OVERLAPPING CLASSES IN IMBALANCED DATASETS

OVERLAPPING CLASSES IN IMBALANCED DATASETS

BY

WALEED A. ALMUTAIRI, M.Sc., B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Doctor of Philosophy (2023) McMaster University

(Computing and Software) Hamilton, Ontario, Canada

TITLE:	Overlapping classes in imbalanced datasets
AUTHOR:	Waleed A. Almutairi M.Sc., (Computer Science), NYU Tandon School of Engineering, NY, USA
SUPERVISOR:	Dr. Ryszard Janicki
NUMBER OF PAGES:	xv, 150

Abstract

Big data has become easily available, but there is a need to improve the usefulness of these data, especially when we have an imbalanced dataset and overlapping data points in two or more classes. Machine-learning algorithms have improved in recent years, and many algorithms have been introduced that tackle the issues in data that suffer from imbalanced classes and have overlap in some features.

This will be a problem used to train a classifier in deciding where each data point belongs. Such a situation often occurs when the number of examples that we are interested in is much less in number than the other classes. We can see problems of this kind in many fields, like for example, fraud detection, cancer diagnosis, oil mining, network intrusion, and many others. In this thesis, we will discuss the cases of datasets that are imbalanced and overlapping in some data points. The main problem to be dealt with is how to make a better judgment regarding the gray area between the minority class and the majority class and the overlap between the two. We will provide characteristics of the imbalanced dataset scenarios in the classification phase and then try to provide a better solution. Then, we will discuss the cost of the learning process together with algorithms and techniques for solving these issues.

To my family

Acknowledgements

First and foremost, I wish to start by expressing my sincere gratitude to Allah, who has granted countless blessing, knowledge, and opportunity to the writer, so that I have been finally able to accomplish the thesis. I would like to express my sincere gratitude to my advisor Prof. Ryszard Janicki for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study. I am very thankful to the members of my supervisory committee: Dr. Franya Franek and Dr. Fei Chiang for their helpful suggestions, instructions, discussions, and insightful comments on my research. I would also like to acknowledge my sponsor, King Abdulaziz City for Science and Technology (KACST).

Finally, I would like to express my deepest acknowledgments to my dear family for their support, and encouragement.

Contents

List of Figures

List of Tables

Abbreviations

Abbreviations

KNN	k-Nearest Neighbors
SVDD	Support Vector data description
SMOTE	Synthetic Minority Oversampling Technique
B-SMOTE	Borderline-SMOTE.
ADASYN	Adaptive Synthetic sampling
SVM	Support Vector Machine
ENN	Ensemble Neural Networks.
T-Link	Tomek link.
ROC	Receiver operating characteristic curve.
AUC	Area under the ROC Curve.
ML	Machine learning.

SL	Supervised Learning.
USL	UnSupervised Learning.
DBSCAN	Density-based spatial clustering of applications with noise.
SSL	Semi-Supervised Learning.
RL	Reinforcement learning.
ODE	Overlap degree Estimator.
DT	Decision Tree.
RF	Random Forest.
G-mean	Geometric Mean.
UCI	University of California, Irvine.
KEEL	Knowledge Extraction based on Evolutionary Learning.
DCSMOTE	Synthetic minority oversampling using density clustering technique.
DCADYSAN	Adaptive synthetic oversampling and clustering technique .

Chapter 1

Introduction

In many supervised learning algorithms, there is a significant difference between the prior probabilities of different classes, for example, between the probabilities with which an example belongs to the different classes of the classification problem. This situation is known as the class imbalance problem (Chawla *et al.*, 2004),(He and Garcia, 2009),(Sun *et al.*, 2009) and it is common in many real problems from telecommunications, web, finance, ecology, biology, medicine, and oil mining; it can also be considered one of the top problems in data mining today (Yang and Wu, 2006). Furthermore, it is worth pointing out that the minority class is usually the one that has the highest interest (the information we want to learn or to understand the pattern) from a learning point of view, and it also implies a great cost when it is not well classified (Elkan, 2001).

1.1 General context

Learning and analyzing the data in order to predict class labels has been widely studied in machine learning and in artificial intelligence domains. Traditional classification algorithms assume that the data are balanced classes in the space of distributions. However, in many applications, the instances of one class outnumber the other classes. For example, in credit card fraud detection, direct marketing, the detection of oil spills from satellite images, and network intrusion detection, the targeted class has fewer representative data compared to other classes. Due to the increase of these applications in recent years, learning in the presence of imbalanced data has become an important research topic. It has been shown that when classes are well separated, regardless of the imbalanced ratio, instances can be correctly classified using standard learning algorithms (Sun *et al.*, 2009). However, having a class imbalance in complex datasets results in the incorrect classification of the data, particularly the minority class examples. Such data complexity includes issues such as overlapping classes, within-class imbalance, outlines, and noises. Within the class, the imbalance occurs when a class is scattered into smaller subparts representing separate sub-concepts. Sub-concepts with limited representatives are called small disjuncts(He and Garcia, 2009). Classification algorithms are often not able to learn from small disjunct examples. This problem is more noticeable in the case of under-sampling techniques. This is due to the fact that the probability of randomly choosing an instance from small disjuncts within the majority class is very low. These regions may, therefore, remain untrained (López *et al.*, 2013). In this thesis, we will focus on dealing with overlapping in imbalanced data, mainly within the class and between classes that are imbalanced.

1.2 Specific context

Recently, researchers discovered that, for some data, there may be a gray area, or for certain data points, it may be difficult to decide to which data class they belong(Sáez *et al.*, 2019). There are issues with algorithms that classify data in overlapping spaces, and these have some limitations. If we have a visual representation of the dataset, the issue becomes apparent in data that belong to two or more classes, but the distance is too small for example, between two classes to determine whether a data point belongs to class A or class B. The issue is more pronounced in imbalanced datasets where there is a majority class and a minority class. The examples of one class often outnumber the other. The minority class usually represents the most critical concept to learn because acquiring these examples is costly. Although often associated with binary classification, the imbalanced class problem can also occur in multiclass problems and may involve minority classes that are more difficult to classify (FernáNdez *et al.*, 2013), (Lin *et al.*, 2013).

1.3 Related Works

In (Janicki and Soudkhah, 2015), Janicki and Soudkhah formally introduce the concept of "feature domain overlapping" and use this concept for object classification. They assume that less overlapping means more discrimination ability. In addition, in (Koç, 1995), Hakime Koc presented a new methodology of learning from examples, which is a new form of exemplar-based generalization technique based on the representation of overlapping feature intervals. His technique is called classification with overlapping feature intervals (COFI). It's learned from the projections of intervals in

each dimension for each feature. Those intervals correspond to the learned concepts. The overlapping classes or the ambiguous data has been studied for a long time, particularly in the area of character recognition and document analysis (Lewis and Gale, 1994). Tang et al. (Yaohua and Jinghuai, 2007) proposed a KNN-based approach to extract the ambiguous region in the data. Visa et al. (Visa and Ralescu, 2003) performed a fuzzy set representation of the concept and incorporated overlap information in the fuzzy classifiers. In Xiong et al. (Xiong *et al.*, 2010), researchers used the one-class classification Support Vector data description algorithm (SVDD) to capture the overlapping regions in real time data-sets. Handling and dealing with overlapping regions is as importing as identifying such regions. Xiong et al. (Xiong *et al.*, 2010) proposed that the overlapping regions can be handled with three different schemes: discarding, merging and separating. The First scheme Discarding schemes ignore the overlapping region and learn on what is left of the data in the non-overlapping region. For example, SMOTE and Tomek links (Batista *et al.*, 2003). The second scheme is merging, which considers the overlapping area as a new class and use a 2-tier classification model. The upper tier classification focuses on the entire data set with an additional class representing the overlapping region. If the test sample is classified as belonging to the overlapping region, the lower tier classifier, which works only in the overlapping region, is used. For instance, Trappenberg et al. (Trappenberg and Back, 2000) proposed a scheme that refers to the overlapping region class as IDK (I don't Know) and predicts the class label of test data only when it is first classified as IDK. In the result, the authors claim that by losing predication accuracy on IDK, a drastic increase in confidence can be gained on the classification of the remaining data. In a separate scheme, the data from overlapping and nonoverlapping regions are treated

separately to build the learning models. Tang et al. (Yaohua and Jinghuai, 2007) proposed a multimodel classifier named dual rough support vector machine (DR-SVM), which combines SVM and KNN under a rough set technique. KNN is used to extract the overlapping regions. Then, after that, two SVMs are trained for the overlapping and nonoverlapping regions. Prati et al. (Prati *et al.*, 2004), (Batista *et al.*, 2005) illustrated the cause of imbalanced class distribution posing a problem in the presence of a high degree of class overlap. They have shown that an overlap aggravates the problem of imbalance and is sufficient to degrade the performance of the classifier on its own. Garcia et al. (García *et al.*, 2006) analyzed the effect of the combined problems on an instance-based classification scenario. On the other hand, a category of data cleansing methods tackles the problem by cleaning up unwanted overlapping between classes, that is, by removing pairs of minimally distanced nearest neighbors of opposite classes, popularly known as Tomek links (Tomek, 1976). SMOTE+ENN and SMOTE+Tomek Batista *et al.* (2004) utilize the capability of Tomek links to clean the data. However, the cleansing techniques are not desirable for data sets that have an inherent class overlap or absolutely rare minority class samples that can cause loss of highly informative data.

1.4 Contributions

The contributions was divided on the chapters and can be summarize as :

- Detecting and estimating the level of the overlapping: We proposed an algorithm to detect the area that have overlap in binary or multi-classes.
- Proposed an algorithm to build a tree for the pattern of the data that have

overlapping.

- Propose an improved method to oversampling techniques: The proposed algorithm using the clustering method, Using SMOTE or ADYSAN.
- Feature selection and dimensionality reduction: We propose an algorithm for large-scale datasets that have high dimensionality or large number of features.

And other contributions :

- Compare and Evaluate our method with other existing methods.
- Implementing the detecting algorithm using python.
- Implementing the building a tree for the patterns.
- Implementing the oversampling method.

1.5 Related publications

- (Almutairi and Janicki, 2020) in this paper we did some experiments using different data sets, and we used different classifiers to see how overlapping and imbalance in the data can affect the accuracy of the prediction.

1.6 Thesis outlines and structure

- Chapter 1: In the first chapter we discusses the problem we will tackle in this thesis and why these problem will affect the accurate of the prediction of the classifier, with list the contribution toward this topic.

- Chapter 2: In this chapter we go to some depth and background done in this problem in this field, first we will discuss the imbalance issue what are the options to improve the imbalanced dataset with some algorithms in oversampling and undersampling techniques. And some that can be done in the dataset in the preprocessing time.Then we discuses the evaluations methods to calculate the accuracy and why it's won't work for the imbalanced datset. Then show how to get a better way to compute the accuracy. Then we move to list some definitions of machine learning terms, the types of machine leaning problems.

- Chapter 3: In this chapter we go into some depth and background in this problem in this field, first, we will discuss the imbalance issue what are the options to improve the imbalanced dataset with some algorithms in oversampling and undersampling techniques. And some that can be done in the dataset during the preprocessing time. Then we discuss the evaluation methods to calculate the accuracy and why it won't work for the imbalanced dataset. Then show how to get a better way to compute the accuracy. Then we move to list some definitions of machine learning terms and the types of machine learning problems. also, In this chapter, we discuss the first problem in this thesis which is the problem of overlapping. We introduce a method to compute the overlapping ratio in each class by computing the overlap between all the features. In this chapter, we give a brief literature review regarding this issue in the dataset. Some solution is solving this issue by just deleting and treating it as outliers. After that, we discuss the problems in the dataset's outliers and noise, missing values, and high dimensionality of the features. then we introduce the algorithms to :

    1. Detect and compute the ratio of the overlapping in each features.

2. The algorithm for calculating the degree of overlapping using ODE.
3. Build a pattern tree for each class that we can use for oversampling methods.
4. The algorithm for building a tree for oversampling.

In addition, we give some real example of dataset to calculate the overlap degree and build the pattern tree.

In this chapter, we discuss the second issue in the dataset, which is balancing an imbalanced dataset. We list some algorithms that can be used to balance a dataset. Most of these methods are designed to address only one issue, which is the imbalance in the dataset. However, our focus is on addressing both the imbalance and overlapping issues. Some algorithms use oversampling, while others use undersampling. We provide examples of using the overlapping ratio to avoid ambiguity in the overlapping area and using the pattern tree to distribute new synthetic samples equally based on the data we have.

- Chapter 4: In this chapter, we did some experiments and evaluations of the proposed method and compared it with existing methods. Then we give definitions of the classifier and the classifier we will use. The setup of the configurations of the experiments with the real-world datasets, then we list the datasets that are used and the resource of datasets.

- Chapter 5: In this chapter, we propose a method to tackle the curse of dimensionality to select features based on the degree of overlapping. Then we give some details and background on the issues of the large datasets that have high dimensions. Also, brief definitions of some existing methods. So, our method

will reduce the dimensionality by selecting features based on information gain or the higher component that explains the data, then we apply an overlapping estimator to choose from the features, and after that, we apply the oversampling method to balance the dataset using our oversampling method. Then we did some experiments on how the features were selected to reduce the dimensionality and maintain accuracy.

- Chapter 6: In this chapter, we explore the various paths for our thesis, as our method can support both supervised and unsupervised learning. This makes it suitable for generalizing across different types of learning.

- Chapter 7: We provide concluding remarks and final comments.

Chapter 2

Background

In this chapter, we will describe the problem that we have; some of this work was published in (Almutairi and Janicki, 2020). We showed the problem when we had datasets that have both imbalanced and overlapped issues. The degree of overlapping in the classes features one dataset has just one feature overlapping, and another dataset has all the features overlapping.

2.1 Problem Definitions

In real life, data is almost always imbalanced and skewed; many researchers have studied the problem of unbalanced and overlap data and introduced a way of balancing the data and not distinguishing between the two issues or focusing on each problem alone. In (López *et al.*, 2013) , (Denil and Trappenberg, 2010), (Prati *et al.*, 2004) and (García *et al.*, 2006), we see the problem is discussed with synthetic data sets, and have some degree of overlapping or separation. However, the problem with those datasets they used generated the synthetic data with only one feature. In our data

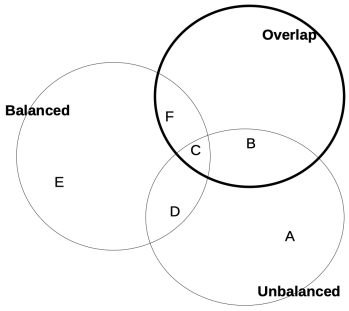
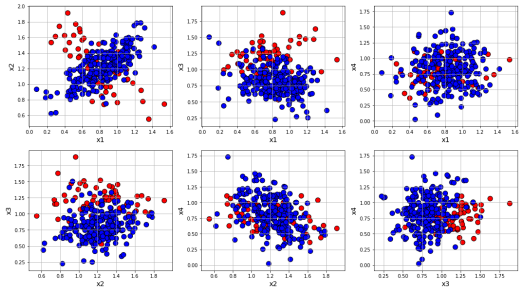


Figure 2.1: Case classification: balanced and imbalanced vs. overlapping

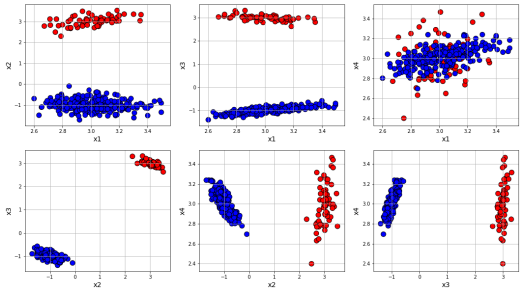
set (Almutairi and Janicki, 2020), we tried to create a real alike data set. We will see it in the next section. In general, the data can come in regarding balance, unbalance, and overlapping, as in Figure 2.1. We see all the possibilities of the data that we can have in real life:

- `A`: The classes are imbalanced with no overlapping.
- `B`: The classes are imbalanced with overlapping.
- `C`: C is obtained by balancing type A dataset and it may overlap.
- `D`: D is obtained by balancing type A dataset but no overlapping.
- `E`: The classes are balanced and no overlapping (very rare in real life).
- `F`: The classes are balanced and have some overlapping.

We used the imbalanced-learn package to generate A Synthetic random data sets, that have two classes or labels. The first data sets are balanced data sets that have a different level of overlapping or separation in the features, as Figure 2.2 shows that. The second type is imbalanced data sets that have 1:4 for class 0: class1, respectively.As shown in Figure 2.2 Each generated data sets have different overlapping over the four features.



(a) Dataset with all the 4 features overlapping.



(b) Dataset with one feature overlap.

Figure 2.2: Example of overlapping and Imbalanced dataset

The first data sets consist of data that generated the features randomly by controlling the separation level. There is a dataset that has all the four features separated. Moreover, others have some variations in the overlapping. Besides, one dataset has all the features overlap.

The second type of data we have generated is for the unbalanced dataset.there are some data with a variety of overlapping or separation. Moreover, we have data that does not overlap and one that overlaps in all features.

2.2 Sampling-based Methods

In data analysis, there are techniques used to adjust the class distribution of a data set, for example, a ratio between the different classes or categories represented. Oversampling and under-sampling (figure 2.3 and 2.4) are opposite and roughly equivalent techniques to be done on majority or minority classes. They both involve using a bias-generated data to get some information on the data that we have, so we then do some analysis on the data set to draw the line between the two classes. The reason for using oversampling is to correct the bias in the original dataset. One scenario where it is useful is when training a classifier using labeled training data from a biased source since labeled training data is valuable but often comes from unrepresentative sources.

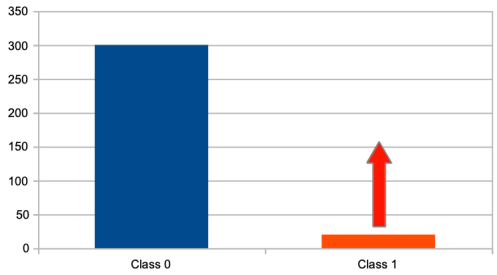


Figure 2.3: Imbalanced dataset Oversampling minority class

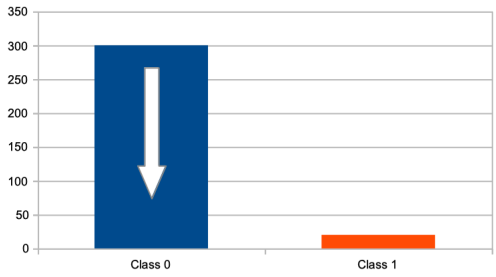


Figure 2.4: Imbalanced dataset undersampling majority class

2.2.1 Tomek Link (T-Link)

Tomek links remove the unwanted overlap between classes where majority class links are removed until all minimally distanced nearest neighbor pairs are of the same class. A Tomek link is defined as follows: given an instance pair: (x_i, x_j) Where $x_i \in S_{min}, x_j \in S_{max}$ and $d(x_i, x_j)$ is the distance between $x_i x_j$, then the pair (x_i, x_j) is called a Tomek link if there's no instances x_k such that $d(x_i, x_k) < d(x_i, x_j) or d(x_j, x_k) < d(x_i, x_j)$. In this way, if two instances form a Tomek link, then either one of these instances is noise, or both are near a border. Thus, one can use Tomek links to clean up overlaps between classes. By removing overlapping examples, one can establish a well-defined cluster in the training set and lead to improve the classification performance (Tomek, 1976).

2.2.2 Synthetic Minority Oversampling Technique (SMOTE):

There are a number of methods available to over-sample a dataset used in a typical classification problem using a classification algorithm to classify a set of images given a labeled training set of images or a set of text documents. The most common technique

is known as SMOTE: Synthetic Minority Over-sampling Technique (Chawla *et al.*, 2002). To illustrate how this technique works, consider some training data which has s samples and$f$ features in the feature space of the data set. Note that these features, for simplicity, are continuous (Wang and Japkowicz, 2004). As an example, consider a data set of plants for clarification. The feature space for the minority class for which we want to over-sample (e.g. leaves length, stem length, and height) are all continuous features. To over-sample, take a sample from these data sets and consider its K nearest neighbors in the feature dimension. To create a synthetic data point, take the vector between one of those k neighbors and the current data point. Multiply this vector by a random number x which lies between 0 and 1. Add this to the current data point to create the new synthetic data point (Han *et al.*, 2005).

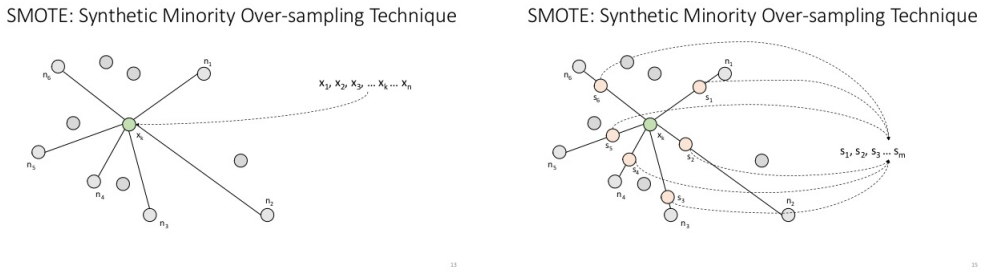


Figure 2.5: SMOTE Technique

This method (Chawla *et al.*, 2002) creates synthetic examples to oversample the minority class, balancing it as demonstrated in Algorithm 1. The amount of over-sampling is a parameter that can be adjusted within the system. Consider a sample taken from the minority class, the point (6,4) and its k-nearest neighbor (4,3), as seen in Table 2.2 $f1$ and $f2$ represent the features. Artificial samples are created until balance is achieved for example, if the majority class C_1 has 600 samples and the minority class C_0 has 200 samples. See Table 2.1 to view the percentage of balancing

in the minority class.

Table 2.1: Incresing The Minority class using SOMTE

	$Class\ C_1$	$Class\ C_0$	$Total\ Samples$
$Original\ dataset\ Smote = 0\%$	600 80%	150 20%	750
$smote = 100\%$ to C_0	600 67%	300 33%	900
$smote = 200\%$ to C_0	600 50%	600 50%	1200

Table 2.2: Example to create synthetic sample using SMOTE

	Smaple(6,4), knn(4,3)
f1_1	6
f2-1	4
f2_1-f1_1	-2
f1_2	4
f2_2	3
f2_2-f1_2	-1
new sample=	$(6,4) + rand(0,1) * (-2,-1)$

2.3 Adaptive Synthetic Sampling ADASYN :

Adaptive Synthetic Sampling (ADASYN) is a machine learning technique specifically designed to tackle the challenge of imbalanced datasets. It operates by generating synthetic samples for underrepresented classes, thereby enhancing the classification performance by balancing the dataset. This reduces the bias that typically favors the majority class, which is a common issue in various real-world applications such as medical diagnostics, fraud detection, and network intrusion detection.ADASYN improves upon earlier methods like SMOTE (Synthetic Minority Over-sampling Technique) by

Algorithm 1 SMOTE(T, N, k)

Input : Number of minority class Samples T; Amount of SMOTE N %; Number of nearest neighbors k

Output: $(N/100) * T$) Synthetic minority class samples

(//If N is less that 100%, randomize the minority class samples as only a random percent of them will Be SMOTEd.)

if $(N < 100)$ **then**

  Randomize the T minority class samples

  $T = (N/100) * T$

  $N = 100$

end

$N = (int)(N/100)$

$k =$ Number of nearest neighbors

$numattrs =$ Number of attributes

$Sample[][]$: array of original minority class samples

$newindex$: keeps a count of number of synthetic samples generated. initialized to 0

$Synthetic[][]$: array for synthetic samples

for $i \leftarrow 1$ *to* T **do**

  Compute k nearest neighbors for i, and save the indices in nnarray

  Populate$(N, i, nnarray)$

end

while $N \neq 0$ **do**

  Choose a random number 1 and k, call it nn. this step chooses one of the k nearest neighbors of i.

  for $attr \leftarrow 1$ *to* $numattrs$ **do**

    Compute: $dif = Sample[nnarray[nn][attr] - Sample[i][attr]$

    Compute: $gap = randomnumberbetween0and1$

    Synthetic$[newindex][attr] = Sample[i][attr] + gap * dif$

  end

  $newindex++$

  $N = N - 1$

end

focusing on generating synthetic samples based on the density distribution of minority classes. This adaptive approach helps to generate more samples for those minority classes that are harder to classify, potentially leading to more robust classification models.In (He *et al.*, 2008) introduced a weighted distribution to add samples to be more specific for decision boundaries,

ADASYN Algorithm Definition:

$$\text{ADASYN}(D_{\text{minor}}, \beta) = \bigcup_{i=1}^{n} \{\text{generate\_samples}(x_i, G_i)\}$$

where D_{minor} is the set of minority class samples, n is the number of minority class samples, x_i is a minority class sample, G_i is the number of synthetic samples to be generated for x_i, and β is the desired balance level after resampling.

Calculation of G_i (Number of Synthetic Samples for Each Minority Class Sample):

$$G_i = \left\lfloor \beta \cdot \frac{\text{dist}(x_i, \text{Majority Class})}{\sum_{j=1}^{n} \text{dist}(x_j, \text{Majority Class})} \right\rfloor$$

Here, $\text{dist}(x_i, \text{Majority Class})$ measures the degree of minority class sample x_i being outnumbered by the majority class, influencing the number of synthetic samples generated.

Overall Goal of ADASYN:

$$\text{Objective}: \quad \min_{\theta} \left| \frac{N_{\text{minor}}}{N_{\text{major}}} - 1 \right|$$

N_{minor} and N_{major} represent the number of samples in the minority and majority classes, respectively, after applying ADASYN. The goal is to minimize the ratio to as close to 1 as possible, indicating a balanced dataset.

These formulas encapsulate the core mechanics of the ADASYN algorithm, highlighting its adaptive nature in generating synthetic samples based on the local distribution of the minority class within a dataset.

Algorithm 2 ADASYN algorithm

Calculate the degree of class imbalance: $d = m_s/m_l$, where $d \in (0, 1]$.
if $d < d_{th}$ **then**
- (a) Calculate the number of synthetic data need to be generated to minority class $G = (m_l - m_s) \times \beta$, where $\beta \in [0, 1]$ is parameter balance level.
- (b) **for** $(x_i \in minorityclass)$ **do**
  - find K nearest neighbors based on Euclidean distance in n dimensional space, where the ratio r_i defined as :
  - $r_i = \Delta_i/K, i = 1, ..., m_s$.
  - (c) Normalize r_i according to $\hat{r_i} = r_i / \sum_{i=1}^{m_s} r_i$, $\hat{r_i}$ is a density distribution $(\sum_i \hat{r_i} = 1)$
  - (d) Calculate the number of synthetic data examples that need to be generated for each minority examples x_i :
  - $g_i = \hat{r_i} XG$,
  - (e) **for** x_i *generate examples 1 to* g_i **do**
    - Randomly choose one minority data example, $x_z i$ from K nearest neighbors fro data x_i.
    - Generate the synthetic data example :
    - $s_i = x_i + (x_z i - x_i) \times \lambda$, where $(x_z i - x_i)$ is the difference vector in n dimensional space , and λ is a random number $\in [0, 1]$.

Table 2.3: The differences and similarities between SMOTE and ADASYN

	SMOTE	ADAYSYN
Number of oversampling	N	β
Synthetic sample generation	Random Point on line	Random Point(s) on line
Similarity measure distance	Euclidian	Euclidian

2.4 The preprocessing phase :

The preprocessing phase in machine learning is a crucial stage where data is cleaned, transformed, and organized to ensure the effectiveness and accuracy of subsequent modeling. During preprocessing, various techniques are employed to deal with common issues such as missing values, noise, and inconsistent data, as well as to convert raw data into a format more amenable to analysis(Obaid *et al.*, 2019).

Recent trends in preprocessing have emphasized the importance of addressing data imbalances and feature scaling among other challenges. For example, normalization methods have been advanced, particularly in specialized fields like microbiome research, where data compositional nature demands tailored preprocessing methods to accurately interpret biological data(Jasner *et al.*, 2021).

2.4.1 Data reduction :

This is when we transform a numerical or alphabetical digital information derived empirically or experimentally into a corrected, ordered, and simplified form. The basic concept is the reduction of multitudinous amounts of data down to the meaningful parts (Liu and Motoda, 2012).

2.4.2 Data cleansing or data cleaning :

Data cleaning is the process of detecting and correcting or removing, if duplication is found, corrupt or inaccurate records from a record set, table, or database, and refers to identifying what is incomplete, incorrect, or inaccurate, such as when we find a more recent or updated version of a record. We remove the old version of the record or the irrelevant parts of the data and then replace, modify, or delete the dirty or poor quality data (Müller and Freytag, 2005). Data cleansing may be performed interactively with data-wrangling tools or with batch processing through scripting (Rahm and Do, 2000).

2.4.3 Resampling :

Resampling is a method in a class of methods that estimate the test error by holding out a subset of the training set from the fitting process, and then applying the statistical learning method to those held-out observations. For supervised learning problems, separate data sets are required for building, training, and, testing for the predictive models. The training error is too promising. The more we fit the data, the lower the training error, but the test error can get higher if we over-fit, and it often will. For these reasons, models are fitted on part of the data and then evaluated

against a holdout set on the test set (Molinaro *et al.*, 2005).

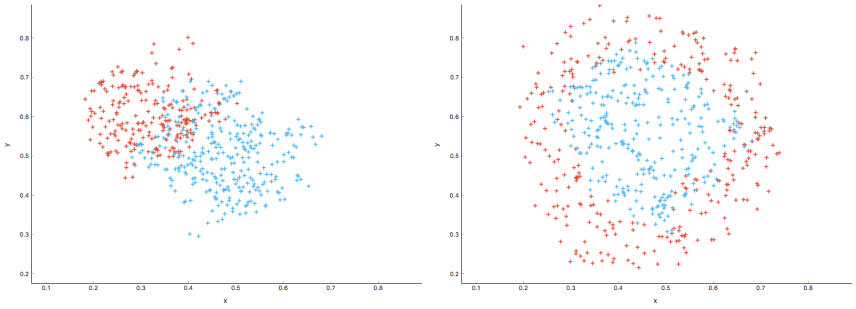


Figure 2.6: Example of overlapping dataset

2.5 Evaluation Imbalanced datasets :

2.5.1 Accuracy Matrix :

The evaluation measures are an important factor in both evaluating the classification performance and supervising the classifier modeling to a better algorithm. In a two-classes problem, the confusion matrix (shown in Table 2.4) shows the results of correctly classified and incorrectly classified examples of each class. Traditionally, the accuracy rate (Eq. (2.5.1)) has been the most commonly used observed measure. Nevertheless, in the structure of imbalanced data sets, accuracy is no longer a proper measure, since it does not distinguish between the number of correctly classified examples of different classes. Therefore, it may lead to incorrect conclusions. For example, a classifier achieving an accuracy of 95% in a dataset with an imbalance ratio value of 9 is not accurate if it classifies all examples as negatives.

$$Accuracy = (\frac{TP + TN}{TP + TN + FP + FN}), \tag{2.5.1}$$

Table 2.4: Confusion Matrix

		Predicted	
		Negative	Positive
Actual	Negative	TN	FP
	Positive	FP	TN

$$Precision = (\frac{TP}{TP + FP}), \tag{2.5.2}$$

$$Recall = (\frac{TP}{TP + FN}), \tag{2.5.3}$$

$$F - Measure = (2.\frac{Precision.Recall}{Precision + Recall}), \tag{2.5.4}$$

Where

TN : number of true negative cases

FP : number of false positive cases

FN : number of false negative cases

TP : number of true positive cases

- Precision: It gives us the fraction of relevant instances from retrieved instances; Finding the percentage of TP instances that were correctly classified. Precision is given by equation 2.5.2. in (Manning and Schutze, 1999) gives the relation between the precision and recall in detail.

- Recall: This metric gives the fraction of relevant instances retrieved from the data; recall is finding the percentage of TF that were correctly classified. Recall

is given by equation 2.5.3. Both precision and recall are finding the TP and TF from relevance (Raghavan *et al.*, 1989).

- F-Measure: It is a harmonic mean of Precision and Recall. The F-score is the score of precision and recall in one number (Powers, 2020). It is given by the equation 2.5.4.

2.5.2 G-mean

The geometric mean (Barandela *et al.*, 2003) is a strategy for learning in class imbalance problems, and it's can be defined as :

$$GM = \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}} \tag{2.5.5}$$

Which can be written in different forms :

$$GM = \sqrt{PositiveAccuracy \times NegativeAccuracy} \tag{2.5.6}$$

$$GM = \sqrt{Sensitivity \times Specificity} \tag{2.5.7}$$

This equation maximizes the accuracy on each of the classes as if the data almost balance; The GM computes the rate of TP_{rate} (Sensitivity) and the rate of TN_{rate} (Specificity).

2.5.3 Area Under Curve

By having imbalanced dataset domains, the evaluation of the classifier's performance must be administered using specific metrics to take into account the class distribution. Especially, a well-known approach to produce an evaluation criterion in an imbalanced dataset design is to use the Receiver Operating Characteristic figure 2.8 (Bradley, 1997). This graphic design enables us to visualize the tradeoff between the benefits (TP-rate) and costs (FP-rate); hence, it evidences that any classifier cannot increase the number of true positives without also increasing the false positives. The Area Under the ROC Curve (AUC) (Huang and Ling, 2005) figure 2.7 corresponds to the probability of correctly identifying which one of the two incentives is noise and which one is signal plus noise. AUC provides a single measure of a classifier's performance for evaluating which model is better on average. The points (0,1) represent the perfect classification. The AUC measure is computed just by obtaining the area of the graph.

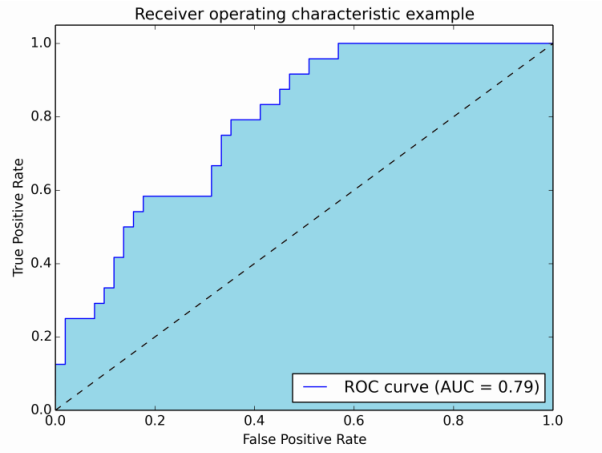


Figure 2.7: AUC Area

The performance of machine learning algorithms is typically evaluated by a confusion matrix as illustrated in table 2.4. The columns are the Predicted class, and the

rows are the Actual class. In the confusion matrix, True Negatives TN is the number of negative examples correctly classified, False Positives FP is the number of negative examples incorrectly classified as positive, False Negatives FN is the number of positive examples incorrectly classified as negative, and True Positives TP is the number of positive examples correctly classified. Predictive accuracy is the performance measure commonly associated with machine-learning algorithms and is defined as Accuracy = (TP +TN)/(TP +FP +TN +FN). In the context of balanced datasets and equal error costs, it is logical to use error rate as a performance metric. The error rate is 1 Accuracy. In the presence of imbalanced datasets with unequal error costs, it is more suitable to use the ROC curve or other similar techniques ((Ling and Li, 1998), (Drummond and Holte, 2000), (Provost and Fawcett, 2001), (Bradley, 1997), (Turney, 2002)).

2.5.4 ROC Curves :

The receiver operating characteristic (ROC) curve is a standard technique for summarizing classifier performance over a range of tradeoffs between true positive and false positive error rates (Swets, 1988). The area under the curve (AUC) is an accepted performance metric for an ROC curve (Bradley, 1997). ROC curves can be presupposed as representing the family of best-choice boundaries for relative costs of TP and FP. On an ROC curve, the X-axis represents %FP = FP/(TN +FP) and the Y-axis represents %TP = TP/(TP +FN). The standard point on the ROC curve would be (0,100), that is, all positive examples are classified correctly, and no negative examples are incorrectly classified as positive. One way an ROC curve can be swept out is by manipulating the balance of training samples for each class in the training

set. Figure 2.7 shows an illustration (Chawla *et al.*, 2002). The line y = x represents the scenario of randomly picking the class. A single operating point of a classifier can be chosen from the trade-off between the %TP and %FP, that is, one can choose the classifier giving the best %TP for an acceptable %FP (Neyman-Pearson method) (Egan, 1975). Area Under the ROC Curve (AUC) Figure 2.7 is a useful metric for classifier performance as it is independent of the decision criterion chosen and prior probabilities. The AUC ratio can learn a dominance relationship between classifiers. If the ROC curves are intersecting, the total AUC is an average comparison between models (Lee, 2000).

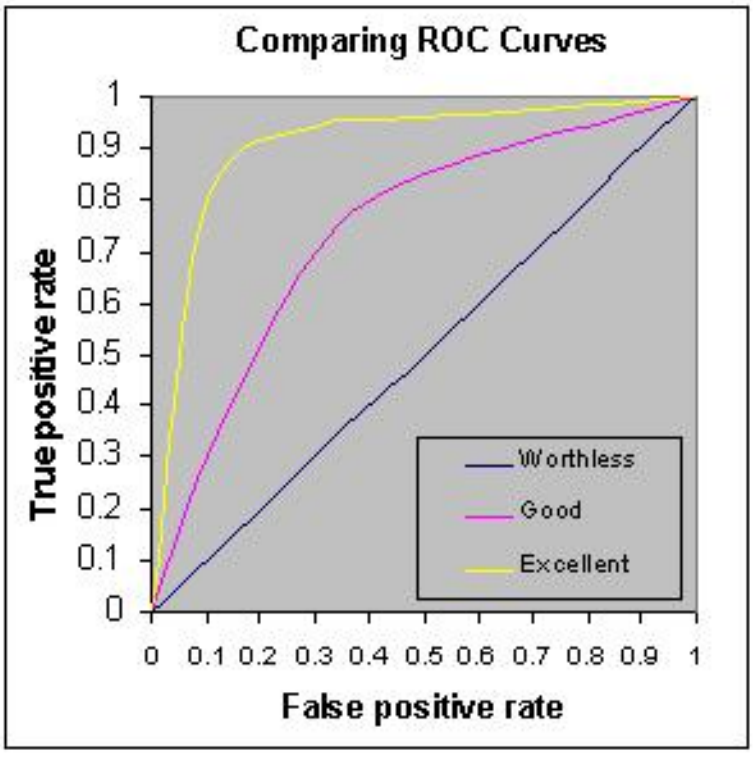


Figure 2.8: ROC curve

2.6 Preprocessing imbalanced datasets:

In (Kubat *et al.*, 1997), they selectively under-sampled the majority class while keeping the original population of the minority class. They have used the geometric mean as a performance measures for the classifier, which can be related to a single point on the ROC curve. The minority examples were divided into four categories, with some noise overlapping the positive class decision region, borderline samples, redundant samples, and safe samples. The borderline examples were recognized using the Tomek links concept (Tomek, 1976). Another related work proposed the SHRINK method that classifies an overlapping region of minority positive and majority negative classes as a positive best fit positive region (Kubat *et al.*, 1998), (Elhassan *et al.*, 2016). (Japkowicz *et al.*, 2000) discussed the effect of imbalance in a dataset. She evaluated three strategies: under-sampling, resampling, and a recognition-based induction scheme. We focus on her sampling approaches. She experimented on artificial 1D data in order to easily measure and create a concept complexity. Two resampling methods were considered. The first one is random resampling, which consisted of resampling the smaller class at random until we got the idea of the boundary region as many samples as the majority class, and the second method is called focused resampling, which is resampling only those minority data that occurred on the boundary between the minority and majority classes. Random under-sampling was studied, which involved under-sampling the majority class samples at random until their numbers matched the number of minority class samples; focused under-sampling involved under-sampling the majority class samples that are lying a distance away. She noted that both the resampling approaches were effective, and she also observed that using the complex sampling techniques did not give any improvement or advantage in the

domain.

2.7 Machine learning:

ML is defined by many definitions, but one can define it as by making an algorithms learn and can improve automatically through experience by using data(Jordan and Mitchell, 2015). We mean by the experience refers to the past information(data) that was used and feed to the learner. The size and the quality of the data will contribute on the accuracy of the predictions made by the learner. The problems can be in many different types like: Text or document application, Speech processing applications, Computer vision application, and many other applications. In 2.9 we can see all the types of Machine learning with respect of how the data was given, with example for each type.

Task	Label space	Output space	Learning problem
Classification	$\mathfrak{L} = \ell$	$\mathcal{Y} = \ell$	learn an approximation $\hat{c} : \mathcal{X} \to \ell$ to the true labelling function c
Scoring and ranking	$\mathfrak{L} = \ell$	$\mathcal{Y} = \mathbb{R}^{\lvert\ell\rvert}$	learn a model that outputs a score vector over classes
Probability estimation	$\mathfrak{L} = \ell$	$\mathcal{Y} = [0,1]^{\lvert\ell\rvert}$	learn a model that out- puts a probability vector over classes
Regression	$\mathfrak{L} = \mathbb{R}$	$\mathcal{Y} = \mathbb{R}$	learn an approximation $\hat{f} : \mathcal{X} \to \mathbb{R}$ to the true labelling function f

Table 2.5: Predictive machine learning scenarios

In table 2.5 a classifier is a mapping $c : \mathcal{X} \to \ell$, where $C = C_1, C_2, ..., C_k$ is a finite and usually small set of class labels. We will sometimes also use C_i to indicate the set of examples of that class. We use the hat to indicate that $\hat{c}(x)$ is an estimate of the true but unknown function $c(x)$. Examples for a classifier take the form $(x, c(x))$, where $x \in \mathcal{X}$ is an instance, and $c(x)$ is the true class of the instance. Learning a classifier involves constructing the function $\hat{c}$ such that it matches c as closely as

possible.

2.8 Term definitions in Machine learning

This section provides definitions for commonly used terms in machine learning:

- **Dataset**: a table with the data from which the machine learns, the data contains the features and the target label to predict. usually we split the data into training data and test data.
- **instances**: are the rows in the table of the dataset some times it's called data point, observation or example.
- **Features**: Are the instances without the label or y. It's the column in the dataset and we use it to predict or classify.
- **Taget**: the label we want to predict, the target is called y or the single instance y_i.
- **Prediction**:The machine learner try to "guess" what the target label should be from the given feature and it's denoted by $\hat{y}$.
- **Learner** or The machine learning algorithm, it's used to extract some way or recipe to predict the target label.

2.9 Machine Learning Problems:

Machine learning problems can be categorized into supervised, unsupervised, and reinforcement learning categories.

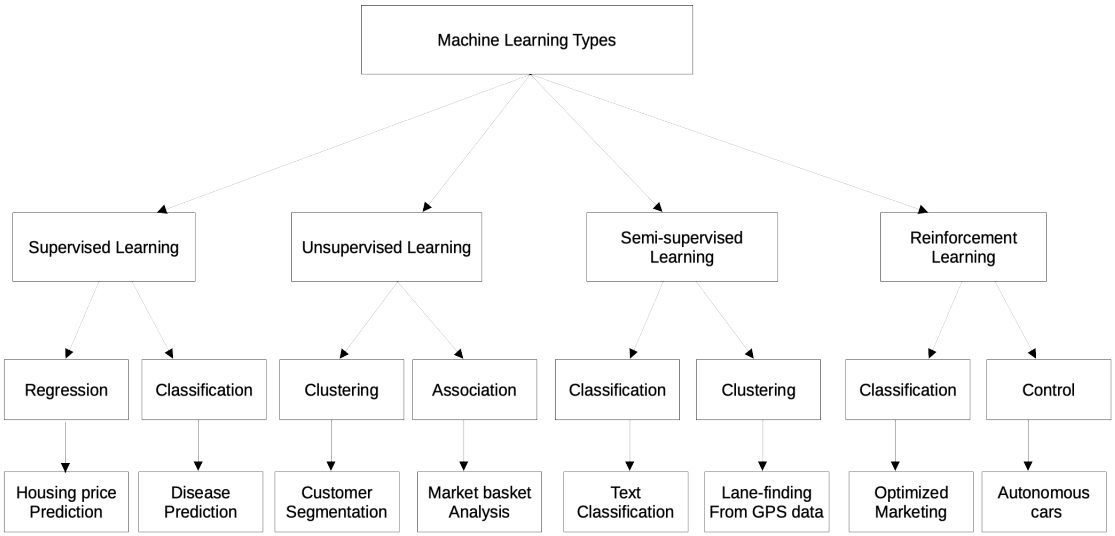


Figure 2.9: Machine learning Types with examples for each type

2.9.1 Supervised Learning-SL :

we say we have a problem of SL When we have given inputs and outputs, the given a set of pairs (x_i, y_j) of samples $x_i \in \mathbb{X}$ and their labels $y_i \in \mathbb{Y}$, so the model will predict the label y_j with new coming sample x_j then the model will classify based on what it's saw. In this type of problems we have two types, the first is classification to predict the label and the second is regression which involves predicting a numerical label(Russell and Norvig, 2002). Some example of SL are K-Nearest Neighbors, Naive Bayes, Support Vector Machines , Decision trees and SVM.

In this types of learning problems their many examples: Emails spam filtering, in this example the given data are e-mail messages x_i with it's labels y_i spam or not spam. So when the user use the filtering model he will use it on new coming emails to filter if the email is spam or not.

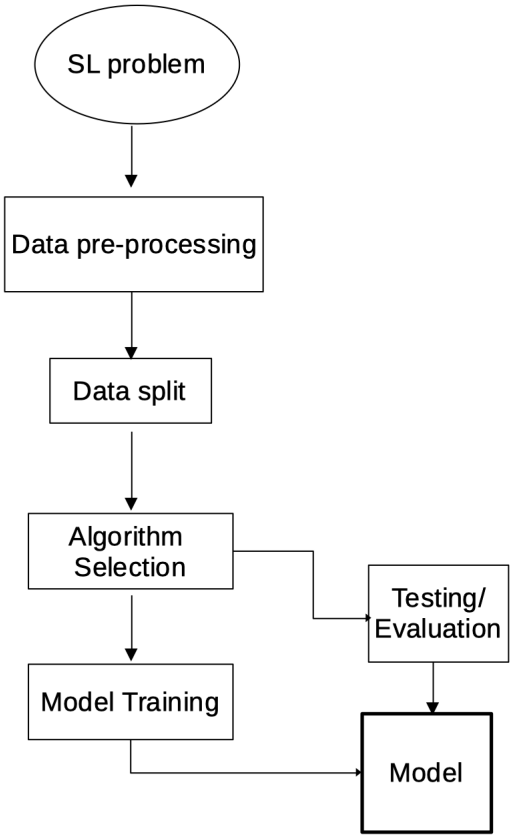


Figure 2.10: Supervised learning

2.9.2 Unsupervised learning -USL :

USL is a type when we have the data x_i but we are missing the labels y_i . There are many algorithms to USL problems, but the main two solutions for the missing labels are, Clustering method which use clustering to find groups in the data. the second method is density estimation that uses summarizing the distribution of the data. in both method by finding the groups or the summarization of the data that means we found the labels or y_i (Hinton *et al.*, 1999). Some example of USL are K-Means and

DBSCAN.

2.10 Semi-Supervised Learning- SSL

SSL is the type of machine learning that has a small amount of labeled data, with a large amount of data that are missing the label.Some times this can of learing refer to as transductive learning or inducative learning (Zhu, 2005). The transductive learning is to interfere the correct labels for a given dataset $x_{l+1}, ..., x_{l+u}$, and the goal of inductive learning is to infer to the correct mapping between $X \rightarrow Y$. The example on this type of model is text document classifier. because it's hard to find a large amount of labeled text documents. so in this case the algorithm learn from a small amount of labeled text documents while still classify a large amount of unlabeled text documents in the training data.

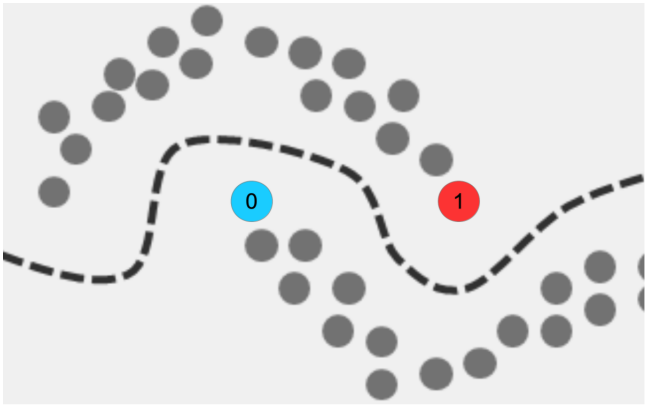


Figure 2.11: SSL with small amount of labels (0,1)

2.11 Reinforcement Learning- RL

RL is the area when we train the machine learning model to make a sequence of decisions. then based on these decisions we make an action, to maximize the notion of cumulative rewards (Sutton and Barto, 2018). RL does't need the data to be labeled (Kaelbling *et al.*, 1996). RL is for the problems that include a long term and a short term reward trade-off. some examples of Reinforcement Learning that implemented successfully on games and can compete with best players in the game like : checkers, Go(AlphaGO) (Silver *et al.*, 2017).

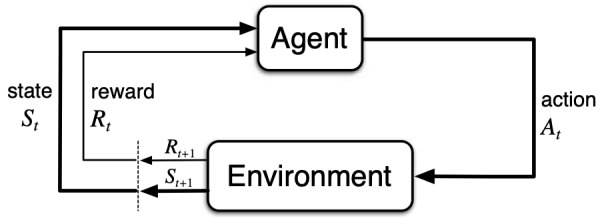


Figure 2.12: Maximizing rewards in Reinforcement Learning

Chapter 3

Overlapping detection and building pattern of the dataset

In this section, We will introduce the new algorithm that improves the accuracy of the classifiers after we use it on the data, which happens to have two problems. The first problem is having imbalanced multi-classes and has overlapping between different classes.

This method tries to have a sense and the pattern of the data more, so it is understanding the complexity of the data like overlapping and find the bounders Like in border-smote algorithm(Han *et al.*, 2005) . We first find that area of complications, the higher the number, the complex is the data. Then after finding those areas, we build a tree based on the findings of the overlapping area, one for overlapping inside the wrong class and the other one for the overlapping near the boundaries. Then apply the proposed oversampling algorithm, which uses the density of the non-overlapping area, the overlapping area the bounders areas, then we apply the best of two worlds that oversample the data (SMOTE+ ADASYN) based on the level of overlapping.

3.1 Related work

The most promising method that can handle different types of learning problems is the clustering method. There are many flavors or clustering methods, but the general idea behind the technique is simple and powerful. The learning algorithms make different assumptions about the data and the target function to improve the output model. For example, some clustering algorithms assume the clusters are spherical (Von Luxburg, 2007). Some algorithms assume that the data have been sampled from a population that consists of the subpopulation data, for example, the mixture of Gaussian method (Dempster *et al.*, 1977).

In (Wu and Chang, 2003) they introduce a derived from SVM the method is for a class-boundary alignment algorithm, they transform the kernel function K when it is possible for the training data to be represented in a vector space. Then the algorithm modifies the kernel matrix K when the data do not have a vector space representation. In (Veropoulos *et al.*, 1999) suggest using different penalty factors C^+ and C^- for positive and negative classes while training the data to reflect the importance to help the SVM algorithm to decide to whom the data point belong.

In (Liu, 2008) propose a method for that classes closely overlapped but not necessarily to be separated before contextual information is exploited. And for those classes have overlapping either discriminating between them or merging them into a meta-class. The paper presents a solution for this problem by a partial discriminative training Scheme. The training pattern of an overlapping class is used as a positive sample of its labeled class and neither positive nor negative samples for its allied classes. But in this method changes the original data. It is also time-consuming.

In (Sit *et al.*, 2009) they proposed a method to allow multiple classes to be predicated during classification and explored several modifications to reduce the number of categories. They used a soft decision to solve the overlapping class problem and analyzed and judged based on these options. Since the data that falls onto the overlapping class area is manually evaluated, it is labor-intensive and high time cost. In (Szmidt and Kukier, 2006) They used a fuzzy set approach. The classifier recognizes the classes, and then an intuitionistic fuzzy set representation deals with imbalanced and overlapping data. But in (Dabare *et al.*, 2019) they used a fuzzy deep neural network based on C-means clustering to improve the classification of classes with overlapping. In (Fu *et al.*, 2015; Devi *et al.*, 2019) they used SVM as a preprocess method overlapping area, but in (Xiong *et al.*, 2013) they used Naive based method to find class with overlapping regin and uses this region with non-overlapping region and uses naive base to classify it as an independent data.

SVM based classifier has many adapted to solve many problems with the data, In the problem of imbalanced and overlapped (Devi *et al.*, 2019) has introduced anomaly detection to detect overlapping and use Tomek-link undersampling algorithm by removing the borderline

3.2 Naive approaches

A naive way to this problem is to remove the outliers or the borderline points in the intersections from two or more classes in data. That will make each cluster is separable. or the points that overlap between two or more features of other classes. Also, another naive way is to start building clusters and ignore the data point near or between both classes and randomly choose to which it belongs.

3.3 Dataset problems

Data pre-processing is the most crucial part of the machine learning algorithm. If the dataset is not clean, the ML algorithms won't work correctly. The pre-processing part is worth the time to do cleaning the errors, outliers, and noise, so the training part on the data achieves higher accuracy.

- if some instances are outliers, it may help the algorithm discard them or fix the errors manually.
- If the instances are missing some features, we need to decide if we want to keep it and fix it by filling the missing values with appropriate values depending on the case; for example, if the feature is the age, we fill it with the median age.

3.3.1 Outlies And noise

When the data is skewed or has noise, this can affect the performance of the classifier. Noisy instances can occur when some instances are corrupted or have issues, such as the satellite images or some problem with the hardware that collects the data. In (Xiong *et al.*, 2006) suggested some solutions for outliers and noisy data to improve the accuracy of the classifier by removing the outliers and noisy instances using the clustering technique.

3.3.2 missing values

Missing values is another challenge in dataset that can affect the perormance of the classifier to build a model from the training dataset.In (Acuna and Rodriguez, 2004) they suggested some remedies for missing values (Chan and Dunn, 1972).

- Deletion case: In this method, we delete or discard all the instances with missing values.
- Mean Imputation: In this case, we fill the missing values with the mean.
- Median Imputation: Use the median instead of using the mean to avoid the outliers.
- KNN Imputation: This method uses the neighboring instances that are similar to fill the missing value with an appropriate value.

3.4 High dimensionality

The meaning of high dimensionality in data is when numbers of features F in the data is larger than the number of observations or instances I, $F >> I$. The high dimensionality problem, also known as the curse of dimensionality (Bellman, 1961) , describes poor performance in generalizing local nonparametric estimators as the dimensionality increases.

Techniques for Dimensionality reduction :

- Features selection methods: Is a method to reduce the irrelevant features that do not help that much to improve the performance of the classifier (Murphy, 2012).
- Matrix factorization : Matrix factorization is a method to reduce a matrix into a constituent part that makes it easier to calculate more complex matrix operations by doing matrix decomposition. The most common way for ranking

the components is principal components analysis, or PCA (Wold *et al.*, 1987), (Abdi and Williams, 2010).

- Manifold Learning: This method aims to reduce the dimensionality to make it easy to visualize. In this algorithm, we take a random projection of the data to create a low dimensional projection of high dimensional data (Cayton, 2005).

3.4.1 Imbalanced and Overlapping

We have a problem that is the focus of this thesis. The issues are in 2.1. The following sections will look at this problem in detail and do experiments with datasets that have these issues to improve accuracy using the proposed algorithms.

3.5 Formal Mathematical Definitions

Definitions and Concepts

Sampling:

Is the process of selecting a subset of elements from a larger set, known as the population. Formally, let P be a population with N elements, and let S be a sample of size n such that $S \subseteq P$ and $|S| = n$, where $|\cdot|$ denotes the cardinality of a set.

Undersampling:

Is a technique used to reduce the size of the majority class in an imbalanced dataset. Formally, let C_m and C_M be the minority and majority classes in a dataset D, respectively, with $|C_m| < |C_M|$. Undersampling involves selecting a subset $C'_M \subseteq C_M$ such that $|C'_M| = |C_m|$, resulting in a new balanced dataset $D' = C_m \cup C'_M$.

Oversampling:

Is a technique used to increase the size of the minority class in an imbalanced dataset. Formally, let C_m and C_M be the minority and majority classes in a dataset D, respectively, with $|C_m| < |C_M|$. Oversampling involves creating a set C'_m by duplicating elements of C_m or generating synthetic elements, such that $|C'_m| = |C_M|$, resulting in a new balanced dataset $D' = C'_m \cup C_M$.

3.5.1 Imbalanced Dataset

Let $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ be a dataset for a binary classification problem, where x_i represents the feature vector and $y_i \in \{0, 1\}$ the class label for each instance i. The dataset is said to be imbalanced if the number of instances in one class significantly exceeds the number in the other, i.e.,

$$|\{i \mid y_i = 0\}| \gg |\{i \mid y_i = 1\}| \quad \text{or} \quad |\{i \mid y_i = 1\}| \gg |\{i \mid y_i = 0\}|$$

where $|\cdot|$ denotes the cardinality of a set.

3.5.2 Overlapping in Imbalanced Datasets

Overlapping occurs when the feature space of the minority class (C_{min}) intersects with the majority class (C_{maj}). Formally, let F_{min} and F_{maj} be the feature spaces for the minority and majority classes, respectively. Overlapping is said to occur if:

$$F_{\text{min}} \cap F_{\text{maj}} \neq \emptyset$$

So when we have instances of different classes in a dataset are not well-separated and share common regions in the feature space. Formally, let C_i and C_j be two different classes in a dataset D. The overlapping region O between C_i and C_j is defined as $O = \{x \in D | x \in C_i \cap C_j\}$, where $\cap$ denotes the intersection of two sets. and the intersection region has a non-negligible measure in the feature space.

3.5.3 Impact on Classification Algorithms

A classification algorithm $f : X \rightarrow \{0, 1\}$ may experience difficulty in accurately predicting the minority class in the presence of overlapping, as the decision boundary $f^{-1}(0.5)$ may lean towards the majority class. This is due to a higher representation of C_{maj} in the training data, potentially causing misclassification of instances in $F_{\text{min}} \cap F_{\text{maj}}$.

3.5.4 Importance in Real-World Applications

In many applications such as fraud detection or disease diagnosis, where D is imbalanced and C_{min} (e.g., fraud cases, diseased patients) is of greater interest, accurately classifying instances of C_{min} is crucial.

3.5.5 Addressing Overlapping

To mitigate the impact of overlapping, sampling techniques can be applied:

- **Over-Sampling C_{min}:** Add copies of instances or generate synthetic instances in C_{min} to increase its representation.

These methods aim to modify D such that the new dataset D' reduces the overlap in feature space and balances the class distribution, enhancing the classifier's ability to distinguish between C_{min} and C_{maj}.

3.6 Binary classes vs. multi-classes Data

- Binary Classification: This is the process of classifying a given data with two labels or two classes. For instance, most of the medical diagnoses are binary classes that have a positive or negative diagnose. But in our case, in the imbalanced or rare dataset, the accuracy would be inappropriate, So we need to do an extra step for balancing the data (Koyejo *et al.*, 2014).

- Multi-Classification: Is the process of classifying given data into different classes > 2. In this type of data, we can use, One-vs-rest (Bishop, 2006) we need to train the classifier on a single class per class, with that samples of that class as positive samples and all the other samples as negatives. So the classifier's output is a score for its decision. The second way to classify multiclass is by using one-vs-one, which is $k(K-1)/2$, so we divide the problem into a multi binary classifier for K-way multiclass problem

3.7 Calculating Weights Using Overlapping of Feature Values

In (Janicki and Soudkhah, 2015), the authors assigned some weights to features, by measuring their discriminatory power, using overlappings, and proposed some notation and theory that allow to do that. The authors of (Janicki and Soudkhah, 2015) also employ the ideas of pairwise comparison paradigm and use some fundamental properties of arithmetic and geometric means.

some notation for there method :

$\mathbb{R}$ is the set of real numbers, $\mathbb{N}$ is the set of natural number and by $|A|$ the cardinality of a set A. Then in $\langle i_1, \ldots, i_m \rangle$ to denote the set $\{i_1, \ldots, i_m\} \subseteq \mathbb{N}$, such that $i_s \leq i_r \iff s \leq r$ for all $i_s, i_r \in \{i_1, \ldots, i_m\}$. and this can be written as :

$$\langle i_1, \ldots, i_m \rangle \subseteq \langle j_1, \ldots, j_l \rangle$$

if $\{i_1, \ldots, i_m\} \subseteq \{j_1, \ldots, j_l\}$, $i_s \leq i_r \iff s \leq r$ for all $i_s, i_r \in \{i_1, \ldots, i_m\}$, and $j_s \leq j_r \iff s \leq r$ for all $j_s, i_r \in \{j_1, \ldots, j_m\}$. For every $p \in \mathbb{N}$, the Cartesian product $\mathbb{R}^p$ can be written as $\mathbb{R}^p = \mathbb{R}_1 \times \ldots \times \mathbb{R}_p$, where $\mathbb{R}_i = \mathbb{R}$ for $i = 1, \ldots, p$, and $\mathbb{R}$ denotes the set of real numbers. For every $\langle i_1, \ldots, i_m \rangle \subseteq \langle 1, \ldots, p \rangle$, $\mathbb{R}^{\langle i_1, \ldots, i_m \rangle}$ is defined as

$$\mathbb{R}^{\langle i_1, \ldots, i_m \rangle} = \mathbb{R}_{i_1} \times \ldots \times \mathbb{R}_{i_m},$$

where $\mathbb{R}_{i_j} = \mathbb{R}$ for $j = 1, \ldots, m$.

The standard projection on the set of dimensions $\langle i_1, \ldots, i_m \rangle \subseteq \langle 1, \ldots, p \rangle$ is defined as a mapping $\pi_{i_1 i_2 \ldots i_m} : \mathbb{R}^p \to \mathbb{R}^{\langle i_1, \ldots, i_m \rangle}$ such that for any
$\mathbf{z} = (z_1, \ldots, z_p) \in \mathbb{R}^p$:

$$\pi_{i_1 \ldots i_m}((z_1, \ldots, z_p)) = (z_{i_1}, \ldots, z_{i_m}) \in \mathbb{R}^{\langle i_1, \ldots, i_m \rangle}.$$

It is assumed that we have a data sample of n vectors $\mathbf{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, p features $F_1, ..., F_p$ (so $\mathbf{x}_i \in \mathbb{R}^p$ for each $i = 1, ..., n$), and k groups $\mathbf{G}_1, ..., \mathbf{G}_k$ such that $\mathbf{S} = \mathbf{G}_1 \cup ... \cup \mathbf{G}_k$ and $\mathbf{G}_i \cap \mathbf{G}_j = \emptyset$ if $i \neq j$.
It is argued in (Janicki and Soudkhah, 2015) that when we have a feature F_{i_0} such that $\pi_{i_0}(\mathbf{G}_r) \cap \pi_{i_0}(\mathbf{G}_t) = \emptyset$ if $r \neq t$, then the feature F_{i_0} alone is sufficient to construct the partition $\mathbf{G}_1, ..., \mathbf{G}_k$ of $\mathbf{S}$.

The measures of feature importance based on their discriminatory power are defined as follows.
Let $F_{i_1}, \ldots, F_{i_m}$ be a subset of features $F_1, \ldots, F_p$. For every $\mathrm{x} = (x_{i_1}, \ldots, x_{i_m}) \in \mathbb{R}^{\langle i_1, \ldots, i_m \rangle}$ and every $\mathbf{A} \subseteq \mathbb{R}^n$, define a set $\mathsf{C}_{i_1, \ldots, i_m}(\mathrm{x}, \mathbf{A})$ and an index $\mathsf{c}_{i_1 \ldots i_m}(\mathrm{x}, \mathbf{A})$ as

$$\begin{aligned} \mathsf{C}_{i_1, \ldots, i_m}(\mathrm{x}, \mathbf{A}) &= \\ &\{\mathbf{y} \in \mathbf{A} \mid \pi_{i_k}(\mathbf{y}) = x_{i_k}, k = 1, \ldots, m\} \qquad (3.7.1) \\ \mathsf{c}_{i_1 \ldots i_m}(\mathrm{x}, \mathbf{A}) &= |\mathsf{C}_{i_1, \ldots, i_m}(\mathrm{x}, \mathbf{A})| \qquad (3.7.2) \end{aligned}$$

The set $\mathsf{C}_{i_1, \ldots, i_m}(\mathrm{x}, \mathbf{A})$ is the set of all $\mathbf{x}$ in $\mathbf{A}$. For two different groups $\mathbf{G}_r$ and $\mathbf{G}_t$ is

defined:

$$\mathsf{c}^{(rt)r}_{i_1\ldots i_m} = \mathsf{c}_{i_1\ldots i_m}(\pi_{i_1\ldots i_m}(\mathbf{G}_r) \cap \pi_{i_1\ldots i_m}(\mathbf{G}_t), \mathbf{G}_r) \tag{3.7.3}$$

$$\mathsf{c}^{(rt)t}_{i_1\ldots i_m} = \mathsf{c}_{i_1\ldots i_m}(\pi_{i_1\ldots i_m}(\mathbf{G}_r) \cap \pi_{i_1\ldots i_m}(\mathbf{G}_t), \mathbf{G}_t) \tag{3.7.4}$$

The mutual overlapping of $\mathbf{G}_r$ and $\mathbf{G}_t$ over the subdomain $\mathbb{R}^{\langle i_1,\ldots,i_m\rangle}$ is defined as:

$$\mathsf{overlap}^{rt}_{i_1\ldots i_m} = \sqrt{\frac{\mathsf{c}^{(rt)r}_{i_1\ldots i_m}}{|\mathbf{G}_r|} \cdot \frac{\mathsf{c}^{(rt)t}_{i_1\ldots i_m}}{|\mathbf{G}_t|}}. \tag{3.7.5}$$

This equation, which is based on a geometric mean , is based on the pairwise comparison. The value of $\mathsf{overlap}^{rt}_{i_1\ldots i_m}$ is some measure of overlapping between *a pair* $\mathbf{G}_r$ and $\mathbf{G}_t$.

A measure of the mutual overlappings for the features $F_{i_1}, \ldots, F_{i_m}$, denoted by $\mathsf{overlap}_{i_1\ldots i_m}$, is derived from from $\mathsf{overlap}^{rt}_{i_1\ldots i_m}$ for all pairs r and t. Since the value of $\mathsf{overlap}_{i_1\ldots i_m}$ is a *measure of central tendency* of all $\mathsf{overlap}^{rt}_{i_1\ldots i_m}$.

will formally define $\mathsf{overlap}_{i_1\ldots i_m}$ as:

$$\mathsf{overlap}_{i_1\ldots i_m} = \frac{1}{\binom{p}{2}} \sum_{r>t}^{r,t=1,\ldots,p} \mathsf{overlap}^{rt}_{i_1\ldots i_m} \tag{3.7.6}$$

it will always have $0 \leq \mathsf{overlap}_{i_1\ldots i_m} \leq 1$ and the smaller the value of $\mathsf{overlap}_{i_1\ldots i_m}$ is the more important the subset $F_{i_1}, \ldots, F_{i_m}$ of features is.

3.8 One-Vs-One binary Decomposition

In one vs. one approach (Sáez *et al.*, 2019), they used a strategy to alleviate the existence of overlapping without modifying the original dataset. It split the overlapping data onto binary small sets; The data will be split int to $C(C-1)/2$. The training samples involving the pair (c_i, c_j) with $i < j$. then the classifier will distinguish between c_i and c_j by the confidence parameter $r_{ij} \in [0, 1]$ and it's stored in a matrix:

$$R = \begin{bmatrix} - & r_{12} & x_{13} & \dots & r_{1C} \\ x_{21} & - & x_{23} & \dots & x_{2C} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{C1} & x_{C2} & x_{C3} & \dots & - \end{bmatrix} \tag{3.8.1}$$

The final class label can be obtained using :

$$Class = \arg \max_{i=1\dots,C} \sum_{1 \le j \ne j \le C} s_{ij}$$

Where $s_{ij} = 1$ if $r_{ij} > r_{ji}$ otherwise $s_{ij} = 0$.

3.9 Overlapping detection and pattern in datasets

The first part introduces an algorithm to find overlapping areas, specifically one set for the overlap inside the incorrect class. The second set addresses overlap at the boundaries. For example, if we have three classes C_1, C_2, and C_3, and the overlap occurs between C_1 and C_3, we identify this as an overlap inside the incorrect class between C_1 and C_3. We also have border overlap among the three classes, denoted as a boundary overlap involving C_1, C_2, and C_3.

We need to determine the size or the level of overlap that have an intersection, or boundaries with other class (O), from class $C1_{maj}$ and $C2_{min}$. (R) is the average ratio:

$$R = \sqrt{R_{maj} \times R_{min}} = \frac{\sqrt{C1_{maj} \times C2_{min}}}{O}. \tag{3.9.1}$$

$$R_{maj} = \frac{C1}{O}, R_{min} = \frac{C2}{O}. \tag{3.9.2}$$

r; is the difference between the two sets ,

$$r = \frac{Max(C1_{maj}, C2_{min})}{Min(C1_{maj}, C2_{min})}. \tag{3.9.3}$$

These parameters computer how many features are overlapping. A significant overlap needs a large (O) and a smaller (R).

Based from the above and from coefficient overlap or Szymkiewicz-Simpson coefficient we have :

$$CO = \frac{O}{Min(C1_{maj}, C2_{min})}. \tag{3.9.4}$$

If we have the option or want to consider the datapoint in the overlap area as a noise, We can calculate the similarity between the features in the overlap sets or remove it. Regardless of the location of the overlapping.

There are many ways to calculate the similarities between the features (F) from different classes. So we want to find the similarity in the distance from, for example, we have Multiclasses(C1, C2,. . .), and we have features from these classes F1, F2,. . . .

$C1(F1, F2, \dots), C2(F1, F2, \dots), C3(F1, F2, \dots), \dots$

Let say we have :

- D are the data sets $(d_1, d_2, ..., d_n)$.
- let d_1 is a data point from class C_1.
- let d_2 is a data point from class C_2.
- let d_3 is a data point from class C_3.
- i are the attributes for each data points$(i_1, i_2, .., i_n)$

$\forall d_1 \in D$ sim$(d_1, d_1) = 1$,
$\forall d_1, d_2 \in D$, $sim(d_1, d_2) = 0$, if d_1 and d_2, are dissimilar.

There many ways to find if we have similar features, that can have possible overlap or not. this data points have the potential to be in the overlap area in the next section we will see how to find the overlap area. here we are interested to find if there are similarity.

$$\text{Dice Similarity}, Sim_D(d_1, d_2) = \frac{2\sum_{i=1}^{n} d_{1i} \times d_{2i}}{\sum_{i=1}^{n} d_{1i}^2 + \sum_{i=1}^{n} d_{2i}^2} \tag{3.9.5}$$

$$\text{Cosine Similarity}, Sim_C(d_1, d_2) = \frac{\sum_{i=1}^{n} d_{1i} \times d_{2i}}{\sqrt{\sum_{i=1}^{n} d_{1i}^2 \times \sum_{i=1}^{n} d_{2i}^2}} \tag{3.9.6}$$

$$\text{Overlap Similarity}, Sim_O(d_1, d_2) = \frac{\sum_{1=1}^{n} d_{1i} \times d_{2i}}{min(\sum_{i=1}^{n} d_{1i}^2, \sum_{i=1}^{n} d_{2i}^2)} \tag{3.9.7}$$

3.9.1 Overlapping Ratio

The imbalance ratio IR for binary classes is calculated as :

$$IR = \frac{\text{Number of majority instances}}{\text{Number of minority instances}} \tag{3.9.8}$$

In the case of the multiclass dataset, as shown in Figure 3.1, the overlapping features are depicted. We can encounter two scenarios of majority versus minority. In the first case, there is a single minority class and multiple majority classes. In the second case, there are multiple minority classes. We can calculate the total number of all minority or majority classes as follows:

$$IR_{multi-class} = \frac{\sum \text{number of majority instances}}{\sum \text{number of minority instances}} \tag{3.9.9}$$

In table 3.1 we have three datasets taken from the UCI machine learning repository.

Dataset	# Classes	# Attributes	# Instances	# $Class_{Min}$	# $Class_{Maj}$	IR
Vertebral Column	3	6	310	1	2	2.1
Wine	3	13	178	2	1	2.7
breast cancer	2	10	699	1	1	1.9

Table 3.1: Calculating the ratio for multi-class and binary class

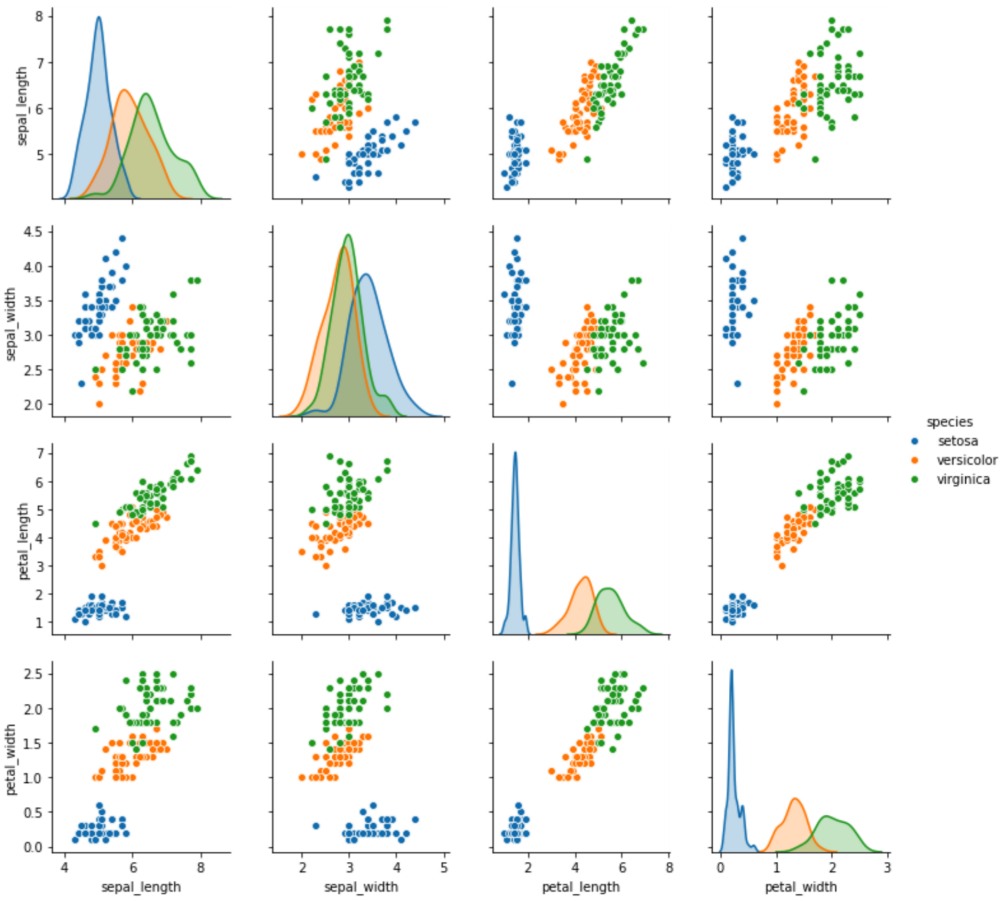


Figure 3.1: Features overlap

3.9.2 Overlapping detection and estimation

We discussed the problem with overlapping in datasets; the harm caused by overlapping between classes is significant. To calculate the degree of overlap in a dataset there and some basic way to find intersections in the features by using the variance, also there is maximum fisher's discriminant ratio (Ho and Basu, 2002) it's obtained

for every feature dimension as :

$$f = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{3.9.10}$$

Where μ_1 and μ_2 are the means. σ_1^2 and σ_2^2 are the the variances (Napierała *et al.*, 2010). So we need to find the differences between the features, in 3.9.10 we have two classes and two features, divided by the total variances of the two features from the two classes. The maximum value is obtained from all the features. The small number of the maximum fisher's discriminant value has a high degree of overlapping. Still, the larger values mean, the less overlapping we have in the dataset, then we combine and find the purity in clusters. To determine the total overlapping, we need to calculate the Overlap Degree Estimator (ODE):

$$ODE = [\frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} + \sum_{i=1}^{k} \frac{m_i}{m} \times P_i] - D_i \tag{3.9.11}$$

- μ are the means of same features from different classes.
- σ are the variance of same features from different classes.
- k is the number of mini clusters.
- m_i is the total number of observations in mini clusters.
- m is the total number of observations.
- P_i is the proportion of majority class in the mini clusters.

- D_i is the duplicated counted instances.

The second part of the ODE equation 3.9.11 is based on the location of the cluster or the mini-cluster to find the purity cluster from the center or mean of its class. The overlapping area makes it impossible to distinguish the classes in this area. Overlapping worsens the problem for an imbalanced dataset because we need to oversample the minority class. We need to be careful in these areas. we will apply the formula of the ODE (Overlap Degree Estimator) to well-known datasets like the Iris dataset, which contains the measurements of the sepal length, sepal width, petal length, and petal width from 150 iris flowers, 50 for the three different flowers; Iris setosa, Iris versicolor, and Iris virginica. (Fisher, 1936). Table 3.2 shows ODE applied to some datasets taken from The KEEL repositories. The small number means the data has a large amount of overlap.

Dataset	**#Attributes**	**#instances**	**IR**	**ODE**
Iris	3	150	-	6.8
Yeast3	8	1448	8.10	4.23
Abalone0918	8	731	16.4	3.9
Segmemt0	19	2308	6.02	5.3
Ecoli3	7	336	8.60	6.26
Glass2	9	214	11.59	2.3

Table 3.2: Overlap Degree Estimator (ODE)

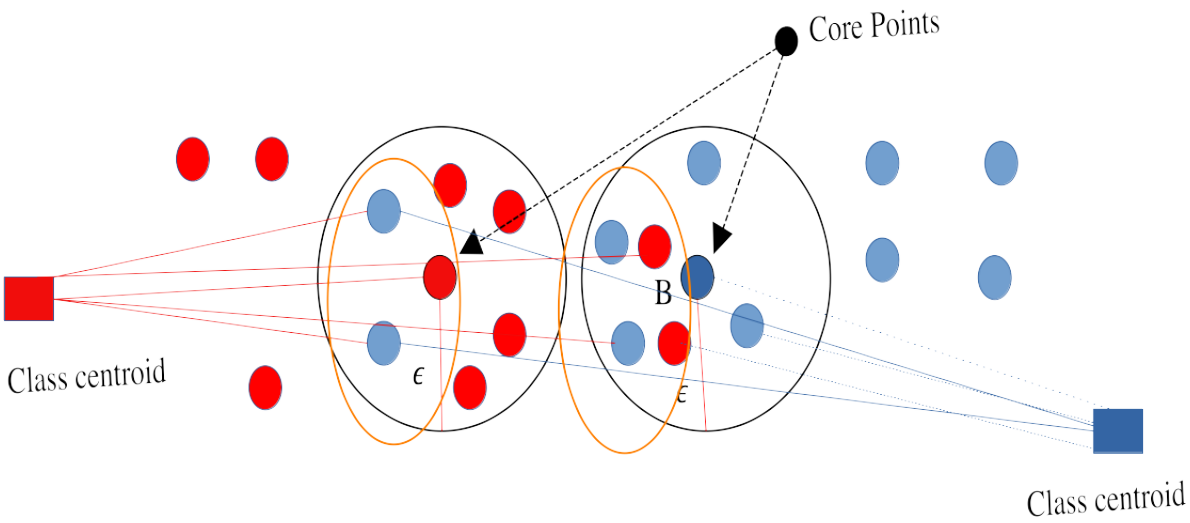


Figure 3.2: Finding features overlap

3.10 The process to find and detect the overlapping between features in classes :

- **For each cluster Find the purity of the cluster or check if the cluster has a mixed points of different classes:**
  $\sum_{i=1}^{k} \frac{m_i}{m} \times P_i$

- **Compute the distance between the desired data point and the centroid for each classes. If we have a point or more that satisfy the conditions:**

  1. for two data point : p_{maj}, p_{min} :
     $distance_{C_{maj}}(p_{maj}) > distance_{C_{maj}}(p_{min})$.
  2. Or if we have for the points p_{min}, p_{maj} :
     $distance_{C_{min}}(p_{min}) > distance_{C_{min}}(p_{maj})$.

- **If the above conditions are satisfied we:**

  Add the mixed cluster to *Overlapping Set*.

- p_{min}: Any point in the mixed cluster from the minority class.
- p_{maj} Any point in the mixed cluster from the majority class.
- $distance_{C_{min}}$: The distance for p_{min} to the centroid.
- $distance_{C_{maj}}$: The distance for p_{maj} to the centroid.
- We used Euclidean distance to measure the distance between the points and the centroid.

3.11 The path and the Distance to class Center

We need to find the distance to the centers of the features. As in figure 3.2 The path from the data point to its feature centroid is computed to determine the distance. Each mini-cluster that has an overlap is computed to find the total number of overlaps. There are many ways to compute the distances here are some of them :

- **Euclidean distance** : which the one we chose and is the most well-known distances used for numerical data, It is the ordinary distance between two points squared (Dokmanic *et al.*, 2015) . It is the one most used in measuring distances in clustering algorithms. basically it computes the root of squared differences

between the coordinates between two objects.

$$dis(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2} \\ = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{3.11.1}$$

- **Manhattan distance**: In Manhattan distance, we compute the absolute differences among the pair of the coordinates, and it's the distance between the real valued vectors as if the distance only could be measured in right angles. there are no diagonal movements to calculate the distance :

$$dis(p,q) = \sum_{i=1}^{n} |p_i - q_i| \tag{3.11.2}$$

- **Chebyshev distance**: Is the maximum distance along one axis, it's defined as the greatest of the difference between two vectors or points p and q along any coordinate dimension.

$$dis(p,q) = \max_i(|p_i - q_i|) \tag{3.11.3}$$

- **Cosine distance**:It measure the similarity between the non-zero vectors or points p and q of an inner product space. And it is defined to equal the cosine of the angle between them. Two vectors with exactly the same orientation have a cosine similarity of 1, and if the two vectors have the opposed to each other have a similarity of negative 1.

$$dis(p,q) = \cos(\theta) = \frac{p \cdot q}{||p||\,||q||} \tag{3.11.4}$$

- **Minkowski distance**:It is q generalization form of both Euclidean distance and the Manhattan distance. Minkowski distance of order j where j is an integer between two points:

$$dis(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i|^j \right)^{1/j} \tag{3.11.5}$$

Where the values of j then we will have :

j = 1 Manhattan distance.

j = 2 Euclidean distance.

j = ∞ Chebyshev distance.

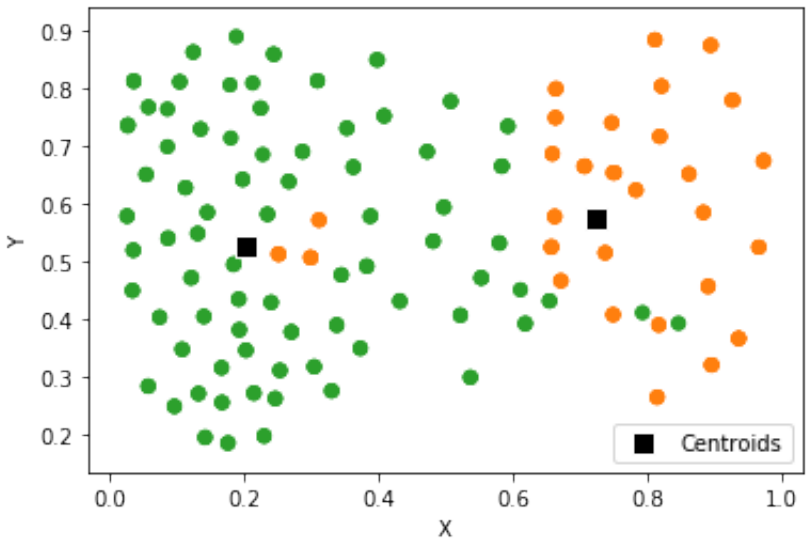


Figure 3.3: Dataset with centroids

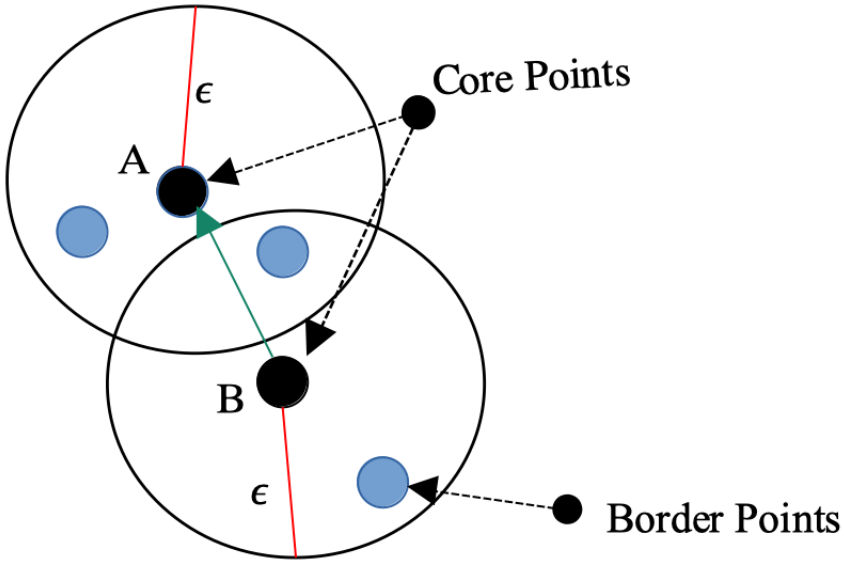


Figure 3.4: Miniclusters to build trees

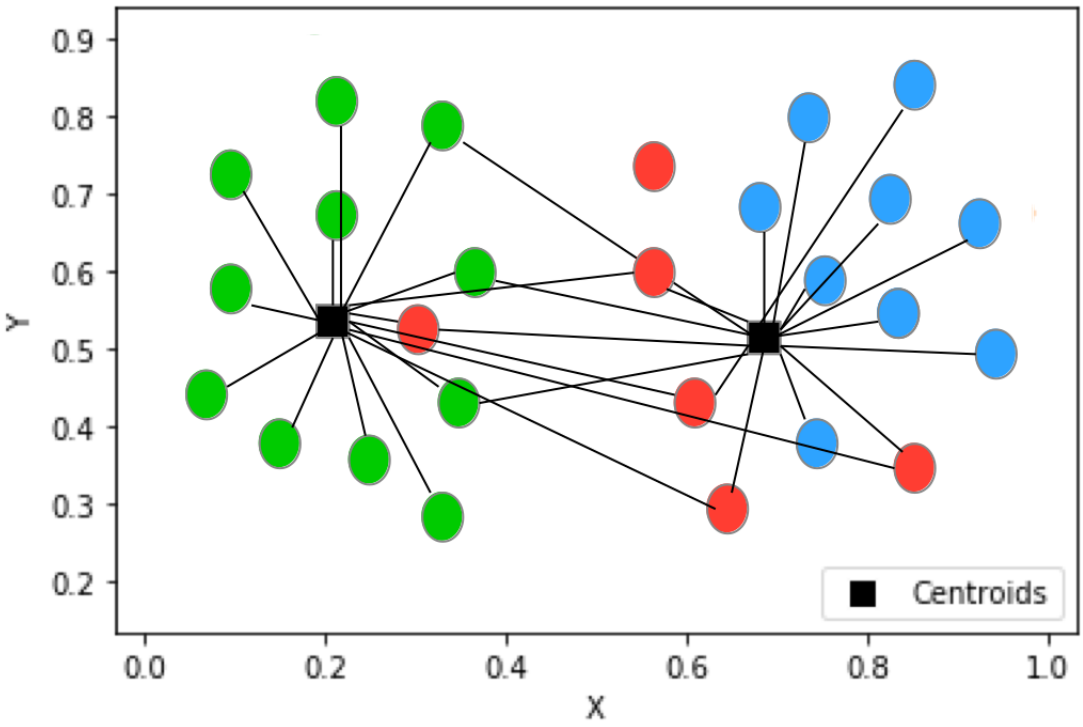


Figure 3.5: Pattern trees with Miniclusters

3.11.1 Building tree pattern for the dataset using Clustering and density method

In this section, we aim to construct a tree based on mini-clusters, as depicted in Figure 3.5. Each tree originates from the centroid and extends to the end of each leaf of the data class. If there are three classes, for instance, we will construct three corresponding trees. From these trees, we can discern the patterns or shapes of each class. Based on the distribution and overlap of the classes, we can generate synthetic samples. The quantities generated are either equally distributed across each mini-cluster or similarly distributed, depending on the specific case.

The path from the centroid to the clusters illustrates whether the data for each class involve instances that enter or overlap with another class. Figure 3.6 shows the possible locations of these overlapping and borderline areas.

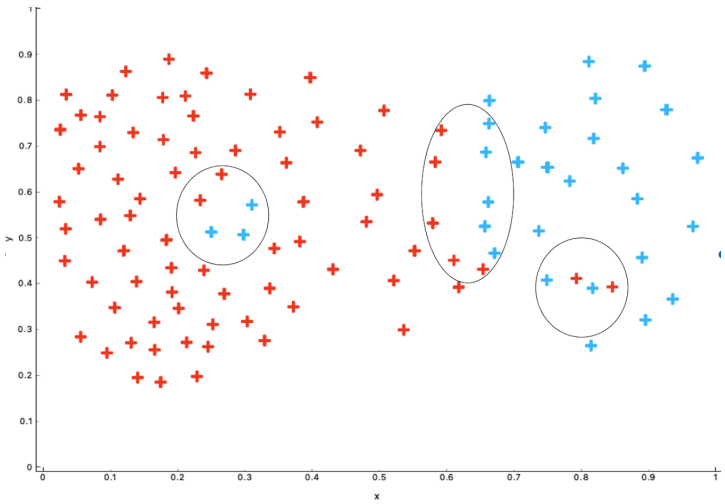


Figure 3.6: Overlapping locations vs Border line area

3.12 Oversampling Using Density Clustering

In this section will show how we will balance the dataset; for the minority class, we will apply an oversample technique. Based on the pattern trees and for each mini-cluster, we will add synthetic data points until we will have $M_{min} <= \beta$ where $\beta = 1$, the classes have precisely the same number of data samples.

Density-based spatial clustering of applications with noise DBSCAN (Ester *et al.*, 1996) is a data clustering algorithm based on the density of the data, this algorithm is grouping data based on similar characteristics. In Algorithm 3 we see the definition of the abstract DBSCAN algorithm, in which we identify the core points for the

chosen and defined numbers of the neighboring points. Then after that, we assign the computed data cores to cluster points. Lastly, we loop over the non-core points to assign border points or assign these points as noise points.

Algorithm 3 DBSCAN Algorithm

Compute neighbors of each point and identify core points.
Join neighboring core points into clusters.
foreach *non-core point* **do**
    Add to a neighboring core point if possible .
    Otherwise, add to noise.

In DBSCAN it identifies the clusters by looking at local density-based in the optimal value of ϵ. The most powerful feature of DBSCAN is its ability to identify outliers. It does not require the number of clusters to be specified at the beginning of the clustering, Unlike K-means.

DBSCAN requires specifying two parameters before starting clustering:

- Epsilon ϵ: which is a parameter to find the radius of the neighborhood around a point X using Euclidean distance method.
- MinPoints: The parameter MinPts is to find the number of neighboring points of point X with ϵ radius.

In Figure 3.8 we calculated the optimal value of ϵ is 0.003 by finding the average k-distances of the neighboring points of the selected k = MinPts using the method used in (Rahmah and Sitanggang, 2016) , We used this method to automatically determine the optimal values of ϵ for each datasets. In (Ester *et al.*, 1996) and (Sander *et al.*, 1998) have some recommendations and suggestions on how to find and choose the values of minPts or Min-samples if we have a 2-dimensional dataset minPts $\geq$ d+1 = 2+1,This will make this process done automatically.

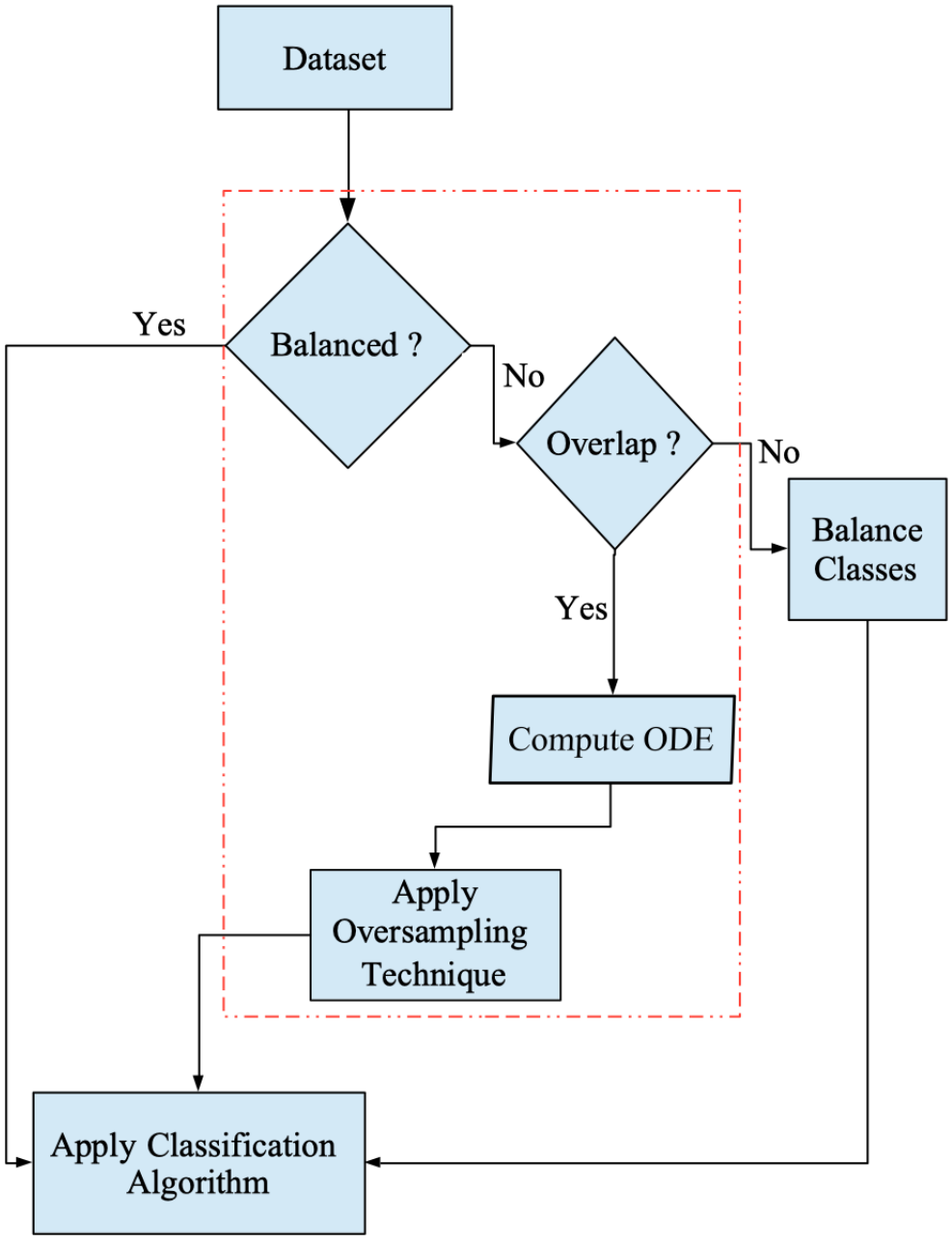


Figure 3.7: Flowchart of the framework

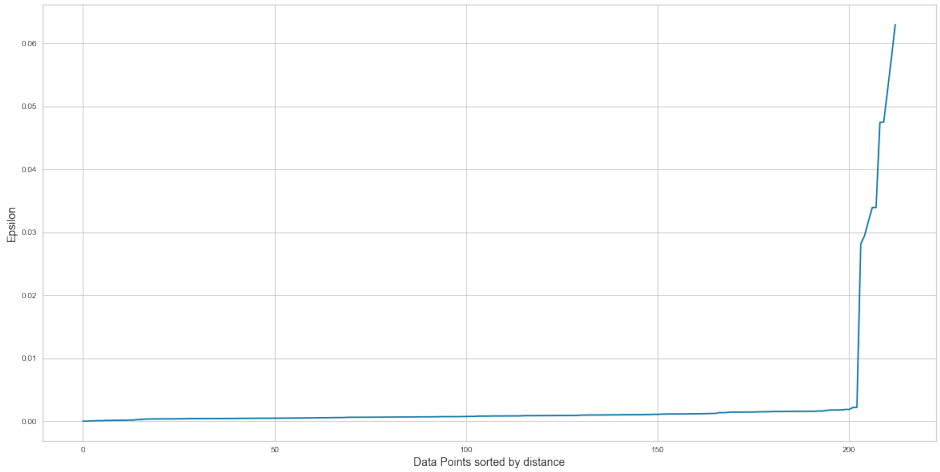


Figure 3.8: Finding Optimal Epsilon

3.13 Theoretical Mathematical Foundations of OverlapCluster Algorithm

3.13.1 Introduction to OverlapCluster

The OverlapCluster algorithm is a new algorithm to cluster that dataset based on its is Density Clustering, it is designed to make mini clusters that can be used to detect, identify and measure overlapping in each features.

3.13.2 Mathematical Background and Definitions

OverlapCluster operates in a metric space (X, dist) with a dataset $D = \{x_1, x_2, \ldots, x_n\}$ and a distance metric $\text{dist} : X \times X \to \mathbb{R}$.

3.13.3 Definition MiniCluster overlapping

A MiniCluster is identified 'has overlappig' if data points from one class are closer to the chosen core point of a different class than to their own class's core point, indicating potential overlap as an example like in figure 3.4. Formally, for clusters C_i and C_j with centroids μ_i and μ_j respectively, a MiniCluster M is defined as:

$$M = \{p \in C_i \mid \text{dist}(p, \mu_j) < \text{dist}(p, \mu_i)\} \cup \{q \in C_j \mid \text{dist}(q, \mu_i) < \text{dist}(q, \mu_j)\} \quad (3.13.1)$$

3.13.4 Adjusted Algorithm Parameters

OverlapCluster uses parameters ϵ and $minPts$, fine-tuned to optimize the detection of overlapping.

3.13.5 Propositions and Proofs

Proposition 3.1 (Existence of MiniClusters): If two points p and q from different classes C_i and C_j, respectively, are within ϵ-distance, a overlap exists.

Proof: Assume $\text{dist}(p, q) \leq \epsilon$ and $p \in C_i$, $q \in C_j$. By the density-reachable property of OverlapCluster, there exists a set of points connecting p and q within ϵ-distance, implying an overlapping region.

3.13.6 Algorithm Adaptation for Overlapping Detection

OverlapCluster includes a post-processing step to identify all MiniClusters by scanning the dataset for points that belong to multiple clusters.

3.14 Introduction to calculating the overall overlap

This section presents a methodology for computing classes overlap, leveraging the approach discussed in Section 3.9.11 for the ODE estimation. The method integrates Between-Class Variance, Within-Class Variance, and Cluster Purity to offer a comprehensive assessment of class distribution and intermixing within clusters. This holistic approach is designed to provide a detailed insight into the extent and nature of class overlap in multidimensional datasets.

3.15 Variance Definitions and Overlap Measurement

3.15.1 Within-Class Variance

The Within-Class Variance for a class C_i within a cluster K_j is defined as:

$$S_{W_{ij}} = \sum_{x \in C_i \cap K_j} (x - \mu_{C_i})^T (x - \mu_{C_i}) \tag{3.15.1}$$

where μ_{C_i} is the mean of the points in class C_i.

3.15.2 Between-Class Variance

The Between-Class Variance for a class C_i is defined as:

$$S_{B_i} = |C_i|(\mu_{C_i} - \mu)^T (\mu_{C_i} - \mu) \tag{3.15.2}$$

where μ is the overall mean of the dataset and $|C_i|$ is the number of points in class C_i.

3.15.3 calculating overlap in all mini-Cluster

Cluster Purity for a cluster K_j is given by:

$$\text{Purity}(K_j) = \frac{1}{|K_j|} \max_{C_i} |\{x \in K_j : x \in C_i\}| \tag{3.15.3}$$

3.15.4 Overlap Measure Using Variance and Purity

The overlap measure incorporating within-class variance, between-class variance, and cluster purity for each cluster K_j can be defined as:

$$\text{Overlap}(K_j) = \left(\frac{\sum_{i=1}^{M} S_{W_{ij}}}{\sum_{i=1}^{M} S_{B_i}} \right) \times (1 - \text{Purity}(K_j)) \tag{3.15.4}$$

where M is the number of classes.

3.15.5 Overall Overlapping Measure

The overall overlap for all clusters by using algorithm 4, considering both variance and purity, is quantified by:

$$\text{Overall Overlap} = \frac{1}{N} \sum_{j=1}^{N} \text{Overlap}(K_j) \tag{3.15.5}$$

where N is the total number of clusters.

3.15.6 Synthetic samples using random locations

In this section we use the proposed algorithm to generate synthetic samples, by understanding the pattern of the shape of the data points o f what we have. This method uses a random sampling to generate the new data that we will add to balance the dataset. In this Algorithm 5 we generate a new samples using Density Clustering in the minority class. The Synthetic minority oversampling using density clustering with noise(DCSMOTE) algorithm is generating the samples based on the density to balance the targeted class(es).

Algorithm 4 Finding clusters

```
Input DataSet DS
Input Distance function dis
Input OverlapSet OS
Input ϵ , minPts
foreach Point p in DS do
    Neighbors N ← RangeQuery(DS,dis,p,ϵ)
    if |N| < minPts then
                                                          ▷ Density check
        label(p) ← Noise
        continue
    if label(p) ∈ OS then
        return false
    c ← next cluster label
    label(p) ← c
    Seed Set S ← N \ {p}                                  ▷ Expand Neighbors
    foreach Point q in S do
        if label(q) = Noise then
            lable(q) ← c
        Neighbors N ← RangeQuery(DS, dis, q, ϵ)
        if |N| ≥ minPts then
                                                          ▷ Check if q is a core point
            S ← S ∪ N                                     ▷ Add new neighbors
```

Algorithm 5 DCSMOTE Algorithm

Input DataSet DS

Input Distance function dis

Input OverlapSet OS , balance ratio β ,IR = M_{maj}/M_{min} ▷ $\beta \in [0, 1]$

Input ϵ , minPts

foreach *Point p* ***in*** *DS* **do**

    Neighbors N ← RangeQuery(DS,dis,p,ϵ)

    if $|N| <$ *minPts* **then**

        label(p) ← newdatapoint (s) ▷ add synthetic data points

        c ← label(p) ▷ add new synthetic data point to cluster

3.15.7 Adaptive Synthetic samples with clustering technique DCADYSAN:

We introduce an algorithm based on the two best aspects of using ADASYN has shown an imposing result. We will use density clustering, reduce the bias, and shift the classification decision boundary in the problematic data point.

1. Calculate the number of the target classes that need to generate examples to balance the minority class: $G = (M_{min} - M_{maj}) \times \beta$; The M_min is the number of the minority class examples and M_maj is the number of the majority class examples. $M_{min} < M_{maj}$ and $M_{min} + M_{maj} = M$; which is the number of the dataset that we have = DS. Where $\beta \in [0, 1]$ is a parameter of which is used to specify the level of data points that we generate a synthetic on the minority class to balance the targeted class when we have $\beta = 1$ means we want a fully balanced minority class as the majority class.

2. For each example $x_i \in M_{min}$ find neighbors within the ϵ minPts based on the Euclidean distance in n dimensional space, and calculate the ratio r_i : $r_i = \Delta_i/K, i = 1, 2, 3....., M_{min}$.

3. Normalize r_i according to $\hat{r_i} = r_i / \sum_{i=1}^{M_{min}} r_i$.

4. Calculate the number of synthetic data point that we want to generate for each data point we choose from M_{min} x_i:
   $g_i = \hat{r_i} \times G$.

5. For each M_{min} data point x_i generate g_i synthetic data point according to the following conditions: 1. Randomly choose one data point from M_{min} from the border points which is not the Core points, for the data from the neighbors of x_i. 2. Generate the synthetic data points: $s_i = x_i + (x_{zi} - x_i) \times \lambda$. where $(x_{zi} - x_i)$ is the difference vector in n dimension space and λ is a random number $\lambda \in [0, 1]$ and has the satisfy the ϵ distance.

In this chapter we introduce the method that we will use in the next chapter, to test and compare it with other methods.

We introduced the following methods :

- **A method to find the overlapping set.**
- **A method to calculate the overlapping degree.**
- **A method to generate or oversampling method to balance the minority class.**
- **An advanced method for oversampling to balance the minority class.**

Chapter 4

Experiments and Evaluation

4.1 Background

This chapter describes the datasets used, their sources, how we split the data, and the rationale behind our chosen methods. What classifier we used, compare our algorithm from chapters 3, with state-of-the-art techniques. The comparison and evaluation are made based on the G-mean and AUC, indicating the accuracy of the methods used to balance the data.

4.2 Classification

Classification is the process of grouping different entities (Goh and Foo, 2007).In AI, it involves building models to predict the class or group to which an entity belongs.. For example, spam detection in email. We have two classes; classspam, not spam. The classifier utilizes the give data(training data) to understand what spam or not spam looks like. So if the classifier has achieved good accuracy, the model generated

by this classifier can be used to detect the unseen emails to decide if this is spam or not spam.

In machine learning, the algorithm that applies to predicting the class or the label is called a classifier. Classification can classify binary class or multiclass datasets, and it's called binary classification, multiclass classification. The binary classification in machine learning is well understood and is more conman, especially in the medical field. There are many classifiers for both binary and multiclass classification algorithms(Har-Peled *et al.*, 2003).

The goal of multiclass classification is to assign an instance to one of the set classes. In application, we have one-vs.-all or one-vs.-the rest, So in one-vs.-all classification uses a binary classifier for each of the possible output classes. We will assign the input instance to the highest confidence class.

4.2.1 Decision Tree

Decision trees are tree-like graphs that model a decision (Quinlan, 1986; Priyam *et al.*, 2013) like the example of the if we have training data of the weather and we want to decide if we need to take an umbrella or jacket figure 4.1. DT are learned by recursively splitting the set of training instances into subsets based of the most possible gain from the selected feature.

The pros of using DT are that it's cheap in computation and easy to see how it's learned from the training data. Also, it can handle missing values and can deal with irrelevant features. But it tends to be prone to overfitting.

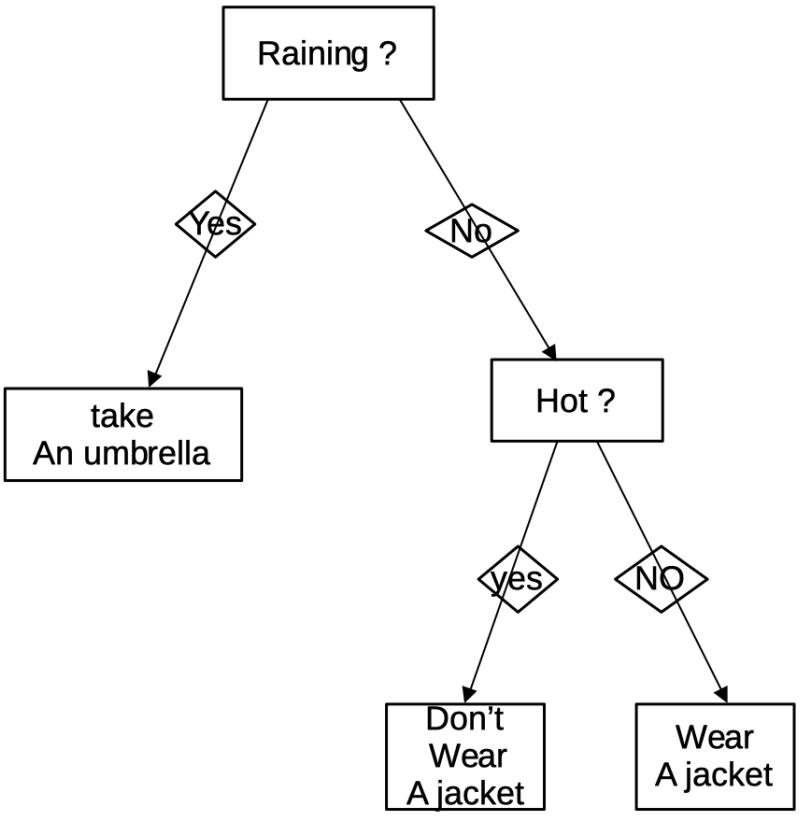


Figure 4.1: Example of a Decision Tree

4.2.2 Support Vector machine

SVM is a supervised learning classifier (Boser *et al.*, 1992), and one of the robust prediction algorithms (Meyer *et al.*, 2003). SVM builds a model that assigns new instances to one class based on the distance from the training model that best fits the model; it maximizes the width between the selected points to the chosen vector and Separating data with the maximum margin. The line that seperate the classes is called a separating hyperplane figure 4.2.

The separating hyperplane has the form of $w^T x + b$. And to find the distance between any points and the separating hyperplane, we measure the perpendicular

line from the point to the separating hyperplane, and it is given by $\frac{|w^T x+b|}{||w||}$.

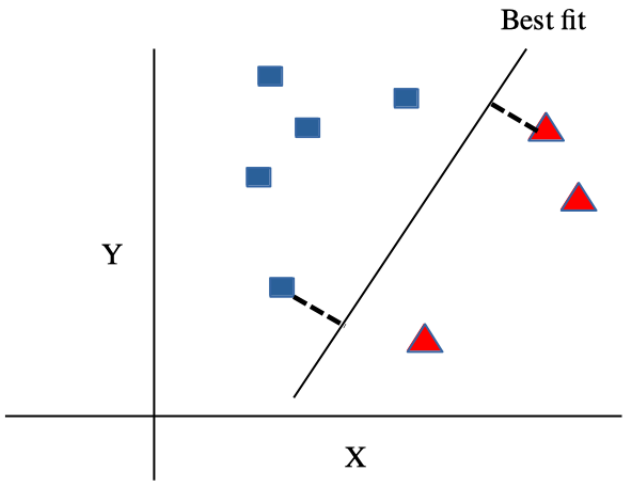


Figure 4.2: Example of a SVM

4.2.3 Random forests

Random forests is a classification method, or it is called random decision forests, which consists of a large number of individual decision trees (Ho, 1995). Each tree has a class prediction, and the class with the most votes will be the tree we choose. RF is a way of taking the average of multiple deep decision trees, trained on different sets of the same training set to reduce the variance (Hastie *et al.*, 2001). RF has a more accurate result than the DT classifier. It can handle missing values and works better to avoid overfitting.

4.2.4 K-nearest neighbors

KNN Is an SL classifier. In the input of KNN, it's consists of the k closest training instances in the dataset (Fix and Hodges, 1989). The output of k-NN is a simple but powerful classification (Altman, 1992). This algorithm can assign weights to local neighbors to decide where the neighboring instance we want to classify belongs like in figure 4.3.

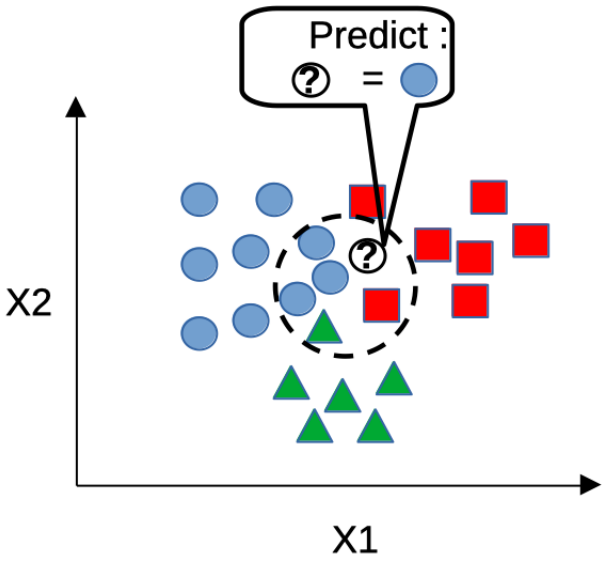


Figure 4.3: Example of a knn

In the experiments , the selection of Decision Trees, KNN (K-Nearest Neighbors), and SVM (Support Vector Machines) as classifiers is strategic for testing the effectiveness of the proposed method. These algorithms are chosen due to their simplicity and distinct learning mechanisms, which provide a diverse testing ground for the proposed technique.

The rationale behind using these simpler algorithms is twofold. Firstly, if the

proposed oversampling method can improve the accuracy of these basic classifiers, it indicates that the method is effective in enhancing the learning process. This improvement in accuracy serves as a strong validation of the proposed technique's ability to address specific challenges in the dataset, Simple algorithms provide a baseline for comparison. If a new method improves the performance of a simple algorithm, it suggests that the method has merit. This is a standard approach in machine learning research to demonstrate the effectiveness of new techniques.(Alpaydin, 2020)

Secondly, demonstrating success with these simpler algorithms suggests that the proposed method is likely to be generalizable and effective across a broader range of classifiers. This is because Decision Trees, KNN, and SVM represent a wide spectrum of learning paradigms, from rule-based to instance-based to margin-based learning. If the proposed method can enhance performance across these diverse algorithms, it is reasonable to expect that it would also work well with more complex models.

4.3 Performance metrics

The process of evaluating the accuracy of the classifier after training the model is called performance metrics. There are different types of evaluation; in the next section, we will see some metrics.

4.3.1 Precision and Recall

As we saw in Chapter 2, precision is defined as the fraction of relevant instances among the retrieved instances. And, the recall is the fraction of the relevant instances retrieved over the total amount of relevant instances(Kent *et al.*, 1955).

4.3.2 Holdout method and k-fold cross validation

Cross-validation is a method of evaluating and comparing learning algorithms by dividing data into two sets or into k-folds (Refaeilzadeh *et al.*, 2009). Holdout method is the most straightforward kind of cross-validation. We split the dataset into two sets, the training set and the testing set. The training set is to train the classifier, and the testing set is to test the model's performance. Usually, we use 80% of the data for training and 20% for evaluating the model.

In cross-validation, when we have a small and limited dataset, it's better to split the data into K numbers of splits, So we train the classifier on most of the data (Browne, 2000). In our test, we will use 10-Folds partitions. In figures 4.4 we have $k = 5$, we train the classifier on all the folds except for one fold to test the model's performance.

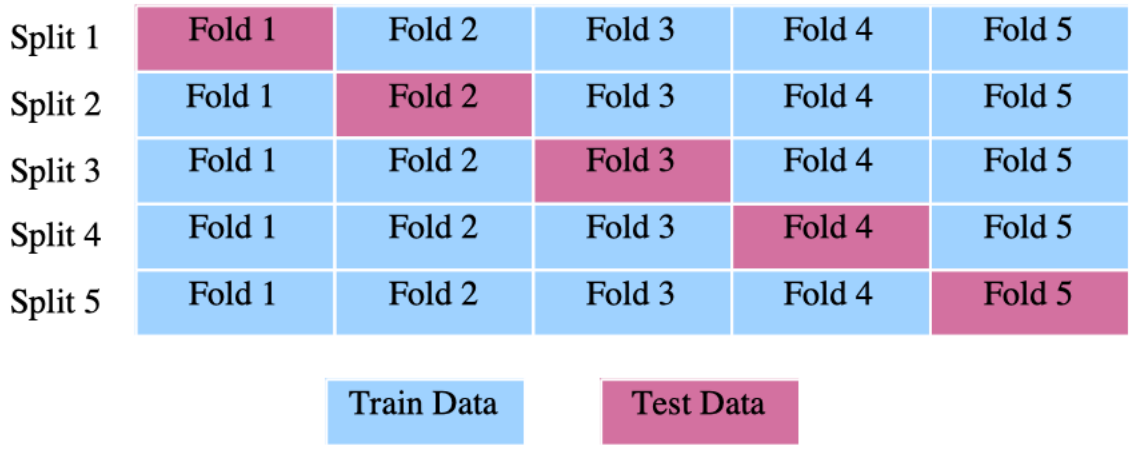


Figure 4.4: Data split into K-fold, k=5

4.4 Datasets

This section provides details on the real-world datasets used to evaluate the performance of our approach. We have used various real-world datasets which are collected and provided by different repositories. We have tested the performance of our approach on some of the datasets from the UCI (Dua and Graff, 2017) Machine Learning Repository of the University of California Irvine. The UCI repository collects various datasets, dataset generators, and domain theories that people can use from data mining and the machine learning community. And we also got some datasets from the Knowledge Extraction based on Evolutionary Learning (KEEL) (Alcalá-Fdez *et al.*, 2011) repositories. In Table 4.6 we see a list of some dataset that we will use, The first column is the name of the data, the second column is the number of instances , The third column is the number of features and the fourth column is the imbalanced ratio : $\frac{M_{maj}}{M_{min}}$. We chose this dataset primarily due to its characteristics of overlapping and imbalance, which present unique challenges and opportunities for our research in machine learning. The dataset's overlapping nature, where different classes share similar features, is essential for developing and testing algorithms designed to effectively distinguish between closely related categories. Additionally, the imbalance in class representation mirrors real-world scenarios where certain outcomes or categories are less frequent but critically important, such as in disease detection or fraud prevention. These features make the dataset an ideal candidate for demonstrating the robustness and effectiveness of new methodologies aimed at handling complex, realistic data structures. Furthermore, its availability in the public domain ensures that our findings are accessible and verifiable, fostering an environment of transparency and collaborative advancement in the scientific community.

4.4.1 Example from the dataset:

Example of the content of one file from table 4.6, We have Ecoli3 dataset which is consist of 336 examples of Ecoli proteins and each example has 8 classes and each class has 7 features as described in :

1. mcg: McGeoch's method for signal sequence recognition.
2. gvh: von Heijne's method for signal sequence recognition.
3. lip: von Heijne's Signal Peptidase II consensus sequence score. (Binary attribute).
4. chg: Presence of charge on N-terminus of predicted lipoproteins. (Binary attribute).
5. aac: score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins.
6. alm1: score of the ALOM membrane spanning region prediction program.
7. alm2: score of ALOM program after excluding putative cleavable signal regions from the sequence.

Classes names and distributions:

Class	Class name	Number of examples	%
cp	cytoplasm	143	42.4%
im	inner membrane without signal sequence	77	23.0%
pp	perisplasm	52	15.5%
imU	inner membrane, uncleavable signal sequence	35	10.4%
om	outer membrane	20	6.0%
omL	outer membrane lipoprotein	5	1.5%
imL	inner membrane lipoprotein	2	0.6%
imS	inner membrane, cleavable signal sequence	2	0.6%

Table 4.1: Ecoli3 dataset

As shown in table 4.1 we see the distribution of the classes, and we see most of the classes are imbalanced.

From equation 3.9.8 and 3.9.9 we calculate the Imbalance Ratio : $\frac{Majority}{Minority}$:

$301/35 = 8.6$

4.5 Experiments setup

The experiments were conducted using the classifiers explained in previous sections. The default for each classifiers has set based from the recommendations from the lecturers in the field, like k in knn is set to k=5 and the k-fold is set to k=10. the following sections of the experiments we will use SOMTE Algorithm and ADASYN algorithm with the improved versions of these algorithms to compare the accuracy with our algorithm. The dataset we will use is taken from the Table 4.6.

Choosing SMOTE (Synthetic Minority Over-sampling Technique) (Fernández *et al.*,

2018) and ADASYN (Adaptive Synthetic Sampling)(Qing *et al.*, 2022) as baseline techniques for addressing class imbalance in machine learning research is a common practice due to several reasons. Firstly, these methods are well-established and have been extensively studied, providing a solid foundation for comparison. By using them as baselines, researchers can benchmark the performance of newer or alternative methods against proven solutions, thereby establishing the relative effectiveness of their proposed approaches.

Furthermore, the popularity and extensive documentation of SMOTE and ADASYN make them easily reproducible by other researchers. This reproducibility is crucial for ensuring that results can be compared across different studies, contributing to the overall reliability and validity of research findings in the field of imbalanced learning. Additionally, their versatility allows them to be applied to a wide range of classification algorithms, making them suitable baselines for studies exploring the impact of class imbalance across various types of classifiers.(Kaur *et al.*, 2019)

Using SMOTE and ADASYN as baselines also helps researchers identify and understand the limitations of these methods in different contexts. This can provide valuable insights into areas where improvements are needed and guide the development of more advanced techniques.

Geometric Mean vs. Arithmetic Mean:

The geometric mean (G-mean) is preferred over the arithmetic mean (Graham and Tokieda, 2020) in imbalanced datasets because it provides a balanced measure of performance across classes. Formally, let TPR (True Positive Rate) and TNR (True Negative Rate) be the performance metrics for the minority and majority classes,

respectively. The G-mean is defined as:

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

In contrast, the arithmetic mean would be:

$$\text{Arithmetic mean} = \frac{TPR + TNR}{2}$$

The G-mean ensures that both TPR and TNR are high for a high overall score, thus preventing a model from ignoring the minority class, which is a common issue in imbalanced datasets. The arithmetic mean, on the other hand, could result in a high overall score even if one of the rates is low, which is not desirable in the context of imbalanced datasets.

The G-mean is a more suitable metric for imbalanced datasets because it provides a balanced evaluation of classification performance across different classes, particularly when there is a significant imbalance between the classes. Here's a more formal explanation:

Let's consider a binary classification problem with a positive (minority) class and a negative (majority) class. Let TPR (True Positive Rate) and TNR (True Negative Rate) be the performance metrics for the positive and negative classes, respectively.

The G-mean is defined as the geometric mean of TPR and TNR:

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

The Arithmetic mean, on the other hand, is defined as the average of TPR and TNR:

$$\text{Arithmetic mean} = \frac{TPR + TNR}{2}$$

Now, let's consider why the G-mean is preferred over the Arithmetic mean in the context of imbalanced datasets:

Sensitivity to Imbalance:

In imbalanced datasets, the positive class (minority) is often more important, and its correct classification is crucial. The G-mean ensures that both TPR and TNR must be high to achieve a high overall score. If either TPR or TNR is low, the G-mean will be significantly affected. This property makes the G-mean sensitive to the performance on the minority class, which is often the class of interest in imbalanced datasets.

Penalizing Poor Performance:

The geometric mean tends to penalize poor performance more heavily than the arithmetic mean. For example, if $TPR = 0.9$ and $TNR = 0.1$, the G-mean will be $\sqrt{0.9 \times 0.1} \approx 0.3$, whereas the Arithmetic mean will be $(0.9 + 0.1)/2 = 0.5$. This property of the G-mean ensures that classifiers cannot achieve a high score by performing well on only one class.

Balance Between Classes:

The G-mean provides a balance between the performance on the positive and negative classes. It requires that both TPR and TNR be high for a high overall score, thus encouraging classifiers to perform well on both classes. In contrast, the Arithmetic mean could result in a high overall score even if one of the rates is low, which is not desirable in the context of imbalanced datasets.

The G-mean is better suited for imbalanced datasets because it ensures a balanced evaluation of classification performance across different classes, penalizes poor performance, and is sensitive to the performance on the minority class.

4.5.1 Baseline Result And model Accuracy:

In this section, we will see the result when we don't use any preprocessing algorithms to compare it with using oversampling algorithms. In all the experiments, we will use the k-fold cross validation k = 10, which means that each fold will have $\frac{336}{10}$ samples. And any algorithms that use k-nearest neighbors k = 5.

The baseline for this test will be based on the original data. We will apply some classification algorithms. Then in the following steps, AS, we can see in table 4.2 the Accuracy metrics: Accuracy score, F1 score, Recall score, Precision score, and g-mean score. Recall or the sensitivity is to measure the accuracy for the positive class, and specificity is a measure of the accuracy of the negative class. The g-mean measures the distinctive property of being independent of the distribution of the samples between classes, and it can see the class balance score. We used the Decision Tree, KNN, and SVM as Classification algorithms.

Alg	Accuracy	Recall	Precision	G-mean
DT	0.795	0.609	0.647	0.617
KNN	0.863	0.753	0.788	0.769
SVM	0.869	0.767	0.807	0.786

Table 4.2: The Baseline Accuracy matrix

As we can see, the Accuracy in each classifier is higher compared to the sensitivity and specificity, and both are captured in the g-mean metric, so we know the majority class dominates as the positive class.

In the next experiment, we will use an oversampling techniques on this data, to see the differences to compare it to our methods.

The classes of the data: Table 4.1 before applying oversampling in as we can see in Figure 4.5.

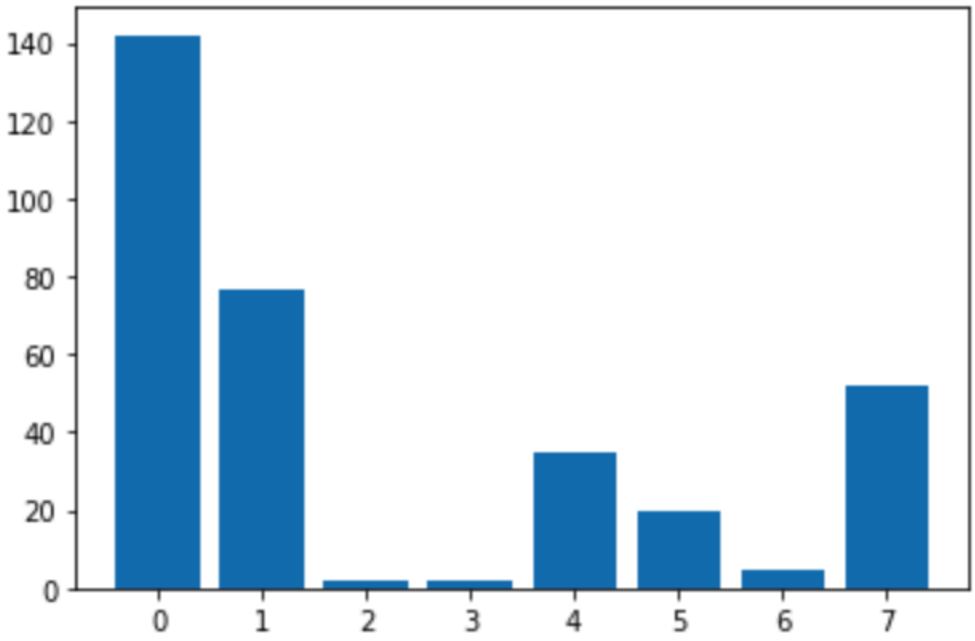


Figure 4.5: Data before applying SMOTE

4.5.2 Oversampling techniques

In this section, we will use some oversampling techniques on the same data to see how it's behave. The oversampling techniques are : SMOTE,BorderlineSMOTE and ADASYN. In table 4.3 we see the SMOTE applied on the same data.

The original data after oversampling the minority classes (7 classes out of one majority class):

Alg	Accuracy	Recall	Precision	G-mean
DT	0.936	0.937	0.940	0.938
KNN	0.955	0.954	0.956	0.955
SVM	0.893	0.890	0.897	0.894

Table 4.3: Data after applying SMOTE

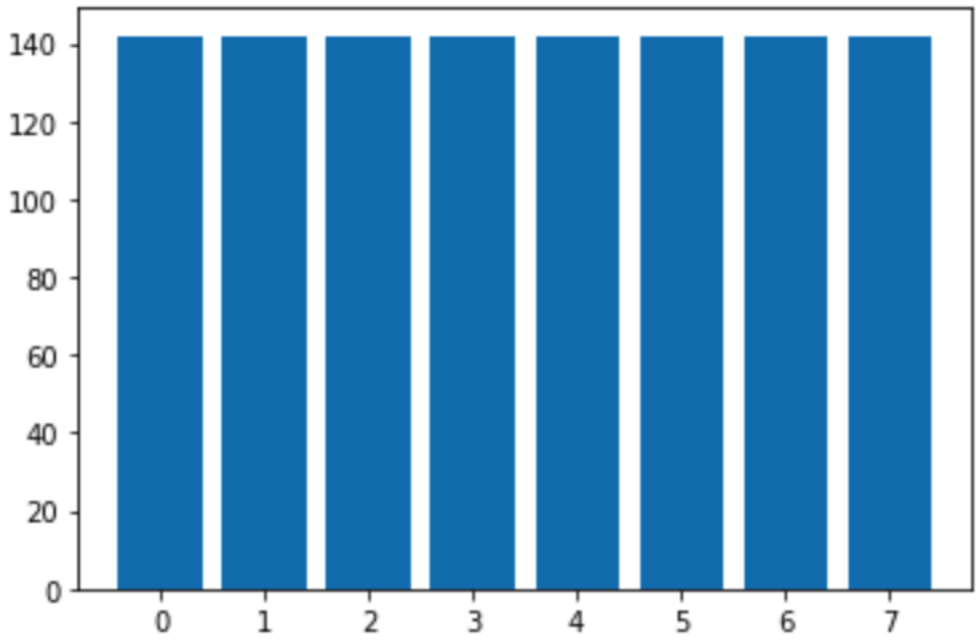


Figure 4.6: Data after applying SMOTE

In this table 4.4 we are using BorederLineSMOTE to oversamples a synthetic samples.

Alg	Accuracy	Recall	Precision	G-mean
DT	0.918	0.812	0.824	0.817
KNN	0.944	0.900	0.895	0.897
SVM	0.884	0.839	0.848	0.843

Table 4.4: Data After applying BoredlineSMOTE

Alg	Accuracy	Recall	Precision	G-mean
DT	0.851	0.713	0.720	0.716
KNN	0.901	0.794	0.818	0.805
SVM	0.907	0.777	0.804	0.790

Table 4.5: Data After applying ADASYN

4.5.3 Discussion baseline vs. Oversampling techniques

In the previous experiment, we have one class that dominates the seven other classes, making learning more difficult using standard machine learning approaches. And to complicate this, we must be careful around the overlapping features when we create the synthetic samples to balance each class. SMOTE algorithm has some limitations; it does not consider the distribution of minority classes and latent noises in the data set. It also frequently overgeneralizes the minority class(es), which may make a misclassification (Popescu *et al.*, 2014).

4.6 Preprocess steps:

The following preprocessing steps were applied to our proposed algorithm:

1. We prepare the dataset by finding the centroids for each class.
2. We apply the clustering step to find the pure cluster and the mixed clusters.
3. Then we build the tables and the trees for each distance to the centroids, for each feature with its density point to find the overlapping areas.
4. applies the oversampling techniques DCSMOTE or DCADYSAN to generate synthetic samples based on their criteria.

4.7 classification process

1. After preparing the dataset, we split the data into 10 folds to train and test the models' accuracy.

2. Create oversampling datasets using known methods (SMOTE, ADASYN, BLSMOTE, Random oversampling and (DCSMOTE or DCADASYN)).

3. Then we calculate the average of the accuracy for each fold, Sensitivity and specificity and G-mean.

In Figure 4.7 we can how the DCSMOTE works It combines oversampling with clustering and density-based sampling. Also In Figure 4.8 we can see the DCADYSAN implementation and how it's bounded by ϵ and how it can be generated oversamples based on each dense cluster.

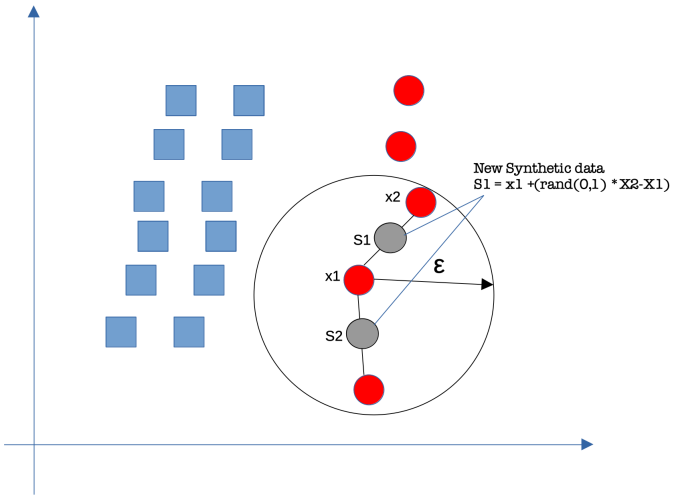


Figure 4.7: Data after applying DCSMOTE

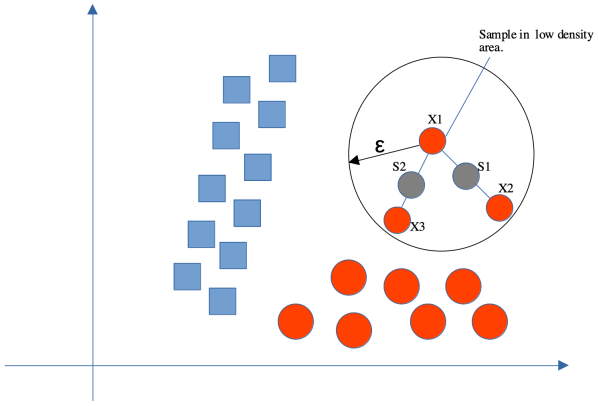


Figure 4.8: Data after applying DCADYSAN

4.7.1 Experiment I: DCSMOTE

In the First experiment, We will use the proposed approach DCSOMTE method. which is based on creating random oversampling of data points. This approach identify the overlap regions in tables 4.7 - 4.10 we can see our method outperformed other methods. we used real-world data sets 4.6 to compare other methods to our method. These experiment shows our method consistently outperformed the known methods. we used oversampling techniques: SMOTE, ADASYN, BLSMOTE, and random over-sampling. then after balancing the class(es) we used SVM classification algorithms to get the accuracy metrics.

The benefits of using this algorithm are: it separates the regions of overlapping area and the non-overlapping area. It gets the density of each area. theses are the main differences between the proposed and other methods.

Dataset	Instances	# Features	IR	Dataset	Instances	# Features	IR
Pageblocks0	5472	10	8.79	Abalone19	4174	8	129.44
Segment0	2308	19	6.02	Shuttle2vs5	3316	9	66.67
Yeast3	1484	8	8.10	Shuttle0vs4	1829	9	13.87
Yeast1	1484	8	2.46	Abalone19vs10111213	1622	8	49.69
Yeast02579vs368	1004	8	9.14	Winequalityred4	1599	11	29.17
Vowel0	988	13	9.98	Yeast4	1484	8	28.10
Vehicle0	846	18	3.25	Yeast6	1484	8	41.40
Vehicle2	846	18	2.88	Winequalitywhite39vs5	1482	11	58.28
Vehicle3	846	18	2.99	Yeast1289vs7	947	8	30.57
Vehicle1	846	18	2.90	Winequalitywhite3vs7	900	11	44.00
Pima	768	8	1.87	Winequalityred8vs67	855	11	46.50
Wisconsin	683	9	1.86	Abalone0918	731	8	16.40
Yeast05679vs4	528	8	9.35	Yeast1458vs7	693	8	22.10
Yeast2vs4	514	8	9.08	Winequalityred3vs5	691	11	68.10
Ecoli3	336	7	8.60	Winequalityred8vs6	656	11	35.44
Ecoli2	336	7	5.46	Abalone21vs8	581	8	40.50
Ecoli1	336	7	3.36	Yeast2vs8	482	8	23.10
Haberman	306	3	2.78	Pageblocks13vs2	472	10	15.86
Ecoli0267vs35	224	7	9.18	Yeast1vs7	459	7	14.30
Ecoli067vs35	222	7	9.09	Led7digit02456789vs1	443	7	10.97
Ecoli067vs5	220	6	10.00	Dermatology6	358	34	16.90
Ecoli0vs1	220	7	1.86	Ecoli4	336	7	15.80
Newthyroid1	215	5	5.14	Ecoli0147vs2356	336	7	10.59
Newthyroid2	215	5	5.14	Ecoli0137vs26	281	7	39.14
Glass6	214	9	6.38	Ecoli0146vs5	280	6	13.00
Glass0123vs456	214	9	3.20	Ecoli01vs5	240	6	11.00
Glass1	214	9	1.82	Glass5	214	9	22.78
Glass0	214	9	2.06	Glass4	214	9	15.46
Ecoli0346vs5	205	7	9.25	Glass2	214	9	11.59
Ecoli046vs5	203	6	9.15	Glass0146vs2	205	9	11.06
Glass015vs2	172	9	9.12	Glass016vs5	184	9	19.44
Iris0	150	4	2.00	Cleveland0vs4	177	13	12.62
Glass04vs5	92	9	9.22	Shuttle2vs4	129	9	20.50

Table 4.6: Datasets taken from UCI and KEEL

Accuracy Values						
Dataset	Baseline	SMOTE	ADASYN	BLSMOTE	Random Oversampling	DCSMOTE
Led7digit02456789vs1	0.893	0.918	0.914	0.940	0.969	**0.993**
Glass0146vs2	0.988	0.984	0.986	0.989	0.989	**0.993**
Cleveland0vs4	0.535	0.977	0.975	0.972	0.979	**0.998**
Shuttle0vs4	0.544	0.982	0.989	0.987	0.948	**0.956**
Abalone0918	0.954	0.887	0.921	0.911	0.954	**0.984**
Vowel0	0.740	0.916	0.926	0.957	0.958	**0.977**
Ecoli0147vs2356	0.859	0.908	0.930	0.948	0.968	**0.998**
Pageblocks0	0.892	0.958	0.983	0.979	0.992	**0.997**
Yeast02579vs368	0.619	0.779	0.792	0.795	0.871	**0.906**
Ecoli067vs5	0.591	0.925	0.960	0.939	0.966	**0.985**

Table 4.7: Accuracy In Experiment DCSMOTE

Sensitivity Values						
Dataset	Baseline	SMOTE	ADASYN	BLSMOTE	Random Oversampling	DCSMOTE
Led7digit02456789vs1	0.488	0.680	0.916	0.934	0.976	**0.986**
Glass0146vs2	0.495	0.557	0.985	0.989	0.988	**0.997**
Cleveland0vs4	0.527	0.975	0.980	0.968	0.980	**0.996**
Shuttle0vs4	0.460	0.846	0.985	0.988	0.953	**0.964**
Abalone0918	0.537	0.839	0.917	0.914	0.952	**0.989**
Vowel0	0.492	0.845	0.931	0.957	0.963	**0.972**
Ecoli0147vs2356	0.487	0.759	0.933	0.957	0.961	**0.994**
Pageblocks0	0.481	0.915	0.979	0.983	0.995	**0.997**
Yeast02579vs368	0.498	0.764	0.807	0.813	0.861	**0.884**
Ecoli067vs5	0.509	0.914	0.953	0.930	0.964	**0.973**

Table 4.8: Sensitivity Values Experiment DCSMOTE

Specificity Values						
Dataset	Baseline	SMOTE	ADASYN	BLSMOTE	Random Oversampling	DCSMOTE
Led7digit02456789vs1	0.483	0.636	0.921	0.933	0.972	**0.991**
Glass0146vs2	0.538	0.519	0.985	0.989	0.987	**0.990**
Cleveland0vs4	0.491	0.964	0.988	0.966	0.979	**0.998**
Shuttle0vs4	0.498	0.896	0.985	0.990	0.954	**0.956**
Abalone0918	0.687	0.819	0.930	0.912	0.956	**0.985**
Vowel0	0.581	0.862	0.934	0.965	0.961	**0.971**
Ecoli0147vs2356	0.534	0.790	0.938	0.947	0.966	**0.997**
Pageblocks0	0.482	0.877	0.986	0.980	0.994	**0.997**
Yeast02579vs368	0.452	0.737	0.806	0.809	0.877	**0.909**
Ecoli067vs5	0.507	0.887	0.949	0.935	0.963	**0.977**

Table 4.9: specificity Values In Experiment DCSMOTE

G-Mean Values						
Dataset	Baseline	SMOTE	ADASYN	BLSMOTE	Random Oversampling	DCSMOTE
Led7digit02456789vs1	0.487	0.653	0.918	0.933	0.974	**0.988**
Glass0146vs2	0.496	0.534	0.985	0.989	0.987	**0.994**
Cleveland0vs4	0.505	0.969	0.984	0.967	0.979	**0.997**
Shuttle0vs4	0.472	0.870	0.985	0.989	0.953	**0.960**
Abalone0918	0.598	0.829	0.924	0.913	0.954	**0.987**
Vowel0	0.530	0.851	0.932	0.961	0.962	**0.972**
Ecoli0147vs2356	0.506	0.773	0.935	0.952	0.963	**0.995**
Pageblocks0	0.481	0.895	0.982	0.981	0.995	**0.997**
Yeast02579vs368	0.466	0.750	0.806	0.811	0.869	**0.896**
Ecoli067vs5	0.503	0.898	0.951	0.932	0.963	**0.975**

Table 4.10: G-mean Values Experiment DCSMOTE

4.7.2 Experiment II: DCADASYN

In the second experiment, We will use the proposed approach DCADASYN method, which is based on density clustering with considering overlap area. We evaluated the model using as discussed in previous sections. We used some algorithms to compare the result as seen in tables:(4.11 - 4.14). The first column is the baseline classifier, then we used other classier to check the Accuracy. The dataset are taken from table 4.6. In all the classifiers, we used the oversampling technique and using the SVM classifier.

In Experiment II the results of 10 datasets taken from table 4.6 with some imbalance and overlap degree, DCADASYN achieved the top performance in all the performance matrices, And we got the highest result, and this is attributed to this algorithm focus more on the overlapping.

Most of the oversampling techniques are oversample sample of the data regards of the location of the data point except from some exception like BLSMOTE.

From tables 4.11 - 4.14 we can see the baseline has achieved poorly by guessing the majority all the time, instated by considering other classes. by oversampling the minority class(es). we achieved an increased performance and the top performance is the proposed algorithm.

Accuracy Values						
Dataset	Baseline	SMOTE	ADASYN	BLSMOTE	Random Oversampling	DCADASYN
Winequalityred8vs67	0.573	0.772	0.786	0.781	0.852	**0.892**
Shuttle2vs5	0.873	0.908	0.896	0.938	0.961	**0.997**
Abalone19vs10111213	0.934	0.938	0.957	0.962	0.987	**0.999**
Winequalitywhite39vs5	0.913	0.910	0.926	0.934	0.978	**0.998**
Ecoli0137vs26	0.821	0.951	0.965	0.972	0.983	**0.986**
Pageblocks13vs2	0.923	0.926	0.950	0.945	0.990	**0.998**
Shuttle2vs4	0.797	0.945	0.958	0.960	0.980	**0.985**
Glass5	0.935	0.941	0.961	0.971	0.984	**0.996**
Yeast6	0.947	0.976	0.989	0.987	0.993	**0.995**
Abalone19	0.960	0.967	0.969	0.980	0.993	**0.997**

Table 4.11: Accuracy Values using DCADASYN

Sensitivity Values						
Dataset	Baseline	SMOTE	ADASYN	BLSMOTE	Random Oversampling	DCADASYN
Winequalityred8vs67	0.490	0.770	0.789	0.785	0.856	**0.881**
Shuttle2vs5	0.492	0.913	0.899	0.933	0.966	**0.994**
Abalone19vs10111213	0.537	0.938	0.956	0.958	0.985	**0.998**
Winequalitywhite39vs5	0.493	0.916	0.930	0.931	0.975	**0.997**
Ecoli0137vs26	0.508	0.951	0.967	0.969	0.981	**0.989**
Pageblocks13vs2	0.478	0.925	0.947	0.953	0.993	**0.999**
Shuttle2vs4	0.484	0.945	0.959	0.962	0.981	**0.985**
Glass5	0.515	0.940	0.956	0.970	0.983	**0.996**
Yeast6	0.494	0.978	0.989	0.988	0.993	**0.995**
Abalone19	0.490	0.968	0.970	0.984	0.995	**0.996**

Table 4.12: Sensitivity Values DCADASYN

Specificity Values						
Dataset	Baseline	SMOTE	ADASYN	BLSMOTE	Random Oversampling	DCADASYN
Winequalityred8vs67	0.503	0.764	0.784	0.782	0.865	**0.890**
Shuttle2vs5	0.512	0.914	0.895	0.928	0.967	**0.995**
Abalone19vs10111213	0.501	0.941	0.955	0.957	0.986	**0.997**
Winequalitywhite39vs5	0.529	0.913	0.934	0.931	0.980	**0.997**
Ecoli0137vs26	0.478	0.953	0.962	0.973	0.982	**0.989**
Pageblocks13vs2	0.492	0.922	0.951	0.951	0.991	**0.998**
Shuttle2vs4	0.488	0.944	0.959	0.958	0.981	**0.984**
Glass5	0.510	0.940	0.956	0.966	0.982	**0.995**
Yeast6	0.495	0.979	0.988	0.988	0.994	**0.996**
Abalone19	0.488	0.966	0.969	0.981	0.993	**0.997**

Table 4.13: Specificity Values DCADASYN

G-Mean Values						
Dataset	Baseline	SMOTE	ADASYN	BLSMOTE	Random Oversampling	DCADASYN
Winequalityred8vs67	0.495	0.767	0.786	0.784	0.860	**0.886**
Shuttle2vs5	0.498	0.914	0.897	0.930	0.967	**0.994**
Abalone19vs10111213	0.516	0.939	0.956	0.957	0.985	**0.998**
Winequalitywhite39vs5	0.507	0.914	0.932	0.931	0.977	**0.997**
Ecoli0137vs26	0.490	0.952	0.964	0.971	0.982	**0.989**
Pageblocks13vs2	0.485	0.923	0.949	0.952	0.992	**0.998**
Shuttle2vs4	0.485	0.945	0.959	0.960	0.981	**0.985**
Glass5	0.510	0.940	0.956	0.968	0.982	**0.995**
Yeast6	0.494	0.979	0.988	0.988	0.993	**0.996**
Abalone19	0.489	0.967	0.970	0.983	0.994	**0.997**

Table 4.14: G-Mean Values Using DCADASYN

4.7.3 Conclusions

In this chapter, extensive experimentation was carried out to test the proposed algorithms DCSMOTE and DCADASYN use an oversampling technique with real-world datasets, and achieved the highest performance, Compared to other exited oversampling techniques. but the difference is that all other methods do not consider both overlapping and imbalance problems simultaneously. Considering both overlapping and imbalanced it's helped the algorithm to learn more from the data. Both methods create synthetic and more data points in the minority class(es). In detecting the overlapping area we gave the algorithm the region to not create more samples. This improvement will be bigger the more we have a large overlapping the more we see higher performance compared to other methods. In the proposed method we consider the density of the data and its shape of it. the can give us a better understanding of what we have.

The first method DCSMOTE uses a random synthetic data point to create more data points in the lower dense area, in the second method DCADASYN it's using the calculated oversampling method in the lower dense area.

Next chapter we will see the problem if we have a large number of features or if the data has a high dimensional space, and how to cope with these issues.

Chapter 5

Features selection and dimensionality reduction

5.1 Introduction

In this chapter, we will present methods for feature selection and dimensionality reduction in datasets that are imbalanced and exhibit feature overlap. we will introduce the method based on the trading of the most gain from the feature if the $overlapfeatures[i]\&imbalanced < overlapfeature[j]\&imabalanced$. So by selecting the feature that makes the Accuracy of the perdition more accurate. Feature selection was used to avoid the class imbalance problems (Wasikowski and Chen, 2009) this work studied the different methods for imbalanced data classification problems. In the next sections, we will see the related work, a comparison between the proposed algorithm and others in relation of the overlapping imbalanced dataset.

5.2 Background

Feature selection methods have been widely applied in research and are a rich subject in the literature. This can be expressed as eliminating the irrelevant and insignificant or redundant features and keeping the most important and optimal features that will produce a better characterization of the patterns for each class. the main reasons for feature selection are to avoid overfitting and reduce the dimension space of the dataset (Guyon and Elisseeff, 2003). There are two main approaches for features selection :

- Filter based approaches.
- Wrapper approach.
- Hybrid method which combine the two previous approaches.

The feature selection problem has been studied by statistics and machine learning researchers for many years. But it has received more attention recently because of discoveries in research in data mining. According to (John *et al.*, 1994) definition, and (Sato *et al.*, 1998) can be labeled as filter models, while in (Caruana and Freitag, 1994) research is classified as wrapped methods. Sometimes feature selection is referred to as subset selection.

5.2.1 Filter based Methods

The filter method is a technique for feature selection in machine learning. It involves selecting a subset of the most relevant features from the original dataset to use in

the model. The goal is to improve the performance of the model by reducing the dimensionality of the data, eliminating irrelevant or redundant features, and reducing the computational complexity of the model. There are several ways to implement the filter method, but one common approach is to use statistical tests to evaluate the relevance of each feature. This can be done by calculating the correlation between each feature and the target variable and selecting only the features that have a high correlation. Other methods include using information gain or chi-squared tests (Witten *et al.*, 2005) to evaluate the relevance of each feature. The selected features are then used to train the machine learning model.

Filter-based methods are used in the preprocessing phase, which will filter the features to choose the best features that increased the accuracy of the output model. some of the filters that can be applied to selecting the features are Information, consistency, distance, similarity, and statistical measures. the feature with the highest rank will be chosen and those with the lowest rank are removed(Dash *et al.*, 2002) (Hall, 2000) (Liu *et al.*, 1996).

The steps for filtering features and dimensionality reduction vs Features overlapping is shown in figure 5.1 we will see it in coming sections.

Information gain IG or sometimes called mutual information in equation 5.2.1 is measuring a relationship between two random variables that sample simultaneously, which calculates the expected reduction in information entropy. let's say for example we have many features for each class $F_1, F_2, \therefore, F_n$. Then we would determine the information gain for each feature (F), $IG(F_1), IG(F_2)$ and so on. After that we rank the features based on the information gained, we can decide on the threshold that we want to select and include all the features that are above the chosen threshold(Lee

and Lee, 2006).

Many feature selection approaches have been proposed in the literature. (Kira and Rendell, 1992) introduced the feature selection method called Relief which randomly selects data points from the dataset and updates the rank of each feature based on the difference between the selected instances and the nearest data point from the opposite class.

The Multi-label Relief Feature (MLReliefF) (Xie *et al.*, 2017) method is a feature selection method for multi-label classification tasks, where each instance can have multiple labels. It is an extension of the Relief algorithm.

The MLReliefF method works by training a multi-label classifier on the dataset, and then calculating the relevance of each feature based on how well the classifier can predict the labels for each instance. The relevance of each feature is calculated by comparing the predicted labels for each instance to the actual labels, and measuring the distance between the two. The features with the smallest distances are considered to be the most relevant, as they have the greatest impact on the classifier's ability to make accurate predictions.

The MLReliefF method is a useful tool for feature selection in multi-label classification tasks, as it can capture complex relationships between the features and the labels. However, it can be computationally expensive, as it involves training multiple classifiers to evaluate the relevance of each feature. Additionally, the performance of the method can be sensitive to the choice of classifier and the performance measure used to evaluate the relevance of each feature.

The chi-squared test is a statistical test that measures the association between two variables, and is often used in feature selection to evaluate the relationship between

a feature and the target variable. It is based on the chi-squared statistic, which is calculated by comparing the observed frequencies of the feature values to the expected frequencies under the null hypothesis.

To use the chi-squared test for feature selection, the following steps can be followed:

- Calculate the chi-squared statistic for each feature by comparing the observed frequencies of the feature values to the expected frequencies under the null hypothesis.
- Compute the p-value for each feature by using the chi-squared statistic and the degree of freedom, the maximum number of logically independent values.
- Select only the features with a p-value below a certain threshold, such as 0.05 or 0.01, to use in the final model.
- Train the machine learning model using the selected features.

The chi-squared test is a useful tool for feature selection, as it can capture complex relationships between the features and the target variable. However, it can be sensitive to the distribution of the data, and may not be appropriate for all types of data. Additionally, the performance of the method can be sensitive to the choice of threshold for the p-value.

There are several advantages to using the filter method for feature selection. Some of the main benefits include:

- Improved model performance: By selecting only the most relevant features, the filter method can improve the performance of the machine learning model by reducing overfitting and increasing the accuracy of predictions.

- Reduced computational complexity: The filter method can reduce the computational complexity of the model by eliminating irrelevant or redundant features, which can save time and resources during training and inference.

- Enhanced interpretability: By reducing the dimensionality of the data, the filter method can make the machine learning model more interpretable, which can be useful for understanding the underlying patterns in the data.

- Improved generalizability: By removing irrelevant or redundant features, the filter method can improve the generalizability of the model, which can be beneficial when applying the model to new or unseen data.

Overall, the filter method can be a useful tool for improving the performance, interpretability, and generalizability of machine learning models.

There are also some disadvantages to using the filter method for feature selection. Some of the main drawbacks include:

- Loss of information: By selecting only a subset of the original features, the filter method can potentially remove important information from the dataset, which can negatively impact the performance of the machine learning model.

- Subjectivity: The selection of features using the filter method can be subjective, and different people may select different subsets of features based on their individual criteria. This can lead to inconsistencies and variations in the results.

- Limited flexibility: The filter method is generally limited to selecting a fixed number of features, which can be inflexible and may not be suitable for all types of data and models.

- Inability to capture complex relationships: The filter method is based on simple statistical tests, which may not be able to capture complex relationships between the features and the target variable. This can limit the ability of the method to identify the most relevant features.

Overall, while the filter method can be useful in some cases, it is important to carefully consider its limitations and potential drawbacks before using it for feature selection.

$$IG(F) = E(S) - \sum_{n} \frac{S_n}{S} E(S_n) \tag{5.2.1}$$

Where F are the features or the attributes, S are the subset of the data $E(S)$ is the entropy of the given dataset and $E(S_n)$ are the entropy of n^th generated by creating subsets of the dataset.

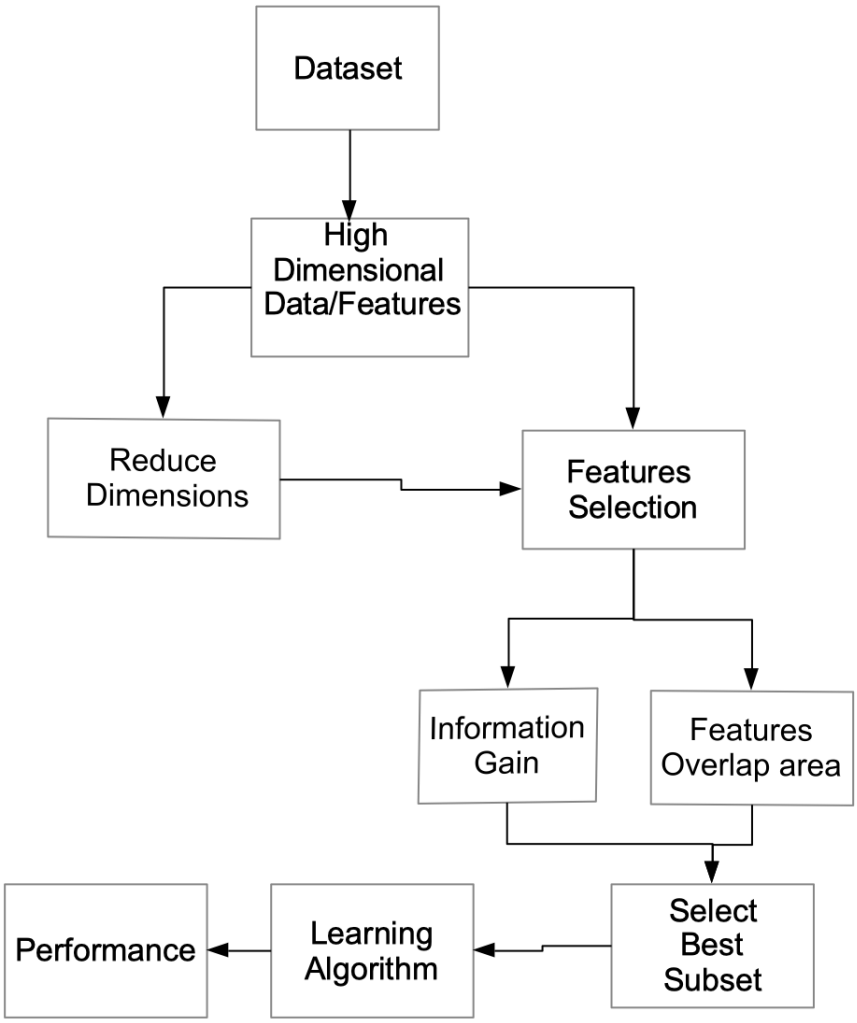


Figure 5.1: Trade off in feature selection and dimensionality reduction

5.2.2 Wrapper methods

The wrapper approach to feature selection is a method of evaluating the usefulness of a set of features in a machine learning model by considering their performance within the model. It involves training a model using a subset of features, evaluating its performance, and then iteratively adding or removing features to improve the

model's performance (Ron and George, 1997).

One of the main advantages of the wrapper approach is that it considers the interaction between features, which can be important in determining the usefulness of a feature. In contrast, the filter approach to feature selection evaluates features independently and does not consider the interaction between features.

There are several different methods that can be used within the wrapper approach, including forward selection, backward selection, and recursive feature elimination. Forward selection starts with an empty set of features and adds one feature at a time, evaluating the performance of the model after each addition. Backward selection starts with all features and removes one feature at a time, evaluating the performance of the model after each removal. Recursive feature elimination repeatedly trains a model using a subset of features, and then removes the least important feature until the desired number of features is reached.

Forward selection is a wrapper method for feature selection that involves starting with an empty set of features and adding one feature at a time, evaluating the performance of the model after each addition. The process is repeated until the desired number of features is reached.

Here is an example of how forward selection might be implemented in a machine learning model:

1. Start with an empty set of features.

2. Train a model using the empty set of features.

3. Calculate the evaluation metric (e.g., accuracy, precision, recall) for the model.

4. Add the feature that results in the greatest improvement in the evaluation

metric.

5. Repeat steps 2-4 until the desired number of features is reached.

Forward selection is a useful method for identifying the most relevant features for a machine learning model, but it can be computationally expensive and is sensitive to the choice of evaluation metric and machine learning algorithm. One of the main challenges of the wrapper approach is that it can be computationally expensive, as it requires training and evaluating multiple models. It is also sensitive to the choice of the evaluation metric and the machine learning algorithm used, and may not always result in the optimal set of features.

Wrapper methods and filter methods are two approaches to feature selection, which is the process of identifying the most relevant and informative features for a machine learning model.

Wrapper methods evaluate the performance of a subset of features within a machine learning model. This is done by training and evaluating multiple models using different subsets of features, and then selecting the subset that results in the best performance. Wrapper methods consider the interaction between features and are able to identify complex relationships within the data. However, they can be computationally expensive and are sensitive to the choice of evaluation metric and machine learning algorithm.

Filter methods, on the other hand, evaluate the importance of each feature independently, without considering the interaction between features. They use statistical or heuristic measures to assess the relevance of each feature, and then select a subset of the most relevant features. Filter methods are generally faster and more efficient than wrapper methods, but they may not be able to identify complex relationships

within the data and may result in a suboptimal set of features.

In summary, wrapper methods and filter methods are two approaches to feature selection that have different strengths and limitations. Wrapper methods consider the interaction between features and are able to identify complex relationships within the data, but are computationally expensive and sensitive to the choice of evaluation metric and machine learning algorithm. Filter methods are faster and more efficient, but may not be able to identify complex relationships within the data and may result in a suboptimal set of features.

5.3 Dimensional reduction

Dimensionality reduction is a technique used in machine learning and data analysis to reduce the number of variables or features in a dataset.The goal of dimension reduction is to reduce the complexity of the data while still preserving the important information contained in the data. This can be useful in a number of different scenarios, such as improving the performance of a machine learning model, reducing the amount of data that needs to be stored or processed, or making the data more interpretable for humans.

There are two main types of dimension reduction techniques: feature selection and feature extraction. Feature selection methods select a subset of the original features in a dataset, while feature extraction methods create new features from the original features. Both types of methods can be unsupervised, where the reduction is performed without using any information about the target variable, or supervised, where the reduction is performed using information about the target variable.

One common method for performing dimension reduction is principal component

analysis PCA (Wold *et al.*, 1987), which is a linear dimension reduction technique. PCA is an unsupervised method that uses singular value decomposition SVD (Golub and Van Loan, 2013)to transform the original data into a new set of orthogonal features, known as principal components, that are ranked in order of importance. The principal components are a set of linearly uncorrelated variables that capture the maximum variance in the original data.

Another popular method for dimension reduction is t-distributed stochastic neighbor embedding t-SNE (Van der Maaten and Hinton, 2008) , which is a nonlinear dimension reduction technique. t-SNE is a supervised method that uses a probabilistic model to map the original data points to a lower-dimensional space, where similar points are grouped together and dissimilar points are separated. t-SNE is often used for visualizing high-dimensional data in two or three dimensions, and has been shown to produce more interpretable and visually appealing results compared to other dimension reduction methods.

In addition to PCA and t-SNE, there are many other methods for performing dimension reduction, each with its own strengths and weaknesses. Some other popular methods include linear discriminant analysis LDA (Fisher, 1936), independent component analysis ICA (Bell and Sejnowski, 1995), and autoencoders. These methods all have different underlying assumptions and algorithms, and may be more or less appropriate for different types of data and tasks.

The formula for principal component analysis (PCA) is given by the equation $X = T \times S \times P^T$, where X is the original data matrix, T is the matrix of principal components, S is the diagonal matrix of singular values, and P is the matrix of feature vectors. This equation shows how the original data matrix can be decomposed into

the product of these three matrices using singular value decomposition (SVD).

To use this formula, you can simply substitute the values of the matrices X, T, S, and P into the equation and perform the matrix multiplication to obtain the resulting matrix. This will give you the decomposition of the original data matrix into the product of the principal components, singular values, and feature vectors. This decomposition can be used for many different purposes, such as reducing the dimensionality of the data, extracting important features or patterns from the data, or solving linear equations.

Singular value decomposition (SVD) is a matrix factorization technique that decomposes a matrix into the product of three matrices. It is a widely used mathematical tool that has applications in many different fields, including machine learning, data analysis, and signal processing.

SVD is a powerful tool that can be used for a variety of different tasks, such as solving linear equations, finding the rank of a matrix, or calculating the principal components of a dataset. It is often used in machine learning and data analysis to reduce the dimensionality of the data, and to extract important features or patterns from the data.

$$A = U \times \Sigma \times V^T \tag{5.3.1}$$

In this equation, A represents the original data matrix, U represents the left singular vectors, Sigma represents the diagonal matrix of singular values, and V^T represents the right singular vectors. This equation shows how the original data matrix A can be decomposed into the product of these three matrices using SVD.

5.4 Features overlapping and Dimensions reduction

Feature overlap can occur not only between different features within a single class but also between features from different classes in a dataset. For example, consider a dataset with two classes, "apple" and "orange", and two features, "color" and "shape". If the "color" feature is highly correlated with the class label (i.e. apples are always red and oranges are always orange), then the "color" feature may not provide any additional information when trying to distinguish between the two classes. Similarly, if the "shape" feature is highly correlated with the "color" feature (i.e. all red objects are round and all orange objects are oval), then the "shape" feature may not provide any additional information when trying to distinguish between the two classes.

In this case, features overlapping between the "color" and "shape" features can lead to poor performance of a machine learning model, as it may rely too heavily on the "color" feature and not be able to accurately classify objects based on their shape. To address this issue, one possible solution would be to apply a dimensionality reduction technique, such as feature selection or feature extraction, to identify and remove the overlapping features from the dataset. This would leave a smaller and more relevant set of features that could be used to train a more accurate and robust machine learning model.

Dimensionality reduction is the process of reducing the number of features in a dataset by selecting a subset of the most relevant or informative features. This can be useful for improving the performance of machine learning models, reducing

overfitting, and making the data easier to visualize and interpret.

There are two main types of dimensionality reduction techniques: feature selection and feature extraction.

Feature selection involves selecting a subset of the original features based on some criterion, such as their relevance to the target variable or their correlation with other features. This can be done using a variety of methods, such as filter methods, which evaluate the relevance of each feature independently, or wrapper methods, which evaluate the relevance of subsets of features by training a machine learning model on the data.

Feature extraction, on the other hand, involves transforming the original features into a new set of features that capture the most important information in the data. This can be done using techniques such as principal component analysis (PCA), which creates a new set of features that are linear combinations of the original features, or non-negative matrix factorization (NMF), which creates a new set of features that are non-negative combinations of the original features.

Overall, dimensionality reduction can be a powerful tool for improving the performance and interpretability of machine learning models, and it is often used in a wide range of applications, from image and text analysis to speech recognition and natural language processing.

5.5 Feature selection and dimensionality reduction vs overlapping and imbalanced dataset

This section describes an algorithm used in our experiment that combines feature selection and feature extraction to reduce dimensionality in datasets characterized by overlapping features and imbalanced classes:

1. Start with the balanced dataset, which contains a set of features and a target variable.

2. Apply a feature selection method to the dataset to identify and remove overlapping features. This could be done using a filter method, such as the chi-squared test, or a wrapper method, such as recursive feature elimination (RFE).

3. Once the overlapping features have been removed, apply a feature extraction method to the remaining features to create a new set of features that capture the most important information in the data. This could be done using a technique such as principal component analysis (PCA) or non-negative matrix factorization (NMF).

4. Train a machine learning model on the new set of features, and evaluate its performance on a test dataset.

5. If the performance of the model is not satisfactory, repeat steps 2-4 with different settings for the feature selection and feature extraction methods, until an optimal set of features is found.

6. Use the final set of features to train a final machine learning model, and use it to make predictions on new data.

5.6 Feature selection based on Features overlapping

This section proposes a theoretical framework for feature selection in multidimensional datasets, focusing on optimizing a balance between minimizing feature overlap and maximizing information gain.

5.6.1 Definitions

Definition 1 (Feature Overlap): Feature overlap quantifies the extent to which two or more features provide redundant information. Given two features F_i and F_j, their overlap can be defined as:

$$\text{Overlap}(F_i, F_j) = \frac{|\{x \mid x \in F_i \cap F_j\}|}{|\{x \mid x \in F_i \cup F_j\}|} \tag{5.6.1}$$

Definition 2 (Information Gain): Information Gain for a feature F in a dataset D is the reduction in entropy or surprise by partitioning the data according to F. It is calculated as:

$$IG(D, F) = H(D) - \sum_{v \in Values(F)} \frac{|D_v|}{|D|} H(D_v) \tag{5.6.2}$$

where $H(D)$ is the entropy of the dataset, and D_v is the subset of D where feature F has value v.

5.6.2 Feature Selection Criterion

Feature selection aims to choose a subset of features that maximizes information gain while keeping the pairwise feature overlap below a defined threshold θ.

5.7 Feature Selection Methodology

5.7.1 Measurement of Overlap and Information Gain

Proposition 1: The effectiveness of a feature set in classification tasks is inversely proportional to the average overlap and directly proportional to the average information gain of the features.

5.7.2 Threshold-Based Feature Selection

Definition 3 (Selection Threshold): The threshold θ is a user-defined parameter that determines the acceptable level of feature overlap.

Proposition 2: Features are selected if their pairwise overlap with any already selected feature is less than θ and their information gain is above a certain percentile of the information gain distribution.

5.8 Algorithm for Feature Selection

5.8.1 Algorithm Description

The proposed algorithm iteratively selects features based on the criteria of minimal overlap and maximal information gain, using the threshold θ.

5.9 Feature Selection Algorithm Based on Overlap and Information Gain

5.9.1 Algorithm to select feature

The algorithm presented in 6 is our proposed method for feature selection. It is designed to select features by minimizing overlap and maximizing information gain within a specified threshold. Initially, the algorithm sorts the features from the set F based on their information gain. It then iteratively adds a feature to the selected set S only if its overlap with the previously selected features is below the threshold θ. This strategy ensures that the chosen features are both informative possessing high information gain and diverse exhibiting low overlap with each other.

Algorithm 6 Feature Selection based on Overlap and Information Gain

Input: Dataset D, Feature set F, Overlap threshold θ
Output: Selected feature subset S
Initialize an empty set S for selected features
Calculate information gain IG for each feature in F
Sort F based on IG in descending order **for** *each feature* f *in* F **do**
  Calculate overlap O between f and each feature in S **if** $O < \theta$ *for all features in* S **then**
    Add f to S
return S

5.10 Experiments

This section examines various oversampling techniques applicable to datasets with a high number of features and large dimensions. One of the major issues with such datasets is the class imbalance, where one class has significantly more instances than

the other. To address this, we will first balance the classes by oversampling the minority and do a trade off between the features with overlapping . We need to choose between different options, and each option has its own advantages and disadvantages. In the context of oversampling techniques and reducing the number of features or dimensions, there is a trade-off between achieving class balance and maintaining the accuracy of predictions. Balancing the classes can improve the performance of machine learning models, but reducing the number of features or dimensions can decrease the accuracy of predictions. Therefore, it is important to carefully evaluate the trade-offs and choose the best approach based on the specific task and dataset at hand.

Once the classes are balanced, we will move on to reducing the number of features or dimensions in the dataset. This is often necessary to improve the performance of machine learning models, especially when working with large datasets. However, this reduction in dimensionality can sometimes lead to a decrease in the accuracy of the predictions. Therefore, we will investigate various feature selection and dimensionality reduction techniques that can be used to decrease the number of features or dimensions without compromising the accuracy of the predictions.

The choice to use Principal Component Analysis (PCA) instead of t-distributed Stochastic Neighbor Embedding (t-SNE) for dimensionality reduction in this research is guided by several considerations, primarily focused on computational efficiency and the suitability of the method for subsequent learning algorithms.

Firstly, PCA is known for its computational efficiency compared to t-SNE. PCA operates through linear algebra techniques involving the eigenvalue decomposition of the data's covariance matrix, which can be computed relatively quickly even with

large datasets. This efficiency makes PCA an attractive option for preprocessing in machine learning, where quick transformations of large datasets are often required.

Secondly, PCA (Gewers *et al.*, 2021) is particularly effective in preserving the global structure of the data. It reduces dimensionality by transforming features into principal components that explain the maximum variance, thereby simplifying the complexity of the data without losing crucial information. This characteristic is beneficial for models that depend on the variance explained by the data. In contrast, t-SNE focuses on preserving local structures and is primarily used for visualizing high-dimensional data in one or two dimensions. While t-SNE excels at revealing patterns at the micro-level, it may distort the global perspective of the data, which can be problematic for predictive modeling.

Additionally, PCA (Wu *et al.*, 2018) provides a level of simplicity and interpretability that is advantageous in many analytical contexts. The principal components generated by PCA are linear combinations of the original features, making it easier to understand how different features influence the model. This is less straightforward with t-SNE, which involves complex cost functions and non-linear transformations that do not yield easily interpretable components.

Considering these factors, PCA was chosen for its ability to efficiently handle large datasets while maintaining a balance between data simplification and structural preservation. This approach ensures that the dimensionality reduction phase supports rather than hinders the performance of subsequent machine learning algorithms, aligning with the research goal of demonstrating the proposed method's efficacy across diverse computational scenarios.

5.10.1 Dataset

In this experiment, we selected a dataset from the UCI machine learning repository. The selected dataset has a large number of features and high dimensions. Some data is multi-class and has an imbalanced distribution with varying degrees of overlap.

Dataset Name	Feature numbers	Instance numbers	Minority Class %	Majority Class %
Optical Recognition of Handwritten Digits	64	5620	9.9	90.1
Musk	168	6598	15.4	84.6
Semeion Handwritten Digit	256	1593	9.73	81.27
Activity recognition using wearable physiological measurements	533	4480	8.14	91.86
Internet Advertisements	1558	3279	16.15	83.75

Table 5.1: Dataset with large features and high dimensions

The baseline comparison we have selected involves evaluating the AUC metric on the dataset after balancing it using the oversampling method with our technique. The next step is to identify and select the features that result in the highest AUC value, and the second experiment is to keep the number of dimensions used to a minimum that maximize the accuracy of the predication that match the accuracy of using all the features.

5.10.2 Experiment I: Feature selection

In this experiment, we will utilize a dataset with a large number of features and compare the accuracy achieved by all the features with that of selected features with highest information gain that achieve a similarly high accuracy, as well as a smaller feature space. In Table 5.2 In this comparison, we use the SVM classifier to calculate AUC metric to evaluate accuracy. We first examine the accuracy without applying any method, followed by the application of oversampling techniques and in the next column we applied our oversampling method to the dataset. Finally, we select a number of features for comparison with the accuracy achieved using all features, shown in the last two columns. In choosing the features we selected the features that can contribute to the accuracy and has less overlapping.

Feature selection is a valuable machine learning technique that involves identifying the most relevant features for a model. The objective of feature selection is to reduce the number of features used in a model while maintaining or enhancing its performance. This is especially helpful when dealing with large datasets that have numerous features, as it can save both memory and time.

While including more features can increase accuracy, it can also escalate the model's complexity and training time. Feature selection helps to mitigate this problem by eliminating redundant or irrelevant features, which can improve the model's accuracy while lowering its complexity. This necessitates fewer resources to train the model, which saves both memory and time.

Furthermore, feature selection can improve the generalization of the model. When a model is trained on all features, it may pick up noise in the data, which can negatively affect its performance on new data. By utilizing feature selection, the model

trains on a subset of the most relevant features, which can improve its generalization and performance on new data. This is particularly advantageous when dealing with datasets that have numerous irrelevant or redundant or overlapping features, which can harm the model's performance.

Dataset Name	Baseline(AUC)	SMOTE (AUC)	DCADASYN(AUC)	#FS(AUC)	#FS(AUC)
Optical Recognition of Handwritten Digits	0.83	0.93	0.96	10(0.76)	40(0.91)
Musk	0.78	0.90	0.94	30(0.81)	80(0.89)
Semeion Handwritten	0.79	0.88	0.91	40(0.71)	100(0.86)
Activity recognition using wearable physiological measurements	0.81	0.93	0.96	100(0.68)	350(0.85)
Internet Advertisements	0.87	0.95	0.97	300(0.83)	1000(0.87)

Table 5.2: Applying Features selection

5.10.3 Experiment II:Dimension Reduction

In this experiment, we will use a dataset with a large number of features and compare the accuracy achieved by all the features with that of reduced dimensions that achieve a similarly high accuracy, as well as a smaller feature space. In Table 5.3, we use the AUC metric to evaluate accuracy. We first examine the accuracy without applying any method, followed by the application of oversampling techniques. In the next column, we apply our oversampling method to the dataset, followed by applying a dimension reduction technique. Finally, we select a number of dimensions or components for comparison with the accuracy achieved using all features, shown in the last two columns.

In this experiment, the aim is to reduce the dimensionality of the data using Principal Component Analysis (PCA). The goal of PCA is to find the most significant

features that explain the majority of the variance in the data. To achieve this, we remove the components that contribute very little to the overall variance of the data. This step not only helps us save time and memory, but it also ensures that we are working with the most informative features in the data.

Moreover, after removing the low-variance components, we can create a new set of feature space options to select from. By doing so, we can avoid using features that overlap and are redundant, which can result in overfitting and decrease the model's performance. Therefore, this technique provides us with a more refined and effective set of features for subsequent analyses.

Dataset Name	Baseline(AUC)	SMOTE (AUC)	DCADASYN(AUC)	#Component(AUC)	#Component(AUC)
Optical Recognition of Handwritten Digits	0.83	0.93	0.96	10(0.76)	22(0.94)
Musk	0.78	0.90	0.94	70(0.88)	100(0.92)
Semeion Handwritten	0.79	0.88	0.91	90(0.86)	200(0.89)
Activity recognition using wearable physiological measurements	0.81	0.93	0.96	150(0.83)	300(0.90)
Internet Advertisements	0.87	0.95	0.97	500(0.90)	800(0.94)

Table 5.3: Applying Dimensionally reduction

5.11 Conclusion

In conclusion, feature selection and dimensionality reduction are useful for datasets with overlapping features, providing the option to avoid them. Overlapping features can lead to redundancy, and multicollinearity, and decrease the model's performance. By selecting the most informative features and reducing the dimensionality of the data, we can address these issues and improve the overall quality of the dataset.

Feature selection involves selecting a subset of relevant features from the dataset

while discarding irrelevant ones. This can be done through various techniques such as mutual information, correlation-based feature selection, and wrapper methods. By selecting the most relevant features, we can reduce the amount of noise in the data and improve the model's accuracy. This approach can also improve the speed and efficiency of the model, as it reduces the number of features that need to be processed.

Dimensionality reduction techniques, such as Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE), can help eliminate redundant and correlated features, leading to a more efficient and accurate model. These techniques can identify the underlying patterns in the data and reduce the number of dimensions while retaining the most informative features.

Dimensionality reduction and feature selection techniques can significantly improve the analysis of large-scale datasets. These techniques help in reducing the computation time required for the analysis, improving the accuracy of the analysis, and eliminating the noise and redundancy in the data. By reducing the dimensionality of the dataset and selecting only the most relevant features in the dataset, the challenges associated with large-scale datasets, such as the curse of dimensionality and overfitting, can be mitigated.

Overall, feature selection and dimensionality reduction techniques are useful tools for addressing overlapping features in a dataset. By using these techniques, we can improve Figure 5.1 and choose the best accuracy and efficiency of the model, obtain more interpretable and reliable results, and ultimately, make better-informed decisions.

Chapter 6

Future Work

In this thesis, we presented a method for detecting overlapping data points in a dataset. This method is based on a clustering algorithm that groups similar data points together. We also presented an algorithm for oversampling the minority class in an imbalanced dataset. This algorithm generates synthetic samples that belong to the minority class, and adds them to the dataset in order to balance the distribution of classes. Finally, we presented a method for selecting features and reducing the dimensionality of a high-dimensional dataset. This method uses a combination of feature selection and dimensionality reduction techniques, such as principal component analysis, to select a subset of informative and relevant features that can be used to represent the data in a lower-dimensional space. Overall, our methods provide a systematic and effective way to address the challenges posed by overlapping and imbalanced datasets, and can improve the performance and reliability of data analysis and modeling techniques. In this chapter, we've discussed several avenues for future work that we're eager to explore and develop. These directions build on the ideas and techniques presented in the previous chapters, and offer promising opportunities

for advancing the field and making new contributions.

6.1 Semi-supervised learning

Semi-supervised learning is a machine learning technique that uses both labeled and unlabeled data to enhance the performance of predictive models.In a semi-supervised learning scenario, the dataset typically contains a large number of unlabeled samples and a smaller number of labeled samples, which are used to train the model. By incorporating the additional information present in the unlabeled data, semi-supervised learning algorithms can improve the accuracy and generalization ability of the model, compared to using only labeled data.

One common challenge in semi-supervised learning is the presence of an imbalanced dataset, where the number of samples belonging to one class or category is significantly larger than the number of samples belonging to other classes (Huynh *et al.*, 2022). This can cause the model to be biased towards the majority class, and to have difficulty accurately predicting the minority class. To address this challenge, semi-supervised learning algorithms often incorporate techniques for balancing the distribution of classes in the dataset, such as oversampling or undersampling. Training a model with imbalanced datasets can lead to poor performance. Class imbalanced learning (CIL) is a way to address this issue by using various methods at the level of data, algorithm, and their combinations (Krawczyk, 2016; Chapelle *et al.*, 2009). However, as far as we know, existing research on CIL has only used labeled datasets for training and has not explored the potential benefits of using unlabeled data.

The main focus of (Li *et al.*, 2011) is the use of the co-training technique in semi-supervised learning for imbalanced sentiment classification. This technique involves generating different views from random feature subspaces. But not dealing with overlapping features.

Our technique incorporates the use of clustering and density estimation to leverage the unlabeled data, allowing the model to benefit from the information contained in these data points. This can help improve the performance of the model on imbalanced and has some overlapping feature datasets.

6.2 Unsupervised learning

Unsupervised learning is a machine learning approach that enables models to learn from data without requiring labeled training examples. The goal of unsupervised learning is to discover hidden patterns or underlying structures in the data. This can be useful for a variety of applications, including clustering data points into groups based on their similarities, and identifying anomalies in the data.

One challenge that can arise when working with unsupervised learning is dealing with imbalanced datasets (Yan *et al.*, 2020; Krawczyk, 2016). This occurs when the dataset has a disproportionate number of examples belonging to one class compared to the other classes. For example, if a dataset is meant to classify animals into dogs and cats, but it only contains 10 cat examples and 1000 dog examples, then the dataset is imbalanced. This can be problematic for unsupervised learning algorithms because they may end up overfitting to the majority class, leading to poor performance on the minority class. To address this issue, one approach is to undersample the majority class or oversample the minority class to balance the dataset. Another approach is

to use a weighted loss function to give more importance to the minority class during training.

In (Nickerson *et al.*, 2001), the authors first use unsupervised clustering to detect imbalances within the positive and negative classes of the dataset. They use this information to avoid increasing the differences in the relative densities of the subcomponents of each class by equalizing the number of members in each subcomponent. The unsupervised clustering technique, Principal Direction Divisive Partitioning (PDDP), is used to determine the internal characteristics of each class. In their experiments, the authors use their knowledge of the subcomponents in each class to guide PDDP to find the number of clusters within each class. The clusters are then resampled so that the discovered clusters across both classes each have the same number of examples. A decision-tree based classifier, is trained and used to classify new examples. The results of the guided resampling technique are compared to the results obtained without resampling and with a blind resampling strategy, which resamples at random without considering within-class imbalances. Imbalanced and overlapping features are common in datasets. Imbalanced features are those that have an unequal distribution of class labels. For example, if one class has many more samples than other classes, it is considered to be an imbalanced feature. Overlapping features occur when two or more features have similar datapoint of class labels. To address these issues, we have introduced a new method to deal with imbalanced and overlapping features. These include data preprocessing, resampling, and feature selection methods.

In summary, unsupervised learning is a useful tool for discovering patterns in data without the need for labeled examples. However, when dealing with imbalanced datasets, it is important to consider the potential impact on the performance of the

model and take steps to address the imbalance. This can include undersampling or oversampling the data, or using a weighted loss function during training.

6.3 Undersampling the dataset

Our data balancing approach focuses only on oversampling.Oversampling and undersampling (Sáez *et al.*, 2016; García and Herrera, 2009) are techniques used to address imbalanced datasets in machine learning. Imbalanced datasets occur when one class in the dataset has a disproportionate number of examples compared to the other classes. This can be problematic for some machine learning algorithms, which may end up overfitting to the majority class and not perform well on the minority class.

Oversampling involves artificially generating additional examples of the minority class in order to balance the dataset. This can be done by sampling with replacement from the minority class to create new, synthetic examples, or by using techniques such as SMOTE (Synthetic Minority Oversampling Technique) to generate new examples that are similar to existing ones in the minority class. By increasing the number of examples in the minority class, oversampling can help improve the performance of the machine learning model on the minority class.

Undersampling, on the other hand, involves reducing the number of examples in the majority class to balance the dataset (Dai *et al.*, 2023). This can be done by simply removing some examples from the majority class at random, or by using techniques such as Tomek's links to identify and remove examples from the majority class that are surrounded by examples from the minority class. By reducing the number of examples in the majority class, undersampling can help improve the performance of the machine learning model on the minority class.

6.4 Different types of data and Extreme class imbalance

In future work, we aim to extend our algorithms to address classification problems with a severe imbalance between the majority and minority classes. In (Ribeiro and Moniz, 2020; Leevy *et al.*, 2018) published studies on addressing the class imbalance in big data, focusing on high-class imbalance with a majority-to-minority class ratio between 100:1 and 10,000:1. This affects the accuracy of the prediction of the classes. One of the most effective techniques for dealing with severely imbalanced distributions is oversampling (Japkowicz and Stephen, 2002). In oversampling, the minority class is replicated multiple times in the dataset so that its representation is increased. This helps to create a more balanced class distribution and improves the performance of the machine learning model. Oversampling can be done manually or automated using various algorithms.Another technique for dealing with high-class imbalance is to use cost-sensitive learning algorithms (Elkan, 2001). These algorithms assign different costs to different classes, so that the model is penalized more for misclassifying minority classes than majority classes. This helps to ensure that the model does not overlook minority classes in the dataset. Furthermore, various feature selection methods such as recursive feature elimination (Guyon *et al.*, 2002) can be used.

Chapter 7

Conclusions

In conclusion, our method for detecting overlaps in datasets and improving imbalanced data has achieved a high level of accuracy. Our approach, which combines advanced algorithms to create a balance the minority class with customizable analysis and feature selection, has consistently outperformed other methods in terms of detecting overlaps and improving the balance of imbalanced datasets.

One key advantage of our method is its use of specialized algorithms that quickly and accurately identify overlaps. By using our approach to balance the minority class, we were able to achieve the highest accuracy in large datasets. These algorithms are able to provide users with clear, actionable insights into the structure and quality of their data, allowing them to make informed decisions about how to address any issues that are discovered.

To evaluate the effectiveness of our method for improving the accuracy of predictions, we conducted a series of experiments on a range of different datasets. These experiments allowed us to compare the performance of our method against other existing approaches, providing a comprehensive assessment of its effectiveness.

In each experiment, we used a standardized set of metrics to evaluate the accuracy of the predictions made by each method. This included measures such as precision, recall, and F1 score, which are commonly used to assess the performance of classification algorithms.

The results of our experiments showed that our method consistently outperformed other approaches in terms of prediction accuracy. In particular, our method was able to achieve higher levels of precision and recall, and had a higher overall F1 score. This demonstrated the effectiveness of our approach in improving the accuracy of predictions.

Our experiments, showed that our method is an effective way to increase the accuracy of predictions in different scenarios. It combines advanced algorithms with customizable analysis and feature selection to provide accurate and useful insights that can help users improve the value and quality of their data. The feature selection component of our method reduces the dimensions of the dataset and identifies the most important features, allowing users to focus on the most relevant and informative aspects of their data and improving the accuracy of the results. In addition, our method has been successful in detecting overlaps and improving imbalanced datasets, providing users with a precise and effective solution to these common issues. Overall, our approach delivers accurate and actionable insights to help users enhance the value and quality of their data.

Bibliography

Abdi, H. and Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, **2**(4), 433–459.

Acuna, E. and Rodriguez, C. (2004). The treatment of missing values and its effect on classifier accuracy. In *Classification, clustering, and data mining applications*, pages 639–647. Springer.

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., and Herrera, F. (2011). Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, **17**.

Almutairi, W. and Janicki, R. (2020). On relationships between imbalance and overlapping of datasets. In *CATA*, pages 141–150.

Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, **46**(3), 175–185.

Barandela, R., Sánchez, J. S., Garcıa, V., and Rangel, E. (2003). Strategies for learning in class imbalance problems. *Pattern Recognition*, **36**(3), 849–851.

Batista, G. E., Bazzan, A. L., and Monard, M. C. (2003). Balancing training data for automated annotation of keywords: a case study. In *WOB*, pages 10–18.

Batista, G. E., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, **6**(1), 20–29.

Batista, G. E., Prati, R. C., and Monard, M. C. (2005). Balancing strategies and class overlapping. In *International Symposium on Intelligent Data Analysis*, pages 24–35. Springer.

Bell, A. J. and Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, **7**(6), 1129–1159.

Bellman, R. (1961). Adaptive control processes: a guided tour princeton university press. *Princeton, New Jersey, USA*, page 96.

Bishop, C. M. (2006). Pattern recognition. *Machine learning*, **128**(9).

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, **30**(7), 1145–1159.

Browne, M. W. (2000). Cross-validation methods. *Journal of mathematical psychology*, **44**(1), 108–132.

Caruana, R. and Freitag, D. (1994). Greedy attribute selection. In *Machine Learning Proceedings 1994*, pages 28–36. Elsevier.

Cayton, L. (2005). Algorithms for manifold learning. *Univ. of California at San Diego Tech. Rep*, **12**(1-17), 1.

Chan, L. S. and Dunn, O. J. (1972). The treatment of missing values in discriminant analysisi. the sampling experiment. *Journal of the American Statistical Association*, **67**(338), 473–477.

Chapelle, O., Scholkopf, B., and Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, **20**(3), 542–542.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, **16**, 321–357.

Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Editorial: Special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, **6**(1), 1–6.

Dabare, R., Wong, K. W., Shiratuddin, M. F., and Koutsakis, P. (2019). Fuzzy deep neural network for classification of overlapped data. In *International Conference on Neural Information Processing*, pages 633–643. Springer.

Dai, Q., Liu, J.-w., and Shi, Y.-h. (2023). Class-overlap undersampling based on schur decomposition for class-imbalance problems. *Expert Systems with Applications*, **221**, 119735.

Dash, M., Choi, K., Scheuermann, P., and Liu, H. (2002). Feature selection for clustering-a filter solution. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 115–122. IEEE.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, **39**(1), 1–22.

Denil, M. and Trappenberg, T. (2010). Overlap versus imbalance. In *Canadian Conference on Artificial Intelligence*, pages 220–231. Springer.

Devi, D., Biswas, S. K., and Purkayastha, B. (2019). Learning in presence of class imbalance and class overlapping by using one-class svm and undersampling technique. *Connection Science*, **31**(2), 105–142.

Dokmanic, I., Parhizkar, R., Ranieri, J., and Vetterli, M. (2015). Euclidean distance matrices: essential theory, algorithms, and applications. *IEEE Signal Processing Magazine*, **32**(6), 12–30.

Drummond, C. and Holte, R. C. (2000). Exploiting the cost (in) sensitivity of decision tree splitting criteria. In *ICML*, volume 1.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Egan, J. P. (1975). Signal detection theory and {ROC} analysis.

Elhassan, T., Aljurf, M., Al-Mohanna, F., and Shoukri, M. (2016). Classification of imbalance data using tomek link (t-link) combined with random under-sampling (rus) as a data reduction method. *Journal of Informatics and Data Mining*, **1**(2).

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

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., *et al.* (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

FernáNdez, A., LóPez, V., Galar, M., Del Jesus, M. J., and 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., Garcia, S., Herrera, F., and Chawla, N. V. (2018). Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, **61**, 863–905.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, **7**(2), 179–188.

Fix, E. and Hodges, J. L. (1989). Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, **57**(3), 238–247.

Fu, M., Tian, Y., and Wu, F. (2015). Step-wise support vector machines for classification of overlapping samples. *Neurocomputing*, **155**, 159–166.

García, S. and Herrera, F. (2009). Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary computation*, **17**(3), 275–306.

García, V., Alejo, R., Sánchez, J., Sotoca, J., and Mollineda, R. (2006). Combined effects of class imbalance and class overlap on instance-based classification. *Intelligent Data Engineering and Automated Learning–IDEAL 2006*, pages 371–378.

Gewers, F. L., Ferreira, G. R., Arruda, H. F. D., Silva, F. N., Comin, C. H., Amancio, D. R., and Costa, L. d. F. (2021). Principal component analysis: A natural approach to data exploration. *ACM Computing Surveys (CSUR)*, **54**(4), 1–34.

Goh, D. and Foo, S. (2007). *Social Information Retrieval Systems: Emerging Technologies and Applications for Searching the Web Effectively: Emerging Technologies and Applications for Searching the Web Effectively*. IGI Global.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix computations*. JHU press.

Graham, C. and Tokieda, T. (2020). An entropy proof of the arithmetic mean–geometric mean inequality. *The American Mathematical Monthly*, **127**(6), 545–546.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, **3**(Mar), 1157–1182.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, **46**(1), 389–422.

Hall, M. A. (2000). Correlation-based feature selection of discrete and numeric class machine learning.

Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing*, pages 878–887. Springer.

Har-Peled, S., Roth, D., and Zimak, D. (2003). Constraint classification for multi-class classification and ranking. *Advances in neural information processing systems*, pages 809–816.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). The elements of statistical learning. springer series in statistics. In ∴ Springer.

He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, **21**(9), 1263–1284.

He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE.

Hinton, G. E., Sejnowski, T. J., *et al.* (1999). *Unsupervised learning: foundations of neural computation*. MIT press.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

Ho, T. K. and Basu, M. (2002). Complexity measures of supervised classification problems. *IEEE transactions on pattern analysis and machine intelligence*, **24**(3), 289–300.

Huang, J. and Ling, C. X. (2005). Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering*, **17**(3), 299–310.

Huynh, T., Nibali, A., and He, Z. (2022). Semi-supervised learning for medical image classification using imbalanced training data. *Computer methods and programs in biomedicine*, **216**, 106628.

Janicki, R. and Soudkhah, M. H. (2015). On classification with pairwise comparisons, support vector machines and feature domain overlapping. *The Computer Journal*, **58**(3), 416–431.

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

Japkowicz, N. and Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent data analysis*, **6**(5), 429–449.

Jasner, Y., Belogolovski, A., Ben-Itzhak, M., Koren, O., and Louzoun, Y. (2021). Microbiome preprocessing machine learning pipeline. *Frontiers in Immunology*, **12**, 677870.

John, G. H., Kohavi, R., and Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Machine learning proceedings 1994*, pages 121–129. Elsevier.

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, **349**(6245), 255–260.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, **4**, 237–285.

Kaur, H., Pannu, H. S., and 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.

Kent, A., Berry, M. M., Luehrs Jr, F. U., and Perry, J. W. (1955). Machine literature searching viii. operational criteria for designing information retrieval systems. *American documentation*, **6**(2), 93–101.

Kira, K. and Rendell, L. A. (1992). A practical approach to feature selection. In *Machine learning proceedings 1992*, pages 249–256. Elsevier.

Koç, H. Ü. (1995). *Classification with overlapping feature intervals*. Ph.D. thesis, Bilkent University.

Koyejo, O., Natarajan, N., Ravikumar, P., and Dhillon, I. S. (2014). Consistent binary classification with generalized performance metrics. In *NIPS*, volume 27, pages 2744–2752. Citeseer.

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

Kubat, M., Matwin, S., *et al.* (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *ICML*, volume 97, pages 179–186. Nashville, USA.

Kubat, M., Holte, R. C., and Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, **30**(2-3), 195–215.

Lee, C. and Lee, G. G. (2006). Information gain and divergence-based feature selection for machine learning-based text categorization. *Information processing & management*, **42**(1), 155–165.

Lee, S. S. (2000). Noisy replication in skewed binary classification. *Computational statistics & data analysis*, **34**(2), 165–191.

Leevy, J. L., Khoshgoftaar, T. M., Bauder, R. A., and Seliya, N. (2018). A survey on addressing high-class imbalance in big data. *Journal of Big Data*, **5**(1), 1–30.

Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc.

Li, S., Wang, Z., Zhou, G., and Lee, S. Y. M. (2011). Semi-supervised learning for imbalanced sentiment classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Lin, M., Tang, K., and Yao, X. (2013). Dynamic sampling approach to training neural networks for multiclass imbalance classification. *IEEE Transactions on Neural Networks and Learning Systems*, **24**(4), 647–660.

Ling, C. X. and Li, C. (1998). Data mining for direct marketing: Problems and solutions. In *KDD*, volume 98, pages 73–79.

Liu, C.-L. (2008). Partial discriminative training for classification of overlapping classes in document analysis. *International Journal of Document Analysis and Recognition (IJDAR)*, **11**(2), 53.

Liu, H. and Motoda, H. (2012). *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media.

Liu, H., Setiono, R., *et al.* (1996). A probabilistic approach to feature selection-a filter solution. In *ICML*, volume 96, pages 319–327. Citeseer.

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

Manning, C. and Schutze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Meyer, D., Leisch, F., and Hornik, K. (2003). The support vector machine under test. *Neurocomputing*, **55**(1-2), 169–186.

Molinaro, A. M., Simon, R., and Pfeiffer, R. M. (2005). Prediction error estimation: a comparison of resampling methods. *Bioinformatics*, **21**(15), 3301–3307.

Müller, H. and Freytag, J.-C. (2005). *Problems, methods, and challenges in comprehensive data cleansing*. Professoren des Inst. Für Informatik.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

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

Nickerson, A., Japkowicz, N., and Milios, E. E. (2001). Using unsupervised learning to guide resampling in imbalanced data sets. In *International Workshop on Artificial Intelligence and Statistics*, pages 224–228. PMLR.

Obaid, H. S., Dheyab, S. A., and Sabry, S. S. (2019). The impact of data pre-processing techniques and dimensionality reduction on the accuracy of machine learning. In *2019 9th annual information technology, electromechanical engineering and microelectronics conference (iemecon)*, pages 279–283. IEEE.

Popescu, A., Broekens, J., and van Someren, M. (2014). GAMYGDALA: An emotion engine for games. *Affective Computing, IEEE Transactions on*, **5**(1), 32–44.

Powers, D. M. (2020). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*.

Prati, R. C., Batista, G. E., and Monard, M. C. (2004). Class imbalances versus class overlapping: an analysis of a learning system behavior. In *Mexican international conference on artificial intelligence*, pages 312–321. Springer.

Priyam, A., Abhijeeta, G., Rathee, A., and Srivastava, S. (2013). Comparative analysis of decision tree classification algorithms. *International Journal of current engineering and technology*, **3**(2), 334–337.

Provost, F. and Fawcett, T. (2001). Robust classification for imprecise environments. *Machine learning*, **42**(3), 203–231.

Qing, Z., Zeng, Q., Wang, H., Liu, Y., Xiong, T., and Zhang, S. (2022). Adasyn-lof algorithm for imbalanced tornado samples. *Atmosphere*, **13**(4), 544.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, **1**(1), 81–106.

Raghavan, V., Bollmann, P., and Jung, G. S. (1989). A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems (TOIS)*, **7**(3), 205–229.

Rahm, E. and Do, H. H. (2000). Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, **23**(4), 3–13.

Rahmah, N. and Sitanggang, I. S. (2016). Determination of optimal epsilon (eps) value on dbscan algorithm to clustering data on peatland hotspots in sumatra. In *IOP conference series: earth and environmental science*, volume 31, page 012012. IOP Publishing.

Refaeilzadeh, P., Tang, L., and Liu, H. (2009). Cross-validation. *Encyclopedia of database systems*, **5**, 532–538.

Ribeiro, R. P. and Moniz, N. (2020). Imbalanced regression and extreme value prediction. *Machine Learning*, **109**, 1803–1835.

Ron, K. and George, H. J. (1997). Wrappers for feature subset selection. *Artificial intelligence*, **97**(1-2), 273–324.

Russell, S. and Norvig, P. (2002). Artificial intelligence: a modern approach.

Sáez, J. A., Krawczyk, B., and Woźniak, M. (2016). Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. *Pattern Recognition*, **57**, 164–178.

Sáez, J. A., Galar, M., and Krawczyk, B. (2019). Addressing the overlapping data problem in classification using the one-vs-one decomposition strategy. *IEEE Access*, **7**, 83396–83411.

Sander, J., Ester, M., Kriegel, H.-P., and Xu, X. (1998). Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data mining and knowledge discovery*, **2**(2), 169–194.

Sato, Y., Nakajima, S., Shiraga, N., Atsumi, H., Yoshida, S., Koller, T., Gerig, G., and Kikinis, R. (1998). Three-dimensional multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. *Medical image analysis*, **2**(2), 143–168.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., *et al.* (2017). Mastering the game of go without human knowledge. *nature*, **550**(7676), 354–359.

Sit, W. Y., Mak, L. O., and Ng, G. W. (2009). Managing category proliferation in fuzzy artmap caused by overlapping classes. *IEEE transactions on neural networks*, **20**(8), 1244–1253.

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

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Swets, J. A. (1988). Measuring the accuracy of diagnostic systems. *Science*, **240**(4857), 1285–1293.

Szmidt, E. and Kukier, M. (2006). Classification of imbalanced and overlapping classes using intuitionistic fuzzy sets. In *2006 3rd International IEEE Conference Intelligent Systems*, pages 722–727. IEEE.

Tomek, I. (1976). Two modifications of cnn. *IEEE Trans. Systems, Man and Cybernetics*, **6**, 769–772.

Trappenberg, T. P. and Back, A. D. (2000). A classification scheme for applications with ambiguous data. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 6, pages 296–301. IEEE.

Turney, P. D. (2002). The identification of context-sensitive features: A formal definition of context for concept learning. *arXiv preprint cs/0212038*.

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, **9**(11).

Veropoulos, K., Campbell, C., Cristianini, N., *et al.* (1999). Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on AI*, volume 55, page 60. Stockholm.

Visa, S. and Ralescu, A. (2003). Learning imbalanced and overlapping classes using fuzzy sets. In *Proceedings of the ICML*, volume 3.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, **17**(4), 395–416.

Wang, B. and Japkowicz, N. (2004). Imbalanced data set learning with synthetic samples. In *Proc. IRIS Machine Learning Workshop*, volume 19.

Wasikowski, M. and Chen, X.-w. (2009). Combating the small sample class imbalance problem using feature selection. *IEEE Transactions on knowledge and data engineering*, **22**(10), 1388–1400.

Witten, I. H., Frank, E., Hall, M. A., Pal, C. J., and DATA, M. (2005). Practical machine learning tools and techniques. In *Data Mining*, volume 2.

Wold, S., Esbensen, K., and Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, **2**(1-3), 37–52.

Wu, G. and Chang, E. Y. (2003). Class-boundary alignment for imbalanced dataset learning. In *ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC*, pages 49–56. Citeseer.

Wu, S. X., Wai, H.-T., Li, L., and Scaglione, A. (2018). A review of distributed algorithms for principal component analysis. *Proceedings of the IEEE*, **106**(8), 1321–1340.

Xie, Y., Li, D., Zhang, D., and Shuang, H. (2017). An improved multi-label relief feature selection algorithm for unbalanced datasets. In *International Conference on Intelligent and Interactive Systems and Applications*, pages 141–151. Springer.

Xiong, H., Pandey, G., Steinbach, M., and Kumar, V. (2006). Enhancing data analysis with noise removal. *IEEE Transactions on Knowledge and Data Engineering*, **18**(3), 304–319.

Xiong, H., Wu, J., and Liu, L. (2010). Classification with class overlapping: a systematic study. In *The 2010 International Conference on E-Business Intelligence*, pages 491–497.

Xiong, H., Li, M., Jiang, T., and Zhao, S. (2013). Classification algorithm based on nb for class overlapping problem. *Appl. Math*, **7**(2L), 409–415.

Yan, K., Huang, J., Shen, W., and Ji, Z. (2020). Unsupervised learning for fault detection and diagnosis of air handling units. *Energy and Buildings*, **210**, 109689.

Yang, Q. and Wu, X. (2006). 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, **5**(04), 597–604.

Yaohua, T. and Jinghuai, G. (2007). Improved classification for problem involving overlapping patterns. *IEICE TRANSACTIONS on Information and Systems*, **90**(11), 1787–1795.

Zhu, X. J. (2005). Semi-supervised learning literature survey, technical report tr 1530, university of wisconsin at madison, usa.