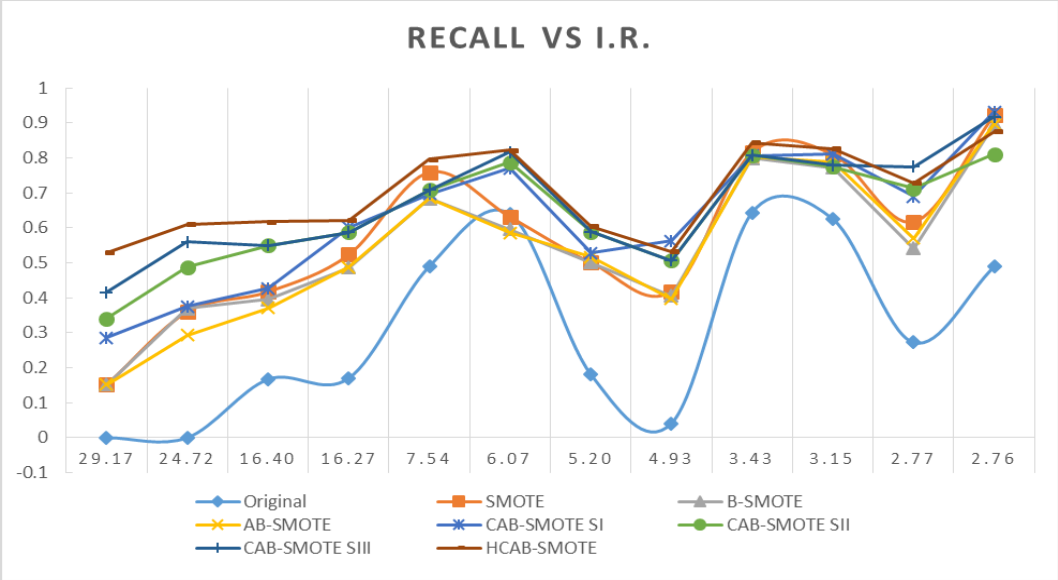


Figure 6.1: Recall V-Axis Against the Datasets Imbalanced Ratios H-Axis

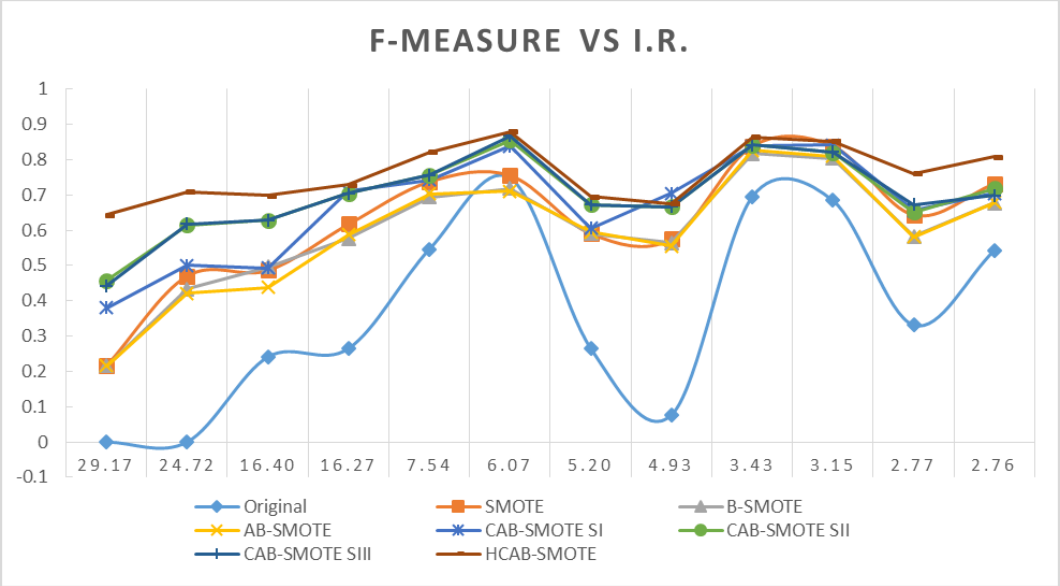


Figure 6.2: F-measure V-Axis Against the Datasets Imbalanced Ratios H-Axis

Figure 6.3 and Figure 6.4 display the Recall and F-measure values against the border ratio percentage from the minority class. The border ratio is calculated by dividing the number of instances found in the border area at Mnn=5 over the number of instances found in the minority class as shown in Equation 6.3. When the Border ratio is equal to one or 100% that means all the minority class is considered as a border. Whereas when the border ratio is more than 90%, the original data get very low recall and F-measure values, which is because more than 90% of the minority instances have more than 3 negative instances within their 5 nearest neighbors, thus having too many noisy instances where the classifier could not build up a good model to predict the minority instances. This issue is improved by oversampling which adds new positive instances into the minority class, increasing their density, therefore enabling the classifier to recognize the minority instances better and deliver higher classification accuracy. The two graphs presented in Figure 6.3 and Figure 6.4 look similar because the F-measure reflects the harmonic mean between recall and precision.

$$\text{Border Ratio} = \frac{\text{Border instances}}{\text{Minority instances}} \tag{6.3}$$

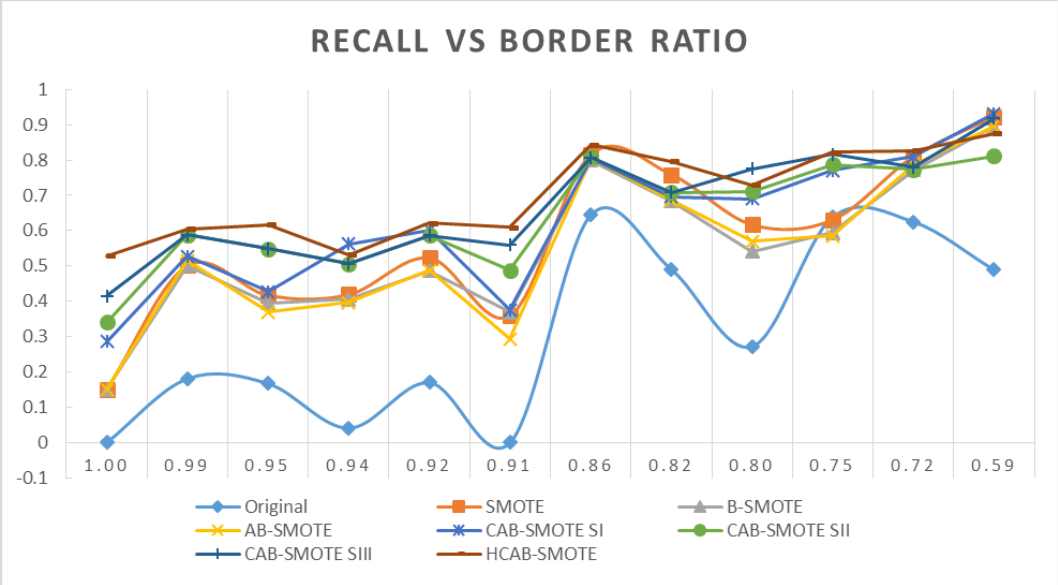


Figure 6.3: Recall $_{\text{V-Axis}}$ Against the Border Ratio $_{\text{H-Axis}}$ Computed in The Dataset.

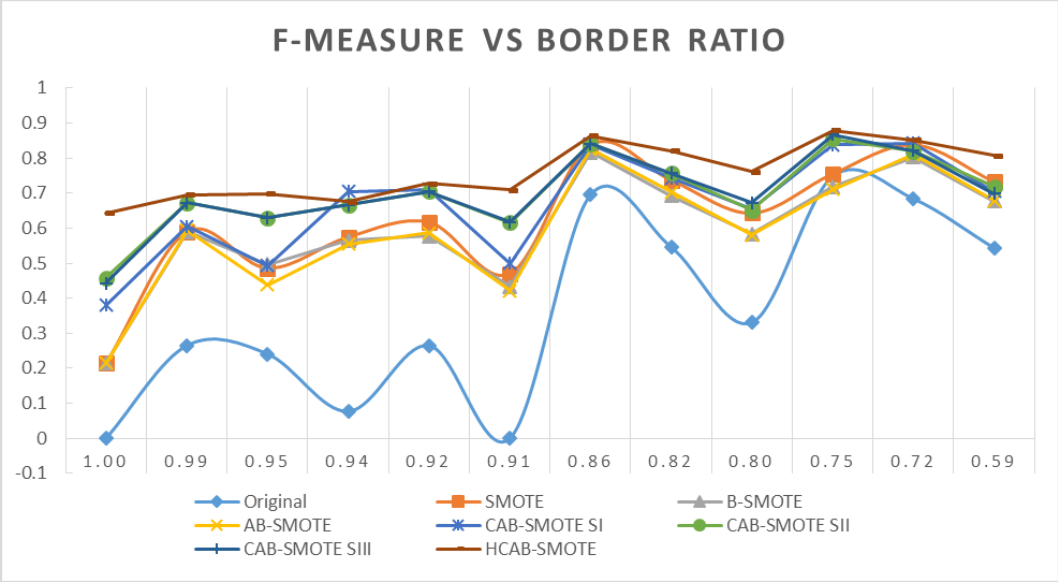


Figure 6.4: F-Measure V-Axis Against the Border Ratio H-Axis.

Figure 6.5 displays the graph showing the newly generated instances over the minority instances ratio against I.R values of the datasets. SMOTE has a value of 1, because it doubles the number of minority instances, while CAB-SMOTE SI sometimes oversamples more than the number of minority instances where it achieved a value of 1.16 due to the differences between clusters size because it clusters the borderline area to different clusters and oversample within small clusters up until each cluster reach the largest cluster number of instances. There is no clear trend in this graph, but it is noticeable that HCAB-SMOTE oversamples less than other techniques. Because the experiment intergraded CAB-SMOTE SII into HCAB-SMOTE and the method tuning perimeters with the best F-measure are picked up and chosen. It is noticeable that HCAB-SMOTE and CAB-SMOTE SII do not always oversample the same number of instances, because different parameters of HCAB-SMOTE with its undersampling process delivered higher F-measure values than that of CAB-SMOTE SII. The same things go for B-SMOTE and AB-SMOTE where they should generate the same number of instances if they had the same tuning parameters, but each method delivered different F-measures with different parameters.

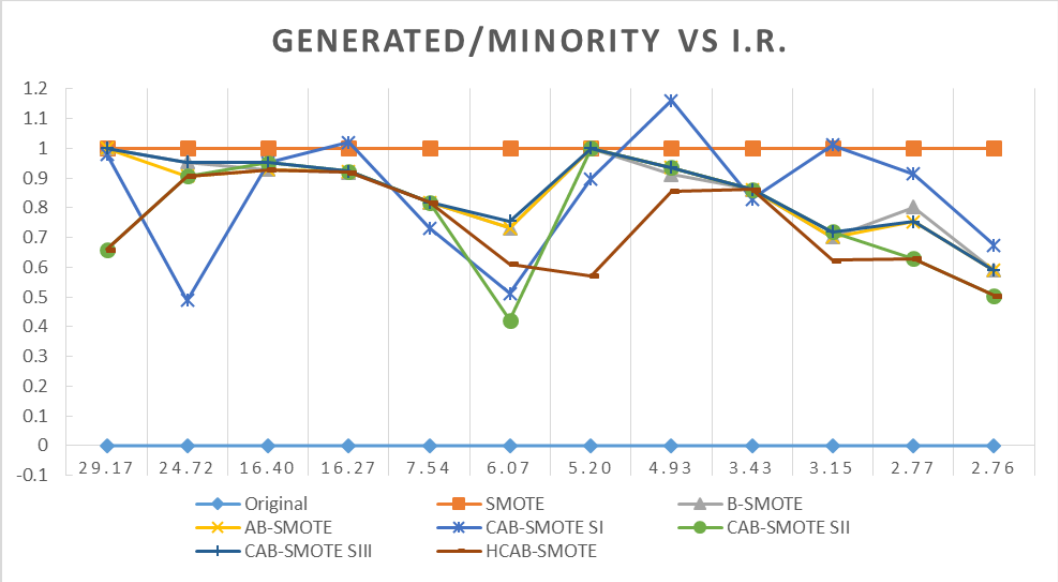


Figure 6.5: Generated to Minority Instances Ratio V-Axis Vs. I.R. Ratio H-Axis.

Figure 6.6 shows the relation between the ratios of the border to the minority against the I.R of each dataset. It indicates that datasets chosen in the experiment that has high I.R. tend to have a border that covers more than 90% of the minority class, and when the I.R. decreases, the border shrinks thus enabling the border-related techniques to generates fewer instances.

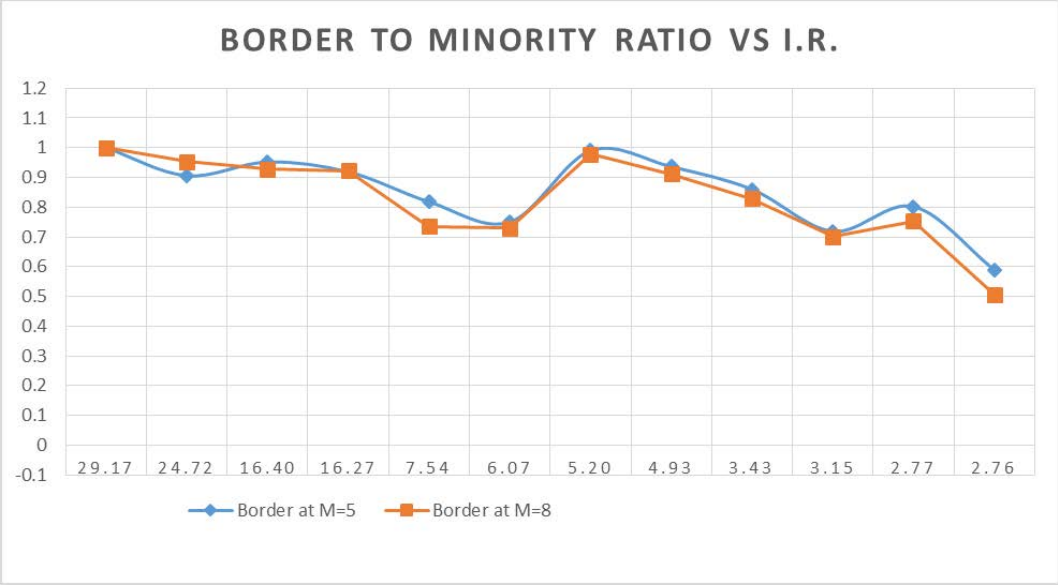


Figure 6.6: Border Ratio V-Axis to I.R. H-Axis In The Dataset.

To track the effects of increasing the oversampling rates on the oversampling techniques applied on the datasets, the thesis applies the oversampling techniques on the existing datasets with different oversampling rates and records the F-measure values. The thesis chose the versions that give the highest F-measure value within the oversampling methods and apply 50, 100, 200, 300, and 400 % oversampling rates. After each run, the thesis records the F-measure values and gathers them in one graph for each dataset. The x-axis shows the value of the oversampling rate, while the y-axis displays the F-measure values which are attained from each oversampling method at a given oversampling rate. Approach number 5 that refers to CAB-SMOTE SI is not included with this experiment because it oversamples the instances inside the clusters to reach the same number of instances found in the largest cluster, thus the number of oversampling instances is limited to the difference in the number of instances found between each cluster and the number of instances found in the largest cluster.

Figure 6.7 shows that when applying a 50% oversampling rate, SMOTE overcomes the other techniques and gives higher F-measure value than other oversampling methods. After that when the oversampling rate is 100%, HCAB-SMOTE took over. After 200% oversampling rate the CAB-SMOTE SIII got the highest F-measure value. The reason behind that is the properties of the dataset, and the effect of the generated instances location, that is there are a lot of different variations of SMOTE, while each one creates new instances in a different system and location, and each method delivers different classification results. When the oversampling rate increases the f-measure value increase till it reaches a certain point and then it stops increasing showing the maximum capacity of the classification algorithm to classify the intended instances.

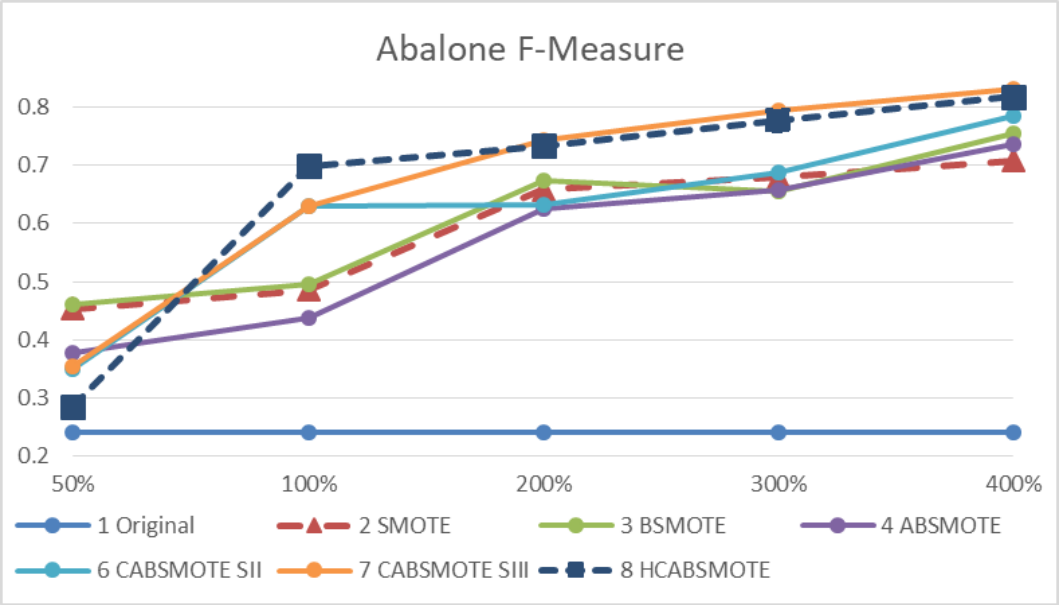


Figure 6.7: Results of Increasing Oversampling for Abalone Dataset

Figure 6.8 indicates that HCAB-SMOTE leads to supply the highest F-measure value throughout all the oversampling rates followed by CAB-SMOTE SIII and SII.

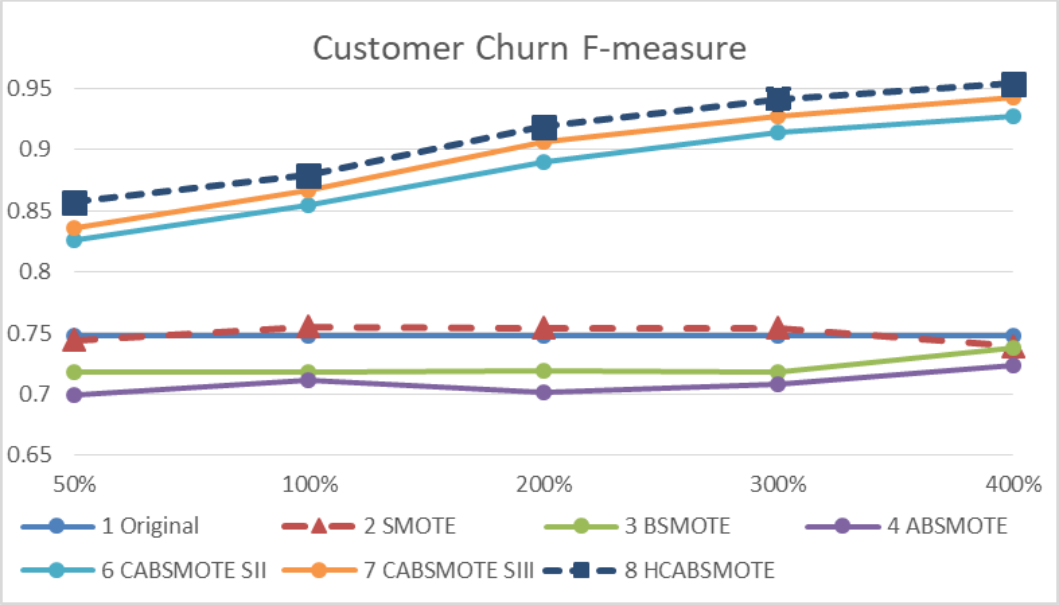


Figure 6.8: Results of Increasing Oversampling for Customer Churn Dataset

Figure 6.9 shows that HCAB-SMOTE overcomes over all the other techniques throughout the different oversampling rates. AB-SMOTE delivered lower f-measure value when it oversampled 50% of the minority instances.

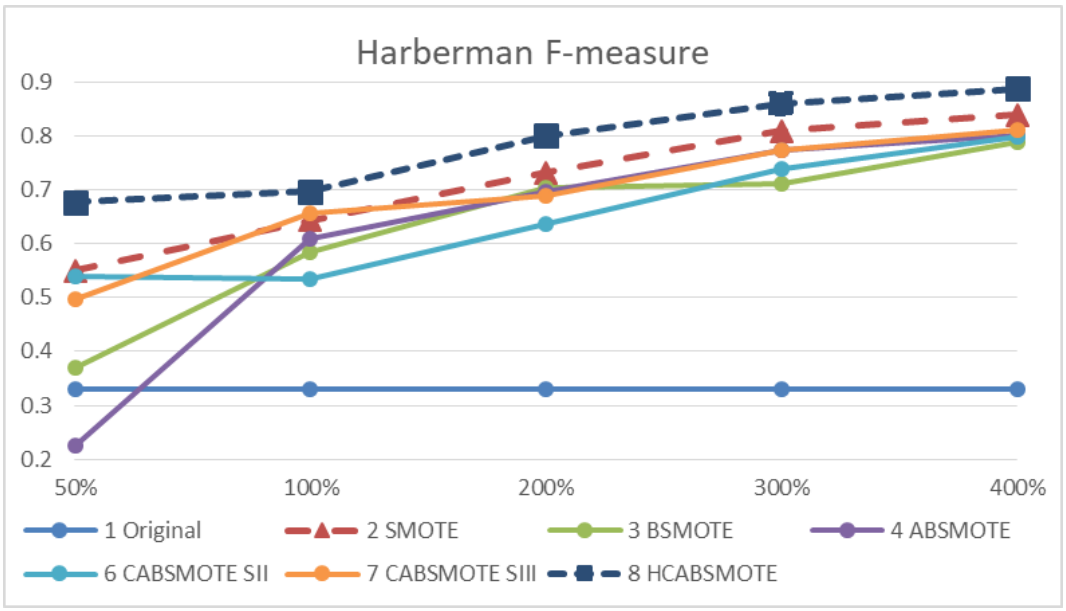


Figure 6.9: Results of Increasing Oversampling for Haberman Dataset

Figure 6.10 shows that HCAB-SMOTE gave the highest F-measure throughout all the oversampling rates, except at 100% where CAB-SMOTE SIII's results were overcome HCAB-SMOTE. Which shows us that small minority borderline cluster might be important in some datasets because of SIII oversamples within all the border clusters. while CAB-SMOTE SII which is used in HCAB-SMOTE oversamples only within clusters that have a number of instances less than half of the instances found in the largest cluster. The thesis chose CAB-SMOTE SII instead of CAB-SMOTE SIII to be in HCAB-SMOTE because it was aiming to get the highest classification accuracy with fewer newly generated instances, which was delivered by CAB-SMOTE SII.

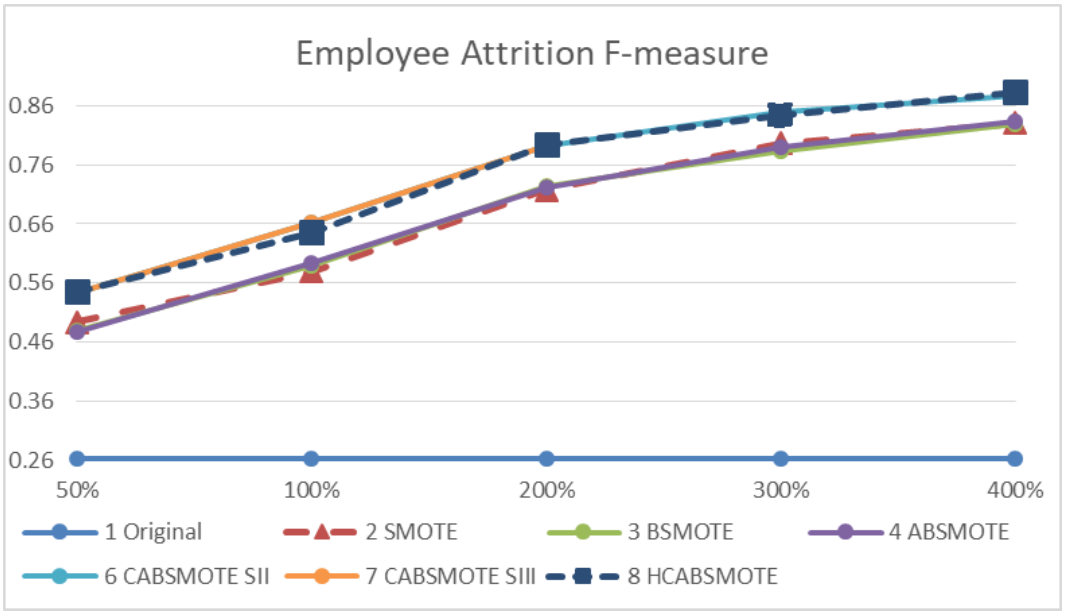


Figure 6.10: Results of Increasing Oversampling for Employee Attrition Dataset

Figure 6.11 and Figure 6.12 present the graphs where HCAB-SMOTE results overcome other oversampling techniques F-measure values through applying different oversampling rates, and they also seem like that the results improved in an equal pace as the oversampling rate increase. When the oversampling was less than 100%, the F-measure values were lower than the value of the original dataset without oversampling.

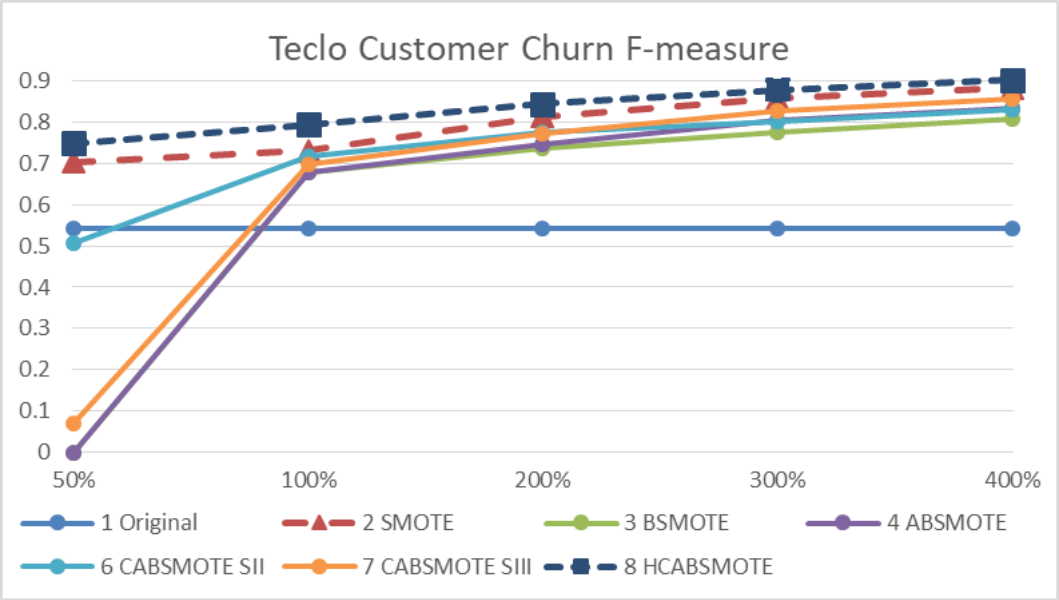


Figure 6.11: Results of Increasing Oversampling for Telco C.C. Dataset

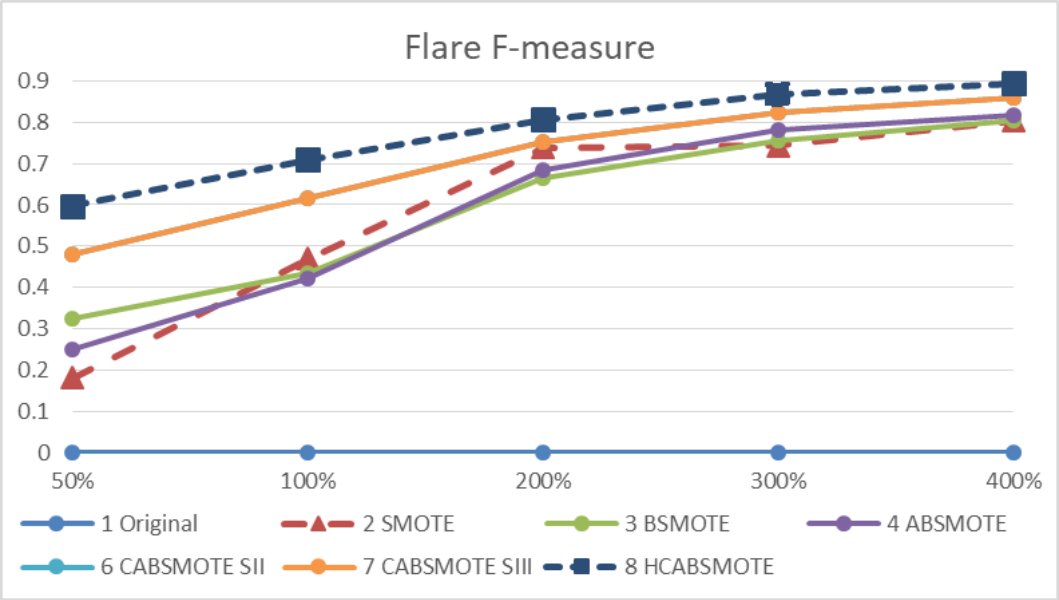


Figure 6.12: Results of Increasing Oversampling for Flare Dataset

Figure 6.13 shows that HCAB-SMOTE F-measure values outperform other oversampling results. The results of SMOTE, B-SMOTE, and AB-SMOTE deliver the same F-measure results throughout all the oversampling rates because all the minority sample was considered as a danger area as seen in Table 6.7 that the border covers 100% of the instances belonging to the minority class, that is why their lines are overlapped over each other in the graph in Table 6.13. The thesis also implies that the effect of the undersampling done in HCAB-SMOTE can be measured by the deference between CAB-SMOTE SII and HCAB-SMOTE F-measure results.

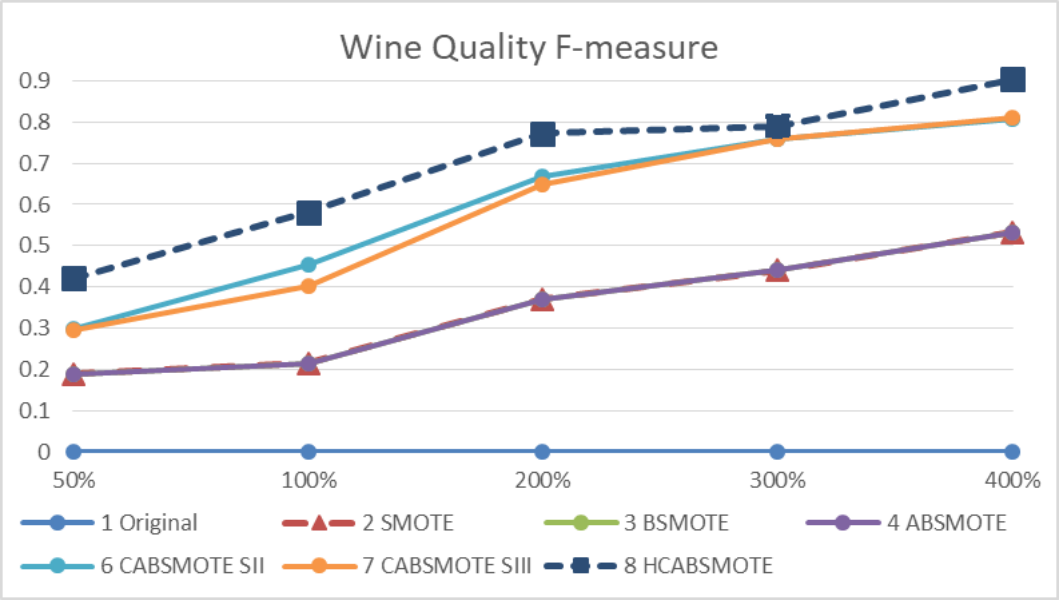


Figure 6.13: Results of Increasing Oversampling for Wine Quality Dataset

Figure 6.14 also shows that HCAB-SMOTE outperforms other oversampling results, but B-SMOTE and AB-SMOTE gave F-measure values less than SMOTE throughout different oversampling rates. Each dataset has its independent properties, where they cannot give the exact classification results nor the oversampling techniques could improve the classification results for all the datasets in the same manner. But the oversampling techniques might deliver the same performance over datasets that have the same or similar properties.

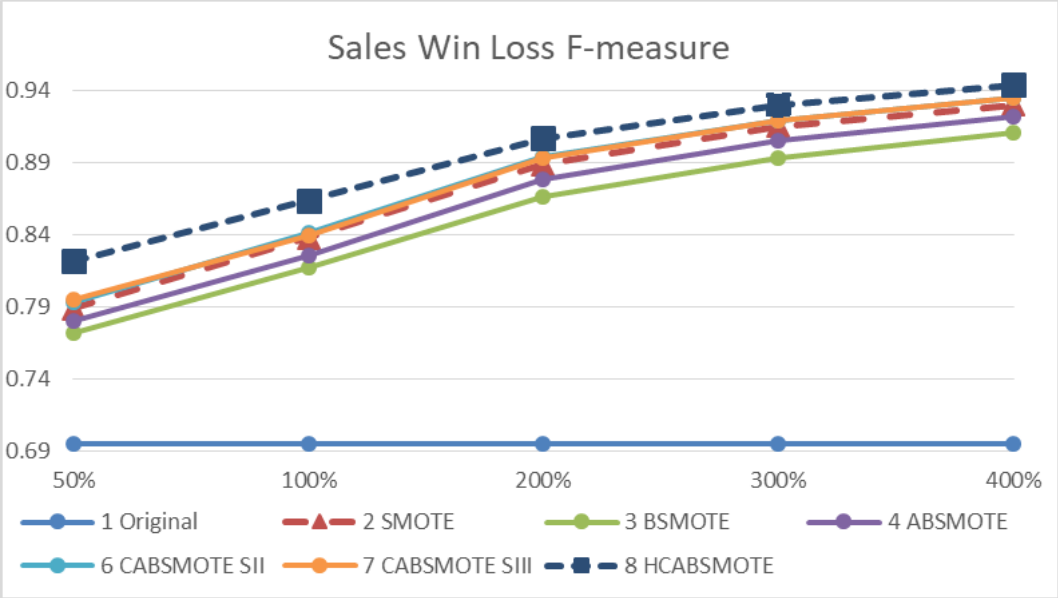


Figure 6.14: Results of Increasing Oversampling for Sales Dataset

Figure 6.15 and Figure 6.16 display the graphs that show the supremacy of HCAB-SMOTE over other oversampling techniques, while B-SMOTE and AB-SMOTE gave results less than SMOTE throughout the oversampling rates.

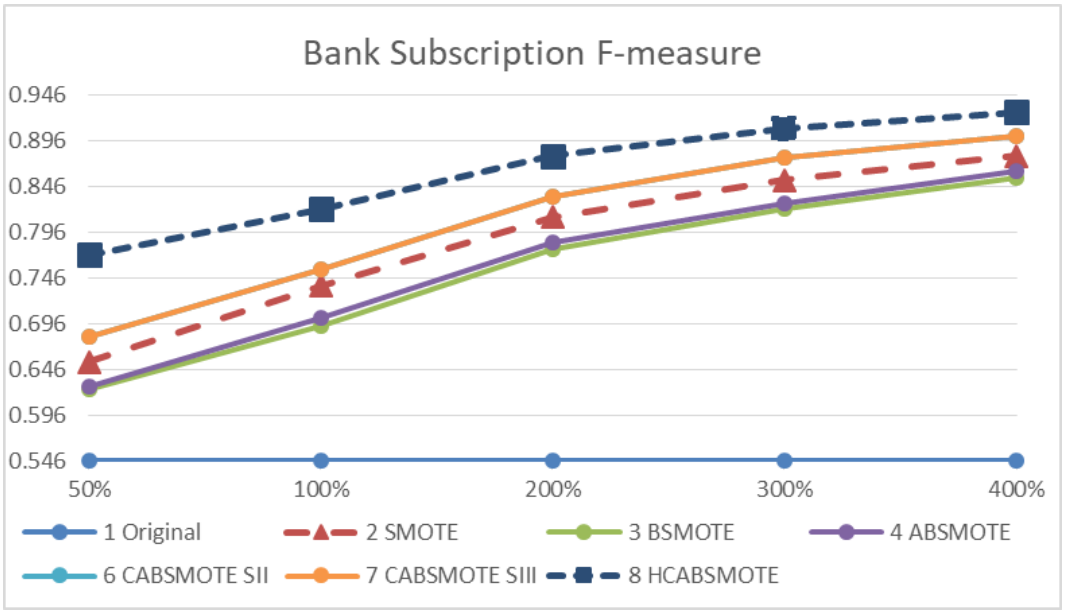


Figure 6.15: Results of Increasing Oversampling for Bank Dataset

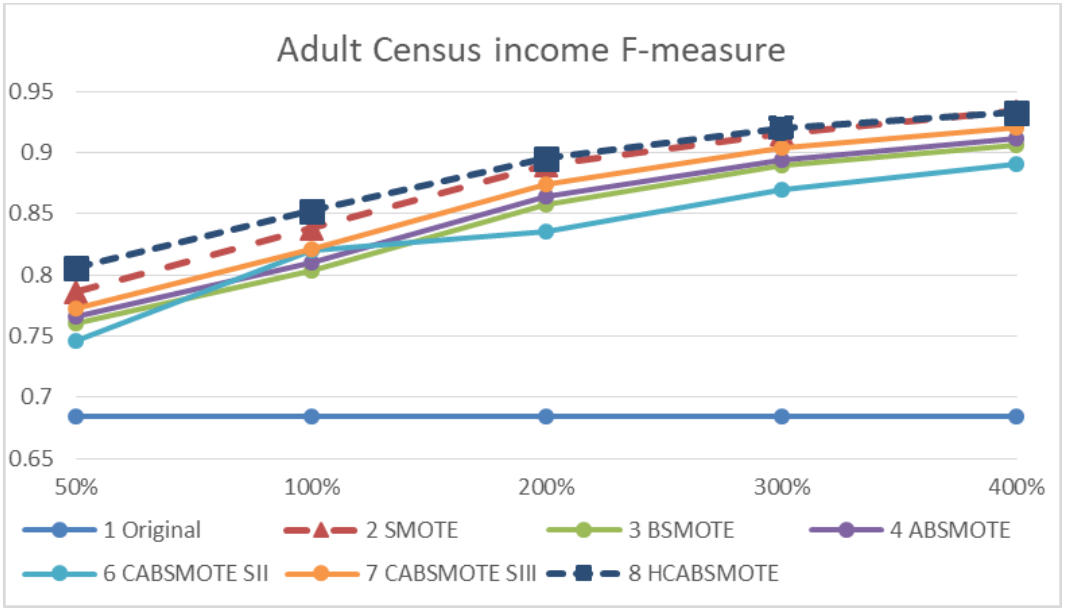


Figure 6.16: Results of Increasing Oversampling for The Adult Dataset

Figure 6.17 and Figure 6.18 shows that HCAB-SMOTE F-measure values overcome other oversampling F-measure values throughout different oversampling rates. Where in the Click Prediction dataset, CAB-SMOTE SII, SII, and HCAB-SMOTE deliver the same F-measure values throughput the different oversampling rates.

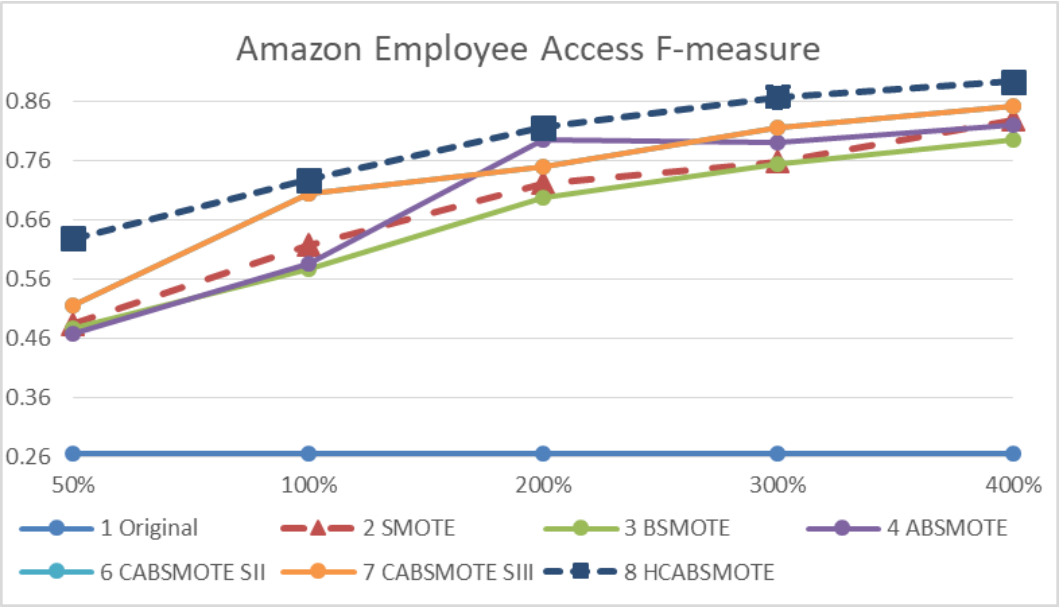


Figure 6.17: Results of Increasing Oversampling for Amazon E.A. Dataset

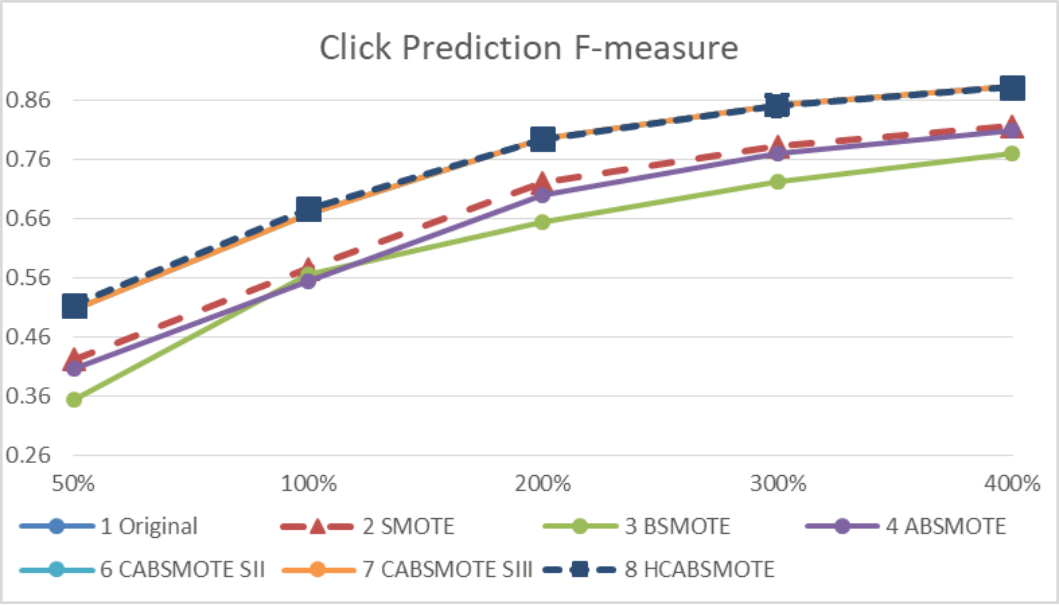


Figure 6.18: Results of Increasing Oversampling for Click Prediction Dataset

Figures 6.7 to 6.18 Displays the F-measure values obtained after applying different oversampling methods with different oversampling rates starting from 50% up to 400%. These graphs gave a clear indication that all the oversampling techniques provide better classification accuracy results when increasing the oversampling rate. It was also noticeable that the F-measure value increases up to a point where it stops increasing, even after oversampling more instances into the datasets. This was visible by decreasing the slope of the line connecting the oversampling F-measure values. In other words, the increased number of instances within the minority class allowed the classification algorithm to better learn and generalize the model to better classify minority positive instances. CAB-SMOTE and HCAB-SMOTE managed to provide the highest classification accuracy results among the original oversampling techniques, verifying that oversampling special areas within the boundaries of the minority borderline is better than increasing the instances density inside all the border region as a whole or around it. Also, each oversampling technique sustained its F-measure ranking between other oversampling techniques throughout the increasing of the oversampling rate.

CHAPTER 7
CONCLUSION

The importance of the reliability and classification accuracy within machine learning models, drive the researchers to consistently develop new ideas and methods to achieve and optimize its process. Machine learning needs datasets to learn from and create models. These datasets have to contain instances divided into equal classes to enable machine learning to create models that classify new unlabeled samples with high accuracy. But it was noticeable that most of the datasets have imbalanced classes problem, where it was hard to gather information or labeled instances which belong to the intended class. Therefore, a lot of datasets have an imbalanced class problem which did not allow the classification algorithm to create good models. For this reason, scholars proposed different methods to increase the classification accuracy for imbalanced class datasets, but most of these methods had to generate a large number of new instances -sometimes, more than 5 times of the minority class- to reach their aim. For this reason, the thesis focused on the oversampling technique SMOTE, and B-SMOTE and optimized them. This thesis proposed AB-SMOTE that focuses on limiting the oversampling region to be within the borderline area, then the thesis developed CAB-SMOTE that cluster the borderline region and oversample each cluster independently, to finally propose HCAB-SMOTE that combine oversampling with undersampling to minimize the number of generated instances while increasing the classification accuracy by creating new instances based on border region cluster sizes using CAB-SMOTE SII, and applying the undersampling technique to remove noisy minority and majority instances from the border area. These oversampling methods were evaluated by measuring the accuracy of classifying the minority instances to be in their actual class, using F-measure values and focusing on it, because it reflects the performance of recall and precision in one measurement. On one hand, the experimental results show that CAB-SMOTE SII can perform efficiently if the user does not want to under-sample the data, while on the other hand if the user did not mind using undersampling methods, the user can use HCAB-SMOTE which will boost the classification accuracy for the minority class with same or fewer generated instances than CAB-SMOTE SII. This thesis optimizes the preprocessing technique to

increase the classification accuracy of classifying instances that fall in minority class in the imbalanced datasets. This optimization of the oversampling technique can be reflected over different fields of business, medical or other applications -each dataset used in the experiment can be a business problem that the machine learning solves- where it will help to increase the reliability of the models created and use them to decrease the financial costs, speed up operations, focus marketing campaigns, increase the revenue, and much more. Machine learning that delivers high classification accuracy assists in marketing, sales forecasting, facilitates accurate medical diagnosing, detects spams and frauds, increases the predictive maintenance in the manufacturing field which leads to cost savings over former timely periodic methods because tasks are performed only when warranted.

Future work will focus on using density-based approaches for clustering the minority borderlines rather than using k-mean clustering. Because some datasets have no clear borders where instances from each class coincide and are found in the same region, thus need a more advanced and complex form of clustering. Another point the future work which can be focused on is reading the dataset beforehand, to analyze its properties to optimize the oversampling input parameters, and predict the best values for KNN, Mnn, K-mean clustering, and the oversampling rate value that will provide the highest classification accuracy for predicting the minority class instances.

REFERENCES

Alberto Fernandez *et al.* (2018) 'SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary', *Journal of Artificial Intelligence Research*, 61, pp. 863–905. Available at: https://www.jair.org/index.php/jair/article/view/11192.

Andrew, W., Greatwood, C. and Burghardt, T. (2017) 'Visual Localisation and Individual Identification of Holstein Friesian Cattle via Deep Learning', *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017*, 2018-Janua, pp. 2850–2859. doi: 10.1109/ICCVW.2017.336.

Bach, M. *et al.* (2017) 'The study of under- and over-sampling methods' utility in analysis of highly imbalanced data on osteoporosis', *Information Sciences*, 384, pp. 174–190. doi: 10.1016/j.ins.2016.09.038.

Batista, G. E. A. P. A., Prati, R. C. and Monard, M. C. (2004) 'A study of the behavior of several methods for balancing machine learning training data', *ACM SIGKDD Explorations Newsletter*, 6(1), pp. 20–29. doi: 10.1145/1007730.1007735.

Becker, R. K. and B. (1996) *Adult Census Income*. Available at: http://archive.ics.uci.edu/ml/datasets/Adult.

Bekkar, M. *et al.* (2013) 'Imbalanced Data Learning Approaches Review.', … *Data Mining & Knowledge …*, 3(4), pp. 15–33. doi: 10.5121/ijdkp.2013.3402.

Bouché, P., Lejeune, P. and Vermeulen, C. (2012) 'How to count elephants in West African savannahs? synthesis and comparison of main gamecount methods', *Biotechnology, Agronomy and Society and Environment*, 16(1), pp. 77–91.

Bunkhumpornpat, C., Sinapiromsaran, K. and Lursinsap, C. (2009) 'Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5476 LNAI, pp. 475–482. doi: 10.1007/978-3-642-01307-2_43.

Bunkhumpornpat, C., Sinapiromsaran, K. and Lursinsap, C. (2012) ‘DBSMOTE: Density-based synthetic minority over-sampling technique’, *Applied Intelligence*, 36(3), pp. 664–684. doi: 10.1007/s10489-011-0287-y.

Cervantes, J. *et al.* (2017) ‘PSO-based method for SVM classification on skewed data sets’, *Neurocomputing*. Elsevier, 228(October 2016), pp. 187–197. doi: 10.1016/j.neucom.2016.10.041.

Challenge, K. A. E. A. (2013) *No Title*. Available at: https://www.kaggle.com/c/amazon-employee-access-challenge.

Chawla, N. V. *et al.* (2002) ‘SMOTE: Synthetic minority over-sampling technique’, *Journal of Artificial Intelligence Research*, 16, pp. 321–357. doi: 10.1613/jair.953.

Chawla, N. V. *et al.* (2003) ‘SMOTEBoost: Improving Prediction of the Minority Class in Boosting’, pp. 107–119. doi: 10.1007/978-3-540-39804-2_12.

Chawla, N. V, Japkowicz, N. and Drive, P. (2004) ‘Editorial : Special Issue on Learning from Imbalanced Data Sets’, *ACM SIGKDD Explorations Newsletter*, 6(1), pp. 1–6. doi: http://doi.acm.org/10.1145/1007730.1007733.

Cohen, G. *et al.* (2006) ‘Learning from imbalanced data in surveillance of nosocomial infection’, *Artificial Intelligence in Medicine*, 37(1), pp. 7–18. doi: 10.1016/j.artmed.2005.03.002.

Crowd Analytix (2012). Available at: http://www.crowdanalytix.com/contests/why-customer-churn/ (Accessed: 21 August 2019).

Cup, K. (2012) *No Title*. Available at: https://www.openml.org/d/1220.

Dash, M. and Liu, H. (1997) ‘Feature selection for classification’, *Intelligent Data Analysis*, 1(3), pp. 131–156. doi: 10.3233/IDA-1997-1302.

Douzas, G., Bacao, F. and Last, F. (2018) ‘Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE’, *Information Sciences*. Elsevier Inc., 465, pp. 1–20. doi: 10.1016/j.ins.2018.06.056.

Dua, Dheeru and Graff, C. (2017) *{UCI} Machine Learning Repository*. Available at:

http://archive.ics.uci.edu/ml (Accessed: 21 August 2019).

Fernández, A. *et al.* (2013) ‘Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches’, *Knowledge-Based Systems*, 42, pp. 97–110. doi: 10.1016/j.knosys.2013.01.018.

García, S. *et al.* (2015) *Instance selection*, *Intelligent Systems Reference Library*. doi: 10.1007/978-3-319-10247-4_8.

Gupta, D., Gupta, R. and Khobragade, P. (2016) ‘Class Imbalance Problem in Data Mining using Probabilistic Approach’, 2, pp. 1–5.

Haberman, S. J. (1976) ‘Generalized Residuals for Log-Linear Models’, *Proceedings of the 9th International Biometrics Conference*, pp. 104–122.

Han, H., Wang, W. and Mao, B. (2005) ‘Borderline-SMOTE : A New Over-Sampling Method in’, pp. 878–887.

IBM & Kaggle Employee Attrition (2018). Available at: https://www.kaggle.com/patelprashant/employee-attrition#WA_Fn-UseC_-HR-Employee-Attrition.csv (Accessed: 21 August 2019).

IBM Analytic, Win Loss (2017). Available at: https://github.com/vkrit/data-science-class/blob/master/WA_Fn-UseC_-Sales-Win-Loss.csv (Accessed: 21 August 2019).

IBM Analytics Telco Customer Churn Dataset (2018). Available at: https://www.kaggle.com/blastchar/telco-customer-churn#WA_Fn-UseC_-Telco-Customer-Churn.csv (Accessed: 21 August 2019).

Japkowicz, N. (2000) ‘Learning from Imbalanced Data Sets: A Comparison of Various Strategies’, *AAAI workshop on learning from imbalanced data sets*, 68, pp. 10–15. doi: 10.1109/TKDE.2008.239.

Keel Datasets, Abalone9-18 (1995). Available at: http://sci2s.ugr.es/keel/dataset.php?cod=116 (Accessed: 21 August 2019).

Keel Datasets, Solar Flare (1989). Available at: https://sci2s.ugr.es/keel/dataset.php?cod=1331#sub1 (Accessed: 21 August 2019).

Keel Datasets, Wine Quality (2009). Available at: https://sci2s.ugr.es/keel/dataset.php?cod=1322 (Accessed: 21 August 2019).

Kellenberger, B., Marcos, D. and Tuia, D. (2018) 'Detecting mammals in UAV images: Best practices to address a substantially imbalanced dataset with deep learning', *Remote Sensing of Environment*. Elsevier, 216(June), pp. 139–153. doi: 10.1016/j.rse.2018.06.028.

Kitsios, F. and Kamariotou, M. (2016a) 'Decision support systems and business strategy: A conceptual framework for strategic information systems planning', *2016 6th International Conference on IT Convergence and Security, ICITCS 2016*, 2016-Janua, pp. 149–153. doi: 10.1109/ICITCS.2016.7740323.

Kitsios, F. and Kamariotou, M. (2016b) 'The impact of information technology and the alignment between business and service innovation strategy on service innovation performance', *ICIMSA 2016 - 2016 3rd International Conference on Industrial Engineering, Management Science and Applications*. doi: 10.1109/ICIMSA.2016.7504042.

Konieczny, R. and Idczak, R. (2016) 'Mössbauer study of Fe-Re alloys prepared by mechanical alloying', *Hyperfine Interactions*, 237(1), pp. 1–8. doi: 10.1007/s10751-016-1232-6.

Kotsiantis, S. B., Zaharakis, I. D. and Pintelas, P. E. (2006) 'Machine learning: A review of classification and combining techniques', *Artificial Intelligence Review*, 26(3), pp. 159–190. doi: 10.1007/s10462-007-9052-3.

Last, F., Douzas, G. and Bacao, F. (2017) 'Oversampling for Imbalanced Learning Based on K-Means and SMOTE', pp. 1–19. Available at: http://arxiv.org/abs/1711.00837.

Laurikkala, J. (2001) 'Improving Identification of Difficult Small Classes By', *Information Sciences*.

Le, T. *et al.* (2018) 'A cluster-based boosting algorithm for bankruptcy prediction in a highly imbalanced dataset', *Symmetry*, 10(7), pp. 1–12. doi: 10.3390/sym10070250.

Ling, C. X. and Sheng, V. S. (2008) ‘Cost-Sensitive Learning and the Class Imbalance Problem’, *Encyclopedia of Machine Learning*, pp. 231–235. doi: 10.1.1.15.7095.

Liu, X. Y. and Zhou, Z. H. (2013) ‘Ensemble methods for class imbalance learning’, *Imbalanced Learning: Foundations, Algorithms, and Applications*, pp. 61–82. doi: 10.1002/9781118646106.ch4.

López, V., Fernández, A. and Herrera, F. (2014) ‘On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed’, *Information Sciences*. Elsevier Inc., 257, pp. 1–13. doi: 10.1016/j.ins.2013.09.038.

Ma, L. and Fan, S. (2017) ‘CURE-SMOTE algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests’, *BMC Bioinformatics*, 18(1), pp. 1–18. doi: 10.1186/s12859-017-1578-z.

MacIejewski, T. and Stefanowski, J. (2011) ‘Local neighbourhood extension of SMOTE for mining imbalanced data’, *IEEE SSCI 2011: Symposium Series on Computational Intelligence - CIDM 2011: 2011 IEEE Symposium on Computational Intelligence and Data Mining*, pp. 104–111. doi: 10.1109/CIDM.2011.5949434.

Moayedikia, A. *et al.* (2017) ‘Feature selection for high dimensional imbalanced class data using harmony search’, *Engineering Applications of Artificial Intelligence*, 57, pp. 38–49. doi: 10.1016/j.engappai.2016.10.008.

Moro, S., Laureano, R. M. S. and Cortez, P. (2011) ‘Using data mining for bank direct marketing: An application of the CRISP-DM methodology’, *ESM 2011 - 2011 European Simulation and Modelling Conference: Modelling and Simulation 2011*, (Figure 1), pp. 117–121. Available at: http://archive.ics.uci.edu/ml/datasets/Adult.

Ofli, F. *et al.* (2016) ‘Combining human computing and machine learning to make sense of big (Aerial) data for disaster response’, *Big Data*, 4(1), pp. 47–59. doi: 10.1089/big.2014.0064.

Omatu, S. *et al.* (2014) ‘Multi-agent Technology to Perform Odor Classification Case Study : Development of a VO for Odor Classification’, pp. 241–252. doi:

10.1007/978-3-319-07596-9.

Oskouei, R. J. and Bigham, B. S. (2017) ‘Over-sampling via under-sampling in strongly imbalanced data’, *International Journal of Advanced Intelligence Paradigms*, 9(1), p. 58. doi: 10.1504/IJAIP.2017.081179.

Pal, M. (2005) ‘Random forest classifier for remote sensing classification’, *International Journal of Remote Sensing*, 26(1), pp. 217–222. doi: 10.1080/01431160412331269698.

Prati, R. C., Batista, G. E. A. P. A. and Monard, M. C. (2004) ‘Class imbalances versus class overlapping: An analysis of a learning system behavior’, *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, 2972, pp. 312–321. doi: 10.1007/978-3-540-24694-7_32.

Provost, F. (2000) ‘Machine learning from imbalanced data sets 101’, *Proceedings of the AAAI’2000 Workshop on …*, p. 3. doi: 10.1.1.33.507.

Qureshi, S. A. *et al.* (2013) ‘Telecommunication subscribers’ churn prediction model using machine learning’, *Eighth International Conference on Digital Information Management (ICDIM 2013)*, (September), pp. 131–136. doi: 10.1109/ICDIM.2013.6693977.

Ramyachitra, D. and Manikandan, P. (2014) ‘Imbalanced Dataset Classification and Solutions: a Review’, *International Journal of Computing and Business Research (IJCBR) ISSN (Online*, 5(4), pp. 2229–6166.

Saito, T. and Rehmsmeier, M. (2015) ‘The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets’, *PLoS ONE*, 10(3), pp. 1–21. doi: 10.1371/journal.pone.0118432.

Stefanowski, J. and Wilk, S. (2008) ‘Selective Pre-processing of Imbalanced Data for’, *Data Warehousing and Knowledge Discovery (Lecture Notes in Computer Science Series 5182)*, pp. 283–292. doi: 10.1007/978-3-540-85836-2_27.

Sun, A., Lim, E. P. and Liu, Y. (2009) ‘On strategies for imbalanced text classification using SVM: A comparative study’, *Decision Support Systems*. Elsevier

B.V., 48(1), pp. 191–201. doi: 10.1016/j.dss.2009.07.011.

T, E. and M, A. (2016) ‘Classification of Imbalance Data using Tomek Link(T-Link) Combined with Random Under-sampling (RUS) as a Data Reduction Method’, *Journal of Informatics and Data Mining*, 1(2), pp. 1–12. doi: 10.21767/2472-1956.100011.

Tek, F. B., Dempster, A. G. and Kale, I. (2010) ‘Parasite detection and identification for automated thin blood film malaria diagnosis’, *Computer Vision and Image Understanding*, 114(1), pp. 21–32. doi: 10.1016/j.cviu.2009.08.003.

Tomek, I. (1976) ‘Tomek Link: Two Modifications of CNN’, *IEEE Trans. Systems, Man and Cybernetics*, pp. 769–772. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4309452.

Wei, X. *et al.* (2017) ‘An ensemble model for diabetes diagnosis in large-scale and imbalanced dataset’, *ACM International Conference on Computing Frontiers 2017, CF 2017*, pp. 71–78. doi: 10.1145/3075564.3075576.

Weka (2020). Available at: https://www.cs.waikato.ac.nz/ml/weka/index.html (Accessed: 10 January 2020).

Wu, D. *et al.* (2016) ‘Mixed-kernel based weighted extreme learning machine for inertial sensor based human activity recognition with imbalanced dataset’, *Neurocomputing*. Elsevier, 190, pp. 35–49. doi: 10.1016/j.neucom.2015.11.095.

Xu, C. *et al.* (2018) ‘Vehicle classification using an imbalanced dataset based on a single magnetic sensor’, *Sensors (Switzerland)*, 18(6), pp. 1–16. doi: 10.3390/s18061690.

Zhang, J. and Mani, I. (2003) ‘KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction’, *Proceedings of the ICML’2003 Workshop on Learning from Imbalanced Datasets*, pp. 42–48.