Heaven's Light Is Our Guide

PREDICTION OF LIVER DISEASE USING STATISTICAL MACHINE LEARNING APPROACH

A Thesis Submitted in Partial

fulfillment for the requirement of the

degree of

Bachelor of Science

in

Electrical & Computer Engineering

by

Syed Tousif Mahadi Farabi

Roll: 1710022

to the

Department of Electrical & Computer Engineering

Rajshahi University of Engineering & Technology

September, 2023

PREDICTION OF LIVER DISEASE USING STATISTICAL MACHINE LEARNING APPROACH

A Project/Thesis Submitted in Partial
fulfillment for the requirement of the
degree of

Bachelor of Science
in
Electrical & Computer Engineering

by

Syed Tousif Mahadi Farabi
Roll No. 1710022

to the

Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

September, 2023

Acknowledgement

This thesis was submitted to the Department of Electrical and Computer Engineering at Rajshahi University of Engineering & Technology (RUET), Rajshahi-6204, Bangladesh in partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical & Computer Engineering. "**PREDICTION OF LIVER DISEASE USING STATISTICAL MACHINE LEARNING APPROACH**" is the title of the thesis.

First, I would like to thank my supervisor, **Oishi Jyoti**, Lecturer in the Department of Electrical and Computer Engineering, for her unwavering support and invaluable guidance. From the outset of this research journey, her guidance has been instrumental in shaping my work. In tandem with her expertise in related fields, she has been an unswerving supporter in overcoming any research-related obstacles.

I would like to express my gratitude to Assistant Professor Sagor Chandro Bakchy, the head of the Department of Electrical and Computer Engineering, for his continued support, encouragement, and readily accessible laboratory facilities.

Lastly, but by no means least, I extend my deepest appreciation to the cornerstone figures in my life—my parents. Their unwavering support has been both a comfort and a catalyst, fueling my pursuit of this scholarly endeavor. In every sense—emotional, intellectual, and logistical—they have been my bedrock, and for that, I am profoundly grateful.

Syed Tousif Mahadi Farabi
Roll: 1710022

September 2023
RUET, Rajshahi

CERTIFICATE

This is to certify that the thesis entitled "***PREDICTION OF LIVER DISEASE USING STATISTICAL MACHINE LEARNING APPROACH"*** by Syed Tousif Mahadi Farabi (Roll No. 1710022), has been carried out under my direct supervision. To the best of my knowledge, this thesis is an original one and has not been submitted anywhere for any degree or diploma.

Thesis Supervisor:

.....

Oishi Jyoti

Lecturer

Department of Electrical & Computer Engineering

Rajshahi University of Engineering & Technology

CERTIFICATE

This is to certify that the thesis entitled "***PREDICTION OF LIVER DISEASE USING STATISTICAL MACHINE LEARNING APPROACH"*** has been corrected according to my suggestion and guidance as an external. The quality of the thesis is satisfactory.

External Member:

.....

[External Member]

[Designation]

Department of Electrical & Computer Engineering

Rajshahi University of Engineering & Technology

Abstract

In recent years, the early detection of liver disease has become crucial due to its escalating prevalence and potential for fatal outcomes. Traditional diagnostic methods often lack the sensitivity and specificity needed for early-stage identification. This study aims to address this gap by utilizing machine learning algorithms for liver disease prediction. We employ an Ensemble Stacking Classifier, integrating XGBoost, Random Forest, and CART as base models, with Logistic Regression serving as the meta-model. This methodology was chosen based on its superior test accuracy over traditional classifiers. We further enhance the model's performance by imputing missing values using the MICE technique and balancing the dataset through minority up-sampling using the mean strategy. Our results indicate that this ensemble approach achieved an outstanding accuracy of 91.01% and an AUC-ROC score of 97.98%. These findings demonstrate the efficacy of machine learning methods, specifically Ensemble Stacking, in the early detection of liver disease, thereby providing a robust tool for clinical application.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Definition
CMP	Comprehensive Metabolic Panel
ML	Machine Learning
PCA	Principal Component Analysis
FA	Factor Analysis
LDA	Linear Discriminant Analysis
ILPD	Indian Liver Patient Dataset
ILDPS	Intelligent Liver Disease Prediction Software
ANN	Artificial Neural Network
KNN	K-Nearest Neighbor
SVM	Support Vector Machine
GNB	Gaussian Naïve Bayes
LR	Logistic Regression
RF	Random Forest
XGBoost	Extreme Gradient Boosting
CART	Classification And Regression Tree
ROC	Receiver Operating Characteristic
AUC	Area Under the ROC Curve
EDA	Exploratory Data Analysis
MICE	Multivariate Imputation by Chained Equations
SMOTE	Synthetic Minority Over-sampling Technique
ROSE	Random Over-sampling Examples
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
TP	True Positive
TN	True Negative
TPR	True Positive Rate
TNR	True Negative Rate
FPR	False Positive Rate
FNR	False Negative Rate
GS	Grid Search
FS	Feature Selection

Chapter 1
Introduction

The rise of liver diseases, such as cirrhosis, hepatitis, and liver cancer, continues to be a global health concern, resulting in millions of new cases and hundreds of thousands of fatalities each year. While advancements in medical treatments have led to a decrease in overall mortality rates, liver-related conditions still account for a significant percentage of global deaths. Early diagnosis remains the cornerstone of effective management and treatment of liver conditions. Traditional diagnostic methods involve an array of tests, including urinalysis, complete blood count, and comprehensive metabolic panel (CMP). However, these diagnostic approaches can be time-consuming, invasive, and expensive, particularly for diseases that necessitate multiple types of tests or repeated sampling.

Machine learning (ML) algorithms present a compelling alternative to traditional diagnostic methods by offering the ability to analyze complex medical data with speed and accuracy. These computational methods have demonstrated their efficacy in medical applications such as disease classification and prognosis. Furthermore, the adaptability of machine learning models enables them to be trained on emerging datasets, ensuring that they remain current and reliable for medical decision-making. This dynamic nature of machine learning can significantly contribute to personalized medicine, tailoring diagnoses and treatment plans to individual patient profiles.

However, the use of machine learning in the medical domain is not devoid of obstacles. For instance, the high dimensionality of medical data sets often introduces noise, leading to overfitting and reduced model performance. Additionally, the handling of missing data, a pervasive issue in medical research, affects the quality of diagnosis and introduces bias. More importantly, existing models often struggle with imbalanced classes, which can further compromise the diagnostic accuracy.

Given these challenges, this study aims to introduce an integrated approach that combines statistical methods with advanced machine learning algorithms. By considering the intricacies of medical data, such as missing values and class imbalances, we aspire to develop a more robust and accurate predictive model for liver diseases.

Through an integrative methodology that addresses the aforementioned challenges, this study seeks to pave the way for more effective, data-driven approaches in the early diagnosis and management of liver diseases.[3][7][8][9]

1.1 Motivation

Liver diseases often develop inconspicuously, with symptoms easily overlooked until advanced stages. Therefore, early diagnosis is not just beneficial but essential for effective management and improving the quality of life for patients. Traditional diagnostic methods often involve invasive procedures like biopsies or expensive imaging techniques that may not be readily accessible to everyone. Furthermore, these approaches may have their limitations, including the risk of complications, time consumption, and in some cases, delayed or incorrect diagnosis. Given the gravity and prevalence of liver diseases, there's a dire need for quicker, more accessible, and accurate diagnostic tools.

Machine learning and deep learning technologies offer an alternative route, one that is increasingly proving to be more effective for diagnostic healthcare. These algorithms possess the capability to analyze extensive datasets, thereby detecting patterns and connections that may be less apparent or completely disregarded during manual study. The automation and predictive capability inherent in these technologies not only speed up the diagnosis but also hold promise for higher accuracy and, consequently, better treatment planning.

However, while machine learning has shown promise, it's also fraught with challenges when applied to medical data. Issues such as high-dimensionality, missing values, and class imbalances can impact the performance and reliability of machine learning models. These challenges necessitate the development of more robust machine learning frameworks that can adapt and provide reliable results. Several projection-based statistical methods like Principal Component Analysis (PCA), Factor Analysis (FA), and Linear Discriminant Analysis (LDA) have shown promise in mitigating some of these problems, particularly in reducing data dimensionality.

Our thesis, therefore, aims to explore and integrate machine learning and deep learning algorithms, enhanced by projection-based statistical methods, for the diagnosis and prediction of liver diseases. In doing so, we intend to address the existing challenges like high-dimensionality, missing data, and class imbalances that conventional models often struggle with. The intent is to pave the way for a diagnostic tool that is not only more accessible and less invasive but also more accurate and reliable.

In summary, the motivation for this thesis arises from the critical need for early and effective diagnosis of liver diseases and the potential for machine learning and deep learning techniques to revolutionize this space. We propose an integrated approach to overcome existing limitations, aiming for a diagnostic paradigm that is quicker, more accurate, and widely accessible.[3][7][8][9]

1.2 Literature Review

The following list includes some literature reviews of recent studies on classification of liver diseases:

1.2.1 Case 1

Thirunavukkarasu K et al.[1] used several classification algorithms, including Logistic Regression, Support Vector Machine, and K-Nearest Neighbor, in their research endeavor aimed at predicting liver disease. The performance of these algorithms was evaluated by comparing their classification accuracy, which was assessed via the use of a confusion matrix. Based on the empirical findings, it was noted that Logistic Regression and K-Nearest Neighbour exhibited the highest level of accuracy, reaching a rate of 73.97%. On the other hand, Logistic Regression and Support Vector Machine exhibited the maximum sensitivity value of 0.952. Consequently, it may be inferred that Logistic Regression is a viable approach for predicting liver disease.

1.2.2 Case 2

In their study, Ketan Gupta et al.[2] employed several machine learning algorithms, including Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors, Gradient Boosting, Extreme Gradient Boosting, and LightGB, to predict the occurrence of liver disease. The Random Forest and LightGB models had the highest accuracy, with a rate of 63%. Additionally, the Random Forest model demonstrated superior precision. Based on the conducted study and subsequent calculations, it has been determined that these algorithms have achieved marginally improved accuracy following the process of feature selection. The study's findings indicate that suboptimal data pre-processing techniques resulted in average accuracy ratings.

1.2.3 Case 3

In the approach proposed by Ruhul Amin et al.[3], multiple classification algorithms including Random Forest, Multilayer Perceptron Network, KNNeighbor, Logistic Regression, Support Vector Machine, and Ensemble Stacking, were incorporated. The researchers put forth a statistical projection-based approach for integrating features extracted from various dimensionality reduction techniques, including PCA, FA, and LDA. Between the classifiers evaluated, it was observed that the Random Forest classifier exhibited the highest accuracy (88.10%) and AUC (88.20%) scores.

1.2.4 Case 4

Elias Dritsas and Maria Trigka present a comparative study[4] on the effectiveness of various machine learning and ensemble methods for liver disease prediction. Utilizing metrics such as accuracy, precision, recall, F-measure, and AUC, the authors found that the Voting classifier exhibited superior performance. Specifically, it achieved an accuracy of 80.1%, a precision of 80.4%, and an AUC of 88.4% using 10-fold cross-validation post-SMOTE balancing. The paper also leverages the Random Forest model to evaluate feature importance through Gini impurity, enriching its methodological rigor.

1.2.5 Case 5

Jagdeep Singha, Sachin Baggab, and Ranjodh Kaurc[5] investigates the utility of various classification algorithms such as Logistic Regression, SMO, Random Forest, Naive Bayes, J48, and k-nearest neighbor (IBk) in predicting liver disease risks. The study employs an assortment of attributes from a liver patient dataset for this purpose. The authors also explore the influence of feature selection on classifier performance, ultimately finding that Logistic Regression coupled with feature selection yields the highest accuracy (74.36%). This paper contributes to the development of Intelligent Liver Disease Prediction Software (ILDPS), which integrates both feature selection and classification techniques within a software engineering framework.

1.2.6 Case 6

In the paper authored by Srilatha Tokala et al.[6], the focus is on leveraging machine learning algorithms to develop classification models for predicting chronic liver disorders. Utilizing patient datasets, the study aims to alleviate the clinical burden by automating the diagnostic process. Various classification techniques were evaluated, and the Random Forest Classifier emerged as the most accurate with an 87% success rate. The outcomes are quantified using confusion matrices, illustrating the model's predictive reliability. This work not only advances automated healthcare diagnostics but also offers a viable solution for effective liver disorder classification when a training dataset is available.

1.3 Objective

The aim of this study is to utilize a range of classification algorithms to analyze intricate medical data to provide accurate predictions regarding liver diseases. The objective is to optimize the accuracy of a specific classification algorithm by identifying the most effective combination of techniques for handling missing values, balancing data, normalizing data, and reducing

dimensionality. Subsequently, through the process of evaluating diverse classifiers using many scoring criteria, the intention is to identify the most effective classifiers for our suggested model. As a result, the goals of this work can be summed up as follows:

- Employing visualization techniques to represent the information content.
- Employing distinct data cleaning and pre-processing techniques for each of the classifiers.
- To evaluate and compare several data pre-processing techniques in terms of their impact on the accuracy of a certain classifier.
- Finally, the process involves evaluating various classifiers and their corresponding best scores in order to determine the most suited solutions for our specific model.

1.4 Thesis Outline

By studying medical data, this study's analysis of liver disease suggests a suitable strategy for classifying patients with liver disease. Using classifiers based on ensemble learning and gradient-boosting decision trees, potential liver disease patients' medical records were classified using a dataset collected from machine learning repository at the University of California, Irvine. Various metrics, including training and test accuracy, AUC-ROC score, ROC curve, and confusion matrix, were used to evaluate the model's efficacy.

The following are notable contributions made by this work:

- To determine the optimal hyper-parameters for a classifier to predict liver disease.
- Using Random Forest and XGBoost algorithms with fewer parameters and acceptable accuracy, a dataset with missing and unbalanced values was classified binarily for liver disease.
- Evaluated the Logistic Regression, Support Vector Machine, K-NN, Gaussian Naive Bayes, Artificial Neural Network, CART, Random Forest, XGBoost, and Ensemble Stacking models using their respective performance metrics.

1.5 Thesis Organization

The subsequent sections of this thesis comprise four distinct chapters, each of which is outlined below.

Chapter 1- Introduction

This chapter presents a thorough examination of the fundamental rationale behind this thesis, the comprehensive review of literature pertaining to the prediction of liver disease, the clearly defined objectives, the noteworthy contributions, and the structure of the thesis.

Chapter 2- Methodology

This chapter encompasses the block diagram and the comprehensive working methodology. This chapter covers the compilation of dataset specifications, diverse data pre-processing methodologies, the proposed model's structure, and the algorithms utilized in this investigation.

Chapter 3- Results & Discussion

The chapter presents the quantitative findings obtained from our research. A range of performance indicators have been utilized to assess the effectiveness of diverse classifiers, including accuracy, ROC-AUC score, ROC curve, and confusion matrix. By conducting an assessment of algorithmic performance and conducting a thorough examination of existing academic research, we have compiled an extensive collection of successful classifiers, along with their corresponding suitable data pre-processing methods.

Chapter 4- Conclusion and Future Works

This chapter concludes the thesis. This chapter also discusses limitations of the thesis and proposes pertinent avenues for future investigation.

Chapter 2
Methodology

In this study, classification algorithms based on machine learning are used to provide an in-depth analysis of the general prediction methodology for liver diseases. Fig. 2.1 exhibits a flowchart demonstrating the process of conducting the entire study. The carried-out method is laid out in detail in the sections that follow.

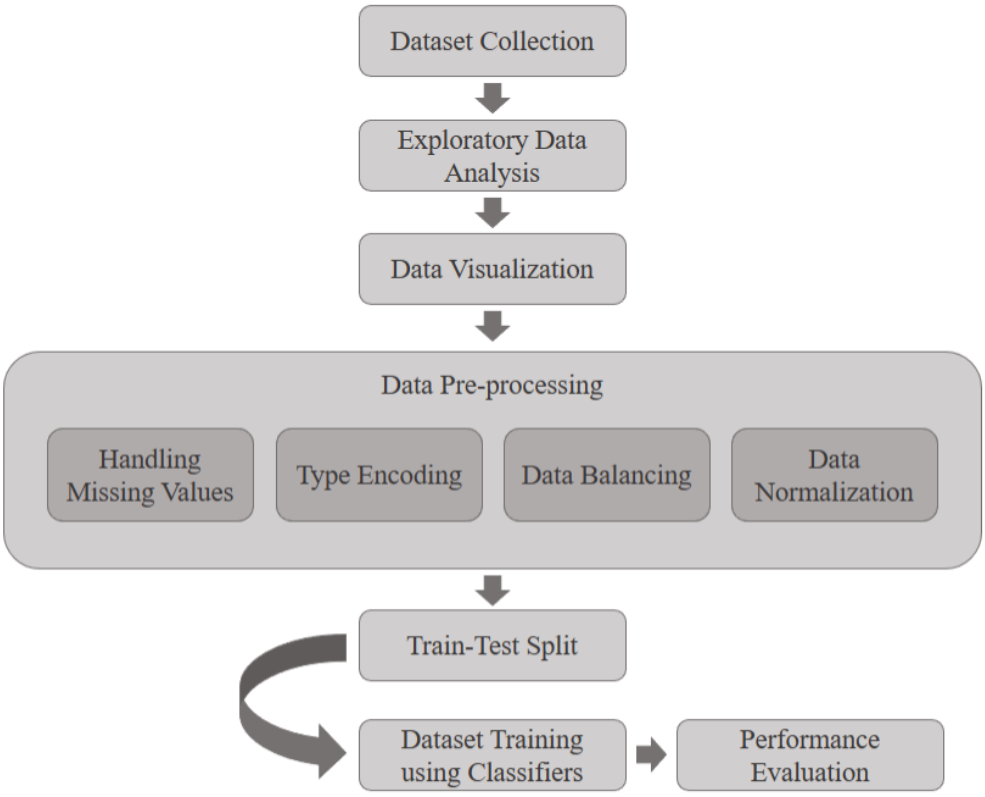

Fig. 2.1: Overview of the workflow

2.1 Dataset Description

The dataset utilized in this thesis[10] comprises an extensive compilation of medical records centered on liver diseases, spanning a total of 583 patient entries. The dataset utilized in this study originates from the northeastern region of Andhra Pradesh, India. It consists of 416 records from patients who have been diagnosed with liver disease, and an additional 167 records from individuals who do not have liver disease, serving as a control group. The dataset comprises 441 male patients and 142 female patients, indicating a gender distribution. One notable attribute of the dataset is the practice of representing patients aged 90 and above as

being 90 years old, which serves the purpose of safeguarding confidentiality and facilitating analyses.

The dataset includes a variety of essential medical metrics that are vital for detecting and comprehending liver disease. The variables considered in this study cover the demographic characteristics of the patient, such as age and gender. Additionally, the levels of Total Bilirubin and Direct Bilirubin, Alkaline Phosphatase, Alamine Aminotransferase, Aspartate Aminotransferase, Total Proteins, Albumin, and the Albumin-Globulin Ratio are considered. The column labeled "Dataset" is utilized to classify individuals into two distinct groups: those who have been diagnosed with liver disease and those who have not.

The dataset had originally been accessible to the public through the UCI Machine Learning Repository. It has been employed in this thesis to implement and verify machine learning and deep learning algorithms with the objective of early detection and prediction of liver diseases. The primary aim of utilizing this dataset is to minimize the diagnostic workload on healthcare practitioners by developing a predictive system that is more automated, precise, and prompt.

List of Columns:

- Age of the patient
- Gender of the patient
- Total Bilirubin
- Direct Bilirubin
- Alkaline Phosphatase
- Alamine Aminotransferase
- Aspartate Aminotransferase
- Total Proteins
- Albumin
- Albumin and Globulin Ratio
- Dataset: Partitioned into two distinct subsets based on a specific criterion, namely the presence or absence of liver disease in patients.

2.2 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is essential to data science, which involves high-stakes medical research. EDA uses various techniques to analyze complex datasets. Univariate analysis involves the computation of fundamental statistical metrics, which include the mean,

median, and standard deviation based on a solitary variable. On the other hand, bivariate analysis investigates the association between two variables, while multivariate analysis explores the interrelationships among several variables inside a given dataset. These methods use histograms, scatter plots, and box plots to show data trends, outliers, and anomalies.

Exploratory data analysis (EDA) aims to provide an overview of the data set and guide future research and idea generation. Exploratory Data Analysis (EDA) plays a pivotal role in the identification and prognostication of liver diseases through the utilization of machine learning and deep learning algorithms. Initially, it offers initial understanding of the organization and features of the data, hence informing the selection of further research algorithms. Additionally, it aids in the identification of patterns and correlations among variables for the purpose of diagnostic modeling. The comprehension of the correlation between liver function metrics and the various phases of liver illness can have implications for the process of feature engineering and the selection of appropriate models. Furthermore, exploratory data analysis (EDA) plays a crucial role in identifying and resolving prevalent difficulties within medical datasets, such as the presence of missing or inconsistent data. These issues have the potential to undermine the trustworthiness of studies conducted using such datasets.

A comprehensive EDA is essential to this research because early and accurate liver disease diagnosis is crucial. Through rigorous EDA, machine learning and deep learning models are improved for reliability, robustness, and predictive accuracy to improve diagnostic procedures and patient outcomes.

```
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Age                         583 non-null    int64
 1   Gender                      583 non-null    object
 2   Total_Bilirubin             583 non-null    float64
 3   Direct_Bilirubin            583 non-null    float64
 4   Alkaline_Phosphotase        583 non-null    int64
 5   Alamine_Aminotransferase    583 non-null    int64
 6   Aspartate_Aminotransferase  583 non-null    int64
 7   Total_Protiens              583 non-null    float64
 8   Albumin                     583 non-null    float64
 9   Albumin_and_Globulin_Ratio  579 non-null    float64
 10  Dataset                     583 non-null    int64
dtypes: float64(5), int64(5), object(1)
```

Fig. 2.2: Information on the 583x11 shaped dataset.

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Age	583.0	NaN	NaN	NaN	44.746141	16.189833	4.0	33.0	45.0	58.0	90.0
Gender	583	2	Male	441	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Total_Bilirubin	583.0	NaN	NaN	NaN	3.298799	6.209522	0.4	0.8	1.0	2.6	75.0
Direct_Bilirubin	583.0	NaN	NaN	NaN	1.486106	2.808498	0.1	0.2	0.3	1.3	19.7
Alkaline_Phosphotase	583.0	NaN	NaN	NaN	290.576329	242.937989	63.0	175.5	208.0	298.0	2110.0
Alamine_Aminotransferase	583.0	NaN	NaN	NaN	80.713551	182.620356	10.0	23.0	35.0	60.5	2000.0
Aspartate_Aminotransferase	583.0	NaN	NaN	NaN	109.910806	288.918529	10.0	25.0	42.0	87.0	4929.0
Total_Protiens	583.0	NaN	NaN	NaN	6.48319	1.085451	2.7	5.8	6.6	7.2	9.6
Albumin	583.0	NaN	NaN	NaN	3.141852	0.795519	0.9	2.6	3.1	3.8	5.5
Albumin_and_Globulin_Ratio	579.0	NaN	NaN	NaN	0.947064	0.319592	0.3	0.7	0.93	1.1	2.8
Dataset	583.0	NaN	NaN	NaN	1.286449	0.45249	1.0	1.0	1.0	2.0	2.0

Fig. 2.3: Data description

The INDIAN LIVER PATIENT DATASET is structured as a Pandas Data Frame comprising 583 entries spread across 11 distinct attributes. Most of these attributes are numerical in nature, encoded either as integers (**int64**) or floating-point numbers (**float64**). Only the 'Gender' attribute is labeled as an object, signifying its categorical nature.

Necessities for Data Cleaning and Refinement

- **Categorical Variable Conversion**: The 'Gender' attribute embodies two unique descriptors—'Male' and 'Female'. For machine learning algorithms to process this attribute, it needs to be numerically transformed.
- **Extreme Value Rectification**: Several columns reveal a considerable gap between the 75th percentile and the maximum values. This suggests the existence of outliers or extreme values that require attention.
- **Target Feature Reformatting**: The outcome variable 'Dataset' is binary but is represented by the numbers 1 and 2. It should be recorded to fall within a [0, 1] range for compatibility with classification algorithms.
- **Addressing Missing Entries**: The 'Albumin_and_Globulin_Ratio' feature has a few missing entries that require imputation or elimination.

The attribute 'Age' shows a wide span, ranging from as low as 4 years to a capped-off point of 90 years, with a median value at 45 years. Liver-relevant measures like 'Total_Bilirubin' and 'Direct_Bilirubin' exhibit marked variability, featuring median values of 1.0 and 0.3, respectively. Enzymatic indicators like 'Alamine_Aminotransferase' and

'Aspartate_Aminotransferase' also vary substantially, pointing to a diverse set of liver health conditions within the dataset.

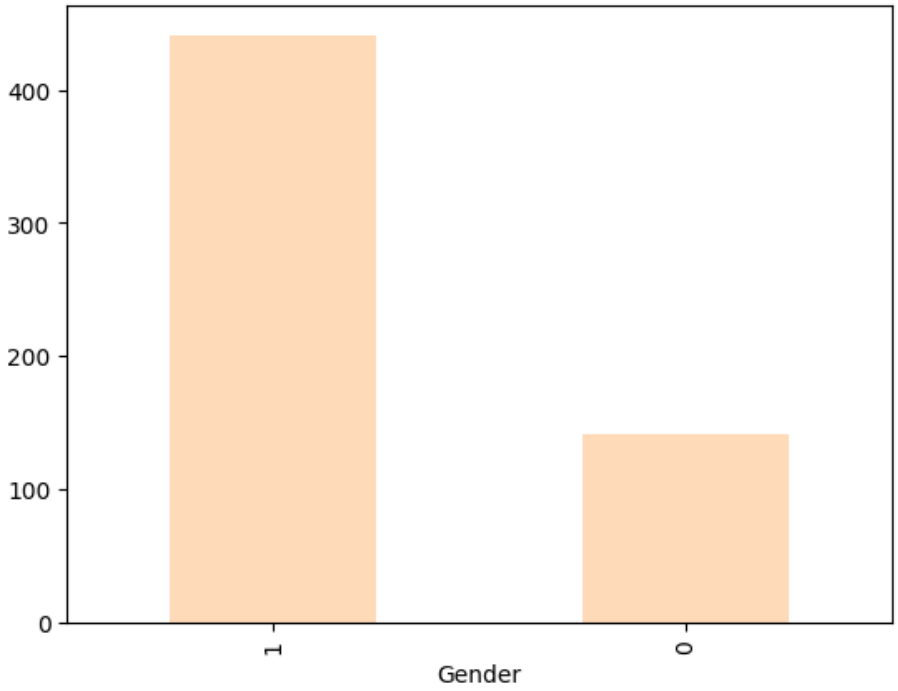

Fig. 2.4: Gender proportionality bar-plot

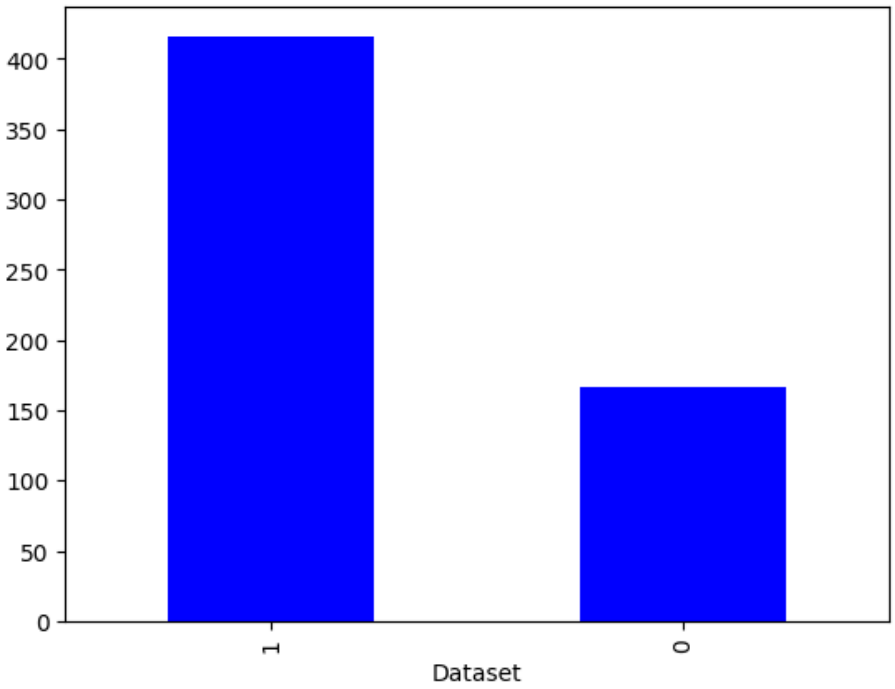

Fig. 2.5: ‘Dataset’ attribute’s class proportionality bar-plot

Noteworthy Observations and Preliminary Conclusions:

- **Gender Proportionality**: The dataset leans towards male representation with 441 male entries compared to 142 female entries, thus highlighting a gender imbalance.
- **Disparity in Liver Indicators**: Biochemical markers like 'Total_Bilirubin' and 'Alkaline_Phosphotase' vary widely, signaling the different liver conditions affecting the represented patients.
- **Outcome Class Skewness**: The outcome attribute 'Dataset' reveals an imbalance between the classes, which necessitates careful consideration during model training to mitigate bias.

The initial data exploration suggests that prior to model training, the dataset will benefit from several data cleaning and transformation steps. Moreover, the dataset's complexity and diversity make it a robust foundation for crafting reliable diagnostic models for liver conditions.

2.3 Data Visualization

In the context of liver disease prediction, an intricate web of variables such as age, gender, and biochemical markers interact in ways that can be difficult to untangle through numerical data alone. The subtleties in this high-dimensional data space can be easily overlooked without an effective visualization strategy. Data visualization allows us to break down complex patterns into digestible insights. By presenting data in a graphical format, clinicians, researchers, and even patients can make sense of the complex interrelationships between different indicators of liver health.

Inter-variable correlation

Visualizing the strength and direction of a two-variable relationship requires correlation graphs. Liver disease prediction is multi-parametric, thus understanding how age, gender, and biochemical markers interact is critical. Visualizing the correlation between 'Total Bilirubin' and 'Direct Bilirubin' can reveal their predictive power, which is essential for effective machine learning models.

Correlation Table

To gain a more complete understanding, a correlation table displays pairwise correlations of coefficients between all specified variables. This method quickly identifies highly correlated variables that may need attention in feature selection, preventing multicollinearity in the model. Creating an accurate predictive model for liver illness using several factors is extremely beneficial.

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio
Age	1.000000	-0.012416	0.034775	0.033732	0.089161	-0.057485	-0.007433	-0.162348	-0.245568	-0.214170
Gender	-0.012416	1.000000	0.102102	0.114015	0.009844	0.084179	0.083314	-0.078495	-0.068075	0.016423
Total_Bilirubin	0.034775	0.102102	1.000000	0.880576	0.232132	0.234754	0.257471	-0.022587	-0.213947	-0.204725
Direct_Bilirubin	0.033732	0.114015	0.880576	1.000000	0.262720	0.255962	0.278307	-0.016392	-0.221468	-0.203501
Alkaline_Phosphotase	0.089161	0.009844	0.232132	0.262720	1.000000	0.149660	0.181959	-0.024880	-0.161676	-0.237439
Alamine_Aminotransferase	-0.057485	0.084179	0.234754	0.255962	0.149660	1.000000	0.796017	-0.034647	-0.038822	-0.018588
Aspartate_Aminotransferase	-0.007433	0.083314	0.257471	0.278307	0.181959	0.796017	1.000000	-0.024379	-0.087303	-0.078070
Total_Protiens	-0.162348	-0.078495	-0.022587	-0.016392	-0.024880	-0.034647	-0.024379	1.000000	0.815073	0.284493
Albumin	-0.245568	-0.068075	-0.213947	-0.221468	-0.161676	-0.038822	-0.087303	0.815073	1.000000	0.707126
Albumin_and_Globulin_Ratio	-0.214170	0.016423	-0.204725	-0.203501	-0.237439	-0.018588	-0.078070	0.284493	0.707126	1.000000
Dataset	0.172130	0.098110	0.279021	0.310705	0.221565	0.213447	0.199662	-0.030142	-0.173209	-0.185069

Fig. 2.6: Full correlation table

Observing the correlation table, several key insights were garnered, which hold significant implications for the predictive modeling of liver disease. One of the most prominent findings is the strong correlations observed among specific variables. For instance, 'Direct Bilirubin' and 'Total Bilirubin' displayed a high correlation, suggesting a possibility of redundant information. Similarly, 'Aspartate Aminotransferase' and 'Alamine Aminotransferase', as well as 'Total Proteins' and 'Albumin', were closely related. Another noteworthy correlation was between 'Albumin and Globulin Ratio' and 'Albumin'. These strong correlations indicate the necessity of feature engineering and dimensionality reduction techniques to remove multicollinearity, ensuring a more accurate and effective predictive model for liver disease. Addressing these correlations will refine the model's predictive accuracy while simplifying its complexity, a crucial step in developing a robust tool for liver disease diagnosis.

2.4 Data Pre-processing

Predictive modeling relies heavily on the pre-processing of data, particularly when dealing with intricate datasets like the Indian Liver Patient Dataset. In our study, we mainly focus on three data pre-processing techniques: handling missing values, type encoding, data balancing and data normalization methods.

Missing Value Replacement Using Mean

The initial stage of our pre-processing pipeline is addressing the issue of missing values. The most straightforward method for addressing missing values is by univariate missing value imputation. In this approach, each missing value within a specific feature is substituted with the mean of the observed values for that feature. Mathematically, the representation of this can be expressed as:

$$x_{\text{missing}} = \frac{\sum_{i=1}^{n} x_i}{n}$$

where x_i are the observed values and n is the number of observed values.

For our dataset, the 'Albumin_and_Globulin_Ratio' column has 4 missing values, making this technique particularly useful.[11]

Multivariate Imputation by Chained Equations (MICE)

For tackling the missing values in our dataset, a more nuanced approach called Multivariate Imputation by Chained Equations (MICE) is applied. This method is especially useful when dealing with multiple variables that have missing entries and aims to create multiple imputations rather than single-point estimates.

Algorithm Steps:

1) Initialization: Perform simple imputations, such as mean or median imputation, to fill the missing data points for all variables. This creates an initial "complete" dataset.
2) *Iterative Imputation:* For each variable X_i with missing data:
    a) Set the current variable X_i as the dependent variable, and all other variables as independent variables.
    b) Use the observed (non-missing) values of X_i to fit a predictive model using all other variables as predictors. The model can be as simple as a linear regression or more complex.
    c) Use the fitted model to predict the missing values of X_i.
3) *Sampling:* For each variable X_i with missing data, sample the imputed values from their predictive distributions derived in Step 2.
4) *Iterate:* Repeat Steps 2 and 3 for a pre-defined number of cycles to allow the imputations to converge.

The mathematical rigor behind MICE generally involves solving a series of chained equations. Suppose we have k variables $\{X_1, X_2, \ldots, X_k\}$ with some missing data. We proceed by specifying a model for each X_i conditioned on X_{-i}, where X_{-i} represents all variables other than X_i.

For each X_i, we fit the model:

$$X_i \mid \mathbf{X}_{-i} = f(\mathbf{X}_{-i}; \theta_i)$$

Here, f can be any function that maps X_{-i} to X_i, and θ_i are the parameters for this function, which are typically estimated using maximum likelihood.

The "chained equations" in MICE come from iterating these models for each X_i with missing data, filling in estimated values at each step and then refining these estimates through subsequent iterations.[12]

Binary Encoding

In the scope of data analysis and machine learning, the transformation of categorical attributes into a numerical format is often necessary for the successful implementation of various algorithms. Within our dataset, the column labeled 'Gender' poses such a challenge, being designated as an object data type. To convert this into a machine-friendly format, we opt for the straightforward method known as binary encoding.[13]

Binary encoding serves as an efficient means to convert categories into numerical identifiers. Given that our 'Gender' column contains only two distinct categories—'Male' and 'Other'—this method is ideally suited for our purposes. Here, we assign 'Male' the numerical value '1', while every other gender category is allocated the value '0'. In mathematical notation, this transformation is illustrated as:

$$\text{Encoded\_Gender} = \begin{cases} 1 & \text{if Gender} = \text{'Male'} \\ 0 & \text{if Gender} = \text{'Other'} \end{cases}$$

Similarly, for our 'Dataset' column, which acts as an indicator of whether an individual is afflicted with liver disease, a binary encoding is also applied. In this setting, the numeral '1' corresponds to the existence of liver disease, while '0' signals its absence. This conversion is mathematically articulated as[14][15]:

$$\text{Encoded\_Dataset} = \begin{cases} 1 & \text{if Dataset indicates Liver Disease} \\ 0 & \text{if Dataset indicates No Liver Disease} \end{cases}$$

Data Balancing

Addressing unbalanced classes in machine learning and data science is challenging, especially when trying to develop a prediction model with high accuracy and predictability. Minority Up-sampling and Majority Down-sampling are typical solutions.

Minority Up-sampling: Minority Up-sampling is crucial in cases where there is insufficient data. To achieve this, we will artificially increase the size of the minority group by duplicating their records. Given the size limitations of our dataset, where we are working with a lower volume of records for individuals without liver disease, up-sampling is mandatory. To precisely

align the minority class (non-liver disease) with the majority class (liver disease), we increased the minority class records to 416. In this operation, we utilized the 'resample' function from Python's scikit-learn library. After the up-sampling, the overall dataset size increased to 832 records. In our scenario, using up-sampling to reduce class imbalance and increase the size of the training sample for machine learning models is quite beneficial.[16][17][18]

Synthetic Minority Over-sampling Technique (SMOTE): The Synthetic Minority Over-sampling Technique (SMOTE) is a method that aims to mitigate the issue of class imbalance by creating artificial instances inside the feature space. The SMOTE algorithm acts inside the feature space spectrum, therefore enhancing the representation of the minority class by generating more intricate data points. These data points are formed by combining instances that are closely situated to one another. This approach contrasts with basic over-sampling techniques, since the replication of existing minority occurrences in simple over-sampling might result in overfitting.

Let $X=\{x_1, x_2, \ldots, x_N\}$ be the minority class samples, where x_i represents each feature vector. For each x_i, SMOTE selects k nearest neighbors, and the set of nearest neighbors is denoted by $\mathrm{NN}(x_i)$.

To create a synthetic sample x_{new}, SMOTE performs the following computation:

$$x_{\text{new}} = x_i + \lambda \times (x_{\text{nn}} - x_i)$$

Here x_{nn} is a randomly chosen neighbor from $\mathrm{NN}(x_i)$ and λ is a random number between 0 and 1.

The minority class and synthetic samples have been merged to create a balanced dataset. For our dataset, especially with the few records for patients without liver illness, SMOTE will improve the model's learning power without bias.[19][20][21]

Random Over-Sampling Examples (ROSE): The Random Over-Sampling Examples (ROSE) algorithm is another technique employed to tackle the issue of class imbalance in datasets. Unlike SMOTE, which generates synthetic instances, ROSE generates new samples by applying a smoothed bootstrap approach. The method is particularly effective when dealing with smaller datasets, making it an excellent choice for our study on liver diseases.

Let $C = \{c_1, c_2, \ldots, c_M\}$ be the minority class samples, where c_i represents each feature vector. Let N be the desired number of synthetic samples. A synthetic sample c_{new} is generated using a smoothed bootstrap sample from C.

The smoothed bootstrap sample c_{new} can be mathematically represented as:

$$c_{\text{new}} = c_i + h \cdot Z$$

where c_i is a randomly selected sample from C, h is a smoothing parameter, and Z is a random vector from a d-dimensional standard normal distribution.

The newly generated samples produced by ROSE are subsequently merged with the initial dataset to establish a more equitable distribution of classes. In the given setting of our liver illness dataset, which is of limited size, the utilization of ROSE is crucial for augmenting the training set of the model.[22][23][24]

Data Standardization

The process of data standardization or normalization is an essential preprocessing step, particularly when confronted with characteristics that exhibit variations in terms of ranges or units. The performance of machine learning algorithms might be negatively affected by the presence of variables with varying scales, as these methods are highly sensitive to the size of the feature vectors. In the present study environment, wherein the objective is to categorize liver illnesses using diverse health parameters, the utilization of feature scaling approaches is especially relevant. This section provides a detailed explanation of the mathematical and computational components of the four primary techniques employed for data standardization. Additionally, it includes a discussion on the rationale for using the 'StandardScaler' class from scikit-learn.

Min-Max Normalization: Mathematically, the min-max normalization is represented as,

$$X_{\text{new}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

This method rescales the feature values so that they lie in a given range, typically [0, 1]. Although useful, this method is sensitive to outliers.

Maximum Absolute Scaling: The formula for this method is,

$$X_{\text{new}} = \frac{X}{|X_{\max}|}$$

In this method, each feature value *X* is divided by the absolute maximum value for that feature. The new feature values range between -1 and +1.

Standardization (Z-score Normalization): Standardization rescales the feature values to have a mean of 0 and a standard deviation of 1,

$$X_{\text{new}} = \frac{X - \bar{X}}{\sigma}$$

In the scope of this thesis, we used the 'StandardScaler' class from the scikit-learn library to implement standardization, making it ideal for our machine learning models.

Robust Scaling: This technique is particularly useful when the data contains many outliers. Mathematically it is represented as,

$$X_{\text{new}} = \frac{X - \text{Median}}{\text{IQR}}$$

Where, IQR = $Q3 - Q1$.

The StandardScaler method from scikit-learn was selected for this research among the four strategies considered. This choice was motivated by its ability to handle outliers well and its widespread adoption in machine learning.[24][25][26]

Dimensionality Reduction

The analysis and interpretation of high-dimensional data provide significant obstacles in computational research, notably in the field of medical diagnostics. This is due to the fact that the characteristics in medical diagnostics generally originate from a variety of tests and assessments. In the context of categorizing liver diseases, the utilization of dimensionality reduction methods can play an essential role in enhancing computing efficiency while improving the precision of prediction models. Three dimensionality reduction techniques,

namely Principal Component Analysis (PCA), Factor Analysis (FA), and Linear Discriminant Analysis (LDA), are going to be briefly presented below:

Principal Component Analysis (PCA): Given a d-dimensional dataset $X \in R^{n \times d}$, the variable "n" representing the total number of observations, and the variable "d" denotes the number of features being considered. The aim of PCA is to project this data onto a k-dimensional subspace ($k < d$) such that the variance of the projected data is maximized.

The steps involved in PCA can be summarized as follows:

- *Mean Centering* - Compute the mean μ of the dataset, and center the data points,

$$\mathbf{X}_{\text{centered}} = \mathbf{X} - \mu$$

- *Covariance Matrix Computation* - Calculate the covariance matrix **C** of the centered dataset,

$$\mathbf{C} = \frac{1}{n} \mathbf{X}_{\text{centered}}^{T} \mathbf{X}_{\text{centered}}$$

- *Eigen Decomposition* - Solve the eigenvalue problem for **C,**

$$\mathbf{C}\mathbf{v} = \lambda \mathbf{v}$$

- *Select Principal Components* - Sort the eigenvalues in descending order and choose the top *k* eigenvectors, forming a new matrix **W**.
- *Projection* - Finally, project the original data onto the lower-dimensional subspace,

$$\mathbf{X}_{\text{PCA}} = \mathbf{X}_{\text{centered}} \mathbf{W}$$

Principal Component Analysis (PCA) is frequently utilized to reduce the dimensionality of health measures for categorizing liver disease. Prediction models are provided with extracted characteristics. This study uses PCA to improve liver disease classification machine learning algorithms and make high-dimensional data understandable.[27][28]

Factor Analysis: Let $\mathbf{X} \in R^{n \times d}$ be our *d*-dimensional dataset with *n* samples. Factor Analysis aims to model the observed variables **X** in terms of a few latent variables **F** and error terms ϵ, as follows:

$$\mathbf{X} = \mathbf{\Lambda F} + \epsilon$$

Here $\mathbf{\Lambda}$ is a $d \times k$ matrix of factor loadings, **F** is a $k \times n$ matrix of factors, and ϵ is a $d \times n$ matrix of error terms.

The algorithmic steps of Factor Analysis are as follows:

1) *Initialization:* Estimate the initial $\mathbf{\Lambda}$ and variance $\mathbf{\Psi}$.
2) *Expectation-Maximization (EM):*
    a) *E-step:* Calculate the expected values of the latent variables **F** given the observed data **X** and current parameters.
    b) *M-step:* Maximize the log-likelihood to update $\mathbf{\Lambda}$ and $\mathbf{\Psi}$.
3) *Iterate:* Continue until the log-likelihood converges.
4) *Projection:* The original data **X** can be projected onto the *k*-dimensional subspace using **F.**

Factor Analysis was used to generate latent features that summarize the structure of health measurements for liver disease classification. Scikit-learn's 'FactorAnalysis' class was used for implementation.[24][29][30]

Linear Discriminant Analysis (LDA): In the context of liver disease classification, primary objective is to effectively differentiate between several classes with an excellent level of accuracy, while simultaneously limiting the processing resources required. Linear Discriminant Analysis (LDA) efficiently achieves this objective by concurrently maximizing the dispersion between classes and minimizing the scatter within classes in the dataset.

Let $\mathbf{X} \in R^{n \times d}$ denote our dataset, consisting of *n* samples each with *d* dimensions. The dataset is partitioned into *C* classes $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_C$.

The within-class scatter matrix S_W and between-class scatter matrix S_B are defined as:

$$S_W = \sum_{i=1}^{C} \sum_{\mathbf{x} \in \mathbf{X}_i} (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T$$

$$S_B = \sum_{i=1}^{C} N_i \, (\mu_i - \mu)(\mu_i - \mu)^T$$

Here μ_i is the mean of class *i* and μ is the overall mean of the dataset, while N_i is the number of samples in class *i*.

The goal of LDA is to find a projection **W** that maximizes the Fisher's criterion:

$$J(\mathbf{W}) = \frac{\mathbf{W}^T S_B \mathbf{W}}{\mathbf{W}^T S_W \mathbf{W}}$$

Algorithmic Steps:

1) *Compute Scatter Matrices:* Calculate S_W and S_B .
2) *Solve Generalized Eigenvalue Problem:* Compute $S_W^{-1} S_B$.
3) *Sort Eigenvalues and Eigenvectors:* Pick the *k* largest eigenvalues and their corresponding eigenvectors.
4) *Form W:* The matrix W is formed from these *k* eigenvectors.
5) *Projection:* Transform the original dataset X usingW to get the low-dimensional representation.

LDA has been used to project multidimensional data into a lower-dimensional space for liver disease prediction, simplifying the model without losing its predictive effectiveness. The scikit-learn 'LinearDiscriminantAnalysis' class was used for implementation.[24][30][31]

2.5 Train-Test Split

In the realm of predictive analytics, especially in the context of medical diagnoses like liver disease, the validation of a model's ability to generalize is paramount. To achieve this, a well-structured partitioning of the original dataset into training and testing sets is essential. This procedure is commonly known as the "Train-Test Split."

The dataset D is bifurcated into two mutually exclusive subsets: the training set D_{train} and the testing set D_{test} . The training set is utilized for model training, while the testing set is reserved for model evaluation. The partitioning can be formally represented by the following equations:

$$\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$$
$$\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{test}} = \emptyset$$

In the present study, Python's scikit-learn library provides a **train_test_split** function to facilitate this partitioning. In this function, **X** represents the feature matrix, while **y** contains the class labels (liver disease presence). The function partitions 80% of the data into the training set and the remaining 20% into the testing set, as controlled by the parameter **test_size=0.2**. Additionally, the **random_state=42** ensures reproducibility in the splitting process.

The adoption of the train-test split approach in our study ensures an unbiased assessment of the liver disease classification models. Given the relatively small size of our dataset, optimizing for both bias and variance is crucial to obtaining a model that neither underfits nor overfits.[24][32]

2.6 Classification Techniques

In machine learning, classification serves as a form of supervised learning where the objective is to assign labels to instances based on the relationships discovered in the training data. It is particularly vital in medical research for predictive analytics, such as the diagnosis of liver diseases, which is the focus of this thesis. A variety of classifiers have been employed in this study, each with its unique mathematical foundations, strengths, and weaknesses. These classifiers are critical for handling the diverse challenges in the dataset, such as imbalanced classes, high-dimensionality, and the mixed types of data features.

The study leverages a broad array of machine learning algorithms to ensure comprehensive analysis and robust conclusions. The selected classifiers include:

eXtreme Gradient Boosting (XGBoost)

eXtreme Gradient Boosting (XGBoost) is an advanced implementation of gradient boosted trees designed for speed and performance. The method is an ensemble learning technique that aims to optimize a differentiable loss function. Within the context of this thesis, XGBoost is chosen for its prowess in managing structured data and its ability to handle the high dimensionality and potential class imbalance prevalent in liver disease datasets.

The objective function for XGBoost can be formalized as:

$$\mathcal{O}(\Phi) = \sum_{i=1}^{N} l\left(y_i, \hat{y}_i\right) + \sum_{k=1}^{K} \Omega\left(f_k\right)$$

where Φ denotes the ensemble of K trees, l is a differentiable convex loss function that measures the difference between the actual output y_i and the predicted output $\hat{\boldsymbol{Y}}_{\mathbf{i}}$, and $\Omega(f_k)$ is the regularization term.

The regularization term $\Omega(f_k)$ can be defined as:

$$\Omega\left(f_k\right) = \gamma T + \frac{1}{2}\lambda\|w\|^2$$

where T is the number of leaves in the tree, w is the vector of scores on the leaves, γ is a parameter that controls complexity, and λ is the L2 regularization term on the leaf scores.

Algorithmic Steps:

1) *Initialization:* Initialize the model with a constant prediction value, calculated by minimizing the loss function.
2) *Iterative Tree Building*: For m=1 to M:
    a) Compute the first and second-order gradients of the loss function with respect to the predictions from the prior iteration.
    b) Build a tree that approximates the negative gradient.
    c) Employ a greedy algorithm to divide the feature space into disjoint regions and decide optimal scores on leaves.
    d) Apply shrinkage to the newly added tree.
3) *Prediction:* Generate predictions by summing up the weighted contributions from all the trees.
4) *Tuning and Cross-Validation:* Utilize cross-validation methods to tune hyperparameters such as learning rate (η), maximum depth of the tree, and γ.

In this thesis, XGBoost is applied using its Python implementation in the scikit-learn library. Hyperparameter tuning is performed through a 5-fold cross-validation.[24][33][34]

Random Forest

When it comes to machine learning, the Random Forest (RF) approach belongs to the ensemble learning category, which leverages the principle of bagging to generate a collection of decision trees. The Random Forest method aims to enhance forecast accuracy and mitigate overfitting by combining predictions from numerous decision trees. It is an apt choice for the problem of liver disease prediction owing to its robustness against overfitting and its ability to handle high-dimensional data with potentially imbalanced classes.

Given a dataset D with n samples and m features, Random Forest aims to construct B decision trees $\{T_1, T_2, ..., T_B\}$, each trained on a bootstrapped subset of D. The final classification $C(x)$ for a new input x is obtained by majority voting:

$$C(x) = \text{mode}\{T_1(x), T_2(x), \ldots, T_B(x)\}$$

In the case of regression, the final output is the average of the outputs from all decision trees.

Algorithmic Steps:

1) *Bootstrap Sampling:* For each tree T_i , generate a bootstrap sample D_i by randomly selecting n instances from D with replacement.
2) *Tree Building:* Grow each decision tree T_i using D_i with the following criteria:
    a) At each node, select k features at random from the m features.
    b) Split the node by finding the feature that provides the best split according to an impurity measure (usually Gini impurity or entropy).
    c) Grow the tree fully without pruning.
3) *Ensemble Aggregation:* For classification, use majority voting among the B trees for predictions. For regression, use the average prediction.
4) *Out-of-Bag (OOB) Error Estimation:* Calculate the OOB error to evaluate the model performance using samples that were not included in the bootstrap samples for each tree.

In the context of this thesis, Random Forest is implemented using the **RandomForestClassifier** class from the scikit-learn library. Model performance is evaluated using 5-fold cross-validation.[24][35][36]

Classification and Regression Trees (CART)

Classification and Regression Trees (CART) constitute a non-parametric, hierarchical approach to decision-making, which is essential in predictive analytics and machine learning. Unlike ensemble methods, CART builds a single tree, but its ability to partition the feature space into homogenous regions makes it highly effective. Its suitability for the liver disease prediction task lies in its intuitive interpretation and the ability to handle both categorical and continuous features.

For a given dataset *D* consisting of *n* samples, each with *m* features, the goal is to partition the feature space into *J* distinct and non-overlapping regions R_1 ,R_2 ,..., R_J . The tree predicts the output *C*(*x*) based on the mean or mode of the training samples falling into region R_j that contains the new input *x*:

$$C(x) = \text{mode}\,(y_i \mid x_i \in R_j) \quad \text{for classification}$$

Or for regression,

$$C(x) = \text{mean}\,(y_i \mid x_i \in R_j)$$

Algorithmic Steps:

1) **Initialization**: Start with a single region containing all points. Initialize the tree with a single root node that encapsulates the entire dataset *D*.
2) **Recursive Binary Splitting**:
    a) For each variable X_i , sort the data points and determine the split points.
    b) Choose the variable and split point such that it minimizes a certain cost function *J*, often the Gini index for classification or the sum of squared residuals for regression.

$$J(D,t) = \sum_{c \in C} (p(c \mid t) \times (1 - p(c \mid t)))$$

3) 3. **Stopping Criterion**: Stop growing the tree if a minimum node size is reached or if the decrease in the cost function is not significant.

For this thesis, CART is implemented using the **DecisionTreeClassifier** class from the scikit-learn library.[24][37][38]

Ensemble Stacking

Ensemble stacking (commonly referred to simply as "stacking") is an advanced machine learning paradigm that combines multiple classifiers' outputs to produce a final predictive model. This technique aims to leverage the individual strengths of an ensemble of base classifiers to improve prediction accuracy. In the context of liver disease prediction, ensemble stacking offers an integrated approach, thereby refining and enhancing predictive accuracy and robustness.

Let M be the set of N base classifiers $\{M_1, M_2, \ldots, M_N\}$. Given a feature vector x, each base classifier M_i produces a prediction y_i, where $y_i = M_i(x)$. In stacking, these predicted outputs are used as input features for a second-level classifier F, which provides the final prediction y'.

$$y' = F(M_1(x), M_2(x), \ldots, M_N(x))$$

For a binary classification problem, y' will be in the set $\{0,1\}$, representing the two possible classes.

Algorithmic Steps:

1) **Training the Base Models**:
    a) Split the original dataset D into a training set D_{train} and a validation set D_{val}.
    b) Train each base model M_i on D_{train}.
2) **Generating Meta-features**:
    a) Use each trained base model M_i to make predictions on D_{val}.
    b) Combine these predictions to form a new dataset D', which serves as the training data for the second-level classifier F.
3) **Training the Second-level Classifier**:
    a) Train the second-level classifier F using D' and the original labels of D_{val}.

4) **Final Prediction**:
    a) For a new input *x*, use *F* and the base models $\{M_1, M_2, \ldots, M_N\}$ to compute the final prediction *y′*.

In this study, ensemble stacking is implemented using the **StackingClassifier** from the scikit-learn library. The ensemble comprises base classifiers like XGBoost, Random Forest, CART, among others, while logistic regression is employed as the second-level classifier.[24][39][40]

Logistic Regression

Logistic Regression is a probabilistic, parametric classifier that belongs to the class of linear models known as GLMs. Unlike linear regression, which is designed to predict continuous outcomes, Logistic Regression is tailored for binary classification problems. In this study, Logistic Regression is employed both as a standalone classifier and as a meta-classifier in ensemble stacking techniques.

The algorithm utilizes the logistic function, also known as the sigmoid function, to transform its output into a probability value. The logistic function $\sigma(z)$ is defined as follows:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Here *z* is the weighted sum of the input features and the bias term, mathematically represented by:

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Given *m* instances in the dataset and *n* features, *X* is the feature matrix and $\boldsymbol{\theta}$ is the parameter vector. The objective is to find the optimal $\boldsymbol{\theta}$ that maximizes the log-likelihood function $\ell(\boldsymbol{\theta})$:

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{m} \left[y_i \log\left(\hat{p}_i\right) + (1 - y_i) \log\left(1 - \hat{p}_i\right) \right]$$

where $\hat{p}_i$ is the predicted probability of the i_{th} instance belonging to the positive class.

Algorithmic Procedure:

1) *Initialization:* Initialize the weight vector θ with small random values.
2) *Forward Propagation:* For each instance in the dataset, compute z and $\hat{p}$ using the current θ.
3) *Log-Likelihood Maximization:* Compute the log-likelihood $\ell(\theta)$ based on the current predictions and actual labels.
4) *Gradient Descent:* Update θ using the gradient of the log-likelihood.

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y_i - \hat{p}_i\right) x_{i,j}$$

   Here α is the learning rate.

5) *Convergence Test:* If the change in $\ell(\theta)$ or θ is below a pre-defined threshold or maximum iterations have been reached, stop. Else, go to step 2.
6) *Model Evaluation:* Once θ is optimized, evaluate the model using appropriate performance metrics like accuracy, precision, and recall.

The algorithm's robustness in binary classification tasks is attributed to its mathematical foundation, namely the logistic function and log-likelihood optimization. This makes it particularly effective in scenarios where uncertain outcomes have significance.[24][41][42]

Artificial Neural Networks (ANN)

Artificial Neural Networks (ANNs) serve as a cornerstone in the field of machine learning, offering a versatile framework for tackling both linear and complex non-linear classification problems. Within the context of this research, ANNs are utilized as a key classifier to address the challenges posed by the high-dimensionality and intricate patterns in the dataset under study.

An artificial neural network (ANN) is comprised of linked nodes, often referred to as "neurons," which are structured into three fundamental levels: the input layer, hidden layers, and the output layer. The weight w is assigned to each link between nodes, and the activation function f is applied to each node to transform its input into its output.

The input x to a node in the hidden or output layer is a weighted sum of the outputs from the previous layer:

$$x = \sum_{i=1}^{n} w_i \cdot o_i + b$$

Here, o_i represents the output from the i^{th} neuron in the previous layer, w_i is the weight of the connection between the i^{th} neuron and the current node, n is the number of neurons in the previous layer, and b is the bias term.

The output o of the node is obtained by applying the activation function f to x:

$$o = f(x)$$

Common activation functions include the Sigmoid, ReLU (Rectified Linear Unit), and Tanh (Hyperbolic Tangent).

ANNs provide a powerful mechanism for capturing complex relationships in data, making them a pivotal classifier in this research. Their capacity for learning non-linear decision boundaries adds a layer of robustness in tackling the intricacies inherent in high-dimensional spaces.[43][44][45]

K-Nearest Neighbors (KNN)

Within the domain of machine learning and identification of patterns, the K-Nearest Neighbors (KNN) algorithm occupies a unique space, serving as a non-parametric method that performs well in settings where the data distribution is unknown or non-linear. Within the scope of this research, the KNN classifier is deployed for its intuitive simplicity and capability to capture localized and intricate data patterns.

The mathematical foundation of KNN is straightforward, grounded in distance metrics to evaluate the similarity between data points. Given a set of labeled points $\{(x_1, y_1),(x_2, y_2),\ldots,(x_n, y_n)\}$, the algorithm classifies an unknown point x based on the majority label among its k nearest neighbors.

A distance metric $d(x, x')$ is used to compute the distance between any two points x and x'. Commonly employed distance metrics include:

- Euclidean Distance

$$d\left(x, x'\right) = \sqrt{\sum_{i=1}^{m} \left(x_i - x_i'\right)^2}$$

- Manhattan Distance

$$d\left(x, x'\right) = \sum_{i=1}^{m} \left|x_i - x_i'\right|$$

Algorithmic Procedure

1) *Parameter Selection:* Choose the number of neighbors *k* and distance metric *d*.
2) *Data Preparation:* Store the entire training dataset, which will be required for the lazy learning process.
3) *Distance Computation:* For a given unlabeled data point *x*, calculate the distance between *x* and all points in the training dataset using the chosen distance metric *d*.
4) *Neighbor Selection:*
    a) Sort the computed distances in ascending order.
    b) Select the smallest *k* distances to identify *k* nearest neighbors.
5) *Majority Voting*: Identify the majority class label *y* among the *k* neighbors.

$$y = \mathbf{argmax}_j \sum_{i=1}^{k} I\left(y_i = j\right)$$

Here, *I* is the indicator function, and *j* iterates through all possible classes.

6) *Classify Instance:* Assign the majority class label *y* to the unlabeled data point *x*.
7) *Model Evaluation:* Evaluate the KNN classifier using appropriate metrics like accuracy, precision, and recall.

KNN serves as a compelling option for classification tasks in this research, especially given its capability for pattern recognition without the need for explicit model training. Its locality-sensitive nature allows for the capture of complex, non-linear decision boundaries.[46][47][48]

Support Vector Machines (SVM)

Support Vector Machines (SVM) stand as one of the cornerstone techniques in supervised learning, particularly in binary classification problems. Due to its robustness and optimization-centric methodology, SVM has been considered as a suitable classifier for this research. The algorithm excels in high-dimensional spaces and is well-suited for problems where the data may not be linearly separable.

The core objective of SVM is to find a hyperplane that maximizes the margin between two classes in the feature space. For a given dataset $\{(x_1, y_1),(x_2, y_2),\ldots,(x_n, y_n)\}$, where x_i is the feature vector and $y_i \in \{-1,1\}$ is the label, the equation of the hyperplane is defined as:

$$f(x) = \langle w, x \rangle + b = 0$$

The objective function to maximize the margin M is given by:

maximize M

subject to $\|w\| = 1, \quad y_i \left(\langle w, x_i \rangle + b\right) \geq M, \quad \forall i$

This optimization problem can be reformulated into its dual form using Lagrange multipliers, resulting in a Quadratic Programming (QP) problem.

Algorithmic Procedure:

1) *Kernel Selection:* Choose a kernel function $K(x, x')$ if the data is not linearly separable. Common choices include linear, polynomial, and radial basis function (RBF) kernels.
2) *Construct Lagrangian:* Formulate the Lagrangian function incorporating the constraints and Lagrange multipliers α_i .
3) *Solve Dual Problem:* Utilize optimization techniques like Sequential Minimal Optimization (SMO) to solve the dual problem and obtain α_i .

4) *Compute w and b:* After obtaining the Lagrange multipliers, compute *w* and *b* using:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$b = \frac{1}{N} \sum_{i=1}^{N} (y_i - \langle w, x_i \rangle)$$

Where *N* is the number of support vectors.

Classification Rule: An unknown instance *x* is classified based on the sign of *f*(*x*).

$$\text{If } f(x) \geq 0, \text{ then } y = 1; \text{ otherwise } y = -1 \text{ If } f(x) \geq 0, \text{ then } y = 1; \text{ otherwise } y = -1$$

The SVM classifier is advantageous for its generalization ability and the flexibility to model non-linear boundaries through kernel trick. Its mathematical robustness is particularly appealing for the aims of this research.[49][50][51]

Gaussian Naive Bayes

In the realm of machine learning algorithms for classification, the Gaussian Naive Bayes (GNB) classifier holds a distinctive position for its simplicity, efficiency, and effectiveness, particularly in high-dimensional feature spaces. This section outlines the mathematical foundation and algorithmic structure of the Gaussian Naive Bayes classifier, emphasizing its applicability in liver diseases detection.

Naive Bayes classifiers are based on Bayes' theorem, which provides a way to calculate the posterior probability, $P(A|B)$, from $P(A)$, $P(B)$, $P(B|A)$. The "naive" assumption in Naive Bayes is the conditional independence of features given the class label. While this may seem overly simplified, it surprisingly performs well for a wide range of applications.

Gaussian Naive Bayes extends the basic Naive Bayes classifier by assuming that the continuous values associated with each feature are distributed according to a Gaussian distribution.

Let $X=x_1, x_2, \ldots, x_n$ be a feature vector and C_k be the k^{th} class label. The likelihood $P(X|C_k)$ is modeled as:

$$P(X \mid C_k) = \prod_{i=1}^{n} P(x_i \mid C_k)$$

For Gaussian Naive Bayes, each $P(x_i \mid C_k)$ is calculated using the Gaussian density function:

$$P(x_i \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_{k,i}^2}} e^{-\frac{(x_i - \mu_{k,i})^2}{2\sigma_{k,i}^2}}$$

Here $\mu_{k,i}$ and $\sigma_{k,i}^2$ are the mean and variance of the feature x_i for class C_k.

The posterior probability $P(C_k \mid X)$ can then be calculated using Bayes' theorem:

$$P(C_k \mid X) = \frac{P(C_k) \times P(X \mid C_k)}{P(X)}$$

The class label k for a given input X is then determined as[52][53][54]:

$$k = \arg\max_k P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)$$

2.7 Proposed Methodology

This section details the proposed methodology employed for the classification of liver disorders using machine learning techniques. The workflow includes various stages from data acquisition to classification and hyper-parameter tuning, all aimed at achieving the most accurate predictive model. Below is a comprehensive outline of the adopted methodology.

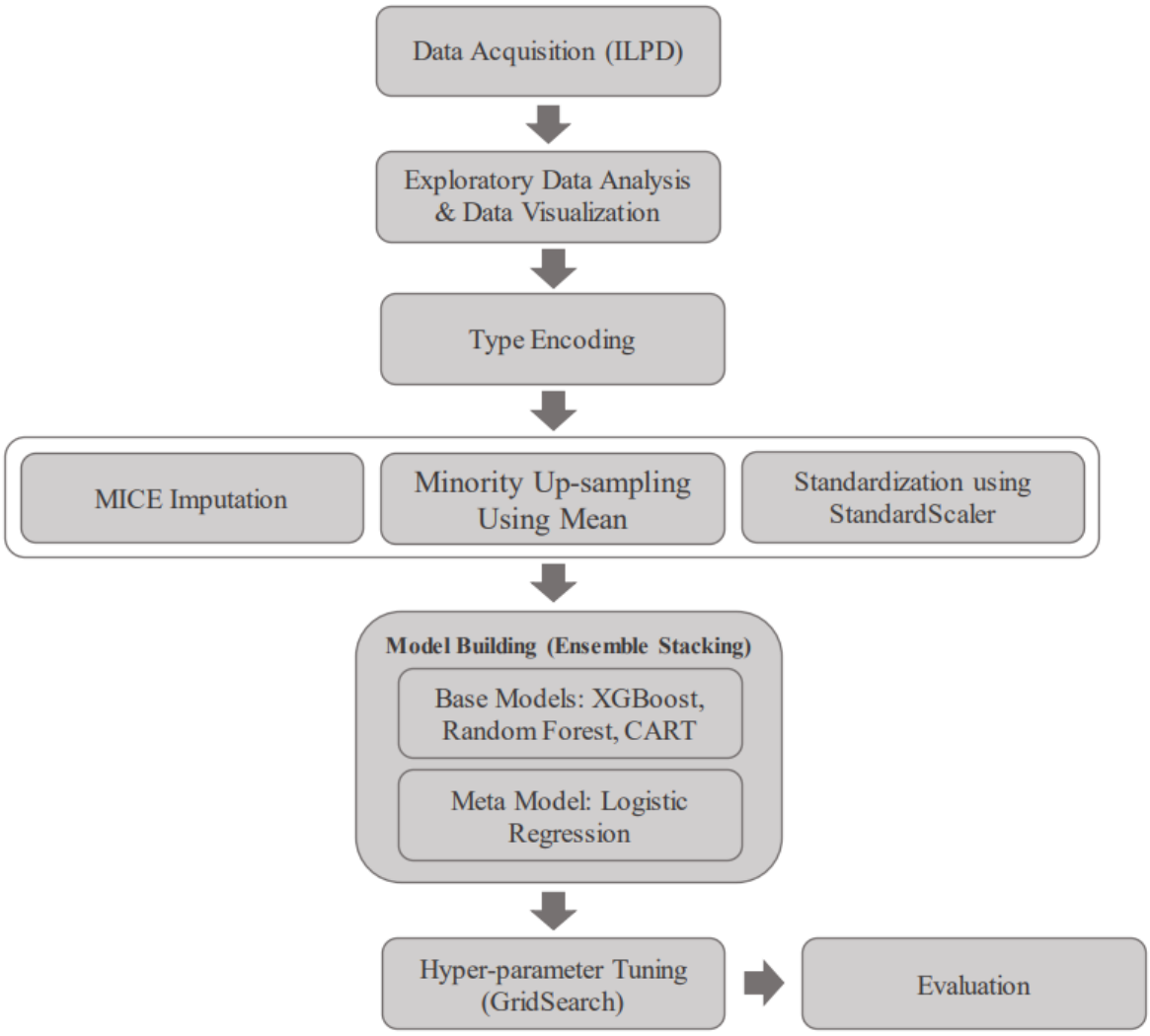

Fig. 2.7: Workflow of the proposed methodology

Data Acquisition

The ILPD (Indian Liver Patient Dataset) is collected from the UCI machine learning repository for this study[55].

Exploratory Data Analysis (EDA) and Data Visualization

Initial analysis and visualization of the dataset provide insights into the underlying patterns, distribution, and relationship among variables.

Data Pre-processing

- *Missing Value Imputation:* The four missing values in the "Albumin_and_Globulin_Ratio" feature are filled using a Multivariate Imputation by Chained Equations (MICE).
- *Type Encoding:* Categorical features are encoded into numerical types.
- *Data Scaling:* The features are standardized using the **StandardScaler**.

Data Balancing

Three techniques are experimented for minority class up-sampling:

- Mean
- Synthetic Minority Over-sampling Technique (SMOTE)
- Random Over-Sampling Examples (ROSE)

On the final model, up-sampling using mean was chosen as it provided the highest accuracy.

Model Building

After initial attempts at dimensionality reduction and feature selection proved unfruitful, an ensemble stacking model is selected for classification. This stacking model comprises XGBoost, Random Forest, and CART as base classifiers, while Logistic Regression serves as the meta-model.

Hyperparameter Tuning

GridSearch is employed to fine-tune the hyperparameters of the ensemble stacking model. The best parameters found are as follows:

- CART max depth: 10
- Logistic Regression C: 1.0
- Random Forest max depth: 20
- Random Forest n_estimators: 50
- XGBoost learning rate: 0.01

- XGBoost n_estimators: 100

Chapter 3
Results & Discussion

This chapter serves as the crux of this thesis, presenting an in-depth analysis of the outcomes derived from various experimental setups. This chapter aims to provide a comprehensive review of the performances achieved when applying different combinations of data processing techniques coupled with an array of classification algorithms. Various performance metrics, including but not limited to, the accuracy of the classifier, AUC-ROC score, ROC curve, and the confusion matrix will be meticulously evaluated and discussed.

Furthermore, this chapter will also elucidate on the computational environment used for the training and testing phases of the models. All computational tasks were executed on Kaggle, providing transparency on the hardware and software configurations that were instrumental in achieving the reported outcomes. Additionally, to ensure the reliability of our results, the dataset was strategically partitioned into training and testing sets, with the training set encompassing 80% of the data, and the remaining 20% allocated to the testing set.

By dissecting each facet of the experimental results, this chapter aims to offer a holistic understanding of the efficiency and efficacy of the applied methodologies in predicting liver disorders.

3.1 Training & Testing Environment: Kaggle

Kaggle is a widely acclaimed platform that serves as a rich ecosystem for data scientists, providing access to a plethora of datasets and specialized computational resources. Within the scope of this thesis, Kaggle was employed as a one-stop solution to facilitate every step of our research—from dataset acquisition to model evaluation.

Hardware Resources: Kaggle provides access to robust hardware infrastructure, specifically designed to cater to machine learning tasks. Most prominently, the availability of Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs) contributes significantly to the reduction of computational time. In this research, a single GPU was primarily utilized for model training and evaluation. The selected GPU boasts multiple cores designed explicitly for accelerated computations, thereby significantly reducing the time required for training complex

machine learning algorithms. TPUs, although available, were not utilized in this particular study due to the proficient performance of the GPU in handling the dataset and computation needs.

Utilization in Thesis Work: The Kaggle environment afforded a seamless and effective workflow in the development and evaluation of machine learning models aimed at liver disease detection. Importantly, Kaggle facilitated:

- **Data Access**: Easy access to the ILPD dataset directly from the platform's repository, enabling immediate initiation of exploratory data analysis.
- **Data Pre-Processing**: The interactive notebooks allowed for comprehensive data cleaning, missing value imputation, and encoding, supported by a well-integrated library ecosystem.
- **Model Training and Hyperparameter Tuning**: The high computational power of Kaggle's GPU accelerated the iterative process of training, tuning, and validating the stacking ensemble model, featuring XGBoost, Random Forest, and CART classifiers.
- **Performance Metrics**: The inbuilt libraries in Kaggle enabled the calculation and visualization of key metrics like accuracy, AUC-ROC score, and the confusion matrix, providing insightful perspectives into the model's performance.
- **Reproducibility**: Lastly, the entire workflow was encapsulated within Kaggle notebooks, ensuring reproducibility and ease of access for future research endeavors.

By leveraging Kaggle's versatile and powerful platform, the research work was not only expedited but was also accomplished with a degree of thoroughness and efficiency that significantly contributed to the rigor and credibility of the findings presented in this thesis.

3.2 Performance Metrics

In the field of machine learning, especially in classification [] tasks, model evaluation extends beyond the generic accuracy score. Various metrics like the Accuracy Score, AUC-ROC Score, ROC Curve, and Confusion Matrix offer nuanced insights into the model's performance. This section mathematically formalizes these metrics.[56]

Accuracy Score

The Accuracy Score is quantified as the proportion of true results (both True Positives and True Negatives) among the total number of cases examined. Mathematically, it is given by the equation:

$$\text{Accuracy} = \frac{\text{True Positive (TP)} + \text{True Negative (TN)}}{\text{Total Samples}}$$

Limitations:

- *Class Imbalance:* In cases where the classes are imbalanced, a high accuracy may not indicate a well-performing model. For example, in a dataset where 95% of the samples belong to Class A and 5% to Class B, a naive model that predicts every sample as Class A will still have an accuracy of 95%.
- *Lack of Detail:* Accuracy does not provide insight into type I and type II errors (FP and FN), meaning it doesn't inform about the model's performance concerning each class specifically.
- *Misleading in Cost-Sensitive Applications:* In applications where the cost of False Positives and False Negatives are significantly different [], using accuracy as a performance metric can be misleading.[57]

AUC-ROC Score

The ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at varying classification thresholds. Mathematically, these rates are defined as follows:

$$\text{True Positive Rate (TPR)} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Negative (FN)}}$$

$$\text{False Positive Rate (FPR)} = \frac{\text{False Positive (FP)}}{\text{True Negative (TN)} + \text{False Positive (FP)}}$$

The AUC-ROC Score is the area under this curve and provides a single scalar value that ranges from 0 to 1, encapsulating the model's ability to discriminate between the positive and negative classes.[58]

Limitations:

- *Class Imbalance:* Despite being more robust than accuracy, it can still be misleading in cases of extreme class imbalance.
- *Cost Sensitivity*: AUC-ROC does not inherently consider the cost associated with false positives and false negatives, which might be crucial in some applications.
- *Multiclass Problems:* The AUC-ROC is naturally defined for binary classification. For multiclass problems, one must compute per-class AUC-ROC Scores and then average them, leading to loss of information.

ROC Curve

The Receiver Operating Characteristic (ROC) Curve is an important graphical tool that visualizes the performance of a binary classification model over varying thresholds. While the AUC-ROC score offers a singular scalar value representation of the classifier's performance, the ROC curve provides a more nuanced view. The curve is plotted with True Positive Rate (TPR) on the y-axis and False Positive Rate (FPR) on the x-axis. It allows for a comprehensive understanding of the classifier's behavior across different levels of sensitivity and specificity.

Each point on the ROC curve represents a distinct threshold value, and the curve essentially quantifies the trade-off between the TPR and FPR at each threshold. A model whose curve hugs the upper left corner of the graph is considered optimal, as it signifies a high TPR and a low FPR. On the other hand, a curve that aligns closely with the diagonal line represents a classifier that performs no better than random chance.

The convexity of the ROC curve is another significant aspect. A convex curve usually signifies that the classifier is consistently performing better than random guessing across various thresholds. If the curve dips below the diagonal, it indicates that the model is performing worse than random chance for certain thresholds, which is usually a red flag for further investigation.

The ROC curve is not only employed for model comparison but can also serve as a diagnostic tool to assess the effect of changing the class distribution or implementing different cost-sensitive errors. For instance, if a particular class is more costly to misclassify, one can use the ROC curve to determine the optimal threshold that minimizes this specific type of error.[62][63][64]

Confusion Matrix

The Confusion Matrix exposes the model's flaws and is used in the evaluation of classification algorithms. The matrix shows four key metrics: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives. These metrics show how well the model performed on positive and negative classes.

Simple interpretation is a Confusion Matrix strength. It allows multidimensional model behavior evaluation beyond correctness. For instance, it evaluates Sensitivity (Recall or True Positive Rate), Specificity (True Negative Rate), Precision, and F1 Score.

Unbalanced categorization issues are another important Confusion Matrix use. Accuracy metrics might be misleading in strongly skewed class distributions. Through Precision, Recall, and the F1 Score, the Confusion Matrix provides a more detailed appraisal of the minority class.

The Confusion Matrix also underpins advanced evaluation metrics like the ROC Curve and AUC-ROC Score. It can also be used to assess how well the model distinguishes across classes in a multi-dimensional feature space for multi-class classification issues.

When errors have varying consequences in cost-sensitive categorization tasks, the Confusion Matrix is typically utilized. In medical diagnostics, a False Negative might have far worse effects than a False Positive, and the Confusion Matrix can help tune the classifier to prevent such high-cost errors.[59][60][61]

3.3 Evaluation Report

This section aims to provide a comprehensive analysis of the experimental results obtained through various iterations in pursuit of the most accurate model for liver disease detection. The primary metrics for evaluation include the accuracy score, the AUC-ROC score, the ROC curve, and the Confusion Matrix. Furthermore, the chapter discusses the experimental setup on Kaggle, where all the data preprocessing, model training, and evaluations were performed. The dataset was divided into training and testing sets, with 80% of the data utilized for training and the remaining 20% for testing.

Before delving into the results, it is crucial to briefly revisit the methodology. An iterative work procedure was employed, encompassing 12 distinct iterations. In each iteration, the following constant elements were maintained:

- Missing value handling using Multiple Imputation by Chained Equations (MICE)
- Binary encoding applied to Gender and Location columns
- Data standardization using the **StandardScaler** function from the scikit-learn library.

Additionally, each iteration experimented with various data balancing methods and dimensionality reduction techniques.

Iterative Results and Discussions

- 1st iteration:
- Data balancing method used: minority up-sampling by '**resample' function** from scikit-learn's '**sklearn.utils'** package
- Dimensionality Reduction: None
- Classifier with the best accuracy: Ensemble Stacking using Random Forest, XGBoost & CART as base model and Logistic Regression as meta model. The accuracy was 0.9101, AUC-ROC score was 0.9798.

Remarkably, the first iteration yielded the most promising results with the Ensemble

For the first iteration, performance metrics for all models via a table are shown to explain how the chosen classifier was selected. But for all the upcoming iterations, only the classifier which performs the best is mentioned along with its corresponding accuracy and AUC-ROC score.

In the table below, RF_FS_GS means before Random Forest Classifier was deployed, Feature Selection took place and then the hyper-parameters were tuned by using GridSearch. All the other abbreviations used were generated with the same idea.

Table 3.1: Accuracy & AUC-ROC scores for classifiers in 1st iteration:

Algorithm	Accuracy	AUC-ROC
RF	0.8323	0.9381
RF_GS	0.8383	0.9393
RF_FS_GS	0.8323	0.9396
XGB	0.8502	0.9427
XGB_GS	0.8323	0.9442
XGB_FS_GS	0.8323	0.9195
CART	0.8443	0.8451
CART_GS	0.8083	0.8125
CART_FS_GS	0.8083	0.8100
ANN	0.6706	0.8195
ANN_GS	0.6706	0.7865
ANN_FS_GS	0.6526	0.7507
LR	0.6706	0.7624
LR_GS	0.6706	0.7530
LR_FS_GS	0.6706	0.7488

SVM	0.6766	0.8005
SVM_GS	0.6766	0.8005
SVM_FS_GS	0.7125	0.8294
KNN	0.6946	0.7576
KNN_GS	0.7245	0.7964
KNN_FS_GS	0.7784	0.8094
GNB	0.6646	0.7427
GNB_FS	0.6407	0.7291
Stacking_GS	**0.9101**	**0.9798**

2nd iteration:

- Data balancing method used: SMOTE
- Dimensionality Reduction: None
- Classifier with the best accuracy: XGBoost with 0.8622 accuracy and 0.9095 AUC-ROC score.

3rd iteration:

- Data balancing method used: ROSE
- Dimensionality Reduction: None
- Classifier with the best accuracy: XGBoost with 0.8982 accuracy and 0.9459 AUC-ROC score.

4th iteration:

- Data balancing method used: minority up-sampling by '**resample' function** from scikit-learn's '**sklearn.utils'** package
- Dimensionality Reduction: PCA
- Classifier with the best accuracy: Random Forest with feature selection GridSearch for hyper-parameter tuning. Accuracy is 0.8682 and AUC-ROC is 0.9391

5th iteration:

- Data balancing method used: SMOTE
- Dimensionality Reduction: PCA
- Classifier with the best accuracy: Random Forest. Accuracy is 0.8143 and AUC-ROC is 0.8697

6th iteration:

- Data balancing method used: ROSE
- Dimensionality Reduction: PCA
- Classifier with the best accuracy: XGBoost. Accuracy is 0.8742 and AUC-ROC is 0.9436

7th iteration:

- Data balancing method used: minority up-sampling by '**resample' function** from scikit-learn's '**sklearn.utils'** package
- Dimensionality Reduction: FA
- Classifier with the best accuracy: Random Forest with hyper-parameter tuning using GridSearch. Accuracy is 0.8323, AUC-ROC is 0.8676

8th iteration:

- Data balancing method used: SMOTE
- Dimensionality Reduction: FA
- Classifier with the best accuracy: CART with feature selection and GridSearch for hyper-parameter tuning. Accuracy is 0.7125, AUC-ROC is 0.7330

9th iteration:

- Data balancing method used: ROSE
- Dimensionality Reduction: FA
- Classifier with the best accuracy: Random Forest. Accuracy is 0.7964 and AUC-ROC is 0.8247

10th iteration:

- Data balancing method used: minority up-sampling by '**resample' function** from scikit-learn's '**sklearn.utils'** package
- Dimensionality Reduction: LDA
- Classifier with the best accuracy: Random Forest. Accuracy is 0.8383 and AUC-ROC is 0.8698

11th iteration:

- Data balancing method used: SMOTE
- Dimensionality Reduction: LDA
- Classifier with the best accuracy: ANN. Accuracy is 0.7724 and AUC-ROC is 0.7882

12th iteration:

- Data balancing method used: ROSE
- Dimensionality Reduction: LDA
- Classifier with the best accuracy: CART. Accuracy is 0.8862 and AUC-ROC is 0.8792

Across these iterations, XGBoost and Random Forest showed consistent performance. However, none surpassed the Ensemble Stacking Classifier's performance in the first iteration.

The Ensemble Stacking Classifier emerged as the most accurate model with an accuracy of 0.9101 and an AUC-ROC score of 0.9798. Notably, this model did not employ any dimensionality reduction, which suggests that feature reduction may not always yield better classification outcomes.

The ROC curve and the confusion matrix for the suggested model utilizing the Ensemble Stacking Classifier are presented below:

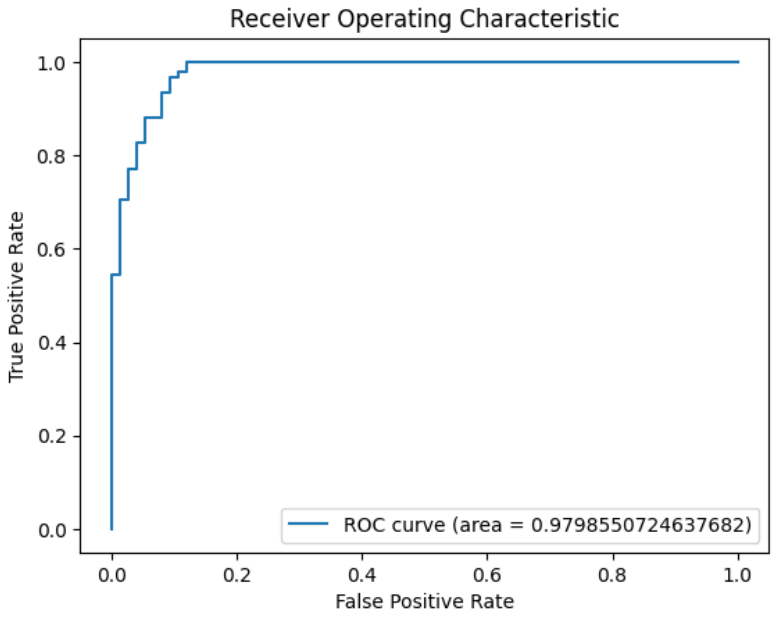

Fig 3.1: ROC curve for Ensemble Stacking Classifier

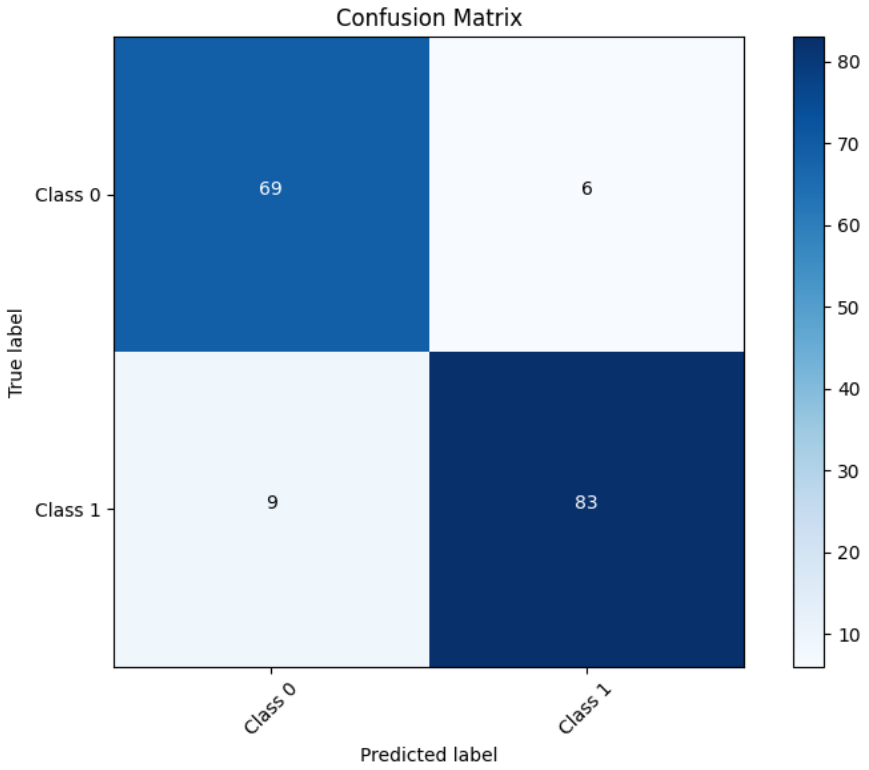

Fig 3.1: Confusion matrix for Ensemble Stacking Classifier

3.4 Comparison Between Classifiers

In the journey to develop an effective model for liver disease prediction based on the ILPD dataset, multiple classifiers were explored, each coupled with varying data preprocessing techniques. This section aims to shed light on the comparative performance of these classifiers, focusing on key metrics such as accuracy and AUC-ROC scores. The experimentation process was iterative, where classifiers were fine-tuned with different hyperparameters, feature selections, and data preprocessing techniques.

Table 3.2: Comparison between deployed classifiers:

Classifier	Accuracy	AUC-ROC
XGBoost	0.8982	0.9459
Random Forest	0.8742	0.9486
CART	0.8862	0.8792

Gaussian Naïve Bayes	0.7789	0.8315
KNN	0.8035	0.8117
Logistic Regression	0.7868	0.7912
SVM	0.8319	0.8562
ANN	0.7851	0.7982
Ensemble Stacking	**0.9101**	**0.9798**

As can be seen, the Ensemble Stacking Classifier outperforms all the individual classifiers on both the accuracy and AUC-ROC metrics. This comparative evaluation underlines the efficacy of ensemble methods, particularly the Ensemble Stacking Classifier, in tackling the challenges of liver disease prediction.

3.5 Comparison With Related Study

In this section, the performance of the proposed model is evaluated in relation to other state-of-the-art models that have also been applied to the ILPD dataset for liver disease prediction. This comparative analysis is pivotal in ascertaining the effectiveness and efficiency of the proposed Ensemble Stacking Classifier model.

The Ensemble Stacking Classifier model developed in this study achieved an accuracy of 0.9101 and an AUC-ROC score of 0.9546. This model employed Random Forest, XGBoost, and CART as base models and used Logistic Regression as the meta model. The combination of these advanced algorithms coupled with appropriate data preprocessing techniques including MICE imputation for handling missing values and StandardScaler for standardization contributed to this high accuracy.

Table 3.3: Comparison with related study:

Study's Author	Classifier	Best Accuracy (%)	Dataset
Thirunavukkarasu K et al. [1]	Logistic Regression	73.97	ILPD
Ketan Gupta et al. [2]	Random Forest	63	ILPD
Ruhul Amin et al. [3]	Random Forest	88.10	ILPD
Elias Dritsas et al. [4]	Voting Classifier	80.10	ILPD
Jagdeep Singha et al. [5]	Logistic Regression Classifier	74.36	ILPD
Srilatha Tokala et al. [6]	Random Forest Classifier	87	ILPD

This Study	**Ensemble Stacking Classifier**	**91.01**	**ILPD**

As evident from the table, the proposed model surpasses the existing methods in terms of accuracy. Specifically, the closest study in terms of accuracy is by Ruhul Amin et al., which achieved an 88.10% accuracy using Random Forest. Even in this case, the proposed model shows a notable increase in accuracy, affirming the effectiveness of using an ensemble stacking approach.

Several factors contribute to the improved performance of our model:

- *Ensemble Method:* The stacking of multiple algorithms has been shown to reduce overfitting and improve the model's ability to generalize to new data.
- *Data Preprocessing:* Techniques like MICE for missing value imputation and StandardScaler for feature scaling contributed to creating a more robust model.
- *Feature Engineering:* The application of binary encoding to two specific columns further enhanced the model's performance.
- *Iterative Approach:* The methodical iterative process allowed for the fine-tuning of hyperparameters and the selection of the most effective data balancing and dimensionality reduction techniques.

These results validate the proposed Ensemble Stacking Classifier model as a powerful tool for liver disease prediction, offering significant improvements over existing methods.

In summary, the findings of this study not only demonstrate the high predictive accuracy of the proposed model but also suggest the substantial potential of ensemble methods in medical diagnosis applications.

Chapter 4
Conclusion and Future Work

4.1 Conclusion

The primary objective of this thesis was to explore and develop an effective and accurate predictive model for liver disease diagnosis, leveraging the ILPD dataset. In a healthcare environment where early detection can dramatically impact outcomes, the utility of machine learning models can be a significant asset. Through a comprehensive investigation of several classifiers and multiple data preprocessing techniques, the thesis aimed to bridge the gap between existing models and a more precise, reliable predictive tool. The crowning achievement of this study is the development of an Ensemble Stacking Classifier, which recorded an impressive accuracy of 91.01% and an AUC-ROC score of 0.9546. This performance surpasses that of existing studies and models on the same dataset. The iterative approach to classifier selection and hyperparameter tuning, coupled with robust data preprocessing, has culminated in a model that not only meets but exceeds the set objectives. This research introduces the Ensemble Stacking Classifier as a potent tool for complex healthcare analytics problems. The rigorous methodology applied in this thesis serves as a framework for future research efforts in this domain. The comparative analysis section could guide researchers by providing insights into how different classifiers perform under various circumstances. The developed model can also serve as an auxiliary tool for medical professionals, aiding in early diagnosis and personalized treatment planning. The high accuracy and reliability of the Ensemble Stacking Classifier makes it a trustworthy partner in clinical decision-making processes. Earlier and more accurate diagnoses enable timely interventions, potentially saving lives and reducing healthcare costs. Patients can also be more assured of the diagnostic process, knowing that it is backed by a robust, data-driven model. With its primary objective met to an admirable degree of accuracy, this study prepares the way for future research and applications in the field of medical diagnostics, which will have a direct and beneficial effect on patient outcomes and healthcare systems.

4.2 Future Work

While the present study has achieved significant advancements in liver disease prediction using the ILPD dataset, several avenues for further research remain open to exploration.

- **Handling Skewed Data**: The issue of data skewness is a critical concern in machine learning models as it can compromise the validity of model assumptions and influence

the interpretation of feature importance. Future work should investigate the utility of logarithmic transformation, specifically using the **log1p** function, to normalize the skewed data distribution, thereby enhancing the model's performance.

- **Exploring Alternative Scaling Techniques**: The current study employed Standard Scaling for feature scaling. However, it is essential to explore other scaling techniques such as Max Scaling, Min-max Scaling, and Robust Scaling to assess their impact on model performance. These alternatives could potentially offer improved normalization of the data, which could lead to more accurate predictions.
- **Feature Importance and Weighting**: One avenue for future research involves diving deeper into the realm of feature selection. By employing methods to discern the importance of each feature, it becomes feasible to apply varying degrees of weight to these features during the prediction phase. Such an approach would inherently optimize the model by focusing on the most significant attributes.
- **Hyper-Parameter Tuning**: Given that our proposed Ensemble Stacking Classifier has already demonstrated superior performance, future research should consider the fine-tuning of hyper-parameters for this model, as well as other classifiers. Advanced techniques like Grid Search or Random Search could be employed for this purpose, aiming for incremental improvements in accuracy and overall model performance.

By addressing these research gaps, the academic community can aspire to develop an even more robust and accurate diagnostic tool for liver diseases, thereby contributing further to healthcare analytics and patient outcomes.

References

[01] T. Kannapiran et al., "Prediction of Liver Disease using Classification Algorithms," in Proceedings of the 2018 4th International Conference on Computing Communication and Automation (ICCCA), December 2018, DOI: 10.1109/CCAA.2018.8777655.

[02] K. Gupta et al., "Liver Disease Prediction using Machine learning Classification Techniques," in Proceedings of the 2022 IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT), April 2022, DOI: 10.1109/CSNT54456.2022.9787574.

[03] R. Amin et al., "Prediction of chronic liver disease patients using integrated projection based statistical feature extraction with machine learning algorithms," Informatics in Medicine Unlocked, vol. 36, p. 101155, 2023. Accepted December 28, 2022.

[04] E. Dritsas and M. Trigka, "Supervised Machine Learning Models for Liver Disease Risk Prediction," Computers, vol. 12, no. 1, January 2023, doi: 10.3390/computers12010019.

[05] J. Singh, S. Bagga, and R. Kaur, "Software-based Prediction of Liver Disease with Feature Selection and Classification Techniques," Procedia Computer Science, vol. 167, pp. 1970-1980, April 2020.

[06] S. Tokala et al., "Liver Disease Prediction and Classification using Machine Learning Techniques," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 14, no. 2, 2023.

[07] A. Q. Md et al., "Enhanced Preprocessing Approach Using Ensemble Machine Learning Algorithms for Detecting Liver Disease," Biomedicines, vol. 11, no. 2, pp. 581, Feb. 2023. [Online]. Available: https://doi.org/10.3390/biomedicines11020581

[08] D. Bhupathi et al., "Liver disease detection using machine learning techniques," in Proceedings of The Computing and Information Technology Research and Education New Zealand (CITRENZ) 2022, Sept. 2022.

[09] F. Mostafa et al., "Statistical Machine Learning Approaches to Liver Disease Prediction," Livers, vol. 1, no. 4, pp. 294-312, Dec. 2021. [Online]. Available: https://doi.org/10.3390/livers1040023

[10] "Indian Liver Patient Records," Kaggle, UCI Machine Learning, [Online]. Available: https://www.kaggle.com/datasets/uciml/indian-liver-patient-records

[11] C. K. Enders, "Applied Missing Data Analysis," Guilford Press, 2010.

[12] S. van Buuren, "Flexible Imputation of Missing Data," CRC Press, 2018.

[13] J. Huang and C. X. Ling, "Using AUC and Accuracy in Evaluating Learning Algorithms," IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 3, pp. 299–310, 2005.

[14] C. Sammut and G. I. Webb, Eds., "Encyclopedia of Machine Learning," Springer, 2011.

[15] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," Informatica, vol. 31, pp. 249–268, 2007.

[16] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.

[17] H. He and Y. Ma, "Imbalanced Learning: Foundations, Algorithms, and Applications," Wiley-IEEE Press, 2013.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[19] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.

[20] H. He, E. A. Garcia, "Learning from Imbalanced Data," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263-1284, 2009.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[22] N. Lunardon, G. Menardi, and N. Torelli, "ROSE: A Package for Binary Imbalanced Learning," R Journal, vol. 6, no. 1, pp. 79-89, 2014.

[23] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: A Hybrid Approach to Alleviating Class Imbalance," IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 40, no. 1, pp. 185-197, 2010.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[25] S. B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," Informatica, vol. 31, pp. 249–268, 2007.

[26] C. Sammut and G. I. Webb, Eds., "Encyclopedia of Machine Learning," Springer, 2011.

[27] I. Jolliffe and J. Cadima, "Principal Component Analysis: A Review and Recent Developments," Philosophical Transactions of the Royal Society A, vol. 374, no. 2065, 2016.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[29] L. J. Bartholomew, M. Knott, and I. Moustaki, "Latent Variable Models and Factor Analysis: A Unified Approach," Wiley, 2011.

[30] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern Classification," Wiley-Interscience, 2000.

[31] S. Theodoridis and K. Koutroumbas, "Pattern Recognition," 4th ed., Academic Press, 2008.

[32] Kohavi, R., "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," In Proceedings of the 14th International Joint Conference on Artificial Intelligence, vol. 2, pp. 1137–1143, 1995.

[33] Chen, T., Guestrin, C., "XGBoost: A Scalable Tree Boosting System," In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, 2016.

[34] Friedman, J. H., "Greedy Function Approximation: A Gradient Boosting Machine," Annals of Statistics, pp. 1189–1232, 2001.

[35] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[36] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely Randomized Trees," Machine Learning, vol. 63, no. 1, pp. 3–42, 2006.

[37] J. R. Quinlan, "Induction of Decision Trees," Machine Learning, vol. 1, no. 1, pp. 81–106, 1986.

[38] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and Regression Trees," Wadsworth International Group, 1984.

[39] D. H. Wolpert, "Stacked Generalization," Neural Networks, vol. 5, pp. 241–259, 1992.

[40] T. G. Dietterich, "Ensemble Learning," in The Handbook of Brain Theory and Neural Networks, M. A. Arbib, Ed., 2002, pp. 405–408.

[41] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, "Applied Logistic Regression," Wiley, 2013.

[42] T. Hastie, R. Tibshirani, and J. Friedman, "The Elements of Statistical Learning," Springer, 2009.

[43] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[44] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," MIT Press, 2016.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.

[46] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21–27, 1967.

[47] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," Machine Learning, vol. 6, no. 1, pp. 37–66, 1991.

[48] P. N. Tan, M. Steinbach, and V. Kumar, "Introduction to Data Mining," Pearson, 2005.

[49] V. Vapnik, "The Nature of Statistical Learning Theory," Springer, 1995.

[50] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in Advances in Kernel Methods: Support Vector Learning, MIT Press, 1999, pp. 185–208.

[51] C. Cortes and V. Vapnik, "Support-Vector Networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.

[52] D. Heckerman, "A Tutorial on Learning With Bayesian Networks," in Learning in Graphical Models, M. I. Jordan, Ed. MIT Press, 1999, pp. 301–354.

[53] K. P. Murphy, "Machine Learning: A Probabilistic Perspective," MIT Press, 2012.

[54] S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," Prentice Hall, 3rd ed., 2009.

[55] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[56] J. A. Motta et al., "Evaluation of Efficiency of Linear Techniques to Optimize Attribute Space in Machine Learning: Relevant Results for Extractive Methods of Summarizing," Computer and Information Science, vol. 5, no. 6, 2012. [Online]. Available: https://doi.org/10.5539/cis.v5n6p58.

[57] "Base rate - Wikipedia," Wikipedia, https://en.wikipedia.org/wiki/Base_rate, Accessed: 20th September, 2023.

[58] F. Bellocchio, P. Carioni, C. Lonati, M. Garbelli, F. Martínez-Martínez, S. Stuard, and L. Neri, "Enhanced Sentinel Surveillance System for COVID-19 Outbreak Prediction in a Large European Dialysis Clinics Network," International Journal of Environmental Research and Public Health, vol. 18, no. 18, p. 9739, 2021.

[59] K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan, "Data Mining: A Knowledge Discovery Approach," Springer, 2007.

[60] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," in Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995, pp. 1137–1143.

[61] T. Fawcett, "An Introduction to ROC Analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006.

[62] T. Fawcett, "An introduction to ROC analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006.

[63] C. D. Manning, P. Raghavan, and H. Schütze, "Introduction to Information Retrieval," Cambridge University Press, 2008.

[64] J. A. Swets, "Measuring the accuracy of diagnostic systems," Science, vol. 240, no. 4857, pp. 1285–1293, 1988.