DEVELOPMENT OF MULTICLASS DATASET RESAMPLING TECHNIQUE BASED ON DATA SIMILARITY DEGREE AND DATA DIFFICULTY FACTORS

BY

DAKO, Dickson Apaleokhai
MTech/SICT/2018/8711

DEPARTMENT OF COMPUTER SCIENCE,
FEDERAL UNIVERSITY OF TECHNOLOGY
MINNA

NOVEMBER, 2021

DEVELOPMENT OF MULTICLASS DATASET RESAMPLING TECHNIQUE BASED ON DATA SIMILARITY DEGREE AND DATA DIFFICULTY FACTORS

BY

DAKO, Dickson Apaleokhai
MTech/SICT/2018/8711

A THESIS SUBMITTED TO THE POSTGRADUATE SCHOOL, FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA, NIGERIA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF MASTER OF TECHNOLOGY IN DEPARTMENT OF COMPUTER SCIENCE, FEDERAL UNIVERSITY OF TECHNOLOGY MINNA

NOVEMBER, 2021

ABSTRACT

One of the most complex machine learning and data classification problem is learning from skewed or imbalanced dataset. These imbalanced preprocessing approaches having received increasing research attention over the years, makes it necessary to access the scope of what have been achieved and what needs to be improved upon. Although numerous techniques for improving classifiers performance have been introduced but most of these techniques are for binary problems; the identification of conditions for the efficient use of these techniques is still an open research problem. This research work developed a multiclass resampling technique using class similarities degree and data difficulty factors. Nearest Neighbours technique was adopted to evaluate the neighbours of each example in the dataset and also the distance between each example *x* and its neighbours. This information about the neighbours of each example was further used to derive their difficulty type (safe and unsafe). 20 samples were selected from each class in the imbalanced dataset; these samples were evaluated using the proposed method to derive the similarity degree between classes. Finally, the similarities degree and difficulty type of each example were used to evaluate the safe level of examples; which then served as the criteria for selecting the examples to oversample and undersample. The new resampling technique, MIRT was tested on five standard imbalanced dataset, which were selected based on their different degree of difficulty level. After resampling the dataset, classification of the dataset was done using KNN, SVM and CART classifier. The performance of the proposed technique, MIRT on CART classifier which achieved a 100 percentage in 4 of the 5 data samples used was better than SOUP, SOUPBag and MRBB resampling techniques which were compared using the G-mean values. Also, among the claasifiers used, CART performed way better than KNN and SVM. Finally, the similarity degree derived from this work can be further apply on dataset with classes more than four; which are more complex to classify.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abbreviation	Description
AUC	Area Under Curve
CART	Classification And Regression Trees
ECOC	Error-Correcting Output Codes Strategy
FN	False Negative
FP	False Positive
FPR	False Positive Ratio
HVDM	Heterogeneous Value Difference Metric
MIRT	Multiclass Informative Preprocessing Technique
MRBB	Multiclass Roughly Balanced Bagging
NN	Nearest Neighbour
OVA	One Verse All
OVO	One Verse One
ROC	Receiver Operating Characteristics
SMOTE	Synthetic Minority Oversampling Technique
SOUP	Similarity Oversampling and Undersampling Preprocessing
SVM	Support Vector Machine
TN	True Negative

TP	True Positive
TPR	True Positive Ratio
MIT	Massachusetts Institute of Technology

ABBREVIATIONS

CHAPTER ONE

1.0 INTRODUCTION

1.1 Background to the Study

With the exponential growth and availability of data on a grand scale from sophisticated and networked systems, such as internet of things, security, surveillance, finance, medicine, academic and other industries. It is vital to advance fundamental understanding of knowledge acquisition and analysis from raw data use for decision-making and prediction processes (De and Do, 2020). Despite the progress made in data engineering techniques and existing knowledge discovery, learning from imbalanced dataset challenges remains a difficulty that has drawn increasing interest from industry and academia (Napierala and Stefanowski, 2016; Fernández *et al.*, 2018).

When one or more classes in a dataset have a little number of instances (minority class) in comparism to the remaining classes in the dataset (majority class) by a significant margin, the dataset is said to be imbalanced. Binary and multiclass datasets are defined as having two classes and more than two classes, respectively. The number of individuals with a certain ailment, such as asthma, in a random sample of 50 people is substantially lesser than the asthma-free patients in number. (Napierala and Stefanowski, 2016). Such instance takes place in many important applications which includes text classification, face and image recognition (Thabtah *et al.,* 2019), word pronunciations learning (Ali *et al.*, 2019), fraud detection, medical diagnosis (Blaszczynski and Lango, 2016). Learning from imbalanced dataset remain one of the most complex difficulties for supervised machine learning (Blaszczynski and Lango, 2016). Imbalanced data classification is a major

difficulty in data mining and machine learning that can be seen in many real-world datasets (Duan *et al.*, 2020).

Owing to the importance of this issues, major contributions have been made to existing and developing techniques. There are three categories for dealing class imbalanced based issues based on the approach that is been adopted in dealing with the imbalance problem: The internal or algorithmic level technique constructs or alters existing algorithms while taking minor class ramifications into account. External or data-level approaches, which involve preprocessing data in order to rebalance class distributions and reduce the unequal distribution effect on the classification process. The internal or algorithmic level technique constructs or alters existing algorithms while taking minor class ramifications into account. And the third category is cost-sensitive strategy, it integrates a variety of misclassification in the learning phase costs for every class by combining data and algorithmic level approaches in the learning phase (Ali *et al.*, 2019).

The American Association for Artificial Intelligence (now the Association for the Advancement of Artificial Intelligence) workshop on Learning from Imbalanced Datasets (AAAI '00), the International Conference on Machine Learning workshop on Learning from Imbalanced Datasets (ICML '00), and the American Association for Artificial Intelligence (now the Association for the Advancement of Artificial Intelligence) workshop on Learning from Imbalanced Datasets (AAAI '00) all reflect the increased interest in imbalanced (He and Garcia, 2009).

The core problem associated with the imbalanced dataset learning problem is that it has the potential to dramatically degrade the efficiency of virtually all standard learning techniques. The majority of classifiers assume or expect equal misclassification costs or balanced class distributions. (He and Garcia, 2009). As a result, when faced with high level of imbalanced datasets, these techniques fail to appropriately capture the data's distributive properties, resulting in inaccurate results for the minority class. (Fernández *et al.,* 2018; Napierala and Stefanowski, 2016). Furthermore, the minority class is often has the most learning appeal, and when it is not correctly classified, it comes at a high cost (Sáez *et al.,* 2016).

Learning algorithms or built models that do not take into account the dilemma of class imbalance can be overwhelmed by the majority class and ignoring the minority class. Consider a binary data collection with 98:2 percent imbalance ratio, in which the majority class makes up ninety-eight percent (98%) of the entire dataset and the minority class only contain two percent (2%). To reduce the erroneous rate, the learning algorithm group all of the samples into the majority class, resulting in a two percent (2%) error rate. The error level may appear small but it effect in real life may lead to great loss such as diagnosing a patient of not having a rare disease such as COVID-19, when actually the patient does. All instances belonging to the minority class are prioritized in this situation, and they are identified as improperly categorised, and the classification scenario is considered totally unsuccessful because the minority class that is of more interest have been totally misclassified (Lin *et al.,* 2017; Jedrzejowicz *et al.,* 2018).

Knowing that there is a disparity in class of a dataset makes learning more difficult, however, the difference by size in class examples is not the only source of possible problems (Sáez *et al.,* 2016; Błaszczyński and Stefanowski, 2018). When the cardinalities ratio for the minority and majority classes are compared, utmost importance to analyse the data difficulty factors which is also known as data intrinsic characteristics, as well as the degree of similarities between classes is required (Napierala and Stefanowski, 2016; Koziarski *et al.,* 2020; Weiss, 2012; Rendón *et al.,* 2020).

Data difficulty factors in dataset is the distribution of class examples within the dataset. This distribution can be categories into sub-concepts, small disjuncts, overlapping, borderline, rare and outlier examples based on their position (Lango *et al.,* 2017). The degree of similarity between classes presents the information about the inter relationship that exact between every pairs of classes (Janicka *et al.,* 2019).

Therefore, this research is on how to address this problem of multiclass imbalance techniques putting into consideration the data intrinsic characteristics and the degree of similarity that exist between classes in the dataset.

1.2 Statement of the Research Problem

Combining class imbalance, data difficulty factors and multiclass into similar problem has huge negative effects on the accuracy level of common classifiers and deteriorates predictions performance (De and Do, 2020). Multiclass imbalanced problem are fairly fresh concept when compared with binary dataset that have received tremendous attention (Lango *et al.,* 2017; Janicka *et al.,* 2019; Wojciechowski *et al.,* 2018). Solving class imbalance issues in multiclass problem, variety of irregularities that do not exist in binary

dataset are encountered, such as, the presence of disjoint majority classes or minority classes (De and Do, 2020; Lango *et al.,* 2017).

Furthermore, despite the methods that have been proposed to handle multiclass imbalanced dataset, information about dataset such as data difficulty level, class distribution concentration and data complexity have not been greatly considered due to difficult nature of dataset and limitations of the decomposition strategies (Fernández *et al.,* 2019).

However, few works have been proposed to handle preprocessing method that uses dataset information such as (Janicka *et al.,* 2019; Lango and Stefanowski, 2018; Napierala and Stefanowski, 2016) and these details can be further explored to standardize the dataset and achieve improved performance of classifiers. Hence, this research is on how to design a preprocessing resampling technique that consider important data intrinsic characteristic and degree of similarity between classes.

1.3 Aim and Objectives of the Study

The aim of this study is to develop a preprocessing technique for multiclass imbalanced dataset based on data similarity degree and data difficulty factors.

This will be achieved by the following objectives:

i. To design a new data level multiclass preprocessing technique.
ii. To implement the designed technique in (i) using jupiter notebook, [a python interactive computing environment and other python libraries which includes pandas, yellowbrick, seaborn, sklearn, matplotlib, numpy].
iii. To evaluate the performance of the technique using the following standard matrix, F-Score, AUC **(**Area Under ROC Curve**)** and Geometric Mean (G-Mean)

iv. To compare the results with existing techniques.

1.4 Significance of the Study

The increasing needs to achieved high-level accuracy in classification tasks, cannot be over emphasized as critical fields such as medicine, aeronautic and engineering have adopted the use of machine learning to a great degree. From the imbalanced nature of real-life dataset, the presences of data intrinsic characteristics and skewness have made classification task much more complex; data level preprocessing techniques reduces these complexities to some notable proportion.

The proposed data level preprocessing technique will balance the cardinalities of classes and will also put into consideration class interrelationships information and data intrinsic characteristic factors thereby achieve an improved dataset for data classification and increases classifier accuracy.

This work intend to develop a resampling technique to reduce the cost of multiclass imbalanced dataset preprocessing and also increase the quality of imbalanced dataset use for machine learning activities by researchers, data scientists, data engineers and data analysts.

1.5 Scope of the Study

This research work is focus on the development of a multiclass preprocessing technique for multiclass dataset using similarity information about the dataset. This work does not includes all aspect of preprocessing task such as data cleaning, empty feature generation

but adopted already existing methods but focused on informatively improving the resampling process using information about the dataset and the data intrinsic characteristics. The proposed resampling technique is for multiclass dataset only, it does not consider binary dataset. However, five (5) real-world multiclass imbalanced datasets gotten from UCI repository, representing different imbalance ratios, data intrinsic characteristics ratio and number of classes have been selected. These datasets have also been adopted in almost all related experimental studies on class imbalance, some of which are Janicka *et al.* (2019), Sáez *et al.* (2016), Galar *et al.* (2011), Fernández *et al.* (2013), Lango *et al.* (2017), Napierala and Stefanowski (2012), Błaszczyński and Stefanowski (2018), Nwe and Lynn (2020) for evaluation of their proposed techniques.

CHAPTER TWO

2.0 LITERATURE REVIEW

2.1 Imbalanced Data

Imbalanced data refers to the disproportion of class examples that exist in a dataset to be used for a machine learning task; in such a dataset, one or more classes are underrepresented (minority classes) in comparism to other class(es) in the dataset (majority classes) (Wang and Yao, 2012) as shown in Figure 2.1. The uneven representation of classes is due to the fact that some classes occurs significantly more frequently in real-life scenarios; for example, in the current coronavirus (COVID-19) outbreak, only 320,000 people have tested positive for the virus out of millions who have been diagnosed in China (Max *et al.,* 2020).

With this issue of class imbalanced dataset, a classifier's performance tends to be biased in favor of some classes (majority class) in the dataset. Classifiers with performance bias operate differently, because there is more dataset available for the classifier to train, solutions in the majority class tend to be more accurate. However, considering the minority class, solutions with inadequate precision are carried out. In real-world applications such as bioinformatics, bleeding detection in medical diagnostics, fraud detection, and education; the problem of imbalanced data distributions is well-posed. Cost sensitive algorithm and neighborhood cleaning rule, safe level SMOTE are the most prevalent approaches for dealing with imbalanced data and they are generally applicable to binary datasets (Kaur *et al.,* 2019).

Learning algorithms struggle with the dataset's class imbalanced ratio since they are designed for balanced classes, generating a bias in favor of the class with more examples. However, from the standpoint of machine learning, the minority class examples tends to be the class of interest since, despite its rarity, it contains more significant and useful knowledge (Krawczyk, 2016).

Numerous methods have been presented in the literature to alleviate difficulties associated with class imbalance. This is thought-provoking and demanding in today's study domains, where many binary class problem, classes problem, cost of misclassified class, negligible disjoints, class overlapping, and magnitude of imbalanced datasets all require attention at the same time. Because multiclass imbalance problems include a variety of essential challenges, binary class problems have gotten increased attention (Ali *et al.*, 2019).

2.2 Types of Imbalanced Data

The classification tasks are categorized into binary or multiclass, based on the number of classes involved. As illustrated in Figure 2.1, a dataset with two classes or more is called binary and a dataset with more than two classes is called multiclass.

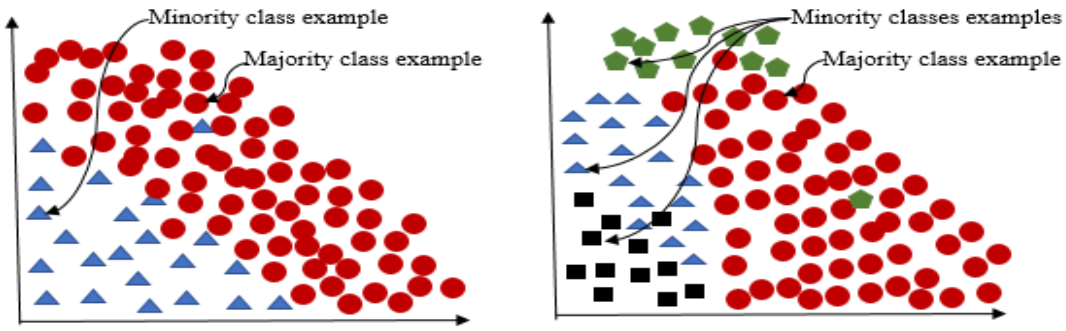

Figure 2.1: Two Possible Class Imbalance Scenarios (left) Binary class imbalanced data distribution, (right) Multiclass imbalanced data distribution (Sáez *et al.*, 2016).

In comparism to what has been accomplished in its binary equivalent, research on multiclass imbalanced classification is still in its initial phase (Cruz *et al.,* 2019; Alejo *et al., 2017*; Lango *et al.,* 2017; Krawczyk, 2016). Also, in binary dataset. In as much as massive achievement have been recorded in binary classification, most of its preprocessing method cannot be directly applied to multiclass dataset which has more complicated situation (Krawczyk, 2016). As the relationships in classes get more complex, handling many minority classes makes the learning task much more complicated. Dealing with multiclass problems, performance on a class may be lost while attempting to obtain it in another class. With such issues in mind, there are many issues to be considered and resolve by novel approaches (Sáez *et al.,* 2016).

In other cases, separating numerous classes with low cardinalities may be beneficial. Taking into consideration the difficulty of distinguishing between two types of asthma (minority classes) and healthy individuals in medicine (majority class). When one type of asthma is classified as a minority class and the other is grouped with the healthy to form the majority class, an intolerable situation occurs in which sick people are mistaken for healthy people. While categorizing all asthmatic patients into a single minority class may be a better solution, it still leads to the unexpected loss of asthma type information (Lango *et al.,* 2017).

2.2.1 Complexity of multiclass dataset

Traditionally, binary datasets have been associated with imbalanced classification. The class of interest and mostly the class with least instance are referred to as the minority or "positive" class and the most occurring class as the majority. As a result, the majority of the studies on this topic has concentrated on emphasizing the acknowledgment of the underrepresented class. Dealing with multi-class issues is difficult, and it becomes even

more difficult when there is an imbalance. When dealing with multiple majority and multiple minority classes, ascertaining which class should be evaluated and optimised at the point of learning a priori, the way it was achieved in studying binary case is difficult (Fernández *et al.,* 2019).

In numerous real-world scenarios, there are countless number of areas whose datasets are multiclass. Some of these areas includes microarray research, protein categorization, medical diagnostics, video mining, activity recognition and target detection are just a few examples. All of these issues have something in common: this is the distribution of instances throughout the classes is not uniform, as most of the classes are quite similar. In this regard, reference must be made to the multiclass case of the imbalanced data classification problems, and as the number of classes grows, so do the difficulty of effectively expressing the entire problem space (Hossen *et al.,* 2018). For the increased number of borders to evaluate in a popular case analysis, reference must be made first to the constraint. However, in the imbalanced scenario, the occurrence of multiple majority class and multiple minority class is the most crucial issue that must be considered. This means that it is no more feasible to pay attention just on a particular class in order to improve the learning techniques geared toward that class (Rendón *et al.,* 2020). When dealing with multiple class imbalanced datasets, however, this is not the only issue. All data inherent qualities that degraded performance in the binary case are now emphasized even more. The degree of similarity across classes in the dataset, as well as the reliance within classes (including overlaps) and relationships between classes, must all be thoroughly examined. This interdependence among the classes makes it difficult to learn from, but it can be used to assess the nature of the dataset during preprocessing (Janicka *et al.,* 2019).

As a result of these certainty, there arise a simple yet important question: what are the ways that multiclass imbalanced datasets be effectively addressed? Though, there is not a straightforward response to this. Overcoming the issue of extending ordinary binary class solutions for used in this context. Contrarily, data-level approaches (preprocessing) are not exactly applied as the search region is enlarged, for example, determining the right sample amount for each class. Diversely, it is because there may be more than one minority class, algorithmic level solutions become more difficult. (Fernández *et al.*, 2019).

To solve all of these concerns, one basic but yet effective method for maintaining standard binary-class imbalanced techniques in multiclass problems must be emphasis: Using decomposition methods (Hossen *et al.*, 2018). Following the divide-and-conquer paradigm, original datasets are separated into binary ones. As a result, a collection of classifiers must be learned, each of which is in charge of one of the innovative binary problems. The results of all the classifiers for a given instance are pooled in the testing phase to make the final verdict (Vluymans *et al.*, 2018). As a result, the challenge of dealing with the multiclass problem shifts from the classifier to the combination stage.

2.2.2 Multiclass data decomposition techniques

The most common approaches for dealing with multiclass problems are class decomposition strategies. This strategy involves reducing the problem into a binary class subtask that can be learned by binary classifier, even in the case of imbalanced data. This approach is required because most standard classifiers are designed for binary problem only (De and Do, 2020). When attempting to categorize any multiclass imbalanced situation, it is evident that the greater the amount of classes involved for classification, the more difficult it turns out to accurately select the query instance's result label. This is

primarily due to the data difficulty factors among the many dataset classes, which rises as increasing classes are interrelated. A divide-and-conquer strategy is a basic yet effective technique to approach this problem. Decomposition approaches, in which the original problem is reduced into numerous easier-to-solve binary subsets, are examples of such methods (Vluymans *et al.,* 2018). The most popular decomposition algorithms, according to Janicka *et al.* (2019), are one-verse-all (OVA), one-verse-one (OVO), and Error-Correcting Output Codes (ECOC).

The OVA and OVO procedures run over all possible pairs of classes or aggregations of classes (for example, one class versus remaining classes) and apply binary problem methods at each iteration. Although the OVA technique keeps original classes, it ignores relationships between them. Although OVO does not aggregate classes, it is more complicated than OVA since it must process all possible pairs of classes (Wojciechowski *et al.,* 2018).

The Error-Correcting Output Codes binarization approach is a common framework for decomposing multiclass issues into binary components. Every class is allocated a specific length n binary string, referred to as a code word, under this strategy. Then, one binary classifier for each bit in the string is trained. The wanted result of a stated classifier is provided by the conformed bit in the code word for this class during the training stage on an instance from class *i*. This procedure can be visualized by a $m \times n$ binary code matrix (Nannes *et al.,* 2020).

2.3 Approaches for Tackling Imbalanced Data Problem

It is because of the importance of issues of class imbalance, significant effort has been made in developing solutions to alleviate them. These proposals can be classified into three

categories based on how they address class imbalance: external or data-driven techniques, internal or algorithmic-driven approaches, and cost-sensitive options. Ensemble learning classifiers are also useful for classifying data that is imbalanced (Ali *et al.,* 2019).

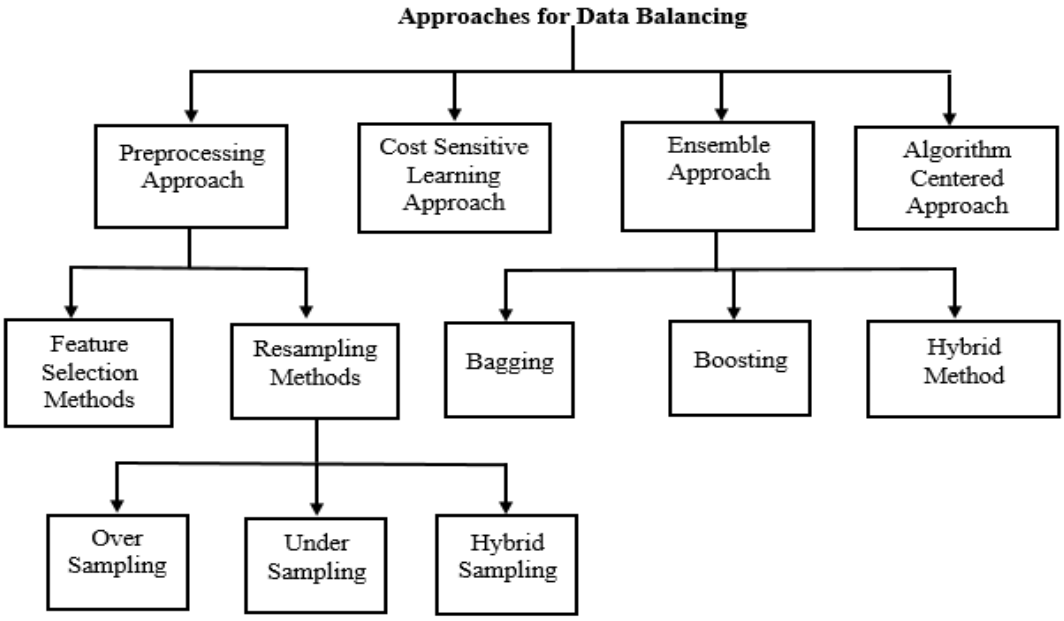

Figure 2.2: Approaches used in Balancing Dataset (Kaur *et al.*, 2019)

2.3.1 Preprocessing approaches

Preprocessing approaches are those techniques that are applied on the dataset to produce a balanced and less difficult dataset for the classifier, these preprocessing techniques are administrated to give rise to a more suitable training data than the original dataset. The techniques that exploit on preprocessing stage are also known as data centred (data level) technique. These methods function through acting directly on data sample space and attempting to lower the imbalance cardinality among the dataset's classes (Kaur *et al.,* 2019). A review of existing algorithms led to the finding that resampling technique together with ensemble methods, particularly oversampling, would be the better solutions for this scenario due to their performance (De and Do, 2020). The preprocessing technique can be categories into:

2.3.1.1 Feature selection method

Feature selection is a dataset preparation stage that picks a subset from all the available attributes or features and removes unnecessary attributes that are not useful. This approaches have been devised to do away with the "curse of dimensionality," which is the amount of evaluation required grows exponentially as the amount of dimensions grows, while preserving or enhancing prediction performance. Filtering, wrapping, and embedding are the three basic ways to feature selection (Fernández *et al.,* 2018).

i. **Filter Method:** Filter methods use statistical or information measurements to choose high-range features. This can be accomplished by evaluating the range of differences between the dataset's features using statistical methods such as ANOVA and T-test (Fernández *et al.*, 2018).

ii. **Wrapper Method:** The wrapper approach classifies the subset of features based on their predictive outcome, utilizing the classifier as a black box. Because a thorough search necessitates 2n separate evaluations, test-and-trial or even greedy strategies are greater practical for finding the optimal choice solutions. To figure out how important each property is to each classifier a sensitivity evaluation could be carried out. (Fernández *et al.*, 2018).

iii. **Embedded method:** Unlike wrapping approaches, embedded methods choose attribute while considering the classifier's architecture and neighborhood information, and are almagamated into the modeling process (Fernández *et al.*, 2018).

2.3.1.2 Resampling method

Despite the fact that resampling is a common strategy for dealing with the problem of class imbalance, the question at hand is what is or finding a way to effectively determine the ideal class distribution given a dataset. Aside from the class distribution issue, another

challenge is a method to efficiently preprocess (resample) the training dataset. Random sampling is straightforward, but it is not always enough. If a dataset class imbalance problem is considered by within class ideas, some duplicate examples space may be favored by random oversampling over others (De and Do, 2020). A better resampling technique would first discover the sub-concepts that make up the class, and then oversample each idea individually to balance the overall distribution. Such a similarity resampling approach, however, raises the cost of data processing. In undersampling of the majority or dominant class in order to make the selected examples more representative raises more issue: sample selection is based on what criterion? Say for example, if distance is used to measure samples, those more occurring or majority class examples that are comparatively distant from minority class examples may represent more majority class attribute, whereas those that are comparably near the minority class examples may be critical in classifier learning algorithms choosing the class boundary region. When it comes to picking high-quality samples, which aspect should be prioritized? Before considering the amount of the examples in each class in the dataset, any resampling technique should consider these difficulties (Sun *et al.*, 2009).

According to Fernández *et al.* (2019), Three types of resampling approaches can be found:

1. **Oversampling**: This is one of the most widely adopted machine learning technique. Random minority oversampling is the most basic type, which just duplicate erratically picked examples from the minority class(es) in the dataset. Oversampling have been demonstrated to be effective, but it can also lead to overfitting. Synthetic Minority Over-Sampling Technique (SMOTE) is a better start-of-the-art sampling method that seeks to solve this problem of class imbalance. It increases the number of examples by generating artificial instances

through interpolating datasets from neighborhood data examples. Some modifications to this oversampling method were offered, such as pay attention exclusively on instances closer to the boundary class. Another method for performing more informed oversampling is to use data preparation. Cluster-based oversampling divides the entire dataset into clusters and then oversamples every subset individually. This minimises both interclass and intraclass disparities. DataBoost-IM, contrarily, uses boosting preprocessing to identify tough cases and then utilises them to produce the synthetic samples. Class-aware sampling is a type of oversampling that is peculiar with stochastic gradient descent to neural networks optimized. The key idea is to ensure that each mini-batch has a homogeneous class distribution and to manage the selection of examples from each class (Buda *et al.,* 2018).

2. **Undersampling**: Another frequent way for ensuring that each class has the same number of examples is to use the same number of examples. Instead of oversampling, examples from majority classes are deleted at random until all classes have the same amount of examples. While it may seem counterintuitive, there is evidence that undersampling is superior to oversampling in some circumstances. This strategy has the considerable disadvantage of discarding a portion of the available data may have been crucial to improve performance. To address this flaw, certain changes were made to more carefully choose which samples should be eliminated. The one-sided selection, for example, identifies redundant samples along the class boundary (Buda *et al.,* 2018).
3. **Hybrids methods**: The hybrid method combines the oversampling and undersampling approaches (Haixiang *et al.,* 2017). This method is mostly used in

multiclass dataset where there is need to reduce of the majority classes and also increase from the minority classes (Fernández *et al.*, 2019).

2.3.2 Cost-sensitive learning

This approach applies a different cost to misclassification of samples in the dataset from distinct classes. Cost-sensitive learning technique refers to a group of algorithms that are delicate to various costs related with specific aspects of the problems under consideration. These cost may be from a variety of sources relating to a specific real-world situation, such as information given by an expert in that domain or information learned during the classifier training phase (Fernández *et al.*, 2019). This method can be applied both at the data level, by specifying costs during resample or feature selection, and at the algorithmic level, by modifying the algorithm to be sensitive to minority class cost. Although less popular than, say, resampling, a comparison of cost-sensitivity and both data-level and algorithm-level techniques shows that cost-sensitivity uses computational resources more efficiently. The disadvantage is that constructing an efficient cost matrix to represent said misclassification is complicated and time-consuming. It should also necessitate subject-matter expertise, as cost attribution should be a well-thought-out procedure (Weiss, 2012). Misclassification costs have been classified into two sorts based on research: example-dependent costs and class-dependent costs. The first assumes that each case should incur a misclassification cost, whereas the second expects that the incorrect categorization should be applied to all classes. As one might assume, the former method is only used in specific scenarios where cost classification by example is simple, whereas the latter is more practical in any context. As a result, cost-sensitive learning can be paired with boosting algorithms to create ensemble techniques in the hopes of achieving superior outcomes (De and Do, 2020).

2.3.3 Algorithm level approaches

For addressing imbalanced datasets, algorithm level approaches might be considered as an alternative to data preprocessing methods. Rather than focusing on changing the training dataset to prevent class skew, this method focuses on changing the classifier learning mechanism. This necessitates a thorough study of the chosen earning technique needed to determine the exact mechanism is behind the majority class bias. Algorithm level solutions do not alter data distributions, making them more compliant to multiple forms of datasets imbalance at the tradeoff of being unique to a single classifier. Instead of changing the provided training data, algorithm-level techniques focus on tweaking current learning algorithms to reduce their concentration in favor of the majority class. This necessitates a thorough understanding of the updated learning algorithm as well as a detailed determination of why it fails to mine skewed distributions. While preprocessing algorithms are more general in that any classifier may be trained to do class balancing subsequently, the approaches covered in this chapter are particular to a certain model. This limits the algorithmic approach's versatility, but also allows for greater specialisation in tailoring the solution to the task at hand. Algorithm based approaches are less common in literature, owing to the fact that algorithm based techniques are more complex to design and implement in comparism to data level technique. In spite of this situation, there are a variety of effective class imbalance results that dependent on direct changes in classifiers. SVMs and their variants, Decision Trees, NN methods, Bayesian classifiers, ANNs, and kernels are just a few of the popular machine learning algorithms that have undergone similar change (Fernández *et al.,* 2019).

2.3.4 Ensemble learning approach

Classifier ensembles, also known as multiple classifier systems in Data Science, combine more than one classifiers output to determine the overall output and are believed to boost

precision when compared to using one classifier. On the other hand, the ensemble learning methods are unable to overcome the problem of class imbalance on their own. When compared to using a single classifier, this method is known to improve accuracy. Any of these ways for dealing with imbalance class issues can be combined into a classifier ensemble to increase overall performance. Ensemble based solutions for class imbalance problem are a type of strategy that has been widely used with success. Some popular ensemble learning methods include Bagging, Boosting (Fernández *et al.*, 2019).

2.3.4.1 Bootstrap aggregating

Bootstrap Aggregating (Bagging) is an ensemble learning approach in which many classifiers are trained using distinct bootstrapped clones of the original training dataset. To put it another way, a new dataset is created for each classifier by selecting without choosing a specific (with replacement) example from the initial dataset. As a result, diversity in Bagging is achieved by the resampling approach, which involves training every one of the classifier with a distinct data subset from the original dataset. The model that is produce should differ owing to variation in the dataset, assuming the corresponding classifier is weak.

Finally, when classifying an unknown instance, weighted vote or a majority choose is employed to determine the class of the instance. The confidence offered by each classifier in the prediction is usually used to execute weighted majority voting (Fernández *et al.*, 2019). Bagging has a number of advantages, one of which is it simplicity in implementation. Bagging also minimises variance since voting effect is comparable to that of averaging in regression, where the minimisation of overfitting effect becomes more visible (Khoshgoftaar *et al.*, 2011).

2.3.4.2 Boosting

Schapire established the concept of boosting in 1990, demonstrating that a weak synthetic sample generator (which is marginally more preferred than random prediction) may be transformed into a strong synthetic sample generator using the PAC learning framework. The algorithm in this family that is mostly represented is AdaBoost. Since the first time that Boosting was used, it was named among the preferred in data mining algorithms. AdaBoost, unlike Bagging, which is just capable of lessen variance, it is also known to its ability to reduce bias (in addition to variance) and, like SVMs, enhances margins (Fernández *et al.*, 2019).

2.3.4.3 Hybrid method

Hybrid methods combine algorithm-oriented, data level approach, and ensemble approaches to accurately solve imbalanced data classification problem (Liu *et al.*, 2019).

2.4 Merit and Demerit of Data Level Preprocessing Approaches

Advantages of Data Centred Approaches (Fernández *et al.*, 2019)

1. It is a simple and widely used method for balancing the training data's class distributions.
2. Investigating the impact of modifying the class distribution to deal with datasets that are imbalanced.
3. Data Centred Approaches are independent of the underlying classifiers

Disadvantages of Data Centred Approaches (Kaur *et al.,* 2019)

1. Achieving the goal of resampling the classes, there is a risk of over-fitting and data loss throughout the resampling process.
2. Choose the best class distribution in a dataset, since the case for choosing the best class distribution varies depending on the dataset, and it affects the classifier's performance.
3. Different distributions in different subclasses contained in a single class, as this increases the dataset's learning complexity.

2.4.1 Merit and demerit of algorithmic approaches

Advantages of Algorithmic Approaches (Kaur *et al.,* 2019)

1) Algorithmic approaches are particular to a given classifier type
2) It minimises the misclassification cost of the minority class

Disadvantages of Algorithmic Approaches (Ali *et al.,* 2019)

1) It is necessary to have a thorough understanding of the chosen strategy in order to determine what specific mechanism is responsible for the majority class bias.
2) A large number of trials are required to determine which one is the most accurate.
3) For dataset shifts, there is a lack of a good validation technique.
4) Algorithmic Approaches necessitate a large amount of processing and storage space.

2.5 Data Intrinsic Characteristics

Although the problem of class imbalance is frequently stated as a deciding criteria for classifier performance decline, there exist times when regular classifiers can get good results. If the dataset is deemed linearly separable, high level accuracy could be obtained

even with the availability of significant class imbalance (or of low complexity). The circumstances that impact nonlinearly separable of datasets are frequently linked to data difficulty factors, also referred to as data intrinsic characteristics (Fernández *et al.*, 2019). The most prevalent distinction is between instances that are safe and those that are unsafe. Safe examples, which are found in relatively homogeneous portion populated solely by instances from a single class, should be simpler for a classifier to learn from, but unsafe examples are thought to be more challenging and more likely to be misclassified (Napierala and Stefanowski, 2016), as shown in Figure 2.3. Understanding the underlying properties of these data, and also their interrelationship to class imbalance problem, is critical for using current and inventing new strategies to deal with data imbalance (Fernández *et al.*, 2019). Here are a number of the data difficulty factors that can be addressed while solving class imbalance problem, which are also grouped as the three sorts of dangerous cases: borderline, rare (small disjunct), and noisy (outlier) examples. (Lango *et al.*, 2017).

2.5.1 Borderline examples

Borderline instance, this can also be referred to as overlapping instance, are examples that are found in the same portion as the class boundary between two or more classes (Napierala and Stefanowski, 2016) as shown in Figure 2.3. When the input features are insufficient to appropriately distinguish between examples of distinct classes, and similar areas of the sample space contain examples from multiple classes, this is referred to as overlapping region (Fernández *et al.*, 2019).

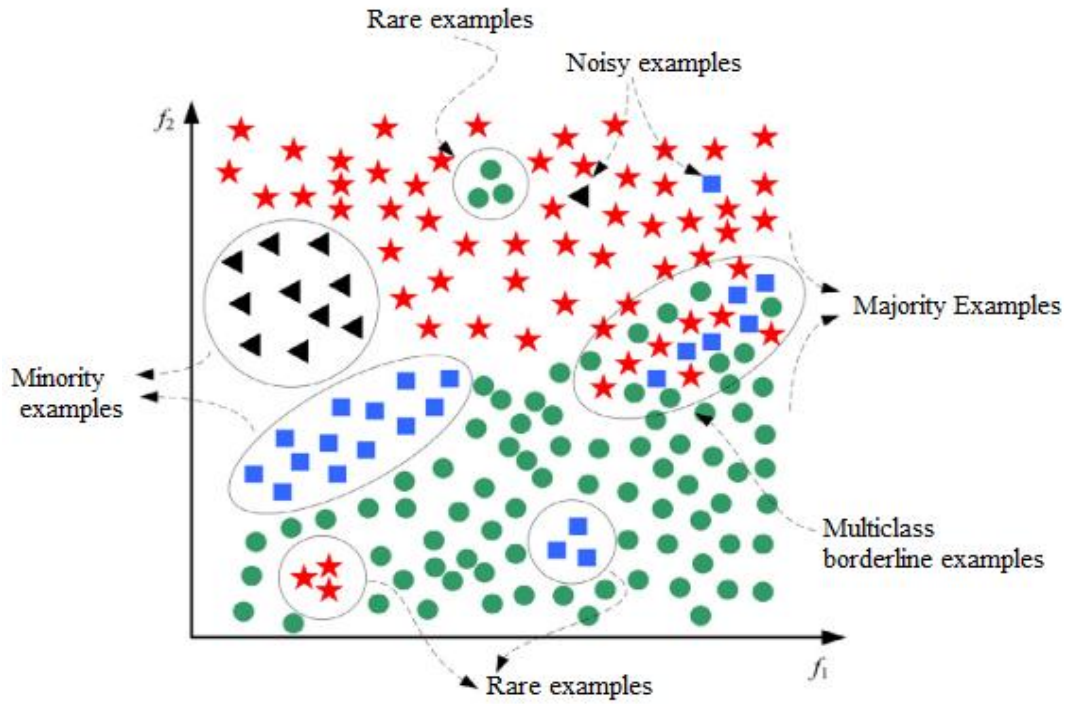

Figure 2.3: Difficult regions in Multiclass Data Distribution (García *et al.*, 2018).

2.5.2 Rare examples

Learning algorithms frequently run into the problem of examples from similar class do not appear in homogeneous portion in the sample space. The "idea" underneath a class is frequently divided into numerous sub concepts that are scattered across the input space. Isolated pair examples positioned on the safe examples of another class and far away from the borderline examples are rare examples. (Fernández *et al.*, 2019).

2.5.3 Outliers examples

In general, there are two categories of noise in machine learning: feature (or attribute) noise and class noise (Fernández *et al.*, 2019). Noisy (class or attribute mistakes) errors degrade the performance of standard classifiers, and they are especially destructive to the minority class (Napierala and Stefanowski, 2016). Individual instances of another class(es) situated within the safe examples of another class are referred to as noisy examples (Napierała *et al.*, 2010).

2.6 Related Studies

Napierala and Stefanowski (2012)**,** proposed a new strategy for identifying distinct types of minority class examples in imbalanced dataset that is focused on examining the examples' local neighborhoods. The proposed analysis of an example's local neighborhood in the original attribute space to determine its type. The class assignment of each minority example's k-nearest neighbors was examined. This is because this method relied on an easy examination of a given amount of neighbors, it was examined to verify if the assigned labels accurately reflected the known distribution. To begin, the experiment was conducted with real-life datasets, which revealed that most datasets has a huge amount of unsafe instances.

Second, the results revealed that all of the investigated classifiers find safe datasets to be rather simple to every classifier. Borderline, rare outlier or noisy examples in dataset are serious source of problems, influencing classifiers in different ways. It was also discovered that the ratio of imbalance and the size of data are not as dominant as the different types of distributions. Finally, when the efficiency of several classifiers were compared on the dataset, it was discovered that J4.8 trees or PART rules and Naive Bayes were the most flavorful to risky forms of instances in the minority class– even for more challenging types. Development of new methods which are able to examine the complexity of real-life datasets and their degree of difficult needs further studies.

In the work of Fernández (2013), binarisation strategies and ad-hoc approaches; a method for analysing the classification of imbalanced datasets with multiple classes; The introduction of a preprocessing mechanism based on SMOTE, known as Static-SMOTE, which iteratively generated fresh instances from the lowest represented class at every

stage, was one of the strategies utilised. Then, a global cost-sensitive strategy was presented, which reweights the examples from every class based on their ratio. Ada-Boost.NC, and specific boosting based methodology for addressing multiclass imbalance problems, is described next.

The work presents an empirical examination of numerous ways for dealing with multiclass imbalanced data situations, the majority of which are reached from a merger of OVO and OVA strategies and binary based approaches, as well as additional ad hoc methods developed specifically for this problem. The following are some key takeaways from the research:

1. For multiple-class imbalanced issues, oversampling strategies have demonstrated to be more vigorous than those formulated on undersampling and cleaning operations in terms of synergy of data level and binarisation techniques.
2. When comparing OVO and OVA approaches, OVO methods have consistently outperformed OVA methods, notably in terms of average performance. The rationale for the results in greater quality is because the paired learning approaches deals with a smaller number of occurrences, making it unlikely to produce imbalanced training datasets, which is the drawback in this situation. Furthermore, the decision bounds of every binary issue in this situation may be significantly easier than in the OVA technique.
3. It is necessary to emphasise that the most effective strategies investigated are those formulated on SMOTE with OVO and OVO with cost sensitive approach.

The author finally determined that binarisation approaches combined with suitable data level or a cost-sensitive strategy are simple but effective mechanisms for improving

classifier accuracy in imbalanced area, although there is still more work to be done on this topic: Scalability, the OVO strategy as a decision making issue, and finally, Intrinsic data features are all non-competent examples in OVO strategy.

Napierala and Stefanowski (2016), explored at the challenges of class distribution in real life dataset by looking at four different categories of minority class instances: safe, rare, borderline, and outliers, all of which can cause classifiers to underperform when learning from unbalanced dataset.

By examining multidimensional visualisations of choosen datasets, the approach validated the prevalence of class distribution issues in real data. Then, in order to identify these types of cases, a new method formulated on examining distribution of a class in a local neighborhood of the examined instance was introduced. Modeling this neighborhood was done in two ways: with k-nearest examples and with kernel functions.

The following are the highlights of the findings:

1. Imbalanced datasets typically include a variety of minority cases in varying amounts.
2. An intriguing result is that outlier examples can make up a significant portion of the minority class -they are found to even outnumber the majority in some datasets.
3. The global imbalance ratio and data amount are less important than the distinct example types.
4. It is handy to distinguish the performance of common classifiers by collecting information on local attributes of the minority class and discriminating between rare, safe, borderline, and outlier examples.

5. Analysing differences between common preprocessing methods might also benefit from considering information on sorts of minority instances.

Napierala and Stefanowski (2016) study paid emphasised on rare and outlier examples, as well as the fact that common data distribution patterns observed across multiple imbalanced datasets can aid in the creation of novel learning algorithms and preprocessing methods for class imbalance.

In multiclass imbalanced datasets, the resampling (oversampling) of distinct classes and samples types was examined by Sáez (2016). The goal was to see how data level approach (oversampling) of some classes and samples types in every class (safe, borderline, rare, or outlier) affected the efficiency of the classifiers constructed.

The classes and instances types been oversampled in the dataset were established after selecting the dataset for this study. To begin, each sample in every class is categorised as safe, rare, borderline, or outlier using the HVDM distance metric for each dataset. Second, take into account all of the well-grounded design discovered in the previous phase. The preprocessing consisted of using an oversampling approach and generating fresh synthetic cases using a scheme much the same to that utilised in binary imbalanced issues by SMOTE approach.

Finally, the performance of alternative classification methods, such as Support Vector Machine (SVM), C4.5, and Nearest Neighbour (NN) rule, used to examine the preprocessed datasets.

Following the analysis of the performance findings, various conclusions concerning the necessity of preprocessing in multiclass imbalanced datasets drawn are:

1) Preprocessing some concrete classes and categories of samples in these classes (safe, rare, borderline, or outliers) can enhance performance that would otherwise be indiscriminately harmed by preprocessing all classes. However, if these classes and types of preprocessing samples are not selected accurately, the results may suffer.
2) Sorts of examples to be preprocessed: To increase the final result, it's critical to focus on the data characteristics of each and every problem, research the distribution of every class, and analyse which instances types should be preprocessed.
3) Choosing the most appropriate classes and examples for preprocessing. When the best instances types and class are chosen to be preprocessed, the outcome show that this preprocessing approach can result in a considerable increase in performance when compared to not preprocessing any classes or not preprocessing at all.

The conclusions found can be used to create unique preprocessing learning algorithms in the future that use this problem structure background knowledge. The conclusions reached can be used to support a variety of multiclass unbalanced learning concepts.

In a multiclass dataset, Lango *et al.* (2017) developed a new technique for assessing the characteristics of samples. The approach determines the safe level by examining the neighborhood of a minority class example as well as additional information about the similarity of neighboring classes to the example class.

The safe level coefficient was generalised in the following way:

Consider the example x, belonging to the minority class Ci. Its safe level is defined as follows in respect to l classes of instances in its local neighborhood:

$$Safe = \frac{1}{n}\sum_{j=1}^{l} nc_j \mu_{ij} \quad (2.1)$$

where ij is the degree of similarity, nCj is the number of class examples, Cj is the number of neighbors in the considered neighborhood of x, and n is the total number of neighbors The paper also developed the idea of class similarity, which is used to extend the process of identifying types of minority instances to a multiclass context and evaluate if safe level values are related to standard algorithm classification performance.

The results demonstrated that this method accurately detects minority class distribution challenges in a variety of artificial and real-world datasets, as measured by values of safe levels for acceptable minority instances. The inability to derive similarity among classes from the dataset is a shortcoming of this study. The author states that by utilising safe levels to adaptively modify resampling, the new method of identifying data complexity in multiclass datasets can be leveraged to develop new preprocessing algorithms.

To balance multiclass imbalanced data in the presence of data difficulty factor, Janicka *et al.* (2019), presented a novel technique called Similarity Oversampling and Undersampling Preprocessing (SOUP), which models interrelationships between classes. This paper's key contribution is a new approach for determining the degree of similarity between classes and then used on Lango *et al.* (2017) way of determining the safe level of examples. This method of determining a safe level was then applied on the SOUP algorithm in the process of oversampling and undersampling process. It was also compared to other well-known

methods such as Global-CS, Static-SMOTE, and Multiclass Roughly Balanced Bagging (MRBB), which it outperformed; it was also compared on decomposition ensembles OVO and OVA, where it also outperformed them. Regardless of its effectiveness, the heuristic approach employed to estimate the similarity level must be carefully reviewed. Nonetheless, SOUP is interested in generalising an underbagging ensemble, such as Neighborhood Balanced Bagging, as a future research topic in order to increase predictive ability.

Nwe and Lynn (2020) proposed an efficient resampling method for skewed distributions in unbalanced datasets. It developed a data preprocessing strategy that focused on the skewed distribution of data points in the imbalanced dataset to improve the efficiency of imbalanced data classification. To tackle the issues associated with imbalanced learning of small disjuncts and short sample size, the author proposes using oversampling and undersampling algorithms based on k-means clustering. The proposed cluster-based resampling approaches also included the Tomek Link-based undersampling method to alleviate the class overlapping problem by deleting the majority samples in overlapping locations. The COTU approach should be expanded to multi-class imbalanced data in the future, and an appropriate value of clusters (K) will be defined for the problems of small disjuncts and small sample size, as the number of clusters (K) affects classification efficiency.

Mahmoud (2020) demonstrated an Oversampling Technique (modified SMOTE) for dealing with imbalanced datasets, which addresses the binary classification of imbalanced datasets. The main goal of the improved SMOTE technique was to create new synthetic minority samples that would reduce the amount of differences between majority and

minority data. When creating new minority data samples, majority data samples are taken into account. The algorithm returns the majority class's k-nearest neighbors as well as the minority class's k-nearest neighbors. The distance between the nearest majority and minority neighbors was then computed and multiplied by a random number. Then, based on the randomly selected minority neighbor, create a new synthetic sample by increasing the distance between the nearest majority neighbor and the nearest minority neighbor.

This strategy was compared to the SMOTE method, which is the usual oversampling approach in literatures, on several datasets using K-Nearest Neighbors, Fuzzy K-Nearest Neighbors, and Support Vector Machines classifiers, and it outperformed the SMOTE method. The author of this paper noted that though this technique was only used with numerical datasets, it might be extended to categorical datasets as well. Another possibility is to use the method with multiclass datasets.

Duan (2020) proposed a novel classifier ensemble framework based on K-means and resampling technique (EKR). To begin, data samples in the majority class were divided into several sub-clusters using K-means, with the k-value determined by the Average Silhouette Coefficient, and then the number of data samples in each sub-cluster was adjusted to match that of the minority classes using resampling technology. Finally, each adjusted sub-cluster and the minority class were combined into several balanced subsets.

Duan (2020) compared to UnderBagging, RUSBoost, SMOTEBagging, and Clustering-based Undersampling (CBU) utilising different data preparation approaches, Duan, (2020) finally concluded that EKR performed better. The author suggests that further study be done to improve the algorithm in detail in order to avoid class overlaps and increase the

number of samples in a subset. In addition, the multi-classification task must be addressed, as well as the development of a multi-classification method.

Kamalov and Denisov (2020), introduced the Multi-Class Combined Cleaning and Resampling (MC-CCR) algorithm as an oversampling technique. The method, which is less impacted by minor disjuncts and outliers than SMOTE, uses an energy-based approach to predict the regions suited for oversampling. It then combines it with a concurrent cleaning operation aimed at decreasing the impact of overlapping class distributions on the learning algorithms' performance. Traditional multi-class decomposition techniques were found to be less influenced by the loss of knowledge regarding inter-class relationships than the MC- CCR. The strong robustness of the suggested technique to noise was demonstrated based on the findings of experimental study conducted for various multiclass imbalanced benchmark datasets. The author indicated that novel approaches of cleaning the majority observations placed near the minority instances, which may be embedded in MC-CCR, should be examined as a direction for future work.

2.7 Summary of Review

After critical study of related works about imbalanced dataset and their preprocessing approaches, it was determined that,

i. An expert should supply the degrees of similarity, or they can be derived from domain knowledge Lango *et al.* (2017). If neither of these options is accessible, some heuristic approaches may be employed. Because the class cardinality ratio does not explicitly reflect the degree of similarity between classes, the heuristics technique devised by Janicka *et al.* (2019) to assess the degree of similarity between classes requires further evaluation.

ii. Since it has also been demonstrated experimentally by Napierala and Stefanowski (2016), Napierala and Stefanowski (2012) that the unsafe examples (categorised into, rare, borderline and outliers) are hard for learning and deteriorate performance for class imbalanced problem, this work will consider the unsafe region during undersampling, paying more attention on borderline region that was not covered in the work of Napierala and Stefanowski (2016).

Finally, this work will design a new technique for evaluating the degree of similarity among classes using the relationship between class examples and implementing it on a new resampling algorithm that is meant to consider unsafe region in the dataset when resampling the dataset.

CHAPTER THREE

3.0 RESEARCH METHODOLOGY

3.1 Approach Used

With machine learning centred around data analysis, which is based on the idea that systems can learn patterns from dataset. This makes it more convincing why expert or domain knowledge should not be depended on, to be able to ascertain the data similarity degree between classes in a given sample of dataset. As proposed in previous literature, data similarity information can be derived from an expert or domain knowledge Lango *et al.* (2017), or by the used of an heuristic approach that evaluate the ratio between classes Janicka *et al.* (2019). Therefore, this study introduces a new technique to ascertain the data similarity degree from dataset, which is used to evaluate the safe level of all examples in a dataset.

3.1.1 System capacity used

1) Dell Laptop (Intel(R) Core (TM)2 Duo CPU 2.00GHz, RAM:4.00GB, 64-bits windows 10pro OS)

3.1.2 Other materials and tools used

1) Anaconda Navigator Software
2) Chrome browser
3) Microsoft word (Office 16)
4) Airtel 4G internet Router
5) Python (Jupyter interactive computing environment, other python libraries used includes pandas, matplotlib, yellowbrick, sklearn, numpy)
6) Microsoft excel (Office 16)

3.2 Research Workflow

The research workflow presents the progression (starting from the imbalanced dataset down to the preprocessed balanced dataset) of this research work; presenting the whole process in the design of the new multiclass resampling technique. Figure 3.1 gives the phases that was covered in this data preprocessing technique development. The workflow in designing new preprocessing technique as presented Figure 3.1, which was further sub-divided in the research framework.

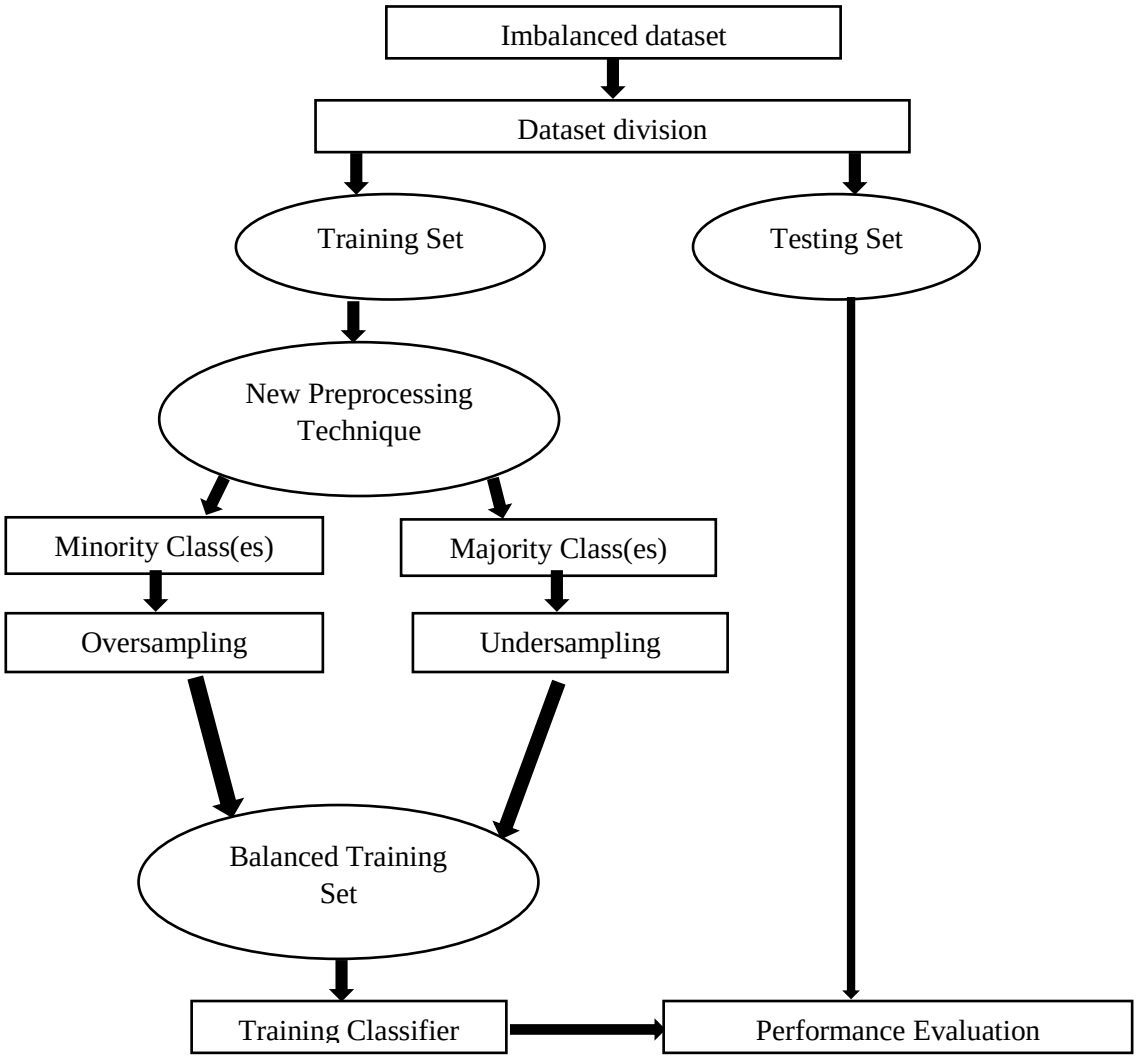

Figure 3.1: Research Workflow

Figure 3.2, the research framework presented in detail the data preprocessing stage, classification stage and performance evaluation stage; these phases gives an overview of what was achieve at each stage of the experiment and the tools used. For the implementation of the new multiclass resampling technique, the data pre-processing stage and classification stage were implemented using python programing language and its libraries.

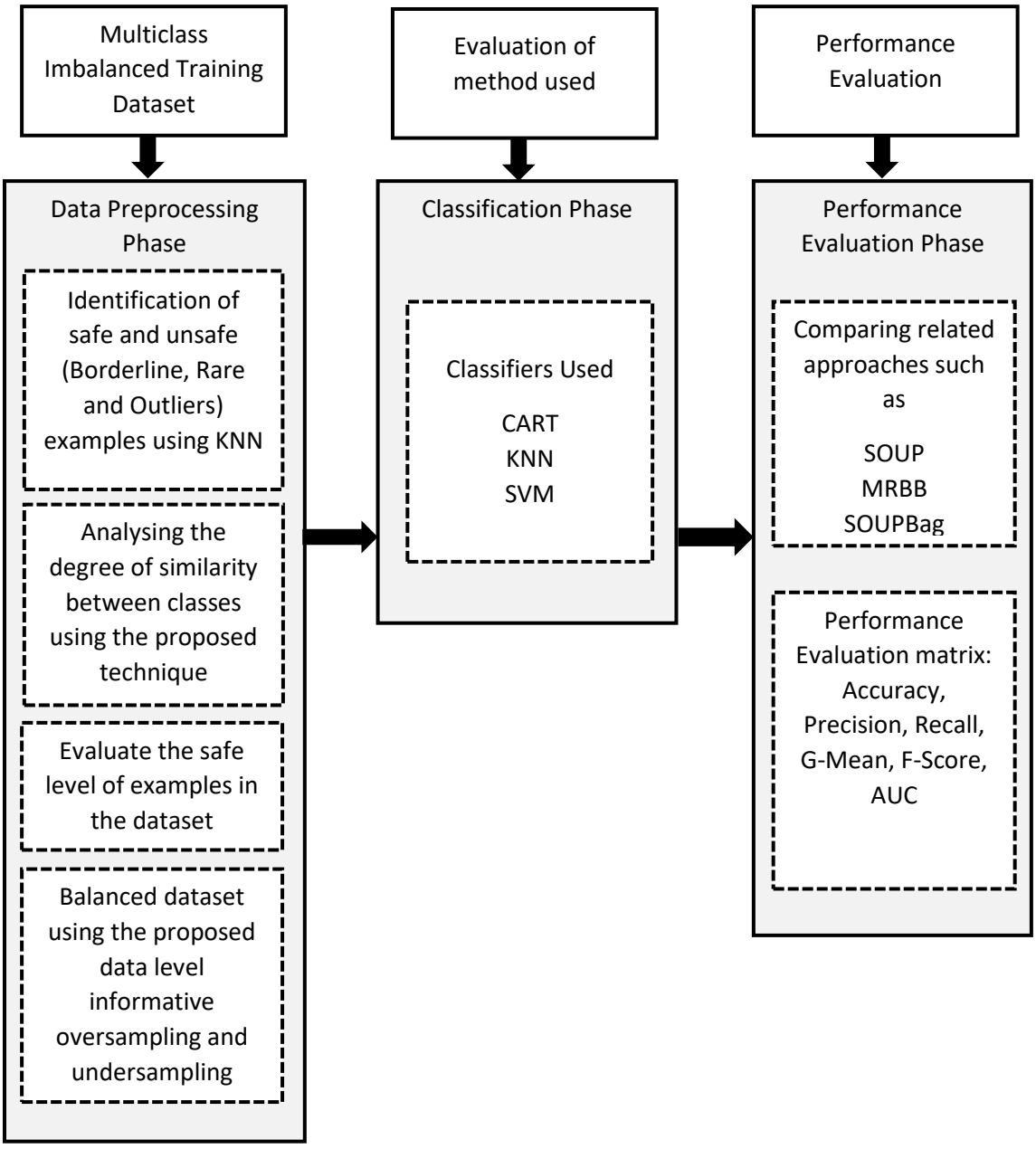

Figure 3.2 Research Framework

3.3 Data Collection

From the nature of this work, it is clear that data level resampling technique is not domain specific, so dataset from diversify field with different number of instances, classes and difficulty levels were used. Critically analysed five (5) diversified real-world datasets from UCI repository were used to evaluate the proposed technique. These datasets have been chosen because they represent different domain, varying degrees of difficulty, has different sizes and imbalance ratios. Above all, these dataset were used based on their popular usage in previous experimental studies of class imbalance problems. The main characteristics of these datasets are presented in Table 3.1.

TABLE 3.1: Characteristics of Multiclass Imbalanced Dataset used

No	Description	No of Attributes	Class Distribution	Minority class(es)	Manority class(es)
1	balance-scale	625	49/288/288	1	2
2	hates_roth	160	30/64/66	1	2
3	Car	1728	65/69/384/1210	2	2
4	Cmc	1473	628/333/511	1	2
5	new_thyriod	225	30/35/150	2	1

3.4 Analysis of the Preprocessing Phases

To achieve the stated aim of this research work, which is the development of a resampling technique for multiclass dataset, the procedure have been spitted into phases as follows:

3.4.1 Identifying the types of examples

Different difficulty variables are associated with different sorts of examples - safe and unsafe (difficult examples). Previous work in class imbalance analyse the ratio between the number of minority and majority examples present in its neighborhood to identify the type of a given case. K-nearest neighbors or kernel functions can be used to model this. In this study, k-nearest neighbors was used because of its simplicity and adoption by the benchmark literature.

3.4.1.1 Categorising the unsafe examples

To categorise the types of example, the class label of the k-nearest neighbor is been analysed. The value for *k* in this work is five (5), due to its usage in imbalanced preprocessing and the Euclidean distance is also been used to calculate the distance between examples. Based on the proportion, the examples are labeled as safe and unsafe, where the unsafe examples are further categorised in the following way as used in previous class imbalance works, in the manner described as proposed by Napierala and Stefanowski, (2016):

5:0 or 4:1 – Safe example is one that is labeled as 4 to 5 examples (further denoted as S).

3:2 or 2:3 – a borderline example (denoted as B). Because the cases with a 3:2 ratio are accurately identified by their neighbors, they may still be safe. However, because the number of neighbors from the interest class and other classes is nearly equal, it's possible that this example is too close to the decision boundary between the classes. As a result, any examples with a 2:3 or 3:2 ratio are considered borderline examples.

1:4 – Only if it has a neighbor from the same class, a rare example (designated as R), has the proportion of neighbors either 1:4 (additionally, in case of 1:4, it must point to the analysed example).

0:5 – Any example with all of its neighbors belonging to a different class is referred to as an outlier, and it is designated by O.

3.4.2 Proposed informative class similarity evaluation technique

To model the relationships between classes in multiclass imbalanced dataset, information of the similarity between pairs of classes were exploited. In general, the intuition behind this similarity degree is as follows: When a particular class's example x has some neighbors from different classes, the neighbors with the highest similarity are preferred. Consider the

asthma learning problem, in which two asthma classes are characterised as being more similar than the non-asthmatic class. If an asthma-type-1 example is not surrounded by just other asthma-type-1 examples (which is the ideal circumstance), it's preferable to have neighbors from the asthma-type-2 class rather than the no-asthma class. Such a neighborhood would allow us to consider the studied example to be safer - it would be easier to recognise it as a member of its class (since it would be less likely to suffer from majority bias). In previous works, the similarity information where acquired from a subject matter expert or domain knowledge. The method of getting the similarity degree between classes was further ascertained by using a heuristic approach which evaluate similarity degree information from the ratio between classes.

This work proposed an approach which models the similarity relationship between minority and majority classes in imbalance dataset. It considered more interior data features, such as the attributes values of randomly selected examples to ascertain the similarities between classes. Since the basic idea of machine learning is learning from data, then the data should be capable of generating the similarities information between classes instead of depending on human knowledge which may be bias or inaccurate.

In more formal terms, it is assumed that for each pair of classes C_i, C_j, the degree of their similarity to itself is defined as $\mu_{ii} = 1$. The degree of similarity as a real $\mu_{ij} \in [0, 1]$. The similarity of a class to another does not have to be symmetric, for example: for some classes C_i, C_j it may happen that $\mu_{ij} \neq \mu_{ji}$. Although the values of μ_{ij} are defined individually for each dataset, the general recommendation is to have higher similarities ($\mu_{ig} \rightarrow 1$) for other minority classes C_g, while similarities to majority classes C_h should be rather low ($\mu_{ih} \rightarrow 0$).

3.4.2.1 Degree of similarities between classes evaluation steps

1. Firstly, randomly select 20 examples from each class in the dataset.
2. Next, evaluate the mean for each class attributes.
3. Take the average of all the mean score for each class to generate a value for each class.
4. Take the difference between each class in the dataset
5. Finally, based on the value for each class, multiply or divide the values by a multiple of 10 to derive a value ranging from 0 to 1. These values now serves as the degree of similarities (μ) between pair of classes.

It is worth noting that this process can only be carried out on numeric attributes or categorical values which can be converted. For example, a dataset with attribute "Male" and "Female" can be represented as 1 and 0 respectively.

3.4.3 Examples safe level evaluation

Imbalanced data difficulty factors correspond to local data characteristics seen in specific sub-regions of the minority class distribution, and the mutual position of an example in relation to examples from other minority and majority class(es) impacts learning classifiers. The Safe Level assessment considers both the degree of similarity across classes and the homogeneity of a k neighborhood.

The safe level coefficient was generalised in the following way:

Consider the case of an example x, who belongs to the minority class C_i. Its safe level is defined as follows in relation to l classes of instances in its neighborhood as:

$$Safe(xc_j) = \frac{1}{n}\sum_{j=1}^{l} nc_j\mu_{ij} \tag{3.1}$$

Where

μ_{ij} is the degree of similarity,

nC_j is the number of examples from class C_j, in the considered neighborhood of x

n is the total number of neighbors.

The safe level value for each example is analysed as follows: the lower the value, the more dangerous (difficult) it is, and vice versa. After successfully evaluating the safe level of all examples in the dataset, the minority classes are informatively oversampled by duplicating the examples with the higher safe level value while the majority class(es) are also informatively undersample by removing the examples with the least safe level until the required threshold is acquired.

3.5 Algorithm Design

The proposed similarity oversampling and undersampling preprocessing technique use the outcome of the similarity information and data difficulty factor to evaluate the safe level which is then applied to determine the examples in the majority classes to undersample and the examples in the minority classes to oversample.

From Table 3.2, line 1 split the entire dataset into their respective classes. Line 2: creates a dataframe which will be used to store the processed balanced dataset. Line 3, evaluate the cardinality been the average of largest minority and the smallest majority class. This cardinality value is then used as benchmark during the resampling process.

Line 4 to 6 evaluate the Nearest Neighbor (5NN) of all the examples in each class. This neighbourhood information is then applied to evaluate the data difficulty factor of each example based on the class of its neighbours. Next, line 7 to 9 evaluate the similarity

between the classes. It continued by processing the safe level of each examples in the dataset using Equation 3.1.

Finally, the undersampling of the examples with the least safe level was done and oversampling of examples with the highest safe level and highest distance were also performed.

3.5.1 Multiclass resampling algorithm design

Start

Input: *S* Original training set of $|S|$ examples with *n* classes; C_{min}: Indexes of minority classes; C_{maj} indexes of majority classes; μ_{ij}: similarities between classes.

Output: S_o: Balanced training set

Stop

Table 3.2: Multiclass Resampling Algorithm

1. **Start**
2. Split dataset S into homogenous parts S_1, S_2, ……, S_n. Each S_i contain all instances from *i* class
3. $S_o = \{\}$
4. m ⟵ avg (mini ∈ Cmaj|Si|, maxj ∈ Cmin|Sj|)
5. **for all** x ∈ *S* **do**
6. the *k* nearest neighbors of *x* according to session 3.3.1.1 is find
7. **end for**
8. **for all** x ∈ *S* **do**
9. calculate the similarity degree between class according to session 3.3.2.1
10. **end for**
11. **for all** x ∈ *S* **do**
12. evaluate the safe level of x_i, according to Equation (3.1) in session 3.3.3
13. **end for**
14. **for all** i ∈ C_{maj} **do**
15. Remove $|S_i|$ - *m*, // Instances in S*i* with the least safe level value
16. So ⟵ So U Si
17. **end for**
18. **for all** *i* ∈ C_{min} **do**
19. Duplicate *m* - $|S_j|$, //S_j , instances with the most safe level values and with the highest distance between examples.
20. So ⟵ So U Sj
21. **end for**
22. **return** S_o
23. **Stop**

3.6 Performance Evaluation Matrix

The performance of the algorithm will be evaluated with the following matrix: F1-score, Geometric-mean (G-Mean), and Area Under curve (AUC). The definition of these evaluation metrics uses in at evaluation, the confusion matrix presented in Table 3.3.

Table 3.3: Confusion Matrix

Confusion matrix		Predicted Labels	
		Positive	Negative
Actual Labels	Positive	TP	FN
	Negative	FP	TN

From Table 3.3, the number of positive samples projected as "positive" is known as True Positive (TP). The number of positive samples predicted as "negative" is known as false negative (FN). The number of negative samples predicted as "positive" is known as false positive (FP). The number of negative samples predicted as "negative" is also known as true negative (TN).

Accuracy: The proportion of correctly predicted samples to total samples, which is computed using Equation (3.2)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

F1-score: The F1-score of positive data takes into account both the Precision and Recall of the classification model, which is the harmonic average of Precision and Recall as in Equation 3.11:

Where p denote the positive instances

$$Precision_P = \frac{TP}{TP + FP} \tag{3.3}$$

$$Recall_P = \frac{TP}{TP + FN} \tag{3.4}$$

$$F1 - Score_p = \frac{2 \times Precision_P \times Recall_P}{Precision_P + Recall_P} \tag{3.5}$$

Similarly, the F1-score for negative samples is derived as follows:

Where n denote the negative instances

$$Precision_N = \frac{TN}{TN + FN} \tag{3.6}$$

$$Recall_N = \frac{TN}{TN + FP} \tag{3.7}$$

$$F1 - Score_N = \frac{2 \times Precision_N \times Recall_N}{Precision_N + Recall_N} \tag{3.8}$$

As a result, synthetic Precision, Recall, and F1-score are generated for the complete dataset as follows:

$$Precision = \frac{Precision_P + Precision_N}{2} \tag{3.9}$$

$$Recall = \frac{Recall_P + Recall_N}{2} \tag{3.10}$$

$$F1 - Score = \frac{F1 - Score_P + F1 - Score_N}{2} \tag{3.11}$$

G-mean, which is defined as Equation (17) can be used to evaluate the overall performance of an algorithm:

$$TPR = \frac{TP}{TP + FN} \tag{3.12}$$

$$TNR = \frac{TN}{FP + TN} \tag{3.13}$$

$$G - mean = \sqrt{TPR \times TNR} \tag{3.14}$$

TPR and TNR are used by G-mean to assess positive and negative class classification performance. The G-mean is not optimal if one of the two is really small.

AUC (Area Under Curve) **ROC** (Receiver Operating Characteristic) curves uses a model's ability to distinguish between classes in classification issues to describe the standard performance measurements. AUC represents the degree or measure of separability, whereas ROC is a probability curve. AUC with a higher value indicates that the model is more accurate at predicting true positives. It also depicts the region under the ROC, which

is a fairly solid categorisation evaluation criteria. The trade-off between TPR and FPR can be seen using the ROC. The AUC scale ranges from 0 to 1. The higher the AUC value, the better the algorithm's performance.

The performance of the proposed resampling technique was evaluated and compared against KNN and CART, F1-score, G-mean, and AUC. These evaluation matrices have been used in this work because of its adoption in evaluation algorithms in previous literatures.

CHAPTER FOUR

4.0 RESULTS AND DISCUSSION

4.1 Degree of Data Difficulty Present in Dataset Used

Presented on Table 4.1 is the value of each data difficulty factor present in the dataset used for the evaluation of the new technique. This data difficulty value of each example was used to evaluate the safe level of examples to ascertain the efficiency of the new technique and to ascertain to what degree does the technique improves performance even in the presence of unsafe (borderline, rare and outlier) examples. From Table 4.1, the most difficult dataset for a classifier to learn from is *hayes-roth*, this difficulty is due to the presence of more unsafe examples such as the borderline, rare and outlier examples than the safe example while the easiest to learn from is the *car* dataset since it contain only safe examples.

Table 4.1: Analysis of Data Difficulty Factors in Dataset

Description	Code	Safe (%)	Borderline (%)	Rare (%)	Outlier (%)
new-thyroid	NT	91	8	1	0
balance-scale	BS	93	6	1	0
cmc	CM	77	20	3	0
hayes-roth	HR	28	54	13	6
car	CA	100	0	0	0

4.1.2 Similarity degree and safe level of classes in the dataset

From the multiclass resampling technique, similarities degree of examples were used in the derivation of the safe level of examples. Firstly, the similarity degree between classes conducted on each of the five data sample used for this study is presented in Table 4.2.

Table 4.2: Dataset Classes Similarity Degree

Dataset Description	new-thyroid	balance-scale	cmc	hayes-roth	Car
Similarity Degree (μ)	μ_{12} = 0.21 μ_{13} = 0.83 μ_{23} = 0.39	μ_{12} = 0.99 μ_{13} = 0.99 μ_{23} = 0.99	μ_{12} = 1.00 μ_{13} = 0.99 μ_{23} = 1.00	μ_{12} = 0.21 μ_{13} = 0.83 μ_{23} = 0.39	μ_{12} = 0.91,μ_{41} = 0.78 μ_{13} = 1.00, μ_{42} = 0.99 μ_{23} = 0.09, μ_{43} = 1.00

In evaluating the performance of the proposed technique KNN, SVM and CART where used for the classification of 5 standard imbalanced dataset after balancing the original dataset. Table 4.3 presents the result of this new technique in terms of accuracy, precision, recall and F-score.

Table 4.3: Summary of Classifiers Result

Dataset Description	Classifier	Accuracy (%)	Precision	Recall	F-score
NT	KNN	96	0.97	0.96	0.96
	CART	100	1.00	1.00	1.00
	SVM	84	0.90	0.85	0.85
BS	KNN	96	0.97	0.96	0.96
	CART	100	1.00	1.00	1.00
	SVM	99	0.99	0.99	0.99
CM	KNN	91	0.91	0.92	0.92
	CART	100	1.00	1.00	1.00
	SVM	78	0.79	0.78	0.78
HR	KNN	72	0.72	0.75	0.72
	CART	99	0.99	0.99	0.99
	SVM	63	0.65	0.64	0.63
CA	KNN	100	1.00	1.00	1.00
	CART	100	1.00	1.00	1.00
	SVM	100	1.00	1.00	1.00

4.1.3 General classifier results analysis

The balanced dataset were classified using 5NN, CART and SVM, from analysis, the nature of dataset in terms of safe and unsafe examples really played a great role in the result as dataset with more safe examples performed generally better than the unsafe. From

the representation in Figure 4.1, despite this data nature, CART performed outstanding, for it was able to achieve a better result in respective of the data nature.

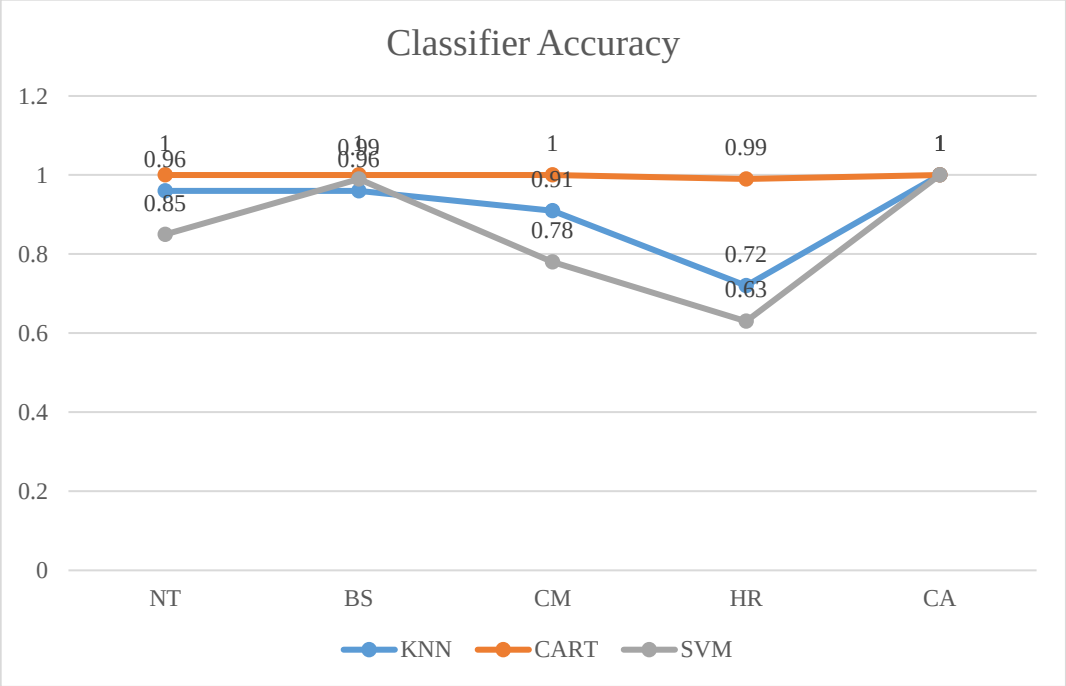

Figure 4.1: Accuracy of Classifiers on the Techniques

The F-Score is used to evaluate an algorithm's performance when dealing with imbalanced cases. It is the weighted average of the precision and recall measures, with false positives (FP) and false negatives (FN) taken into consideration. Figure 4.2 gives the analysis of the F-Score for the new techniques on KNN, CART and SVM classifiers. This analysis also shows that all the classifiers performed well but CART showed a near excellent performance.

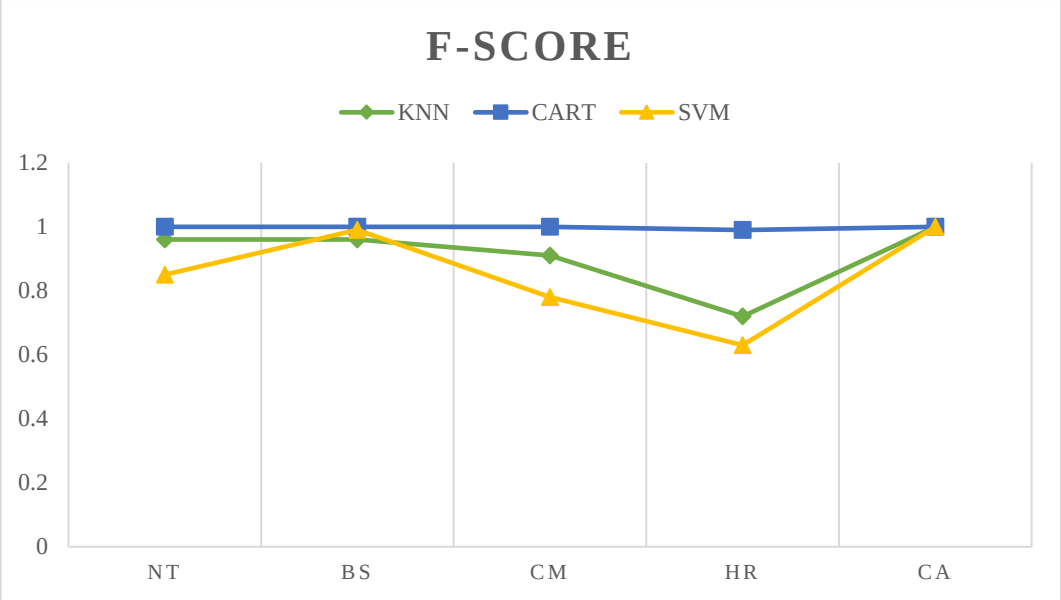

FIGURE 4.2: F-Score of Classifiers on the dataset

4.1.4 Analysis of AUC ROC curve for the technique

Figure 4.3 presents the Area Under Curve (AUC) Receiver Operating Characteristic (ROC) for K nearest neighbor classifier on the HR dataset with value 0.97. The ROC visualise the trade-off between TPR and FPR, with the range of the AUC being 0 to 1. The larger the AUC value, the better the performance of the technique. The diagrams for the AUC ROC curve the other datasets are presented in appendix A.

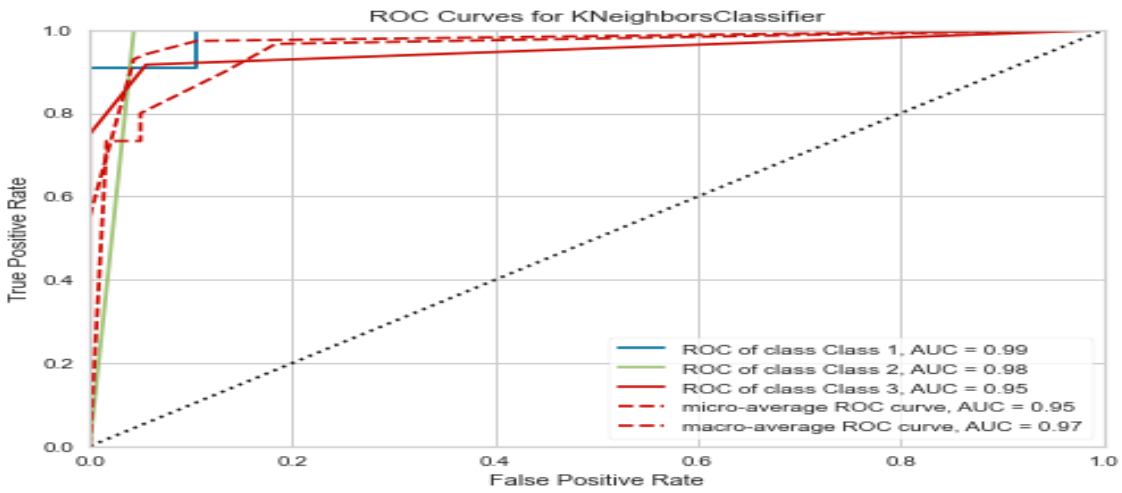

Figure 4.3: ROC Curve for KNN Classifier on hayes_roth Dataset

Figure 4.4 presents the Area Under Curve for the Receiver Operating Characteristic (ROC) for CART classifier on the HR dataset with value 0.94.

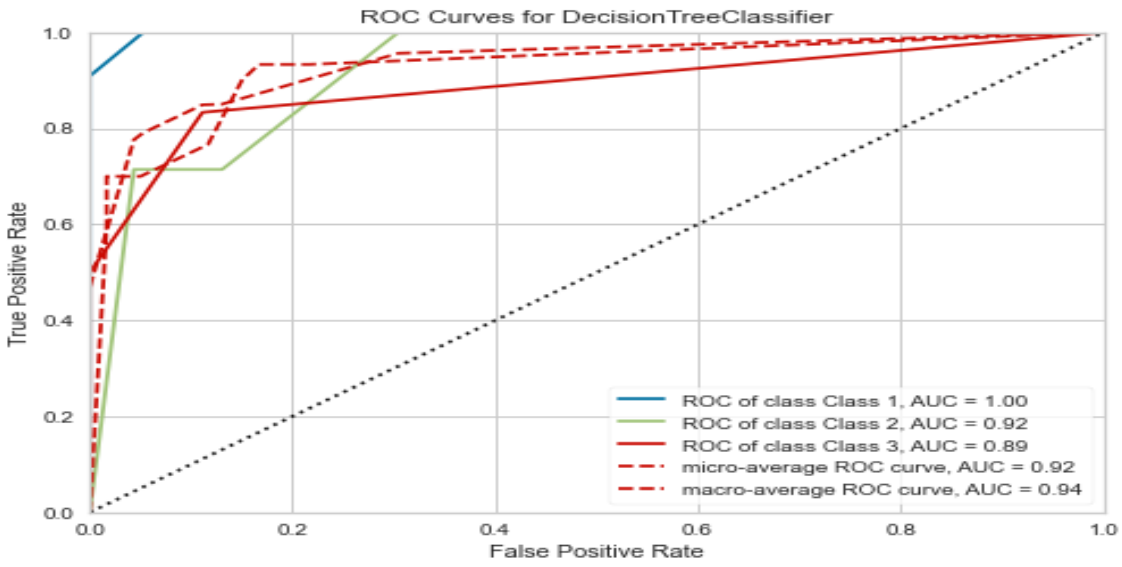

FIGURE 4.4: ROC Curve for CART Classifier on hayes_roth Dataset

Figure 4.5 presents the Area Under Curve for the Receiver Operating Characteristic (ROC) for SVM classifier on the HR dataset with value 0.84.

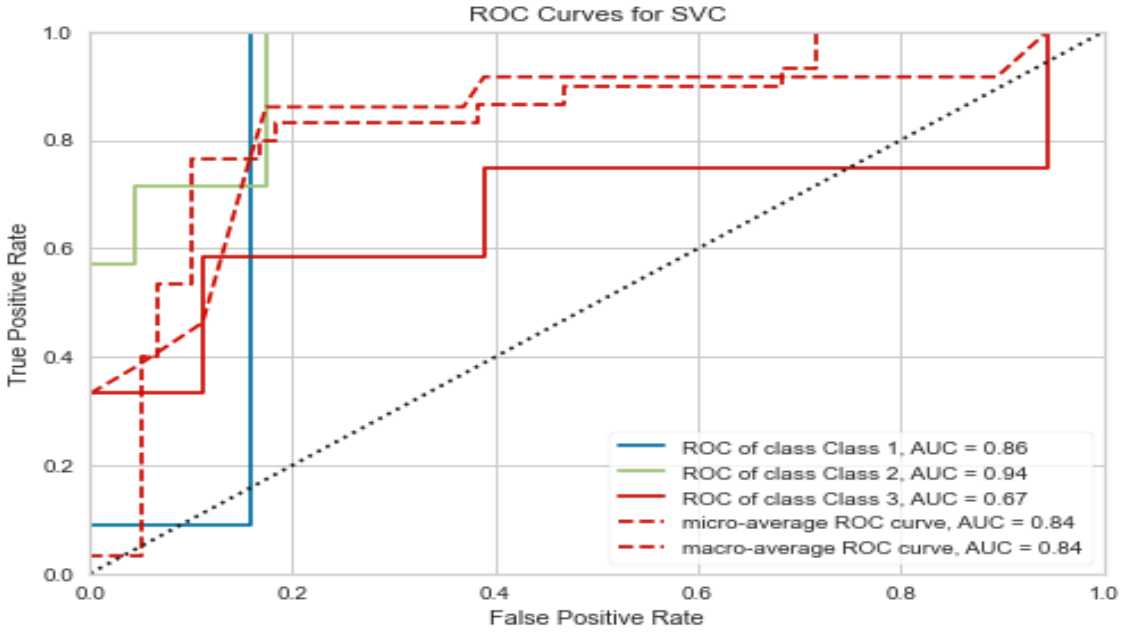

Figure 4.5: ROC Curve for SVM Classifier on hayes_roth Dataset

Table 4.4 presents the AUC ROC values for the new technique on the balanced dataset and classifiers used in this study.

Table 4.4: Summary of the AUC ROC for classifiers and datasets used

Data Description	MIRT		
	KNN	CART	SVM
NT	1.00	1.00	1.00
BS	0.90	0.94	1.00
CM	0.87	0.74	0.82
HR	0.97	0.94	0.84
CA	1.00	1.00	1.00

To further analyse the AUC ROC results for each dataset, the results are presented in Figure 4.6. The results generally shows that KNN performed better than CART and SVM. For the NT and CA datasets all the classifier showed an outstanding performance of 100 percentage. For the most difficult dataset HR, KNN also performed better.

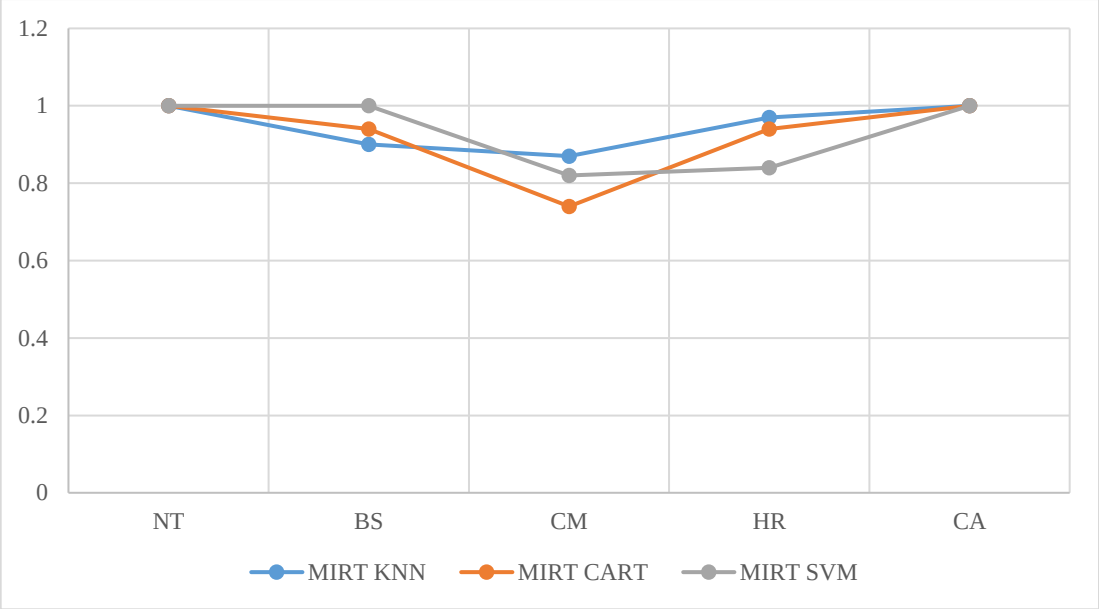

Figure 4.6: Summary of the AUC ROC results for all dataset used

4.2 Discussion of Result

Firstly, from this point on, the proposed technique is referred to as Multiclass Informative Preprocessing Technique (MIRT). The evaluation of MIRT have been done analysing the accuracy, precision, recall, F-score, G-mean and AUC ROC curve results.

Furthermore, from the work of Janicka *et al.* (2019) and De and Do (2020), whose techniques are been use to compare MIRT, it was observed that G-mean value was their main criteria for comparism. MIRT will be compared with some of the recent best performing methods such as SOUP, Multi-class Roughly Balanced Bagging (MRBB) and SOUPBag. From Table 4.5, it is observed that MIRT technique performed better than SOUP, SOUPBag and MRBB most especially when used with the CART and KNN classifiers.

Table 4.5: G-Mean Comparism of MIRT with SOUP, SOUPBag and MRBB.

Data		MIRT		MRBB	MRBB	SOUP	SOUPBag
Description	KNN	CART	SVM	KNN	DT	J4.8	KNN
NT	0.983	**1.00**	0.883	0.730	0.977	0.922	0.897
BS	0.971	**1.00**	0.995	0.704	0.637	0.585	0.750
CM	0.933	**0.985**	0.837	0.545	0.531	0.535	0.485
HR	0.931	**1.000**	0.640	N/A	N/A	0.835	N/A
CA	**1.000**	**1.000**	**1.000**	0.957	0.811	0.941	0.851

Figure 4.7 compares the performance of MIRT with SOUP, SOUPBag and MRBB. MIRT balanced dataset was classified using KNN, SVM and CART, with KNN and CART performing better that SVM, even with the HR dataset, that has greater percentage of unsafe examples than the safe, CART performance was still outstanding compare to SVM and KNN. From the G-mean result, KNN was used in the classification of all dataset from all techniques, MIRT KNN results has an overall better performance than the other techniques.

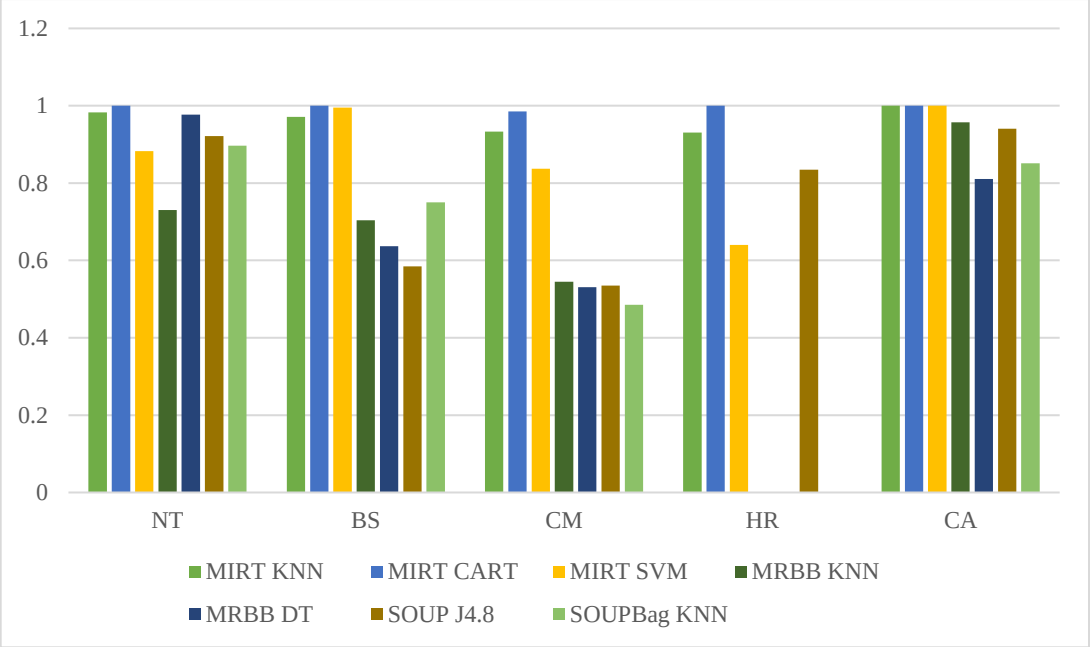

Figure 4.7: Comparism of MIRT with SOUP, SOUPBag and MRBB

From the results obtained by running the MIRT balanced dataset on the three classifiers and comparing it with some of the best methods, as an example, consider the following:

i. Overall performance of MIRT is better in all datasets used, because of its performance even in the presence of unsafe examples, just as in HR.

ii. ii. Using Decision Trees (CART) as a classifier appears to perform better with MIRT, owing to its superior performance when compared to KNN and SVM.

4.2.1 Experimentation

This presents the step by step procedure used to implement MIRT resampling technique. Firstly, some python libraries were imported into Jupiter notebook, which is the python interactive environment used for the implementation of the algorithm. The libraries includes: Numpy, Pandas, Scikit-learn, Imblearn, Seaborn, Yellowbrick and Matplotlib.

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the python programming language. All manipulation of

the dataset was carried it using the pandas library. Numpy is a fast and versatile python numeric computing library for arithmetic and logical operation in the implementation. Scikit-learn is a powerful data prediction library for machine learning process, it was built on Numpy, Matplotlib, Scipy. It is the most used python data prediction library for machine learning process. The predictions of this work was done using sklearn library. Imbalanced-learn (imported as imblearn) is an open source, MIT-licensed library relying on scikit-learn (imported as sklearn) and provides tools when dealing with classification with imbalanced classes. Seaborn, Yellowbrick and Matplotlib are all python data visualisation library, use for data visualisation purposes.

After importing all necessary python library, the dataset was also read into the computing environment. Next, missing attributes are search for in the dataset and were filled using already existing approach in pandas. It was proceeded by randomly selecting twenty (20) samples from the dataset which were evaluated to deduce the data similarity degree between classes. The steps for evaluation the data similarity degree was presented in session 3.4.2.1.

Next, the five (5) nearest neighbors of all the examples were evaluated using sklearn.neighbors. From the result of the evaluation, each data sample was grouped into safe, borderline, rare and outlier difficulty type. This was proceeded by using the data difficulty information and data similarity degree information to evaluate the safe level of examples.

After successfully evaluating the safe level of all examples in the dataset, the minority classes are informatively oversampled by duplicating the examples in the dataset with the higher safe level value while the majority class(es) are also informatively undersample by removing the examples with the least safe level until the required threshold is acquired; which then produces a balanced dataset. Classification was done on the balanced dataset using KNN, SVM and CART classifier. The results were computed on different visualisation library for better representation. The following metrics were used to evaluate the performance of the algorithm: precision, recall, f1-score, AUC ROC, G-mean; which were already implemented in sklearn.

CHAPTER FIVE

5.0 CONCLUSION AND RECOMMENDATIONS

5.1 Summary

This research work developed a multiclass resampling technique using class similarities degree and data difficulty factors information from the dataset. The data difficulty and similarity information's were implemented using KNN to determine the neighbours of each example in the dataset and also the distance between each example and its neighbours. The information about the neighbours of each example was used to derive their difficulty type (safe and unsafe). It went further to select 20 samples from each class in the dataset which were evaluated to derive the similarity degree between classes. Finally, the similarities degree and difficulty type of each example were used to evaluate the safe level of examples; which then served as the criteria for selecting the oversampling and undersampling examples. This proposed technique was tested on 5 standard imbalanced dataset, which were selected based on their difficulty level. After resampling the dataset, classification of the dataset was done using KNN, SVM and CART classifier. The performance of the proposed technique, MIRT on CART classifier which achieved a 100 percentage in 4 of the 5 data samples used was better than SOUP, SOUPBag and MRBB resampling techniques which were compared using the G-mean values. Also, among the claasifiers used, CART performed way better than KNN and SVM. The performance of the proposed technique was outstanding when compared with Similarity Oversampling and Undersampling Preprocessing SOUP resampling technique.

5.2 Conclusions

With the study of similarities degree, data intrinsic characteristic and multiclass imbalanced nature of dataset the following conclusion were drawn:

The aim of this study was to develop a multiclass resampling technique using class similarities degree and data difficulty factor. This was achieved by using KNN to determine the neighbours of each examples, these examples are further categorised as safe, borderline, rare and outliers based on the class of its neighbours. Next, 20 samples from each class in the dataset were evaluated to derive the similarity degree between classes. These data difficulty type of each example and similarity information were used to evaluate the safe level of each example in the dataset; the safe level of example serves as the criteria for selecting the oversampling and undersampling examples.

This proposed technique was implemented using Jupiter notebook, a python interactive computing environment and other python libraries which includes pandas, yellowbrick, seaborn, sklearn, matplotlib, numpy. Multiclass Informative Resampling Technique (MIRT) was evaluated using standard parameters such as F-Score, AUC **(**Area Under ROC Curve**)** and Geometric Mean (G-Mean). These evaluation metrics where also implemented in python.

Finally, MIRT was compared with SOUP, SOUPBag and MRBB resampling technique and was found to outperforming them all; most especially the result from CART classifier with outstanding accuracy for all datasets used. From the comparism presented in Table 4.3, it was observed that MIRT performed better than SOUP, SOUPBag and MRBB due to its outstanding performance in the presence of complex unsafe data difficulty factors.

The MIRT increase the rate of detection of the minority class in the presence of complex data difficulty factor and also increased classifier performance which is very key in building a resampling technique.

5.3 Contributions to Knowledge

The contributions reached at the end of this work could be summarised as:

(i) Design of an approach for evaluating the degree of similarities between classes in multiclass imbalanced dataset.

(ii) Development of a new resampling technique for informatively identifying the appropriate examples to oversample and undersample.

5.4 Recommendations

The main achievements of this study were highlighted in the section above, however, some areas of research that can be further explore are:

(i) This method for evaluating class similarity degree should also be applied on binary preprocessing technique.

(ii) New multiclass resampling technique design should put into consideration data intrinsic characteristic and inter-relationship between classes and examples; avoiding uninformative random oversampling and random undersampling.

(iii) The design method for evaluating the similarity degree between classes can be used with other resampling methods such as SMOTE, ADASYN to improve performance.

REFERENCES

Alejo, R., Monroy-de-Jesús, J., Ambriz-Polo, J. C., & Pacheco-Sánchez, J. H. (2017). An improved dynamic sampling back-propagation algorithm based on mean square error to face the multi-class imbalance problem. *Neural Computing and Applications*, *28*(10), 2843–2857. Retrieved from https://doi.org/10.1007/s00521-017-2938-3

Ali, H., Salleh, M. N. M., Saedudin, R., Hussain, K., & Mushtaq, M. F. (2019). Imbalance class problems in data mining: A review. *Indonesian Journal of Electrical Engineering and Computer Science*, *14*(3), 1552–1563.

Blaszczynski, J., & Lango, M. (2016). Diversity Analysis on Imbalanced Data Using Neighbourhood and Roughly Balanced. *International Conference on Artificial Intelligence and Soft Computing* , 552–562.

Błaszczyński, J., & Stefanowski, J. (2018). Local data characteristics in learning classifiers from imbalanced data. *Studies in Computational Intelligence*, 738, 51-85.

Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, *106*, 249–259. Retrieved from https://doi.org/10.1016/j.neunet.2018.07.011

Cruz, R. M. O., Souza, M. A., Sabourin, R., & Cavalcanti, G. D. (2019). Dynamic Ensemble Selection and Data Preprocessing for Multi-Class Imbalance Learning. *International Journal of Pattern Recognition and Artificial Intelligence*, *33*(11).

De, A., & Do, N. (2020). Techniques to deal with imbalanced data in multi-class problems : A review of existing methods. *Mestrado Integrado em Engenharia Informática e Computação*, 5(2).

Duan, H., Wei, Y., Liu, P., & Yin, H. (2020). A novel ensemble framework based on K-means and resampling for imbalanced data. *Applied Sciences (Switzerland)*, *10*(5), 1684.

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2019). Learning from imbalanced data. *Studies in Computational Intelligence* (807), 81-110.

Fernández, A., López, V., Galar, M., Del Jesus, M. J., & Herrera, F. (2013). Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-Based Systems*, *42*, 97–110.

Fernández, A., Salvador, G., Galar, M., Prati C, R., Krawczyk, B., & Herrera, F. (2018). Learning from imbalanced data Sets. *IEEE Transactions on Knowledge and Data Engineering* (21), 978.

Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, *44*(8), 1761–1776.

García, S., Zhang, Z. L., Altalhi, A., Alshomrani, S., & Herrera, F. (2018). Dynamic ensemble selection for multi-class imbalanced datasets. *Information Sciences*, *445*, 22–37.

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, *73*, 220–239. Retrieved from http://dx.doi.org/10.1016/j.eswa.2016.12.035

He, H., & Garcia, E. A. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284.

Hossen, I., Goh, M., Connie, T., & Aris, A. (2018). Combining Sampling and Ensemble Classifier for Multiclass Imbalance Data Learning, *Knowledge-Based Systems 3*(2), 241–251.

Janicka, M. A., Lango, M. A., & Stefanowski, J. E. (2019). Using Information On Class Interrelations To Improve Classification Of Multiclass Imbalanced Data : A New Resampling Algorithm. *International Journal of Applied Mathematics and Computer Science.*, *29*(4), 769–781.

Jedrzejowicz, J., Kostrzewski, R., & Neumann, J. (2018). Imbalanced data classification using MapReduce and relief. *Journal of Information and Telecommunication*, 2(2), 217-230.

Kamalov, F., & Denisov, D. (2020). Gamma distribution-based sampling for imbalanced data. *Knowledge-Based Systems*, *207*, 106368. Retrieved from https://doi.org/10.1016/j.knosys.2020.106368

Kaur, H., Pannu, H. S., & Malhi, A. K. (2019). A Systematic Review on Imbalanced Data Challenges in Machine Learning : Applications and Solutions. *ACM Computing Surveys (CSUR)*, *52*(4), 1-36.

Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2011). Comparing boosting and bagging techniques with noisy and imbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, *41*(3), 552–568.

Koziarski, M., Woźniak, M., & Krawczyk, B. (2020). Combined Cleaning and Resampling algorithm for multi-class imbalanced data with label noise. *Knowledge-Based Systems*, *204*, 106223. Retrieved from https://doi.org/10.1016/j.knosys.2020.106223

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

Lango, M., Napierala, K., & Stefanowski, J. (2017). Evaluating Difficulty of Multi-class Imbalanced Data. *International Symposium on Methodologies for Intelligent Systems*, 312–322.

Lango, M., & Stefanowski, J. (2018). Multi-class and feature selection extensions of Roughly Balanced Bagging for imbalanced data. *Journal of Intelligent Information*

Systems, *50*(1), 97–127.

Lin, W., Tsai, C., Hu, Y., & Jhang, J. (2017). Clustering-based undersampling in class-imbalanced data. *Information Sciences*, *410*, 17–26.

Liu, Y., Wang, Y., Ren, X., Zhou, H., & Diao, X. (2019). A Classification Method Based on Feature Selection for Imbalanced Data. *IEEE Access*, *7*, 81794–81807.

Mahmoud, A., El-Kilany, A., Ali, F., & Mazen, S. (2020). A Novel Oversampling Technique To Handle Imbalanced Datasets. *Communications of the ECMS*, 177–182.

Max, R., Hannah, R., & Ortiz-Ospina, E. (2020). Coronavirus Disease (COVID-19) – Research and Statistics. Retrieved from *OurWorldInData.org*

Nannes, B. B., Quax, R., Ashikaga, H., Bernus, O., & Ha, M. (2020). Computational Science. *Twenty Years of Computational Science. Lecture Notes in Computer Science*, (12140). Retrieved from http://link.springer.com/10.1007/978-3-030-50423-6

Napierala, K., & Stefanowski, J. (2012). Identification of Different Types of Minority Class Examples in Imbalanced Data. *International Conference on Hybrid Artificial Intelligence Systems* , 139–150.

Napierala, K., & Stefanowski, J. (2016). Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, *46*(3), 563–597. Retrieved from http://dx.doi.org/10.1007/s10844-015-0368-1

Napierała, K., Stefanowski, J., & Wilk, S. (2010). Learning from imbalanced data in presence of noisy and borderline examples. *International conference on rough sets and current trends in computing*, *6086 LNAI*, 158–167.

Nwe, M. M., & Lynn, K. T. (2020). Effective resampling approach for skewed distribution on imbalanced data set. *IAENG International Journal of Computer Science*, *47*(2), 234–249.

Rendón, E., Alejo, R., Castorena, C., Isidro-Ortega, F. J., & Granda-Gutiérrez, E. E. (2020). Data sampling methods to deal with the big data multi-class imbalance problem. *Applied Sciences (Switzerland)*, *10*(4), 1276.

Sáez, J. A., Krawczyk, B., & Woźniak, M. (2016). Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. *Pattern Recognition*, *57*, 164–178. Retrieved from http://dx.doi.org/10.1016/j.patcog.2016.03.012

Sun, Y., Wong, A. K. C., & Kamel, M. S. (2009). Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, *23*(04), 687–719.

Thabtah, F., Hammoud, S., & Kamalov, F. (2019). Data Imbalance in Classification : Experimental Evaluation. *Information Sciences*, 513, 429-441.

Vluymans, S., Fernández, A., Saeys, Y., Cornelis, C., & Herrera, F. (2018). Dynamic affinity-based classification of multi-class imbalanced data with one-versus-one decomposition: a fuzzy rough set approach. *Knowledge and Information Systems*, *56*(1), 55–84.

Wang, S., & Yao, X. (2012). Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *42*(4), 1119–1130.

Weiss, G. M. (2013). Foundations of Imbalanced Learning. *Imbalanced Learning: Foundations, Algorithms, and Applications*, 13-41.

Wojciechowski, S., Wilk, S., & Stefanowski, J. (2018). An Algorithm for Selective Preprocessing of Multi-class Imbalanced Data. *International Conference on Computer Recognition Systems*, 238-247.

APPENDICES

APPENDIX A: AUC Curve for the datasets

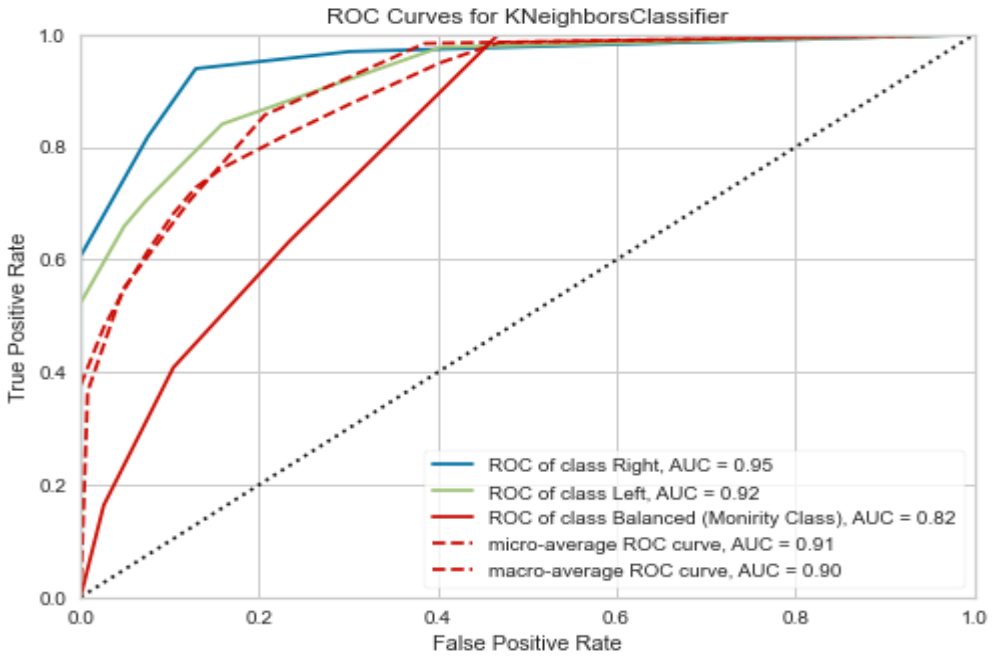

FIGURE 1: ROC Curve for KNN Classifier on balance-scale Dataset

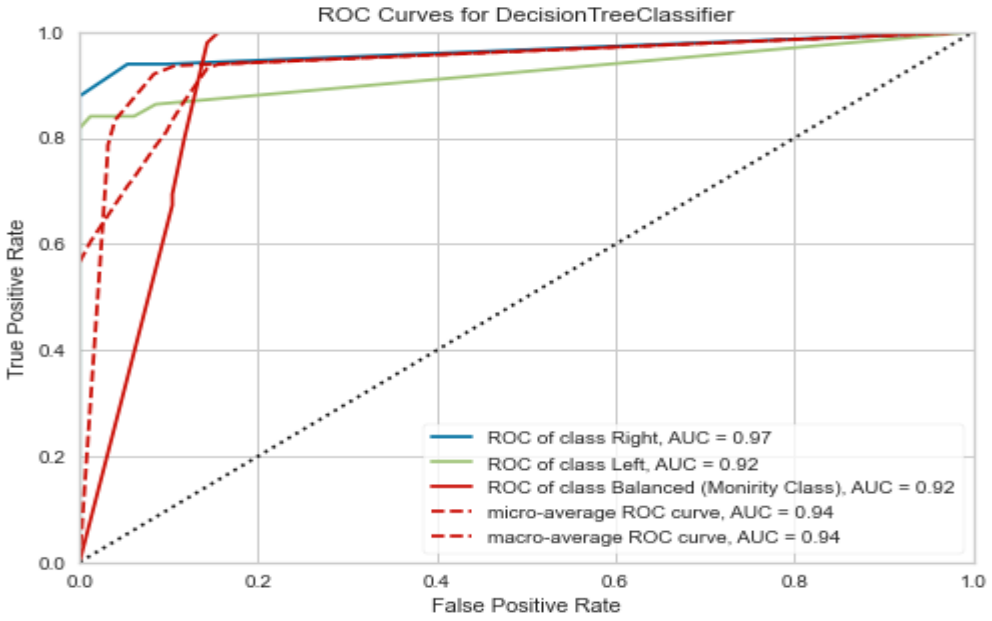

FIGURE 2: ROC Curve for CART Classifier on balance-scale Dataset

APPENDIX B: Imbalanced data for hayes-roth dataset

S/No	Class	Oxin	Onine	Hormone	Tsh	Resin
1	1	92	2	1	1	2
2	1	36	2	2	1	1
3	1	105	3	2	1	1
4	1	81	1	2	1	1
5	1	94	1	1	2	1
6	1	20	1	1	3	3
7	1	50	1	2	1	1
8	1	68	3	3	2	1
9	1	89	3	1	3	2
10	1	19	3	2	1	3
11	1	118	2	1	2	1
12	1	16	3	2	1	3
13	1	91	2	3	2	1
14	1	30	1	1	3	2
15	1	57	3	2	1	1
16	1	114	2	2	1	3
17	1	66	1	1	1	2
18	1	74	3	2	1	1
19	1	106	3	1	2	1
20	1	130	2	1	1	2
21	1	54	1	1	1	2
22	1	67	3	3	1	1
23	1	69	3	3	3	1
24	1	127	3	1	2	1
25	1	96	1	1	1	2
26	1	121	2	1	3	2
27	1	123	2	1	2	1
28	1	42	2	2	1	3
29	1	5	1	3	2	1
30	1	95	2	3	2	1
31	1	119	3	1	3	2
32	1	93	2	1	2	1
33	1	132	2	2	1	1
34	1	108	1	1	2	1
35	1	120	1	1	3	2
36	1	35	1	2	1	3
37	1	112	1	1	1	3
38	1	59	1	1	1	2
39	1	1	3	2	1	1
40	1	28	1	1	2	1
41	1	97	2	1	3	1
42	1	51	3	1	1	2

43	1	103	2	2	1	1
44	1	7	1	2	1	1
45	1	15	1	3	2	1
46	1	126	3	1	2	1
47	1	45	3	1	1	2
48	1	131	2	3	1	3
49	1	17	2	1	1	2
50	1	40	2	1	2	1
51	1	9	3	1	1	2
52	2	10	2	1	3	2
53	2	113	1	1	3	2
54	2	80	3	1	3	2
55	2	60	2	1	2	2
56	2	85	3	2	1	2
57	2	52	1	2	2	1
58	2	79	3	2	2	1
59	2	23	3	2	1	3
60	2	25	2	1	2	2
61	2	37	1	2	1	3
62	2	116	3	1	2	2
63	2	88	1	1	2	2
64	2	77	3	2	2	1
65	2	82	1	2	1	2
66	2	84	2	2	2	1
67	2	86	2	2	1	2
68	2	6	3	2	1	3
69	2	115	1	2	1	3
70	2	33	1	2	2	3
71	2	39	3	2	1	2
72	2	53	3	2	1	2
73	2	70	2	2	2	1
74	2	78	2	1	2	2
75	2	129	2	2	1	2
76	2	73	3	1	2	2
77	2	26	1	1	2	2
78	2	104	1	1	2	2
79	2	2	2	1	3	2
80	2	41	1	1	3	2
81	2	62	3	1	2	2
82	2	98	3	3	3	2
83	2	109	2	2	1	3
84	2	31	3	3	2	1
85	2	34	2	2	1	2
86	2	63	2	2	2	1
87	2	65	2	3	2	3
88	2	117	1	3	2	1

89	2	56	2	2	1	2
90	2	76	3	2	2	1
91	2	29	3	3	2	1
92	2	111	2	3	2	1
93	2	49	1	2	1	2
94	2	58	1	2	2	1
95	2	32	2	3	2	1
96	2	99	2	2	3	2
97	2	24	1	2	3	3
98	2	124	3	3	2	2
99	2	14	1	2	2	1
100	2	71	3	1	2	2
101	2	90	1	2	1	2
102	2	21	1	2	2	1
103	3	83	3	1	4	1
104	3	61	2	4	2	2
105	3	107	1	1	3	4
106	3	125	3	4	2	4
107	3	122	2	2	3	4
108	3	8	2	4	1	4
109	3	3	1	4	1	1
110	3	110	2	4	3	1
111	3	64	3	4	3	2
112	3	11	1	2	4	2
113	3	128	1	1	2	4
114	3	4	2	4	4	2
115	3	48	1	3	2	4
116	3	102	3	1	4	2
117	3	75	1	2	4	4
118	3	47	1	4	2	1
119	3	46	3	4	1	2
120	3	18	2	2	4	3
121	3	27	1	4	4	1
122	3	22	3	1	4	4
123	3	87	2	2	4	1
124	3	72	2	2	1	4
125	3	55	1	4	2	3
126	3	101	3	3	1	4
127	3	100	2	3	4	1
128	3	13	3	3	4	2
129	3	38	2	1	1	4
130	3	43	3	2	2	4
131	3	12	3	4	1	3
132	3	44	1	1	4	3

APPENDIX C: Balanced data for hayes-roth dataset

S/No	Class	Oxin	Onine	Hormone	Tsh	Resin
0	1	94	1	1	2	1
1	1	96	1	1	1	2
2	1	93	2	1	2	1
3	1	120	1	1	3	2
4	1	94	1	1	2	1
5	1	96	1	1	1	2
6	1	93	2	1	2	1
7	1	94	1	1	2	1
8	1	94	1	1	2	1
9	1	95	2	3	2	1
10	1	96	1	1	1	2
11	1	96	1	1	1	2
12	1	94	1	1	2	1
13	1	94	1	1	2	1
14	1	95	2	3	2	1
15	1	93	2	1	2	1
16	1	94	1	1	2	1
17	1	95	2	3	2	1
18	1	95	2	3	2	1
19	1	120	1	1	3	2
20	1	120	1	1	3	2
21	1	96	1	1	1	2
22	1	94	1	1	2	1
23	1	94	1	1	2	1
24	1	120	1	1	3	2
25	1	96	1	1	1	2
26	1	94	1	1	2	1
27	1	94	1	1	2	1
28	1	94	1	1	2	1
29	1	93	2	1	2	1
30	1	93	2	1	2	1
31	1	120	1	1	3	2
32	1	96	1	1	1	2
33	1	93	2	1	2	1
34	1	120	1	1	3	2
35	1	96	1	1	1	2
36	1	120	1	1	3	2
37	1	94	1	1	2	1
38	1	95	2	3	2	1
39	1	94	1	1	2	1
40	2	76	3	2	2	1
41	2	32	2	3	2	1
42	2	25	2	1	2	2

43	2	85	3	2	1	2
44	2	23	3	2	1	3
45	2	23	3	2	1	3
46	2	60	2	1	2	2
47	2	78	2	1	2	2
48	2	115	1	2	1	3
49	2	80	3	1	3	2
50	2	85	3	2	1	2
51	2	76	3	2	2	1
52	2	84	2	2	2	1
53	2	32	2	3	2	1
54	2	76	3	2	2	1
55	2	31	3	3	2	1
56	2	60	2	1	2	2
57	2	80	3	1	3	2
58	2	85	3	2	1	2
59	2	79	3	2	2	1
60	2	32	2	3	2	1
61	2	79	3	2	2	1
62	2	82	1	2	1	2
63	2	25	2	1	2	2
64	2	32	2	3	2	1
65	2	79	3	2	2	1
66	2	32	2	3	2	1
67	2	31	3	3	2	1
68	2	25	2	1	2	2
69	2	78	2	1	2	2
70	2	77	3	2	2	1
71	2	86	2	2	1	2
72	2	24	1	2	3	3
73	2	84	2	2	2	1
74	2	76	3	2	2	1
75	2	60	2	1	2	2
76	2	115	1	2	1	3
77	2	60	2	1	2	2
78	2	80	3	1	3	2
79	2	115	1	2	1	3
80	3	72	2	2	1	4
81	3	44	1	1	4	3
82	3	22	3	1	4	4
83	3	38	2	1	1	4
84	3	83	3	1	4	1
85	3	100	2	3	4	1
86	3	38	2	1	1	4
87	3	75	1	2	4	4
88	3	100	2	3	4	1

89	3	72	2	2	1	4
90	3	128	1	1	2	4
91	3	64	3	4	3	2
92	3	22	3	1	4	4
93	3	128	1	1	2	4
94	3	12	3	4	1	3
95	3	102	3	1	4	2
96	3	13	3	3	4	2
97	3	75	1	2	4	4
98	3	11	1	2	4	2
99	3	64	3	4	3	2
100	3	44	1	1	4	3
101	3	3	1	4	1	1
102	3	55	1	4	2	3
103	3	110	2	4	3	1
104	3	128	1	1	2	4
105	3	102	3	1	4	2
106	3	18	2	2	4	3
107	3	125	3	4	2	4
108	3	18	2	2	4	3
109	3	48	1	3	2	4
110	3	64	3	4	3	2
111	3	72	2	2	1	4
112	3	47	1	4	2	1
113	3	128	1	1	2	4
114	3	100	2	3	4	1
115	3	4	2	4	4	2
116	3	12	3	4	1	3
117	3	72	2	2	1	4
118	3	4	2	4	4	2
119	3	72	2	2	1	4

APPENDIX D: Code for the Resampling Technique

```
import numpy as np
import pandas as pd
import math
import random
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_predict
from sklearn import metrics
from sklearn import svm
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.dummy import DummyClassifier
# Read dataset
data=pd.read_csv("new-thyroid.csv")
# Add heading column to dataset
data.columns=["id","Class","Oxin","Onine","Hormone","Tsh","Resin"]
# Use id as index
data.set_index("id", inplace=True)
# Group data by Class
gb=data.groupby("Class")
for Class, Class_df in gb:
    print(Class)
    print(Class_df)
gb.get_group(1)
# Select 15 sample from the dataset
```

```
class_samples=gb.get_group(1).sample(n=20)

print(type(class_samples))

# Take the mean of the column in the selected 20 samples and evaluation

mean_sum=class_samples["Oxin"].mean() + class_samples["Onine"].mean() + class_samples["Hormone"].mean() + class_samples["Tsh"].mean() + class_samples["Resin"].mean()

column_length=len(data.columns) -1

similar_val_1=mean_sum/column_length

print(similar_val_1)

# Take the mean of the column in the selected 20 samples and evaluation

class_2_samples=gb.get_group(2).sample(n=20)

mean_sum_2=class_2_samples["Oxin"].mean() + class_2_samples["Onine"].mean() + class_2_samples["Hormone"].mean() + class_2_samples["Tsh"].mean() + class_2_samples["Resin"].mean()

similar_val_2=mean_sum_2/column_length

print(similar_val_2)

# Take the mean of the column in the selected 20 samples and evaluation

class_3_samples=gb.get_group(3).sample(n=20)

mean_sum_3=class_3_samples["Oxin"].mean() + class_3_samples["Onine"].mean() + class_3_samples["Hormone"].mean() + class_3_samples["Tsh"].mean() + class_3_samples["Resin"].mean()

similar_val_3=mean_sum_3/column_length

print(similar_val_3)

#evaluating

sim_23=similar_val_3 - similar_val_2

sim_23=sim_23/10

sim_23=1 - sim_23

sim_12=similar_val_1 - similar_val_2

sim_12=sim_12/10

sim_12=1 - sim_12

sim_13=similar_val_3 - similar_val_1

sim_13=sim_13/10

sim_13=1 - sim_13

print("Similarity between 2 and 3: ", sim_23)
```

```
print("Similarity between 2 and 1: ", sim_12)
print("Similarity between 1 and 3: ", sim_13)
m=math.ceil((149+35)/2)
len(gb)
# Take the neighbors of each samples
neigh = KNeighborsClassifier(n_neighbors=5)
#neigh.fit(X_train, Y_train)
neigh.fit(data, data.Class)
result=neigh.kneighbors(data, return_distance=True)
print(type(result))
print(result)
arr1=result[0]
print(type(arr1))
print(arr1.ndim)
arr2=result[1]
print(type(arr2))
print(arr1.ndim)
print(arr2)
print(arr1)
print(arr1.size)
addVal=[]
s=1
for i in arr1:
    _sum=int(i[0] + i[1] + i[2] + i[3] + i[4])
    addVal.append(_sum)
    print(i ," ", _sum, " ")
    s=s+1
print(len(addVal))
print(addVal)
arr3=np.array(addVal)
print(type(arr3))
print(arr3.size)
```

```
print(arr3)
g=1
for i in addVal:
    print(g ," ", i)
    g=g+1
newarr = arr3.reshape(214, 1)
print(newarr.ndim)
new_arr2=np.append(arr2, newarr, axis=1)
print(type(new_arr2))
print(new_arr2)
print(new_arr2[0])
arr2
print(new_arr2[0][1],  new_arr2[0][2],  new_arr2[0][3], new_arr2[0][4],new_arr2[0][5])
# Creating and populating a dictionary
new_dist={}
for i, v in data.Class.iteritems():
    new_dist[i]=v
print(new_dist)
# Creating and populating a dictionary
new_dist2={}
print(new_dist2)
print(new_dist3)
# <!-- Populating a dictionary  -->

neigh_count.append([i,new_dist3[i],class_example,c1,c2,c3,safe_level,new_arr2[i][5]])
new_main_df = pd.DataFrame(neigh_count,columns=['s_no','real_index','Class','c_one','c_two','c_three','safe_level','distance'])
new_main_df.set_index("s_no", inplace=True)
print(new_main_df)
for i, row in enumerate(new_main_df.itertuples(), 0):
    print(i, row.real_index)
new_classes=new_main_df.groupby("Class")
```

```
s=1
# In[146]:
#Loading the Classifier model
model=svm.SVC(C = 1, probability=True)
clf = DecisionTreeClassifier(random_state=0, max_depth=3)
knn = KNeighborsClassifier(n_neighbors=5)
#Train the model using the training sets
model.fit(X_train, Y_train)
clf = clf.fit(X_train, Y_train)
knn = knn.fit(X_train, Y_train)
#prule_pred_proba=prule.predict_proba(X_test)
model_pred_proba=model.predict_proba(X_test)
clf_pred_proba=clf.predict_proba(X_test)
knn_pred_proba=knn.predict_proba(X_test)
model_X_train_pred=model.predict(X_train)
model_X_test_pred=model.predict(X_test)
clf_X_train_pred=clf.predict(X_train)
clf_X_test_pred=clf.predict(X_test)
knn_X_train_pred=knn.predict(X_train)
knn_X_test_pred=knn.predict(X_test)
print("Accuracy for SVM Test Data:                    ", metrics.accuracy_score(Y_test,
model_X_test_pred))
print("Accuracy for Decision Tree Classifier Test Data:    ", clf.score(X_test, Y_test,
sample_weight=None))
print("Accuracy for KNN Test Data:                    ", knn.score(X_test, Y_test,
sample_weight=None))
cm_model_test=confusion_matrix(Y_test, model_X_test_pred)
```