UHASSELT
KNOWLEDGE IN ACTION

Maastricht University

Faculty of Sciences

School for Information Technology

Master of Statistics and Data Science

Master's thesis

Predicting Cirrhosis Patient Survival Using Machine Learning: A Data-Driven Approach

Meseret Assefa Kerga

Thesis presented in fulfillment of the requirements for the degree of Master of Statistics and Data Science, specialization Data Science

SUPERVISOR :
dr. ir. Axel FAES

2024
2025

Faculty of Sciences

School for Information Technology

Master of Statistics and Data Science

Master's thesis

Predicting Cirrhosis Patient Survival Using Machine Learning: A Data-Driven Approach

Meseret Assefa Kerga

Thesis presented in fulfillment of the requirements for the degree of Master of Statistics and Data Science, specialization Data Science

SUPERVISOR :

dr. ir. Axel FAES

Acknowledgements

First of all, I thank God and Saint Mary for being with me through everything.

My deepest thanks go to my supervisor, Dr. ir. Axel FAES, for all the helpful advice, constant support, and encouragement that were so important to the successful finishing of this thesis.

I would like to extend my gratitude to all of the lecturers in this master's program. Your passionate teaching truly left an impression and kept me encouraged. To my fellow students and project teammates, your collaboration, support, and friendship made this journey much richer.

I am sincerely grateful to my employer, ReelMetrics, for their continued support and flexibility throughout my academic journey.

I lost childhood friends and a family member while studying. This piece honors my loved ones; your memory lives on in me.

To my dear husband and sweet son, Haniel: thank you for your unending encouragement, patience, and faith in me. Your understanding of the sacrifices I made balancing studies and work truly means the world. And to my gorgeous baby, Johan: when I got pregnant, I worried I wouldn't finish my studies, but you brought me joy, energy, and countless blessings. I can't wait to spend more time with you.

The love, prayers, and support of my family have made me stronger. A special thanks to my mother for regularly checking in and encouraging me to stay on track with my studies and thesis writing.

Meseret Assefa Kerga
Leiden, Netherlands

Abstract

Background: Liver cirrhosis is a deadly, chronic illness with rising global rates. Patient outcomes must be accurately predicted for timely intervention, treatment planning, and care prioritizing.

Objective: This study will assess machine learning models' ability to predict cirrhosis patients' survival outcomes using clinical and demographic data from diagnosis or follow-up.

Methodology: The Cirrhosis Patient Survival Prediction dataset, sourced from the UCI Machine Learning Repository and collected at the Mayo Clinic, contains 418 patient records and numerical and categorical features. The classification target comprises three mutually exclusive survival outcomes: alive, liver transplant, and death. Four supervised machine learning algorithms Logistic Regression, K-Nearest Neighbors, Random Forest, and Extreme Gradient Boosting were trained and evaluated under four different class imbalance handling techniques: SMOTE, random undersampling, SMOTE combined with undersampling, and SMOTEENN. Model performance was assessed using macro-averaged F1-score as primary metrics.

Results: The XGBoost classifier with SMOTE had the best performance, with a macro F1-score of 60.2%. In comparison, KNN with undersampling performed poorly, achieving a macro F1-score of 44.6% , a 35% improvement from the lowest to the highest-performing arrangement. Bilirubin, Ascites, Prothrombin time, Age, Sex, and Hepatomegaly were the most important survival predictors, according to XGBoost feature importance analysis.

Conclusion: This study demonstrates that machine learning, particularly ensemble-based methods such as XGBoost, can effectively model complex, imbalanced multiclass survival data in a clinical context. The integration of SMOTE improves generalizability across underrepresented classes. Furthermore, the interpretability of key clinical features enhances trust in the model's predictions and strengthens its potential as a decision-support tool for managing cirrhosis patients.

Keywords: *Liver Cirrhosis, Machine Learning, XGBoost, Survival Prediction, SMOTE, F1-Score, ROC-AUC, Feature Importance.*

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
AST	Aspartate Aminotransferase
AUC	Area Under the ROC Curve
CV	Cross-Validation
DL	Deep Learning
EDA	Exploratory Data Analysis
ENN	Edited Nearest Neighbors
KNN	K-Nearest Neighbors
LR	Logistic Regression
MELD	Model for End-Stage Liver Disease
ML	Machine Learning
MLP	Multilayer Perceptron
Na	Sodium
RF	Random Forest
ROC	Receiver Operating Characteristic
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
XGBoost	Extreme Gradient Boosting

1 Introduction

1.1 Background

Liver cirrhosis is a long-term, irreversible disease in which good liver tissue is slowly replaced by fibrotic scar tissue, which eventually makes the liver less able to do its job. It is one of the main causes of illness and death throughout the world, killing over a million people every year and putting a lot of stress on healthcare systems [1]. People with cirrhosis often have major problems include portal hypertension, ascites, hepatic encephalopathy, and liver failure. These problems can lead to repeated hospital stays and a lower quality of life.

Even while treatments and tests have gotten better, cirrhosis is still a big problem for doctors, especially in places with few resources. So, it's very important to be able to forecast patient outcomes early and accurately in order to improve survival, prioritize candidates for liver transplantation, and organize timely therapies. Making strong tools to forecast how long someone with cirrhosis will live not only helps with individualized patient care, but it also helps health policy makers make better decisions and make the most use of limited medical resources.

Traditionally, models such as the Model for End-Stage Liver Disease (MELD) have been employed to predict short-term survival in patients with cirrhosis. These scoring systems rely primarily on a limited set of static laboratory and clinical measurements, which may not adequately reflect the dynamic nature of the disease [2]. As a result, they are often insufficient for long-term survival prediction or for tailoring personalized treatment plans.

To address these limitations, recent research has introduced time-varying approaches that leverage longitudinal patient data. For example, Goldberg and Zarnegarnia [2] demonstrated that extended Cox models utilizing dynamic laboratory data yield improved survival predictions compared to traditional methods. Additionally, artificial intelligence, particularly machine learning, has emerged as a valuable tool for modeling complex, nonlinear relationships within healthcare data. A study by [3] utilized artificial neural networks to predict in-hospital mor-

tality among cirrhotic patients with hyponatremia, surpassing the performance of classical scoring models such as MELD and MELD-Na.
AI is changing the way medical professionals work in significant ways by making diagnoses more accurate and cutting down on the amount of work they have to do. For instance, AI-based systems are now being used to help diagnose coeliac disease, which speeds up the process and helps doctors make decisions in the clinic [4]. According to [5], AI can now diagnose coeliac disease with pathologist-level accuracy.
AI and ML have a lot of potential, but there are still a lot of practical and ethical problems that need to be solved. These include the need for clinical datasets that are high-quality and well-annotated, models that are easy to understand, and models that operate well with existing clinical workflows. Data scientists and healthcare practitioners need to work closely together to build models that are not just correct but also clear and useful.
In conclusion, there is a lot of potential for enhancing the prediction of cirrhosis survival outcomes by the incorporation of machine learning approaches with dynamic patient data. Reliable survival models can improve resource allocation in liver care, guide clinical decision-making, and assist in the early identification of high-risk patients.
The remainder of this thesis is organized as follows:

- *Chapter 2: Dataset Description* - presents an overview of the dataset used in this study, including its structure, features, and challenges such as missingness and class imbalance.
- *Chapter 3: Methodology* - details the machine learning algorithm employed in this study, addresses challenges such as data missingness and class imbalance, outlines resampling techniques, preprocessing steps, feature engineering, model training pipeline, and evaluation metrics.
- *Chapter 4: Results and Discussion* - presents the outcomes of exploratory data analysis and machine learning experiments, evaluates model performance, examines feature importance, and analyzes the effects of imbalance

management.

- *Chapter 5: Conclusion and Future Work* - summarizes the key findings, outlines limitations, and suggests directions for future research.
- *Chapter 6: Ethics, Societal Relevance, and Stakeholder Awareness* - reflects on ethical considerations such as data privacy, fairness, and transparency; discusses the broader societal impact of predictive modeling in healthcare; and identifies key stakeholders affected by this research.

1.2 Objective of the study

This work aims to create and assess machine learning models forecasting cirrhosis patient survival. Specifically, this study aims to determine the most efficient ML model for survival prediction, evaluate the clinical characteristics that notably affect patient outcomes, and investigate how different class imbalance management strategies affect model performance. The study's main objectives are: (1) Which ML model best predicts cirrhosis survival? (2) Which clinical features are most important for survival outcomes? (3) How do different class imbalance techniques, such as Synthetic Minority Over-sampling Technique (SMOTE), Random Undersampling, their combination, and SMOTE with the Edited Nearest Neighbors (ENN) affect model performance? By addressing these questions, this work intends to increase predictive accuracy, improve model interpretability, and support better clinical decision-making for cirrhosis care utilizing answers to these queries.

2 Dataset Description

This study utilizes the Cirrhosis Patient Survival Prediction Dataset, sourced from the UCI Machine Learning Repository [6,7]. The dataset comprises clinical records of 418 patients diagnosed with liver cirrhosis, originally collected from the Mayo Clinic. The dataset is designed to support or enable building machine learning models that can predict patient survival outcomes using both demographic and medical characteristics.

The dataset contains 20 variables, encompassing both numerical and categorical features. Key attributes include:

- *Age*: Demographic.
- *Sex*: Demographic.
- *Bilirubin, Albumin, Alkaline Phosphatase, AST, Prothrombin Time (Protime)*: Laboratory markers indicative of liver function and disease progression.
- *Ascites*: Fluid accumulation in the abdomen, often a sign of liver disease severity.
- *Hepatomegaly*: Liver enlargement, which can be indicative of liver dysfunction or disease.
- *Edema*: Swelling caused by fluid retention in tissues, which can occur due to liver cirrhosis.
- *Stage*: Histological stage of liver fibrosis (ordinal feature).
- *Drug*: Whether the patient received D-penicillamine treatment (binary feature).
- *Status (Target Variable)*: The final survival status of the patient - "C" (censored: alive), "CL" (censored: liver transplant), and "D" (death).

The target variable in this study is "Status," which contains three classes of categories. This makes the task a multiclass classification problem. The dataset stands out because it comprises clinical information, has various feature types (categorical and continuous variables), and has some missing values. These traits provide a realistic and challenging setting for using and testing machine learning models in healthcare. A comprehensive overview of each variable, including its type, clinical significance, units, and instances of missing data, is provided in Table 1.

Table 1: Description of Dataset Variables (adapted from the original UCI source)

Variable Name	**Role**	**Type**	**Demographic**	**Description**	**Units**	**Missing Values**
ID	ID	Integer	-	Unique identifier	-	No
N_Days	Other	Integer	-	Days from registration to outcome (death, transplant, or study end)	-	No
Drug	Feature	Categorical	-	Type of drug (D-penicillamine or placebo)	-	Yes
Age	Feature	Integer	Age	Patient's age in days	Days	No
Sex	Feature	Categorical	Sex	Gender of the patient (M or F)	-	No
Ascites	Feature	Categorical	-	Presence of ascites (Y/N)	-	Yes
Hepatomegaly	Feature	Categorical	-	Liver enlargement (Y/N)	-	Yes
Spiders	Feature	Categorical	-	Presence of spider angiomas (Y/N)	-	Yes
Edema	Feature	Categorical	-	Edema status (N, S, Y)	-	No
Bilirubin	Feature	Continuous	-	Serum bilirubin level	mg/dL	No
Cholesterol	Feature	Integer	-	Serum cholesterol level	mg/dL	Yes
Albumin	Feature	Continuous	-	Serum albumin level	g/dL	No
Copper	Feature	Integer	-	Urinary copper level	µg/day	Yes
Alk_Phos	Feature	Continuous	-	Alkaline phosphatase enzyme level	U/Liter	Yes
SGOT	Feature	Continuous	-	Serum glutamic-oxaloacetic transaminase	U/mL	Yes
Triglycerides	Feature	Integer	-	Serum triglycerides	-	Yes
Platelets	Feature	Integer	-	Platelet count	mL/1000	Yes
Prothrombin	Feature	Continuous	-	Blood clotting time	Seconds	Yes
Stage	Feature	Categorical	-	Histological stage of liver disease (1–4)	-	Yes
Status	Target	Categorical	-	Patient outcome: C, CL, or D	-	No

3 Methodology

In this study, four machine learning algorithms were selected to model a multiclass classification problem, where the target variable consists of three mutually exclusive categories. The chosen models, Logistic Regression (LR), K-Nearest Neighbors (KNN), Random Forest (RF), and Extreme Gradient Boosting (XGBoost) are widely recognized in the statistical learning literature for their effectiveness in handling both categorical and numerical data [8]. Their selection is motivated by their distinct strengths: LR for modeling linear relationships and interpretability; KNN for capturing local, nonlinear patterns; RF for robustness and handling feature interactions; and XGBoost for its scalability and high predictive performance in multiclass tasks.

To supplement the textual descriptions, visual illustrations for each algorithm are included. These figures, adapted from reputable educational sources, offer an intuitive understanding of how each model makes decisions and what types of decision boundaries they form in a classification task.

Logistic Regression (LR): Logistic Regression is a fundamental classification technique that models the log-odds of the outcomes as a linear function of the predictors. While it is most commonly used for binary classification, it can be naturally extended to handle multiclass classification through the multinomial logistic regression approach.

$$\log\left(\frac{P(Y=k)}{P(Y=K)}\right) = \beta_{0k} + \beta_{1k}X_1 + \beta_{2k}X_2 + \cdots + \beta_{pk}X_p, \quad \text{for } k = 1, 2, 3 \tag{1}$$

This model is appropriate when the classes are mutually exclusive, and it allows for probabilistic interpretation of predictions. Logistic Regression is favored for its simplicity and interpretability, especially when Linear relationships between predictors and outcomes are assumed [8, 9].

Figure 1 illustrates how logistic regression creates decision boundaries in a multiclass classification problem. The softmax function takes the model's raw output and turns it into probabilities, making sure that the total of all the probabilities for all the classes is 1. This lets the model figure out how likely each class is and put each input into the class that is most likely to be appropriate.

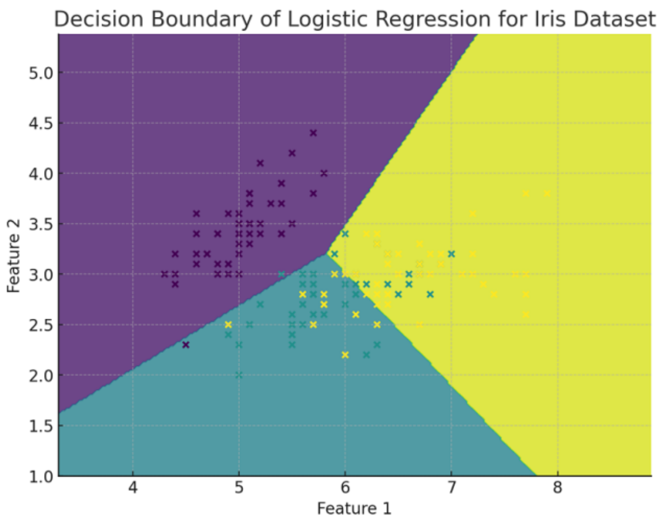

Figure 1: Decision boundaries formed by logistic regression. Adapted from VitalFlux website

K-Nearest Neighbors (KNN): is a non-parametric algorithm that classifies a new observation based on the majority class among its k nearest neighbors in the training data. It does not assume a specific functional form for the relationship between the predictors and the response variable, which makes it suitable for capturing complex nonlinear patterns. Two key factors that have a significant impact on how well the KNN model performs are the the parameter k and the distance metric utilized for determining how close two data points are to each other. Even though KNN is straightforward to understand, it often does well in multiclass classification tasks, especially when the feature space is well-structured and appropriate for distance-based comparisons [8].
Figure 2 shows that the value of k has a direct effect on the model's decision bounds. A smaller value of k makes the model very sensitive to patterns in local data, which makes the decision surfaces more irregular and perhaps overfitting. On the other hand, a bigger k value includes a greater neighborhood, which makes the decision boundaries smoother, less sensitive to noise, and maybe better at

generalizing to new data.

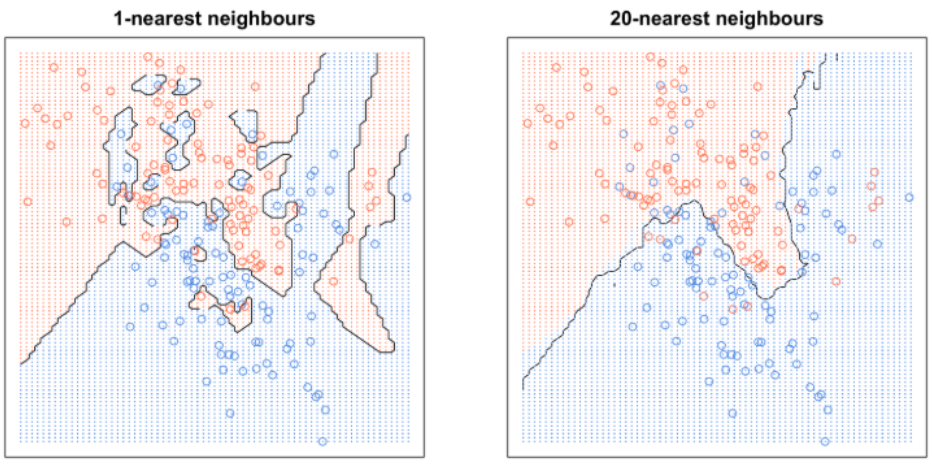

Figure 2: Decision boundaries using KNN with different values of k. Adapted from Github

Random Forest (RF): is a robust ensemble learning method for classification and regression tasks. At its foundation, Random Forest makes a lot of separate decision trees. This group was made utilizing two important ways to randomize: This ensemble is made up of bootstrapped samples of the training data (which means that some of the data is used again) and random selections of features. This two-layer randomness makes sure that every tree in the forest is different, which adds to the diversity of the ensemble. For classification problems, the final prediction is the one that gets the most votes from all the trees. For regression, it's the average of all the predictions. There are numerous good things about this method. Because different trees smooth out the unique patterns learnt by any one tree, Random Forest is less likely to overfit, which is a common problem with single decision trees. It depicts how complicated predictor variables interact without scaling features or dealing with outliers. Random Forest also does a good job at classifying several classes and working with datasets that have missing values and a mix of variable types, such numerical and categorical characteristics. [8,10]. Figure 3 illustrates the ensemble structure of a random forest, emphasizing the diversity among individual trees and how they collectively reduce variance and improve generalization.

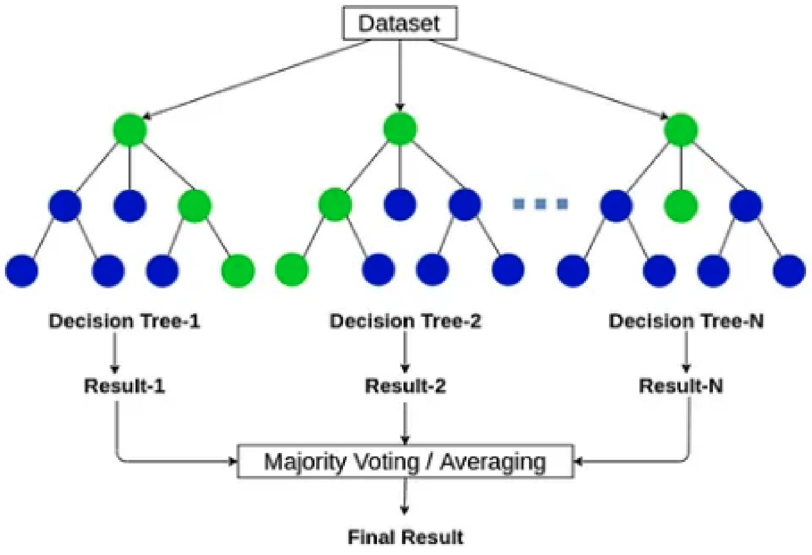

Figure 3: Illustration of a Random Forest. Adapted from Medium.

Extreme Gradient Boosting (XGBoost): is a gradient boosting algorithm that works fast and may be used on an extensive scale. XGBoost generates decision trees one after the other, while ensemble methods like Random Forest build them all at once. The ensemble trains each new tree to make predictions and fix errors (the discrepancies between predicted and actual values) that were made by all of the prior trees. This method of correcting errors over and over again makes the model more accurate. Softmax objective functions let XGBoost classify things into more than one class. Adding regularization methods like L1 and L2 penalties straight to the objective function makes the model more robust by reducing overfitting and improving generalization.
XGBoost's merits are its high accuracy in predicting outcomes across a wide range of tasks, its ability to manage missing data without having to explicitly fill it in, and its fast computation speed thanks to streamlined algorithms and parallel processing. XGBoost does better than other machine learning algorithms in the actual world, especially when it comes to categorization benchmarks and real-

life situations [11]. Figure 4 shows how a gradient boosting decision tree model is built. The visualization demonstrates how each successive tree is specifically constructed to model and reduce the residual errors from the predictions of the preceding ensemble, thereby iteratively improving the overall model's fit to the data.

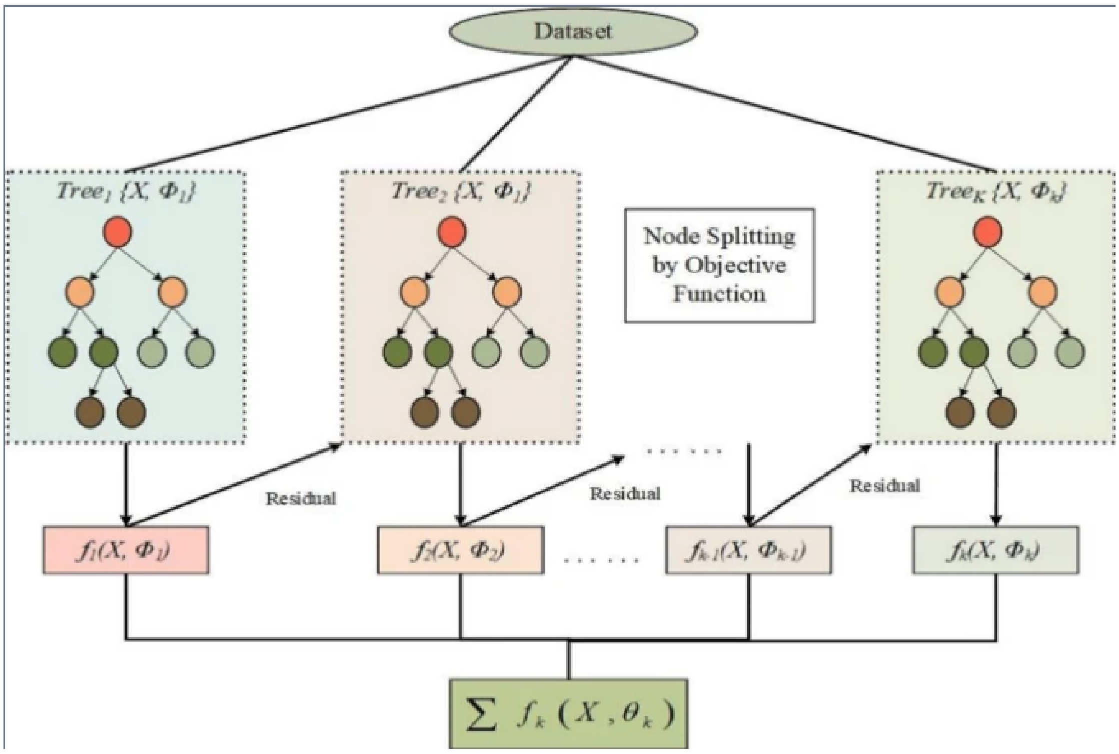

Figure 4: Illustration of Boosting architecture in XGBoost. Adapted from Bio-DataMining.

3.1 Exploratory Data Analysis

Before utilizing machine learning models, it is necessary to do exploratory data analysis (EDA), which is an essential phase in this investigation since it offers a more in-depth comprehension of the dataset. EDA is a useful tool for analyzing the distribution of the target variable in order to identify any class imbalances. This is particularly important when considering the multiclass character of the survival prediction objective. In addition to this, it makes it possible to identify connections between clinical characteristics and the results of patient survival, which provides insights into potential predictors. The exploration of feature relationships is accomplished through the utilization of correlation analysis and visualization techniques such as box plots and histograms. On the other hand, missing value analysis is carried out in order to ascertain the level of missingness and the potential impact it has on the dataset. By performing EDA, we are able to make educated judgments regarding the selection of features, the preprocessing of data, and the appropriate treatment of missing data. This helps to ensure that the predictive models are constructed on a dataset that is both well understood and structured [8, 12, 13].

3.2 Class Imbalance Handling Techniques

Class imbalance is a common challenge in real-world classification tasks, particularly in the medical and healthcare domains, where positive instances (e.g., disease cases) are significantly outnumbered by negative ones. Training models on such skewed datasets often results in biased classifiers that favor the majority class, leading to poor generalization, especially for the minority class. To mitigate this issue, various data-level resampling techniques were employed in this study, including SMOTE, Random Undersampling, a hybrid of both, and SMOTE + Edited Nearest Neighbors (SMOTEENN).

Synthetic Minority Oversampling Technique: A popular oversampling method called SMOTE [14] creates synthetic instances for the minority class in order to rectify class imbalance. SMOTE interpolates between existing minority class examples, as opposed to naive oversampling, which duplicates existing samples and runs the risk of overfitting. SMOTE chooses one or more of its closest neighbors for a particular minority instance and generates new synthetic samples along the line segments that connect them. This leads to a more general decision region for the minority class and improves classifier sensitivity. Mathematically, a synthetic instance x_{new} is generated as:

$$x_{\text{new}} = x_i + \delta \cdot (x_{nn} - x_i)$$

where x_i is a minority sample, x_{nn} is one of its k-nearest neighbors, and $\delta \in [0, 1]$ is a random number.

Random Undersampling: This technique involves reducing the size of the majority class by randomly discarding instances until the class distribution becomes more balanced [15]. While this technique is computationally efficient and straightforward, it risks removing potentially informative samples, which may degrade model performance if not applied judiciously. Nonetheless, when combined with oversampling, it can lead to improved results by reducing training time and class dominance.

SMOTE + Random Undersampling: To leverage the benefits of both techniques, a hybrid approach combining SMOTE and Random Undersampling was also explored [16]. In this strategy, SMOTE is first applied to synthetically boost the minority class, followed by random undersampling of the majority class to achieve a more balanced and compact training set. This two-step process helps to alleviate both overfitting (by synthetic diversification) and majority-class bias (by sample reduction), resulting in better-balanced decision boundaries.

SMOTE + Edited Nearest Neighbors(SMOTEENN): is an advanced hybrid resampling method that combines SMOTE with the Edited Nearest Neighbors (ENN) rule. After applying SMOTE, the ENN algorithm is used to clean the dataset by removing ambiguous or misclassified instances. Specifically, Edited Nearest Neighbors(ENN) eliminates samples (from either class) that differ from the majority class among their k-nearest neighbors [17]. This not only addresses class imbalance but also reduces overlapping between classes and eliminates noise from the dataset.

SMOTEENN has been shown to produce more robust and generalizable classifiers, particularly in noisy or complex datasets. Its denoising capability makes it preferable for sensitive applications such as healthcare, where data quality significantly influences model outcomes.

Figure 5 compares the several resampling methods used in this study. The image below shows how several resampling strategies, like oversampling, undersampling, and hybrid approaches, modify the class distribution of the dataset used in this thesis. The original dataset had an uneven class distribution. These solutions strive to make the representation more even, which could help machine learning models operate better in the future. Figure 5 offers a picture that lets us see how effectively these strategies operate. Section 4.2.3 will talk about how solutions for dealing with class imbalance affect the performance of models.

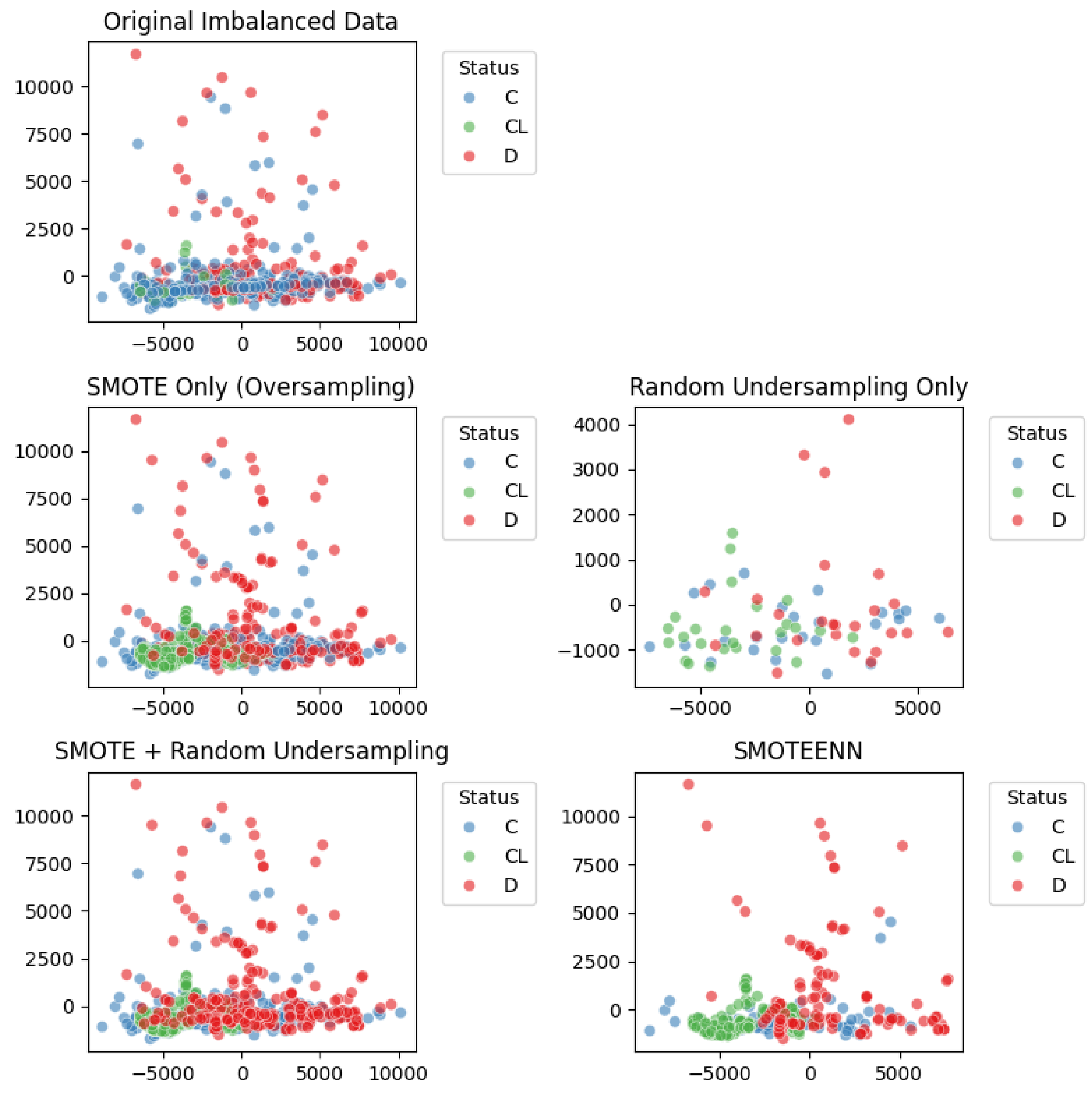

Figure 5: Visual comparison of resampling techniques on class distribution.

3.3 Data Preprocessing

To prepare the dataset for multiclass classification, a series of preprocessing steps were applied to ensure data quality and model readiness. The initial step involved addressing missing values. Rows or columns with excessive missing data were either removed or imputed using appropriate strategies, such as mode imputation for categorical variables and mean or median imputation for numerical variables [18]. Categorical features were transformed into a numerical format using one-hot encoding, which is suitable for most machine learning algorithms. For ordinal variables, ordinal encoding was used to preserve the inherent order. Numerical features were standardized using z-score normalization, which is particularly useful for distance-based models like K-Nearest Neighbors (KNN) and also ensures stable convergence in algorithms such as logistic regression and neural networks [19].

To maintain the original distribution of all three target classes during training and evaluation, a stratified train-test split was employed. Additionally, k-fold cross-validation (with stratification) was used for model validation to ensure consistent performance across different subsets of the data [8].

To further optimize model performance, a systematic hyperparameter tuning approach was applied using Grid search. Grid search exhaustively explores a predefined set of hyperparameter combinations for each model and selects the configuration that yields the best cross-validated performance. This process was implemented using GridSearchCV with stratified 5-fold cross-validation to ensure robust and fair evaluation across all classes. Hyperparameters such as tree depth, learning rate, and the number of estimators (for ensemble models) or regularization terms (for logistic regression) were tuned. By incorporating grid search into the training pipeline, the study ensures that each model operates under optimal settings, reducing the risk of underfitting or overfitting while improving generalization to unseen data [20].

Feature selection was performed using a combination of correlation analysis and model-based importance scores (e.g., from tree-based classifiers). This step helped reduce dimensionality and eliminate redundant or irrelevant features, thereby improving model performance and interpretability [21].

3.4 Missing Value Imputation

Building machine learning models requires first handling missing values, a vital preprocessing stage. Missing data was addressed in this work using straightforward imputation methods, including mean or median. Median imputation was better than mean imputation for features that were skewed or likely to be outliers because the target variable was uneven, and some feature outlier values were present. Median imputation gives a better estimate for most of the data since it is less affected by extreme values, making it more resistant to outliers. On the other hand, outliers can greatly affect mean imputation, which might affect the model. Mean imputation can be utilized for features with a distribution that was about symmetric and had no extreme values. This is because it keeps the original average of the data. The imputation technique we used in this study, median imputation, reduces the distortion caused by missing data while keeping the dataset's original structure. The study backs up these kinds of practices. Basic imputation methods are suggested as quick and effective ways to fix the problem for datasets with moderate amounts of missing data and imbalanced target distributions [22].

3.5 Evaluation Metrics

Given that the target variable consists of three distinct classes, evaluation was carried out using metrics well-suited to multiclass classification, particularly in the context of imbalanced class distributions. While overall accuracy offers a high-level overview of performance, it may be misleading in the presence of class imbalance, as it is biased toward the majority class [23]. Therefore, more nuanced and robust metrics were adopted for model evaluation:

- *Accuracy:* Measures the overall proportion of correctly classified samples. Though intuitive, it does not distinguish between types of errors and is inadequate when class distributions are skewed.
- *Precision, Recall, and F1-Score (Macro, Micro, and Weighted Averages):*

  - Macro-averaged: Computes metrics independently for each class and averages them, treating all classes equally, regardless of class frequency.
  - Micro-averaged: Aggregates all true positives, false negatives, and false positives to compute metrics globally, which gives more weight to the majority classes.
  - Weighted-averaged: Averages metrics for each class while taking into account class support (i.e., the number of true instances per class).

- *Multiclass ROC-AUC:* Although traditionally designed for binary classification, ROC-AUC can be extended to multiclass settings using One-vs-Rest (OvR) or One-vs-One (OvO) strategies. It assesses the model's ability to rank predictions correctly and distinguish between multiple classes [24].

In line with the goal of identifying the most robust and well-balanced model for survival prediction, macro-averaged F1-score was selected as the primary evaluation metric. This metric captures the harmonic mean of precision and recall across all classes, treating each class equally and balancing both false positives and false negatives. It is especially appropriate for multiclass problems with imbalanced distributions, offering a fair comparison of model performance across minority and majority classes [25].

To provide additional insight, macro-averaged recall was included as a secondary evaluation metric. The F1-score assesses the equilibrium between sensitivity and specificity, whereas macro recall alone measures the model's capacity to identify all true cases, irrespective of class size, which is particularly important in clinical contexts where the oversight of high-risk patients can lead to severe consequences. The multiclass ROC-AUC was presented to assess the model's discriminative capability from a probabilistic viewpoint. Collectively, these metrics provide a thorough and equitable assessment methodology for determining the most appropriate machine learning model for predicting cirrhosis survival.

4 Results and Discussion

4.1 Explanatory Data Analysis

To better understand the dataset's characteristics and identify potential patterns, we did an Exploratory Data Analysis (EDA) to better understand the dataset's properties and look for possible patterns. We looked at the distribution of features, the target variable (Status), and any missing values in the dataset.

The distribution of continuous features (Figure 12) reveals features(e.g., N. Days, Age, Bilirubin, Cholesterol) significant variation across different Status categories. In general, the box plots show the median (central tendency), the interquartile range (spread), and any possible outliers for each variable in each status group. This helps you examine how these medical indicators change depending on the patient's condition. For example, depending on the patient's status category, one can see variances in the median levels, the range of values, and the presence of extreme values for each measured feature. Similarly, the histogram for categorical features (Figure 11) highlights that some features, such as Ascites, Edema, and Spiders etc, may be strong indicators of survival status. These visualizations help in understanding how different clinical attributes relate to patient survival outcomes.

The missingness heatmap (Figure 6) illustrates the presence of missing data across multiple features, with notable gaps in Drug, Ascites, Hepatomegaly, Spiders, Copper, SGOT, etc. Identifying the data issues in such a visualization suggests that careful handling of missing data, such as imputation or exclusion, is necessary before modeling.

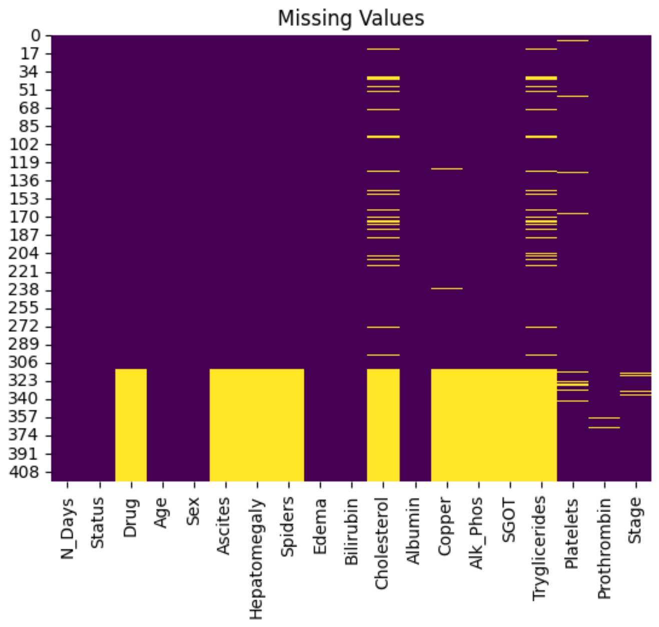

Figure 6: Missing in the dataset

Lastly, the target variable distribution (Figure 7) shows an imbalance, where the percentage distribution for each class (C, D, CL) is 55.5%, 38.5% and 6% , respectively suggesting that when building predictive models, techniques such as SMOTE, Undersampling, etc need to be used to address class imbalance.

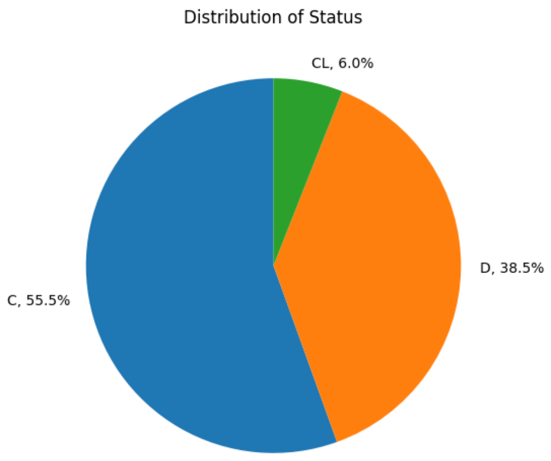

Figure 7: Distribution of the Target Variable (Status)

The distribution graph (Figure 12)provides an overview of how several categorical variables such as Ascites, Drug, Edema, Hepatomegaly, Sex, and Spiders are distributed across the three patient outcome groups: D (Deceased), C (Censored), and CL (Censored due to liver transplant). Notable differences emerge in the distributions of variables like Ascites, Sex, and Spiders. For instance, the majority of patients in the 'D' group report 'No' for Ascites, while the 'C' and 'CL' groups show more variability. Additionally, there is a higher proportion of females in the 'D' group compared to the others. These observed differences may suggest potential associations between clinical attributes and patient outcomes, warranting further investigation in the modeling phase.

4.2 ML Algorithm Findings

4.2.1 Model Comparison and Selection

Four classifiers, Logistic Regression, K-Nearest Neighbors (KNN), Random Forest, and XGBoost, were evaluated to meet the first research goal to find the most suitable machine learning model for predicting cirrhosis patient survival. Every model was evaluated together with four distinct class imbalance handling techniques: SMOTE, Random Undersampling, SMOTE combined with Undersampling, and SMOTEENN. While macro-averaged recall and ROC-AUC were viewed as additional measures of model robustness and class separability, macro-averaged F1-score served as the primary metric for evaluating model performance.
A comprehensive comparison of all models and configurations is presented in Table 2. The results indicate that the XGBoost classifier, in combination with the SMOTE class imbalance handling technique, obtained the highest macro F1-score (0.602), as well as the highest ROC-AUC (0.85) and competitive macro recall (0.598) among all combinations. This suggests that XGBoost is particularly effective at capturing non-linear relationships in imbalanced multiclass data and that synthetic oversampling via SMOTE substantially improves its ability to generalize across all classes.
The best-performing setup is summarized in Table 3. Although Random Forest with SMOTEENN and Logistic Regression with SMOTE + Undersampling also performed well, the XGBoost + SMOTE combo provided the most balanced and consistent performance across all three criteria. These results imply that synthetic oversampling methods help ensemble tree-based models better, probably because of their ability to exploit intricate feature interactions generated by resampling. Figure 8 visually compares macro F1-scores across all model and resampling configurations, showing the advantage of XGBoost + SMOTE over other combinations.
In addition to tabular metrics, Figure 13 provides a visual comparison of the macro-average ROC-AUC curves for all models across different imbalance handling strategies. The XGBoost model combined with SMOTE achieved the most favorable ROC curve, consistently staying above the diagonal baseline and yielding

the highest AUC. These findings further confirm the model's strong discriminative ability. Other models, such as Random Forest with SMOTEENN and Logistic Regression with undersampling, also showed reasonably good curves but did not outperform XGBoost in terms of both area under the curve and balance across classes. The ROC curves reinforce the numerical findings reported in Tables 2 and 3.

Table 2: Performance Comparison of ML Models with Different Resampling Techniques

Model	**Imbalance Handling Method**	**F1 Macro**	**Macro Recall**	**ROC-AUC**
Logistic Regression	SMOTE	0.560	0.636	0.817
Logistic Regression	UnderSampling	0.576	0.678	0.811
Logistic Regression	SMOTE + UnderSampling	0.581	0.657	0.821
Logistic Regression	SMOTEENN	0.543	0.630	0.802
KNN	SMOTE	0.524	0.572	0.741
KNN	UnderSampling	0.446	0.542	0.681
KNN	SMOTE + UnderSampling	0.524	0.572	0.741
KNN	SMOTEENN	0.477	0.582	0.722
Random Forest	SMOTE	0.570	0.570	0.845
Random Forest	UnderSampling	0.561	0.663	0.810
Random Forest	SMOTE + UnderSampling	0.565	0.558	0.854
Random Forest	SMOTEENN	0.588	0.633	0.846
XGBoost	SMOTE	0.602	0.598	0.853
XGBoost	UnderSampling	0.489	0.561	0.782
XGBoost	SMOTE + UnderSampling	0.593	0.593	0.850
XGBoost	SMOTEENN	0.583	0.650	0.843

Table 3: Best Model Based on Macro F1-score

Model	**Resampling**	**F1 Macro**	**Macro Recall**	**ROC-AUC**
XGBoost	SMOTE	0.602	0.598	0.853

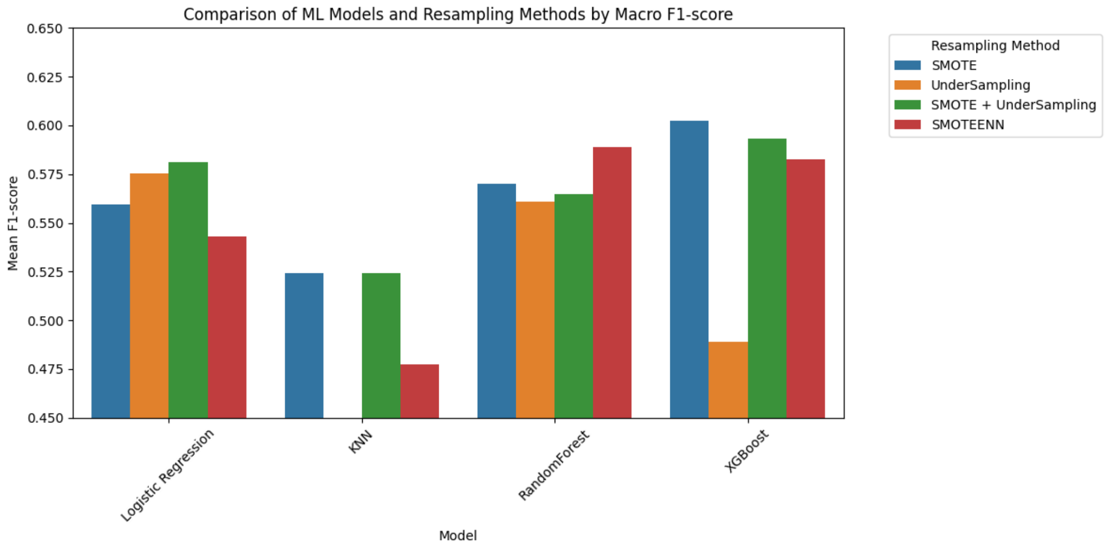

Figure 8: Comparison of ML Models and Resampling Methods by Macro F1-score

The confusion matrix presented in Figure 9 further supports the performance evaluation by illustrating how well the final model distinguishes between the three patient status classes. The model demonstrated strong predictive ability for the majority class C, correctly classifying 35 out of 47 instances, and a reasonably good performance for class D, with 21 out of 32 samples accurately predicted. Notably, the model also detected 2 out of 5 instances of the minority class CL, which is a critical advancement given the class imbalance present in the dataset. However, some misclassifications occurred, particularly with CL samples being confused with class C and D. However, the overall distribution of predictions shows that the model can generalize across all classes, and it is far more effective at finding uncommon situations. These results show that the final model has acquired proper decision bounds, even for the less common CL category.

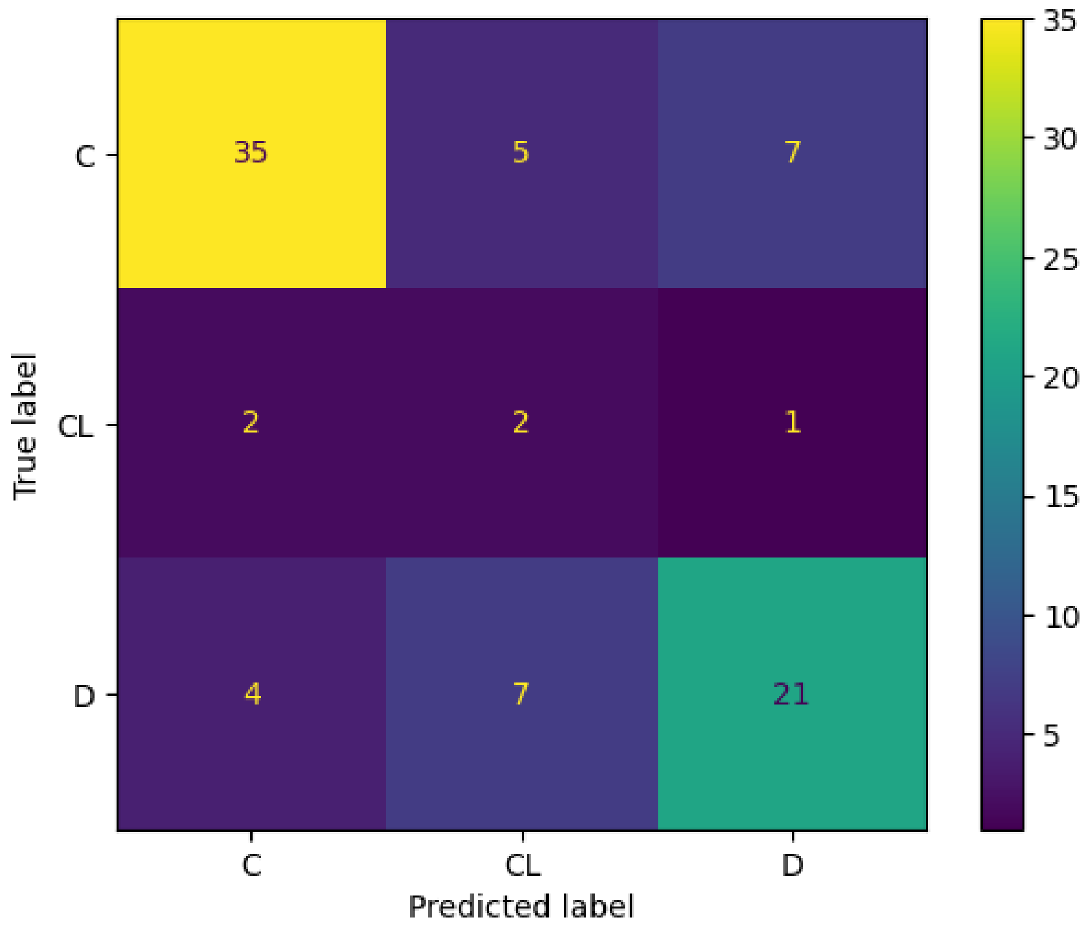

Figure 9: Confusion matrix of the final XGBoost model trained with SMOTE.

4.2.2 Feature Importance Analysis

Following monitoring how well the classification performed, it is important to know which features helped the model make the best predictions. Using the final XG-Boost model trained on SMOTE-balanced data, which showed better recall for the minority class CL and strong overall performance, we did a feature importance analysis. Feature importance scores were derived from the final XGBoost model trained with SMOTE resampling to answer the second research objective, which was to determine which clinical features have the most significant influence on the outcomes of cirrhosis survival. We took advantage of the built-in

gain-based significance metric that is included in XGBoost. This metric indicates each feature's average contribution to the model's prediction performance overall individual decision trees.
As demonstrated in Figure 10, the most important predictive feature was *Bilirubin*, followed closely by *Ascites status*, *Prothrombin time*, and *Age*. These findings are consistent with the clinical understanding that has been formed on the progression of cirrhosis. According to this understanding, higher Bilirubin levels and prolonged Prothrombin time are signs of severe liver disease. In addition, the presence of Ascites, which is a sign of decompensated cirrhosis, is associated with a poor prognosis and an increased chance of death. In addition to Serum Albumin, Alkaline Phosphatase, and Platelet count, other informative aspects were also included. These factors are known to have connections with liver health and liver fibrosis. It is also interesting to note that demographic and less stressed characteristics, such as Hepatomegaly status and Sex, were also shown to be among the top-ranked variables. These could be secondary correlations or effects that are distinct to a group of people. Still, the fact the fact that they happen suggests that adding a more complete clinical context to predictive modeling can be helpful. The overall result of the feature importance analysis is that it supports both clinical expectations and the interpretability of the model, thereby providing insights into which characteristics have the most influence on survival classification.

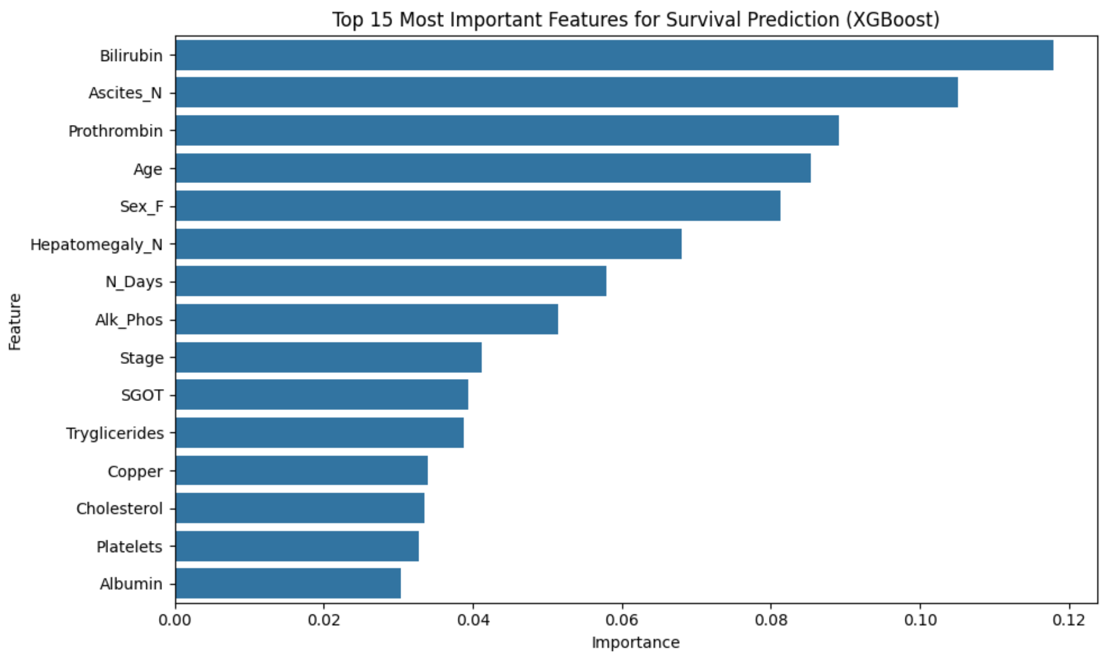

Figure 10: Top 15 Most Important Clinical Features According to XGBoost

4.2.3 Impact of Class Imbalance Strategies

Minimizing class imbalance became an essential part of improving the performance of all classifiers. The results show a clear pattern: models that used oversampling methods, especially SMOTE, usually did better than those that used undersampling or a mix of the two.

Table 2 illustrates that XGBoost, in conjunction with SMOTE, attained the greatest macro F1-score (0.602) and a robust macro ROC-AUC (0.853), indicating that oversampling facilitated the model's ability to effectively learn minority class patterns while maintaining performance on the majority class. Conversely, the identical model employing undersampling produced a markedly diminished macro F1-score (0.489) and ROC-AUC (0.782), suggesting that excessive reduction of the majority class may lead to performance deterioration due to the loss of information.

A comparable pattern is noted with Random Forest, wherein SMOTE and SMOTE combined with undersampling produced enhanced macro F1-scores (0.583–0.584) relative to undersampling alone (0.570). Even Logistic Regression, a more elementary linear model, demonstrated enhancement when combined with SMOTE-based methodologies.

Hybrid techniques such as SMOTE combined with Undersampling and SMOTEENN yielded inconsistent outcomes. Although they frequently enhanced recall (e.g., Logistic Regression with Undersampling attained the highest macro recall at 0.678), they did not uniformly result in superior overall F1-scores, indicating a compromise between sensitivity and precision. This highlights the necessity of synchronizing the imbalance technique with the principal assessment metric, namely, the macro F1-score particularly when the reduction of false negatives is paramount.

The results suggest that oversampling strategies, especially SMOTE, are good at fixing the imbalance in multiclass clinical datasets and making predictions more stable across models. Figure 5 shows how different resampling methods change the class distribution, with the SMOTE technique in particular showing a clear improvement in model performance, which aligns with this.

4.2.4 Hyperparameter Tuning and Grid Search

Hyperparameter adjustment was conducted using a grid search technique within cross-validation to enhance model performance. For every combination of model and resampling technique, the search was performed across a specified grid of hyperparameter values (e.g., regularization strength for logistic regression, number of neighbors for KNN, and tree depth and learning rate for ensemble models). The principal evaluation criterion for grid search was the macro-averaged F1-score, aligning with the study's focus on equitable performance across classes.

Table 5 indicates the optimal hyperparameter configurations determined for each model and imbalance technique. XGBoost combined with SMOTE attained the greatest F1-score of 0.602, utilizing a learning rate of 0.1 and 200 estimators. Likewise, Random Forest with SMOTE combined with undersampling exhibited optimal performance with an unconstrained tree depth and 200 estimators. Significantly, the majority of models preferred simpler configurations (e.g., KNN consistently opted for $k = 3$), indicating that performance improvements were mostly due to resampling tactics rather than increased hyperparameter complexity. These findings illustrate the significance of optimization, even for basic models, especially when combined with class imbalance correcting methods.

5 Conclusion and Future Work

5.1 Conclusion

The goal of this study was to build and evaluate machine learning models that predict cirrhosis patients' survival. The study had three main goals: finding the best classifier, finding the most important clinical features for patient survival, and determining which class imbalance mitigation techniques are effective for the best performance model.

The XGBoost classifier integrated with SMOTE was the top-performing model, attaining a macro-averaged F1-score of 0.602 and a macro ROC-AUC of 0.853. The results demonstrate the model's capacity to sustain robust predictive performance across all three survival categories. Feature importance analysis revealed Bilirubin, Albumin, Prothrombin time, and Ascites as the most important predictors, confirming the results of previously published clinical studies on the progression of liver function diseases.

Regarding the management of class imbalance, SMOTE exhibited the most reliable performance improvements, mainly when utilized with tree-based ensemble models. The findings underscore the significance of incorporating resampling algorithms into the modeling pipeline when dealing with imbalanced clinical data.

The work illustrates the capability of machine learning, particularly when integrated with meticulous preprocessing and resampling, to improve the precision and clarity of survival prediction models. These findings may facilitate more proactive and data-driven clinical decision-making in cirrhosis management, especially in identifying high-risk patients and prioritizing treatment.

5.2 Drawbacks of the used Methods

The ML models in this work have issues. The linear relationship between predictors and survival log-odds may not be accurate in complex medical datasets [8]. This assumption prevents it from describing non-linear interactions, which could misclassify patients with unusual illness progression. LR is susceptible to outliers, which modify coefficient values and reduce prediction accuracy. However, K-Nearest Neighbours(KNN) needs further development as the collection grows. Distance estimations get more expensive as feature space dimensions increase. Known as the "curse of dimensionality" [26]. Additionally, noisy or redundant features can harm KNN's classification abilities. When classes are imbalanced, KNN supports the majority class due to proximity-based voting.
However, Random Forest (RF) and XGBoost are strong ensemble algorithms with their issues. Random Forest models are called "black boxes" since each forecast is hard to interpret. Healthcare applications require model openness, making this a critical issue [10]. RF requires careful adjustment of the hyperparameter to avoid overfitting and may be challenging to run on computers with many trees. Despite its effectiveness, XGBoost is challenging to master due to its numerous hyperparameters, such as learning rate, tree depth, and regularization terms [11]. XGBoost can "overfit" to training data if not correctly tuned, reducing its generalization. This study addressed class imbalance with SMOTE, Random Undersampling, and Balanced Bagging. Each has issues. SMOTE can cause overfitting and information loss from the majority class due to false noise and random undersampling [14]. The Balanced Bagging Classifier works well but requires a lot of computer power. These issues demonstrate the trade-offs of using machine learning to predict clinical trial outcomes.

5.3 Future Research Ideas

This study performed a satisfactory task by using a publicly available dataset to find the best machine-learning model for predicting the survival of cirrhosis patients. It used standard class imbalance techniques to deal with the dataset's natural imbalance in the target variable. However, there are still many areas where more research could be done. One interesting direction is to add more or different clinical data sources. For instance, combining time-series data from labs, imaging results, or electronic health records (EHR) from different institutions could make the models more generalizable and better at making predictions. Adding sociodemographic and behavioral health characteristics may also provide new predictors and better show how patients differ in the actual world.

In terms of methods, future research could look at more advanced or specialized algorithms like LightGBM, CatBoost, or designs based on Deep Learning (DL), like recurrent neural networks, especially for medical data collected over time or in a sequence. Also, this study chose macro F1-score as the main way to compare performance across all classes fairly. However, if reducing false negatives becomes crucial in clinical decision-making, future research could focus on macro recall or cost-sensitive learning instead.

6 Ethics, Relevance, and Stakeholders

6.1 Ethical Considerations

The UCI Machine Learning Repository has the dataset used in this study to predict cirrhosis patients' survival [6, 7]. Ethical use is crucial as the data comes from medical records. Obtaining informed consent from patients is crucial in clinical research to ensure their data is used with their consent and awareness. This dataset is public and anonymous; therefore, we do not require patient permission to participate in a secondary study. However, it is morally wrong not to handle this data responsibly and make sure that the study's results don't hurt anyone or be used for bad purposes. The study's findings should be responsibly distributed to avoid misleading the medical community about cirrhosis prognosis.

This study also prioritizes data safety, fair AI projections, and bias-free machine learning algorithms. Most people know that predictive algorithms can take up biases from their data, which can lead to unequal treatment of some patients. Avoiding this requires data pretreatment measures like missing value handling and fairness awareness. In addition to performance measurements, model evaluation considers fairness indicators to ensure predictions are fair across patient types. This study aims to assist healthcare professionals in using AI safely while maintaining openness and responsibility.

6.2 Societal Relevance and Stakeholder Awareness

This finding has important effects on society, especially when it comes to making life better for those with liver cirrhosis. This condition kills and sickens many people throughout the world. Early and precise patient survival prediction makes it feasible to prioritize organ transplants, create more proactive and personalized treatment plans, and make better use of healthcare resources. This work helps the growing efforts to use data-driven intelligence in healthcare. The ultimate goal is to minimize unnecessary consequences and improve patients' quality of life by utilizing machine learning to improve clinical decisions.

One of the key reasons for this study is to make important stakeholders, like patients, data scientists, clinicians, and healthcare officials, more aware of the issues. The study shows how important it is to make models that are fair and easy to understand, fit in with clinical workflow, and take into account the real concerns of healthcare workers. Transparency and ethics have also been emphasized to ensure that machine learning predictions are used fairly and reasonably, especially in industries with a lot at stake, like health. So, the research improves technical skills and builds trust and teamwork among people who work with patients.

References

[1] Sumeet K Asrani, Harshad Devarbhavi, John Eaton, and Patrick S Kamath. Burden of liver diseases in the world. *Journal of hepatology*, 70(1):151–171, 2019.

[2] David S Goldberg and Afsoon Zarnegarnia. Longitudinal prediction of survival in patients with cirrhosis: A dynamic artificial intelligence model. *Hepatology Communications*, 7(11):e02709, 2023.

[3] Zhongcheng Bai, Yichen Xu, Nan Li, Zheng Lin, Hong Liu, and Lihua Zhang. Prediction of in-hospital mortality in patients with liver cirrhosis and hyponatremia using artificial neural network. *Scientific Reports*, 12(1):14928, 2022.

[4] Ian Sample. Ai tool diagnoses coeliac disease in seconds, scientists say, 2024. Accessed: 2025-03-29.

[5] Harry Neely, Arif Awan, Timothy Robbins, Paul Bassett, Ramesh Arasaradnam, Nasir Rajpoot, Elizabeth A. Montgomery, M. Zameer U. Shah, Justin P. Bishop, Pietro R. Spagnolo, et al. Machine learning achieves pathologist-level coeliac disease diagnosis. *NEJM AI*, 1(2):Article e2400738, 2024.

[6] E. Dickson, P. Grambsch, T. Fleming, L. Fisher, and A. Langworthy. Cirrhosis patient survival prediction. UCI Machine Learning Repository, 1989. [Dataset].

[7] UCI Machine Learning Repository. Cirrhosis patient survival prediction dataset, 2023. `https://archive.ics.uci.edu/dataset/878/cirrhosis+patient+survival+prediction+dataset-1`.

[8] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer, 2013.

[9] David W Hosmer, Stanley Lemeshow, and Rodney X Sturdivant. *Applied Logistic Regression*. John Wiley & Sons, 2013.

[10] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[11] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[12] John W. Tukey. *Exploratory Data Analysis.* Addison-Wesley, 1977.

[13] David Pyle. *Data Preparation for Data Mining.* Morgan Kaufmann, 1999.

[14] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[15] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[16] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[17] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. 6(1):20–29, 2004.

[18] Max Kuhn and Kjell Johnson. *Applied Predictive Modeling.* Springer, 2013.

[19] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* O'Reilly Media, 2019.

[20] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[21] Jason Brownlee. *Machine Learning Mastery With Python.* Machine Learning Mastery, 2019.

[22] Pedro J Garc'ia-Laencina, Jose L Sancho-Gomez, and Anibal R Figueiras-Vidal. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19(2):263–282, 2010.

[23] C Ferri, J Hernández-Orallo, and R Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38, 2009.

[24] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[25] Scikit-learn Developers. Classification metrics: F1-score. `https://scikit-learn.org/stable/modules/model_evaluation.html#f1-score`, 2023. Accessed: 2025-05-22.

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

7 Appendix

Figures and Tables

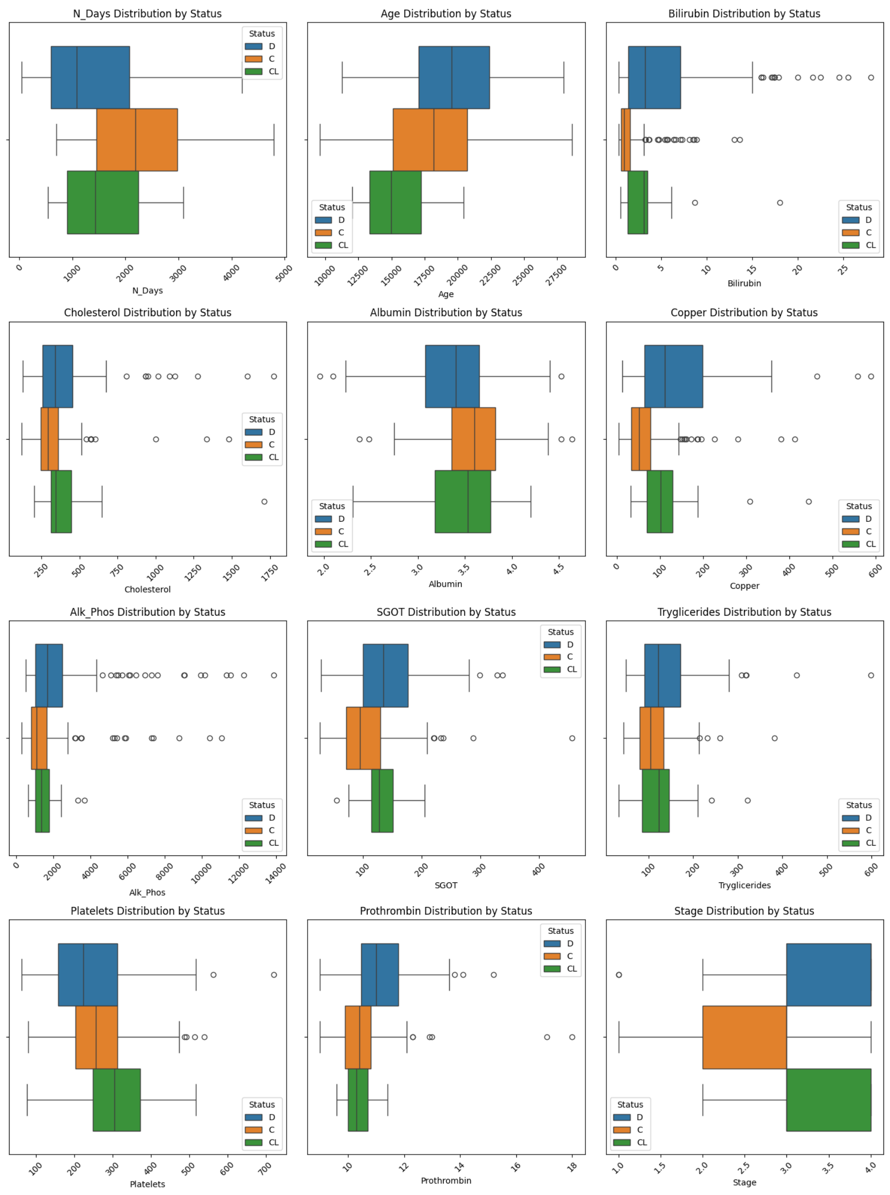

Figure 11: Distribution of Continuous Features by Status

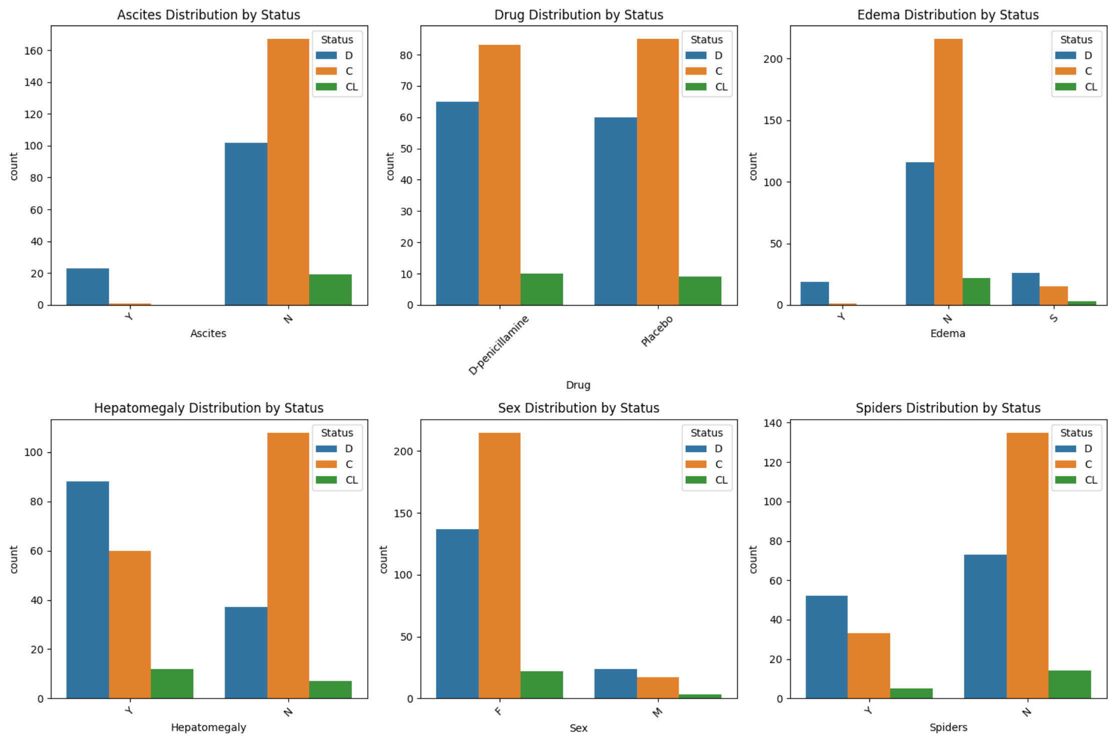

Figure 12: Distribution of Categorical Features

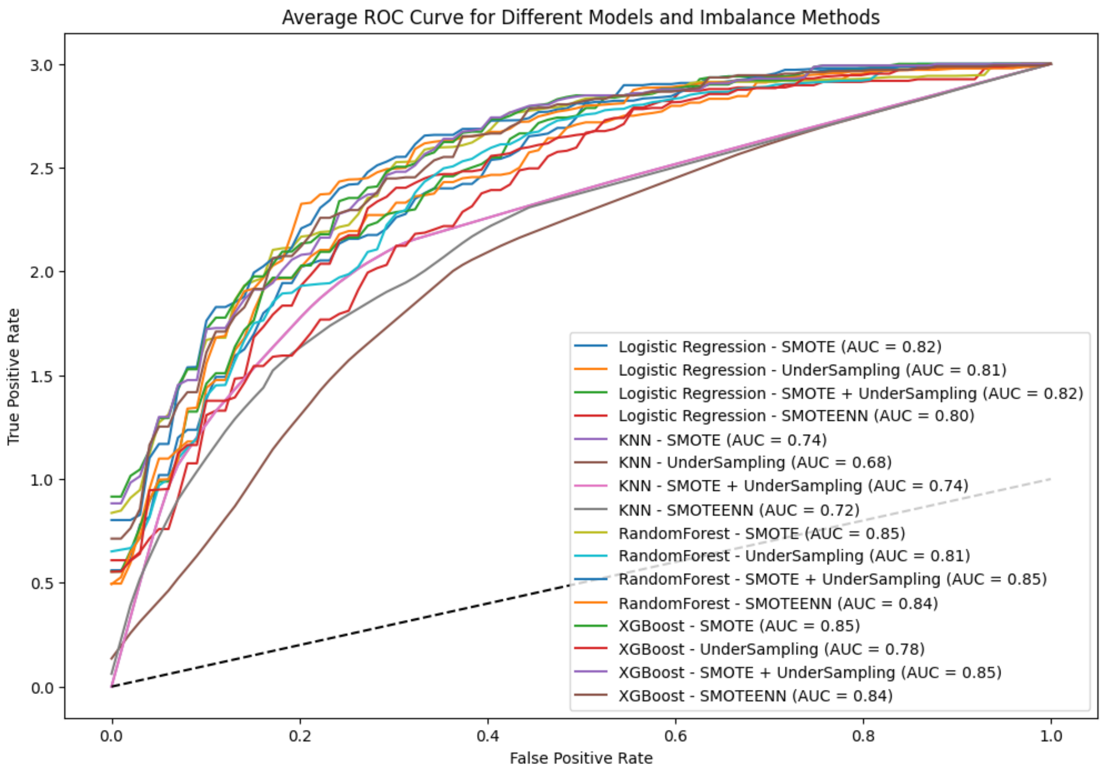

Figure 13: Average ROC Curve for Different Models and Imbalance Methods

Table 4: Model Evaluation Metrics Summary sorted by Mean F1-Score

Model	**Imbalance Method**	**Mean F1-score**	**Mean Macro Recall**	**Mean ROC-AUC**	**Best Params**
XGBoost	SMOTE	0.602	0.598	0.853	{'learning_rate': 0.1, 'n_estimators': 200}
XGBoost	SMOTE + UnderSampling	0.593	0.593	0.850	{'learning_rate': 0.1, 'n_estimators': 200}
RandomForest	SMOTEENN	0.589	0.633	0.846	{'max_depth': None, 'n_estimators': 100}
XGBoost	SMOTEENN	0.583	0.650	0.843	{'learning_rate': 0.1, 'n_estimators': 100}
Logistic Regression	SMOTE + UnderSampling	0.581	0.657	0.821	{'C': 0.1}
Logistic Regression	UnderSampling	0.576	0.678	0.811	{'C': 0.1}
RandomForest	SMOTE	0.570	0.570	0.854	{'max_depth': 10, 'n_estimators': 100}
RandomForest	SMOTE + UnderSampling	0.565	0.558	0.854	{'max_depth': None, 'n_estimators': 200}
RandomForest	UnderSampling	0.561	0.663	0.810	{'max_depth': None, 'n_estimators': 200}
Logistic Regression	SMOTE	0.560	0.636	0.817	{'C': 10}
Logistic Regression	SMOTEENN	0.543	0.630	0.802	{'C': 10}
KNN	SMOTE	0.524	0.572	0.741	{'n_neighbors': 3}
KNN	SMOTE + UnderSampling	0.524	0.572	0.741	{'n_neighbors': 3}
XGBoost	UnderSampling	0.489	0.561	0.782	{'learning_rate': 0.1, 'n_estimators': 200}
KNN	SMOTEENN	0.477	0.582	0.722	{'n_neighbors': 3}
KNN	UnderSampling	0.446	0.542	0.681	{'n_neighbors': 3}

Table 5: Best Hyperparameters Selected via Grid Search

Model	**Imbalance Method**	**Best Parameters**
XGBoost	SMOTE	{learning_rate=0.1, n_estimators=200}
RandomForest	SMOTE + Undersampling	{max_depth=None, n_estimators=200}
Logistic Regression	Undersampling	{C=0.1}
KNN	All	{n_neighbors=3}

Table 6 presents the detailed classification report of the final XGBoost model trained with SMOTE. The table includes per-class precision, recall, F1-score, and support, along with macro and weighted averages, as calculated on the test set.

Table 6: Classification report for the final XGBoost + SMOTE model

Class	**Precision**	**Recall**	**F1-score**	**Support**
C	0.85	0.74	0.80	47
CL	0.14	0.40	0.21	5
D	0.72	0.66	0.69	32
Accuracy			0.69	84
Macro avg	0.57	0.60	0.56	84
Weighted avg	0.76	0.69	0.72	84

Python Code

1. Import Libraries and Setup

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split,
   StratifiedKFold, GridSearchCV
from sklearn.metrics import accuracy_score,
   precision_recall_fscore_support, roc_auc_score, roc_curve
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.combine import SMOTEENN
from sklearn.preprocessing import LabelEncoder, StandardScaler,
   OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
import warnings
warnings.filterwarnings("ignore")
```

2. Load and Preprocess Data

```
# Load dataset
df = pd.read_csv("/Uscirrhosis.csv")
df.drop(columns=["ID"], inplace=True)
X = df.drop(columns=["Status"])
y = df["Status"]

# Identify categorical and numerical columns
categorical_cols = X.select_dtypes(include=["object"]).columns
```

```
numerical_cols = X.select_dtypes(exclude=["object"]).columns

# Define preprocessing pipeline
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
preprocessor = ColumnTransformer(transformers=[
    ('num', numeric_transformer, numerical_cols),
    ('cat', categorical_transformer, categorical_cols)
])

# Apply preprocessing
X_processed = preprocessor.fit_transform(X)
```

3. Split Dataset

```
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X_processed, y
    , test_size=0.2, random_state=42, stratify=y)
# Encode target variable
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
```

4. Define Models and Hyperparameters

```
# Define models
models = {
    "Logistic Regression": LogisticRegression(solver='liblinear',
        random_state=42),
    "KNN": KNeighborsClassifier(),
    "RandomForest": RandomForestClassifier(random_state=42),
    "XGBoost": XGBClassifier(eval_metric='logloss', random_state
       =42)
}

# Define hyperparameter grids
param_grids = {
    "Logistic Regression": {'C': [0.1, 1, 10]},
    "KNN": {'n_neighbors': [3, 5, 7]},
    "RandomForest": {'n_estimators': [100, 200], 'max_depth': [
       None, 10]},
    "XGBoost": {'learning_rate': [0.01, 0.1], 'n_estimators':
       [100, 200]}
}
```

5. Define Imbalance Handling Techniques

```
# Define imbalance handling techniques
imbalance_methods = {
    "SMOTE": SMOTE(sampling_strategy='auto', random_state=42),
    "UnderSampling": RandomUnderSampler(sampling_strategy='auto',
       random_state=42),
    "SMOTE + UnderSampling": lambda X, y: RandomUnderSampler(
       random_state=42).fit_resample(SMOTE(random_state=42).
       fit_resample(X, y)[0], SMOTE(random_state=42).fit_resample(
       X, y)[1]),
    "SMOTEENN": SMOTEENN(random_state=42)
}
```

6. Model Evaluation with Cross-Validation and F1-Score

```
# Store results with best parameters
results = []
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
# Set up ROC curve plot
plt.figure(figsize=(12, 8))

for model_name, model in models.items():
```

```
for method_name, method in imbalance_methods.items():
    f1_scores = []
    macro_recalls = []
    roc_auc_scores = []
    mean_fpr = np.linspace(0, 1, 100)
    mean_tpr = np.zeros_like(mean_fpr)
    best_params_list = []
    for train_idx, val_idx in skf.split(X_train,
       y_train_encoded):
        X_train_fold, X_val_fold = X_train[train_idx], X_train
           [val_idx]
        y_train_fold, y_val_fold = y_train_encoded[train_idx],
            y_train_encoded[val_idx]
        # Apply imbalance method
        if callable(method):
            X_train_fold, y_train_fold = method(X_train_fold,
               y_train_fold)
        else:
            X_train_fold, y_train_fold = method.fit_resample(
               X_train_fold, y_train_fold)
        # Hyperparameter tuning (F1 macro-based)
        param_grid = param_grids.get(model_name, None)
        if param_grid:
            grid_search = GridSearchCV(model, param_grid, cv
               =3, scoring='f1_macro', n_jobs=-1)
            grid_search.fit(X_train_fold, y_train_fold)
            model = grid_search.best_estimator_
            best_params_list.append(grid_search.best_params_)
        else:
            model.fit(X_train_fold, y_train_fold)
            best_params_list.append("Default")
        # Evaluate
        y_pred = model.predict(X_val_fold)
        y_pred_prob = model.predict_proba(X_val_fold)

        f1 = precision_recall_fscore_support(y_val_fold,
           y_pred, average='macro')[2]
        macro_recall = recall_score(y_val_fold, y_pred,
                average='macro')
            roc_auc = roc_auc_score(y_val_fold, y_pred_prob,
                multi_class='ovr', average='macro')

            f1_scores.append(f1)
            macro_recalls.append(macro_recall)
            roc_auc_scores.append(roc_auc)
            for i in range(y_pred_prob.shape[1]):
                fpr, tpr, _ = roc_curve(y_val_fold == i,
                    y_pred_prob[:, i])
                mean_tpr += np.interp(mean_fpr, fpr, tpr)

        mean_tpr /= skf.get_n_splits()
        plt.plot(mean_fpr, mean_tpr, label=f"{model_name} - {
            method_name} (AUC = {np.mean(roc_auc_scores):.2f})")

        # best parameters
        from collections import Counter
        final_best_params = Counter(map(str, best_params_list)).
            most_common(1)[0][0]
        results.append([
            model_name,
            method_name,
            np.mean(f1_scores),
            np.mean(macro_recalls),
            np.mean(roc_auc_scores),
            final_best_params
        ])

plt.plot([0, 1], [0, 1], 'k--')
plt.title("Average ROC Curve for Different Models and Imbalance
    Methods")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.show()
```

7. Model Comparison Summary

```
# Model comparison summary result
results_df = pd.DataFrame(results, columns=[
    "Model", "Imbalance_Method", "Mean F1-score", "Mean Macro
       Recall",
    "Mean ROC-AUC", "Best Params"
])
print("\nFull Evaluation Summary:")
results_df
```

8. Visualize Model Comparison by F1-score

```
# Visualization of Model comparison using F1-score
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12, 6))
sns.barplot(data=results_df, x="Model", y="Mean F1-score", hue="
   Imbalance_Method")
plt.title("Comparison of ML Models and Resampling Methods by Macro
    F1-score")
plt.ylabel("Mean F1-score")
plt.ylim(0.45, 0.65)
plt.xticks(rotation=45)
plt.legend(title="Resampling Method", bbox_to_anchor=(1.05, 1),
   loc='upper left')
plt.tight_layout()
plt.show()
```

9. Train and Evaluate the Best Model

```
# Train best model
# Rebuild and train best model on full training data
xgb_best = XGBClassifier(learning_rate=0.1, n_estimators=200,
   eval_metric='logloss', use_label_encoder=False)
```

```
# Apply SMOTE to the training set
smote = SMOTE(random_state=42)
X_train_bal, y_train_bal = smote.fit_resample(X_train,
   y_train_encoded)

# Fit model
xgb_best.fit(X_train_bal, y_train_bal)

# Best model Evaluation
import numpy as np
import pandas as pd
from xgboost import XGBClassifier
from sklearn.metrics import classification_report,
   confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from imblearn.ensemble import BalancedBaggingClassifier

# Load and preprocess data
df = pd.read_csv("cirrhosis.csv")
df.drop(columns=["ID"], inplace=True)
X = df.drop(columns=["Status"])
y = df["Status"]
# Preprocessing pipeline
numeric_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])
categorical_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
preprocessor = ColumnTransformer([
    ('num', numeric_transformer, numerical_cols),
    ('cat', categorical_transformer, categorical_cols)
])
X_processed = preprocessor.fit_transform(X)
```

```
# Encode target
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Split dataset (stratify ensures CL is represented in both sets)
X_train, X_test, y_train, y_test = train_test_split(X_processed,
   y_encoded, test_size=0.2, stratify=y_encoded, random_state=42)

# Train model
bbc = BalancedBaggingClassifier(
    estimator=XGBClassifier(learning_rate=0.1, n_estimators=200,
       eval_metric='mlogloss', use_label_encoder=False),
    sampling_strategy='auto',
    n_estimators=10,
    replacement=False,
    random_state=42,
    n_jobs=-1
)

bbc.fit(X_train, y_train)
y_pred = bbc.predict(X_test)

# Evaluation
from sklearn.metrics import classification_report,
   confusion_matrix, ConfusionMatrixDisplay
report = classification_report(y_test, y_pred, target_names=
   label_encoder.classes_)
print(report)

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=
   label_encoder.classes_)
disp.plot()
```