

**A Comprehensive approach to the detection of Liver disease using
Machine Learning Techniques with comparison of different
Oversampling Techniques on Imbalanced Liver Disease Dataset**

by

Mir Samsul Arefin – 180021209

Chowdhury Sadeeya Naimah – 180021219

Rayeed Rahman – 180021238

A Thesis Submitted to the Academic Faculty in Partial Fulfillment of the Requirements
for the Degree of

**BACHELOR OF SCIENCE IN ELECTRICAL AND ELECTRONIC
ENGINEERING**



Department of Electrical and Electronic Engineering

Islamic University of Technology (IUT)

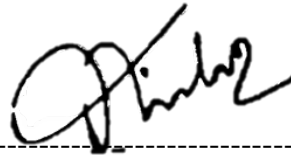
The Organization of Islamic Cooperation (OIC)
Board Bazar, Gazipur-1704, Bangladesh

May 2023

CERTIFICATE OF APPROVAL

The thesis titled “A Comprehensive approach to the detection of Liver disease using Machine Learning Techniques with comparison of different Oversampling Techniques on Imbalanced Liver Disease Dataset” submitted by Mir Samsul Arefin (180021209), Chowdhury Sadeeya Naimah (180021219), and Rayeed Rahman (180021238) has been found as satisfactory and accepted as partial fulfillment of the requirement for the degree of Bachelor of Science in Electrical and Electronic Engineering on May 2023.

Approved by:



(Signature of the Supervisor)

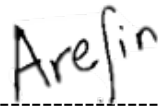
Mirza Muntasir Nishat

Assistant Professor

Department of Electrical and Electronic Engineering
Islamic University of Technology

DECLARATION OF CANDIDATES

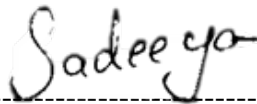
It is hereby declared that this thesis or any part of it has not been submitted elsewhere for award of any degree or diploma.



(Signature of the Candidate)

Mir Samsul Arefin

Student ID: 180021209



(Signature of the Candidate)

Chowdhury Sadeeya Naimah

Student ID: 180021219



(Signature of the Candidate)

Rayeed Rahman

Student ID: 180021238

DEDICATION

We would like to dedicate this thesis to our family members and everyone who has given us unwearied support throughout the entirety of our existence and every situation of our life. They have always been a source of motivation for us. They pushed us ahead and showed us how to make the correct decisions. They never fail to inspire us to work hard and move forward to overcome life's difficulties. They have provided us with the protection, wisdom, and fortitude we need to face difficult situations.

ACKNOWLEDGEMENTS

First, we would want to express our heartfelt gratitude to Almighty Allah, our Creator, for creating and instilling in us the intellect to educate ourselves with worldly knowledge and, therefore, complete our thesis research. **Prof. Dr. Md. Ashraful Hoque**, Professor, Department of EEE, IUT, is our respected supervisor. We owe him a debt of gratitude for his continuous advice, care, and support in our pursuit of a career in electrical and electronic engineering. He has always encouraged us to learn new things and broaden our horizons to keep our minds sharp. We would not be exploring the power electronics area if it were not for his motivation. He has encouraged us to learn the fundamentals of the specific field and shown us how to proceed in the right direction

Mr. Mirza Muntasir Nishat, Assistant Professor, Department of EEE, IUT, is our co-supervisor, and we are grateful for his constant mentoring and genuine efforts in our thesis research. He devoted his considerable time guiding and motivating us to finish the work. We conducted a study and collected numerous informative analyses under his leadership to get positive results. Moreover, he gave us the most efficient technique to better understand our research. Without his help, we would become lost and unable to pick the best course of action.

Mr. Fahim Faisal, Assistant Professor, Department of EEE, IUT, served as our co-supervisor and mentored us throughout the research process. He has always been encouraging and motivating us to complete our work correctly. In addition, he has motivated us to study the primary goal of our project, which has given us greater confidence in our ability to build skills while working on the thesis.

Finally, we owe a debt of gratitude to our family for encouraging and assisting us in overcoming life's challenges, as well as enchanting us with their wonderful words. Last but not least, we would like to express our gratitude to our friends for their unconditional support and for keeping our spirits upbeat throughout this journey.

ABSTRACT

The liver is one of the most important organs in the body. It is responsible for controlling the chemical balance of the bloodstream as well as the removal of waste products among other vital functions. Liver disease is important to be diagnosed early on as symptoms do not begin to show until most of the liver is already damaged. Machine learning could be a crucial tool in the prediction of liver disease in patients which could lead to early diagnosis and also early treatment. In this study a dataset with 583 instances has been pre-processed and the imbalance had been handled in 5 separate ways, namely, Synthetic Minority Oversampling Technique (SMOTE), Adaptive Synthetic (ADASYN), Synthetic Minority Oversampling Technique and Conformal Clustering (CC), Synthetic Minority Oversampling Technique and Tomeklinks and Synthetic Minority Oversampling Technique and edited nearest neighbor (SMOTE+ENN). Then various machine learning algorithms like Decision Tree Classifier, Logistic Regression, Gaussian Naïve Bayes, Random Forest Classifier, K-Nearest Neighbors, and Support Vector Machine algorithms etc has been used. The experiment gave the best result when SMOTE+ENN was used as the imbalance handling technique with an accuracy of 98.37%. This accuracy was found using the support vector machine (SVM) approach. Therefore, this study shows the comparative analysis of the different imbalance handling techniques and the one which performs the best among each of these. It presents SMOTE+ENN as the best in case of this specific dataset.

Table of Contents

CERTIFICATE OF APPROVAL	i
DECLARATION OF CANDIDATES	ii
DEDICATION	iii
ACKNOWLEDGEMENTS	iv
ABSTRACT.....	v
LIST OF TABLES.....	viii
LIST OF FIGURES.....	x
LIST OF ACRONYMS.....	xi
CHAPTER 1	1
INTRODUCTION.....	1
1.1 Introduction	1
1.2 Problem Statement.....	2
1.3 Thesis Organization.....	3
CHAPTER 2	4
Background Study	4
CHAPTER 3	6
Methodology.....	6
3.1 Data Collection.....	6
3.1.1 Data Source:.....	6
3.1.2 Data Structure:.....	7
3.1.3 Data Features:.....	8
3.1.4 Data Preprocessing:	9
3.1.4.1 Cleaning Duplicate Data:.....	9
3.1.4.2 Handling Missing Data:	9
3.1.4.3 Data Encoding:	10
3.1.4.4 Transforming Skew Data:.....	10
3.2 Feature Selection:	14
3.3 Scaling:	15
3.3.1 Min-Max Scaling:.....	16
3.3.2 Standard Scaling:.....	18

3.4 Handling Imbalanced Data:.....	19
3.4.1 SMOTE:.....	21
3.4.2 ADASYN:	23
3.4.3: SMOTE + TOMEKLINKS.....	25
3.4.4 SMOTE+Cluster Centroids.....	30
3.4.5 SMOTE+ENN (Edited Nearest Network)	33
CHAPTER 4	37
Results.....	37
4.1 Before Imbalance Handling.....	41
4.2 After Imbalance Handling	42
4.2.1 SMOTE:.....	42
4.2.2 ADASYN:	43
4.2.3 SMOTE + CC.....	44
4.2.4 SMOTE + TOMEKLINKS.....	46
4.2.5 SMOTE + ENN.....	47
4.3 Hyperparameter Tuning.....	48
4.3.1 SMOTE:.....	49
4.3.2 ADASYN	50
4.3.3 SMOTE+CC:	51
4.3.4 SMOTE+TOMEKLINKS.....	52
4.3.5 SMOTE + ENN.....	53
CHAPTER 5	54
Discussion.....	54
CHAPTER 6	60
Limitations	60
CHAPTER 7	61
CONCLUSION AND FUTURE WORKS	61
References	62

LIST OF TABLES

No.	Title	Page No.
3.1	Statistical information of the attributes	
4.1	Performance of Machine learning and Deep learning models before Imbalance handling	
4.2	Results after SMOTE applied to dataset	
4.3	Result after ADASYN applied to dataset	
4.4	Result after SMOTE + CC applied to dataset	
4.5	Result after SMOTE + TOMEKLINKS applied to dataset	
4.6	Result after SMOTE + ENN applied to dataset	
4.7	Performance Comparison of Machine Learning Algorithms with SMOTE and GridSearchCV	
4.8	Performance Comparison of Machine Learning Algorithms with SMOTE and RandomizedsearchCV	
4.9	Performance Comparison of Machine Learning Algorithms with ADASYN and GridSearchCV	
4.10	Performance Comparison of Machine Learning Algorithms with ADASYN and RandomizedSearchCV	
4.11	Performance Comparison of Machine Learning Algorithms with SMOTE+CC and GridSearchCV	
4.12	Performance Comparison of Machine Learning Algorithms with SMOTE+CC and RandomizedSearchCV	
4.13	Performance Comparison of Machine Learning Algorithms with SMOTE + TOMEKLINKS and GridSearchCV	
4.14	Performance Comparison of Machine Learning Algorithms with SMOTE + TOMEKLINKS and RandomizedSearchCV	
4.15	Performance Comparison of Machine Learning Algorithms with SMOTE+ENN and GridSearchCV	

4.16	Performance Comparison of Machine Learning Algorithms with SMOTE+ENN and RandomizedSearchCV	
------	---	--

LIST OF FIGURES

No.	Title	Page No.
3.1	Histogram for Age with Disease	
3.2	Observing skewness before and after transformation	
3.3	Correlation heatmap of attributes	
3.4	Before and after Min-Max Scaling	
3.5	Before and after Standard Scaling	
3.6	Example of balanced and imbalanced data	
3.7	Before, during and after applying SMOTE to a dataset	
3.8	Before and after ADASYN applied to a dataset	
3.9	a) Original Dataset b) Finding Tomeklinks c) Resampled Dataset	
3.10	Original dataset (left) and Dataset after smote+Tomeklinks (right) has been applied	
3.11	Example of centroid locations before (left) and after (right) one k-means iteration. K-means moves the centroids towards the actual cluster centers	
3.12	Samples simulation plot after the SMOTE-ENN sampled	
4.1	Structure of a confusion matrix	
4.2	Comparative figures of all the performance parameter (accuracy, precision, recall and f1 score) for each machine learning algorithm with each imbalance handling technique applied	

LIST OF ACRONYMS

Abbreviated Form	Description
SMOTE	Synthetic Minority Over-sampling Technique
ADASYN	Adaptive Synthetic
CC	Cluster Centroid
ENN	Edited Nearest Neighbor
ML	Machine Learning
DL	Deep Learning
KNN	k-nearest neighbors algorithm
ANN	Artificial Neural Network
SMOTE + NC	Synthetic Minority Over-sampling Technique for Nominal and Continuous
XGBoost	Extreme Gradient Boosting
AdaBoost	Adaptive Boosting
RF	Random Forest
SVM	Support Vector Machines
ILDV	Indian Liver Patient Dataset
LSTM	Long Short-Term Memory Networks
CV	Cross Validation

CHAPTER 1

INTRODUCTION

1.1 Introduction

The advancement of science and technology naturally raises the standard of living of human beings. This applies to every aspect of life including, but not limited to, food and nutrition, transport, communication, healthcare etc. In the domain of human healthcare, one of the most important organs to take note of is the liver. The liver, which also excretes bile, controls most of the chemical levels of the bloodstream. This helps the liver remove waste products [1]. Any of the wide array of conditions which may cause damage to the liver is labelled as “liver disease”. Cirrhosis (scarring) can develop over time due to liver disease and without treatment more and more healthy tissue will be replaced with scar tissue which can prevent the liver from working as it should. Liver failure and liver cancer may result from untreated liver disease [2].

The likelihood of a person getting inflicted with liver disease depends on risk factors like alcohol consumption, obesity, diabetes, and hepatitis B and C virus infections. It becomes arduous to get a diagnosis of liver disease early on as the liver still functions without much symptoms even if some parts of it is damaged [3]. Therefore, a significant number of people die annually due to liver disease. Cirrhosis, as well as chronic liver disease, is responsible for around two million lives being lost each year throughout the world [4], and the modern lifestyle which includes desk jobs, increased alcohol consumption and smoking aids in the risk of liver disease. In the fight against liver disease, to give the affected a better chance of survival, it is crucial to diagnose the liver disease at the early stages [5].

In recent times, machine learning has been a powerful tool to analyze, evaluate and predict various medical conditions [6]. With a big enough dataset with varied parameters, machine learning and deep learning algorithms can find out, to a significant degree of accuracy, the relation between different symptoms to the possibility of having liver disease [7].

In Bangladesh, there is a doctor to patient ratio of 5.26 to 10,000 [8]. Therefore, this could prove to be a difference maker in third world countries like Bangladesh where there is a deficit of healthcare professionals.

1.2 Problem Statement

Liver diseases pose significant health challenges worldwide, affecting millions of individuals and leading to substantial morbidity and mortality rates [9]. Early and accurate detection of liver diseases plays a crucial role in improving patient outcomes, as timely intervention and treatment can prevent disease progression and associated complications [10]. However, conventional diagnostic approaches for liver diseases often rely on invasive procedures and expert interpretation, leading to potential delays, subjectivity, and resource limitations.

In recent years, machine learning techniques have demonstrated remarkable potential in various medical applications, including disease diagnosis and prediction. Leveraging the power of machine learning algorithms, particularly in the field of liver disease detection, has the potential to significantly enhance diagnostic accuracy and efficiency, thereby revolutionizing clinical practice.

Despite the promise of machine learning in liver disease detection, several challenges need to be addressed to ensure its successful implementation. Firstly, the availability of large-scale, diverse, and well-annotated datasets of liver disease cases is essential for training robust machine learning models. However, the scarcity of such datasets and the need for ensuring data quality and representativeness hinder the development of accurate and reliable models.

Secondly, the interpretability and explainability of machine learning models used for liver disease detection are crucial for clinical acceptance and trust. Physicians and healthcare professionals require transparency in the decision-making process to understand the reasoning behind the predictions made by these models and to ensure the integration of their expertise into the diagnostic workflow.

Furthermore, the integration of machine learning models into existing healthcare systems and workflows poses logistical challenges, including the need for seamless integration, scalability, and interoperability [11]. Overcoming these challenges is essential to facilitate the practical implementation and adoption of machine learning-guided liver disease detection in real-world clinical settings.

Therefore, this thesis aims to address the aforementioned challenges and contribute to the advancement of machine learning-guided liver disease detection. The research will focus on developing accurate and interpretable machine learning models by leveraging diverse and well-annotated liver disease datasets

By successfully addressing these challenges, this research endeavors to enhance diagnostic accuracy and efficiency in liver disease detection, leading to improved patient outcomes, reduced healthcare costs, and a significant advancement in the field of medical diagnostics.

1.3 Thesis Organization

This thesis focuses on using machine learning algorithms on a liver disease dataset to gain a high accuracy in the prediction of liver disease in a patient. This is done following the pre-processing of the dataset using various methods.

- ✓ In chapter 2, the literature review or background study is given where the research works done in this domain is reviewed.
- ✓ In Chapter 3, the methodology has been discussed which includes details about the data, where it has been sourced from and how it is structured. The processing techniques used have been mentioned in this section which includes the imbalance handling techniques used. This section also holds the different machine learning algorithms which were used.
- ✓ In chapter 4, the results which were found after the different algorithms were are discussed. This involves the results before and after handling imbalance and also after cross validation is performed.
- ✓ In Chapter 5, the results are discussed in detail. The different algorithms and imbalance handling techniques are compared to find the best performing one.
- ✓ In Chapter 6, the limitation of this study is given.
- ✓ In Chapter 7, the conclusion of the thesis is included, including a quick review of the findings and some recommendations for further research.

CHAPTER 2

Background Study

As time progresses so do the tools which assist in the workloads of human beings. Machine learning is rapidly becoming one of these tools which is aiding the world of disease prediction. Various research has been done on the utilization of machine learning techniques to find liver disease in recent years. The health industry has experienced a spike in the creation of applications as machine learning has grown in popularity in recent years [12]. Several studies have been conducted in the domain of liver disease detection using machine learning algorithms. For instance, Ritesh Choudhary et al. used Logistic Regression, Support Vector Classifier, Naïve Bayes, Random Forest, and Gradient Boosting algorithms to diagnose liver disease and predict risk [1]. The study found that Logistic Regression achieved the best result with an accuracy of 71%. Similarly, Elias Dritsas et al. proposed voting as their model of choice and used SMOTE as their imbalance handling technique, achieving an accuracy of 80.1% [13]. Shuwei Weng et al. used SMOTE-NC and 10-fold cross-validation, performing eight different machine learning algorithms on a liver disease dataset for the Chinese population [14]. They found XGBoost to be the best predictor among all tested methods with an accuracy of 89.7%. Other studies also compared multiple algorithms, such as Srilatha Tokala et al., who found Random Forest to perform the best with an accuracy of 87% [15]. Hong-Ye Peng et al. developed and validated five machine learning models for NAFLD (Non-alcoholic fatty liver disease), among which XGBoost demonstrated the best performance with an accuracy of 0.89 [16]. Furthermore, some studies explored multiple algorithms and found that Random Forest, Light GB, and AdaBoosting algorithm gave better results, such as Ketan Gupta et al.'s analysis of liver disease [17]. Bendi Venkata Ramana et al. used ten popular classification algorithms with a combination of four feature selection methods [18], while Xieyi Pei et al. used SMOTE as their imbalance handling technique and found XGBoost to have a better prediction of Fatty Liver disease among individuals with minimal variables [19]. A.Sivasangari et al. analyzed the liver disease dataset and found SVM to have the highest accuracy of 95.18% [20]. Shamima Akter et al. also analyzed liver disease data and found Random Forest and CART to have 94% and 95% accuracy, respectively [21]. Maria Alex Kuzhippallil et al. used feature selection and outlier elimination, finding that all 10 machine learning algorithms' accuracies improved, with

Random Forest having the highest accuracy of 88% [22]. Mohammad Fathi et al. presented their analysis on two different datasets using SVM for the classification of liver disease, achieving an accuracy of 90.9% and 92.2% for the ILDP and BUPA datasets, respectively [23].

CHAPTER 3

Methodology

3.1 Data Collection

Before machine learning (ML) can be effectively applied, the process of data collection plays a pivotal role. Data serves as the lifeblood for ML algorithms, providing the necessary information for them to learn and make accurate predictions or decisions. The quality, quantity, and diversity of the collected data directly impact the performance and reliability of ML models. By amassing a substantial and diverse dataset, researchers and practitioners can ensure that the models are trained on a wide range of scenarios, enabling them to generalize and handle various real-world situations. Furthermore, comprehensive data collection allows for the identification of patterns, correlations, and insights that might otherwise remain hidden. Thus, data collection serves as the foundation upon which ML algorithms are built, allowing them to extract valuable knowledge and deliver meaningful results.

3.1.1 Data Source:

The "Indian Liver Patient Records" dataset used in this research was graciously provided by Dr. S. Sridhar, Dr. Bevera Lakshmana Rao, K. Ravi Kanth, and B. Sai Prasad [24]. Initially, the dataset was utilized for a comparative analysis of liver patients from the United States and India, conducted by these esteemed researchers [25]. The study aimed to explore and compare various aspects of liver diseases, such as prevalence, risk factors, and treatment outcomes, between the two countries. Subsequently, recognizing its value for the broader scientific community, the dataset was made publicly available. This decision allowed researchers worldwide to access and utilize the dataset for further investigations and advancements in liver disease research.

3.1.2 Data Structure:

The "Indian Liver Patient Records" dataset is structured as a single CSV file, presenting a well-organized format for analysis. It contains 583 records from the North East region of Andhra Pradesh, India, comprising 416 liver patient records and 167 non-liver patient records. Each row represents a patient's comprehensive medical information, while the columns capture different attributes and features relevant to liver disease. The dataset's tabular structure allows for efficient organization and enables researchers to explore correlations between various features and the occurrence of liver disease. The dataset includes a "Dataset" column serving as the class label, distinguishing liver patients from non-liver patients. Additionally, it consists of 441 male patient records and 142 female patient records. Notably, for privacy protection, any patient aged over 89 is labeled as "90". This dataset provides valuable insights into liver disease patterns in the specified region, making it an essential resource for researchers studying liver diseases and their associated factors in this population.

Sl No.	Attribute Name	Max	Mean	Min	Standard Deviation
1	Age	90	44.75	4	16.18
2	Total Bilirubin	75	3.298	0.4	6.209
3	Direct Bilirubin	19.7	1.486	0.1	2.8
4	Alkaline Phosphotase	2110	290.57	63	242.93
5	Alamine Aminotransferase	2000	80.71	10	182.62
6	Aspartate Aminotransferase	4929	109.91	10	288.91
7	Total Protiens	9.6	6.48	2.7	1.08
8	Albumin	5.5	3.14	0.9	0.795
9	Albumin and Globulin Ratio	2.8	0.94	0.3	0.3195
10	Datasetc(target)	2	1.286	1	0.452

Table 3.1: statistical information of the attributes

During our preliminary investigation, we discovered that the dataset contained a minimal number of duplicated entries, with only 13 instances of duplicated data. Additionally, we

identified that there were four missing values specifically related to the Albumin and Globulin Ratio feature.

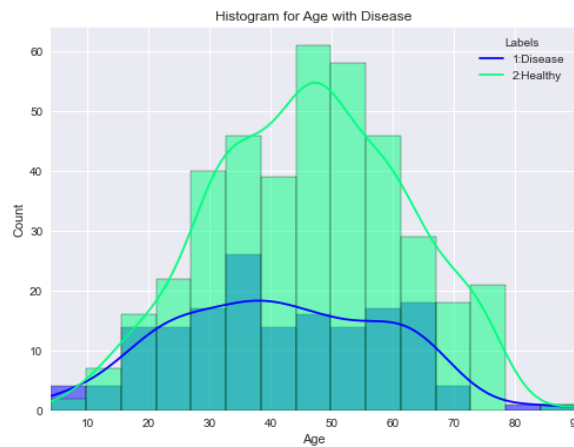


Fig 3.1: Histogram for Age with Disease

3.1.3 Data Features:

The features in the "Indian Liver Patient Records" dataset play a crucial role in identifying and understanding liver diseases. Age, as an important feature, provides insights into the patient's age group, which can be a significant factor in determining the susceptibility to liver diseases. Liver diseases often exhibit age-related patterns, and certain conditions may be more prevalent in specific age ranges [26]. Gender, another feature, is significant because liver diseases can exhibit gender-specific variations [27]. By considering the gender of the patient, researchers can analyze and identify potential disparities or trends in the occurrence and progression of liver diseases between male and female populations. Biochemical markers such as Total Bilirubin (TB), Direct Bilirubin (DB), Alkaline Phosphatase (Alkphos), Alamine Aminotransferase (Sgpt), and Aspartate Aminotransferase (Sgot) provide critical information about liver function and damage. Elevated levels of these markers can indicate liver dysfunction and help in the diagnosis and monitoring of liver diseases. Total Proteins (TP), Albumin (ALB), and the Albumin and Globulin Ratio (A/G Ratio) are essential indicators of liver health. Abnormal levels of these proteins can signify liver disorders, as they reflect the

liver's synthetic and detoxification functions. A decrease in albumin levels and alterations in the A/G ratio can point towards liver disease progression. The Selector feature, used to split the data into liver disease and non-liver disease groups, is crucial in building predictive models and evaluating the performance of algorithms. By categorizing patients based on expert opinions, researchers can train models to accurately identify liver disease cases. Overall, these features collectively provide valuable information for the identification, diagnosis, and monitoring of liver diseases. Understanding the relationships and patterns between these features and liver disease outcomes can assist in developing effective prediction models and improving patient care and management strategies.

3.1.4 Data Preprocessing:

3.1.4.1 Cleaning Duplicate Data:

Dataset may contain rows with duplicate sets of values. When all the values in all the columns match exactly or are identical, there are said to be duplicate rows [28]. The rows with duplicate values were eliminated in order to guarantee data integrity and reduce redundancy in the dataset for forecasting liver disease. We ensure the dataset contains unique and non-redundant information by dropping the duplicate rows, boosting the efficacy and integrity of subsequent research and model development.

3.1.4.2 Handling Missing Data:

Dataset occasionally contain null or missing values. These missing data can be handles either by dropping the records or by using various data imputation techniques. In the proposed work, the missing data has been handled by filling in the missing values using two techniques.

For the data type having integer or float numbers the missing data has been handled by filling in the data by the mean value of the corresponding feature. This method, known as mean imputation, involves calculating the average value of the feature from the available data and replacing the missing values with this average. Mean imputation is an easy-to-use method that aids in preserving the dataset's general statistical characteristics [29]. The imputed dataset maintains the same mean value for the feature by substituting missing values with the mean, reducing the effect on the distribution and avoiding potential bias.

For the object type data, the missing data has been filled up by using the mode value of the corresponding feature. The value of a variable that appears the most frequently is its mode. We

impute the missing data with the value that is most frequently seen for that specific feature by replacing the missing values with the mode. This method makes sure that the imputed values are in line with the majority of the observations while maintaining the distribution and frequency of the existing data.

3.1.4.3 Data Encoding:

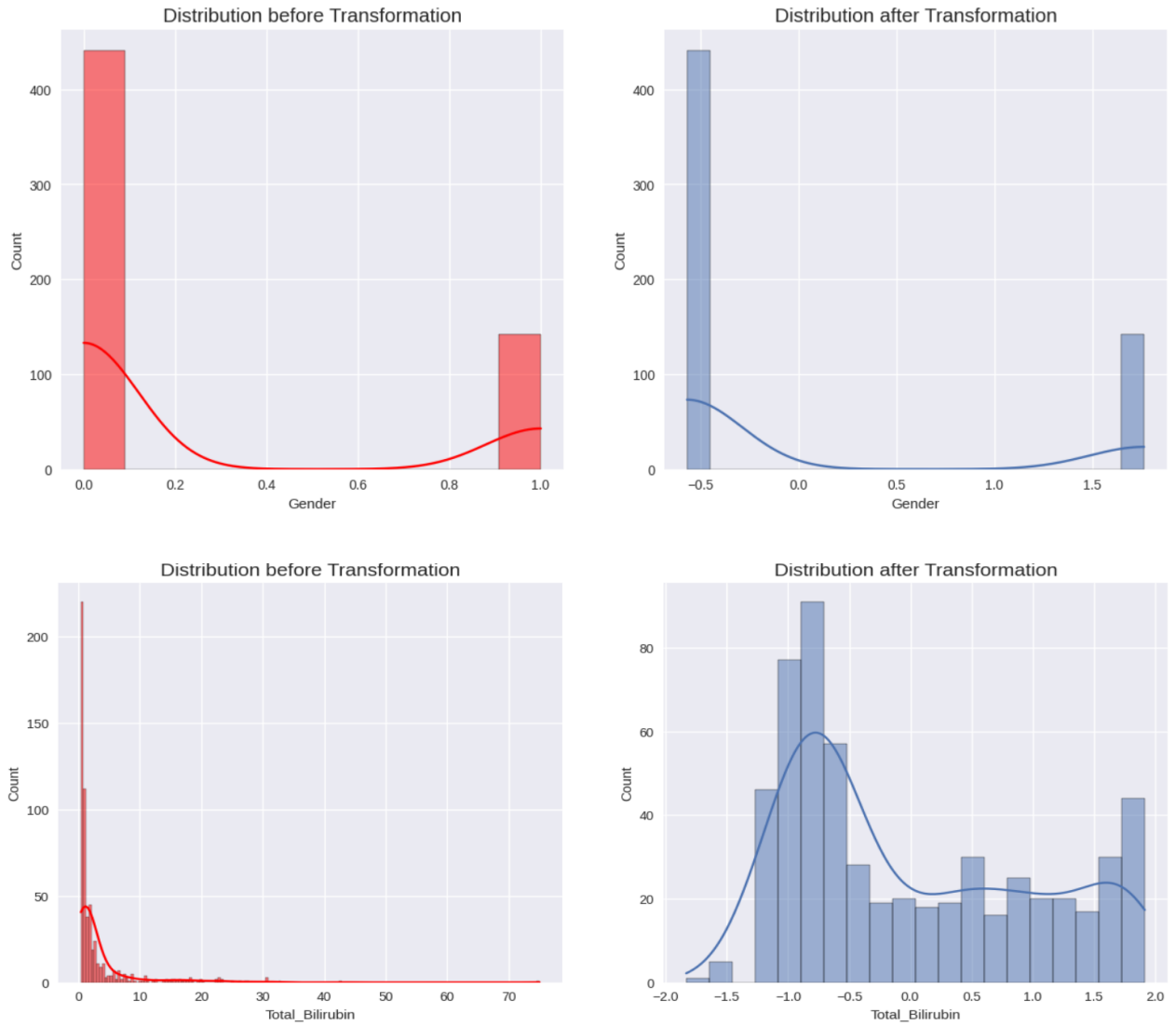
Before training different models, categorical data must be transformed into numerical values. Data replacement is used to perform this conversion. There is only one categorical feature—gender—in the Indian Liver dataset. The female and male classes in gender columns are replaced as 0 and 1, respectively. This procedure preserves the fundamental data included in the categorical characteristics while also ensuring compliance with the selected algorithms.

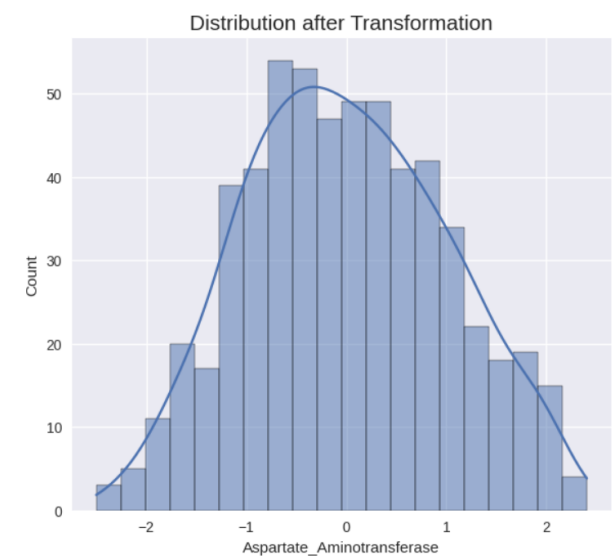
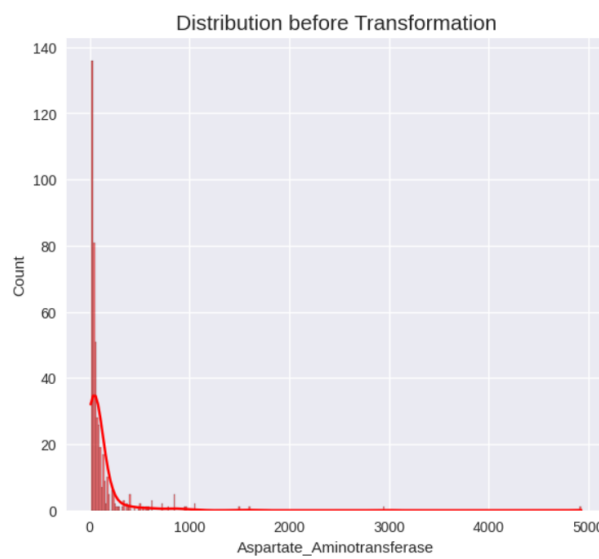
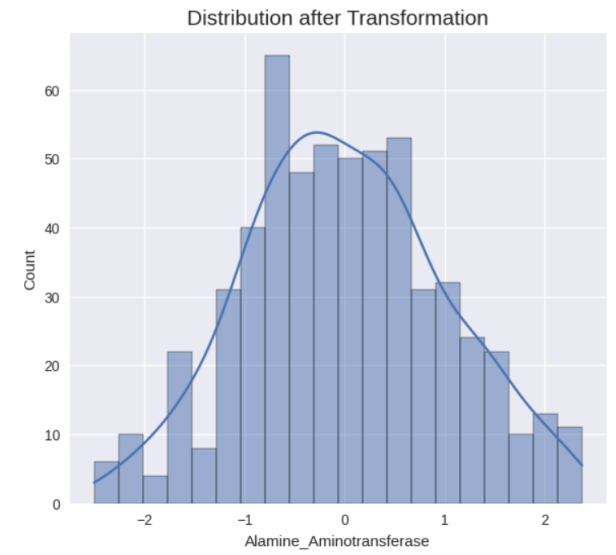
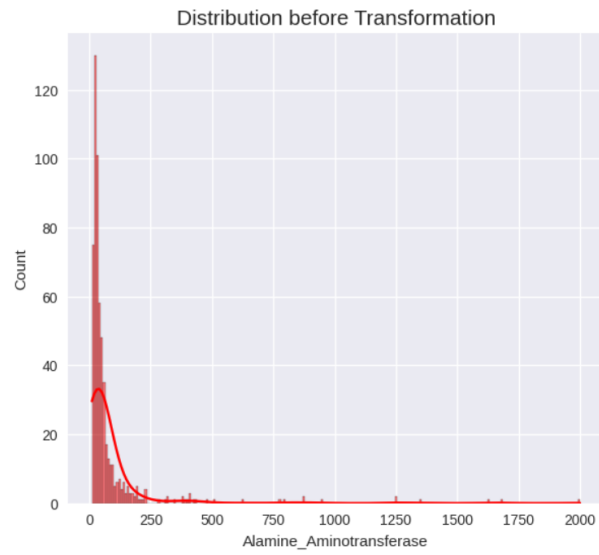
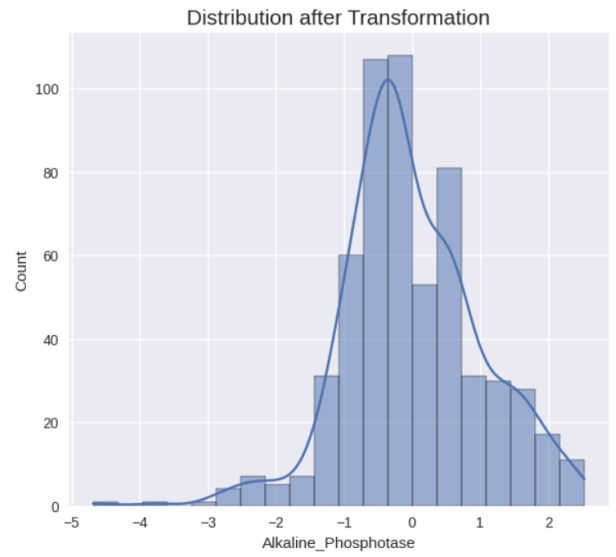
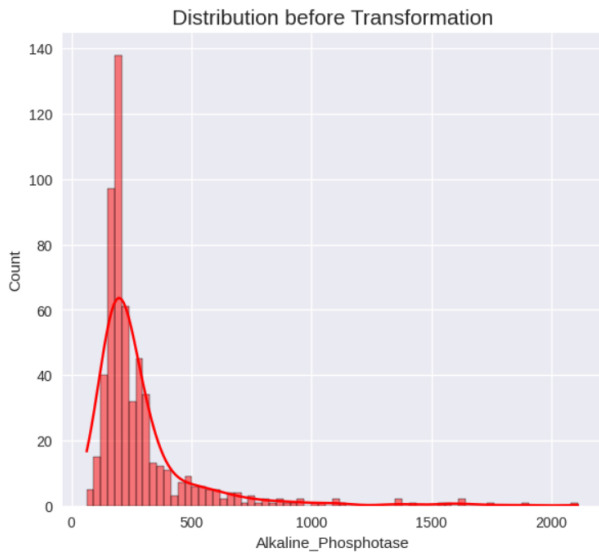
3.1.4.4 Transforming Skew Data:

The term "skewness" describes the asymmetry of a variable's distribution, with a positive skew denoting a longer tail on the right side and a negative skew denoting a longer tail on the left. Skewed data can have a negative impact on performance by going against model presumptions or by changing how feature importances are interpreted. There many techniques by which the skewness of the data can be brought into a symmetrical distribution. The Yeo-Johnson transformation was used to solve the issue of skewed data in the context. A power transformation method that can handle both positively and negatively skewed data is the Yeo-Johnson transformation. The Yeo-Johnson transformation can handle data with zero or negative values, unlike other transformations like the logarithmic or square root transformations [30]. A more symmetrical distribution was achieved by applying the Yeo-Johnson transformation to the skewed variables, which improved the data's suitability for statistical methods and machine learning algorithms that presume normality. The overall patterns and relationships within the data are preserved but the impact of outliers and extreme numbers is reduced. The effectiveness of machine learning models may be enhanced by this. The equation (1) for is given below-

$$x_i'^{\lambda} = \begin{cases} \frac{x_i^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0, x_i \geq 0 \\ \log(x_i + 1) & \text{if } \lambda = 0, x_i \geq 0 \\ \frac{-[(-x_i+1)^{2-\lambda}-1]}{2-\lambda} & \text{if } \lambda \neq 2, x_i < 0 \\ -\log(-x_i + 1) & \text{if } \lambda = 2, x_i < 0 \end{cases}$$

The distribution graph of the data before and after the transformation has been applied can be observed from the figure (3.2)





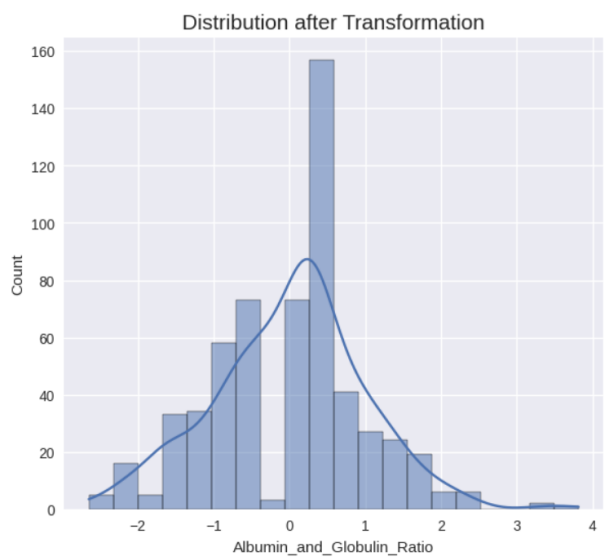
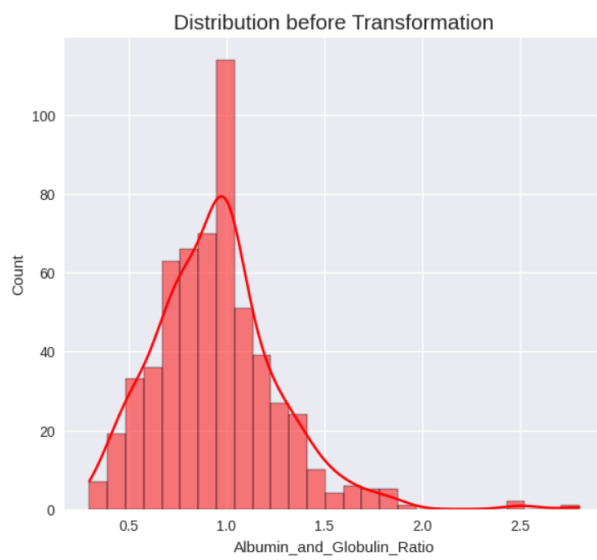
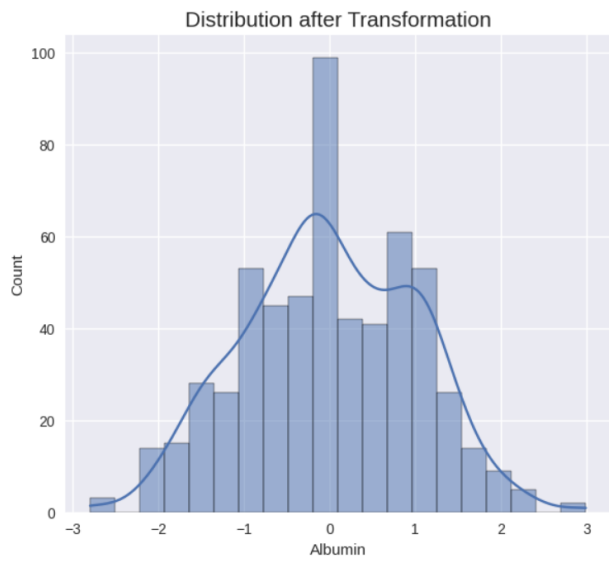
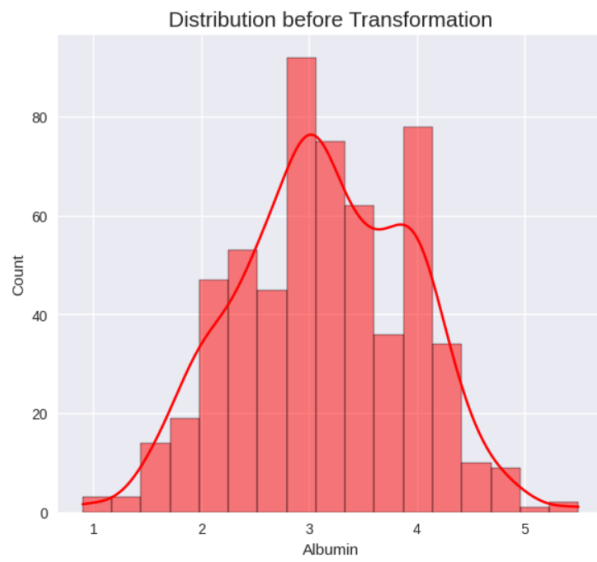
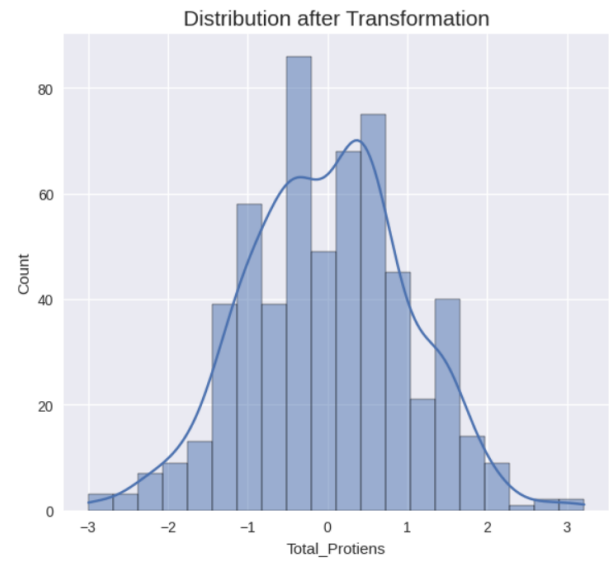
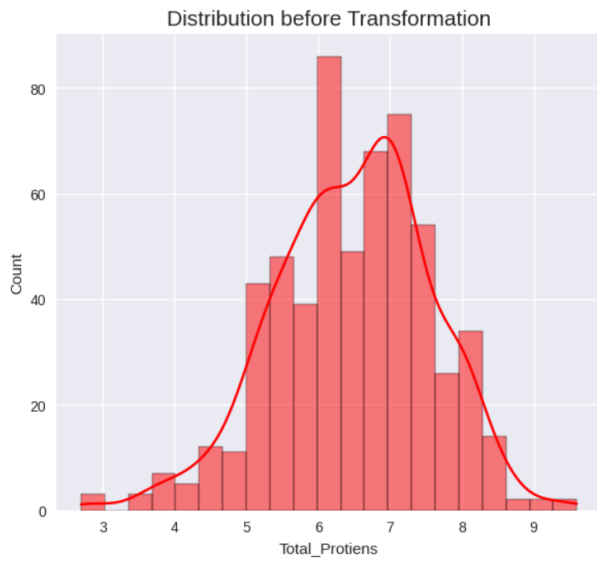


Fig 3.2: Observing skewness before and after transformation

3.2 Feature Selection:

Data Dropping due to correlation:

Data dropping was done using a correlation heatmap to simplify the dataset and increase the effectiveness of the predictive modeling procedure. The correlation heatmap revealed information about the connections between the various dataset variables. According to the correlation heatmap's findings, the column 'Direct Bilirubin' was chosen and then deleted from the dataset. This column was eliminated because of its high correlation with another variable, which suggested a strong link or duplication of information. This column's removal aids in the reduction of multicollinearity, which can have a negative effect on the effectiveness and interpretability of machine learning models.

The dataset is now more refined and suited for further research due to data dropping based on the correlation heatmap. With the help of this procedure, the remaining variables are made to be less heavily reliant on one another and a more accurate picture of the independent traits that are involved in predicting liver disease is given.

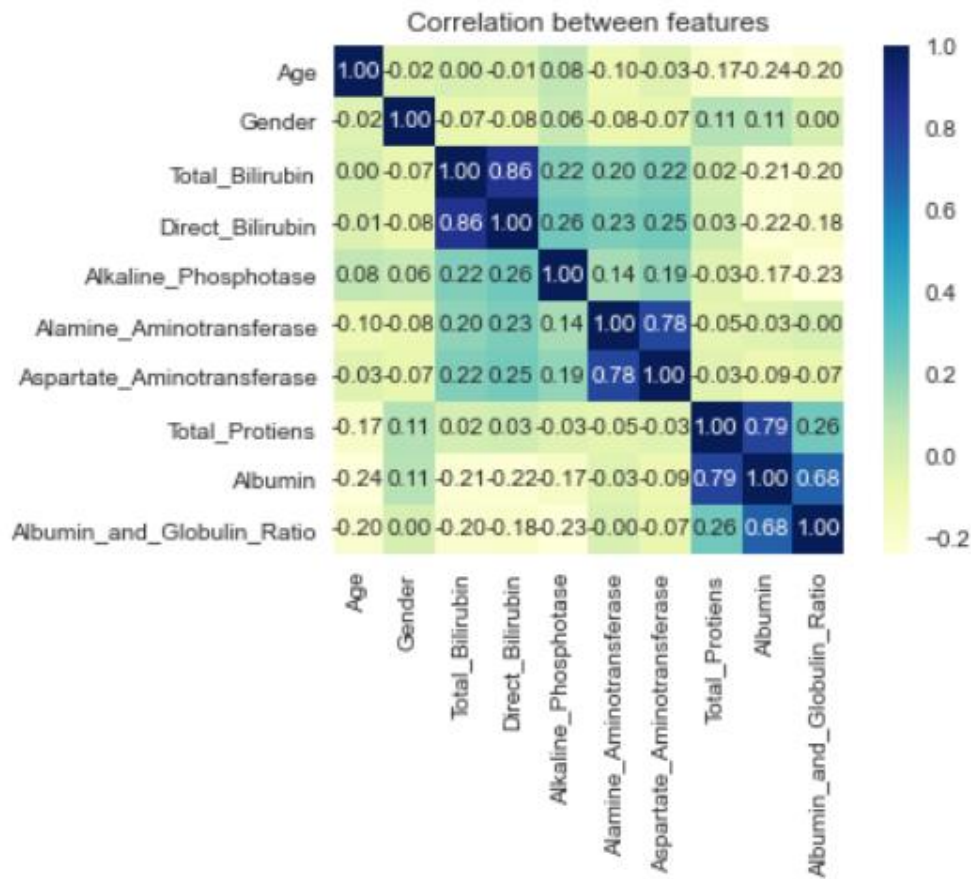


Fig 3.3: Correlation heatmap of attributes

3.3 Scaling:

Scaling, in the context of machine and deep learning, refers to the process of adapting or transforming the input data to a specific range or distribution that is suitable for the learning algorithms and neural networks [31]. It involves manipulating the feature values to ensure that they are on a similar scale or have comparable magnitudes.

Scaling is essential in machine and deep learning for several reasons:

Numerical Stability: Scaling helps to stabilize the learning process by preventing numerical instabilities that can arise when working with features that have significantly different scales. Large differences in feature magnitudes can cause issues like slow convergence, vanishing/exploding gradients, and difficulties in optimizing the model.

Improved Optimization: Scaling can enhance the efficiency and effectiveness of optimization algorithms. Gradient-based optimization techniques, such as stochastic gradient descent

(SGD), rely on the magnitudes of gradients to adjust the model parameters. When features are on different scales, the gradients can vary widely, leading to suboptimal convergence. Scaling mitigates this problem by ensuring that the gradients have consistent magnitudes.

Balanced Influence: Scaling ensures that all features contribute proportionally to the learning process. If some features have larger scales than others, they might dominate the learning process and overshadow the importance of other features. By scaling the features, their influences are balanced, allowing the model to learn from all the relevant information [32-33].

A few different kinds of scaling which has been used in this study are Min-Max scaling and standard scaling.

3.3.1 Min-Max Scaling:

Min-max scaling, also known as normalization, is a data preprocessing technique used to transform numerical features into a common scale. The goal is to map the values of a feature to a specific range, typically between 0 and 1. This process is crucial when dealing with features that have different scales or units, as it ensures fair comparisons and prevents certain features from dominating the learning process [35].

To perform min-max scaling, we follow a straightforward formula for each feature:

$$X_{\text{scaled}} = (X - X_{\text{min}}) / (X_{\text{max}} - X_{\text{min}})$$

Here, X represents the original value of a feature, X_{min} is the minimum value in that feature, and X_{max} is the maximum value. By subtracting the minimum value and dividing by the range (the difference between the maximum and minimum values), we obtain the scaled value X_{scaled} within the desired range. A visual aid for min-max scaling is given below.

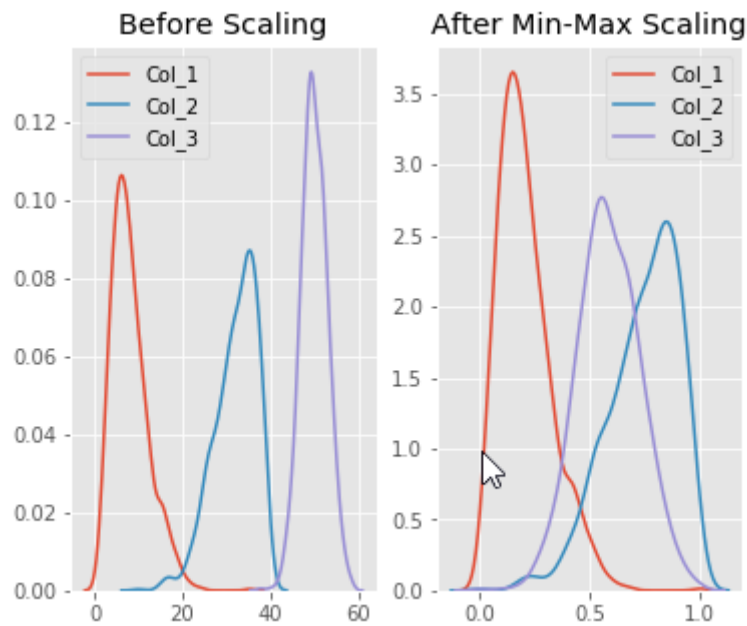


Figure 3.4: Before and after Min-Max Scaling

One significant advantage of min-max scaling is that it maintains the shape and distribution of the original feature while transforming its values. It is particularly useful in scenarios where the scale of features varies widely. For example, if one feature ranges from 0 to 100 and another from 0 to 100,000, the latter feature would dominate the learning process if left unscaled. By applying min-max scaling, we bring both features to a comparable range, ensuring that they contribute equally during model training.

Min-max scaling plays a crucial role in machine learning algorithms, especially those that utilize gradient-based optimization methods. Without scaling, features with larger magnitudes can have a more significant impact on the model's weights and biases, leading to biased or inefficient learning. By scaling the features to a common range, we mitigate the influence of outliers and improve the convergence speed and performance of the learning algorithm.

It's worth noting that min-max scaling is sensitive to outliers. If a feature contains extreme values, they can disproportionately affect the scaling process and potentially compress the majority of the values. In such cases, alternative scaling techniques like standardization (Z-score scaling) may be more appropriate, as they are more robust against outliers.

Overall, min-max scaling is a vital preprocessing step in machine learning. It ensures that features are uniformly scaled to a common range, allowing models to make fair and accurate comparisons. By employing this technique, we enable efficient learning, mitigate the impact of varying feature scales, and improve the overall performance of machine learning algorithms.

3.3.2 Standard Scaling:

Standard scaling, also known as Z-score scaling or standardization, is a popular technique used for feature scaling in machine learning. It transforms numerical features to have zero mean and unit variance, resulting in a standardized distribution [34].

The formula for standard scaling is as follows:

$$X_{\text{scaled}} = (X - X_{\text{mean}}) / X_{\text{std}}$$

Here, X represents the original value of a feature, X_{mean} is the mean of that feature, and X_{std} is the standard deviation. By subtracting the mean and dividing by the standard deviation, we obtain the scaled value X_{scaled} . A visual aid of Standard Scaling is given below.

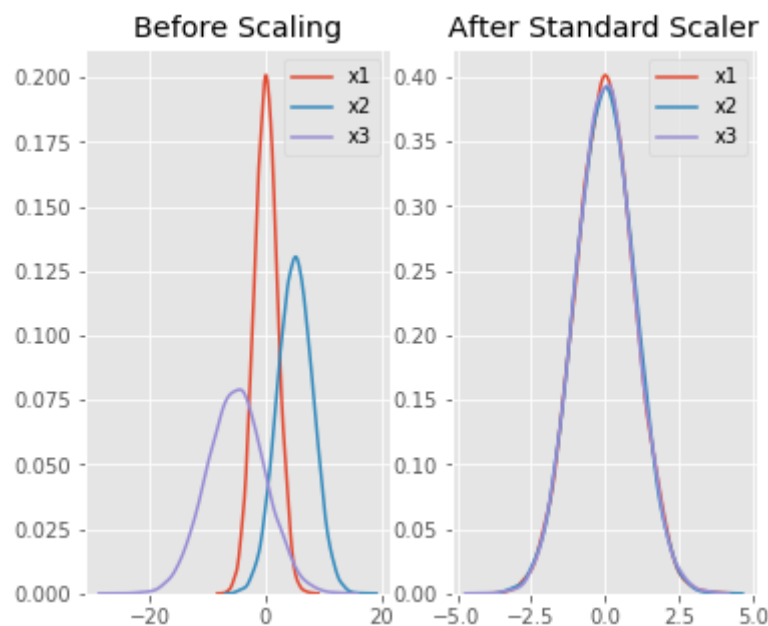


Figure 3.5: Before and after Standard Scaling

Standard scaling offers several advantages in machine learning. First and foremost, it brings features to a common scale, making them directly comparable. This is especially useful when features have different units or scales, as it ensures that each feature contributes proportionally

to the learning process. Standardization also helps in cases where the distribution of a feature is skewed or has outliers. By transforming the data to have zero mean and unit variance, it reduces the impact of extreme values and makes the data more suitable for algorithms that assume a Gaussian distribution.

Another advantage of standard scaling is that it simplifies the interpretation of feature importance. Since all features are on the same scale, the magnitude of their coefficients or weights in a model can directly indicate their relative importance. This is particularly valuable in linear models, where the coefficients represent the contributions of features to the output.

However, it is important to note that standard scaling is sensitive to outliers. Outliers can have a significant influence on the mean and standard deviation, affecting the scaled values. In such cases, robust scaling techniques that are less affected by outliers, such as min-max scaling or quantile scaling, may be more appropriate.

Additionally, standard scaling does not guarantee a specific range for the scaled values. The transformed values can be positive or negative, depending on their relation to the mean. If maintaining a specific range is necessary, alternative scaling techniques like min-max scaling may be more suitable.

In summary, standard scaling is a widely used technique in machine learning that transforms features to have zero mean and unit variance. It enables fair comparisons between features, simplifies feature interpretation, and is particularly effective in handling skewed distributions. However, it can be sensitive to outliers and does not enforce a specific range for the scaled values. Careful consideration should be given to the specific characteristics of the data and the requirements of the machine learning algorithm when choosing the appropriate scaling technique.

3.4 Handling Imbalanced Data:

Imbalanced data refers to a situation where the classes or categories in a dataset are not represented equally. In other words, one class has a significantly larger number of instances compared to the other class(es), resulting in an imbalance in the distribution of the classes [35]. This issue is common in various real-world scenarios, such as fraud detection, disease diagnosis, anomaly detection, and rare event prediction.

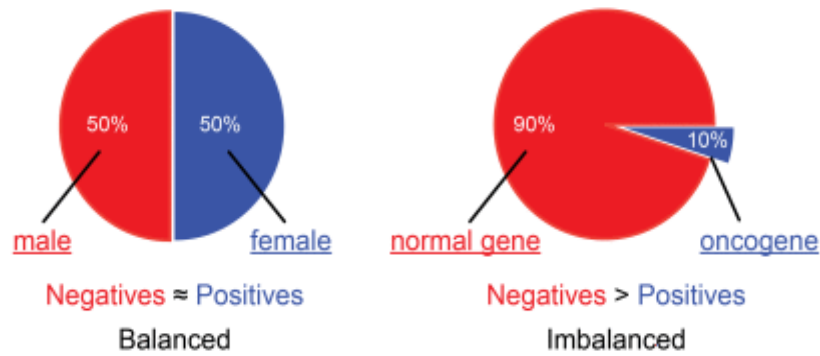


Figure 3.6: Example of balanced and imbalanced data

Imbalanced data can cause several challenges in machine learning [36-37]:

Biased model performance: When faced with imbalanced data, machine learning models tend to favor the majority class, as it provides higher accuracy by simply predicting the majority class for most instances. As a result, the model may have poor performance in predicting the minority class, which is often the class of interest in many applications.

Misleading evaluation metrics: Traditional evaluation metrics, such as accuracy, can be misleading when dealing with imbalanced data. For instance, if a dataset has 95% of instances belonging to the majority class and only 5% belonging to the minority class, a model that predicts the majority class for all instances would achieve 95% accuracy. However, this model provides no meaningful insights about the minority class. Therefore, alternative evaluation metrics like precision, recall, F1-score, or area under the receiver operating characteristic curve (AUC-ROC) are often used to assess model performance on imbalanced datasets.

Sampling bias: Imbalanced data can introduce sampling bias, where the model learns to favor the majority class due to the abundance of its instances. This bias can lead to incorrect predictions and reduced generalization on unseen data. The model may fail to capture the patterns and characteristics specific to the minority class.

Difficulty in detecting rare events: Imbalanced data often occurs in scenarios where the minority class represents rare events or anomalies that are of particular interest. Identifying these rare events becomes challenging due to their scarcity in the dataset. Machine learning

models trained on imbalanced data may struggle to recognize and accurately classify these rare events.

To handle the imbalance, there are various kinds of imbalance handling techniques. The ones used in this study is described in detail in the following articles.

3.4.1 SMOTE:

SMOTE (Synthetic Minority Over-sampling Technique) is a popular technique used to address the issue of imbalanced data by generating synthetic samples for the minority class. It aims to increase the diversity and representation of the minority class in the dataset. SMOTE works by creating synthetic examples along the line segments connecting pairs of minority class instances [38].

Below are the step-by-step process of SMOTE:

1. Identify the minority class: First, we identify the minority class in the imbalanced dataset. The minority class is the class with fewer instances.
2. Select a minority class instance: Randomly select an instance from the minority class as the starting point for the synthetic sample generation.
3. Find k nearest neighbors: Calculate the k nearest neighbors for the selected instance. The value of k is a user-defined parameter.
4. Choose a random neighbor: Randomly select one of the k nearest neighbors and denote it as the neighbor instance.
5. Create a synthetic sample: For each feature in the dataset, calculate the difference between the feature values of the selected instance and the neighbor instance. Multiply this difference by a random number between 0 and 1. Add the result to the feature values of the selected instance to create a new synthetic sample.
6. Repeat the process: Repeat steps 2-5 to generate a desired number of synthetic samples for the minority class.

By generating synthetic samples, SMOTE increases the number of instances in the minority class, resulting in a more balanced distribution between the classes.

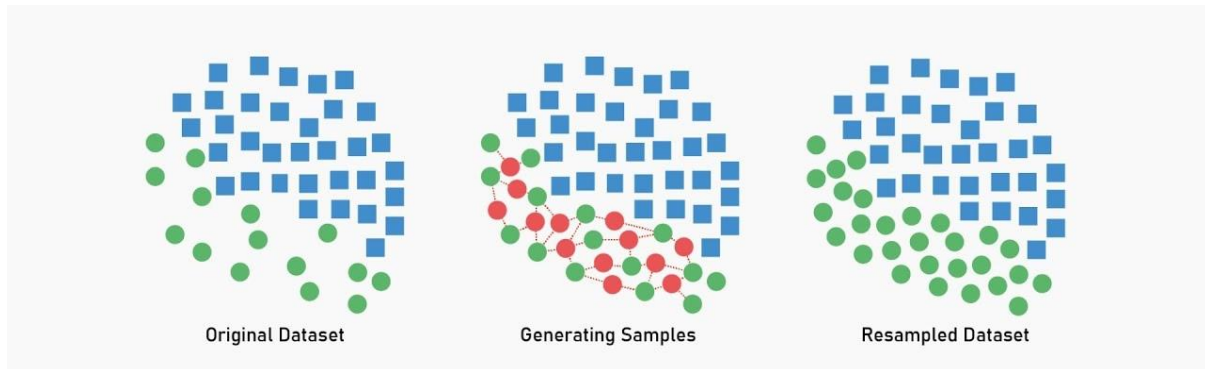


Figure 3.7: Before, during and after applying SMOTE to a dataset

The equation used in SMOTE to create synthetic samples is as follows:

$$\text{Synthetic sample} = \text{Instance} + (\text{Neighbor} - \text{Instance}) * \text{random_number}$$

Here, "Instance" represents the feature values of the selected instance, "Neighbor" represents the feature values of the neighbor instance, and "random_number" is a random value between 0 and 1.

Advantages of SMOTE:

SMOTE helps to address the class imbalance problem by increasing the representation of the minority class, thereby improving the model's ability to learn its patterns [38]. The synthetic samples generated by SMOTE are based on the existing instances, which helps to preserve the characteristics and distribution of the minority class [49].

Disadvantages of SMOTE:

SMOTE may introduce some degree of overfitting, as the synthetic samples are created by interpolating existing instances [39]. This can potentially amplify the noise or outliers present in the minority class. SMOTE is less effective in scenarios where the minority class instances are highly overlapping or densely packed, as the synthetic samples may not introduce significant diversity.

3.4.2 ADASYN:

ADASYN (Adaptive Synthetic Sampling) is a data augmentation technique specifically designed to handle imbalanced datasets. It focuses on generating synthetic samples for the minority class based on the density distribution of the instances [40]. ADASYN aims to address the limitation of SMOTE, where the synthetic samples are equally generated for all instances regardless of their level of difficulty in learning [42].

Here is the step-by-step process of ADASYN:

- Identify the minority class: Similar to other techniques, ADASYN starts by identifying the minority class in the imbalanced dataset.
- Compute the level of imbalance: Calculate the imbalance ratio of the dataset, which is the ratio of the majority class instances to the minority class instances.
- Calculate the required number of synthetic samples: Determine the number of synthetic samples to be generated for each minority class instance based on its difficulty in learning. This calculation takes into account the imbalance ratio and the density distribution of instances.

Compute the synthetic samples: For each minority class instance, ADASYN performs the following steps [41]:

- a. Calculate the k nearest neighbors for the instance. The value of k is a user-defined parameter.
- b. Determine the relative density of each neighbor by considering the imbalance ratio. The relative density is a measure of how much the instance is surrounded by majority class instances compared to the minority class instances.
- c. Calculate the contribution factor for each neighbor based on its relative density. The contribution factor reflects the importance of the neighbor in generating synthetic samples.
- d. Generate synthetic samples for the instance by interpolating between the instance and its selected neighbors. The number of synthetic samples is determined by the required number of synthetic samples calculated in step 3, and the contribution factor is used to determine the weights for the interpolation.

The equation used in ADASYN to calculate the contribution factor is as follows:

Contribution Factor = (Relative Density of Neighbor) / (Sum of Relative Densities of Neighbors)

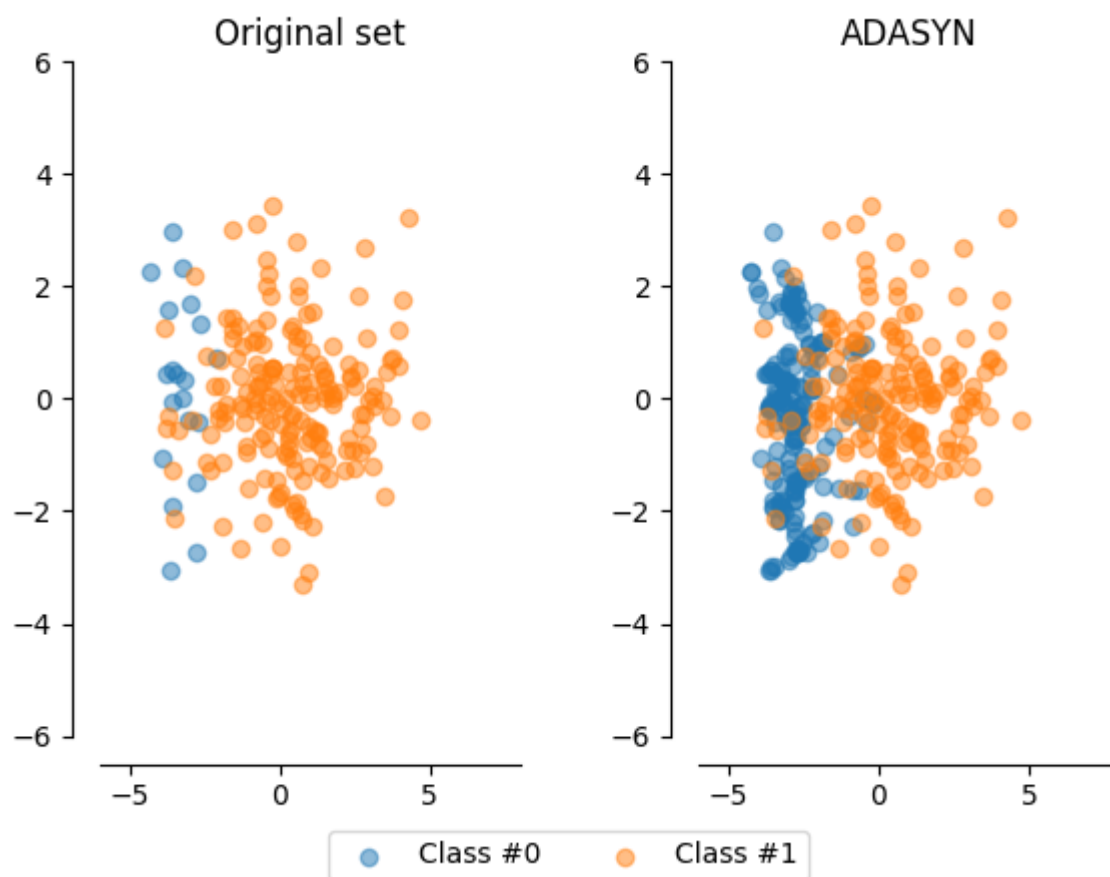


Figure 3.8: Before and after ADASYN applied to a dataset

Advantages of ADASYN:

- ADASYN adapts to the density distribution of instances, focusing on generating synthetic samples for the instances that are more difficult to learn.
- It helps to further address the class imbalance problem by increasing the representation of the minority class, especially in regions where it is sparsely represented.

Disadvantages of ADASYN:

- ADASYN may still introduce some degree of overfitting if the synthetic samples amplify noise or outliers in the minority class.
- The performance of ADASYN heavily relies on the choice of the k parameter, which determines the number of nearest neighbors considered.

3.4.3: SMOTE + TOMEKLINKS

SMOTE is a popular oversampling technique that generates synthetic samples for the minority class by interpolating new instances between existing minority class instances. It helps to balance the class distribution and improve the performance of classifiers. However, SMOTE may also generate noisy or irrelevant synthetic instances, which can affect the classifier's performance. To address this issue, SMOTE+Tomek Links combines SMOTE with the Tomek Links under sampling technique [43].

Tomek links:

Tomek Links are a technique used for cleaning up imbalanced datasets in machine learning. They are pairs of samples from different classes that are close to each other but are considered to be misclassified instances or outliers. By removing these instances, the decision boundary of a classifier can be improved [44].

Here's the process of identifying Tomek Links:

For each instance in the dataset, calculate its distance to the nearest instance of the opposite class.

If the distance to the nearest instance of the opposite class is smaller than the distance to the nearest instance of the same class, then the pair of instances is considered a Tomek Link.

Once Tomek Links are identified, there are two common approaches for handling them:

Tomek Link Undersampling: In this approach, one instance of each Tomek Link pair is removed from the dataset. By removing these instances, the overlapping region between the

classes is reduced, making it easier for classifiers to discriminate between the classes. The equation below illustrates the removal of instances:

$$D' = D - \{(x, y) \mid x \text{ and } y \text{ form a Tomek Link pair}\}$$

Where:

D' - The new dataset after removing Tomek Link instances.

D - The original dataset.

(x, y) - A pair of instances forming a Tomek Link.

Tomek Link Combination: In this approach, the majority class instances involved in the Tomek Link pairs are removed, while the minority class instances are retained. This method aims to enhance the separability of the minority class by eliminating majority class instances that are near the minority class. The equation below illustrates this process:

$$D' = D - \{x \mid x \text{ is a majority class instance involved in a Tomek Link pair}\}$$

Where:

D' - The new dataset after removing majority class instances in Tomek Links.

D - The original dataset.

x - A majority class instance involved in a Tomek Link pair.

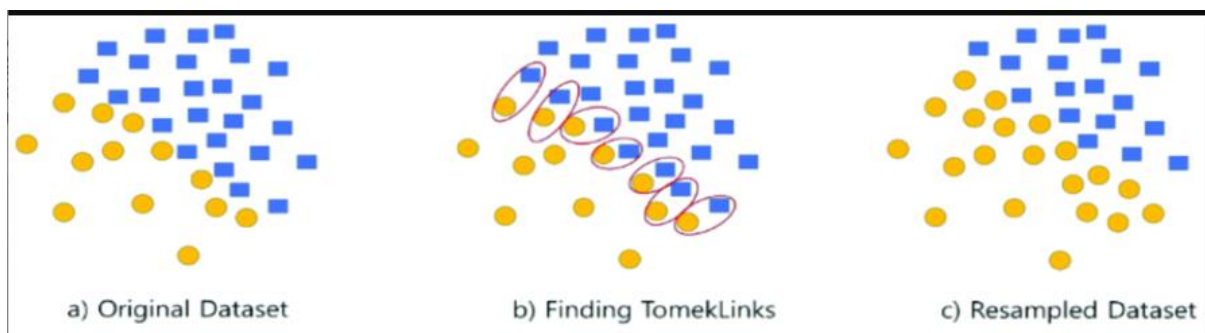


Figure 3.9: a) Original Dataset b) Finding Tomeklinks c) Resampled Dataset

Now, to apply SMOTE+Tomek links involves the following steps [45]:

1. Apply SMOTE: Generate synthetic instances for the minority class by interpolating between existing minority class instances. This step increases the number of minority class instances. The parameters for SMOTE include the number of synthetic instances to generate (N) and the number of nearest neighbors to consider (K).
2. Identify Tomek Links: After applying SMOTE, identify Tomek Links between the synthetic minority class instances and the majority class instances. Tomek Links are pairs of instances, one from the minority class and one from the majority class, that are closest to each other but belong to different classes. This step aims to identify and remove noisy or misclassified synthetic instances.
3. Remove instances involved in Tomek Links: Remove the instances involved in the Tomek Links identified in the previous step. This can be achieved by selecting one of the instances in each Tomek Link for removal. The decision on which instance to remove can be based on various criteria, such as removing the instance from the majority class to reduce noise.
4. The resulting dataset after removing the instances involved in Tomek Links is the final preprocessed dataset.

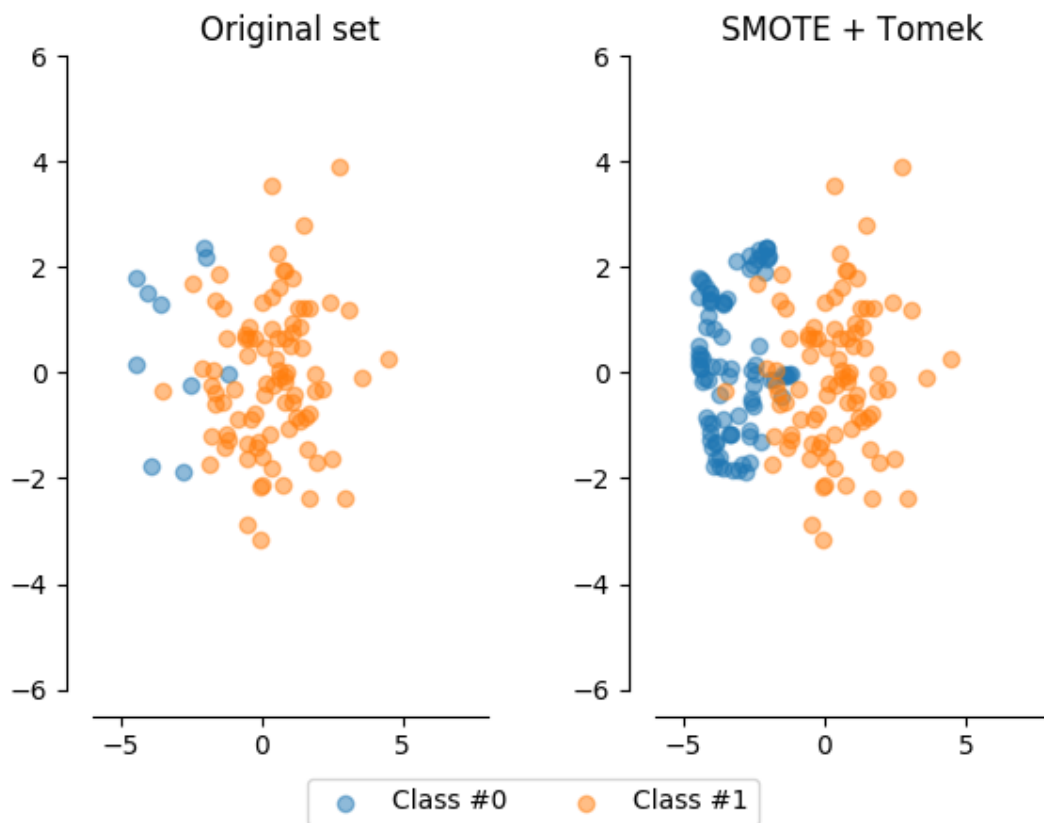


Figure 3.10: Original dataset (left) and Dataset after smote+Tomeklinks (right) has been applied

Advantages of SMOTE+TOMEKLINKS:

1. Improved performance: SMOTE+Tomek Links can effectively address the class imbalance problem by oversampling the minority class and undersampling the majority class [46-48]. This can lead to improved performance of classifiers, especially when dealing with imbalanced datasets.
2. Noise reduction: By applying Tomek Links after SMOTE, noisy or misclassified synthetic instances generated by SMOTE can be identified and removed. This helps in reducing the noise in the dataset and improves the quality of the minority class samples [50].
3. Preserves important instances: SMOTE+Tomek Links focuses on preserving the important minority class instances while removing noisy instances and instances near the decision boundary. This can help classifiers to better generalize and make more accurate predictions.

4. **Simplicity:** The implementation of SMOTE+Tomek Links is relatively straightforward. It involves applying SMOTE to generate synthetic instances and then applying Tomek Links to remove undesirable instances. This simplicity makes it easy to integrate into the machine learning pipeline.

Disadvantages of SMOTE+Tomek Links:

1. **Potential information loss:** Although SMOTE+Tomek Links aim to remove noisy or misclassified instances, there is a possibility of removing valid instances that are near the decision boundary. This can lead to information loss and may affect the classifier's performance.
2. **Increased computational complexity:** Applying both SMOTE and Tomek Links can increase the computational complexity compared to using either technique alone. Generating synthetic instances with SMOTE and then identifying and removing Tomek Links require additional computational resources and time.
3. **Sensitivity to parameter selection:** SMOTE+Tomek Links, like other machine learning techniques, has certain parameters that need to be set appropriately. The performance of SMOTE+Tomek Links can be sensitive to the selection of these parameters, such as the number of synthetic instances generated by SMOTE or the distance threshold used in identifying Tomek Links.
4. **Dependency on data distribution:** SMOTE+Tomek Links may not be equally effective for all types of imbalanced datasets. Its performance can depend on the specific characteristics and distribution of the data. It may not provide significant improvements in scenarios where the class imbalance is extreme or when the minority class is highly overlapping with the majority class.

It's important to note that the effectiveness of SMOTE+Tomek Links may vary depending on the dataset and the specific problem at hand. It is always recommended to experiment and evaluate different techniques to find the most suitable approach for a given scenario.

3.4.4 SMOTE+Cluster Centroids

Cluster Centroids:

Cluster centroids are representative points that summarize the characteristics of a cluster. In the context of clustering algorithms, such as k-means, cluster centroids are the mean or median values of the feature vectors within a cluster [51]. They serve as prototypes or central points that represent the underlying data distribution of each cluster.

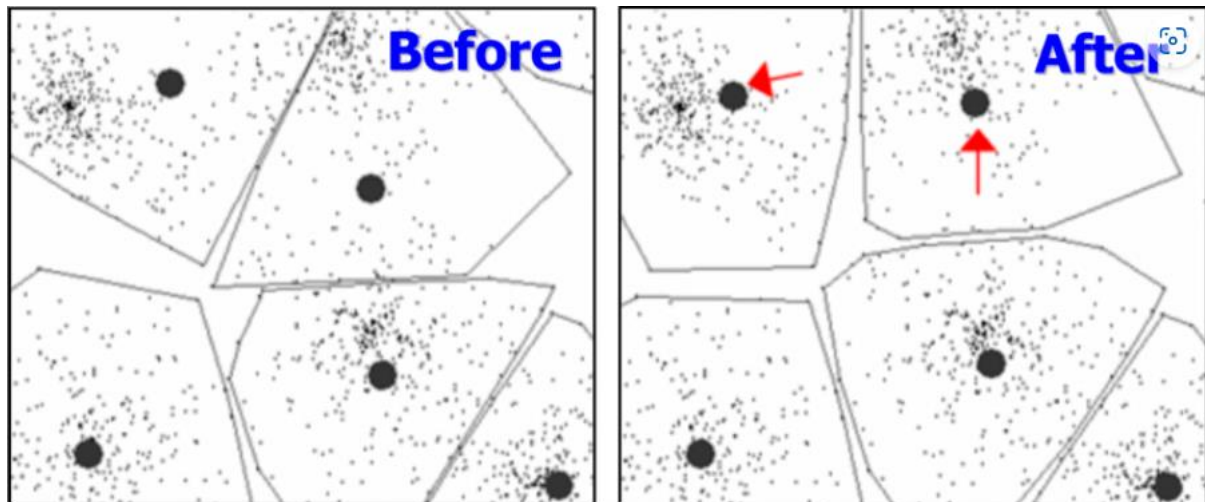


Figure 3.11: Example of centroid locations before (left) and after (right) one k-means iteration. K-means moves the centroids towards the actual cluster centers

Here's how cluster centroids are calculated:

1. Given a clustering algorithm (e.g., k-means), the algorithm assigns data points to clusters.
2. For each cluster, the cluster centroid is calculated as the mean or median of the feature vectors belonging to that cluster.
3. The resulting cluster centroids represent the central tendencies of their respective clusters.

SMOTE+Cluster Centroids:

SMOTE+Cluster Centroids is a combination of two techniques: SMOTE and cluster centroids. It is primarily used for addressing the class imbalance problem in machine learning.

Here's the process of using SMOTE+Cluster Centroids:

1. Apply cluster-based undersampling: Initially, the majority class is undersampled using a clustering algorithm (e.g., k-means). The algorithm clusters the majority class instances and selects cluster centroids as representative points.

- The equation below shows the selection of cluster centroids:

$$C = \{c_1, c_2, \dots, c_k\} = \text{ClusterCentroids}(D', N)$$

Where:

C - Set of cluster centroids.

D' - The majority class instances after undersampling using clustering.

N - The desired number of cluster centroids.

2. Apply SMOTE: Apply the SMOTE algorithm to oversample the minority class by generating synthetic instances. The synthetic instances are created by interpolating between minority class instances and their nearest neighbors.

- The equation below illustrates the SMOTE process:

$$S = \text{SMOTE}(D, N, K)$$

Where:

S - Set of synthetic instances.

D - The minority class instances.

N - The desired number of synthetic instances.

K - The number of nearest neighbors to consider during the interpolation.

3. Combine minority class with synthetic instances: Combine the original minority class instances with the synthetic instances generated by SMOTE.

$$D'' = D \cup S$$

Where:

D'' - The final dataset with both original minority class instances and synthetic instances.

Advantages of SMOTE+Cluster Centroids:

1. Effective handling of class imbalance: SMOTE+Cluster Centroids can address class imbalance by oversampling the minority class using SMOTE and undersampling the majority class using cluster centroids. This approach helps in balancing the class distribution and improves the performance of classifiers [53].
2. Reduced risk of overgeneralization: By generating synthetic instances based on the characteristics of the minority class and undersampling the majority class using cluster centroids, SMOTE+Cluster Centroids can reduce the risk of overgeneralization. It helps in preserving the underlying data structure of the minority class and avoids oversampling the noisy majority class instances.
3. Improved representation of the minority class: By combining the original minority class instances with synthetic instances, SMOTE+Cluster Centroids provides a more representative and diverse set of samples for the minority class. This can help classifiers to learn and generalize better on the minority class [52].

Disadvantages of SMOTE+Cluster Centroids:

1. **Dependency on clustering algorithm:** The effectiveness of SMOTE+Cluster Centroids heavily relies on the quality and accuracy of the clustering algorithm used to select the cluster centroids. If the clustering algorithm fails to capture the underlying data structure or if the clusters are not well-separated, it may impact the overall performance of SMOTE+Cluster Centroids.
2. **Computational complexity:** SMOTE+Cluster Centroids can be computationally expensive, especially when dealing with large datasets. It involves performing both clustering and oversampling steps, which may require significant computational resources and time.
3. **Sensitivity to parameter selection:** SMOTE+Cluster Centroids, like other machine learning techniques, requires appropriate parameter selection. Parameters such as the number of cluster centroids, the number of nearest neighbors for SMOTE, and the clustering algorithm itself need to be carefully chosen. Poor parameter selection can lead to suboptimal results.
4. **Sensitivity to dataset characteristics:** The performance of SMOTE+Cluster Centroids can vary depending on the specific characteristics of the dataset. If the minority class instances are not well-separated or if the clusters overlap significantly, it may be challenging for the technique to effectively address the class imbalance.
5. **Potential information loss:** The undersampling step using cluster centroids may result in the loss of some information from the majority class. This information loss can affect the classifier's ability to generalize accurately, particularly if important instances are removed.

3.4.5 SMOTE+ENN (Edited Nearest Network)

ENN (Edited Nearest Neighbors):

ENN (Edited Nearest Neighbors) is a data cleaning technique used for addressing the class imbalance problem in machine learning. It aims to remove noisy or misclassified instances from both the majority and minority classes [54]. ENN examines each instance and removes it if its class label does not match the majority of its nearest neighbors.

Here's how ENN works:

1. For each instance in the dataset, ENN identifies its k nearest neighbors based on a distance metric (e.g., Euclidean distance).
2. If the class label of the instance does not match the majority class label of its k nearest neighbors, the instance is considered noisy or misclassified and is removed from the dataset.

ENN helps in eliminating instances that are likely to contribute to misclassifications or increase the complexity of the decision boundary. By removing such instances, ENN can improve the performance of classifiers on imbalanced datasets.

SMOTE+ENN:

SMOTE+ENN is a combination of two techniques: SMOTE (Synthetic Minority Over-sampling Technique) and ENN (Edited Nearest Neighbors). It is used for addressing class imbalance in machine learning datasets.

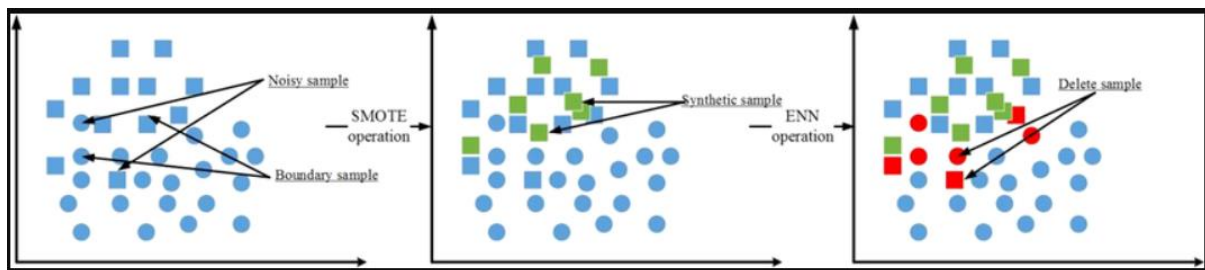


Figure 3.12: Samples simulation plot after the SMOTE-ENN sampled

Here's the process of using SMOTE+ENN [55]:

1. Apply SMOTE: Initially, SMOTE is applied to oversample the minority class by generating synthetic instances. SMOTE generates synthetic samples by interpolating between existing minority class instances and their nearest neighbors.

- The equation below illustrates the SMOTE process:

$$S = \text{SMOTE}(D, N, K)$$

Where:

S - Set of synthetic instances.

D - The minority class instances.

N - The desired number of synthetic instances.

K - The number of nearest neighbors to consider during interpolation.

2. Apply ENN: After applying SMOTE, the ENN technique is applied to remove noisy or misclassified instances from the dataset, considering both the majority and minority classes. ENN examines each instance and removes it if its class label does not match the majority of its nearest neighbors.

Advantages of SMOTE+ENN:

1. Improved classification performance: SMOTE+ENN can help improve the performance of classifiers on imbalanced datasets by addressing both oversampling and undersampling [56-58]. It generates synthetic instances for the minority class using SMOTE while removing noisy or misclassified instances using ENN.
2. Reduction of noise and irrelevant instances: By applying ENN after SMOTE, noisy or irrelevant instances that could potentially affect the classifier's performance are removed. This helps in reducing the impact of noisy data on the learning process and improves the quality of the dataset [59-60].
3. Preserving important instances: SMOTE+ENN aims to remove instances that are likely to contribute to misclassifications or increase the complexity of the decision boundary [61]. However, it takes care to preserve important instances by considering the majority of nearest neighbors during the removal process.

Disadvantages of SMOTE+ENN:

1. Potential information loss: ENN may remove valid instances that are misclassified or have minority class neighbors, but are still important for the classifier's decision

boundary. This can lead to information loss and potentially affect the performance of the classifier.

2. Sensitivity to parameter selection: SMOTE+ENN requires appropriate parameter selection, such as the number of synthetic instances to generate with SMOTE and the number of nearest neighbors to consider in ENN [62]. Choosing suboptimal parameters may impact the effectiveness of the technique.
3. Computational complexity: Applying both SMOTE and ENN can increase the computational complexity, particularly when dealing with large datasets. Generating synthetic instances and performing neighbor-based computations for ENN can be computationally [63].

CHAPTER 4

Results

After rigorous data handling and machine and deep learning model implementation we will see the results in this chapter. This section will give us the quantitative analysis of the performance matrices of our model on the data. The data is presented in three stages, before imbalance handling, after imbalance handling and after hyper-parameter tuning. The performance parameters for our study are accuracy, precision, F1 score and recall. Each of these performance parameters describes a different attribute of the model. To understand the interpret the results successfully the confusion matrix needs to be elaborated.

A confusion matrix represents the predictive performance of a model on a dataset [64]. For a binary class dataset (which consists of, suppose, “positive” and “negative” classes), a confusion matrix has four essential components:

True Positives (TP): Number of samples correctly predicted as “positive.”

False Positives (FP): Number of samples wrongly predicted as “positive.”

True Negatives (TN): Number of samples correctly predicted as “negative.”

False Negatives (FN): Number of samples wrongly predicted as “negative.”

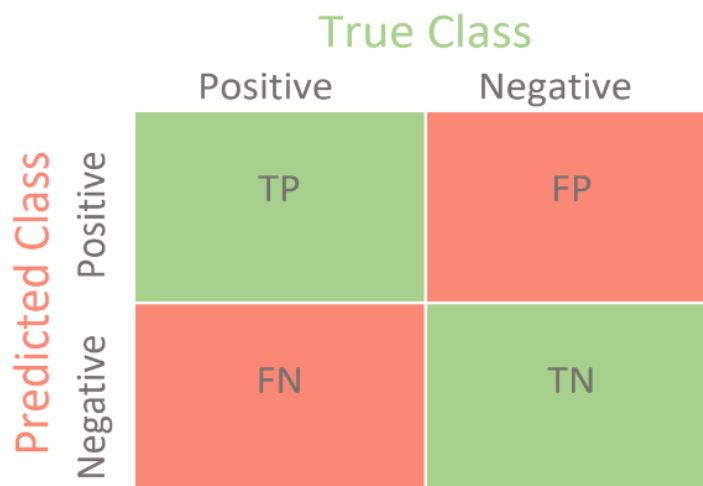


Figure 4.1: Structure of a confusion matrix

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right [65]. Formally, accuracy has the following definition:

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

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

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

Precision is another indicator of a model's performance. Precision is the proportion of positive identifications which is actually correct [66]. Precision is defined as:

$$Precision = \frac{TP}{TP + FP}$$

It is a measure of how well our model can predict correctly the positive identifications, in this case it would be the measure of how well the model can correctly predict patients who do have liver disease.

Recall, also known as the true positive rate (TPR), is the percentage of data samples that a machine learning model correctly identifies as belonging to a class of interest—the “positive class”—out of the total samples for that class. Recall tries to answer the question of what proportion of actual positives were identified correctly. Mathematically, recall is defined as follows:

$$Recall = \frac{TP}{TP + FN}$$

F1 score is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model. The accuracy metric computes how many times a model made a correct prediction across the entire dataset. This can be a reliable metric only if the dataset is class-balanced; that is, each class of the dataset has the same number of samples. In the real world, as well as in this case the dataset is class-imbalanced, often making this metric unviable. For example, if a binary class dataset has 90 and 10 samples in class-1 and class-2, respectively, a model that only predicts "class-1," regardless of the sample, will still be 90% accurate. Accuracy computes how many times a model made a correct prediction across the entire dataset. However, can this model be called a good predictor? This is where the F1 score comes into play.

Precision measures how many of the "positive" predictions made by the model were correct. Recall measures how many of the positive class samples present in the dataset were correctly identified by the model. Precision and recall offer a trade-off, i.e., one metric comes at the cost of another. More precision involves a harsher critic (classifier) that doubts even the actual positive samples from the dataset, thus reducing the recall score. On the other hand, more recall entails a lax critic that allows any sample that resembles a positive class to pass, which makes border-case negative samples classified as "positive," thus reducing the precision. Ideally, we want to maximize both precision and recall metrics to obtain the perfect classifier. The F1 score combines precision and recall using their harmonic mean, and maximizing the F1 score implies simultaneously maximizing both precision and recall. Thus, the F1 score has become the choice of researchers for evaluating their models in conjunction with accuracy [67].

The F1 score is defined based on the precision and recall scores, which are mathematically defined as follows:

$$F1\ Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

Therefore,

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

In terms of the basic four elements of the confusion matrix, by replacing the expressions for precision and recall scores in the equation above, the F1 score can also be written as follows:

$$F1\ Score = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

4.1 Before Imbalance Handling

After the data was pre-processed using the various method, machine and deep learning algorithms were applied. These algorithms were used before any kind of imbalance handling was performed. The results are given below:

ML Algorithm	Accuracy	Precision	Recall	F1 Score
Logistic Regression	.7094	.7115	.9487	.8132
SVM	.66666	.66666	1	.8
Random Forest	.66666	.696969	.8846	.77966
Naïve Bayes Gaussian	.5897	.9687	.3974	.5636
Naïve Bayes Multinomial	.6923	.7916	.7307	.76
Decision Tree	.6495	.7078	.8076	.75449
KNN	.6324	.6881	.8205	.7485
Gradient Boosting Classifier	.7008	.7263	.8846	.7976
XGboost Classification	.6666	.7142	.8333	.7692
ANN	.6752	.7272	.8205	.7710
Multi-layer Perceptron Classifier	.6666	.6695	.9871	.7979
LSTM	.6666	.6666	1	.8

Table 4.1 Performance of Machine learning and Deep learning models before Imbalance handling

From the table above, the noteworthy value comes from the SVM and LSTM, which gave a recall value of 1.

4.2 After Imbalance Handling

From the upper result although most of the values for accuracy, precision, recall and F1 scores are realistic, the Recall scores of 1 for SVM and LSTM cannot be overlooked. Finding a recall value of 1 for both LSTM and SVM models is not common and is generally unlikely, although it depends on the specific characteristics of the dataset and the problem being solved. While a recall value of 1 is desirable, it is more common to see values below 1, as achieving perfect recall typically indicates a well-behaved and separable dataset or potential overfitting. To prevent some of these problems, a number of imbalance handling techniques were used and the models were run again to get new results.

4.2.1 SMOTE:

ML Algorithm	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.69	0.86	0.64	0.73
SVM	0.68	0.90	0.58	0.71
Random Forest	0.65	0.72	0.78	0.75
Naïve Bayes Gaussian	0.59	0.96	0.41	0.57
Naïve Bayes Multinomial	0.61	1	0.42	0.59
Decision Tree	0.63	0.72	0.71	0.72
KNN	0.64	0.77	0.65	0.71

Gradient Boosting Classifier	0.69	0.77	0.77	0.77
XGboost Classification	0.70	0.76	0.79	0.78
ANN	0.65	0.73	0.74	0.74
Multi-layer Perceptron Classifier	0.67	0.85	0.61	0.71
LSTM	0.53	0.83	0.38	0.52

Table 4.2: Results after SMOTE applied to dataset

After SMOTE was used for imbalance handling the best result for Accuracy came out to be 0.7 from XGBoost Classification. Best result for precision came from Naïve Bayes Multinomial with a value of 1, Best recall score was from XGBoost classifier with a value of 0.79 and best F1 score also came from XGBoost classifier with a score of 0.78

4.2.2 ADASYN:

ML Algorithm	Accuracy	Precision	Recall	F1 Score
Logistic Regression	.6495	.8627	.5641	.6821
SVM	.6495	.9111	.5256	.6666
Random Forest	.6923	.7560	.7948	.7749
Naïve Bayes Gaussian	.5897	.9687	.3974	.5636
Naïve Bayes Multinomial	.5726	1	.3589	.5283
Decision Tree	.6923	.7625	.7820	.7721
KNN	.6666	.8095	.6538	.7234

Gradient Boosting Classifier	.6923	.7837	.7435	.7631
XGboost Classification	.6923	.7386	.8333	.7831
ANN	.6581	.7317	.7692	.7499
Multi-layer Perceptron Classifier	.6324	.8301	.5641	.6717
LSTM	.4957	.9130	.2692	.4158

Table 4.3: Result after ADASYN applied to dataset

The best result for accuracy came from a number of algorithms. Random forest, Decision tree, Gradient Boosting Classifier and XGBoost classifier all gave a value of 0.6923. A precision value of 1 was found from Naïve Bayes Multinomial which was the highest. Maximum value of recall was found from XGBoost classification which came out to be 0.8333 and maximum F1 score was found from XGBoost classification with a score of 0.7831.

4.2.3 SMOTE + CC

ML Algorithm	Accuracy	Precision	Recall	F1 Score
Logistic Regression	.6581	.8518	.5897	.6969
SVM	.6752	.9	.5769	.7031
Random Forest	.6923	.7692	.7692	.7692
Naïve Bayes Gaussian	.5982	.9696	.4102	.5765
Naïve Bayes Multinomial	.6068	1	.4102	.5818

Decision Tree	.6837	.7469	.7948	.7701
KNN	.6581	.7968	.6538	.7183
Gradient Boosting Classifier	.7179	.7922	.7820	.7870
XGboost Classification	.7008	.7654	.7948	.7798
ANN	.6410	.6836	.8589	.7613
Multi-layer Perceptron Classifier	.6495	.8363	.5897	.6917
LSTM	.5042	.8571	.3076	.4528

Table 4.4: Result after SMOTE + CC applied to dataset

The best accuracy when using SMOTE + CC as the imbalance handling technique came from Gradient Boosting Classifier with a value of 0.7179. The best result for precision came from Naïve Bayes Multinomial with a value of 1. The best result for Recall came from ANN with a value of 0.8589 and the best result of F1 score was 0.7870 which came from XGBoost Classification.

4.2.4 SMOTE + TOMKLINKS

ML Algorithm	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.67	0.85	0.61	0.71
SVM	0.66	0.89	0.56	0.69
Random Forest	0.68	0.73	0.82	0.77
Naïve Bayes Gaussian	0.59	0.96	0.41	0.57
Naïve Bayes Multinomial	0.62	1	0.43	0.60
Decision Tree	0.69	0.76	0.77	0.77
KNN	0.64	0.77	0.65	0.70
Gradient Boosting Classifier	0.68	0.76	0.75	0.76
XGboost Classification	0.66	0.71	0.82	0.76
ANN	0.68	0.76	0.77	0.76
Multi-layer Perceptron Classifier	0.65	0.85	0.58	0.69
LSTM	0.52	0.84	0.34	0.49

Table 4.5: Result after SMOTE + TOMKLINKS applied to dataset

Next is the SMOTE + TOMKLINKS method. The best result for accuracy came from decision tree with a value of 0.69. Best precision value came from Naïve Bayes Multinomial which gave a value of 1. Best Recall value came from Random forest and XGBoost

classification which gave a value of 0.82. And the best F1 score came from Decision tree which gave a value of 0.77

4.2.5 SMOTE + ENN

ML Algorithm	Accuracy	Precision	Recall	F1 Score
Logistic Regression	.6495	.9111	.5256	.6666
SVM	.6581	.9318	.5256	.6721
Random Forest	.6837	.8059	0.6923	.7448
Naïve Bayes Gaussian	.6239	.9722	.4487	.6140
Naïve Bayes Multinomial	.6068	1	.4102	.5818
Decision Tree	.5897	.7777	.5384	.6363
KNN	.5726	.7916	.4871	.6031
Gradient Boosting Classifier	.6752	.9	.5769	.7031
XGboost Classification	.7179	.9090	.6410	.7518
ANN	.6153	.7538	.6282	.6853
Multi-layer Perceptron Classifier	.6495	.9302	.5128	.6611
LSTM	.4700	.8636	.2435	.3800

Table 4.6: Result after SMOTE + ENN applied to dataset

The best accuracy when SMOTE + ENN was used came from XGBoost Classification which gave a value of 0.7179. The best value of precision came from the Naïve Bayes Multinomial which gave a value of 1. The best value of Recall came from Random Forest with a value of 0.6923. And the best value of F1 score came from the XGBoost Classification with a value of 0.7518.

From the above tables we can see the results of the machine and deep learning algorithms once the different imbalance handling method has been used. Below is the summary of the best performance which has been found for each of the imbalance handling method used.

4.3 Hyperparameter Tuning

Hyperparameter tuning was performed after the results of the newly balanced data was found. The results of all the algorithms are given below. Hyperparameter tuning is an important step to prevent overfitting or underfitting. Two hyperparameter tuning techniques have been applied, these are, `gridsearchcv` and `randomizedsearchcv`. We can verify that the model is optimal for the particular job and gives the best potential outcomes by picking the optimum collection of hyperparameters [68].

`GridSearchCV` [69] is guaranteed to identify the optimum hyperparameter combination inside the search space, but it can be computationally costly, particularly when working with a large number of hyperparameters. `RandomizedSearchCV`, on the other hand, is less computationally expensive and can still identify appropriate hyperparameters by randomly searching the search space. There is no assurance, however, that it will locate the optimal hyperparameters. In the testing done, this holds true as `GridSearchCV` has better results than `RandomizedSearchCV` for every ML algorithm used.

Below are the results after hyper-parameter tuning for each of the imbalance handling method used:

4.3.1 SMOTE:

For GridsearchCV

model	score	bs_5folds	bp_5	br_5	bf_5	bs_10folds	bp_10	br_10	bf_10
svm	0.713559	0.835871	0.904298	0.760492	0.823387	0.829939	0.899323	0.757398	0.816492
randomforest	0.686148	0.846329	0.893517	0.805048	0.842321	0.837357	0.893435	0.780660	0.827881
naive_bayes_gaussian	0.557335	0.677582	0.910183	0.396839	0.548818	0.678995	0.891618	0.411230	0.559376
naive_bayes_multinomial	0.557335	0.668693	0.695839	0.606760	0.647084	0.662555	0.688282	0.603298	0.640356
decision_tree	0.637931	0.739837	0.752377	0.722037	0.736376	0.741198	0.739659	0.757576	0.745755
knn	0.651578	0.705708	0.825403	0.535777	0.643436	0.721905	0.845094	0.553476	0.660035
logistic_regression	0.713647	0.714586	0.797346	0.577349	0.667211	0.714442	0.792545	0.582888	0.668610
Gradient_Boosting_Classifier	0.708445	0.772342	0.816032	0.731255	0.764077	0.764969	0.813438	0.718627	0.755124
XGboost Classification	0.696318	0.821187	0.862865	0.787270	0.818429	0.834636	0.881711	0.798574	0.831020

Table 4.7: Performance Comparison of Machine Learning Algorithms with SMOTE and GridSearchCV

For RandomizedSearchCV

model	score	bs_5folds	bp_5	br_5	bf_5	bs_10folds	bp_10	br_10	bf_10
svm	0.713559	0.794499	0.849346	0.737006	0.783318	0.720325	0.881398	0.508913	0.642157
random_forest	0.686148	0.812266	0.844560	0.787094	0.810399	0.828468	0.895027	0.757041	0.816382
naive_bayes_gaussian	0.557335	0.676100	0.910183	0.393898	0.545422	0.678995	0.891618	0.411230	0.559376
naive_bayes_multinomial	0.557335	0.668693	0.695839	0.606760	0.647084	0.662555	0.688282	0.603298	0.640356
decisionjree	0.637931	0.756024	0.786804	0.704653	0.739958	0.739794	0.767833	0.718449	0.734521
knn	0.651578	0.705708	0.825403	0.535777	0.643436	0.721905	0.845094	0.553476	0.660035
logistic_regression	0.713647	0.716068	0.797895	0.580334	0.669355	0.712972	0.790017	0.582888	0.667518
Gradient_Boosting_Classifier	0.708445	0.772342	0.823537	0.728270	0.763792	0.763411	0.808490	0.715686	0.752190
XGboost Classification	0.696318	0.821187	0.862865	0.787270	0.818429	0.834636	0.881711	0.798574	0.831020

Table 4.8: Performance Comparison of Machine Learning Algorithms with SMOTE and RandomizedsearchCV

4.3.2 ADASYN

For GridsearchCV

model	score	bs_5folds	bp_5	br_5	bf_5	bs_10folds	bp_10	br_10	bf_10
svm	0.713559	0.821301	0.895391	0.718876	0.796683	0.822774	0.887358	0.736809	0.802369
random_forest	0.705026	0.818403	0.866255	0.748244	0.800958	0.815714	0.866483	0.743048	0.796220
naive_bayes_gaussian	0.557335	0.675738	0.865599	0.393108	0.539442	0.677246	0.876053	0.393939	0.540561
naive_bayes_multinomial	0.557335	0.687943	0.860835	0.414442	0.558047	0.686338	0.869428	0.405615	0.547132
decision_tree	0.648130	0.750355	0.748575	0.727788	0.737132	0.756177	0.766633	0.725134	0.741282
knn	0.651578	0.688687	0.821686	0.461194	0.588809	0.693023	0.810867	0.482353	0.601815
logistic_regression	0.713647	0.713210	0.811871	0.532046	0.641199	0.707578	0.814408	0.527184	0.633167
Gradient_Boosting_Classifier	0.694652	0.768012	0.812845	0.694996	0.745029	0.775424	0.826030	0.707487	0.754985
XGboost_Classification	0.696318	0.822771	0.863219	0.766023	0.809079	0.823106	0.860143	0.778253	0.811758

Table 4.9: Performance Comparison of Machine Learning Algorithms with ADASYN and GridSearchCV

For RandomizedSearchCV

model	score	bf_3_bs_5folds	bp_5	br5	bf_5	bs_10folds	bp_10	br_10	bf_10
svm	0.713559	0.808320	0.866280	0.718788	0.784916	0.822774	0.887358	0.736809	0.802369
random_forest	0.705026	0.812658	0.851797	0.745127	0.793364	0.818592	0.860356	0.760873	0.802421
naive_bayes_gaussian	0.557335	0.675738	0.865599	0.393108	0.539442	0.674389	0.865797	0.393939	0.538824
naive_bayes_multinomial	0.557335	0.687943	0.860835	0.414442	0.558047	0.686338	0.869428	0.405615	0.547132
decision_tree	0.641379	0.733448	0.749645	0.692098	0.717188	0.746605	0.767342	0.701693	0.728082
knn	0.651578	0.688687	0.821686	0.461194	0.588809	0.693023	0.810867	0.482353	0.601815
logistic_regression	0.713647	0.711761	0.807603	0.532046	0.640100	0.707536	0.808927	0.533066	0.635508
Gradient_Boosting_Classifier	0.694652	0.766573	0.810729	0.694996	0.744012	0.773996	0.824288	0.707487	0.753932
XGboost_Classification	0.696318	0.822771	0.863219	0.766023	0.809079	0.818841	0.858486	0.775401	0.807579

Table 4.10: Performance Comparison of Machine Learning Algorithms with ADASYN and RandomizedSearchCV

4.3.3 SMOTE+CC:

For GridsearchCV

model	score	bs_5folds	bp_5	br5	bf5	bs_10folds	bp_10	br_10	bf_10
svm	0.713559	0.838780	0.906384	0.772388	0.824540	0.834219	0.897221	0.768717	0.819409
random_forest	0.696493	0.818007	0.890472	0.751493	0.801814	0.840255	0.889935	0.798752	0.832341
naive_bayes_gaussian	0.557335	0.683279	0.876442	0.411633	0.535237	0.684372	0.890115	0.410784	0.536760
naive_bayes_multinomial	0.557335	0.656939	0.715502	0.583231	0.624845	0.668437	0.721199	0.561497	0.622936
decision_tree	0.655172	0.724891	0.747472	0.704478	0.717748	0.73378	0.752811	0.71016	0.72686
knn	0.651578	0.714455	0.854361	0.532748	0.644032	0.732221	0.877869	0.550000	0.662553
logistic_regression	0.713647	0.708486	0.806906	0.550658	0.642164	0.718591	0.819939	0.564795	0.656939
Gradient_Boosting_Classifier	0.699825	0.769303	0.828004	0.725066	0.755572	0.773617	0.826723	0.724688	0.760038
XGboost_Classification	0.696318	0.815087	0.870540	0.775066	0.806250	0.840277	0.897366	0.792959	0.829503

Table 4.11: Performance Comparison of Machine Learning Algorithms with SMOTE+CC and GridSearchCV

For RandomizedSearchCV

model	score	_bs_5folds	bp_5	br5	bf_5	bs_10folds	bp_10	br_10	bf_10
svm	0.713559	0.801765	0.890097	0.710623	0.772573	0.791352	0.888007	0.682977	0.756139
random_forest	0.696493	0.794434	0.857660	0.736918	0.779314	0.837270	0.892631	0.789929	0.827490
naive_bayes_gaussian	0.557335	0.683279	0.876442	0.411633	0.535237	0.684372	0.890115	0.410784	0.536760
naive_bayes_multinomial	0.557335	0.656939	0.715502	0.583231	0.624845	0.668437	0.721199	0.561497	0.622936
decision_tree	0.655172	0.714455	0.817018	0.559350	0.655776	0.750154	0.765869	0.745544	0.746345
knn	0.651578	0.714455	0.854361	0.532748	0.644032	0.732221	0.877869	0.550000	0.662553
logistic_regression	0.713647	0.708486	0.814329	0.541791	0.637835	0.718591	0.819939	0.564795	0.656939
Gradient_Boosting_Classifier	0.699825	0.769292	0.816893	0.733977	0.757135	0.772103	0.825266	0.718806	0.757153
XGhoost_Classification	0.696318	0.813627	0.873582	0.763521	0.800658	0.826975	0.886282	0.778164	0.814627

Table 4.12: Performance Comparison of Machine Learning Algorithms with SMOTE+CC and RandomizedSearchCV

4.3.4 SMOTE+TOMEKLINKS

For GridsearchCV

model	score	bs_5folds	bp_5	br_5	bf_5	bs_10folds	bp_10	br_10	bf_10
svm	0.713559	0.858261	0.932261	0.775577	0.845132	0.856779	0.939834	0.766288	0.842016
random_forest	0.689451	0.847505	0.896921	0.791154	0.838894	0.849159	0.915755	0.775758	0.836774
naive_bayes_gaussian	0.557335	0.688542	0.907442	0.420481	0.572077	0.688678	0.905137	0.423958	0.574842
naive_bayes_multinomial	0.557335	0.671875	0.695806	0.615625	0.652678	0.665625	0.678261	0.634375	0.654741
decision_tree	0.636119	0.745313	0.767406	0.715625	0.737535	0.765625	0.808872	0.715625	0.754259
knn	0.651578	0.721306	0.847792	0.538990	0.656082	0.730601	0.857379	0.554735	0.670826
logistic_regression	0.713647	0.746136	0.870370	0.582308	0.694173	0.749471	0.861670	0.598390	0.701603
Gradient_Boosting_Classifier	0.699825	0.794598	0.848478	0.741346	0.785697	0.782356	0.834799	0.729261	0.773583
XGboost_Classification	0.696318	0.828864	0.878634	0.778846	0.821950	0.835168	0.887262	0.778883	0.826166

Table 4.13: Performance Comparison of Machine Learning Algorithms with SMOTE+TOMEKLINKS and GridSearchCV

For RandomizedSearchCV

model	score	bs_5folds	bp_5	br_5	bf_5	bs_10folds	bp_10	br_10	bf_10
svm	0.713559	0.856698	0.928315	0.775481	0.843339	0.805385	0.868550	0.726042	0.787716
random_forest	0.689451	0.828791	0.865551	0.787981	0.821737	0.818053	0.859057	0.769602	0.809353
naive_bayes_gaussian	0.557335	0.685405	0.911425	0.411106	0.564582	0.688678	0.905137	0.423958	0.574842
naive_bayes_multinomial	0.557335	0.671875	0.695806	0.615625	0.652678	0.665625	0.678261	0.634375	0.654741
decisionjree	0.636119	0.736822	0.769635	0.694375	0.723256	0.726034	0.737926	0.710511	0.722520
knn	0.651578	0.721306	0.847792	0.538990	0.656082	0.730601	0.857379	0.554735	0.670826
logistic_regression	0.713647	0.744598	0.863373	0.585433	0.694429	0.747909	0.858409	0.598390	0.700434
Gradient_Boosting_Classifier	0.699825	0.797699	0.852645	0.741346	0.787919	0.782404	0.833422	0.729451	0.773119
XGboost_Classification	0.696318	0.828864	0.878634	0.778846	0.821950	0.833582	0.880252	0.785038	0.825697

Table 4.14: Performance Comparison of Machine Learning Algorithms with SMOTE+TOMEKLINKS and RandomizedSearchCV

4.3.5 SMOTE + ENN

For GridsearchCV

model	score	bs_5folds	bp_5	br5	bf_5	bs10folds	bp_10	br_10	bf_10
svm	0.713559	0.983710	0.971903	0.984615	0.977474	0.983709	0.972381	0.984615	0.977450
random_forest	0.706604	0.932136	0.959304	0.846154	0.898719	0.943168	0.953022	0.884615	0.915110
naive_bayes_gaussian	0.557335	0.875194	0.985714	0.653846	0.782880	0.877778	0.938495	0.700000	0.798434
naive_bayes_multinomial	0.557335	0.739620	0.960000	0.312000	0.459300	0.740285	0.900000	0.314744	0.448673
decision_tree	0.634424	0.891485	0.895872	0.792308	0.838724	0.899700	0.905866	0.807692	0.850516
knn	0.651578	0.910441	0.981781	0.761538	0.855556	0.915841	0.992308	0.769231	0.861459
logistic_regression	0.713647	0.918586	0.963333	0.800000	0.873302	0.921246	0.972106	0.800000	0.874723
Gradient_Boosting_Classifier	0.699825	0.915957	0.936000	0.815385	0.870523	0.929595	0.964126	0.830769	0.890815
XGboost_Classification	0.696318	0.921362	0.947636	0.823077	0.880204	0.943093	0.968290	0.869231	0.912795

Table 4.15: Performance Comparison of Machine Learning Algorithms with SMOTE+ENN and GridSearchCV

For RandomizedSearchCV

model	score	bs_5folds	bp_5	br_5	bf_5	bs_10	bp_10	br_10	bf_10
						folds			
svm	0.713559	0.98371	0.971903	0.984615	0.977474	0.983709	0.972381	0.984615	0.97745
random_forest	0.706604	0.924028	0.943333	0.838462	0.887285	0.926802	0.954242	0.838462	0.889584
naive_bayes_gaussian	0.557335	0.875194	0.985714	0.653846	0.78288	0.877778	0.938495	0.7	0.798434
naive_bayes_multinomial	0.557335	0.73962	0.96	0.312	0.4593	0.740285	0.9	0.314744	0.448673
decision_tree	0.634424	0.864347	0.805038	0.830769	0.814654	0.902402	0.914957	0.807692	0.854139
knn	0.651578	0.910441	0.981781	0.761538	0.855556	0.915841	0.992308	0.769231	0.861459
logistic_regression	0.713647	0.918586	0.963333	0.8	0.873302	0.921246	0.972106	0.8	0.874723
Gradient_Boosting_Classifier	0.699825	0.915920	0.937905	0.815385	0.871652	0.929505	0.964126	0.830769	0.890815
XGboost Classification	0.696318	0.921362	0.947636	0.823077	0.880204	0.943093	0.96829	0.869231	0.912795

Table 4.16: Performance Comparison of Machine Learning Algorithms with SMOTE+ENN and RandomizedSearchCV

CHAPTER 5

Discussion

Before any imbalance was handled the data gave us results which had some anomalies. For example, before handling imbalance the Recall score of SVM came out to be 1. This means that the model has achieved perfect recall on the given dataset. Recall, also known as sensitivity or true positive rate, is a performance metric that measures the proportion of actual positive instances correctly identified by the model.

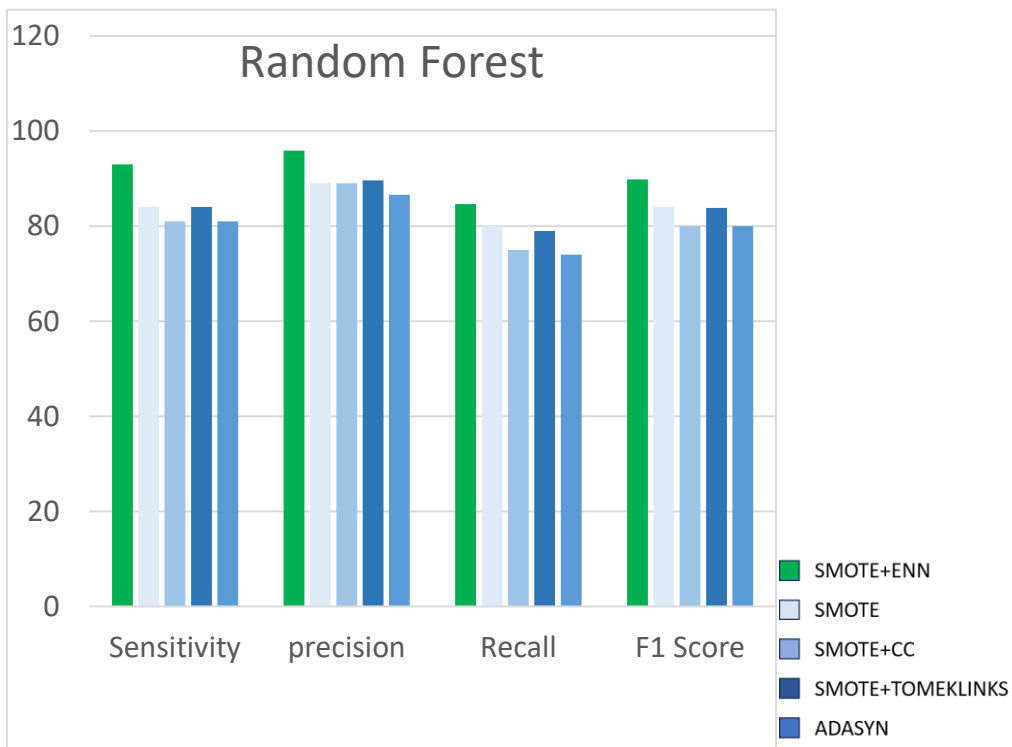
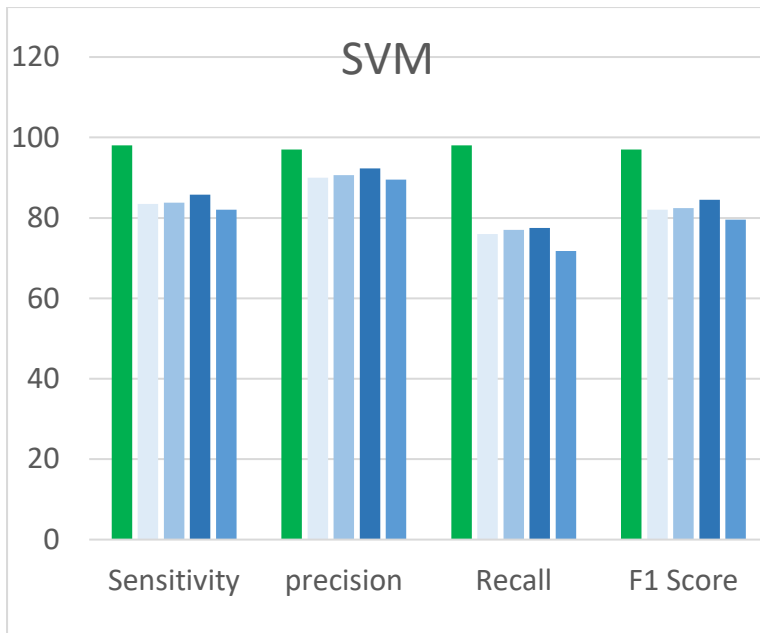
When recall = 1, it indicates that the SVM has correctly identified all positive instances in the dataset without any false negatives. In other words, the model did not miss any positive instances and identified all of them correctly. This is the ideal scenario for recall, as it signifies that the model has achieved the highest level of sensitivity in detecting positive instances.

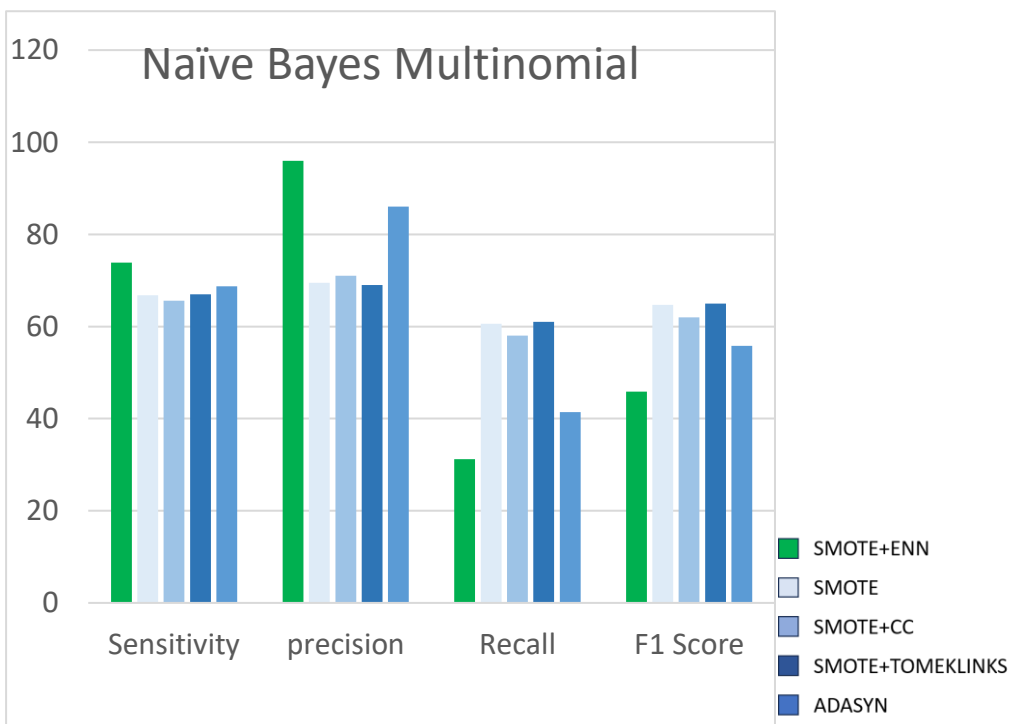
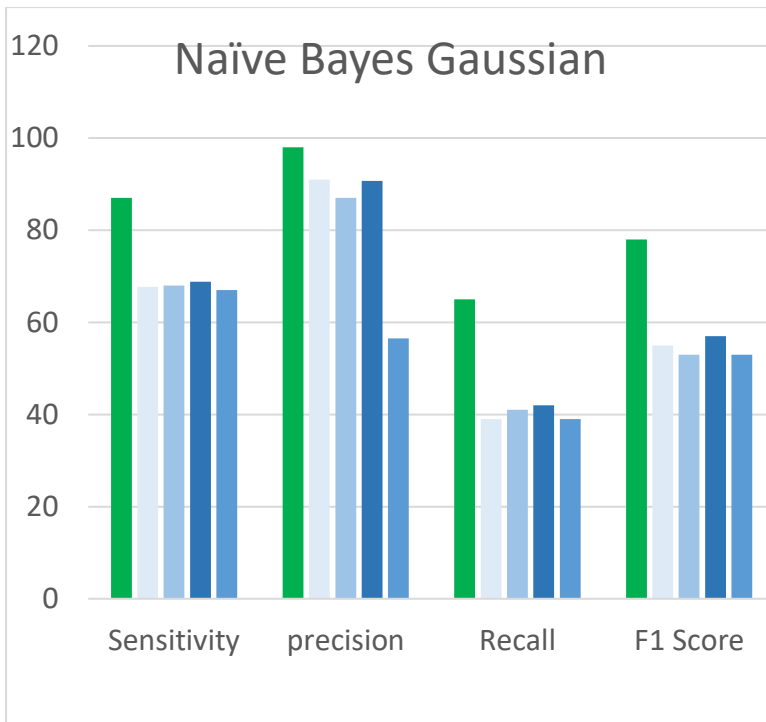
However, it is highly unlikely that the model can correctly identify all the positive instances. It is more likely that the data being used has some imbalance which needs to be taken care of.

After the data has been handled for imbalance, the result is tabulated for every imbalance handling method used. Another unlikely outcome was observed after handling the imbalance. For every imbalance handling method used, the precision of Naïve Bayes Multinomial for each imbalance handling method used came to be 1. This means all instances predicted as positive by the model are indeed true positives, and there are no instances falsely classified as positive. This is also unlikely that perfect precision can be found. Hyperparameter tuning could be a good way to solve this anomaly.

And it is seen that after hyperparameter tuning has been done, the precision value of Naïve Bayes Multinomial becomes a more realistic value. The results after performing hyperparameter tuning with 5-fold and 10-fold cross validation has been tabulated above in the results chapter.

Below is a comparison of the machine learning model performance for every imbalance handling method used. The graphs have the accuracy, precision, recall and F1 scores for every machine learning algorithm used. The analysis utilized the results obtained from GridSearchCV, employing a 10-fold cross-validation technique exclusively.





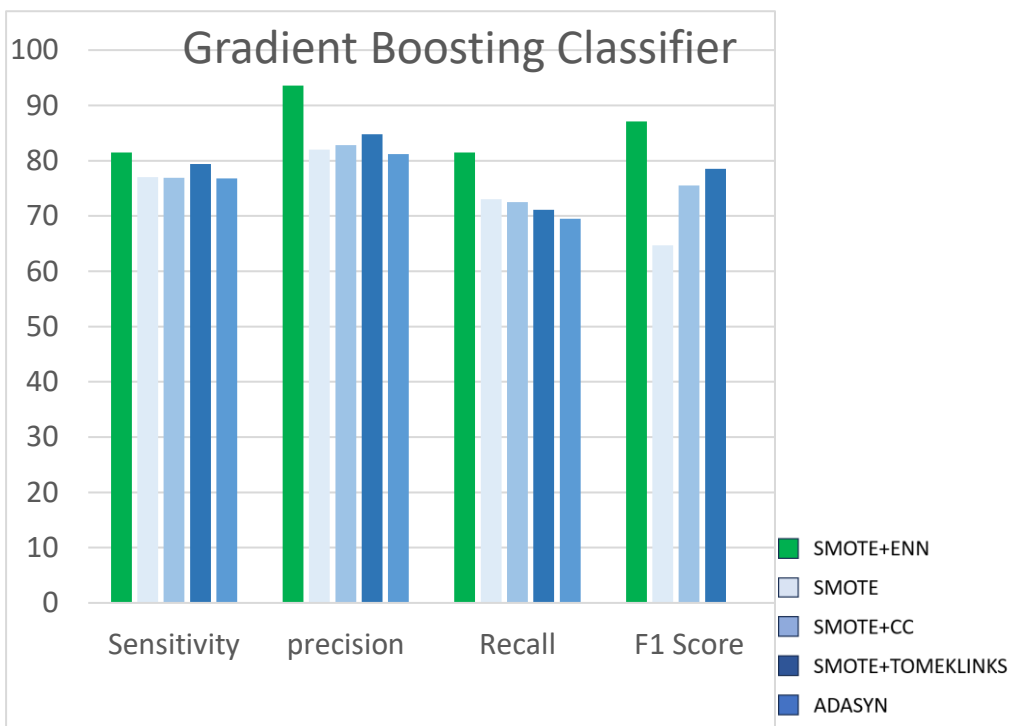
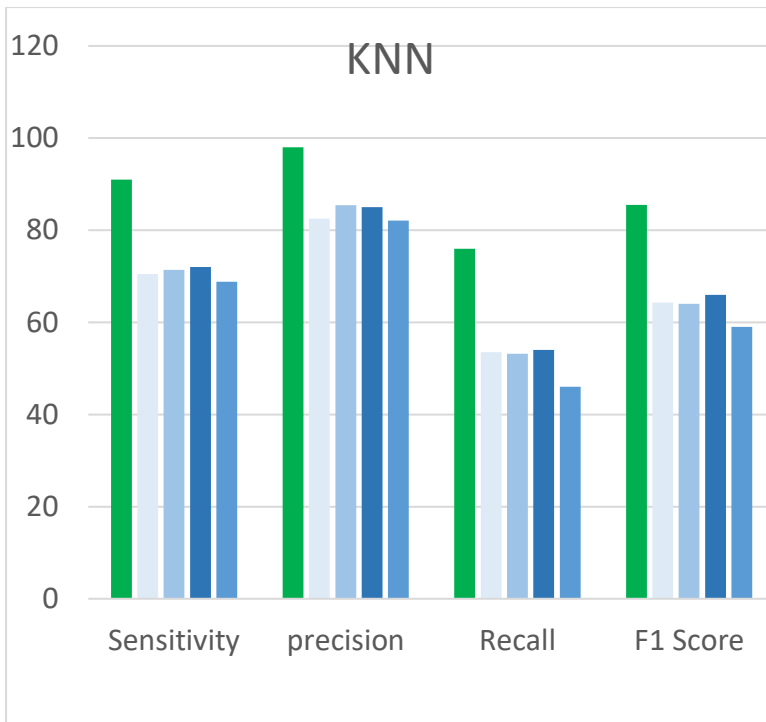




Figure 4.2: Comparative figures of all the performance parameters (accuracy, precision, recall and f1 score) for each machine learning algorithm with each imbalance handling technique applied.

From the above comparison we can see that besides the recall and F1 score of Naïve Bayes Multinomial, SMOTE+ENN gives the best result across all the machine learning algorithms used for all the performance parameters.

Best working algorithm (highest score): SVM with a score of 0.713559.

For the evaluation metrics:

The best accuracy was found from SVM with a value of 0.983710 (5-fold) and 0.983709 (10-fold). Best precision Naive Bayes (Gaussian) with a value of 0.985714 (5-fold) and KNN with a value of 0.992308 (10-fold). The best recall was found using SVM with a value of 0.984615 (5-fold and 10-fold). Best F1 score using SVM with a value of 0.977474 (5-fold) and 0.977450 (10-fold). Therefore, the best working algorithm overall is SVM, and it also achieved the highest accuracy, recall, and F1 score. Naive Bayes (Gaussian) and KNN performed the best in terms of precision.

CHAPTER 6

Limitations

The dataset used in the study consists of only 583 samples, which may be considered relatively small for training a complex machine learning model. The limited number of instances could potentially affect the model's ability to capture the full range of patterns and variations present in liver disease data. The dataset comprises data solely from Indian patients, which introduces a potential limitation in terms of generalizability. The characteristics and risk factors associated with liver disease may differ among populations worldwide. Therefore, the findings and predictions made by the model may not fully reflect the nuances and diversity of liver disease in a global context. Since the dataset exclusively focuses on Indian patients, the model's predictions might be biased towards the specific characteristics and health conditions prevalent in the Indian population. The model may not account for the unique factors or variations present in other populations, potentially limiting its applicability and generalizability to individuals from different ethnic backgrounds. The dataset used in the study may lack certain important variables that could contribute to a more comprehensive understanding of liver disease. Absence of relevant features, such as genetic factors, lifestyle habits, or other underlying health conditions, could limit the model's predictive accuracy and overall performance.

CHAPTER 7

CONCLUSION AND FUTURE WORKS

In this fast-growing world, in terms of technology and population, there is a grave requirement to use all the resources at hand to provide swift and proper service to the ever-growing population. In the field of medicine, the use of modern technology like Artificial Intelligence has become a crucial way to provide accurate and reliable prediction of different diseases so that treatment can be started as soon as possible. Liver disease is one of the diseases which needs to be identified very early on as late diagnosis may result in treatment beginning too late. In this study, 5 different Imbalance handling techniques have been used on a liver disease dataset after the dataset had been pre-processed. Among all the imbalance handling methods used, it is seen that SVM, applied to the SMOTE+ENN dataset, gave us the best results. The best accuracy, recall and F1 score was found using SVM all of which gave a value of 98.37%, 98.46% and 97.74% respectively. The best precision came from Naive Bayes (Gaussian) with a value 98.57% (5-fold) and KNN with a value of 99.23% (10-fold). The ROC_AUC score for SVM also came at 98% for 5-fold cross validation. Therefore, SMOTE+ENN is best at handling this kind of dataset. With the promising result found in this study, a notable contribution can be made in the prediction of liver disease and an automated system can be put in place which allows computer-aided diagnosis system for e-healthcare applications.

References

1. Choudhary, R., Gopalakrishnan, T., Ruby, D., Gayathri, A., Murthy, V.S. and Shekhar, R., 2021. An Efficient Model for Predicting Liver Disease Using Machine Learning. *Data Analytics in Bioinformatics: A Machine Learning Perspective*, pp.443-457.
2. Dritsas, E. and Trigka, M., 2023. Supervised Machine Learning Models for Liver Disease Risk Prediction. *Computers*, 12(1), p.19.
3. Tokala, S., Hajarathaiyah, K., Gunda, S.R.P., Botla, S., Nalluri, L., Nagamanohar, P., Anamalamudi, S. and Enduri, M.K., 2023. Liver Disease Prediction and Classification using Machine Learning Techniques. *International Journal of Advanced Computer Science and Applications*, 14(2).
4. Weng, S., Hu, D., Chen, J., Yang, Y. and Peng, D., 2023. Prediction of Fatty Liver Disease in a Chinese Population Using Machine-Learning Algorithms. *Diagnostics*, 13(6), p.1168.
5. Peng, H.Y., Duan, S.J., Pan, L., Wang, M.Y., Chen, J.L., Wang, Y.C. and Yao, S.K., 2023. Development and validation of machine learning models for nonalcoholic fatty liver disease. *Hepatobiliary & Pancreatic Diseases International*.
6. Kavakiotis, I., Tsave, O., Salifoglou, A., Maglaveras, N., Vlahavas, I. and Chouvarda, I., 2017. Machine learning and data mining methods in diabetes research. *Computational and structural biotechnology journal*, 15, pp.104-116.
7. Ramana, B.V., Babu, M.P. and Venkateswarlu, N.B., 2012. Liver classification using modified rotation forest. *International Journal of Engineering Research and Development*, 6(1), pp.17-24.
8. Pei, X., Deng, Q., Liu, Z., Yan, X. and Sun, W., 2021. Machine learning algorithms for predicting fatty liver disease. *Annals of Nutrition and Metabolism*, 77(1), pp.38-45.
9. Xiao, J., Wang, F., Wong, N.K., He, J., Zhang, R., Sun, R., Xu, Y., Liu, Y., Li, W., Koike, K. and He, W., 2019. Global liver disease burdens and research trends: analysis from a Chinese perspective. *Journal of hepatology*, 71(1), pp.212-221.
10. Khan, M.A.R., Afrin, F., Prity, F.S., Ahammad, I., Fatema, S., Prosad, R., Hasan, M.K. and Uddin, M., 2023. An effective approach for early liver disease prediction and sensitivity analysis. *Iran Journal of Computer Science*, pp.1-19.
11. Alber, M., Buganza Tepole, A., Cannon, W.R., De, S., Dura-Bernal, S., Garikipati, K., Karniadakis, G., Lytton, W.W., Perdikaris, P., Petzold, L. and Kuhl, E., 2019. Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences. *NPJ digital medicine*, 2(1), p.115.
12. Kolachalama, V.B. and Garg, P.S., 2018. Machine learning and medical education. *NPJ digital medicine*, 1(1), p.54.
13. Dritsas, E. and Trigka, M., 2023. Supervised Machine Learning Models for Liver Disease Risk Prediction. *Computers*, 12(1), p.19.

14. Weng, S., Hu, D., Chen, J., Yang, Y. and Peng, D., 2023. Prediction of Fatty Liver Disease in a Chinese Population Using Machine-Learning Algorithms. *Diagnostics*, 13(6), p.1168.
15. Tokala, S., Hajarathaiyah, K., Gunda, S.R.P., Botla, S., Nalluri, L., Nagamanohar, P., Anamalamudi, S. and Enduri, M.K., 2023. Liver Disease Prediction and Classification using Machine Learning Techniques. *International Journal of Advanced Computer Science and Applications*, 14(2).
16. Peng, H.Y., Duan, S.J., Pan, L., Wang, M.Y., Chen, J.L., Wang, Y.C. and Yao, S.K., 2023. Development and validation of machine learning models for nonalcoholic fatty liver disease. *Hepatobiliary & Pancreatic Diseases International*.
17. Gupta, K., Jiwani, N., Afreen, N. and Divyarani, D., 2022, April. Liver Disease Prediction using Machine learning Classification Techniques. In *2022 IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT)* (pp. 221-226). IEEE.
18. Ramana, B.V., Babu, M.S.P. and Venkateswarlu, N.B., 2011. A critical study of selected classification algorithms for liver disease diagnosis. *International Journal of Database Management Systems*, 3(2), pp.101-114.
19. Pei, X., Deng, Q., Liu, Z., Yan, X. and Sun, W., 2021. Machine learning algorithms for predicting fatty liver disease. *Annals of Nutrition and Metabolism*, 77(1), pp.38-45.
20. Sivasangari, A., Reddy, B.J.K., Kiran, A. and Ajitha, P., 2020, October. Diagnosis of liver disease using machine learning models. In *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)* (pp. 627-630). IEEE.
21. Akter, S., Shekhar, H.U. and Akhteruzzaman, S., 2021. Application of biochemical tests and machine learning techniques to diagnose and evaluate liver disease. *Advances in Bioscience and Biotechnology*, 12(6), pp.154-172.
22. Kuzhippallil, M.A., Joseph, C. and Kannan, A., 2020, March. Comparative analysis of machine learning techniques for indian liver disease patients. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)* (pp. 778-782). IEEE.
23. Fathi, M., Nemati, M., Mohammadi, S.M. and Abbasi-Kesbi, R., 2020. A machine learning approach based on SVM for classification of liver diseases. *Biomedical Engineering: Applications, Basis and Communications*, 32(03), p.2050018.
24. OUR DATASET
25. Ramana, B.V., Babu, M.S.P. and Venkateswarlu, N.B., 2012. A critical comparative study of liver patients from USA and INDIA: an exploratory analysis. *International Journal of Computer Science Issues (IJCSI)*, 9(3), p.506.
26. Schmucker, D.L., 2005. Age-related changes in liver structure and function: Implications for disease?. *Experimental gerontology*, 40(8-9), pp.650-659.
27. Harrison-Findik, D.D., 2010. Gender-related variations in iron metabolism and liver diseases. *World journal of hepatology*, 2(8), p.302.
28. Rahm, E. and Do, H.H., 2000. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4), pp.3-13.

29. Mehrotra, D.V., Liu, F. and Permutt, T., 2017. Missing data in clinical trials: control-based mean imputation and sensitivity analysis. *Pharmaceutical statistics*, 16(5), pp.378-392.
30. Weisberg, S., 2001. Yeo-Johnson power transformations. *Department of Applied Statistics, University of Minnesota*. Retrieved June, 1, p.2003.
31. [31] <https://www.atoti.io/articles/when-to-perform-a-feature-scaling/#:~:text=Feature%20scaling%20is%20a%20method,during%20the%20data%20preprocessing%20step>.
32. Abdennour, N., Ouni, T. and Amor, N.B., 2021, November. The importance of signal pre-processing for machine learning: The influence of Data scaling in a driver identity classification. In 2021 IEEE/ACS 18th International Conference on Computer Systems and Applications (AICCSA) (pp. 1-6). IEEE.
33. Alshafer, H., 2021. Studying the effects of feature scaling in machine learning (Doctoral dissertation, North Carolina Agricultural and Technical State University).
34. <https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/>
35. <https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/>
36. <https://machinelearningmastery.com/imbalanced-classification-is-hard/>
37. Ito, A., Saito, K., Ueno, R. and Homma, N., 2021. Imbalanced data problems in deep learning-based side-channel attacks: Analysis and solution. *IEEE Transactions on Information Forensics and Security*, 16, pp.3790-3802.
38. Blagus, R., Lusa, L. SMOTE for high-dimensional class-imbalanced data. *BMC Bioinformatics* 14, 106 (2013). <https://doi.org/10.1186/1471-2105-14-106>
39. Hadwan, M., Al-Sarem, M., Saeed, F. and Al-Hagery, M.A., 2022. An improved sentiment classification approach for measuring user satisfaction toward governmental services' mobile apps using machine learning methods with feature engineering and SMOTE technique. *Applied Sciences*, 12(11), p.5547.
40. R. Blagus and L. Lusa, "Evaluation of SMOTE for High-Dimensional Class-Imbalanced Microarray Data," *2012 11th International Conference on Machine Learning and Applications*, Boca Raton, FL, USA, 2012, pp. 89-94, doi: 10.1109/ICMLA.2012.183.
41. Haibo He, Yang Bai, E. A. Garcia and Shutao Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, 2008, pp. 1322-1328, doi: 10.1109/IJCNN.2008.4633969.
42. He, H., Yang, B., Garcia, E.A. and Li, S.A., adaptive synthetic sampling approach for imbalanced learning. *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*; June 2008; Hong Kong, China.
43. Ramadhan, N.G., 2021. Comparative Analysis Of Adasyn-Svm And Smote-Svm Methods On The Detection Of Type 2 Diabetes Mellitus. *Scientific Journal Of Informatics*, 8(2), pp.276-282.

44. <https://machinelearningmastery.com/combine-oversampling-and-undersampling-for-imbalancedclassification/#:~:text=SMOTE%20is%20an%20oversampling%20method,dataset%20that%20have%20different%20classes.>
45. <https://www.kdnuggets.com/2016/08/learning-from-imbalancedclasses.html/2#:~:text=For%20example%2C%20Tomek%20links%20are,majority%20instance%20of%20the%20pair.>
46. Monard, M.C. and Batista, G.E.A.P.A., 2002. Learning with skewed class distributions. *Advances in Logic, Artificial Intelligence and Robotics*, 85, pp.173-180.
47. Hairani, H., Anggrawan, A. and Priyanto, D., 2023. Improvement Performance of the Random Forest Method on Unbalanced Diabetes Data Classification Using Smote-Tomek Link. *JOIV: International Journal on Informatics Visualization*, 7(1), pp.258-264.
48. Chandra, W., Suprihatin, B. and Resti, Y., 2023. Median-KNN Regressor-SMOTE-Tomek Links for Handling Missing and Imbalanced Data in Air Quality Prediction. *Symmetry*, 15(4), p.887.
49. Q. Ning, X. Zhao and Z. Ma, "A Novel Method for Identification of Glutarylation Sites Combining Borderline-SMOTE With Tomek Links Technique in Imbalanced Data," in *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 19, no. 5, pp. 2632-2641, 1 Sept.-Oct. 2022, doi: 10.1109/TCBB.2021.3095482.
50. Zhou, H., Dong, X., Xia, S. and Wang, G., 2021. Weighted oversampling algorithms for imbalanced problems and application in prediction of streamflow. *Knowledge-Based Systems*, 229, p.107306.
51. Jeatrakul, P., 2012. Enhancing classification performance over noise and imbalanced data problems (Doctoral dissertation, Murdoch University).
52. Xu, Z., Shen, D., Nie, T., Kou, Y., Yin, N. and Han, X., 2021. A cluster-based oversampling algorithm combining SMOTE and k-means for imbalanced medical data. *Information Sciences*, 572, pp.574-589.
53. Singh, N.D. and Dhall, A., 2018. Clustering and learning from imbalanced data. *arXiv preprint arXiv:1811.00972*.
54. Buabeng, A., Simons, A., Frempong, N.K. and Ziggah, Y.Y., 2021. A novel hybrid predictive maintenance model based on clustering, smote and multi-layer perceptron neural network optimised with grey wolf algorithm. *SN Applied Sciences*, 3(5), p.593.
55. <https://towardsdatascience.com/imbalanced-classification-in-python-smote-enn-method-db5db06b8d50>
56. Gao, Q., Jin, X., Xia, E., Wu, X., Gu, L., Yan, H., Xia, Y. and Li, S., 2020. Identification of orphan genes in unbalanced datasets based on ensemble learning. *Frontiers in Genetics*, 11, p.820.
57. More, A., 2016. Survey of resampling techniques for improving classification performance in unbalanced datasets. *arXiv preprint arXiv:1608.06048*.
58. Lamari, M., Azizi, N., Hammami, N.E., Boukhamla, A., Cheriguene, S., Dendani, N. and Benzebouchi, N.E., 2021. SMOTE-ENN-Based Data Sampling and Improved Dynamic

- Ensemble Selection for Imbalanced Medical Data Classification. In *Advances on Smart and Soft Computing: Proceedings of ICACIn 2020* (pp. 37-49). Springer Singapore.
59. Kumari, M. and Subbarao, N., 2022. A hybrid resampling algorithms SMOTE and ENN based deep learning models for identification of Marburg virus inhibitors. *Future Medicinal Chemistry*, 14(10), pp.701-715.
 60. Puri, A. and Kumar Gupta, M., 2022. Improved hybrid bag-boost ensemble with K-means-SMOTE-ENN technique for handling noisy class imbalanced data. *The Computer Journal*, 65(1), pp.124-138.
 61. Azhar, N.A., Pozi, M.S.M., Din, A.M. and Jatowt, A., 2022. An Investigation of SMOTE based Methods for Imbalanced Datasets with Data Complexity Analysis. *IEEE Transactions on Knowledge and Data Engineering*.
 62. Kabir, M.F. and Ludwig, S., 2018, December. Classification of breast cancer risk factors using several resampling approaches. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 1243-1248). IEEE.
 63. Wang, Q., Luo, Z., Huang, J., Feng, Y. and Liu, Z., 2017. A novel ensemble method for imbalanced data learning: bagging of extrapolation-SMOTE SVM. *Computational intelligence and neuroscience*, 2017.
 64. Xu, Z., Shen, D., Nie, T. and Kou, Y., 2020. A hybrid sampling algorithm combining M-SMOTE and ENN based on random forest for medical imbalanced data. *Journal of Biomedical Informatics*, 107, p.103465.
 65. <https://www.sciencedirect.com/topics/engineering/confusionmatrix#:~:text=A%20confusion%20matrix%20is%20a,performance%20of%20a%20classification%20algorithm>.
 66. <https://deepai.org/machine-learning-glossary-and-terms/accuracy-error-rate>
<https://www.javatpoint.com/precision-and-recall-in-machine-learning>
 67. <https://www.iguazio.com/glossary/recall/#:~:text=Recall%2C%20also%20known%20as%20the,total%20samples%20for%20that%20class>.
 68. <https://www.v7labs.com/blog/f1-score-guide#:~:text=for%20Machine%20Learning-,What%20is%20F1%20score%3F,prediction%20across%20the%20entire%20dataset>.
 69. <https://www.anyscale.com/blog/what-is-hyperparameter-tuning#:~:text=Hyperparameter%20tuning%20consists%20of%20finding,better%20results%20with%20fewer%20errors>.
 70. Ahmad, G.N., Fatima, H., Ullah, S. and Saidi, A.S., 2022. Efficient medical diagnosis of human heart diseases using machine learning techniques with and without GridSearchCV. *IEEE Access*, 10, pp.80151-80173.
 71. Chakraborty, D. and Elzarka, H., 2019. Advanced machine learning techniques for building performance simulation: a comparative analysis. *Journal of Building Performance Simulation*, 12(2), pp.193-207.

