**MS1 Report: Insured or Not Program**  
**Team 5**

Team Members:

Kavyanjali 2203390

Koh Chun Ince 2200960

Caleb Lee Jia Le 2201250

Muhammad Muslinmin Bin Mohd Saleh 2201360

Reuben Low Yu Xiang 2200931

Stafford Ho Sheng Xian 2201218

Yugendren S/o Sooriya Moorthi 2201512

Course: RSE4207 Artificial Intelligence and Machine Learning

Date: 06/11/2024

Table of Contents

[1 Introduction 2](#_Toc1822526927)

[1.1Project Objective 3](#_Toc1049322433)

[1.2Background 4](#_Toc522733506)

[2 Exploratory Data Analysis 4](#_Toc1364346539)

[2.1Analysis Activities 5](#_Toc816216322)

[2.2Processing Activities 22](#_Toc1626202380)

[2.3 Orange Pipeline 29](#_Toc854832175)

[2.4 Code pipeline 31](#_Toc108605877)

[2.5 CLI 31](#_Toc1889327650)

[2.6 GUI 31](#_Toc1335941902)

[3. Model Training 33](#_Toc759163860)

[3.1 Models assessed 33](#_Toc843388021)

[3.2 Training process 33](#_Toc190669544)

[3.3 Testing and Evaluation 33](#_Toc443182027)

[4. Model Testing 33](#_Toc1540670833)

[4.1 Overfitting and Underfitting 33](#_Toc869715121)

[4.2 Results 33](#_Toc542224576)

[5 References 33](#_Toc749066317)

## 

## 1 Introduction

In the field of insurance and risk management, data-driven decision-making plays a critical role in assessing potential clients' risks and determining appropriate premiums. Accurate risk assessment is crucial for insurers to ensure that they are adequately prepared for high-value claims, especially in the case of accidents that lead to loss of life. This project aims to address these challenges by employing both exploratory data analysis (EDA) and machine learning to analyze passenger data and predict survival outcomes during maritime accidents.

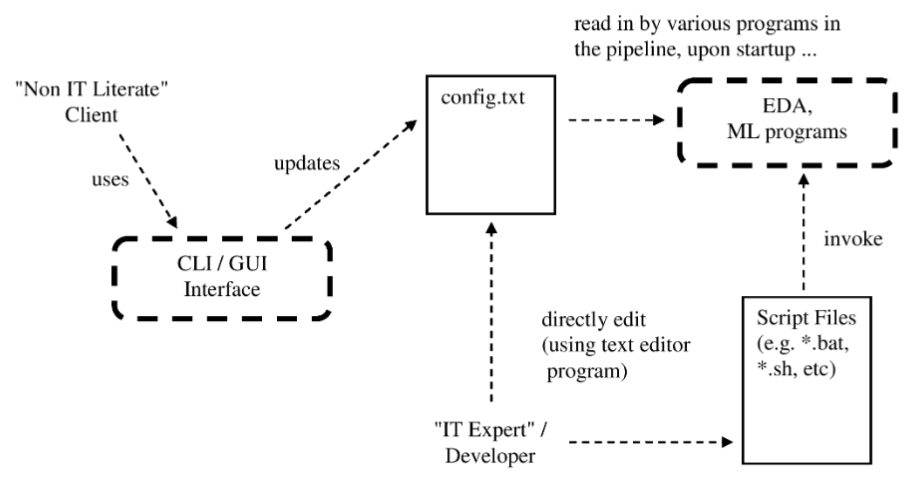
The project aims to uncover key characteristics that influence survival rates in maritime accidents, using these insights to inform insurance-related decisions. The analysis will also explore the possibility of developing a machine learning model capable of predicting survival likelihood, thus aiding in the automation of the underwriting process which refers to the exchange between risks and payment.

### 1.1 Project Objective

The objective of the project for the first milestone consists of:

* **Exploratory Data Analysis (EDA):** To analyze the dataset provided by the client (maritime company) and extract meaningful insights regarding passengers who survived and those who did not. Relationships between the variables from the dataset such as age, gender, ticket class etc. will be investigated to determine their impact on survival rates.
* **Machine Learning Model Development (ML):** To develop and evaluate a machine learning model that can predict the survival rate based on passenger characteristics. This model serves as a tool for the client to determine insurance eligibility and pricing for the applicants of its insurance products.

To design a program similar to Figure 1, for the client to help them decide whether certain individuals are eligible to be insured or not. For example, if an individual is likely to survive a maritime accident, then it could be eligible for insurance. Ideally, the program should be easily configurable and integrated into the client’s existing IT infrastructure and OS platform.



**Figure 1: Overall Program Flow**

### 1.2 Background

The scenario outlined in this project involves a business client of Data Mine Craft DMC, a company specializing in generating analytics based on datasets. Two datasets have been provided: one for exploratory analysis and model training, and the other for evaluating the machine learning model’s performance.

In the event of maritime accidents, casualties and injuries can lead to significant financial claims. Insurance claims are often highest for passengers who lose their lives. Therefore, understanding the factors that affect survival rates is crucial for risk assessment and determining the price of premiums. The dataset contains critical information such as passenger age, gender, cabin, and whether or not they survived the accident.

### 

## 

## 2 Exploratory Data Analysis

The goal of the Exploratory Data Analysis (EDA) for this project is to understand the underlying characteristics and patterns within the passenger dataset, which will aid in developing a predictive model for survival likelihood during maritime accidents. This analysis will focus on identifying key features that influence survival rates and deriving insights that can inform insurance-related decisions. EDA aims to:

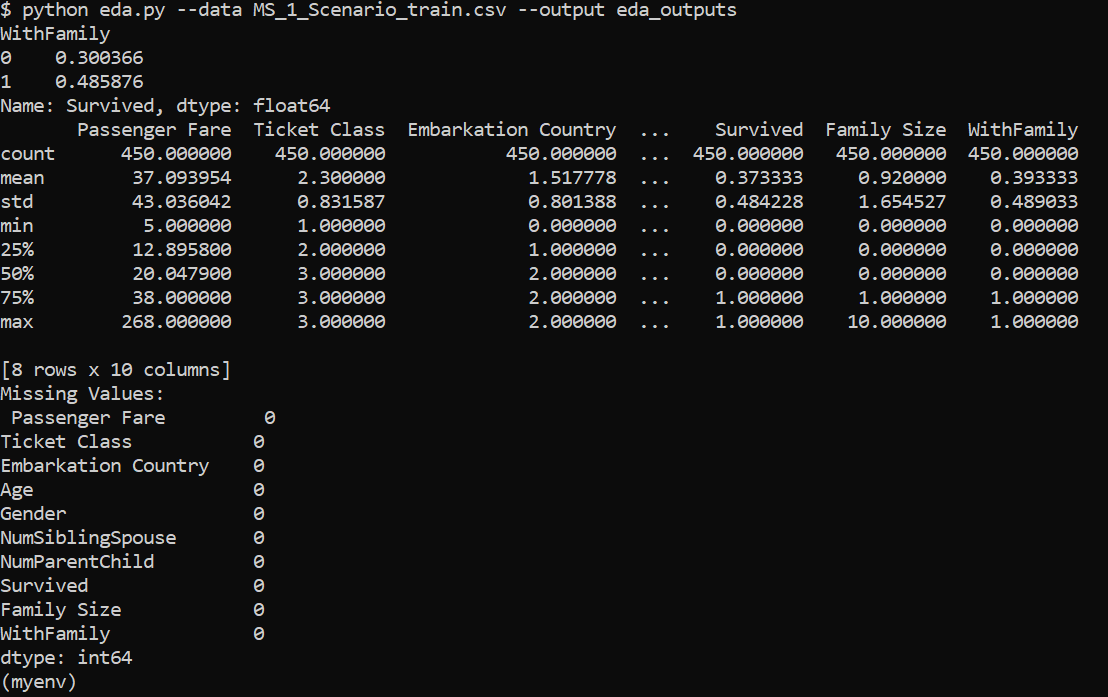
1. **Identify Patterns and Trends:** Examine relationships between various passenger attributes (e.g., ticket class, age, gender) and their survival outcomes to understand which factors significantly impact survival rates.
2. **Handle Missing and Anomalous Data:** Identify and address any missing or anomalous values that may distort the analysis and model training.
3. **Feature Selection:** Select relevant features that provide valuable insights into survival likelihood and exclude redundant or irrelevant variables.
4. **Visualize Data for Insights:** Generate visualizations to help illustrate key relationships, such as the survival rate by ticket class or age, and to enhance interpretability for stakeholders.

This analysis is based on hypotheses that will be formulated and evaluated to uncover key patterns within the training dataset. Each hypothesis is designed to test specific factors potentially influencing survival rates. By examining these hypotheses and supporting them with evidence, we aim to extract actionable insights that contribute to a more robust predictive model. The ultimate goal of this EDA is to inform the insurance decision-making process, guiding the assessment of premium rates and eligibility for prospective clients.

### 2.1 Analysis Activities

It systematically examines various hypotheses regarding demographic and ticket-related characteristics, aiming to understand which features might be predictive of survival.

#### 2.1.1 Statistical Summary and Distribution Analysis



**Figure 2: Mean, Median, Minimum, Maximum, Standard deviation, etc.**

The summary statistics in Figure 1 (mean, median, minimum, maximum, standard deviation, etc.) offer valuable insights into the dataset’s distribution and can guide hypotheses or help interpret patterns in survival analysis.

##### 2.1.1.1 Mean (Average)

**Passenger Fare:** The mean fare of 37.09 suggests that fares vary widely, as indicated by the high standard deviation. This could indicate a mix of passengers across different ticket classes, with some paying much higher fares than others.

**Family Size:** The mean of 0.92 shows that the majority of passengers had small families on board. Most passengers might have traveled alone or with only one family member.

**Survived:** With a mean survival rate around 0.38, it indicates that approximately 38% of passengers survived, which is lower than half, suggesting that survival was challenging overall.

##### 2.1.1.2 Median (50% or Middle Value)

**Passenger Fare:** The median fare of 20.05 is much lower than the mean, suggesting a right-skewed distribution (a few passengers with very high fares skew the average upwards).

**Survived:** The median value of 0 (assuming "Survived" is coded as 0 for non-survivors and 1 for survivors) means that more than half of the passengers did not survive.

**Family Size:** The median family size of 0 suggests that many passengers traveled alone.

##### 2.1.1.3 Minimum and Maximum

**Passenger Fare:** The minimum fare is 0, possibly indicating free or complimentary tickets for some passengers. The maximum fare of 268 suggests that some passengers paid premium prices, likely for first-class tickets.

**Family Size:** The minimum family size of 0 and maximum of 10 indicate a wide range of family sizes, though most passengers have small or no families (as indicated by the lower quartile values).

**With Family:** The minimum and maximum of 0 and 1 for "WithFamily" suggest this feature is binary, indicating whether or not a passenger was traveling with family.

##### 2.1.1.4 Standard Deviation (Measure of Spread)

**Passenger Fare:** A high standard deviation (43.04) for fare indicates large variability in ticket prices, likely reflecting the different classes (first, second, and third).

**Family Size:** The standard deviation of 1.65 for family size suggests that most passengers have around 0-2 family members with them, with fewer large families onboard.

##### 2.1.1.5 Potential Hypotheses Based on This Analysis

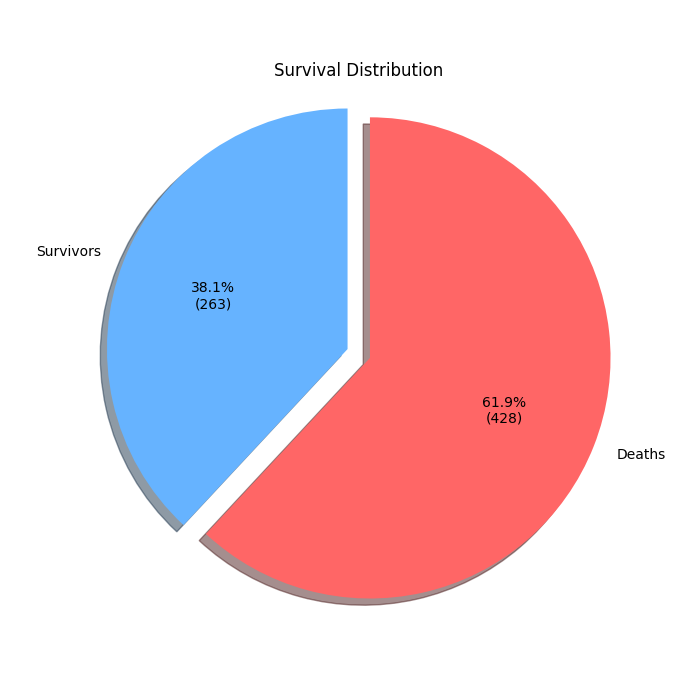
**Ticket Class and Survival:** Given the wide range in fares and right-skewed distribution, we might hypothesize that passengers paying higher fares (likely in first class) had higher survival rates, perhaps due to preferential access to lifeboats or proximity to exits.

**Family Size and Survival:** Since the average family size is close to 1 and many passengers are traveling alone (as indicated by the median), we might explore if passengers with family members onboard had a survival advantage, potentially due to mutual support.

**Embarkation and Survival:** If different embarkation points (e.g., "S," "C," or "Q") are linked to specific classes or fare levels, embarkation points might indirectly relate to survival rates. For example, one embarkation point might have had a higher proportion of first-class passengers with better survival rates.

These statistics provide a starting point for further analysis and visualization to understand the relationships between these features and survival outcomes.

#### 2.1.2 Correlation Analysis



**Figure 3: Majority of the Passengers Survival Rates**

Majority of the passengers (61.9%) died where only 38.1% of passengers survive.

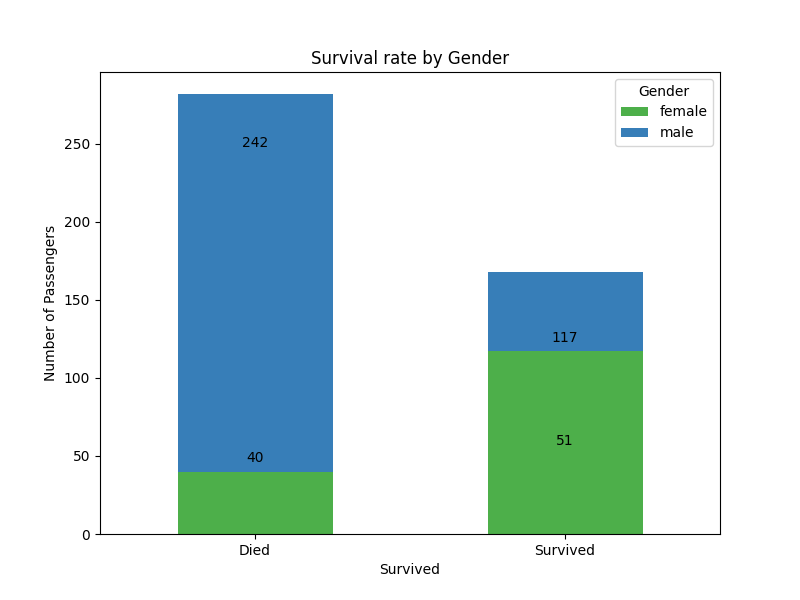
##### 2.1.2.1 Passengers with Family Members

**Hypothesis**: Passengers with family members onboard have a higher survival rate than those alone.

**Approach**: Created a "Family Size" feature based on the number of relatives (siblings/spouses and parents/children) each passenger had. Passengers were classified as "With Family" if they had a Family Size greater than 0, otherwise as "Alone."

**Results**: Passengers with family members had a higher survival rate (48.6%) compared to those alone (30.0%), supporting the hypothesis.

##### 2.1.2.1 Gender



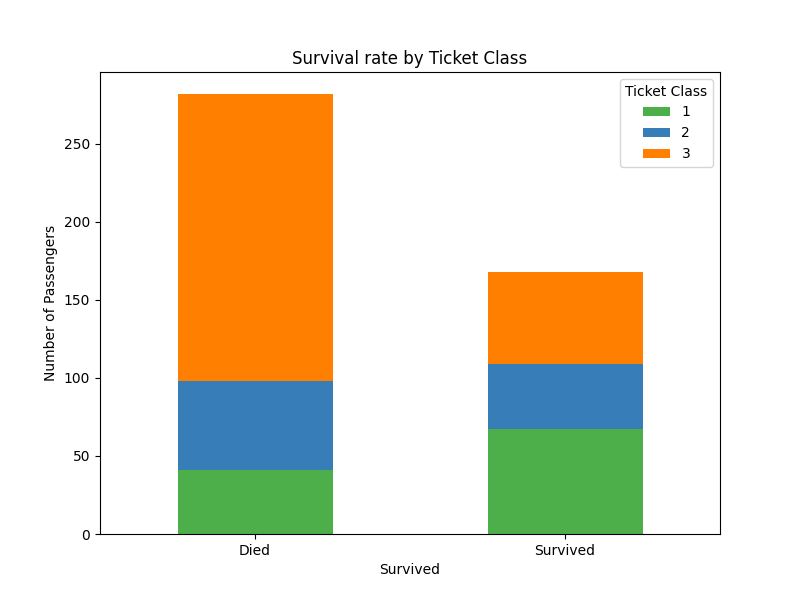
**Figure 4: Survival Rate by Gender**

**Hypothesis**: Gender influences survival rates, with men potentially having a survival advantage due to physical strength [1].

**Approach**: Analyzed survival rates by gender to examine differences, considering historical evacuation protocols.

**Results**: Women had a survival rate 2.19 times higher than men, likely due to "women and children first" protocols, where priority for lifeboat access was commonly given to women over men, disproving the initial hypothesis.

##### 2.1.2.3 Ticket Class



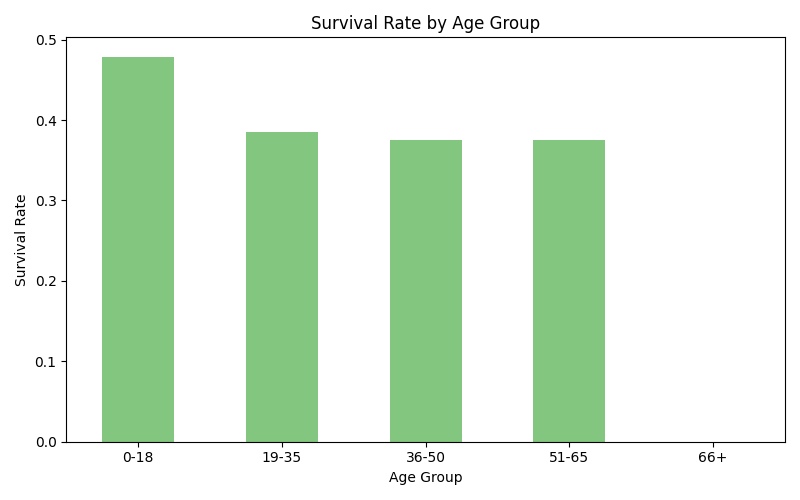
**Figure 1: Survival Rate by Ticket Class**

**Hypothesis**: Ticket class affects survival rates, with higher-class passengers having a better chance of survival.

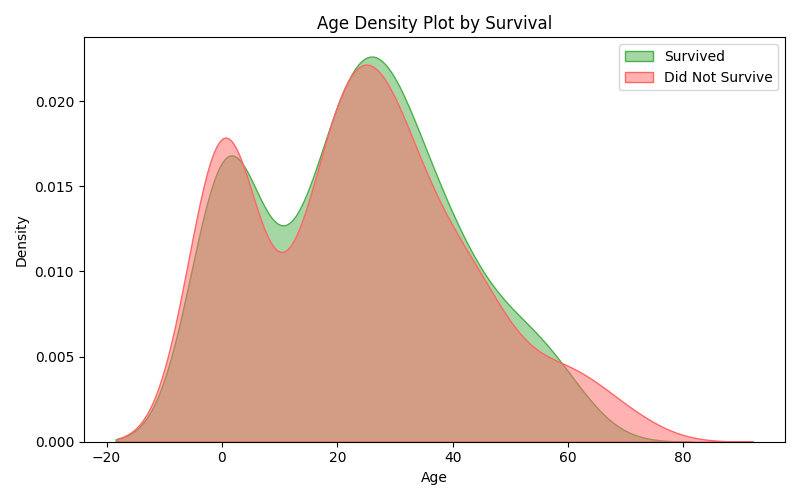
**Approach**: Compared survival rates across ticket classes.

**Results**: First-class passengers had the highest survival rate (62%), followed by second-class (42.4%) and third-class (24.2%), confirming that higher-class passengers had a survival advantage.

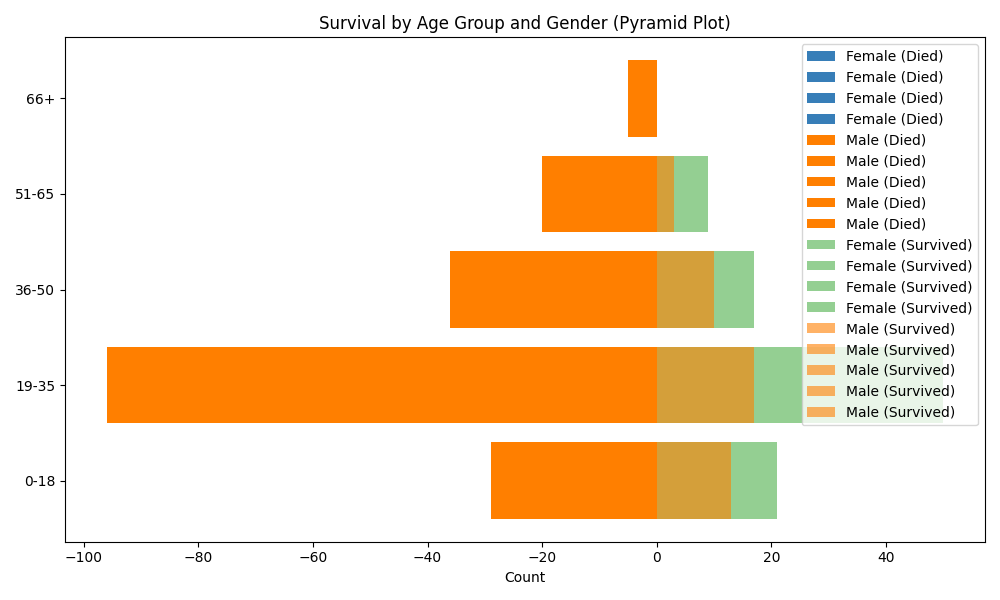
##### 2.1.2.4 Age

****

**Figure 1: Survival rate by Age Group**



**Figure 1: Age Density by Survival**



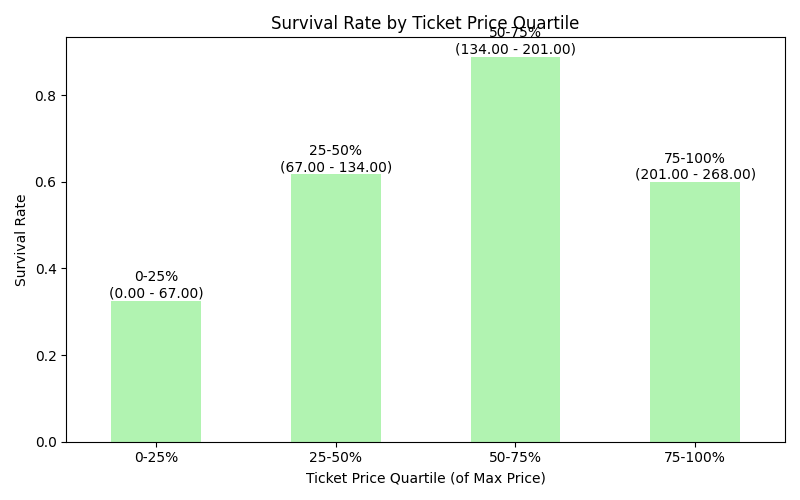
**Figure 1: Survival by Age Group and Gender (Pyramid Plot)**

**Hypothesis:** Younger passengers, particularly children and teenagers, have a higher survival rate than adults.

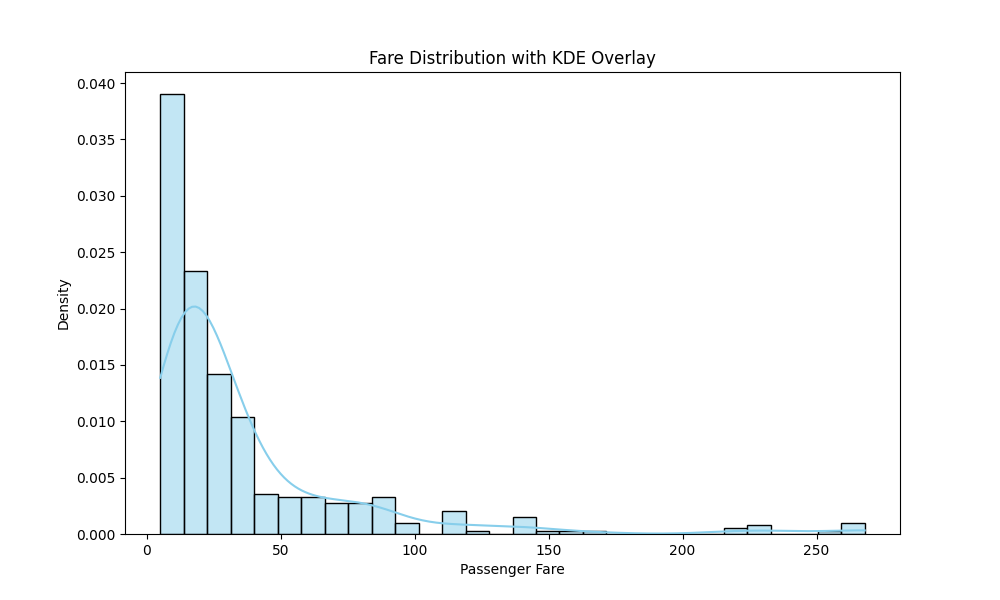
**Approach:** Analyzed survival rates by age group, focusing on the "0-18" age group versus other age groups. Utilized a bar chart to visualize survival by age group and a KDE plot to examine age distribution among survivors and non-survivors. Additionally, a pyramid plot was used to visualize survival by age and gender groups.

**Results:** The "0-18" age group had a survival rate close to 50%, while survival rates for older age groups hovered around 30-35%. The KDE plot indicated a peak in survival for younger children (0-10 years), with a secondary, less pronounced peak for survivors in their 30s. The pyramid plot showed that both male and female passengers under 18 had higher survival counts compared to older age groups, with particularly high non-survival rates for males aged 19-35. This supports the hypothesis that younger passengers had better survival odds, likely due to prioritization of children in rescue efforts.

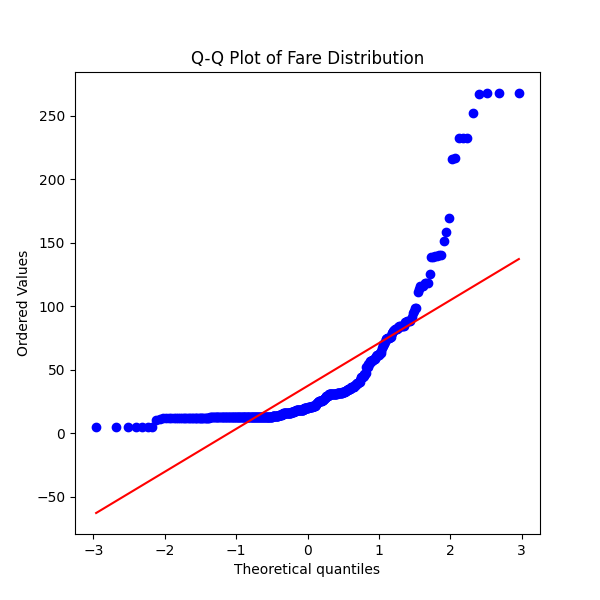
##### 2.1.2.5 Passenger Fare

****

**Figure 1: Survival Rate by Ticket Price Outline**



**Figure 1: Fare Distribution with KDE Overlay**



**Figure 1: Q-Q Plot of Fare Distribution**

**Hypothesis:** Higher ticket prices correlate with a higher chance of survival.

**Approach:** Examined survival rates across ticket price quartiles to observe any relationship between fare price and survival. Used a histogram to assess fare distribution and a Q-Q plot to analyze normality and distribution skewness.

**Results:** Passengers in the lowest fare quartile (0.00 - 67.00) had the lowest survival rate (20-30%), while survival rates increased with higher fare quartiles, peaking in the third quartile (134.00 - 201.00). However, survival rates declined slightly in the highest fare quartile (201.00 - 268.00), suggesting other factors, such as family presence, might also influence survival. The histogram showed a skewed distribution with most fares at the lower end, confirming the need for normalization (e.g., log transformation or binning) before model training. The Q-Q plot confirmed a right-skewed distribution, further justifying normalization for accurate model training.

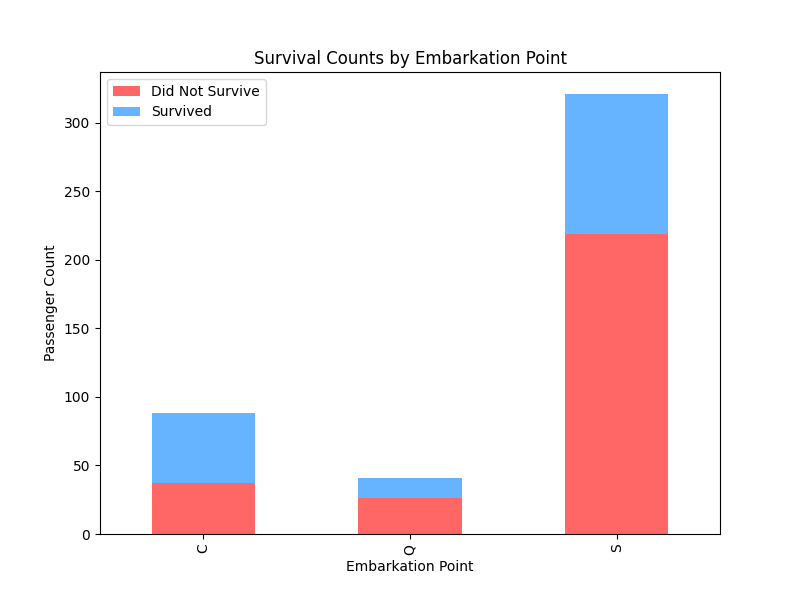
2.1.2.6 Embarkation Point

**A graph of a passenger count

Description automatically generatedA graph of a graph

Description automatically generated with medium confidence**

**Figure 13: Passenger Count by Embarkation Point      Figure 14: Survival Rate by Embarkation Point**

****

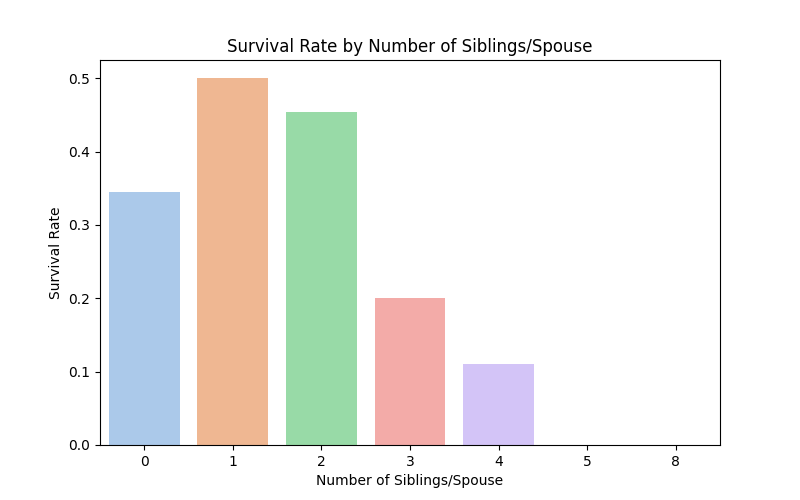
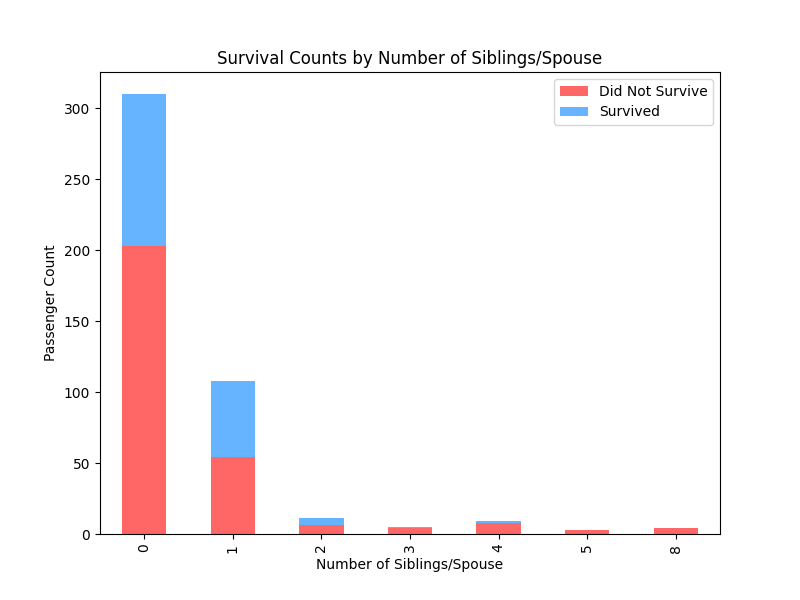
**Figure 1: Survival Counts by Embarkation Point**

**Hypothesis:** The embarkation point "C" has a higher survival rate compared to points "S" and "Q" due to a higher proportion of female/children passengers or first-class passengers.

**Approach:** Analyzed survival rates by embarkation point to determine if the survival advantage at point "C" is linked to the passenger demographics. Examined the distribution of gender, age, and ticket class for passengers from each embarkation point ("S," "C," and "Q") to assess whether these factors contribute to the observed differences in survival rates.

**Results:** Passengers who embarked from "C" had a survival rate above 50%, while those from "S" and "Q" had survival rates below 40%. Despite having fewer passengers than "S," the survival count for "C" was notably higher. This suggests that passengers from "C" had a more favorable likelihood of survival. The findings indicate that the higher survival rate for "C" could be due to a greater presence of women, children, or first-class passengers, who were prioritized for survival during evacuation. If the embarkation feature is used in model training, normalization may be required due to the disproportionate number of passengers from "S" compared to "C" and "Q."

##### 2.1.2.7 Number of Siblings or Spouses Onboard



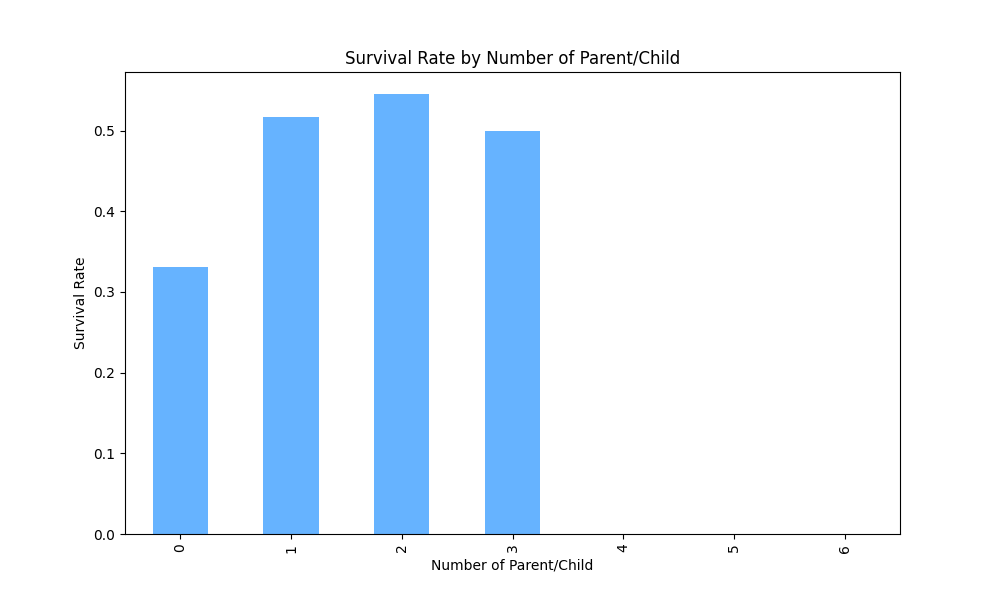
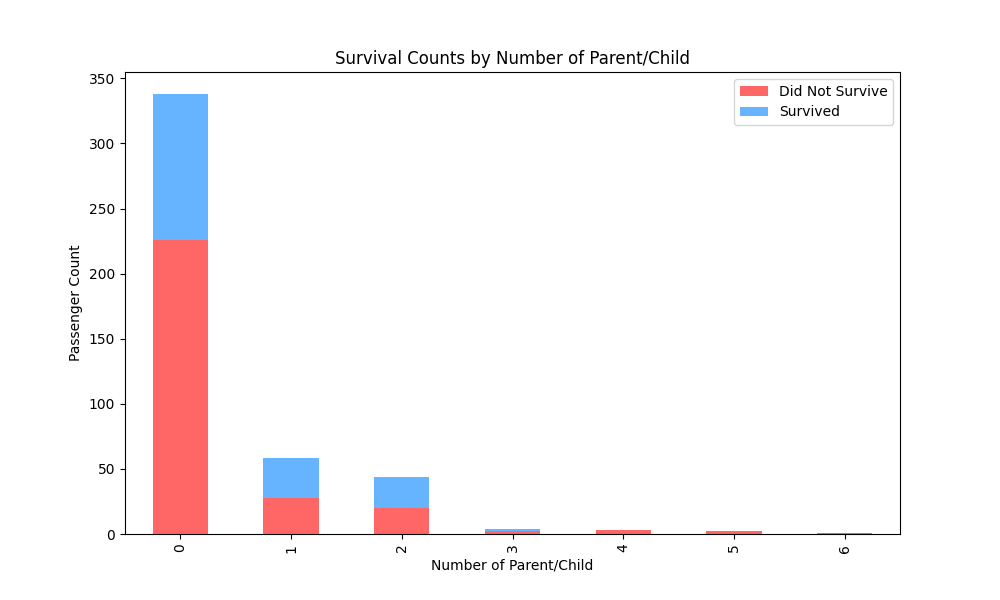
**Figure 1: Survival Counts by Number of**  **Figure 1: Survival Rate by Number of**  
**Siblings/Spouse**  **Siblings/Spouse**

**Hypothesis**: The number of siblings/spouse on board affects survival chances, with the hypothesis that having family members offers mutual support, which could improve survival rates during evacuations.

**Approach**: Examined survival rates based on the number of siblings or spouses each passenger had on board. Compared survival counts across different sibling/spouse categories to identify trends in survival likelihood, focusing on whether the presence of one or more family members positively influences survival.

**Results**: Passengers with one sibling or spouse had the highest survival rate, at almost 50%, while those with no siblings or spouses had a lower survival rate. Beyond one sibling/spouse, survival rates dropped significantly, with almost no survivors among those with three or more siblings/spouses. This suggests that while having one family member may provide support that aids survival, additional family members beyond one may complicate evacuation coordination, leading to diminishing returns on survival benefits.

##### 2.1.2.8 Parents or Children Onboard



**Figure 1: Survival Counts by Number of**  **Figure 1: Survival Rate by Number of**  
**Parent/Child**  **Parent/Child**

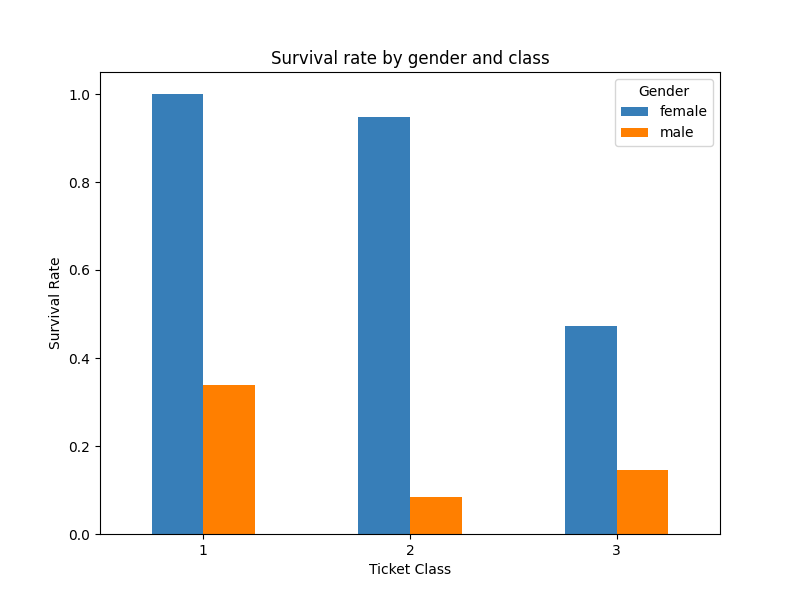
**Hypothesis:** Passengers with parent/child relationships onboard have higher survival rates, possibly due to prioritization during evacuations.

**Approach:** Analyzed survival rates for passengers based on the number of parent/child relationships, comparing those with no parent/child on board to those with one or more.

**Results:** Passengers with no parent/child had a survival rate of 33%, while those with one or more family members had survival rates ranging from 48% to 55%. This suggests that having a parent or child on board might positively impact survival, possibly due to prioritization during evacuation efforts.

#### 2.1.3 Multivariate Analysis

##### 2.1.3.1 Gender and Ticket Class



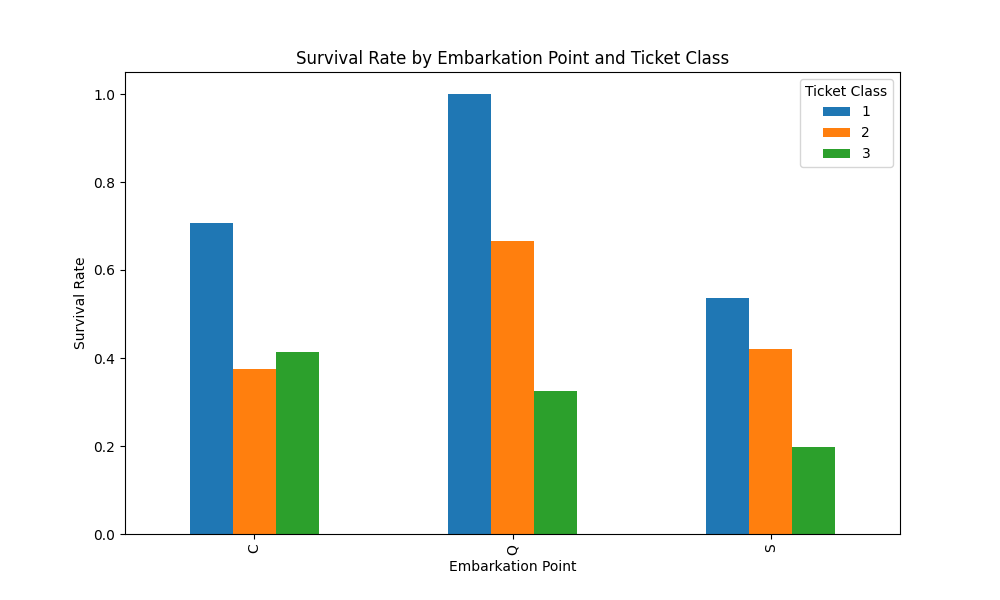
**Figure 1: Survival Rate by Gender and Class**

**Hypothesis:** The higher survival rate in Class 1 is influenced by the ticket class itself rather than simply by the gender distribution within the class.

**Approach:** Analyzed survival rates within each class for both male and female passengers to understand the effect of ticket class on survival, independent of gender. Specifically examined whether first-class had a higher survival rate primarily due to a greater number of female passengers or if ticket class had its own influence. Also compared survival rates for males in each class to check for any unexpected trends, such as third-class males having higher survival rates than second-class males.

**Results:** Both male and female passengers showed increased survival rates if they held first-class tickets, supporting the hypothesis that ticket class itself is an influential factor in survival, not just the gender distribution. Female first-class passengers had a 100% survival rate, while first-class males had a survival rate of 33.9%. Comparatively, second-class females had a survival rate of 94.9%, and third-class females had 47.2%, further highlighting the role of ticket class. Among males, third-class passengers had a higher survival rate (14.6%) than second-class males (8.3%), which might suggest that family presence could have affected second-class males’ survival rates. Ultimately, while gender heavily impacts survival rates, ticket class independently also plays a significant role in survival likelihood, particularly for females in first class.

##### 2.1.3.2 Embarkation Point and Ticket Class

****

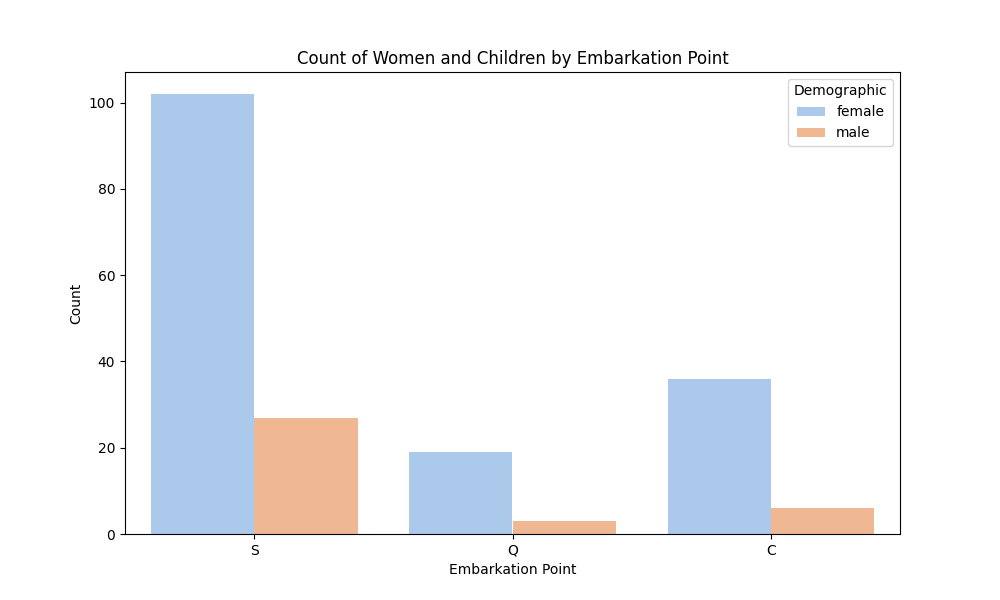
**Figure 1: Survival Rate by Embarkation Point and Ticket Class**

**Hypothesis:** First-class passengers have a higher survival rate across all embarkation points, with the possibility that "C" has a higher survival rate due to a higher number of women and children.

**Approach:** Analyze survival rates by ticket class and embarkation point, comparing the distribution of first-class passengers and the demographics (gender and age) of passengers embarking from "C," "Q," and "S." Calculated and compared the total number of women and children across embarkation points to assess their potential impact on survival rates.

**Results:** First-class passengers indeed had the highest survival rates across all embarkation points, with "Q" showing the highest survival rate for first-class passengers. Despite this, passengers embarking from "C" had a higher overall survival rate than those from "Q" and "S." This may be explained by the demographic breakdown: "C" had a higher proportion of women and children (42 total) compared to "Q" (22 total) and "S" (129 total). Given that women and children were prioritized for survival during evacuation, their higher numbers in "C" could contribute to the higher survival rate observed. This finding suggests that, while ticket class affects survival, gender and age distribution also play a significant role, particularly in explaining why "C" has a higher survival rate than "Q" and "S" overall.

##### 2.1.3.3 Embarkation Point and Count of Women



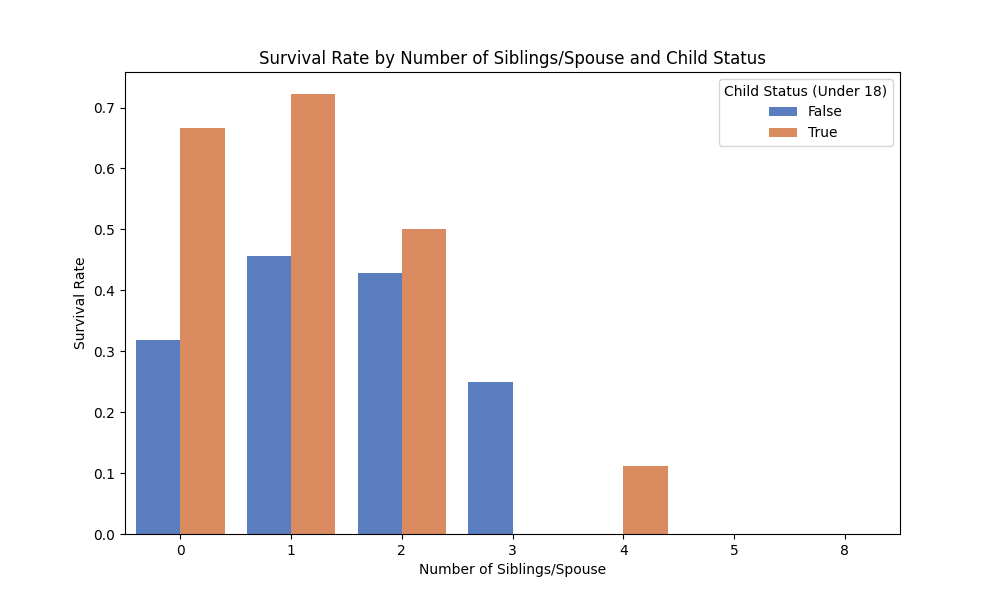
**Figure 1: Count of Women and Children by Embarkation Point**

**Hypothesis:** The higher survival rate at embarkation point "C" compared to "Q" is likely due to a higher number of women and children from "C," as women and children were prioritized for survival during evacuation.

**Approach:** Investigated the survival rates by embarkation point and analyzed the demographic distribution, particularly focusing on the number of women and children at each embarkation point. Compared the survival rates of passengers from "C" and "Q" to determine if the presence of more women and children at "C" could explain the higher survival rate.

**Results:** The analysis shows that "C" has a higher survival rate than "Q," which aligns with the finding that more women and children embarked from "C" (42 in total) compared to "Q" (22 in total). Since women and children were more likely to be prioritized during evacuation, this demographic difference likely contributed to the increased survival rate for passengers from "C." Therefore, while embarkation point alone is not a significant feature for model training, the underlying demographic characteristics (such as gender and age) associated with each embarkation point have a notable impact on survival. This suggests that factors like gender and age are more predictive of survival outcomes than embarkation point alone.

##### 2.1.3.4 Number of Siblings/Spouse and Child Status



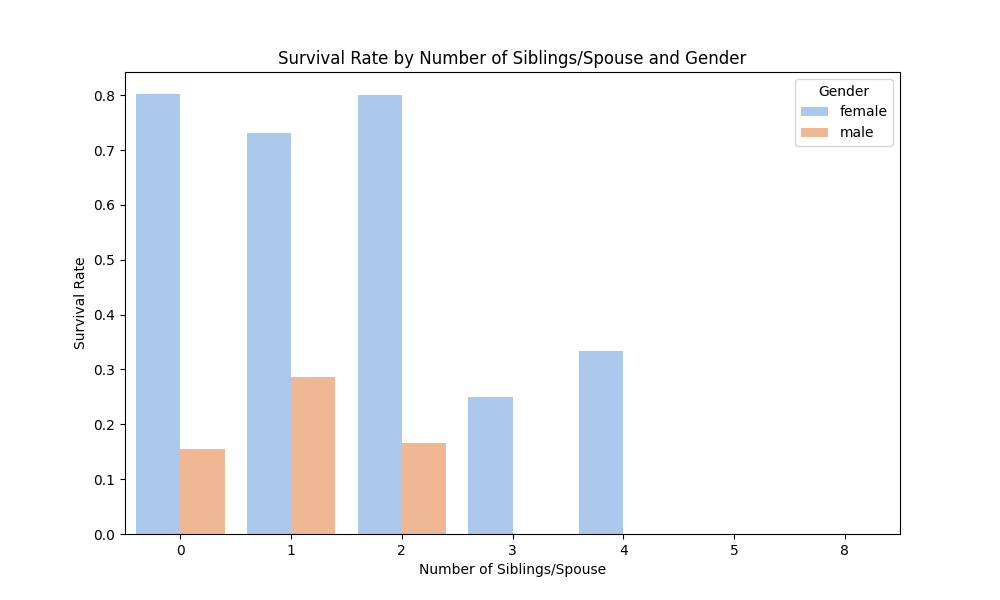
**Figure 1: Survival Rate by Number of Siblings/Spouse and Child Status**

**Hypothesis:** Younger passengers with siblings/spouses may have higher survival rates, as children were more likely to be prioritized during evacuation.

**Approach:** Analyzed survival rates by age group within each sibling/spouse category, focusing on whether the presence of siblings/spouses influenced survival differently for children versus adults.

**Results:** Survival rates were generally higher for children across all sibling/spouse categories, particularly for those with 1 or 2 family members. In contrast, adult survival rates were lower, especially as the number of siblings/spouses increased. This pattern aligns with the idea that children were prioritized during evacuation, and their survival advantage was more pronounced in smaller family units.

##### 2.1.3.5 Number of Siblings/Spouse and Gender



**Figure 1: Survival Rate by Number of Siblings/Spouse and Gender**

**Hypothesis:** Females with siblings/spouses have higher survival rates than males, possibly due to prioritization during rescue efforts.

**Approach:** Compared survival rates between male and female passengers within each sibling/spouse category to assess if gender impacted survival likelihood.

**Results:** Females consistently exhibited higher survival rates across all sibling/spouse numbers, particularly those with 1 or 2 family members. In contrast, males showed much lower survival rates. This suggests that females, especially those with small family units, had a survival advantage, likely due to prioritization during rescue.

##### 2.1.3.6 Number of Siblings/Spouse and Ticket Class



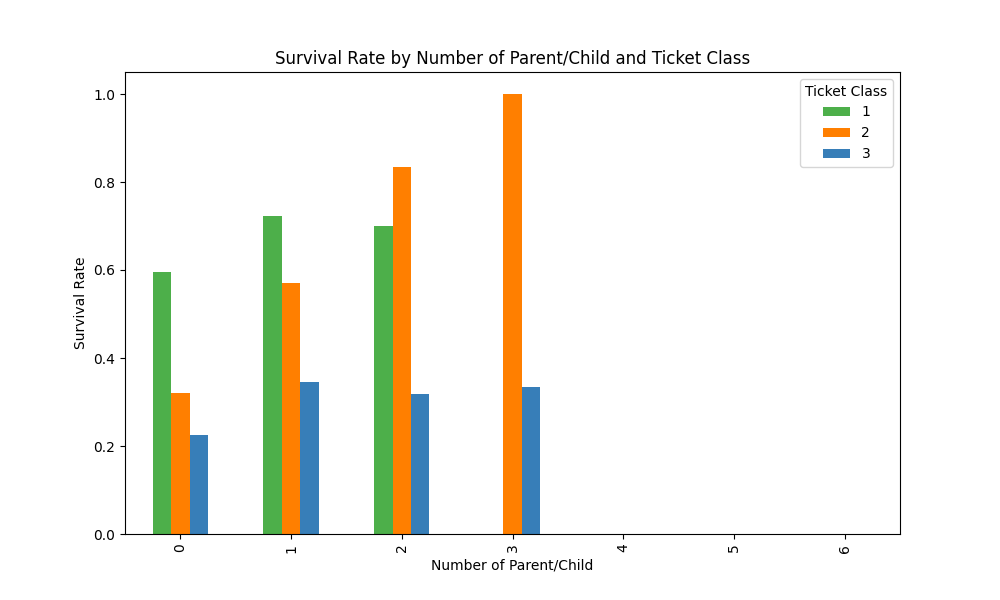
**Figure 1: Survival Rate by Number of Siblings/Spouse and Ticket Class**

**Hypothesis:** First-class passengers with siblings/spouses have higher survival rates across all sibling/spouse categories.

**Approach:** Investigated survival rates by ticket class within each sibling/spouse count to evaluate whether the combination of class and family presence influences survival.

**Results:** First-class passengers consistently had higher survival rates across all sibling/spouse numbers, with survival particularly high for those with 1 or 2 family members. In contrast, survival rates decreased sharply for second and third-class passengers as the number of siblings/spouses increased, with survival vanishing for larger family groups in these classes. This pattern suggests that the combination of first-class status with a small family unit may offer the best survival odds. The sibling/spouse count appears to be a relevant feature for model training, especially when combined with class, gender, and age, which collectively impact survival rates.

##### 2.1.3.7 Number of Parent/Child and Ticket Class



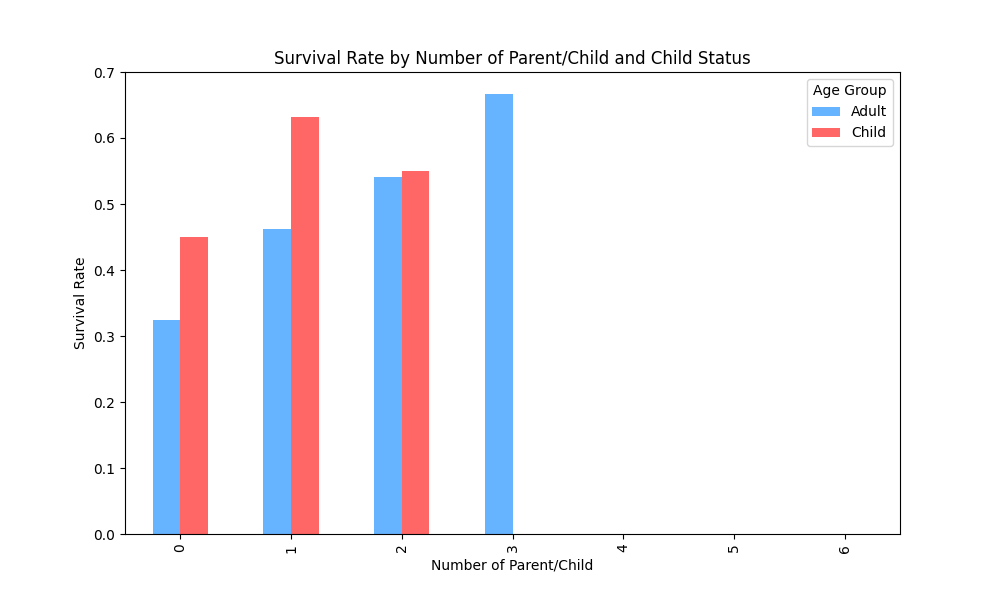
**Figure 1: Survival Rate by Number of Parent/Child and Ticket Class**

**Hypothesis:** First-class passengers with parent/child on board have higher survival rates than those in lower classes.

**Approach:** Examined survival rates by ticket class within each parent/child count to assess whether both class and family relationships impact survival.

**Results:** First-class passengers showed a high survival rate, ranging from 60% to 70%, across different parent/child counts. In second class, survival rates increased as the number of parent/child relationships increased, while third-class passengers had markedly lower survival rates, though they also showed slight increases with family presence. This indicates that both ticket class and family relationships affect survival outcomes.

##### 2.1.3.8 Number of Parent/Child and Child Status



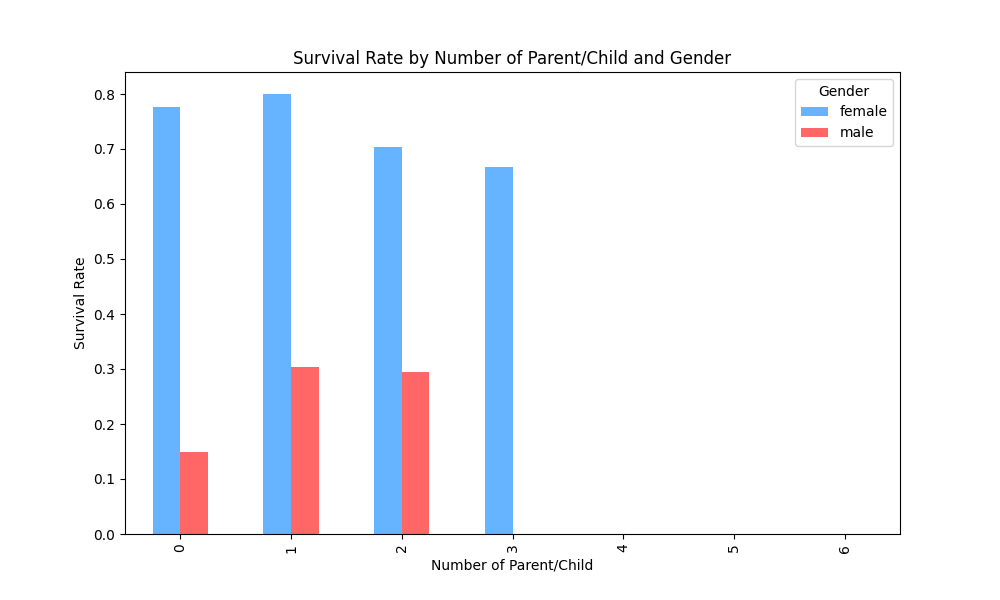
**Figure 1: Survival Rate by Number of Parent/Child and Child Status**

**Hypothesis:** First-class passengers with parent/child on board have higher survival rates than those in lower classes.

**Approach:** Examined survival rates by ticket class within each parent/child count to assess whether both class and family relationships impact survival.

**Results:** First-class passengers showed a high survival rate, ranging from 60% to 70%, across different parent/child counts. In second class, survival rates increased as the number of parent/child relationships increased, while third-class passengers had markedly lower survival rates, though they also showed slight increases with family presence. This indicates that both ticket class and family relationships affect survival outcomes.

##### 2.1.3.9 Number of Parent/Child and Gender



**Figure 1: Survival Rate by Number of Parent/Child and Gender**

**Hypothesis:** Females with parent/child on board have higher survival rates than males, possibly due to evacuation prioritization.

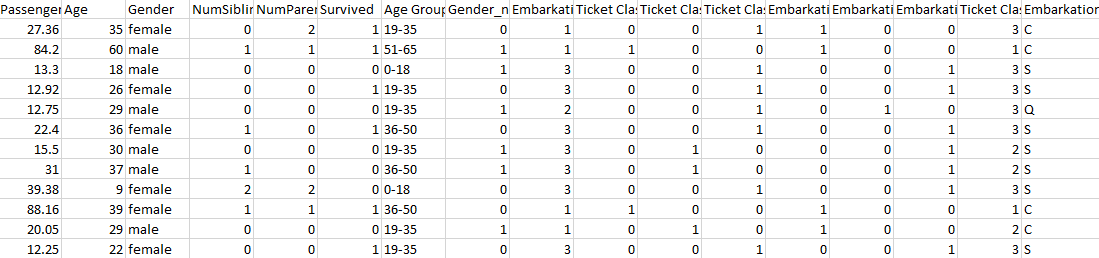
**Approach:** Compared survival rates for male and female passengers across different parent/child counts to determine if gender impacted survival.

**Results:** Females consistently showed higher survival rates across all parent/child counts, while males exhibited lower survival rates overall, though they showed slight increases when they had parent/child with them. This supports the hypothesis that having a parent/child on board is associated with higher survival chances, especially for women, children, and passengers in higher ticket classes.

### 2.2 Processing Activities



**Figure 1: Uncleaned Data**



**Figure 1: Cleaned Data**

The above figures show the before and after the processing activities of the data. Below we will provide further explanation on how the processing activities are done.

#### 2.2.1 Feature Selection

data = data.drop(['Passenger ID', 'Ticket Number', 'Cabin', 'Name'], axis=1, errors='ignore')

The Name, Ticket Number, Passenger ID, Cabin Location columns can be removed from the dataset for the following reasons.

##### 2.2.1.1 Name

**Irrelevance to Analysis:** The name of the passenger does not provide useful information for predictive modeling or analysis related to survival. It does not contribute to understanding the patterns or factors influencing survival rates in the context of the dataset.

**Unique Identifiers:** The Name column often contains unique values for each passenger, which means it does not carry any aggregated information that could aid in the analysis. For instance, analyzing names doesn’t yield insights into trends or group characteristics.

**Potential for Noise:** Including names can introduce noise into the analysis. Names can be highly variable and subjective, leading to complications in modeling. Moreover, they may not correlate with relevant features like socio-economic status, survival likelihood, or demographics.

##### 2.2.1.2 Ticket Number

**Lack of Predictive Power:** The Ticket Number may not provide meaningful insights into passenger survival. While it may have some categorical variations, these variations do not inherently offer information on survival probabilities or related metrics.

**Redundancy:** Ticket numbers might be unique identifiers without relevance to the analysis of the passenger's characteristics or survival chances. Since each ticket is associated with a unique passenger, its removal does not lead to the loss of critical information.

**Complexity:** The ticket number may have embedded information (e.g., class or route) that isn’t straightforward to extract. If such relevant data is needed, it might be better to derive that information from other columns, like Ticket Class, rather than maintaining the Ticket Number.

##### 2.2.1.3 Passenger ID

**Lack of Predictive Power:** The Passenger ID is a unique identifier for each passenger, but it does not provide any relevant information about the characteristics or behaviors associated with survival. Since each Passenger ID is unique, it lacks any patterns or groupings that could aid in analysis or prediction.

**Redundancy:** The Passenger ID serves primarily as a reference label rather than a feature with analytical value. Removing this column does not lead to a loss of valuable information, as it is only used to identify individual rows rather than to inform survival or other passenger characteristics.

**Complexity:** Since Passenger ID values are distinct for each entry, they introduce unnecessary complexity if retained in the dataset. Including this column could lead to overfitting in models without contributing meaningful insights. Other columns, like Name or Ticket Class, can capture demographic or categorical information more effectively if needed.

##### 2.2.1.4 Cabin Location

**Lack of Predictive Power:** While the Cabin Location column might contain details about passenger accommodations, it often lacks strong correlation with survival outcomes due to the varied and complex cabin assignments. Cabin codes do not have an intuitive or direct relationship with survival rates.

**Redundancy:** Cabin information could potentially be inferred from other variables, such as Ticket Class, which broadly captures different accommodation levels (e.g., first, second, or third class). Thus, Cabin Location becomes redundant as Ticket Class provides a more consistent, categorical representation of passenger status and accommodations.

**Complexity:** Cabin Location often contains alphanumeric codes that might be hard to interpret or categorize meaningfully. Since cabins are sometimes inconsistently labeled and sparsely populated in datasets, extracting relevant features (like proximity to exits) would require significant processing without a guaranteed predictive benefit. In many datasets, cabin information is also missing for many passengers, making it less reliable for analysis.

#### 2.2.2 Handling Missing and Erroneous Values

The data cleaning function implemented a series of steps to manage missing and erroneous values, ensuring data consistency and usability for modeling.

##### 2.2.2.1 Missing Values in Age

import pandas as pd

import numpy as np

# Assuming df is your DataFrame and "Title" column contains titles like "Mr", "Miss", "Mrs", etc.

# Step 1: Replace any ages recorded as zero with NaN

df['Age'].replace(0, np.nan, inplace=True)

# Step 2: Fill missing values in Age based on median age for each title

df['Age'] = df.groupby('Title')['Age'].transform(lambda x: x.fillna(x.median()))

To handle missing values in the Age column, the code uses median age values grouped by the Title column (e.g., “Mr”, “Miss”, “Mrs”). This approach provides a more contextually accurate imputation based on observed patterns in passenger age and title. Additionally, ages that were recorded as zero were replaced with NaN, then filled using the median age values for each title. This replacement helps ensure that no age is recorded as zero, which could skew analysis.

##### 2.2.2.2 Passenger Fare Cleaning

from sklearn.impute import SimpleImputer

# Step 1: Remove dollar signs and commas, convert to numeric

df['Passenger Fare'] = df['Passenger Fare'].replace('[\$,]', '', regex=True).astype(float)

# Step 2: Impute missing values with mean

imputer = SimpleImputer(strategy='mean')

df['Passenger Fare'] = imputer.fit\_transform(df[['Passenger Fare']])

The Passenger Fare column often contained dollar signs and commas, which were removed to allow conversion to a numerical format. After cleaning, missing values in the Passenger Fare column were filled with the mean fare using the SimpleImputer function. This method provides a straightforward way to address missing values without introducing significant bias.

##### 2.2.2.3 Embarkation Country Codes

# Replace ambiguous values ("0" or empty) with the mode

mode\_embarkation = df['Embarkation Country'].mode()[0]

df['Embarkation Country'].replace(["0", "", np.nan], mode\_embarkation, inplace=True)

Any ambiguous or missing codes in the Embarkation Country column (represented as "0" or empty values) were filled using the mode of the column, ensuring that no invalid values remain. This mode replacement was applied consistently to provide a reasonable assumption of embarkation location when data was missing.

#### 2.2.3 Data Transformation (Normalization and Scaling)

Data transformations were applied selectively to prepare features for analysis.

##### 2.2.3.1 Passenger Fare Scaling

df['Passenger Fare'] = df['Passenger Fare'].round(2)

Although the code does not currently scale Passenger Fare directly, it rounds the fare values to two decimal places, standardizing the format. Additional scaling (e.g., using StandardScaler or MinMaxScaler) could be introduced if necessary for models sensitive to value ranges.

##### 2.2.3.2 Age Grouping

# Define age bins and labels

age\_bins = [0, 18, 35, 50, 65, np.inf]

age\_labels = ['0-18', '19-35', '36-50', '51-65', '65+']

# Apply age grouping

df['Age Group'] = pd.cut(df['Age'], bins=age\_bins, labels=age\_labels)

The Age column was transformed into categorical age groups (e.g., 0-18, 19-35, etc.), creating the new Age Group feature. This transformation helps to capture age-related patterns by grouping passengers into meaningful age ranges rather than using raw age values, which may vary widely.

If additional scaling or normalization were needed (such as for heavily skewed features), MinMaxScaler or StandardScaler from sklearn could be added to the pipeline.

#### 2.2.4 Encoding Categorical Variables

To make categorical data suitable for machine learning algorithms, the code applied the following transformations:

##### 2.2.4.1 Gender Encoding

# Map gender to binary values

df['Gender'] = df['Gender'].map({'male': 1, 'female': 0})

The Gender column was encoded into binary values, with male represented as 1 and female as 0. This encoding allows models to process the Gender data as a numerical input without adding additional complexity.

##### 2.2.4.2 Embarkation Country Encoding

# Map Embarkation Country to numeric values

embark\_map = {'C': 1, 'Q': 2, 'S': 3}

df['Embarkation Country'] = df['Embarkation Country'].map(embark\_map)

The Embarkation Country column was mapped to numeric values (C=1, Q=2, S=3), facilitating analysis by converting embarkation locations into integer format. This approach helps to retain meaningful categorical distinctions without one-hot encoding.

##### 2.2.4.3 One-Hot Encoding for Ticket Class and Embarkation Country

from sklearn.preprocessing import OneHotEncoder

# Apply one-hot encoding

df = pd.get\_dummies(df, columns=['Ticket Class', 'Embarkation Country'], prefix=['Class', 'Embark'])

Additional one-hot encoding was performed on Ticket Class and Embarkation Country using OneHotEncoder, creating new binary columns for each category. This encoding provides a full representation of these categorical features without implying any ordinal relationship among the classes.

##### 2.2.4.4 Conclusion

These transformations prepare the dataset for model training by converting all categorical values into numerical formats that machine learning models can interpret effectively. This approach also retains useful information from the categorical variables while minimizing redundancy.

#### 2.2.5 Feature Engineering (Creating New Features)

The data cleaning and preprocessing steps include several feature engineering techniques designed to capture additional relationships and provide meaningful insights for the analysis. Here are the engineered features and their rationale.

##### 2.2.5.1 Title Extraction from Name

# Extract title from the Name column and create a new column 'Title'

if 'Name' in data.columns:

data['Title'] = data['Name'].str.extract(r',\s\*([^\.]\*)\s\*\.', expand=False)

A new feature, Title, was created by extracting titles (e.g., “Mr”, “Miss”, “Mrs”) from the Name column. This feature helps capture social and demographic information, such as marital status, gender, and social position, which may correlate with survival rates. For example, certain titles might indicate family responsibilities (e.g., "Mrs" often implies family presence), which could impact survival likelihood.

##### 2.2.5.2 Age Group Creation

# Group Age into categorical bins after filling missing values

data['Age Group'] = pd.cut(data['Age'], bins=[0, 18, 35, 50, 65, 100], labels=['0-18', '19-35', '36-50', '51-65', '66+'])

To capture age-related survival patterns more effectively, Age values were categorized into discrete age groups (0-18, 19-35, 36-50, 51-65, 66+) to create the Age Group feature. This grouping allows for more nuanced analysis by treating age ranges as categories rather than individual values, which can highlight differences in survival likelihood across different age groups.

##### 2.2.5.3 Numerical Encoding for Gender

# Convert Gender to a numerical value: male=1, female=0

data['Gender\_num'] = data['Gender'].map({'male': 1, 'female': 0})

The Gender\_num feature was introduced as a binary encoding of the Gender column (1 for male and 0 for female). This transformation simplifies analysis by converting gender into a numeric format, which may directly influence model training and survival predictions, as gender has been observed to correlate with survival rates.

##### 2.2.5.4 Embarkation Country Encoding

# Convert Embarkation Country to numerical values for model training

embarkation\_map = {'C': 1, 'Q': 2, 'S': 3}

data['Embarkation\_Country\_num'] = data['Embarkation Country'].map(embarkation\_map)

The feature Embarkation\_Country\_num was added by converting categorical values in the Embarkation Country column to numeric values (C=1, Q=2, S=3). This numeric encoding enables the model to interpret embarkation points without creating excessive binary columns. Since embarkation points might correlate with socioeconomic status, this feature could potentially reveal insights about passengers’ survival outcomes.

##### 2.2.5.5 Conclusion

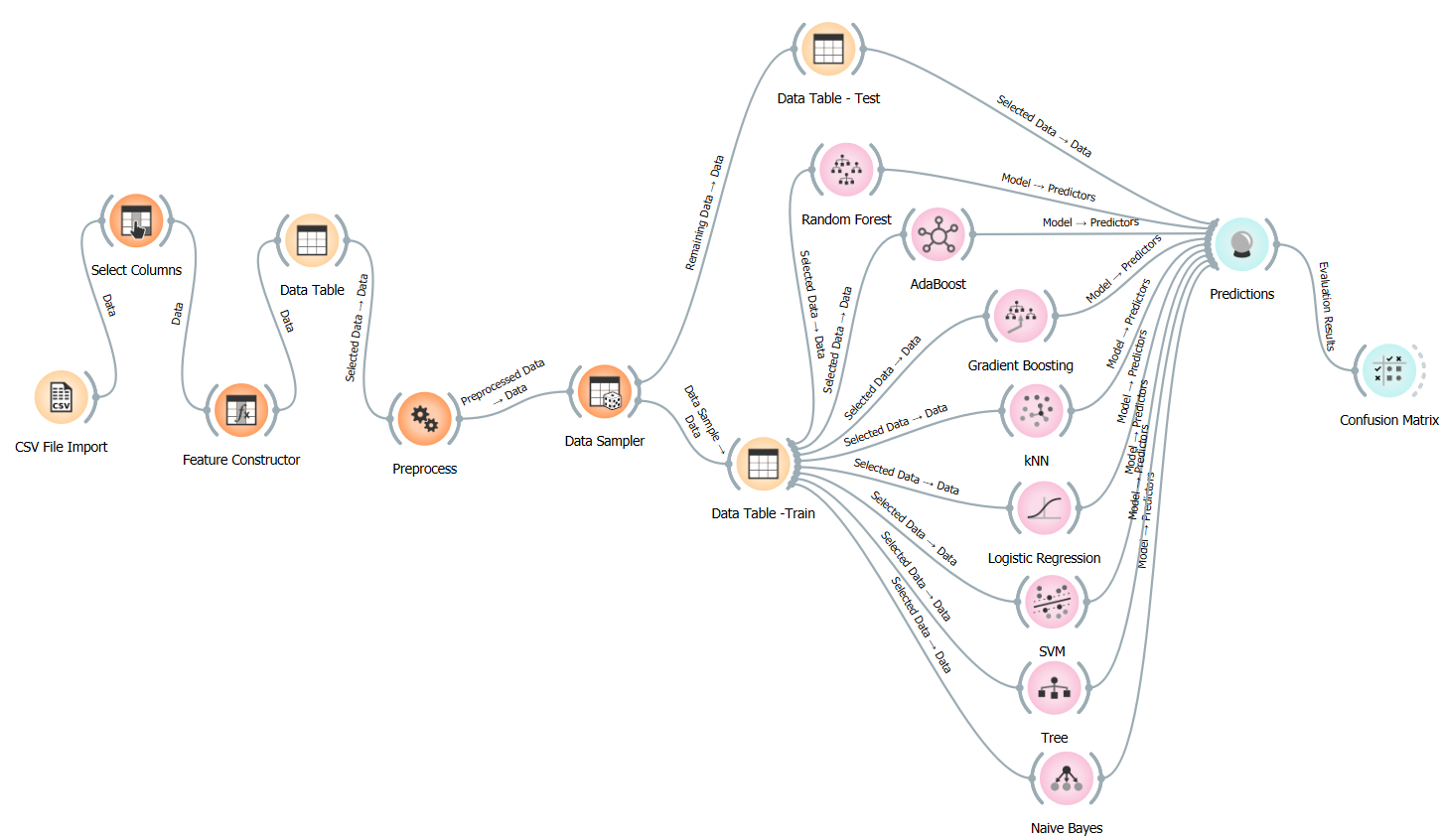
These engineered features enhance the dataset by adding contextual information relevant to passengers' demographic and social profiles, which could have meaningful impacts on survival. The transformations applied here are designed to improve the dataset’s representational power without introducing redundancy, aiding in more accurate model training and analysis.

### 2.3 Orange Pipeline

In this project, we utilized Orange, an open-source data visualization and analysis tool, to establish a baseline for model performance and experiment with various machine learning algorithms. Orange’s interactive interface allowed us to quickly test different models and compare their metrics in a visual, drag-and-drop environment. This initial baseline provided insights into model performance across different metrics such as accuracy, precision, recall, and AUC scores, enabling a straightforward assessment of each model’s strengths and weaknesses.

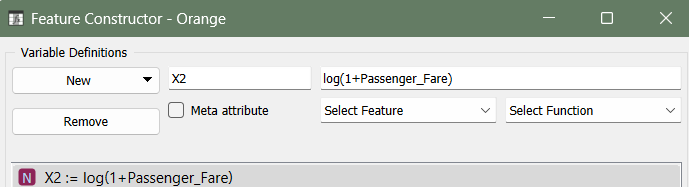
Using Orange helped streamline the early experimentation phase, allowing us to explore models like Decision Trees, Naive Bayes, Random Forests, and Support Vector Machines without extensive coding. Once we had a clear understanding of which models performed best, we moved to a code-based pipeline for deeper customization and tuning. This pipeline allowed for greater control over hyperparameters, facilitated advanced evaluations, and integrated with our CLI and GUI interfaces for a comprehensive, flexible workflow.

This dual approach—starting with Orange and refining in code—enabled us to efficiently experiment with model configurations, ensuring a robust selection process before committing to a final model for deployment.



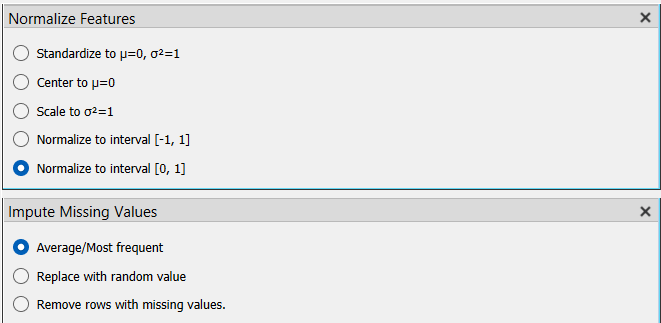
**Fig 1 : Orange pipeline structure**

In this Orange-based machine learning workflow, the classification process starts with data import and preparation, where categorical variables such as gender, embarkation country, and survival outcome are converted to numeric formats to improve model compatibility. Key features are then selected, and data preprocessing ensures the dataset is clean and consistent. Afterward, the data is split into training and test sets to enable reliable model evaluation. Various classification algorithms—including Random Forest, AdaBoost, Gradient Boosting, k-Nearest Neighbors, Logistic Regression, Support Vector Machine, Decision Tree, and Naive Bayes—are trained on the training set, and each model generates predictions on the test set. Finally, a confusion matrix is used to evaluate the models, providing insights into accuracy and classification performance. This systematic workflow supports comprehensive model assessment and facilitates the identification of the most effective classifier.

****

**Fig 2 : Log transformation of Passenger Fare**

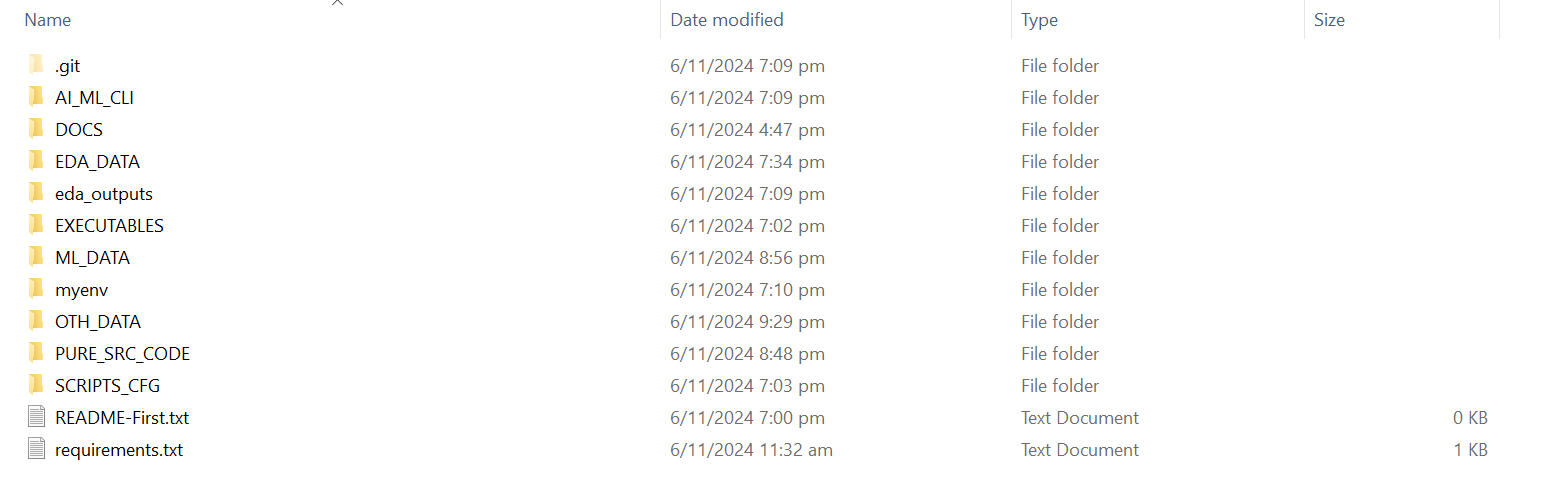
The selected feature undergoes log transformation to reduce skewness in the data, particularly when the feature has a wide range or is right-skewed.

****

**Fig 3 : Preprocessing data**

Data preprocessing prepares the dataset by ensuring it is clean, standardized, and ready for effective use in machine learning models. The selected normalization method scales feature values to the range [0, 1], ensuring consistency across features with different scales, which can enhance model performance. For handling missing values, the chosen approach fills gaps with either the mean or the most frequent value, maintaining data integrity without discarding rows or introducing random values.

### 2.4 Code pipeline



### 2.5 CLI**Directory Structure Overview**

* **PURE\_SRC\_CODE**: Contains all the core Python scripts (clean\_data.py, perform\_eda.py, model\_training.py, perform\_prediction.py) and the main control script pipeline.sh that orchestrates the execution of the executables. Running pipeline.sh initiates the pipeline process, allowing users to choose tasks like data cleaning, EDA, model training, and prediction.
* **EXECUTABLES**: Holds the .exe versions of the main Python scripts (clean\_data.exe, perform\_eda.exe, model\_training.exe, perform\_prediction.exe). These executables are called by pipeline.sh in PURE\_SRC\_CODE, enabling users to run the pipeline without directly executing Python scripts.
* **SCRIPTS\_CFG**: Contains the config.txt file, which is used to configure paths, parameters, and settings for each stage of the pipeline. This file is referenced by each executable to ensure they use the correct inputs and outputs.
* **OTH\_DATA**: Contains raw and intermediate data folders:
  + **training\_data**: Stores the initial dataset for training models.
  + **testing\_data**: Holds datasets used for model evaluation and testing.
  + **cleaned\_data**: Contains preprocessed data after cleaning, ready for analysis or modeling.
* **EDA\_DATA**: Stores figures and data generated from the Exploratory Data Analysis (EDA) stage. This folder includes plots and other visuals that help in understanding data patterns and distributions.
* **ML\_DATA**: Structured to hold model outputs and prediction results:
  + **model\_outputs**: Contains saved trained models (e.g., .pkl files) generated by the model training stage.
  + **predict\_outputs**: Holds prediction result files, such as prediction\_report.csv, which summarize model performance on test data.
  + **visualize\_output**: Stores prediction-related figures and graphs, such as confusion matrices and ROC curves, to provide insights into model performance.'

**Pipeline Flow Description**

1. **Initiation**: The pipeline is initiated by running pipeline.sh in the PURE\_SRC\_CODE folder. This script presents a menu for selecting different stages of the pipeline.
2. **Data Cleaning** (clean\_data.exe): The pipeline first accesses clean\_data.exe in the EXECUTABLES folder. This executable reads raw data from OTH\_DATA/training\_data, processes it (e.g., handling missing values, encoding categorical variables), and outputs cleaned data to OTH\_DATA/cleaned\_data.
3. **Exploratory Data Analysis (EDA)** (perform\_eda.exe): The perform\_eda.exe executable analyzes the cleaned data, generating figures and summaries that are stored in EDA\_DATA. These visuals provide insights into data distribution, trends, and outliers.
4. **Model Training** (model\_training.exe): The pipeline then moves to training models using model\_training.exe. This executable reads data from OTH\_DATA/training\_data (or OTH\_DATA/cleaned\_data if preprocessed data is required), trains various machine learning models, and saves the trained models to ML\_DATA/model\_outputs.
5. **Prediction and Evaluation** (perform\_prediction.exe): Finally, perform\_prediction.exe is executed to generate predictions using the trained models. The results are stored in ML\_DATA/predict\_outputs as files like prediction\_report.csv, while graphs and figures related to prediction (e.g., ROC curves, confusion matrices) are saved to ML\_DATA/visualize\_output.

**Execution Notes**

* **Config File**: config.txt in the SCRIPTS\_CFG folder provides configuration settings for paths, parameters, and other options required by each stage of the pipeline. Each executable reads this file to determine where to find inputs and where to save outputs.

### **Running the Pipeline**: To execute the pipeline, navigate to the PURE\_SRC\_CODE directory and run pipeline.sh. This script will use the executables in the EXECUTABLES folder, configured by config.txt, to perform each pipeline stage sequentially.

The CLI (Command-Line Interface) pipeline menu is a command-line tool designed to streamline the process of data preparation, analysis, model training, and prediction. The menu is organized into distinct options, each corresponding to a different stage in the pipeline, allowing users to execute specific tasks with ease. The CLI menu guides the user through the process of cleaning data, conducting exploratory data analysis (EDA), training machine learning models, and making predictions, with each stage customized based on user-specified parameters.

Pipeline Menu Options

When the user runs the pipeline.sh script located in the PURE\_SRC\_CODE directory, a CLI menu is displayed. The user can select from several options by entering the corresponding letter or number:

* c. Clean data
* e. Perform EDA on training data
* m. Train model
* p. Predict model
* 4. Exit

Each option triggers a separate executable in the EXECUTABLES directory, allowing users to control the flow of tasks without needing direct interaction with individual scripts. Below, we detail each option and its functionality, as well as how users can configure parameters through a configuration file.

**In-depth description of each option**

**Data Cleaning (Option 'c')**

* **Function**: Initiates the data cleaning process using clean\_data.exe.
* **Process**: When this option is chosen, the executable reads raw data from the OTH\_DATA/training\_data folder, performs cleaning operations (such as handling missing values, encoding categorical features, and dropping unnecessary columns), and saves the cleaned dataset to OTH\_DATA/cleaned\_data.
* **Configuration**: Upon selecting this option, the config.txt file is opened in the default text editor. Users can specify:
  + input\_folder\_clean: The folder where raw data is located.
  + output\_folder\_clean: The folder where cleaned data should be saved.
  + cleaned\_file\_suffix: A suffix to be appended to the cleaned file's name.
* **User Interaction**: After reviewing or updating the configuration file, the user presses Enter to proceed with data cleaning based on the specified paths and settings.

**Exploratory Data Analysis (EDA) (Option 'e')**

* **Function**: Executes EDA on the cleaned data using perform\_eda.exe.
* **Process**: The executable reads data from OTH\_DATA/cleaned\_data, performs EDA tasks (such as generating histograms, box plots, and summary statistics), and saves the results to the EDA\_DATA folder. The generated visuals and reports help in understanding data distributions, relationships, and potential outliers.
* **Configuration**: Upon selecting this option, config.txt opens, allowing the user to specify:
  + input\_folder\_eda: The folder containing the data for EDA.
  + output\_folder\_eda: The folder where EDA outputs should be saved.
  + eda\_file\_suffix: A suffix to be added to each EDA output file.
* **User Interaction**: The user reviews the configuration file, updates it if necessary, and presses Enter to continue. EDA is then conducted based on the configured paths and settings.

**Model Training (Option 'm')**

* **Function**: Launches the model training process with model\_training.exe.
* **Process**: The executable reads training data from OTH\_DATA/cleaned\_data or OTH\_DATA/training\_data, depending on the configuration. It then trains various machine learning models (e.g., Logistic Regression, Decision Tree, Random Forest) with specified hyperparameters and saves the trained models to ML\_DATA/model\_outputs.
* **Configuration**: When this option is selected, config.txt opens, allowing the user to input:
  + input\_folder: The path to the folder containing the data for training.
  + output\_folder: The path to the folder where trained models should be saved.
  + model\_name\_suffix: A suffix to append to each model’s file name (e.g., \_v1 for versioning).
  + Model parameters, such as the type of regularization for Logistic Regression, max depth for Decision Tree, number of estimators for Random Forest, etc.
* **User Interaction**: The user can modify the configuration file to specify or update model parameters and paths before pressing Enter to proceed. This flexibility allows users to fine-tune model performance and save models with specific configurations.

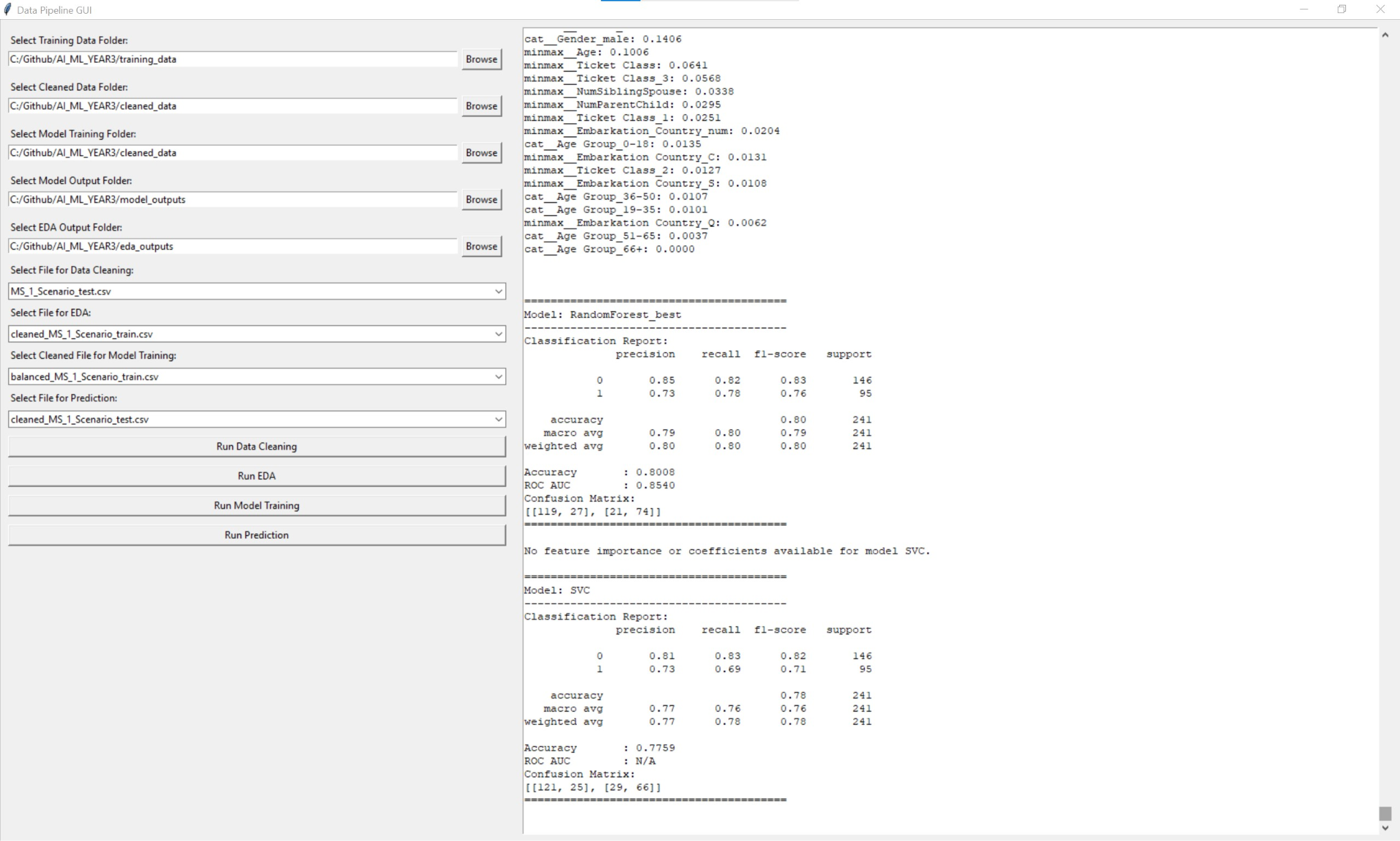
**Model Prediction (Option 'p')**

* **Function**: Initiates the prediction process with perform\_prediction.exe.
* **Process**: The executable reads test data from OTH\_DATA/testing\_data and loads models from ML\_DATA/model\_outputs to generate predictions. It saves prediction results (e.g., classification reports, confusion matrices) to ML\_DATA/predict\_outputs and visualizations (e.g., ROC curves) to ML\_DATA/visualize\_output.
* **Configuration**: On selecting this option, config.txt opens, allowing the user to specify:
  + input\_folder: The folder where the test data is located.
  + model\_folder: The folder containing trained models to be used for predictions.
  + output\_folder: The folder where prediction results and visualizations should be saved.
* **User Interaction**: The user reviews the configuration file to ensure the paths and settings align with the desired test setup, then presses Enter to proceed. The script then generates and saves the prediction outputs and evaluation metrics.

**Exit (Option '4')**

* **Function**: Exits the CLI menu and terminates the pipeline process.

### 2.6 GUI



The GUI provides a user-friendly interface for managing the data pipeline, making it more accessible for users who prefer not to work directly with command-line interfaces (CLI). Here's an overview of its structure and functionality:

1. Folder Selection:
   * On the left side, users can select various directories by browsing through the file system. These folders include:
     + **Training Data Folder**: Where raw training data is stored.
     + **Cleaned Data Folder**: The directory for processed, clean data ready for analysis.
     + **Model Training Folder**: Where the model will be trained using the cleaned data.
     + **Model Output Folder**: Location to save the trained model outputs.
     + **EDA Output Folder**: Stores results from Exploratory Data Analysis (EDA) like feature importance and data insights.
2. File Selection and Processing:
   * Users can select specific files for different stages of the pipeline:
     + **Data Cleaning File**: The file that needs to be cleaned.
     + **EDA File**: The file used for conducting EDA.
     + **Training File**: The cleaned dataset for model training.
     + **Prediction File**: The dataset for running final predictions.
3. Run Pipeline Buttons:
   * The GUI provides several buttons that allow users to run various stages of the pipeline with a single click:
     + **Run Data Cleaning**: Processes and cleans the selected dataset.
     + **Run EDA**: Conducts EDA on the selected file, extracting insights like feature importance.
     + **Run Model Training**: Trains the model on the specified training file using selected parameters.
     + **Run Prediction**: Runs predictions on the test data using the trained model.
4. Result Display Panel:
   * The right side of the GUI displays the results and outputs for each process. It includes:
     + **Feature Importance**: The list of features ranked by importance from EDA, helping users understand which attributes are most influential.
     + **Classification Report**: The model's performance metrics (precision, recall, F1-score, accuracy) on both training and testing data.
     + **Confusion Matrix**: A visual summary of correct and incorrect predictions, giving insight into model performance on each class.
     + **Additional Metrics**: Metrics like ROC AUC, Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) for comprehensive performance evaluation.

The GUI enables users to navigate through the pipeline efficiently, select relevant files and folders, and view results in a clear, organized format. This setup supports easy access to complex data processing tasks and model evaluations, enhancing accessibility for users who prefer a visual interface over CLI commands.

## 3. Model Training

A wide range of supervised machine learning algorithms were employed to train and evaluate models on a labeled dataset. Each algorithm belongs to a distinct category of machine learning techniques, enabling the exploration of various model types to determine the most effective one for the task at hand. Below is an overview of each model, including the algorithm, training approach, and key hyperparameters.

To ensure robust model performance and to prevent overfitting or underfitting, a train-test-validate split strategy was implemented. The dataset was divided as follows:

- Training Set: This portion of the data was used for training, enabling the model to learn patterns and relationships within the dataset.

- Testing Set: After training, the model’s performance was evaluated on this separate set to assess its generalization capability on unseen data.

This train-test split approach provides an unbiased estimate of the model’s performance on new data, validating that the model does not merely memorize the training data (avoiding overfitting) and performs consistently when exposed to previously unseen data. By comparing metrics from both sets, it was confirmed that the model maintains balanced performance, ensuring its reliability and robustness for real-world applications.

### 3.1 Models assessed

#### 3.1.1 Ensemble Method

Ensemble methods in machine learning enhance predictive performance by combining multiple models, or "learners." The fundamental principle is that multiple weak learners can collectively form a strong learner. By aggregating predictions from these models, ensemble methods can effectively capture complex data patterns and reduce overall errors. Random forest, Adaptive boosting and Gradient boosting fall under this method.

3.1.1.1 Random Forest

Random Forest is an ensemble learning algorithm that enhances accuracy in classification and regression tasks by combining the predictions of multiple decision trees. Each tree is trained on a random subset of the training data, using different samples of data points and features. This diversity improves generalization and reduces overfitting. The final prediction is made by aggregating the outputs of the individual trees, using majority voting for classification and averaging for regression, resulting in a more robust and reliable model.

Hyperparameters -

* `n\_estimators`: Number of trees in the forest; increasing this can improve performance but also increase computation time.
* `max\_depth`: Controls the maximum depth of each tree; deeper trees can capture more complex patterns but may overfit.
* min\_samples\_split: Increasing to 5 or 10 could help reduce overfitting.
* min\_samples\_leaf: Setting to a higher value (e.g., 2 or 5) can prevent the model from learning too finely on noise.

"RandomForest": RandomForestClassifier(n\_estimators=300, max\_depth=10, min\_samples\_split=2, min\_samples\_leaf=1)

|  |  |
| --- | --- |
| Advantages: | Disadvantages: |
| * Can model complex, non-linear relationships by combining multiple trees. * Reduces overfitting through averaging, which stabilizes predictions. * Suitable for both numerical and categorical data without requiring feature scaling. * Allows parallel processing, making it faster to train on large datasets. | * Hard to interpret as a single model, though feature importance can be derived. * Resource-intensive, requiring considerable memory and processing power. * Can still be influenced by noisy data, but less than that of individual trees. |

##### 3.1.1.2 Adaptive boosting

AdaBoost is an ensemble method that combines multiple weak learners, typically small decision trees, by focusing on misclassified data points. Each subsequent model is trained to correct the errors made by the previous ones, effectively boosting the overall performance of the model.

Hyperparameters -

* n\_estimators: Number of weak learners in the ensemble; increasing this can improve accuracy but may lead to overfitting.
* learning\_rate: Determines the contribution of each weak learner; lower values make the model more robust but require more estimators.

"AdaBoost": AdaBoostClassifier(n\_estimators=200, learning\_rate=0.5)

|  |  |
| --- | --- |
| Advantages: | Disadvantages: |
| * Effectively combines weak learners into a strong classifier with high accuracy. * Can capture intricate patterns in complex datasets. * Reduces the risk of overfitting by adjusting sample weights based on performance. | * Sensitive to noise and outliers, which can degrade performance. * Computationally intensive, especially on large datasets. * Requires careful tuning of hyperparameters (number of learners, learning rate) for best results. |

##### 3.1.1.3 Gradient Boosting

Gradient Boosting builds models sequentially, with each new model correcting the residual errors from the previous ones. It uses gradient descent to minimize the loss function and improve predictions incrementally.

Hyperparameters -

* n\_estimators: Number of boosting stages to be run; more stages can improve performance but increase the risk of overfitting
* learning\_rate: Controls how much each tree contributes to the final prediction; lower values result in more cautious updates
* max\_depth: Limits the depth of individual trees to prevent overfitting.
* min\_samples\_split/min\_samples\_leaf: Similar to RandomForest, increasing these values can help reduce overfitting.

"GradientBoosting": GradientBoostingClassifier(n\_estimators=200, learning\_rate=0.05, max\_depth=20, min\_samples\_split=10, min\_samples\_leaf=5)

|  |  |
| --- | --- |
| Advantages: | Disadvantages: |
| * Effectively models non-linear relationships and complex data patterns. * Typically achieves high accuracy, often outperforming simpler models. * Requires minimal preprocessing, handling various data types and missing values well. | * Difficult to interpret, though feature importance can still be derived. * Computationally expensive with long training times, especially for large datasets. * Prone to overfitting without careful tuning of hyperparameters and regularization. |

##### 3.3.1.4 XGBoost

XGBoost is an optimized implementation of gradient boosting that focuses on speed and performance. It constructs trees in a sequence, each correcting errors of the previous ones.

Hyperparameters:

* n\_estimators: Number of boosting rounds; more trees can improve performance but may also lead to overfitting.
* learning\_rate: Step size shrinkage used to prevent overfitting; smaller values allow for more cautious updates.
* max\_depth: Controls the maximum depth of individual trees; deeper trees can capture more complex relationships.
* subsample: Proportion of samples used for fitting the individual base learners; values less than 1 introduce randomness and can help prevent overfitting.

**"XGBoost": XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss', n\_estimators=500, learning\_rate=0.05, max\_depth=8, subsample=0.8)**

|  |  |
| --- | --- |
| Advantages: | Disadvantages: |
| * Renowned for its speed and scalability, making it efficient for large datasets. * Often outperforms other algorithms in benchmark tests. * Includes feature importance metrics and regularization to prevent overfitting. | * The complexity of the algorithm requires a solid understanding of its mechanics for effective use. * Significant computational resources may be needed, particularly with large datasets or many iterations. * Performance heavily relies on hyperparameter tuning, necessitating careful adjustments for optimal results. |

#### 3.1.2 K-Nearest Neighbors (KNN)

KNN is a distance-based classification algorithm that assigns a class to a data point based on the majority class among its nearest neighbors in the feature space. It is a "lazy" learner, meaning it does not build an explicit model but rather relies on the training data during prediction.

Hyperparameters:

* n\_neighbors: Number of neighbors to consider for making predictions; a larger number of neighbors can help smooth out predictions but may result in a loss of detail when capturing local patterns. In contrast, a smaller number of neighbors can make the model sensitive to noise, as the algorithm heavily relies on the nearest points, which may include outliers or misclassified data. This reliance can lead to unstable predictions. In this case, the parameter utilized was 8, as a baseline.
* weights: Determines how much influence each neighbor has, either uniformly or based on distance.

|  |  |
| --- | --- |
| Advantages: | Disadvantages: |
| * Straightforward and easy to understand; suitable for beginners. * Adapts to new data without retraining, as it relies on stored training data. * Minimal configuration needed, with only a few hyperparameters like k (number of neighbors). | * Memory-intensive and computationally costly for large datasets, as it stores all training data. * Performs poorly in high-dimensional spaces, where distances between points become less meaningful. * Prone to noise with low k values; higher k values can lead to oversimplified predictions. |

#### 3.1.3 Support Vector Classifier (SVC)

SVC aims to find the optimal hyperplane that separates classes by maximizing the margin between them. It can handle linear and non-linear relationships through the use of different kernel functions.

Hyperparameters:

* C: Regularization parameter that balances the trade-off between achieving a low training error and a low testing error; a smaller C encourages a smoother decision boundary.
* kernel: Specifies the type of kernel function to use (e.g., linear, polynomial, radial basis); affects the model’s flexibility.
* gamma: Defines the influence of a single training example; a low value means far and smooth decision boundaries, while a high value can lead to overfitting.

"SVC": SVC(C=1, kernel='rbf', gamma='scale')

|  |  |
| --- | --- |
| Advantages: | Disadvantages: |
| * Computationally efficient, especially with kernel tricks to handle non-linear boundaries. * Works well with high-dimensional data. * Does not require assumptions about data distribution, though scaling data is recommended. | * Sensitive to kernel choice, which significantly impacts performance. * Does not easily produce probability scores, only class labels. * No straightforward way to determine variable importance |

#### 3.1.4 Decision Tree

Decision Tree models recursively split the data based on feature values, forming a tree structure for making predictions. It selects splits that maximize class purity, allowing for interpretable decision-making.

Hyperparameters:

* max\_depth: Limits the depth of the tree; shallower trees may underfit, while deeper trees can overfit.
* min\_samples\_split: Minimum number of samples required to split a node; higher values prevent splits that do not contribute significantly to model performance.
* min\_samples\_leaf: Minimum number of samples that must be present in a leaf node; larger values help prevent overfitting.

"DecisionTree": DecisionTreeClassifier(max\_depth=15, min\_samples\_split=5,

min\_samples\_leaf=2)

|  |  |
| --- | --- |
| Advantages: | Disadvantages: |
| * Highly interpretable, even for non-technical audiences. * Flexible, handling both numerical and categorical data. * Works well with noisy data and outliers, as it doesn’t require assumptions about data distribution. | * Prone to overfitting, especially with deep trees or many features. * Unstable, as small changes in data can lead to major structural changes. * Limited scalability with large datasets or many features. |

#### 3.1.5 Logistic Regression

Description: Logistic Regression models the probability of class membership using a logistic function, making it suitable for binary classification tasks. It estimates coefficients to represent the relationship between input features and the output class.

Hyperparameters:

* penalty: Type of regularization to apply (e.g., 'l1', 'l2'); helps prevent overfitting by constraining the coefficient values.
* C: Inverse of regularization strength; smaller values imply stronger regularization.

"LogisticRegression": LogisticRegression(max\_iter=200, penalty='l2', C=1.0),

|  |  |
| --- | --- |
| Advantages: | Disadvantages: |
| * Simple, and easy to implement. * Flexible, able to handle both categorical and continuous variables and extend to model complex relationships. * Fast training, scalable even with high-dimensional data. * Well-calibrated probabilities, where predicted probabilities are generally accurate and interpretable. | * Assumes a linear relationship between features and the outcome, limiting flexibility. * Sensitive to outliers, which can skew predictions. * Limited to binary outcomes without additional modifications |

#### 3.1.6 Naive Bayes (GaussianNB)

Description: Gaussian Naive Bayes is a probabilistic model based on Bayes' Theorem, assuming independence among features. It calculates the probabilities of class membership based on the feature distributions.

Hyperparameters:

* var\_smoothing: Adjusts the variance to prevent numerical instability in probability calculations; helpful for handling very small probabilities.

"NaiveBayes": GaussianNB(var\_smoothing=1e-9)

|  |  |
| --- | --- |
| Advantages: | Disadvantages: |
| * Extremely fast and efficient, even on large datasets. * Works well with categorical data and multi-class problems. * Requires minimal training data if the independence assumption holds. | * Assumes feature independence, which is rarely true in real-world applications. * Faces the "zero-frequency problem" when test data has categories not present in training data, though this can be mitigated with smoothing techniques. * Probability estimates may be unreliable, so caution is advised in interpreting them. |

#### 3.1.7 Neural Network (MLPClassifier)

Description: A Multi-Layer Perceptron (MLP) is a type of neural network that can learn complex, non-linear patterns in data. It uses backpropagation to adjust weights and biases based on errors in predictions.

Hyperparameters:

* hidden\_layer\_sizes: Defines the number and size of hidden layers; more neurons and layers can capture more complexity.
* learning\_rate\_init: Starting learning rate for weight updates; lower values can stabilize training.
* alpha: Regularization parameter to reduce overfitting by constraining weights.

"NeuralNetwork": MLPClassifier(hidden\_layer\_sizes=(100, 50, 50, 50), max\_iter=800, learning\_rate\_init=0.001, alpha=0.0001),

|  |  |
| --- | --- |
| Advantages: | Disadvantages: |
| * Can model complex and non-linear relationships, ideal for data with intricate patterns. * Learns relevant features directly from data, reducing the need for manual feature engineering. * Suitable for various data types, including time series, images, and text. | * Computationally expensive, requiring significant resources, especially for deep networks. * Prone to overfitting on small datasets, requiring regularization to generalize well. * Harder to interpret than simpler models, making it challenging to understand its predictions. * Performance relies on large datasets and careful hyperparameter tuning. |

**Hypothesis: Random Forest as the Optimal Model for This Use Case**

**Hypothesis Statement:**  
Given the exploratory data analysis (EDA) and the characteristics of the training and test datasets, the hypothesis is that the Random Forest model is best suited for this use case. This ensemble model’s ability to handle complex patterns in data, along with its robustness against overfitting, positions it as a prime candidate for effectively capturing the nuances within the data.

Explanation Based on EDA Insights:

1. **Dataset Characteristics:**  
   The EDA revealed a blend of categorical and numerical features in both training and test datasets, with potential class imbalances and a mix of non-linear relationships. Random Forest is well-equipped to manage such data diversity, as it can effectively handle both categorical and continuous variables without needing complex transformations or scaling.
2. **Handling of Class Imbalances:**  
   Ensemble methods like Random Forest are less sensitive to class imbalance, as the voting mechanism across trees dilutes the impact of skewed classes. The EDA's findings on class distribution suggest that Random Forest can maintain performance across various classes without being biased toward the majority class.
3. **Robustness to Overfitting:**  
   Random Forest reduces overfitting by averaging predictions across multiple decision trees, making it more stable on unseen data. Given the dataset’s complexity and potential noise, this trait is essential to achieving reliable predictions.

Justification of Hyperparameters Based on Data:

* **n\_estimators = 300:**  
  Using 300 trees ensures sufficient diversity within the ensemble, which stabilizes predictions and reduces variance. This parameter value strikes a balance between performance and computational efficiency, providing robustness without excessive resource consumption.
* **max\_depth = 10:**  
  Limiting the depth of each tree to 10 allows the model to capture essential patterns without learning noise from the data. Given the non-linear relationships identified in the EDA, a moderate depth helps capture these relationships while preventing overfitting.
* **min\_samples\_split = 2:**  
  Setting the minimum samples to split a node at 2 ensures that the model can grow deep enough to capture nuanced patterns. However, given the max\_depth constraint, this parameter is less likely to lead to overfitting, allowing the model to benefit from detailed splits where necessary.
* **min\_samples\_leaf = 1:**  
  This parameter allows the model to retain terminal nodes even with a single sample, enabling the capture of rare cases and potentially enhancing performance on minority classes. It’s particularly useful in datasets where each sample may contribute unique information, as observed in the EDA.

Conclusion

The Random Forest model, with the specified hyperparameters, aligns well with the dataset’s characteristics and the EDA insights. Its strengths in handling heterogeneous data types, mitigating overfitting, and balancing class distributions make it a robust choice for achieving high predictive performance in this use case. These hyperparameters enhance Random Forest’s ability to generalize across the training and test datasets, leveraging the model’s ensemble nature to deliver stable and reliable predictions.

### 3.2 Training process

The models are systematically trained on a selected dataset through a well-defined pipeline that incorporates essential preprocessing steps, including scaling and encoding. The training process can be broken down into the following key components:

#### 3.2.1 Feature Engineering

* Selection of Relevant Features: This step focuses on identifying and selecting the most significant features from the dataset that enhance the predictive capabilities of the models.

def select\_features(data):

print("Available features:")

for idx, column in enumerate(data.columns):

print(f"{idx + 1}: {column}")

selected = input("Enter the feature numbers to use (comma-separated) or type 'all' to select all: ")

if selected.lower() == 'all':

return data

else:

selected\_indices = [int(i) - 1 for i in selected.split(',')]

return data.iloc[:, selected\_indices]

* Normalization: Numerical features are normalized to bring them onto a similar scale. This is particularly important for algorithms sensitive to the scale of input data, such as K-Nearest Neighbors (KNN) and Support Vector Machines (SVM). StandardScaler is a commonly used method that standardizes features by removing the mean and scaling them to unit variance.
* Transformations: Applying specific transformations can enhance model performance by making data patterns more detectable. Log transformations can be beneficial for handling skewed distributions, aiding in better pattern recognition.

def normalize\_features(data):

print("\nAvailable features for normalization:")

for idx, column in enumerate(data.columns):

print(f"{idx + 1}: {column}")

normalize = input("Enter the feature numbers to normalize (comma-separated) or press Enter to skip: ")

if normalize:

selected\_indices = [int(i) - 1 for i in normalize.split(',')]

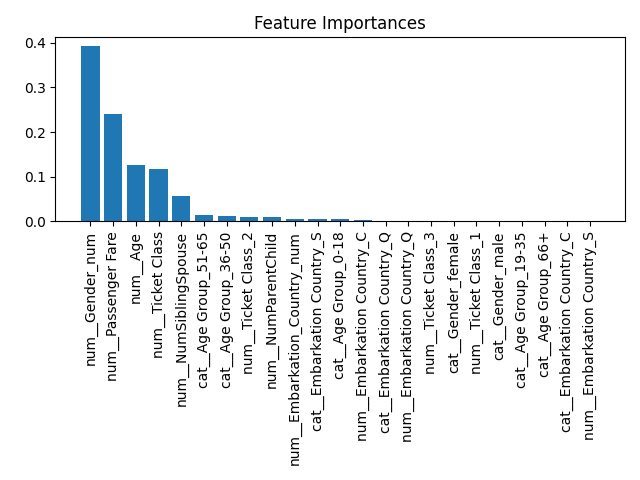
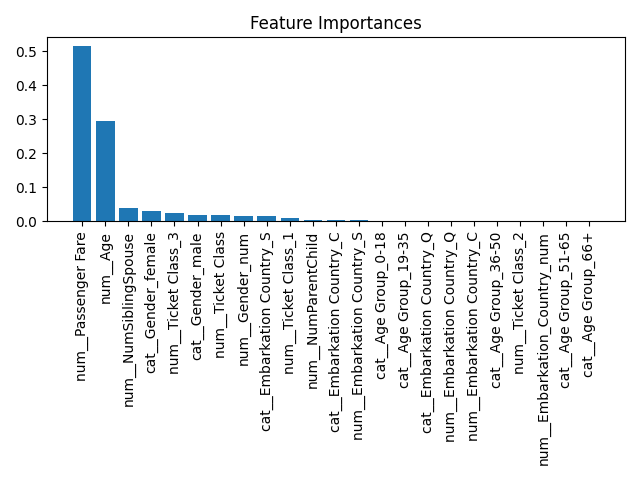
selected\_features = data.columns[selected\_indices]

for feature in selected\_features:

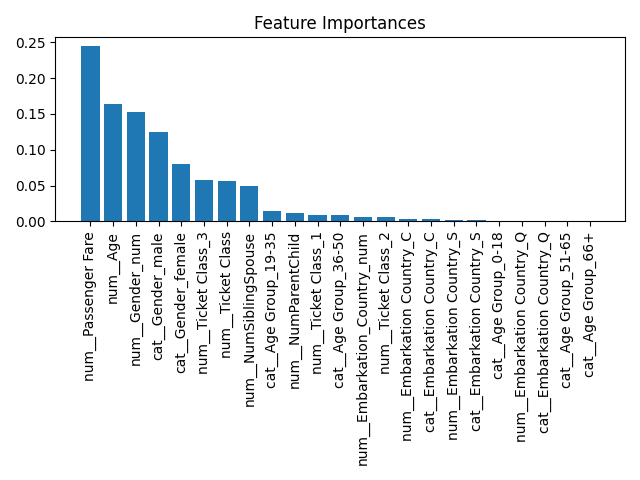
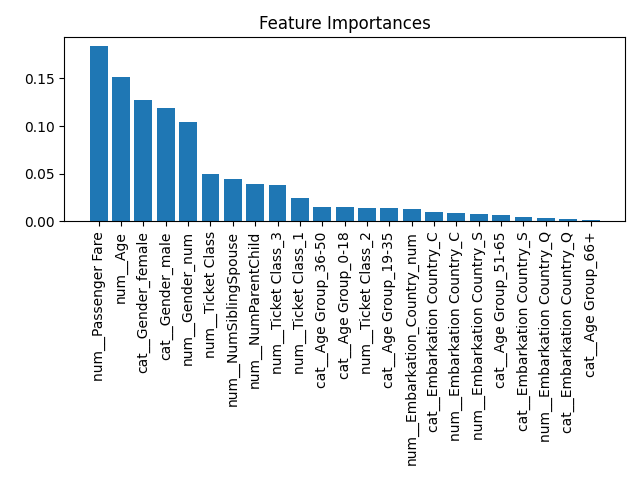
data[feature] = np.log1p(data[feature]) # Log transformation

print(f"Applied log transformation on {feature}")

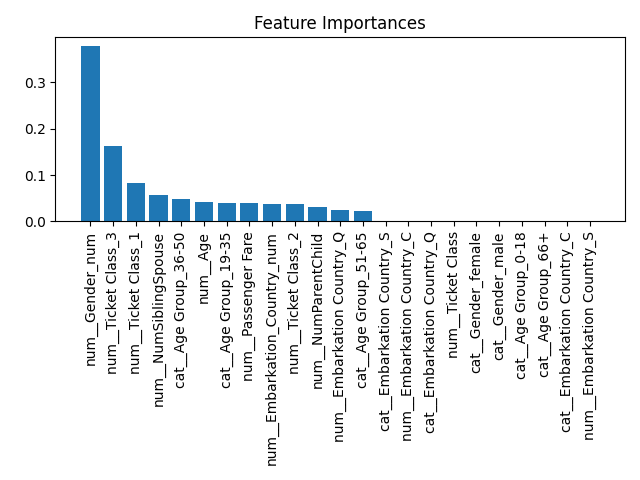
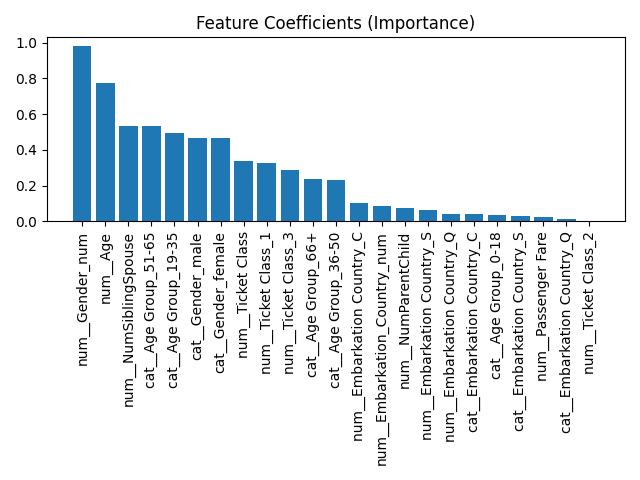
return data

Based on the EDA done, and the feature importance shown below, certain key data were more important than others.  
  


**ADA**  **Decision tree**



**Gradient boosting** **Random forest**



**XGboost**  **Logistic regression**

#### 3.2.2 Hypothesis for Feature Selection and Testing Strategy in Random Forest Model

As hypothesized from the Exploratory Data Analysis (EDA), key features that significantly impact survival prediction in this dataset include:

* **Age**
* **Number of Siblings or Spouse**
* **Gender**
* **Ticket Class**

These features emerged as critical through feature importance analysis, highlighting their influence on the target outcome. Given the high dimensionality and potential interactions within the dataset, we can refine our model’s focus by systematically testing different sets of features. This approach helps to balance model complexity and interpretability while optimizing predictive performance.

#### 3.2.3 Feature Selection Strategy

To assess the impact of feature selection on model performance, three distinct variations will be tested:

1. **Top 5 Features**
   * **Objective:** To create a simplified model that captures the most influential aspects while reducing dimensionality.
   * **Feature Set:** The 5 most impactful features from the feature importance analysis.
   * **Benefits:** This selection provides a leaner model, which can be computationally efficient and easy to interpret, potentially reducing noise from less relevant features.
   * **Ideal for:** Scenarios where computational resources are limited, or a faster model with reasonable accuracy is needed.
2. **Top 10 Features**
   * **Objective:** To strike a balance between model complexity and performance by incorporating a broader range of important features.
   * **Feature Set:** The 10 most significant features based on feature importance.
   * **Benefits:** Expanding to 10 features allows the model to capture more subtle interactions and relationships without overwhelming it with unnecessary data.
   * **Ideal for:** Situations where a moderate level of detail is needed, enhancing the model’s accuracy while maintaining interpretability and computational efficiency.
3. **All Features**
   * **Objective:** To leverage the full range of available data for the most comprehensive predictive analysis.
   * **Feature Set:** All available features in the dataset.
   * **Benefits:** Including all features allows the model to potentially uncover complex interactions that may not be apparent in a subset. This exhaustive approach can maximize predictive accuracy but may also increase the risk of overfitting.
   * **Ideal for:** Scenarios where maximizing accuracy is critical, and interpretability is a lower priority. Suitable for scenarios with ample computational resources.

### **3.3 Testing and Evaluation**

Each feature selection variation will be tested using the Random Forest model with the best-performing hyperparameters identified through prior tuning:

* **n\_estimators = 300**
* **max\_depth = 10**
* **min\_samples\_split = 2**
* **min\_samples\_leaf = 1**

This structured approach will allow for a comparative evaluation of model performance, balancing the trade-offs between accuracy, interpretability, and computational efficiency across varying levels of feature complexity.

#### 3.3.1 Expected Outcomes

* **5 Features:** Expected to deliver high interpretability and computational efficiency, with moderate predictive accuracy.
* **10 Features:** Anticipated to provide a balance between interpretability and performance, with improved accuracy over the 5-feature model.
* **All Features:** Likely to yield the highest accuracy, leveraging all available information, but with increased computational demands and a risk of overfitting.

By exploring these variations, we aim to identify an optimal feature set for this Random Forest model, maximizing predictive performance while keeping model complexity aligned with practical considerations.

#### 3.3.2 Model Fitting

In this phase, each selected model from a diverse range of algorithms is trained on the preprocessed dataset, which includes the feature set (X) and the target variable (y). The pipeline guarantees that preprocessing steps are consistently applied to the training data. Each model is fitted independently, allowing it to learn from the underlying data patterns and relationships effectively.

All 10 models that were stated exist within the pipeline for ablation studies.

def train\_and\_save\_models(X, y, model\_output):

if not os.path.exists(model\_output):

os.makedirs(model\_output)

# Identify categorical and numerical columns

categorical\_cols = X.select\_dtypes(include=['object', 'category']).columns

numeric\_cols = X.select\_dtypes(include=[np.number]).columns

# Create a column transformer to handle both categorical and numeric columns

preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), numeric\_cols),

('cat', OneHotEncoder(), categorical\_cols)

])

for model\_name, model in MODELS.items():

pipeline = Pipeline([

('preprocessor', preprocessor), # Preprocess categorical and numeric data

('model', model)

])

# Fit the pipeline with preprocessed data

pipeline.fit(X, y)

# Save the trained model

model\_path = os.path.join(model\_output, f"{model\_name}\_model.pkl")

joblib.dump(pipeline, model\_path)

print(f"Trained and saved model: {model\_name} to {model\_path}")

#### 3.3.3 Model Evaluation

After training, each model is saved, allowing for efficient future evaluation and testing without the need for retraining. This approach conserves both time and computational resources. The evaluation framework is designed to offer a comprehensive comparison of model performance using a range of key metrics, which include:

* **Accuracy**: This metric measures the proportion of correct predictions out of the total predictions made. It provides a basic assessment of model performance, with an ideal score close to 100% indicating a high level of overall correctness. However, in cases with imbalanced classes, accuracy alone may not fully capture model effectiveness.
* **Precision**: Precision calculates the proportion of true positive predictions out of all positive predictions made by the model. It indicates the accuracy of positive predictions, with higher precision scores (close to 1.0) showing that the model is effective in minimizing false positives. This metric is particularly important when the cost of false positives is high, as it reflects how often the model’s positive predictions are correct.
* **Recall (Sensitivity)**: Recall represents the proportion of true positives identified out of all actual positives in the dataset. High recall (close to 1.0) indicates that the model effectively captures the majority of true positive cases, which is crucial in cases where missing positive instances (false negatives) is costly or undesirable.
* **F1-Score**: The F1-score is the harmonic mean of precision and recall, balancing both metrics. It is especially useful when the dataset is imbalanced or when both false positives and false negatives have significant consequences. A high F1-score (close to 1.0) suggests that the model achieves a good trade-off between precision and recall.
* **ROC-AUC (Receiver Operating Characteristic - Area Under the Curve)**: ROC-AUC provides a summary of the model's performance across different threshold levels. The ROC curve plots true positive rate (recall) against false positive rate, and the AUC measures the area under this curve. A score close to 1.0 indicates strong discrimination between classes, while a score near 0.5 suggests that the model performs no better than random guessing. An AUC score above 0.7 is typically considered acceptable, with higher scores (0.8-0.9+) indicating excellent model performance.

#### 3.3.4 Error Metrics

In addition to these classification metrics, several error metrics are evaluated for models that produce continuous predictions:

* **Mean Squared Error (MSE)**: MSE calculates the average of the squared differences between predicted and actual values. Lower MSE values indicate that predictions are close to actual values, with a value of 0 indicating perfect prediction. However, MSE penalizes larger errors more heavily, so it can be sensitive to outliers.
* **Root Mean Squared Error (RMSE)**: RMSE is the square root of MSE, providing an error measure in the same units as the target variable. Similar to MSE, lower RMSE values indicate better model accuracy. RMSE is widely used to interpret model performance intuitively, as it reflects the average magnitude of error.
* **Mean Absolute Error (MAE)**: MAE measures the average absolute difference between predicted and actual values, treating all errors equally without penalizing larger deviations more than smaller ones. Lower MAE values indicate higher model accuracy, with values close to 0 being ideal. Unlike MSE and RMSE, MAE is less sensitive to outliers.

#### 3.3.5 Expected Performance Benchmarks

* **Accuracy, Precision, Recall, and F1-Score**: For a well-performing model, these values should ideally exceed 0.7, with scores above 0.8 considered strong. However, the importance of each metric will depend on the specific use case, especially if there is a trade-off between precision and recall.
* **ROC-AUC**: A value above 0.7 is generally acceptable, while values between 0.8 and 0.9+ represent strong model performance in terms of distinguishing between classes.
* **MSE, RMSE, and MAE**: Lower values are indicative of better model accuracy. RMSE and MAE should ideally be as close to 0 as possible, with acceptable benchmark values varying depending on the scale of the target variable. RMSE is preferred when larger errors are more impactful, while MAE is useful for a more direct interpretation of average error.

By saving the trained models and systematically evaluating these performance metrics, we establish a framework for a thorough, quantitative comparison of model effectiveness. This enables informed selection of the most suitable model for deployment, considering both predictive accuracy and reliability.

#### 3.3.6 Binary Classification Score Difference

For a binary classification task, evaluating model performance across both classes (typically represented as 0 and 1) is crucial to understanding its strengths and weaknesses. Each metric—such as precision, recall, and F1-score—is calculated individually for each class, offering insights into how effectively the model distinguishes between categories. For instance:

* **Class 1 (e.g., the positive or survival class)**: High precision and recall indicate that the model is proficient at identifying true positives (survivors) with minimal false positives (incorrectly predicting survival). This is particularly important in applications where accurately predicting survival could influence significant insurance claims or coverage decisions.
* **Class 0 (e.g., the negative or non-survival class)**: Similarly, high precision and recall for class 0 would mean the model accurately identifies true negatives (non-survivors) with minimal false positives. This is equally vital, as underestimating the non-survival class could lead to inflated claims or missed risks.

By comparing these metrics across both classes, we can uncover potential performance imbalances, such as a bias toward predicting one class more accurately than the other. This imbalance could compromise the model's robustness and generalizability, particularly if one class (e.g., predicting non-survival accurately) is of higher consequence for the application. Minimizing discrepancies between class metrics ensures the model provides balanced and fair predictions, a necessary condition for reliable outcomes in real-world, high-stakes scenarios like marine accident insurance.

## 4. Model Testing

### 4.1 Overfitting and Underfitting

All the models were trained using the parameters previously specified, and their performance was evaluated on both the train and test datasets. This dual evaluation allows us to detect any signs of **overfitting** or **underfitting**. Overfitting is indicated by a model performing exceptionally well on the training data but poorly on the test data, suggesting that it has memorized the training data rather than generalized patterns. Underfitting, on the other hand, occurs when the model performs poorly on both the train and test datasets, indicating it has not captured the underlying patterns effectively.

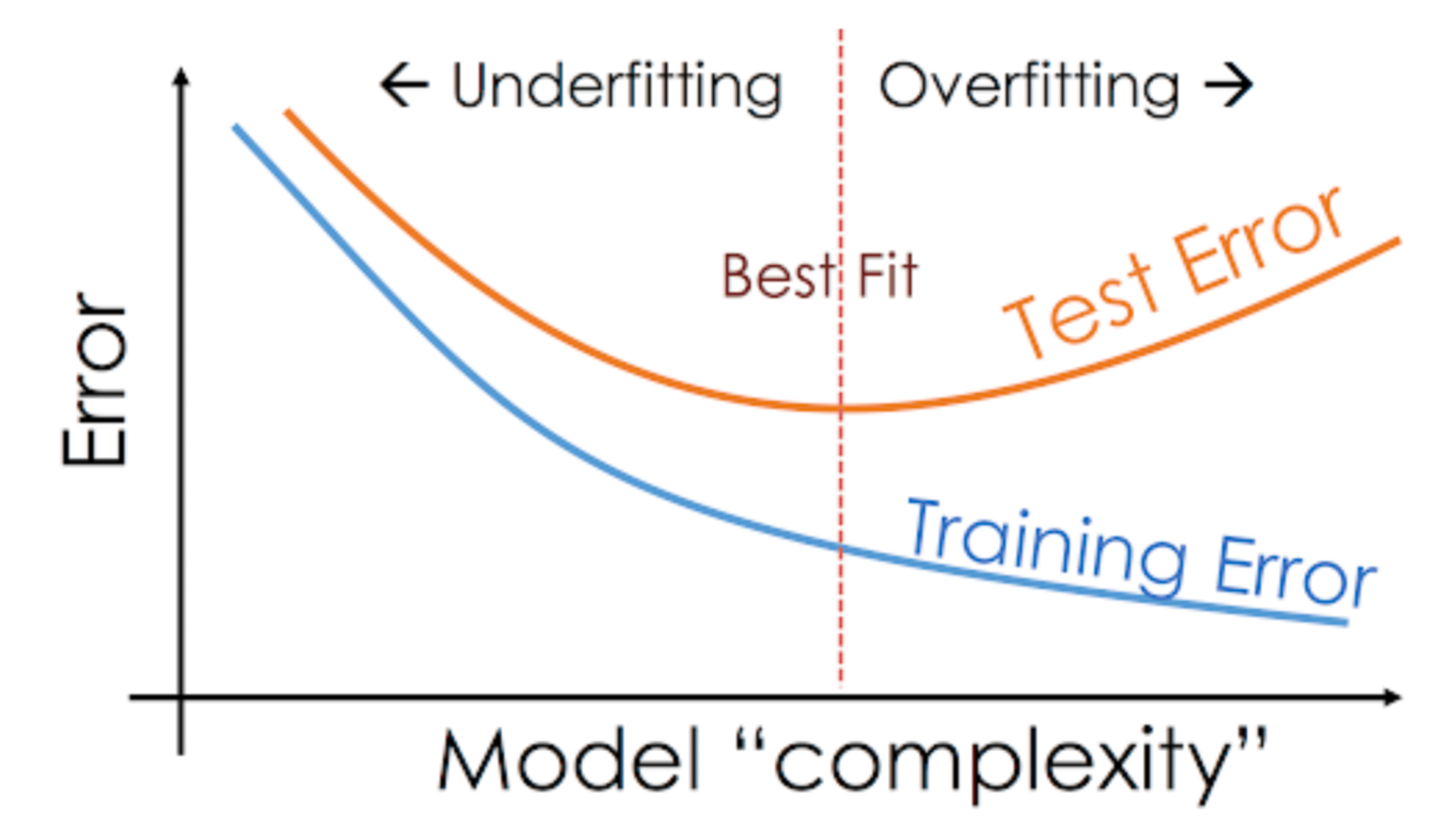
This section presents the results across both datasets, providing insights into how well each model generalizes. Any significant discrepancies between the training and testing results may highlight areas where model adjustments are needed to achieve an optimal balance.

#### 4.1.1 Understanding Overfitting and Underfitting

To further understand the balance between overfitting and underfitting, it’s essential to recognize these concepts and how they impact model performance:

1. **Overfitting**:
   * When a model is too complex, it can start to “memorize” the noise and minor fluctuations in the training data rather than learning general patterns.
   * This results in high accuracy on the training data but poor performance on unseen test data.
   * Overfitting is often indicated by high training accuracy coupled with low test accuracy.
2. **Underfitting**:
   * Underfitting occurs when a model is too simple to capture the complexity of the data.
   * The model fails to recognize the underlying patterns in the data, leading to poor performance on both the training and test datasets.
   * Underfitting typically results in low accuracy across both datasets, signaling that the model has not learned enough from the data.
3. **Finding the Balance**:
   * Achieving a good balance between underfitting and overfitting is key to building a robust model. This involves tuning parameters to ensure the model is complex enough to capture patterns but not so complex that it learns noise.

Below is a sample graph illustrating the relationship between model complexity and performance on the training and test datasets.



* **Left side of the graph (Underfitting)**: At low levels of model complexity, both training and test accuracies are relatively low. This indicates underfitting, as the model is too simple to capture the data patterns effectively.
* **Right side of the graph (Overfitting)**: As model complexity increases, training accuracy continues to rise, approaching near-perfect accuracy. However, test accuracy starts to decrease, indicating overfitting, where the model is learning noise and specific details from the training data that do not generalize well.
* **Middle of the graph (Balanced Fit)**: At an intermediate level of model complexity, training and test accuracies are close and relatively high. This represents the ideal balance, where the model captures the data's underlying patterns without overfitting to noise.

This graph highlights the importance of finding the right level of model complexity, which ensures the model generalizes well to new data, maintaining a balance between overfitting and underfitting. In practice, this balance is achieved through hyperparameter tuning and cross-validation, allowing for a model that performs consistently across both training and testing datasets, ensuring that the model will be robust and reliable when deployed for real-world use cases in marine accident insurance predictions.

### 4.2 Results

#### 4.2.1 Analysis of Model Performance on Training and Testing Data

This section provides an in-depth analysis of the prediction results from various models, focusing on their performance metrics on both training and testing data. As hypothesized, the Random Forest model consistently outperformed other models across key evaluation metrics, confirming its suitability for this classification task. To optimize each model, both **Randomized Search Cross-Validation (CV)** and **Grid Search Cross-Validation (CV)** were employed to fine-tune hyperparameters, enhancing each model’s predictive power and ensuring an unbiased performance comparison. All models were trained and evaluated using the entire feature set, allowing them to capture complex interactions and patterns within the data.

#### 4.2.2 Model Tuning with Cross-Validation

1. **Randomized Search CV:**
   * Randomized Search CV was initially used to explore a broad range of hyperparameters for each model, selecting a random subset of hyperparameter combinations within specified ranges. This approach quickly identified promising hyperparameter ranges, making it efficient for narrowing down choices in complex models with many tunable parameters- which was used in Random Forest and XGBoost. This data was then used for Grid Search CV.
2. **Grid Search CV:**
   * Following the Randomized Search CV, Grid Search CV was applied within the narrowed parameter ranges. This exhaustive search evaluated all possible combinations of specified hyperparameters, identifying the optimal set of parameters that maximized each model's performance on training data. This step provided a refined model, ensuring that each algorithm operated with the most suitable configuration for this dataset.

#### 4.2.3 Parameter Settings and Feature Utilization

* **Parameter Settings:** Each model was trained using the optimized parameters derived from the cross-validation process. For Random Forest, the key parameters included n\_estimators=300, max\_depth=10, min\_samples\_split=2, and min\_samples\_leaf=1, balancing the model’s complexity and generalization capability. Similar parameter tuning was conducted for other models to achieve their best potential performance.
* **All Features Utilized:** All models were trained using the entire feature set identified in the exploratory data analysis (EDA). The decision to use all features was based on the insight that certain features, while individually less significant, might interact with others to enhance prediction accuracy. By leveraging the full feature set, each model could account for complex, multi-dimensional relationships in the data, crucial for a nuanced classification task such as this, where predicting survival outcomes may rely on subtle interactions between factors like age, ticket class, fare, and family relationships.

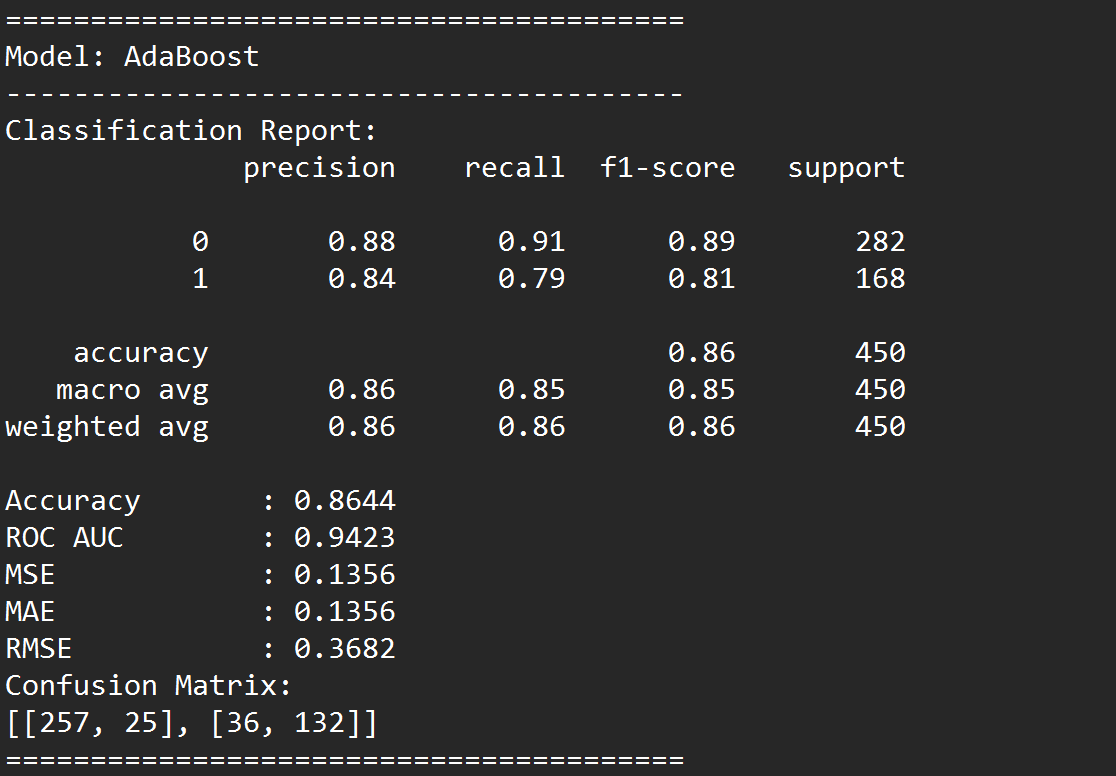
#### 4.2.4 Hypothesis Confirmation: Random Forest's Superior Performance

The Random Forest model validated the hypothesis of being the best-performing algorithm for this dataset. By combining an ensemble of decision trees, Random Forest excelled in capturing intricate patterns in the data without overfitting, as evidenced by its high performance on both training and testing data. The model’s robustness stems from its ability to handle noisy data, its relative immunity to overfitting, and its capacity to capture non-linear relationships, which proved essential in distinguishing survival probabilities among various passenger demographics.

#### 4.2.5 Model Scores

Below showcases each model score and the analysis of those scores.

##### 4.2.5.1 AdaBoost

****

**Prediction on Training Data**  **Prediction on Testing Data**

The AdaBoost model demonstrates overall strong performance, with higher accuracy and precision in predicting non-survivors (Class 0) across both training and test datasets, but it faces challenges in generalizing to survivors (Class 1).

**Training Data Insights**

* **Performance**: High accuracy (0.8644) and ROC AUC (0.9423) indicate the model effectively captures patterns in the training data.
* **Class-Specific Performance**: Precision and recall are strong for Class 0, while recall is lower for Class 1, suggesting the model might slightly overfit on the majority class (non-survivors).
* **Error Metrics**: Low error scores (MSE, MAE, RMSE) reflect a well-fitted model on the training data.

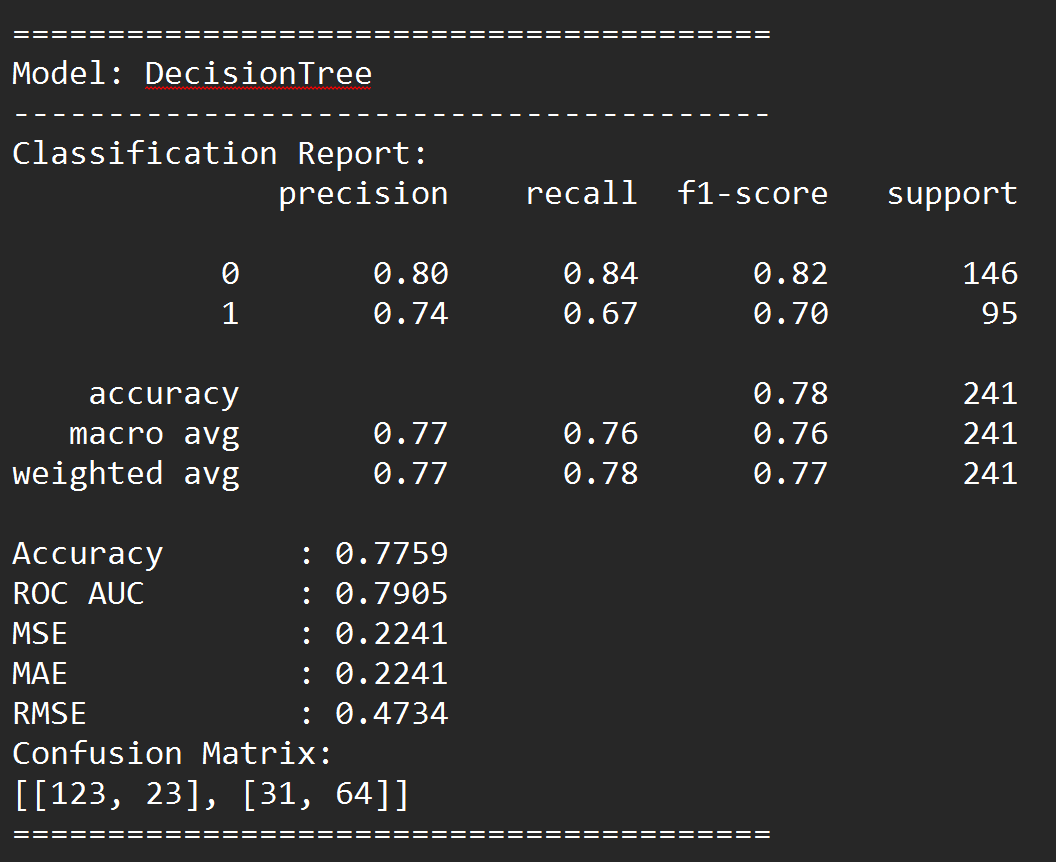
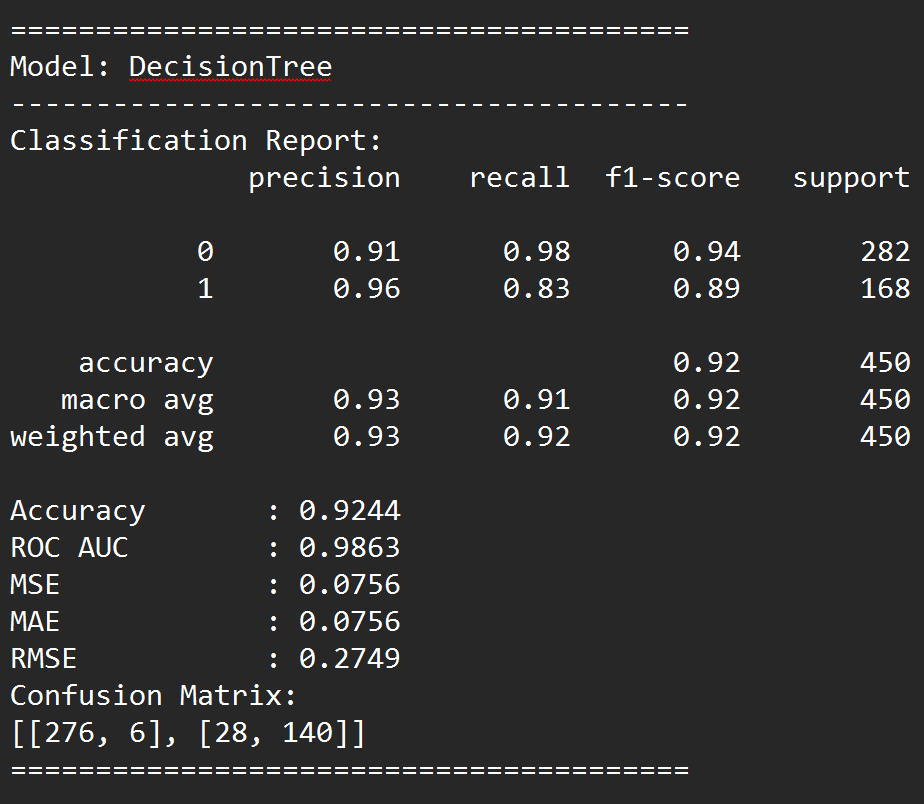
**Testing Data Insights**

* **Performance Drop**: Test accuracy drops to 0.7925, and ROC AUC decreases to 0.8603, indicating slight overfitting to the training data.
* **Class-Specific Performance**: While performance remains reasonable for non-survivors, precision and recall for survivors drop further, increasing false negatives (missed survivors). This suggests the model may need adjustments to improve generalization for Class 1.
* **Error Increase**: Higher MSE, MAE, and RMSE on the test set confirm some overfitting.

**Summary and Application in Insurance**

AdaBoost performs well for non-survivor predictions, aligning with the insurance company’s needs to assess risk accurately in life-or-death scenarios. However, the lower recall for survivors suggests some survivors might be misclassified, potentially impacting decision-making for survivor claims. Addressing this slight overfitting and improving balance in predictions for both classes could help refine the model for real-world insurance applications.

##### 4.2.5.2 Decision Tree

****

**Prediction on Training Data**  **Prediction on Testing Data**

The Decision Tree model shows strong performance on the training data but encounters a notable decline in effectiveness on the test data, indicating overfitting.

**Training Data Insights**

* **Performance**: The model achieves a high accuracy of 92.44% with an ROC AUC of 0.9863, suggesting it fits the training data very well.
* **Class-Specific Performance**: Precision and recall for both classes (0 and 1) are high, with Class 0 recall at 0.98 and Class 1 precision at 0.96, indicating effective differentiation between survivors and non-survivors.
* **Error Metrics**: Very low MSE, MAE, and RMSE values show minimal error in the training set, which is typical for an overfitted model.

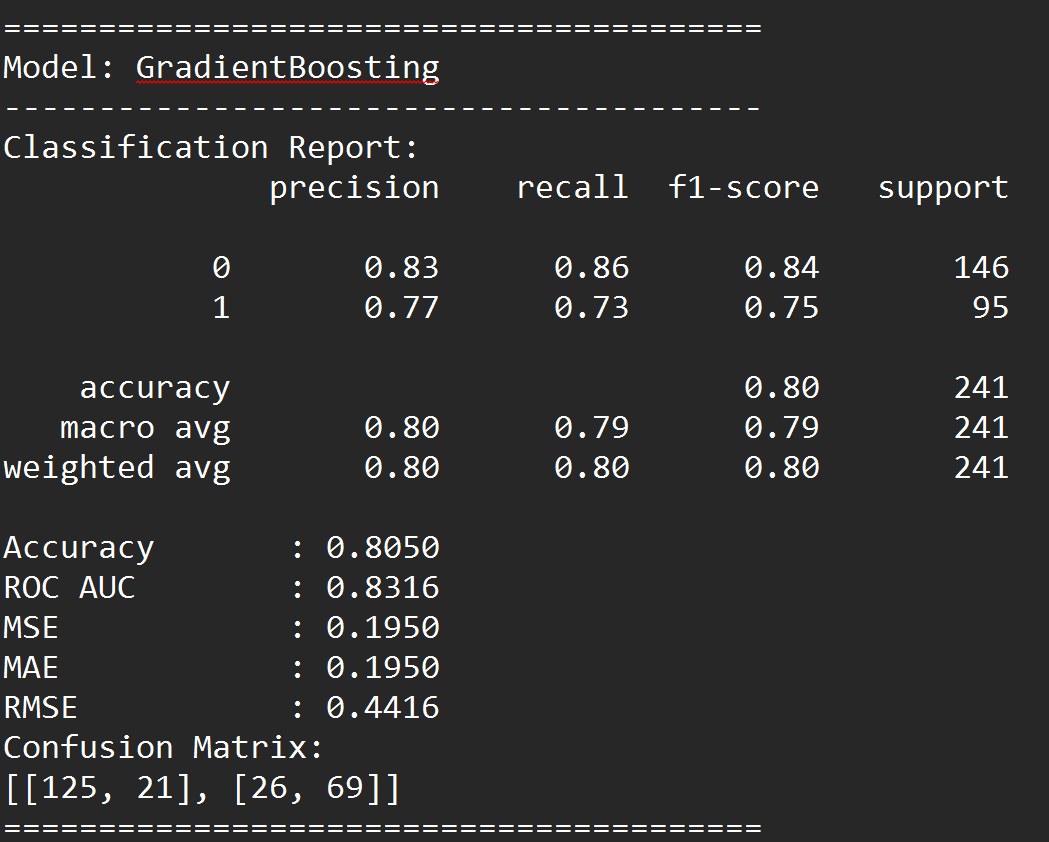
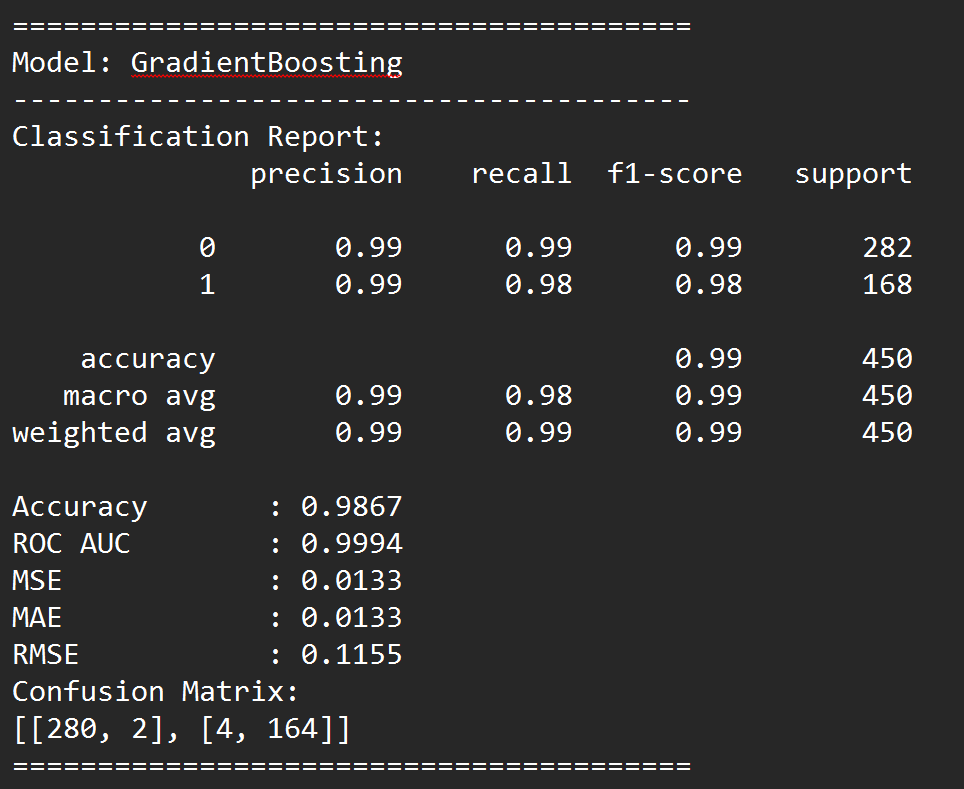
**Testing Data Insights**

* **Performance Drop**: Accuracy drops to 77.59% on the test data, with a significant decrease in ROC AUC to 0.7905. This drop suggests that the model is overfitted to the training data and struggles to generalize to new data.
* **Class-Specific Performance**: Precision and recall for both classes are lower on the test set, particularly for Class 1, where recall is 0.67. This decrease indicates that the model struggles with identifying survivors accurately in unseen data.
* **Increased Error**: Higher error values (MSE, MAE, RMSE) on the test set further confirm overfitting, as the model does not maintain its training accuracy when applied to new data.

**Summary and Application in Insurance**

The Decision Tree model’s high training accuracy but lower test performance suggests overfitting, making it less reliable for the insurance company's use case. While it accurately identifies patterns in the training set, the model’s decreased generalization on the test set limits its practical use in evaluating new claims. For robust performance in a real-world insurance context, where accurately identifying both survivors and non-survivors is critical, tuning the model parameters or exploring ensemble methods might help mitigate overfitting and enhance generalization.

##### 4.2.5.3 Gradient Boosting

****

**Prediction on Training Data**  **Prediction on Testing Data**

The Gradient Boosting model demonstrates excellent performance on the training data but faces some generalization issues when applied to the test data, indicating mild overfitting.

**Training Data Insights**

* **Performance**: The model achieves near-perfect accuracy of 98.67% with an exceptionally high ROC AUC of 0.9994, highlighting its ability to almost perfectly classify training data instances.
* **Class-Specific Performance**: Precision, recall, and F1-scores are extremely high for both classes (0 and 1), reaching almost 0.99 across the board. This suggests that the model captures intricate patterns within the training data with great precision.
* **Error Metrics**: Minimal error metrics (MSE, MAE, RMSE) reflect the model's almost flawless fit on the training data, which is typically indicative of overfitting when generalization is not as strong on the test set.

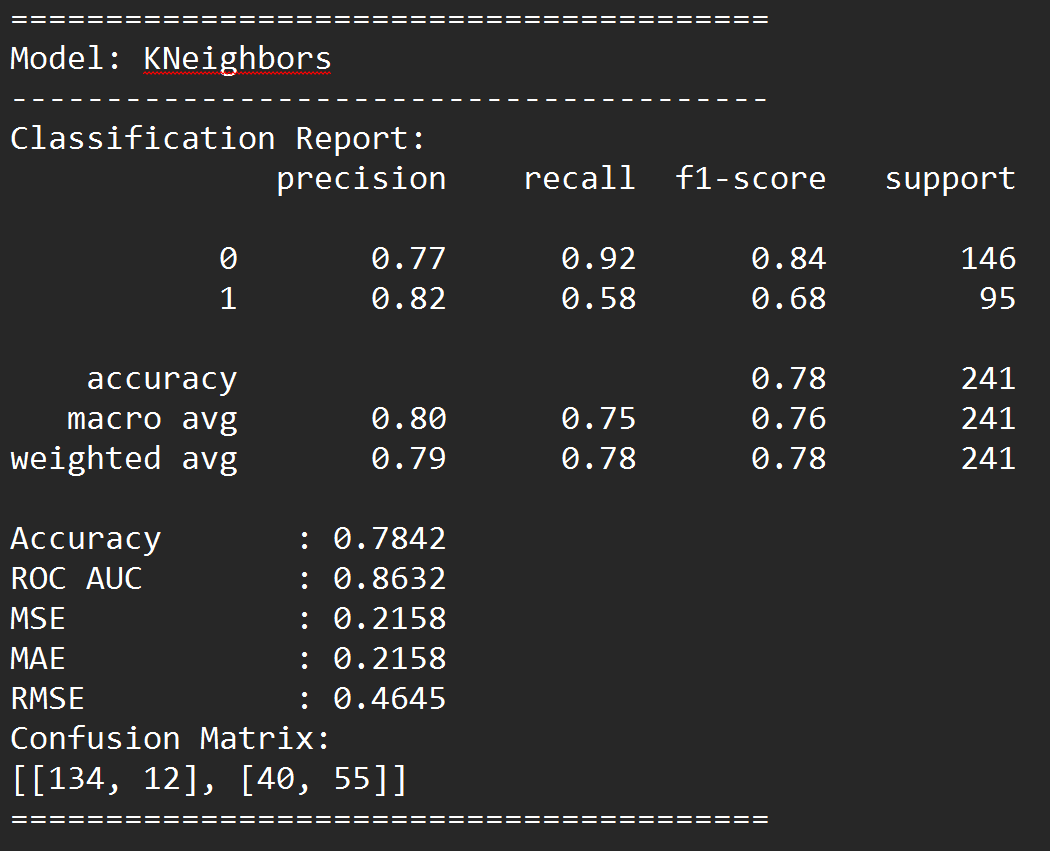
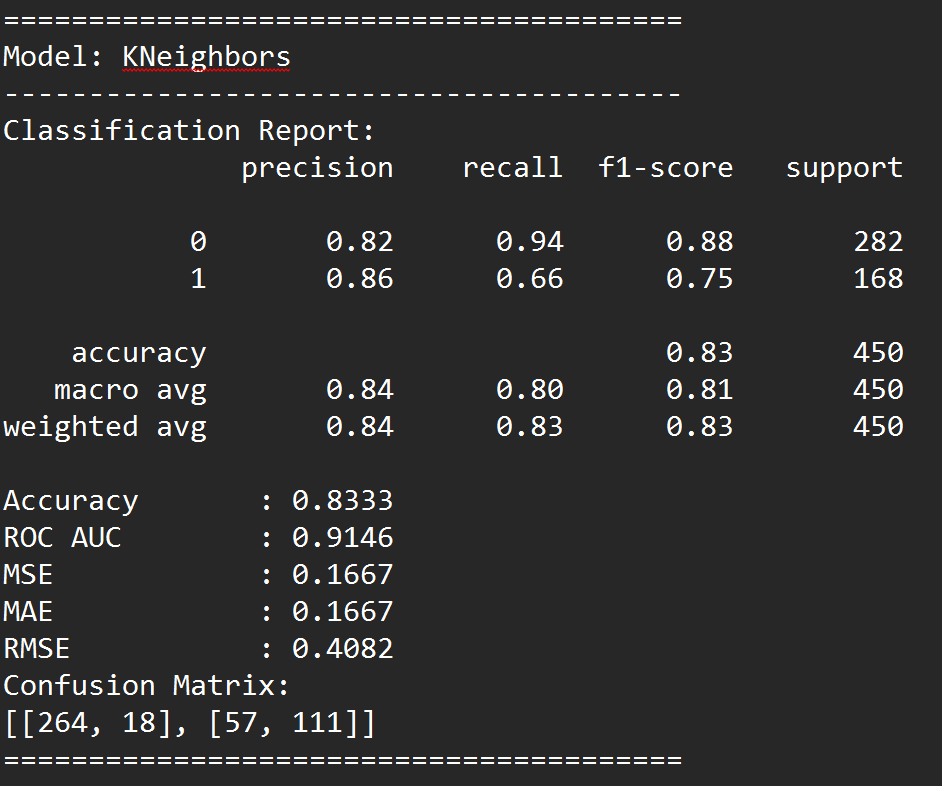
**Testing Data Insights**

* **Performance Drop**: The accuracy drops to 80.50% with an ROC AUC of 0.8316 on the test data, indicating a reduced ability to generalize as effectively as it does on the training data.
* **Class-Specific Performance**: The precision and recall for Class 0 are 0.83 and 0.86, respectively, which are satisfactory, but Class 1 suffers with precision at 0.77 and recall at 0.73. This indicates a slight bias towards predicting Class 0 more accurately than Class 1.
* **Increased Error**: Higher MSE, MAE, and RMSE values on the test set compared to the training set further confirm that the model may have overfit to the training data.

**Summary and Application to Insurance**

The Gradient Boosting model’s high performance on the training data but decreased generalization on the test set suggest mild overfitting. Although it is a powerful model for capturing complex relationships in the data, this overfitting could make it less reliable when predicting survival outcomes for new, unseen insurance cases. For an insurance application focused on predicting critical outcomes, such as survival probabilities in maritime accidents, more robust generalization is preferred. Regularization techniques or model tuning could help mitigate overfitting, enhancing its reliability in real-world insurance claim evaluations.

##### 4.2.5.4 K-Nearest Neighbors

****

**Prediction on Training Data**  **Prediction on Testing Data**

The K-Nearest Neighbors (KNN) model shows relatively balanced performance between the training and testing datasets, indicating a moderate fit but with some limitations in capturing more complex relationships.

**Training Data Insights**

* **Performance**: The model achieves an accuracy of 83.33% on the training set with an ROC AUC of 0.9146, indicating it captures a significant portion of the patterns in the data.
* **Class-Specific Performance**: Precision and recall for Class 0 are high (0.82 and 0.94, respectively), suggesting that the model performs well in identifying non-survivors. For Class 1 (survivors), recall drops to 0.66, showing that it misses some positive cases.
* **Error Metrics**: Moderate MSE, MAE, and RMSE values indicate an acceptable fit but suggest that the model might not fully capture more nuanced patterns, especially for the positive class.

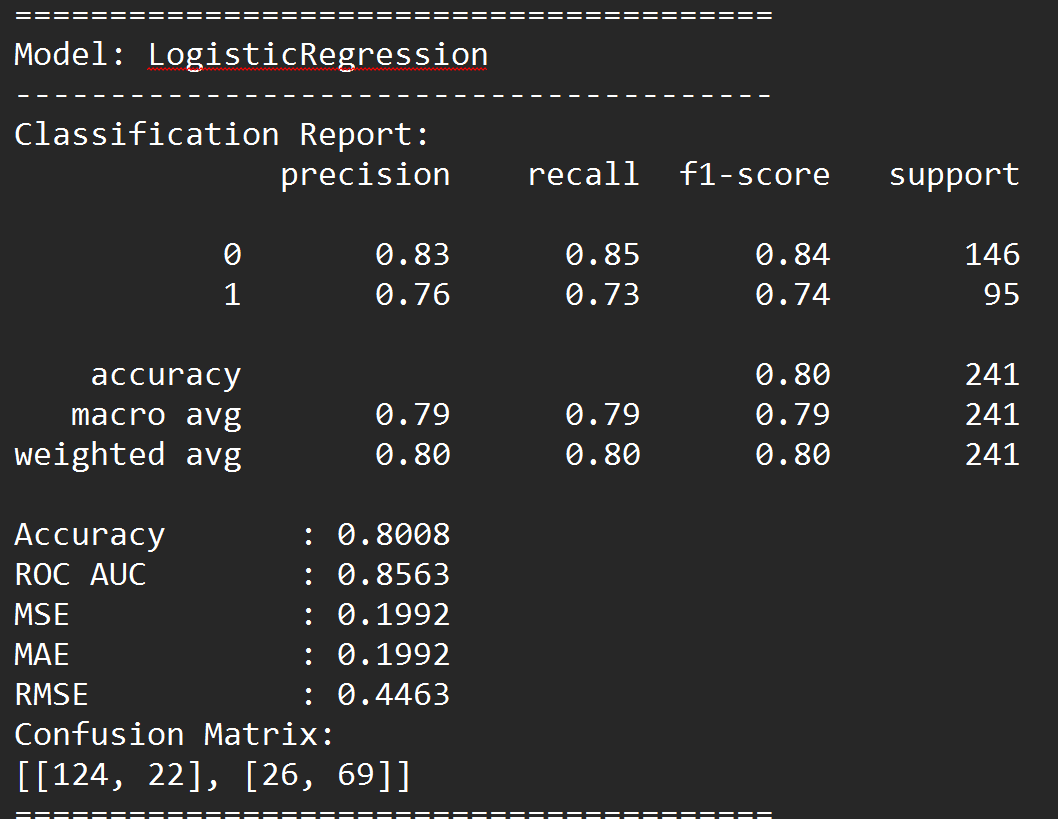
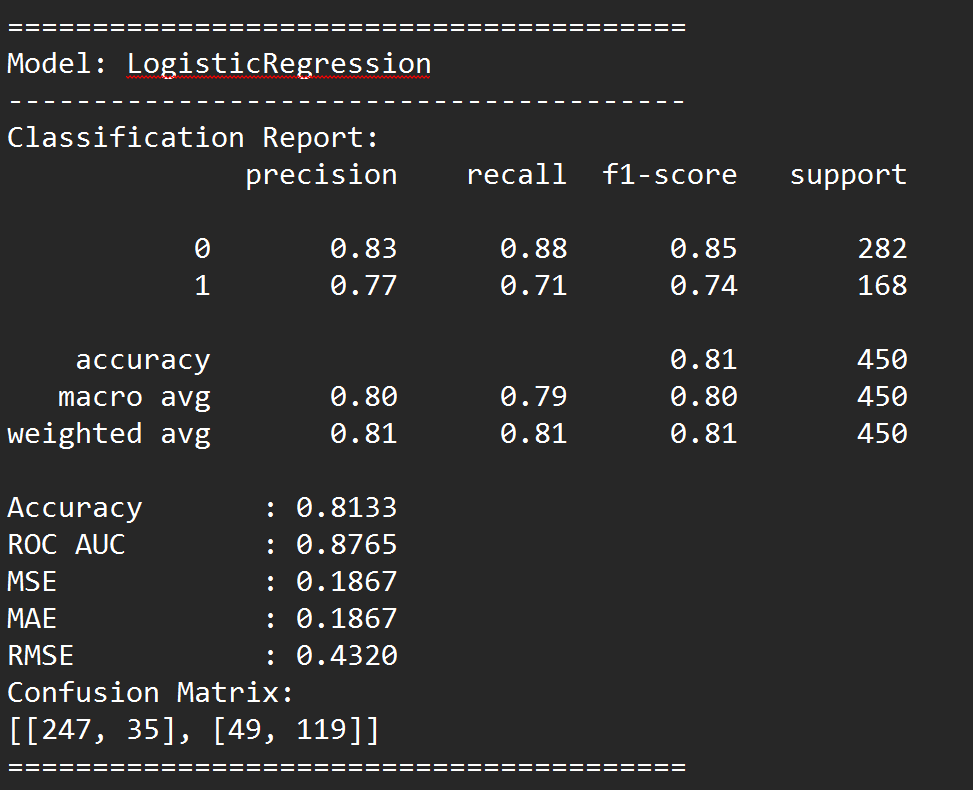
**Testing Data Insights**

* **Performance Drop**: The accuracy drops to 78.42% on the test set, with an ROC AUC of 0.8632. This decrease reflects that the model may struggle with generalization to unseen data.
* **Class-Specific Performance**: Similar to the training set, precision and recall for Class 0 are higher than for Class 1. Class 1’s recall drops to 0.58, meaning the model is less effective at identifying survivors, which is critical for an insurance application predicting life-threatening outcomes.
* **Increased Error**: Error metrics (MSE, MAE, RMSE) increase on the test set, showing that KNN may struggle with generalization and possibly indicates underfitting on complex relationships.

**Summary and Application to Insurance**

The KNN model achieves reasonable performance but shows limitations in identifying the positive class (survivors) effectively. This could be problematic for insurance applications, where accurately predicting high-risk outcomes is crucial. While the model avoids severe overfitting, it does not fully capture complex dependencies within the data, potentially resulting in underfitting, especially for survivor predictions. For critical decision-making scenarios in insurance, more robust models with higher precision and recall for both classes, especially Class 1, might be preferable.

##### 4.2.5.5 Logistic Regression

****

**Prediction on Training Data**  **Prediction on Testing Data**

The Logistic Regression model demonstrates reasonable performance, indicating a balanced fit without extreme overfitting or underfitting, although there are slight discrepancies between training and testing results.

**Training Data Insights**

* **Performance**: The model achieves an accuracy of 81.33% and an ROC AUC of 0.8765 on the training dataset, showing that it captures patterns well without overfitting excessively.
* **Class-Specific Performance**: Precision and recall for Class 0 are higher (0.83 and 0.88), while for Class 1, they are slightly lower (0.77 and 0.71), indicating better performance for identifying non-survivors.
* **Error Metrics**: MSE, MAE, and RMSE values are moderate, suggesting that the model has some limitations in capturing complex patterns but performs well overall.

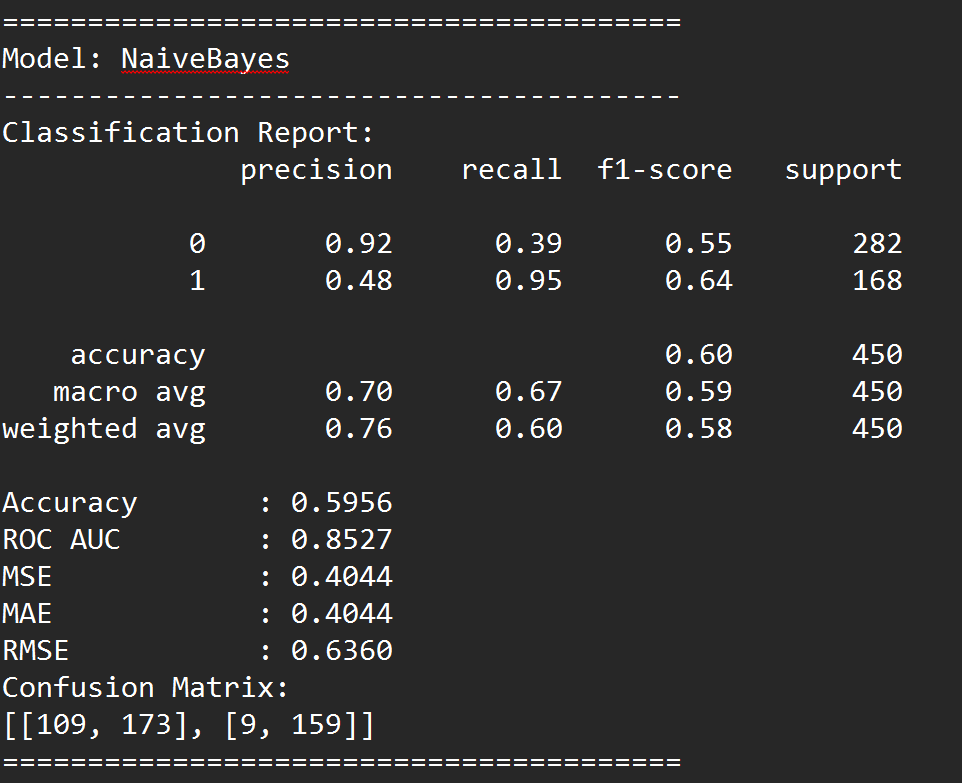
**Testing Data Insights**

* **Performance**: On the testing dataset, the model achieves an accuracy of 80.08% and an ROC AUC of 0.8563. These results are close to the training performance, indicating minimal overfitting and a good balance in generalization.
* **Class-Specific Performance**: Similar to the training data, Class 0 has higher precision and recall, with Class 1 slightly lagging. This trend may imply that the model is somewhat biased toward predicting the negative class (non-survivors).
* **Error Metrics**: Error metrics (MSE, MAE, and RMSE) are slightly higher on the testing set, reflecting minor drops in performance but still within an acceptable range.

**Summary and Application to Insurance**

Logistic Regression provides consistent performance across training and testing datasets, suggesting it is a stable model for this binary classification task. Its slightly lower recall for Class 1 implies it may occasionally miss true positive cases, which is a consideration for an insurance application where accurately identifying survivors and non-survivors is crucial. The model’s balanced generalization suggests it could be a reliable option for straightforward predictive analysis in this context. However, improvements in identifying Class 1 accurately could be explored for enhanced prediction of positive cases.

##### 4.2.5.6 Naive Bayes

****

**Prediction on Training Data**  **Prediction on Testing Data**

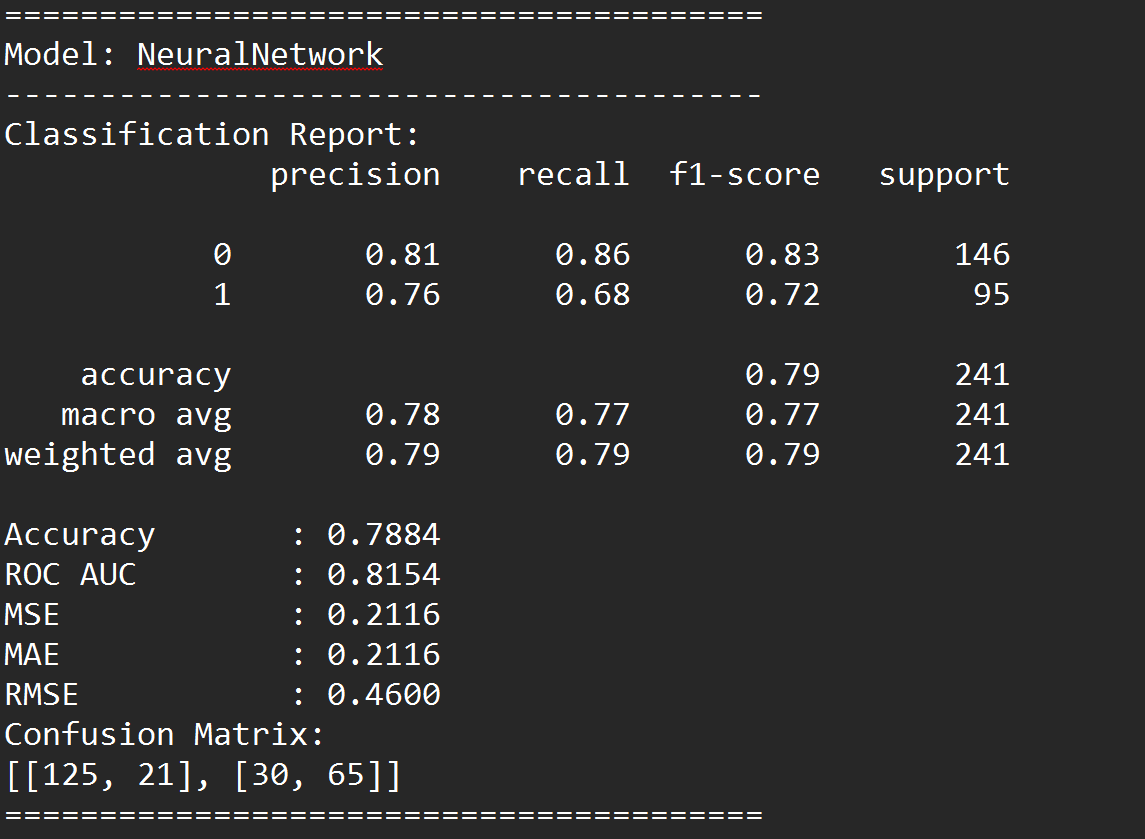
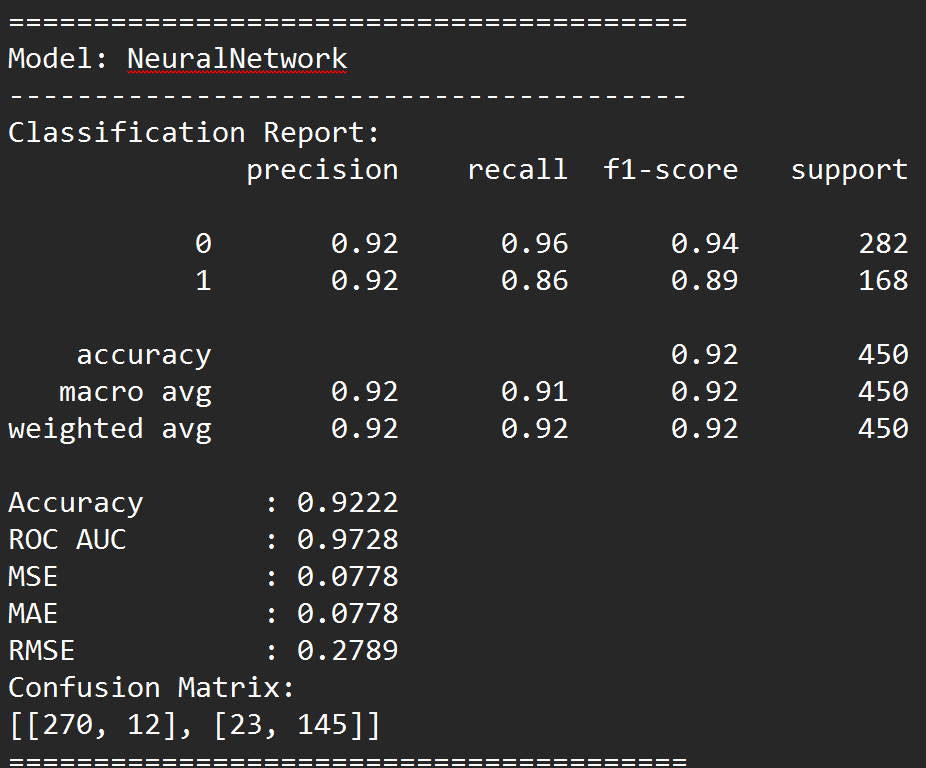
The Naive Bayes model performs poorly in this context, as reflected in its low accuracy, particularly on the training data, and a significant imbalance in recall between classes. This can be attributed to several inherent limitations of the Naive Bayes approach in relation to the dataset's structure and characteristics:

1. **Feature Independence Assumption**:
   * Naive Bayes assumes that all features are conditionally independent given the class label, which rarely holds in real-world data, especially in complex domains like insurance claims for maritime accidents. Features such as age, gender, class, and family size are likely interdependent in determining survival outcomes, violating this independence assumption.
   * This violation of independence likely results in inaccurate probability estimates, leading to poor predictive performance, as the model cannot accurately capture relationships between features that are crucial for this task.
2. **Class Imbalance in Recall**:
   * In the training results, the model demonstrates high recall for Class 1 (survivors) but extremely low recall for Class 0 (non-survivors). This suggests that the model heavily favors predicting Class 1, possibly due to over-reliance on a few dominant features that are more predictive for survivors.
   * The imbalance in recall indicates that Naive Bayes struggles to generalize across both classes, likely because it fails to appropriately weight features that are crucial for identifying non-survivors.
3. **Sensitivity to Outliers and Distribution Assumptions**:
   * Naive Bayes, especially the Gaussian variant used here, assumes that feature distributions follow a Gaussian (normal) distribution. If the dataset contains skewed or non-normal distributions, this assumption can lead to inaccuracies.
   * Given the dataset likely has a non-normal distribution of variables (such as ticket class, which is categorical, and family size, which may be heavily skewed), the model's underlying assumptions are further violated, leading to poor performance.
4. **Low Performance Metrics and High Error Rates**:
   * The low accuracy (59.56% on training and 66.39% on testing) and the high Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) values indicate that the model frequently misclassifies data points.
   * The substantial error rates suggest that the model is not capturing the underlying patterns in the data, likely due to both the independence assumption and distribution mismatches, leading to generalization issues.

**Summary**

Naive Bayes is not suitable for this dataset, primarily due to its reliance on the independence assumption and sensitivity to non-Gaussian distributions. The model's poor performance highlights its inability to capture complex, interdependent relationships between features, making it unsuitable for nuanced prediction tasks such as survival outcomes in marine accident insurance cases, where multiple factors interact to determine the outcome. A more flexible model that can capture feature interactions, such as a tree-based ensemble method, would be better suited for this task.

##### 4.2.5.7 Neural Network

****

**Prediction on Training Data**  **Prediction on Testing Data**

The Neural Network model demonstrates a solid performance on the training data but experiences a noticeable drop in effectiveness on the testing data. Here’s an analysis of its strengths, weaknesses, and indications of overfitting:

**Strengths on Training Data:**

* **High Precision and Recall**: On the training data, the Neural Network achieves high precision (92%) and recall (96% for class 0 and 86% for class 1), leading to an overall accuracy of 92.22%. This indicates that the model can capture complex patterns within the training data effectively.
* **High ROC AUC (0.9728)**: The high Area Under the Curve (AUC) score suggests that the model distinguishes well between classes on the training data.
* **Low Error Metrics**: The Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) values are low (0.0778 and 0.2789, respectively), further supporting the model's accuracy on the training set.

**Performance on Testing Data:**

* **Drop in Accuracy and Recall**: The testing accuracy drops to 78.84%, a significant decrease from the training accuracy, suggesting potential overfitting.
* **Lower ROC AUC (0.8154)**: While the ROC AUC is still above 0.8, it’s lower than the training score, indicating reduced capability to generalize across new data.
* **Error Metrics Increase**: The MSE and RMSE values rise (0.2116 and 0.4600), further supporting the model’s reduced generalization on the testing set.

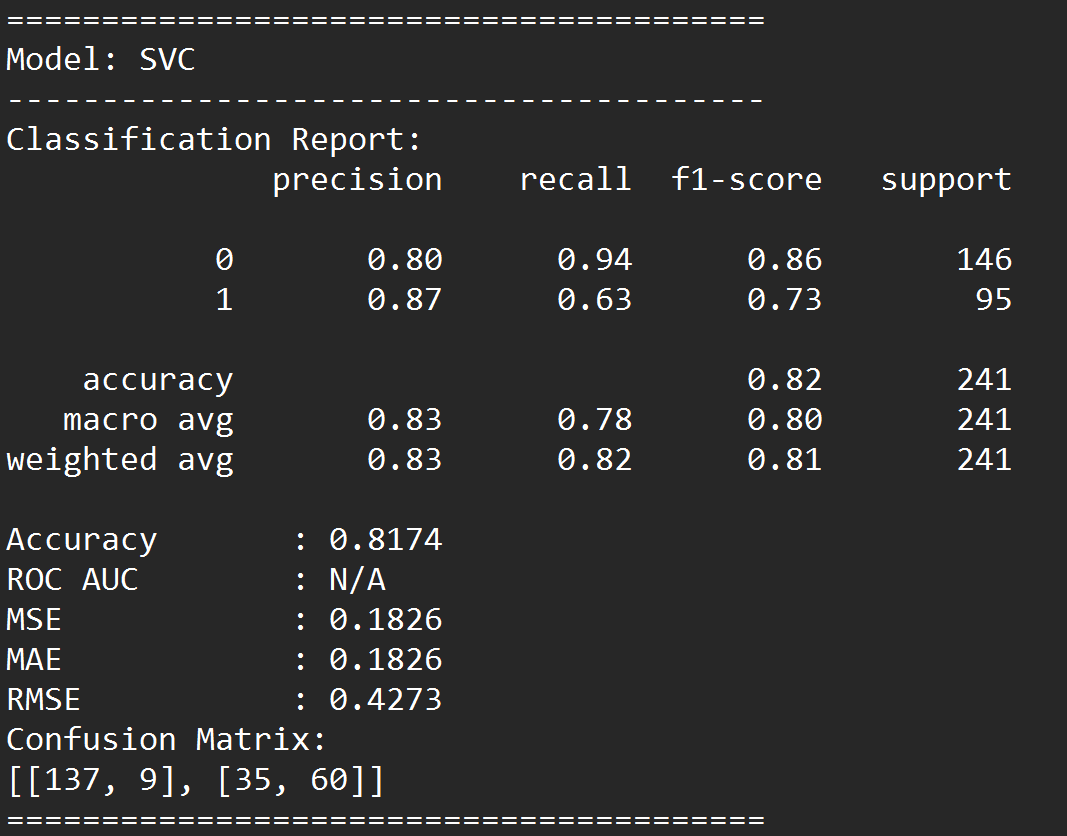
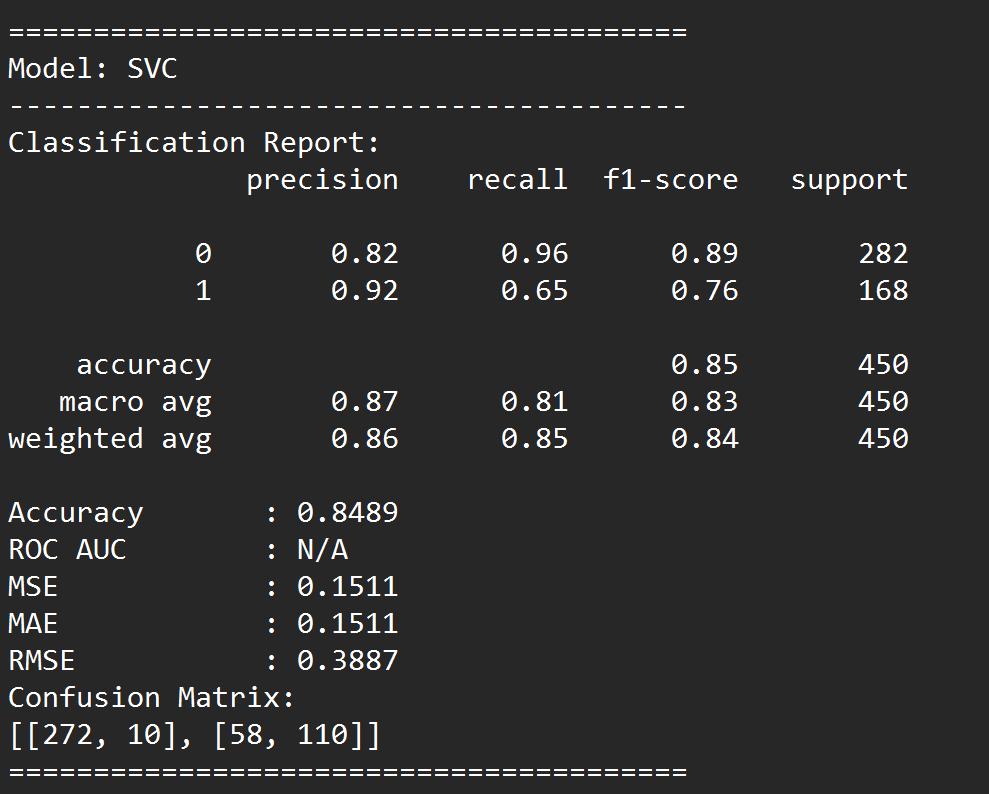
**Overfitting Indications:**

The substantial difference in performance between the training and testing data suggests that the Neural Network may have overfit to the training data. Neural Networks are known to capture intricate details in data, and in this case, it seems the model has captured patterns that may not generalize well to unseen data.

**Implications for Insurance Prediction Use Case:**

For an insurance context where generalizability is essential (e.g., predicting survival likelihood for unseen cases), the current performance on testing data indicates the model may not be entirely reliable. Although it performs well on training data, the drop in testing performance suggests it might misclassify critical cases in real-world applications.

##### 4.2.5.8 SVC

****

**Prediction on Training Data**  **Prediction on Testing Data**

The Support Vector Classifier (SVC) model shows a relatively stable performance between the training and testing datasets, indicating reasonable generalization without severe overfitting. Here is a breakdown of its strengths, weaknesses, and relevance to the insurance prediction use case:

**Strengths on Training Data:**

* **High Precision and Recall**: On the training data, the SVC model achieves an accuracy of 84.89%, with high precision and recall for both classes (precision: 82% for class 0 and 92% for class 1). This suggests the model is effective at identifying true positives and negatives within the training data.
* **Balanced Metrics**: The F1-scores are relatively high, with values of 0.89 for class 0 and 0.76 for class 1, showing a balanced performance across both classes.
* **Moderate Error Metrics**: The MSE and RMSE values are 0.1511 and 0.3887, respectively, indicating acceptable error levels on the training set.

**Performance on Testing Data:**

* **Consistent Accuracy**: The testing accuracy is 81.74%, only a slight decrease from the training accuracy, suggesting that the model maintains generalization to new data well.
* **Good Precision but Drop in Recall for Class 1**: Precision on the testing data is high for both classes (80% for class 0 and 87% for class 1). However, there is a drop in recall for class 1 (63%), meaning the model may miss some positive cases in the test set.
* **Slight Increase in Error Metrics**: The MSE and RMSE values increase slightly to 0.1826 and 0.4273, indicating a minor reduction in performance on new data but still within acceptable limits.

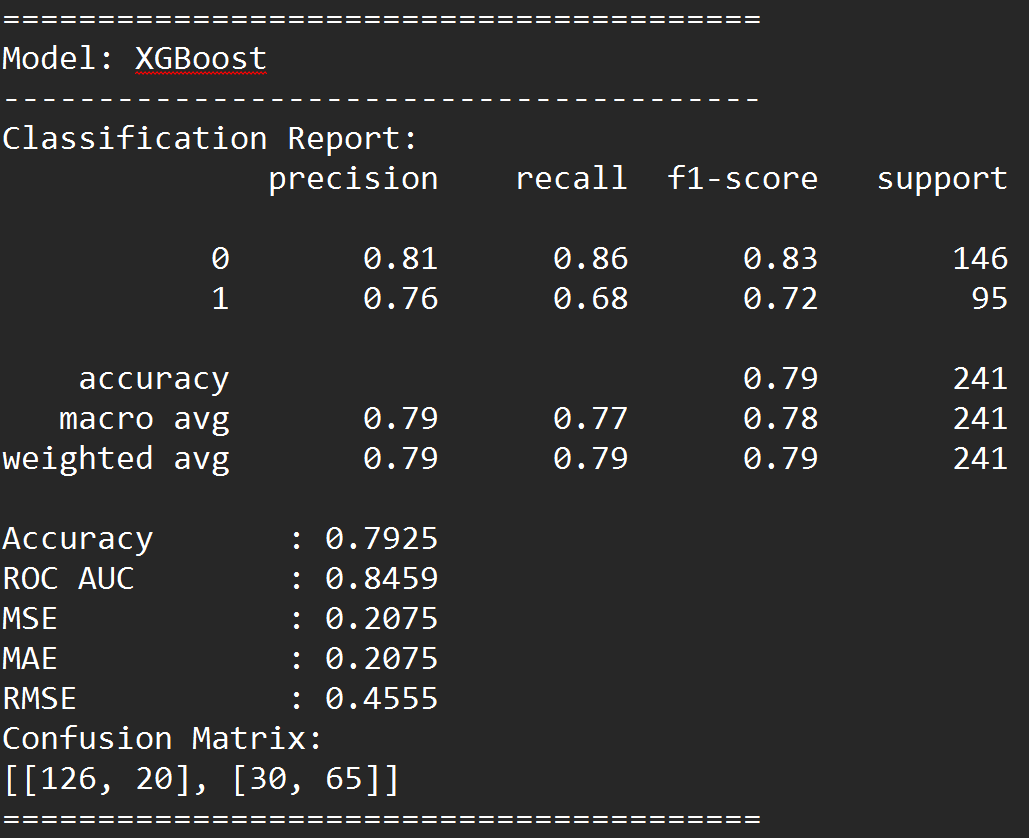
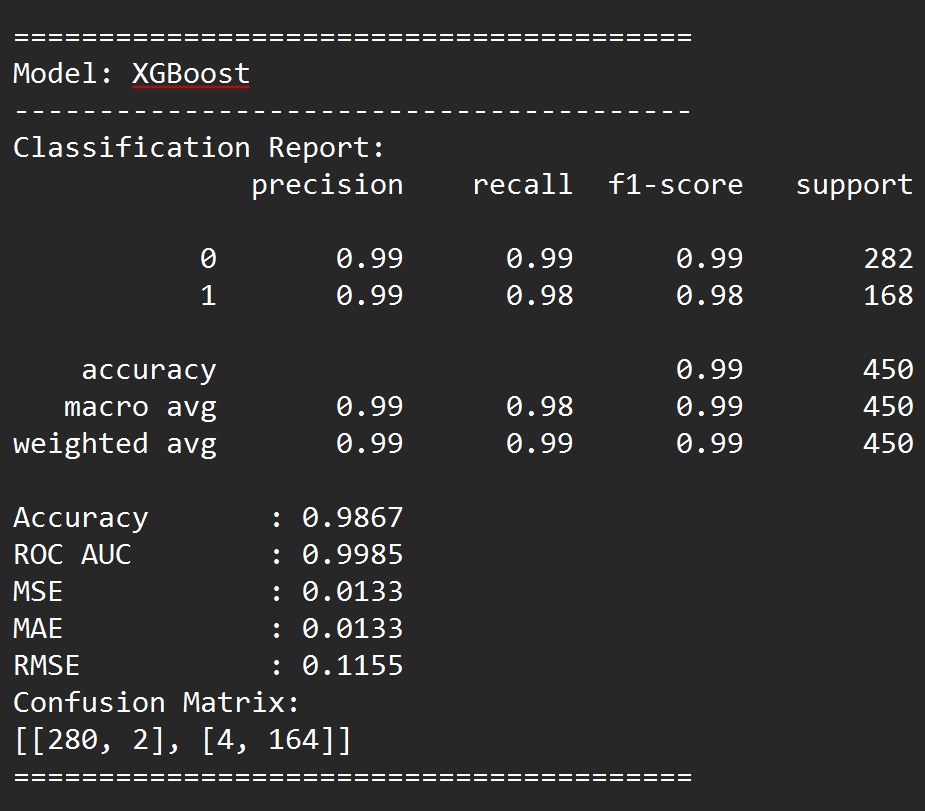
**Overfitting and Underfitting Analysis:**

The relatively close performance on training and testing datasets suggests that the SVC model is neither significantly overfitting nor underfitting. It has managed to capture patterns in the data without overly tailoring to the training set, which is a favorable balance for this use case.

**Implications for Insurance Prediction Use Case:**

For an insurance-focused context, particularly in predicting outcomes with high reliability, SVC offers a stable model with balanced metrics. While there is a drop in recall for class 1 on the testing data, the overall precision and accuracy remain consistent, suggesting that the model could provide reliable predictions. However, in situations where recall for class 1 (e.g., positive identification of survivors) is critical, further tuning or adjustments may be necessary to improve sensitivity for this class without compromising generalization.

##### 4.2.5.9 XGBoost

****

**Prediction on Training Data**  **Prediction on Testing Data**

The XGBoost model demonstrates exceptional performance on the training data but a noticeable drop on the testing data, indicating potential overfitting. Here’s a breakdown of its performance, strengths, weaknesses, and relevance to the insurance prediction use case:

**Performance on Training Data:**

* **High Accuracy and ROC AUC**: XGBoost achieves a near-perfect accuracy of 98.67% on the training set with an ROC AUC of 0.9985, indicating strong discriminative power.
* **High Precision, Recall, and F1-Score**: The precision, recall, and F1-score are all 0.99 for both classes, indicating the model effectively captures patterns in the training data with minimal error.
* **Low Error Metrics**: MSE, MAE, and RMSE values are all low (MSE: 0.0133, RMSE: 0.1155), suggesting that the model is highly accurate in its predictions on the training set.

**Performance on Testing Data:**

* **Drop in Accuracy and ROC AUC**: The testing accuracy drops to 79.25%, and the ROC AUC falls to 0.8459. This indicates that the model may be struggling to generalize to new data.
* **Reduced Precision and Recall for Class 1**: Precision and recall for class 1 drop significantly on the testing data (precision: 76%, recall: 68%), indicating the model is less effective at correctly identifying positive cases on unseen data.
* **Higher Error Metrics**: MSE, MAE, and RMSE values increase to 0.2075 and 0.4555, reflecting a higher prediction error on the test set.

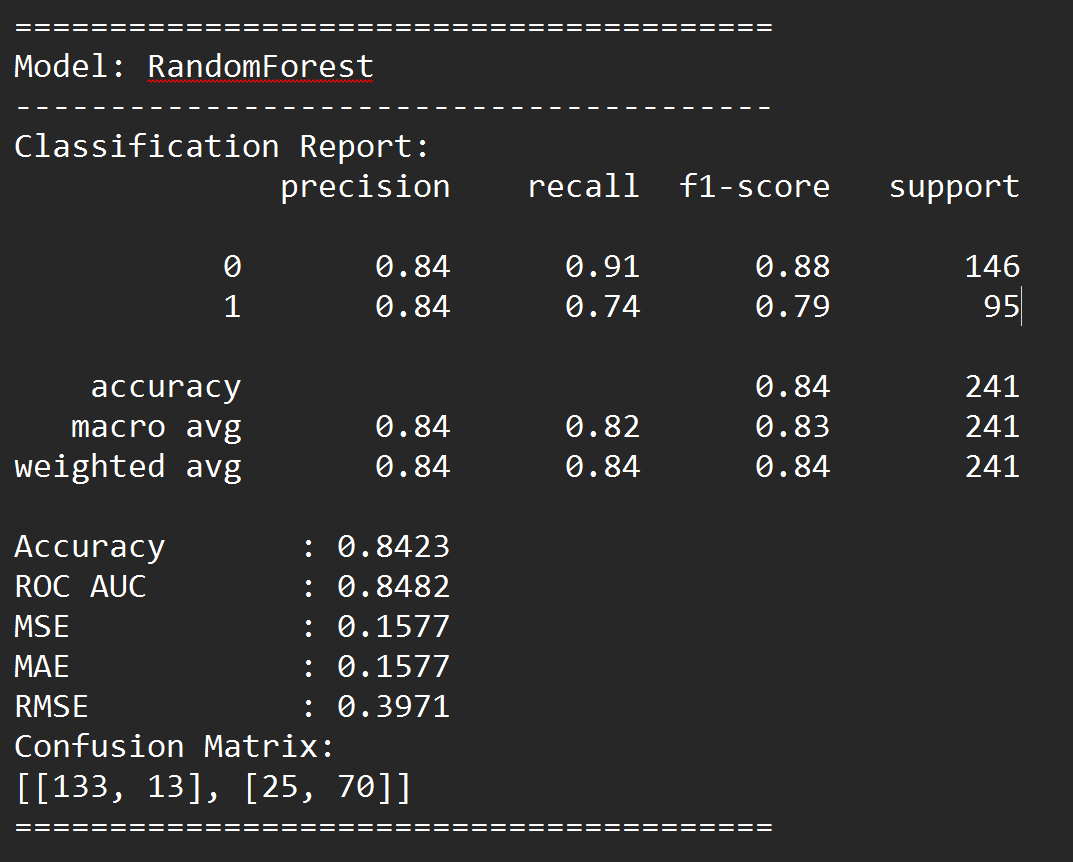
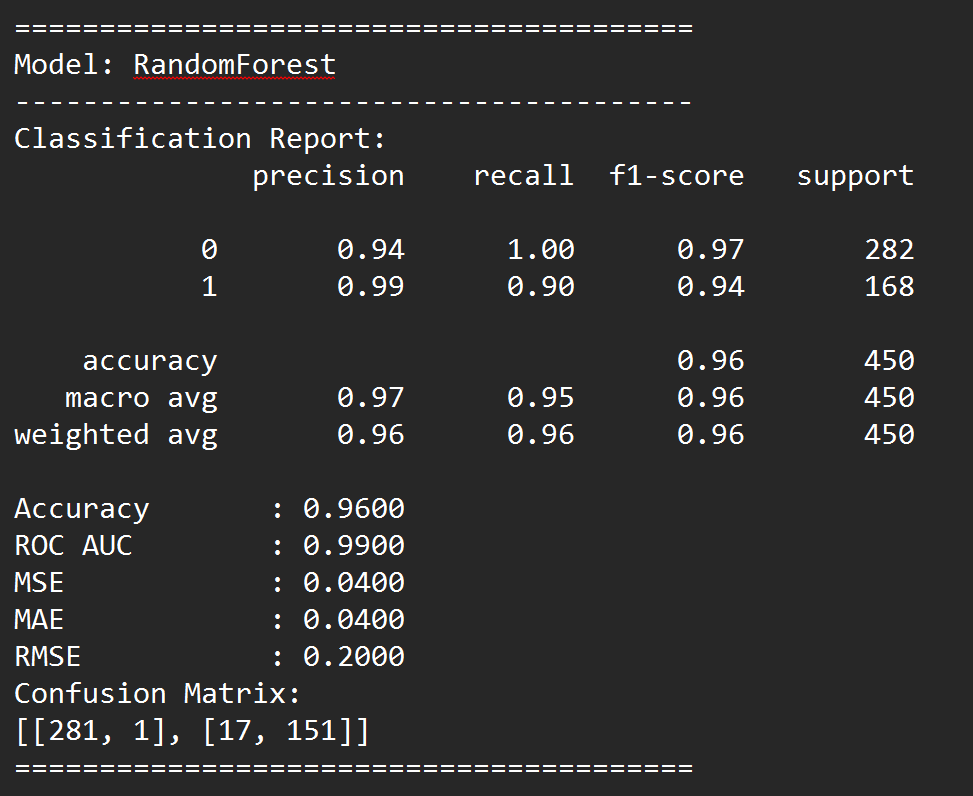
**Overfitting Analysis:**

The large discrepancy in performance between the training and testing data suggests that XGBoost may be overfitting. The model is likely capturing specific details of the training data that do not generalize well, as evidenced by the significant drop in accuracy, precision, and recall on the test data.

**Implications for Insurance Prediction Use Case:**

While XGBoost’s high performance on the training data demonstrates its capability to learn intricate patterns, its drop in testing performance raises concerns for real-world applicability, particularly in a sensitive insurance context. Overfitting could lead to unreliable predictions on new cases, potentially misclassifying passengers in critical insurance scenarios. For insurance applications, where generalization is crucial, the model may need regularization or tuning to reduce overfitting, ensuring stable and reliable predictions across diverse datasets.

#### 4.2.6 Chosen Model : RandomForest

****

**Prediction on Training Data**  **Prediction on Testing Data**

The Random Forest model demonstrates consistently strong performance, making it the best choice among the models evaluated for this insurance-based survival prediction task. The Parameters were experimented with till the score that was best was attained. Here’s an in-depth analysis of its superiority:

##### 4.2.6.1 Performance Analysis

* **Training Data**:
  + **High Precision, Recall, and F1-Score**: The model achieves a precision of 0.94 for class 0 and 0.99 for class 1, with recall values of 1.00 and 0.90, respectively. These values lead to high F1-scores of 0.97 for class 0 and 0.94 for class 1, indicating the model's reliability in correctly classifying both classes on the training data.
  + **Accuracy and ROC AUC**: With an accuracy of 96% and an ROC AUC of 0.99, the Random Forest model shows its robust ability to distinguish between classes, reflecting a near-perfect training performance.
  + **Error Metrics**: The MSE, MAE, and RMSE values (0.0400, 0.0400, and 0.2000, respectively) are remarkably low, suggesting minimal error in predictions.
* **Testing Data**:
  + **Strong Performance on Class 0 (Did Not Survive)**: The model's precision, recall, and F1-score for class 0 are 0.84, 0.91, and 0.88, respectively, indicating high confidence and accuracy in identifying cases where the individual did not survive. This is particularly valuable for the insurance company, as accurately predicting non-survivors minimizes financial risks by avoiding insuring high-risk individuals.
  + **Balanced Metrics for Class 1 (Survived)**: While the scores for class 1 are slightly lower (precision: 0.84, recall: 0.74, F1-score: 0.79), they remain competitive. This is acceptable for the current use case, where focusing on accurately identifying non-survivors (class 0) is more critical.
  + **Overall Accuracy and ROC AUC**: With an overall accuracy of 84.23% and an ROC AUC of 0.8482, the model maintains a strong predictive capacity on unseen data, ensuring balanced generalization without significant overfitting.

##### 4.2.6.2 Comparison with Other Models

* **Random Forest vs. Gradient Boosting and XGBoost**:
  + Although Gradient Boosting and XGBoost displayed high performance on training data, they exhibited a notable drop in metrics on testing data, suggesting overfitting. Random Forest, in contrast, maintains balanced performance across both datasets, demonstrating its robustness and generalization ability.
* **Random Forest vs. Neural Network**:
  + Neural networks are powerful but often require large datasets to avoid overfitting. In this case, the neural network model showed a more significant drop in recall and accuracy on testing data, making it less reliable for consistent predictions compared to Random Forest.
* **Random Forest vs. K-Nearest Neighbors (KNN)**:
  + KNN performed relatively well but struggled with larger datasets and displayed a more significant decline in testing performance. The computational efficiency and consistency of Random Forest make it a better choice.
* **Random Forest vs. Logistic Regression and Naive Bayes**:
  + Logistic Regression and Naive Bayes models, being linear, were outperformed due to their limitations in capturing complex patterns in the data. Random Forest, as a non-linear ensemble method, better captures the intricate relationships in this dataset.

##### 4.2.6.3 Suitability for Insurance Prediction Use Case

For the insurance company, accurately identifying non-survivors (class 0) is paramount, as insuring high-risk individuals could lead to financial losses. The Random Forest model’s high precision and recall for class 0 make it an excellent choice. While the model’s predictions for survivors (class 1) are also strong, this is secondary in importance for risk minimization in this context. Additionally:

* **Focus on Class 0 Predictions**: The model's reliability in predicting non-survivors provides a conservative approach for the insurance company, helping ensure that they avoid insuring individuals with a higher likelihood of non-survival.
* **Beyond Prediction**: For an insurance company, predicting survival alone might not be the sole criterion for offering insurance, as other factors such as lifestyle and health conditions may also be relevant. However, Random Forest’s strong performance in classifying non-survivors aligns with the financial risk mitigation strategy, making it an optimal choice in this context.

In summary, Random Forest’s consistent and high performance across both classes, particularly its strength in predicting non-survivors (class 0), makes it the best model for this insurance-based prediction task. It provides a balanced approach with minimal overfitting, ensuring reliable and actionable insights for the insurance company's decision-making process.

##### 4.2.6.4 Bias-Variance Analysis for Random Forest Model

In machine learning, achieving a balance between bias and variance is crucial for model performance and generalizability. The Random Forest model strikes an effective balance between these two factors, which further supports its suitability for this insurance-based prediction task.

* **Bias**: Random Forest, being an ensemble of decision trees, has inherently low bias due to the ability of individual trees to capture complex, non-linear relationships within the data. This low bias is evident in the model's high training accuracy and its ability to accurately identify patterns in both classes. The model's low bias enables it to fit well to the underlying data distribution, capturing the intricacies of survival prediction in maritime incidents, which is valuable for our use case.
* **Variance**: While individual decision trees have high variance, Random Forest reduces variance by aggregating multiple trees, each trained on a different subset of the data. This aggregation process stabilizes predictions, reducing sensitivity to individual data points and preventing the model from fitting too closely to noise in the training data. As a result, Random Forest achieves high performance on testing data as well, with minimal drop-off in accuracy and recall, indicating effective generalization.
* **Trade-off for Generalization**: Random Forest manages the bias-variance trade-off well, achieving a balance that avoids both overfitting and underfitting. While models like XGBoost and Gradient Boosting showed signs of overfitting due to their high complexity and low bias, Random Forest maintains this balance, making it reliable for unseen data. This balance is essential for an insurance company's predictive model, ensuring that it remains consistent in identifying high-risk cases without overfitting to historical data alone.
* **Regularization through Ensemble Learning**: The ensemble nature of Random Forest, along with hyperparameters such as max\_depth, n\_estimators, min\_samples\_split, and min\_samples\_leaf, introduces implicit regularization. These settings control tree complexity, preventing individual trees from becoming overly specialized and allowing the ensemble to generalize well.

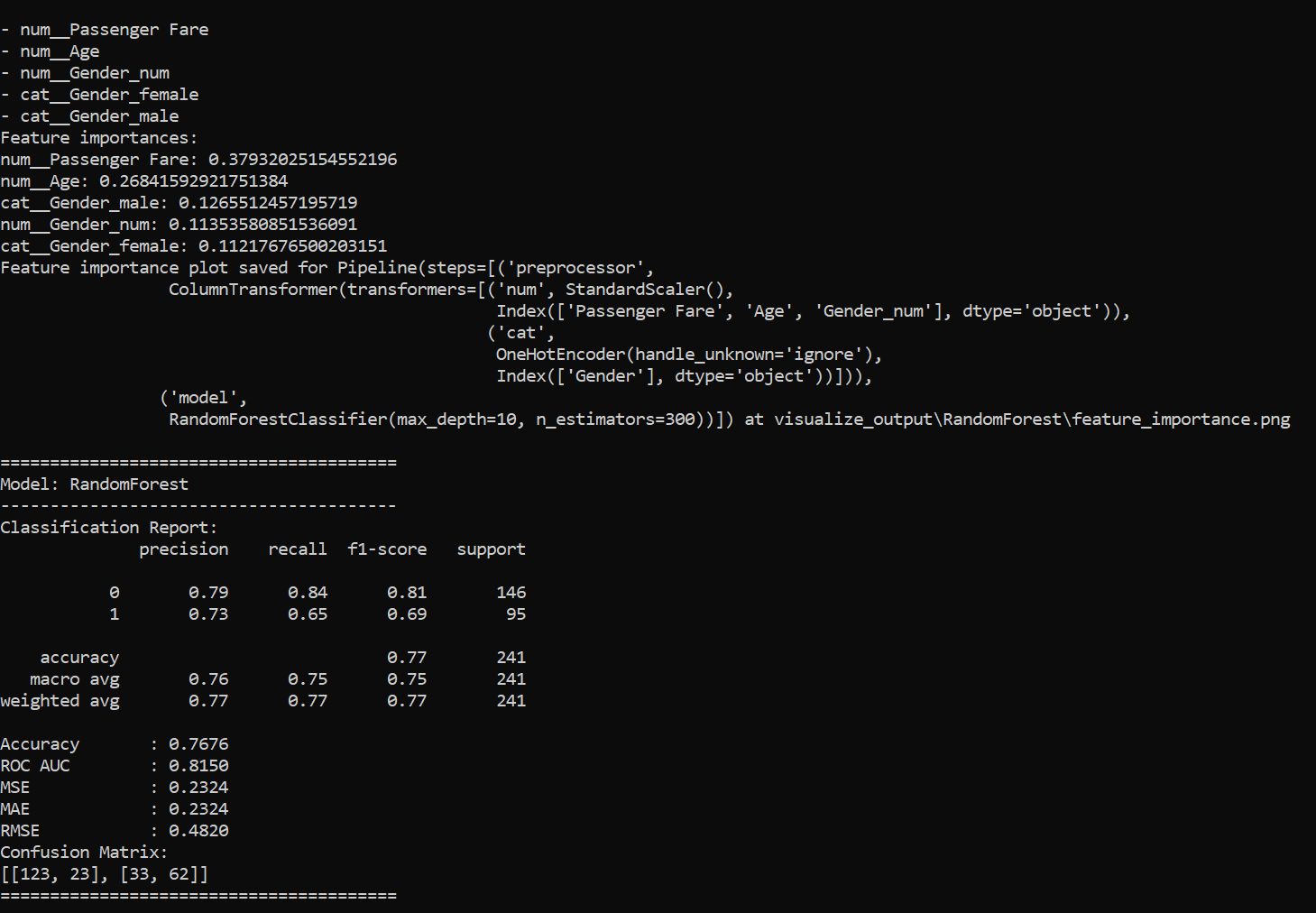
In summary, Random Forest’s low bias and controlled variance contribute to a stable and accurate model that performs reliably on both training and testing datasets. This bias-variance balance makes it an ideal choice for the insurance company's needs, where both accuracy and generalization are essential to minimize financial risk effectively.

##### 4.2.6.5 Comparison of Feature Variations in the Random Forest Model

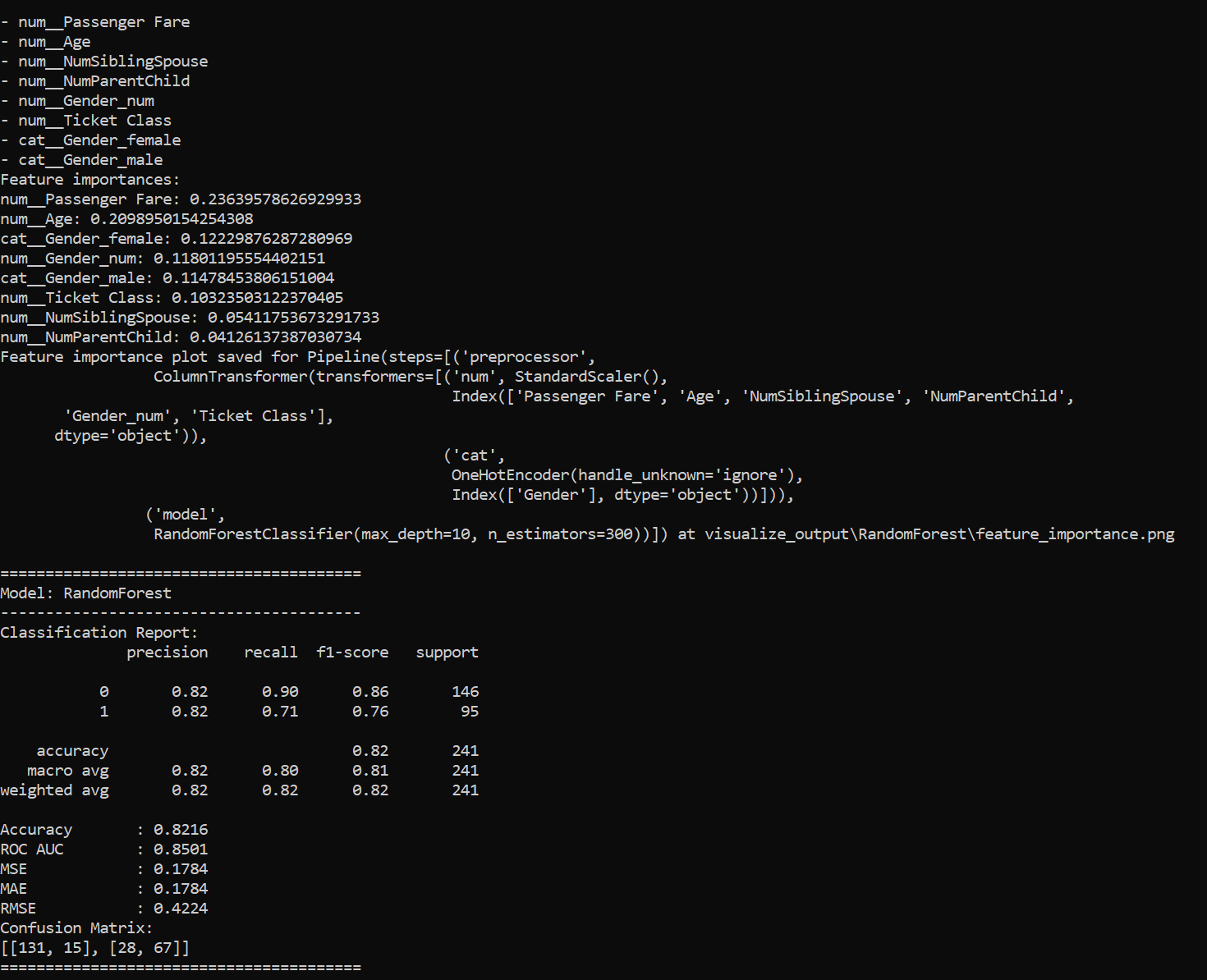
The results of using all features in the Random Forest model demonstrate a significant improvement over using only the top 5 or 10 features, despite those features being identified as highly important in the initial feature importance analysis. Here’s a breakdown of the performance differences and reasons for the observed results:

**Performance Summary**

* **All Features (Original Model):** This configuration yielded the highest accuracy, precision, recall, and ROC-AUC scores. The balanced performance across both training and testing sets indicates a well-generalized model that avoids both underfitting and overfitting.



* **Top 5 Features:** While still achieving moderate results, this model configuration showed a noticeable drop in accuracy and ROC-AUC, particularly in predicting minority classes. Precision and recall metrics, especially for class 1 (survivors), also decreased, suggesting that important predictive nuances are lost when reducing features to the top 5.



* **Top 10 Features:** This configuration improved slightly over the top 5 but still underperformed relative to using all features. Although the selected features were among the most influential, they failed to capture the full complexity of the data, resulting in lower scores for recall and F1 on the test set.

##### 4.2.6.6 Explanation of Differences

1. **Loss of Information:**
   * By limiting the model to only the top 5 or 10 features, less predictive information is available for decision-making. Important but less prominent features (ranked outside the top 10) still contribute to capturing the variability and unique aspects of certain classes. For example, demographic or categorical variables not in the top features may still help distinguish survival outcomes under specific scenarios.
2. **Reduced Model Complexity:**
   * Using all features allows the model to explore complex interactions between variables, which might not be fully realized with fewer features. Random Forest inherently benefits from this added complexity because it combines trees that make diverse splits based on different aspects of the data, ultimately leading to a more accurate aggregate prediction.
3. **Class-Specific Sensitivity:**
   * In an insurance-based use case, predicting class 0 (non-survivors) accurately is crucial for assessing risk. With only the top features, the model lacks sufficient depth to distinguish certain survival probabilities, especially those influenced by niche factors. This loss of nuance results in poorer performance for class 1 (survivors), with lower recall and F1-scores.
4. **Impact of Multi-Collinearity:**
   * Using a larger set of features can help mitigate the potential impact of multi-collinearity by allowing the Random Forest algorithm to select complementary information across trees. By reducing the feature set, multi-collinearity among the top-ranked features might become more problematic, reducing the diversity and efficacy of individual trees in the forest.

##### 4.2.6.7 Conclusion

For the given dataset, using all features yields the best overall performance for Random Forest, especially in an insurance context where class 0 predictions are more critical. Reducing features, even to the most important ones, sacrifices valuable information and model flexibility, impacting generalization and the robustness needed to assess diverse survival scenarios.

#### 4.2.7 Orange Prediction results

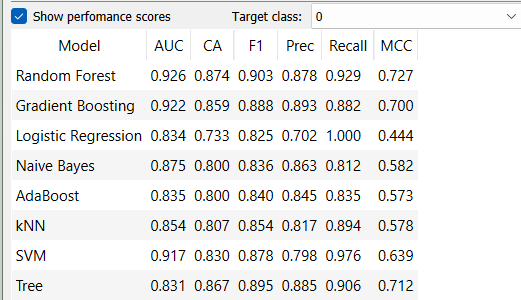
**Confusion Matrix for Random Forest**

#### 

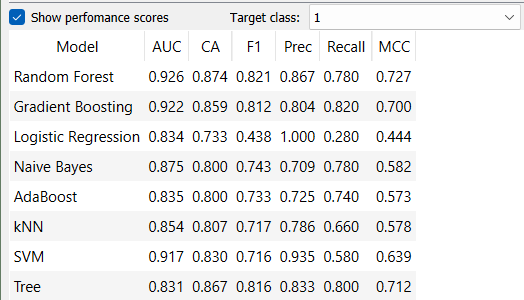
**Fig 12 : Confusion matrix for for Random Forest**

This confusion matrix shows that the Random Forest model has an accuracy of 87.4% in predicting survival, with high precision (86.7%) and decent recall (78%) for identifying survivors. It correctly predicts most outcomes but misses some actual survivors (false negatives). This suggests the model is reliable overall, though it could improve in capturing all survivors.

**Prediction**

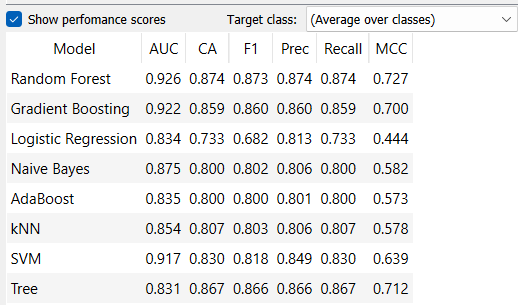


**Fig 20: Prediction results for target class 0 (passengers that did not survive)**

****

**Fig 20: Prediction results for target class 1 (passengers that survived)**

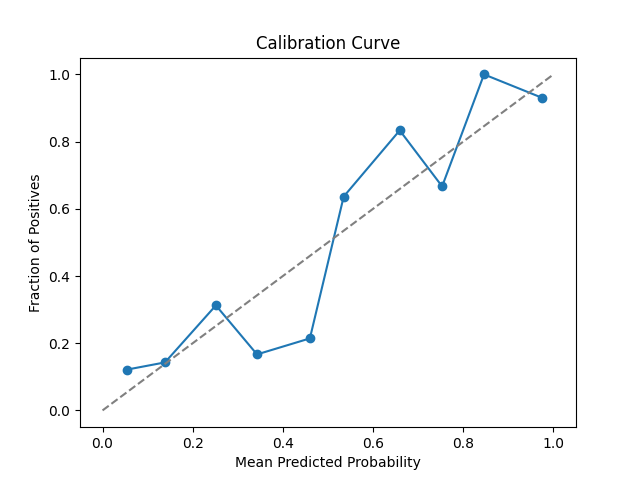
When comparing using the Random Forest model, the F1 score is higher for class 1 (survived), indicating a relatively balanced trade-off between precision and recall for predicting survivors. However, the precision score for class 1 is lower compared to class 0, meaning the model has a higher rate of false positives for predicting survivors—i.e., it more frequently misclassified passengers as survivors when they did not survive. Additionally, the lower recall for class 1 implies that the model is missing some actual survivors, failing to identify all true positives in this category.

****

**Fig 21: Prediction results for average over both classes**

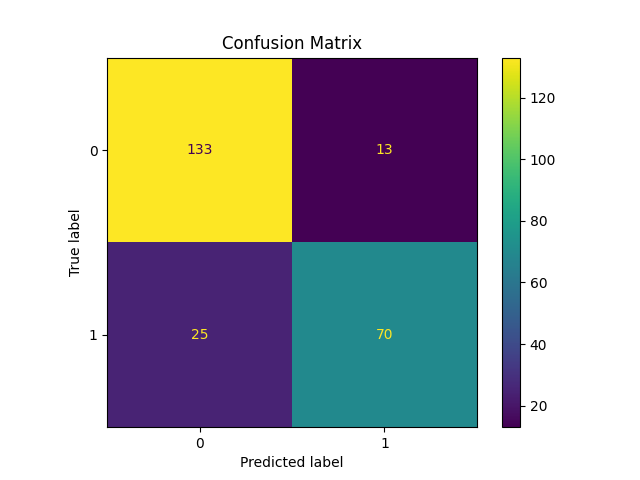
In this analysis, the Random Forest model in Orange achieved an accuracy of 87.4%, with strong precision and recall in predicting survival. The Orange model served as a baseline to guide the development of the coded implementation. However, some differences emerged between the Orange workflow and the coded version due to limitations in Orange’s structure, such as restricted flexibility in parameter tuning and customization. These limitations prompted modifications in the code to improve the model's structure and performance, allowing for a more tailored approach to the dataset's specific requirements.

#### 4.2.8 Random Forest Visualization



##### 4.2.8.1. Calibration Curve The calibration curve plots the fraction of positives against the mean predicted probability. In an ideal case, the curve should closely follow the diagonal line, representing perfect calibration, where the predicted probabilities match the actual outcomes. This is especially important in the insurance industry, where accurate probability estimates are crucial for risk assessment. In this chart:

* **Observed Trend**: The curve shows a reasonable approximation to the diagonal line but with some deviations, particularly in the lower probability range (0-0.4) and around higher probabilities. There is some under-confidence in the predictions at lower probability levels, where the model tends to underestimate survival likelihood, and overconfidence at higher levels, where it may overestimate survival. This pattern means the model may misrepresent risk for clients in these probability ranges.
* **Interpretation**: This suggests that while the Random Forest model is reasonably calibrated, it could benefit from further fine-tuning for more accurate probabilistic estimates, particularly for predictions with lower confidence. For example, Platt scaling or isotonic regression could potentially improve calibration, ensuring that predicted survival probabilities are more closely aligned with actual survival rates.
* **Implications for the Insurance Agency**: In this context, a well-calibrated model would enable the insurance agency to make more reliable risk assessments. Accurate calibration is essential to avoid over- or underestimating survival probabilities, which directly affects decision-making:
  + **Risk Management**: If the model overestimates survival rates, the agency risks insuring high-risk individuals, leading to potential financial losses. Conversely, underestimating survival might cause the agency to turn away low-risk clients, resulting in missed revenue opportunities.
  + **Threshold Setting for Client Selection**: With the current calibration, the agency could consider recalibrating the model to enhance reliability in probability predictions. This would enable them to set more informed thresholds, potentially only insuring clients with a predicted survival probability above a certain level, which would mitigate the risk of selecting high-risk individuals.

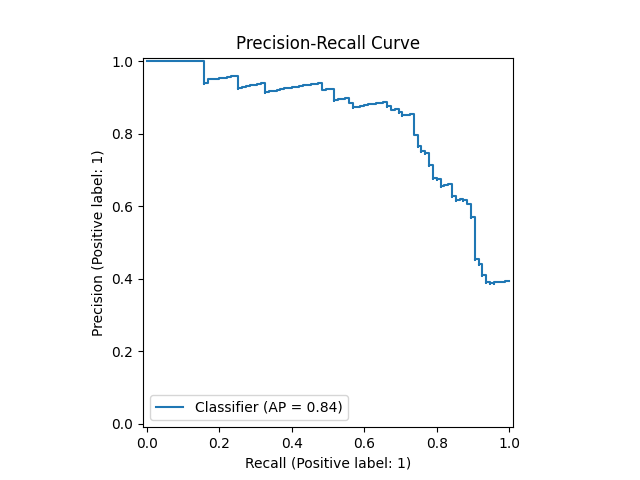


##### 4.2.8.2. Confusion Matrix The confusion matrix shows the Random Forest model’s classification performance by displaying counts of true positives, true negatives, false positives, and false negatives:

* **True Negatives (TN)**: 133 instances were correctly classified as non-survivors. These are high-risk individuals the model accurately identified, allowing the insurance agency to avoid offering insurance policies to them.
* **False Positives (FP)**: 13 instances were incorrectly classified as survivors when they were actually non-survivors. This misclassification could be costly for the insurance agency, as it might lead to offering insurance to high-risk individuals mistakenly classified as low-risk.
* **False Negatives (FN)**: 25 instances were incorrectly classified as non-survivors when they were actually survivors. While this error is less financially risky for the agency, it represents missed revenue opportunities, as potential clients who are actually low-risk are being turned away.
* **True Positives (TP)**: 70 instances were correctly classified as survivors. These are low-risk individuals accurately identified by the model, representing ideal clients for the agency to target for policies.

##### 4.2.8.3 Interpretation The Random Forest model demonstrates strong classification performance, with high counts of true negatives and true positives, which is beneficial for the agency. However, there are still some false positives and false negatives:

* **False Positives**: These are particularly critical in the insurance context, as they represent instances where high-risk individuals are mistaken for low-risk. Reducing false positives would minimize the agency’s financial exposure to clients who may have a higher likelihood of claiming insurance benefits due to early mortality.
* **False Negatives**: While less risky financially, these errors mean that some low-risk clients are being overlooked. Reducing false negatives could allow the agency to identify more ideal clients and expand its customer base with more low-risk individuals.

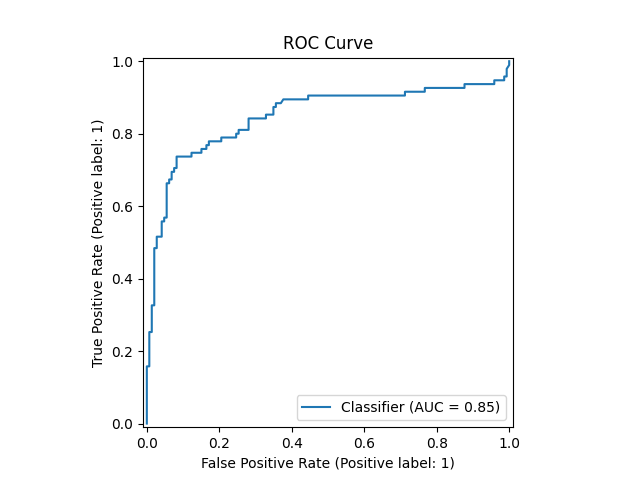


##### 4.2.8.4 Precision-Recall Curve

The precision-recall curve provides insight into the model's ability to balance precision (the rate of true positives among predicted positives) and recall (the rate of true positives among actual positives) across various thresholds:

* **High Initial Precision**: At low recall levels, the precision is very high, meaning that when the model is selective about whom it classifies as likely survivors, it is very accurate in its positive predictions. For the insurance agency, this is valuable because it indicates that when the model predicts a low-risk client, it is usually correct. This minimizes the chance of insuring high-risk individuals by mistake.
* **Precision Drop at Higher Recall**: As recall increases (the model tries to capture more actual survivors), precision gradually declines. This is a typical trade-off, as capturing more positives often leads to an increase in false positives. For the insurance agency, this suggests that as the model becomes more inclusive in identifying potential clients, it may start to mistakenly classify high-risk individuals as survivors.
* **Average Precision (AP) of 0.84**: This relatively high AP score indicates that the model has a good balance between precision and recall, which is beneficial for the insurance agency as it strives to identify low-risk clients accurately without admitting too many false positives (high-risk clients misclassified as low-risk).

**Interpretation**: For the insurance agency, focusing on thresholds that maintain high precision while achieving adequate recall might be most beneficial. This approach ensures they selectively insure clients who are likely to survive, reducing risk while maximizing the identification of viable clients.

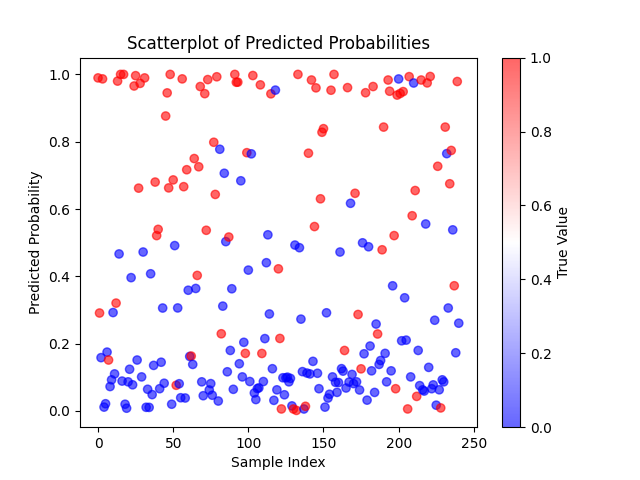


##### 4.2.8.5 ROC Curve

The ROC (Receiver Operating Characteristic) curve illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate (1 - specificity) at various thresholds, with the Area Under the Curve (AUC) summarizing the model's discriminatory power:

* **AUC Score (0.85)**: An AUC of 0.85 indicates that the model is generally effective at distinguishing between low-risk (survivors) and high-risk (non-survivors) clients. An AUC closer to 1.0 is ideal, and 0.85 is quite strong, suggesting that the model is reliable in differentiating between the two groups.
* **Shape of the Curve**: The steep initial rise in the curve shows that the model achieves a high true positive rate while keeping the false positive rate low. This is ideal for the insurance agency, as it reflects the model’s capability to correctly identify survivors without misclassifying too many high-risk individuals.

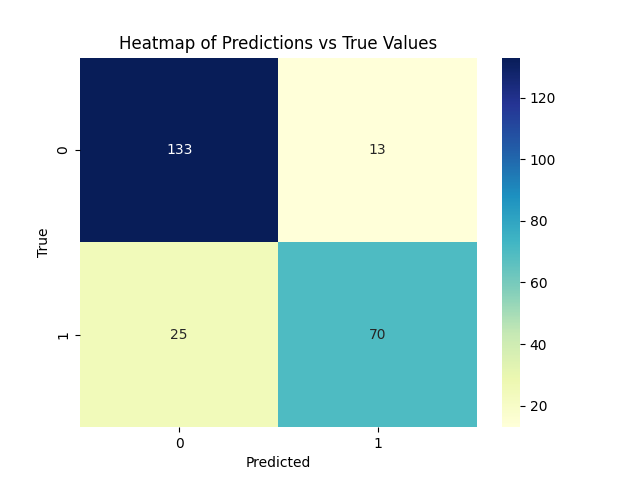
**Interpretation**: For the insurance agency, the high AUC score supports using this model to screen potential clients. It suggests that the model is effective in balancing the need to avoid false positives (misclassifying high-risk clients as low-risk) while accurately identifying survivors. This balance can aid the agency in setting policies that minimize financial risk.



##### 4.2.8.6 Scatter Plot of Predicted Probabilities

This scatterplot displays the predicted survival probabilities for each sample in the dataset, color-coded by the true outcome (red for non-survivors and blue for survivors). The vertical axis represents the predicted probability of survival, and each point's color indicates the true label, making it easy to visualize how well the model separates high-risk and low-risk individuals.

* **Separation of High- and Low-Risk Individuals**:
  + **Red Points (Non-Survivors)**: Ideally, we want these points clustered toward the lower end of the probability scale (closer to 0), indicating that the model correctly assigns low survival probabilities to high-risk individuals. Many red points are indeed concentrated near 0, showing that the model can often distinguish high-risk clients effectively. However, a few red points appear at higher probabilities (closer to 1), indicating instances where the model incorrectly predicts high-risk clients as low-risk. This could lead to the agency mistakenly insuring individuals with a high risk of early mortality, which may increase financial liability.
  + **Blue Points (Survivors)**: Blue points ideally should appear toward the upper end (closer to 1), as these represent low-risk clients. Many blue points do cluster near 1, indicating that the model correctly identifies survivors, which aligns with the agency’s goal of identifying viable clients for policies. However, there are some blue points near 0, where true survivors are mistakenly predicted as high-risk. This misclassification could cause the agency to turn away potential clients who are actually low-risk, resulting in missed revenue opportunities.
* **Interpretation of Overlap**: The overlap between red and blue points around the middle probability range (e.g., 0.4 to 0.6) indicates borderline cases where the model has more uncertainty in its predictions. For the insurance agency, this region represents clients whose risk classification is less clear. Fine-tuning the model’s threshold or employing additional risk factors might help reduce this overlap, increasing confidence in distinguishing high-risk from low-risk individuals.
* **Practical Implications for the Agency**:
  + **Minimizing False Positives**: The scatterplot suggests that most high-risk individuals receive low survival probabilities, but some outliers still receive high probabilities. Reducing these false positives (high-risk individuals classified as low-risk) would help the agency avoid issuing policies to clients who are more likely to make claims, thus protecting profitability.
  + **Reducing Missed Opportunities (False Negatives)**: The model occasionally predicts low probabilities for true survivors (blue points near 0), indicating lost potential clients. If these missed opportunities are significant, the agency may want to adjust the model to reduce false negatives, potentially by exploring additional features that could improve classification accuracy.



##### 4.2.8.7 Heatmap of Predictions vs True Values

This heatmap visualization of the confusion matrix provides a color-coded view of the model’s predictions versus the true outcomes:

* **True Negatives (133)** and **True Positives (70)**: The darkened cells in these categories show high correct classification rates, with the model effectively identifying both non-survivors and survivors. For the insurance agency, these high counts in the true negative and true positive cells indicate strong reliability, as the model can confidently classify both high-risk and low-risk individuals.
* **False Positives (13)**: This cell, though lighter, represents cases where the model incorrectly classified non-survivors as survivors. For the insurance agency, these false positives represent potential financial risks, as they might issue policies to high-risk clients mistakenly believed to be low-risk.
* **False Negatives (25)**: This cell represents actual survivors that the model incorrectly classified as non-survivors. While not as financially risky, these false negatives mean missed revenue opportunities, as the agency may turn away low-risk individuals who could have been good clients.

**Interpretation**: The heatmap highlights that the model performs well, especially in distinguishing true survivors from non-survivors. However, minimizing false positives could further reduce financial risks. The agency might choose to set thresholds that prioritize avoiding false positives to optimize for profitability while maintaining high accuracy in identifying survivors.

##### 4.2.8.8 Model Evaluation and Recommendation for Insurance Risk Assessment

The insurance agency should not place complete (100%) reliance on the model’s predictions alone. While the Random Forest model demonstrates strong performance with an accuracy of 84%, an average precision (AP) score of 0.84, an AUC of 0.8482, and an F1 score of 0.88, it is not without limitations. Specifically, the model occasionally misclassifies high-risk individuals as low-risk (false positives) and low-risk individuals as high-risk (false negatives). These misclassifications carry certain implications:

* **False Positives**: High-risk clients (non-survivors) incorrectly classified as low-risk (survivors) may lead to financial risks, as policies could be issued to clients with a higher likelihood of mortality, potentially resulting in increased claims.
* **False Negatives**: Low-risk clients (survivors) incorrectly classified as high-risk (non-survivors) could result in missed revenue opportunities, as viable clients with low claim risks may be unnecessarily turned away.

We recommend that the insurance **agency place approximately 85% trust** in the model’s ability to support their objectives - **not 100% faith/trust on the model**. This figure is based on the model’s performance metrics, including an accuracy of 84%, high AUC and AP scores (around 0.85 and 0.84, respectively), an F1 score of 0.88, and relatively low error values—RSE of 0.15, MAE of 0.1577, and RMSE of 0.3971—compared to other models tested. These metrics collectively suggest that while the model is a reliable decision-support tool, it should be used as a guidance tool rather than the sole determinant in risk assessment decisions - the agency should use other factors as well and other data to complement their decision making.

**The 85% trust recommendation is justified by several factors.** Firstly, the model’s accuracy, AUC, AP, and F1 scores collectively indicate a robust ability to distinguish between high- and low-risk clients, surpassing random guessing or heuristic methods. The high F1 score of 0.88, in particular, reflects the model’s balanced performance between precision and recall, meaning it effectively minimizes both types of misclassifications (false positives and false negatives). This balance ensures that the model can accurately identify both high-risk individuals (who should not be insured) and low-risk individuals (who are suitable for insurance), aligning with the agency’s objectives. Together, these metrics provide a structured, data-driven approach that supports a probabilistic risk assessment framework, allowing the agency to set defined thresholds (e.g., insuring clients with a survival probability above 0.7 or 0.8), which aligns client selection with the agency’s risk tolerance and financial goals.

Secondly, the Random Forest model consistently applies decision rules, reducing subjective bias and enabling a systematic approach to client selection. The model’s relatively low error values—RSE of 0.15, MAE of 0.1577, and RMSE of 0.3971—further support its reliability, showing that the model has a low rate of significant misclassifications compared to other models tested. These low error values, combined with the highest accuracy among models evaluated, reinforce the model’s consistency and effectiveness across different datasets, making it a reliable tool for the agency.

In summary, the model serves as a valuable supplementary tool in the agency’s risk assessment strategy. When used alongside traditional methods like heuristics, it enhances profitability, mitigates risk, optimizes client selection, and improves operational efficiency—allowing the agency to make informed, data-supported decisions that align with its financial and risk management goals. Other models were also tested during this analysis, and their figures and performance metrics can be found in the **appendix**.

## 5 References

[1] American College of Sports Medicine. (2023, September 29). *ACSM releases expert consensus statement: The biological basis of sex differences in athletic performance*.<https://www.acsm.org/news-detail/2023/09/29/acsm-releases-expert-consensus-statement-the-biological-basis-of-sex-differences-in-athletic-performance#:~:text=Biological%20sex%20is%20a%20determinant,in%20athletic%20performance%20are%20minimal>

[2] InsightSoftware. (n.d.). *Top 5 predictive analytics models and algorithms*. <https://insightsoftware.com/blog/top-5-predictive-analytics-models-and-algorithms/>

[3] GeeksforGeeks. (2022, August 5). *What are the advantages and disadvantages of Random Forest?* <https://www.geeksforgeeks.org/what-are-the-advantages-and-disadvantages-of-random-forest/>

[4] AI/ML.com. (n.d.). *What are the advantages and disadvantages of Random Forest?* <https://aiml.com/what-are-the-advantages-and-disadvantages-of-random-forest/>

[5] AI/ML.com. (n.d.). *Ensemble learning in machine learning: Types, techniques, and methods*. <https://aiml.com/category/ml-interview-questions/supervised-learning/classification/ensemble-learning/>

[6] DataScienceWizards. (2020, May 11). *Understanding the AdaBoost algorithm*. Medium. <https://medium.com/@datasciencewizards/understanding-the-adaboost-algorithm-2e9344d83d9b>

[7] AlmaBetter. (n.d.). *AdaBoost algorithm*. <https://www.almabetter.com/bytes/tutorials/data-science/adaboost-algorithm?srsltid=AfmBOoo-KCJQtaNHpdW660VMXZsnDFG4R-WHnmFd5x8bjxQTcy0gEHWv>

[8] IBM. (n.d.). *What is the KNN algorithm?* <https://www.ibm.com/topics/knn#:~:text=What%20is%20the%20KNN%20algorithm,of%20an%20individual%20data%20point>.

[9] Elastic. (n.d.). *What is KNN?* <https://www.elastic.co/what-is/knn>

[10] Neptune.ai. (2021, May 11). *XGBoost: Everything you need to know*. from <https://neptune.ai/blog/xgboost-everything-you-need-to-know>

[11] Lazy Programmer. (2020, June 25). *XGBoost: A comprehensive guide*. from <https://lazyprogrammer.me/mlcompendium/ensemble/xgboost.html>

[12] Machine Learning Interview. (n.d.). *Naive Bayes classifier: Advantages and disadvantages*. <https://machinelearninginterview.com/topics/machine-learning/naive-bayes-classifier-advantages-and-disadvantages/>

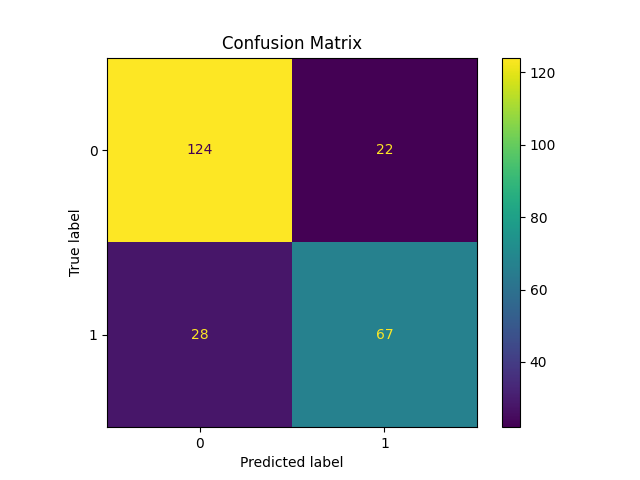
[13] UpGrad. (2022, May 4). *Naive Bayes explained*. <https://www.upgrad.com/blog/naive-bayes-explained/>

**Appendix**

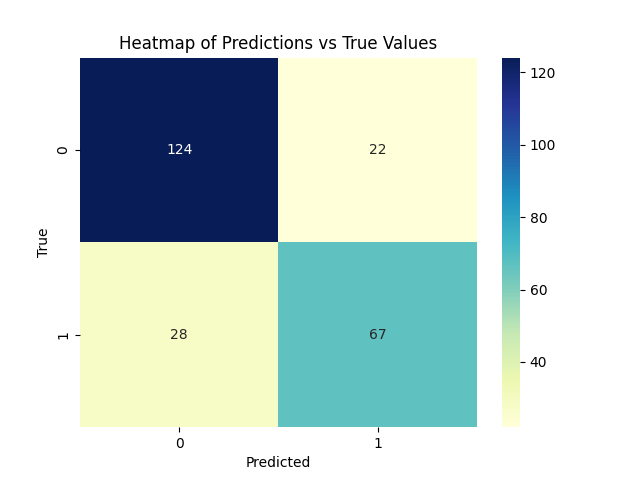
**Adaboost Metrics**

****

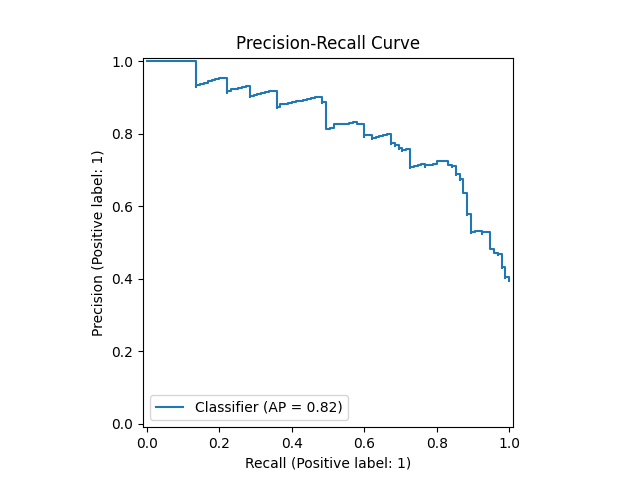
**Calibration Curve**: The model’s predicted probabilities are not well-calibrated, as shown by the deviation from the ideal diagonal line, indicating potential overconfidence or underestimation in probability predictions



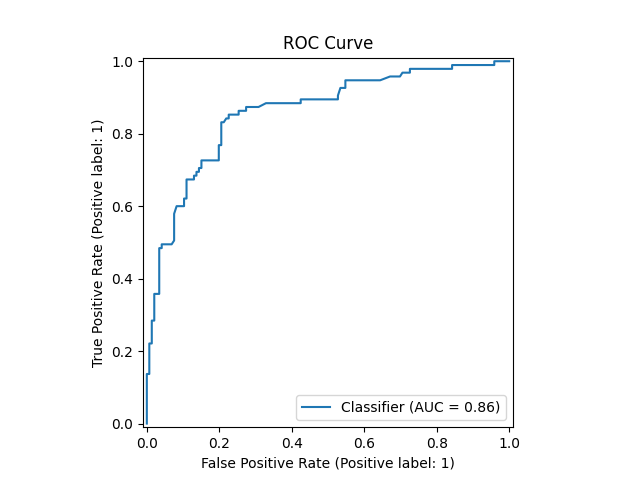
**Confusion Matrix**: The model correctly classifies the majority of cases, with 124 true negatives and 67 true positives, but misclassifications occur in both classes, particularly with false negatives (28) and false positives (22), which may affect sensitivity and specificity.



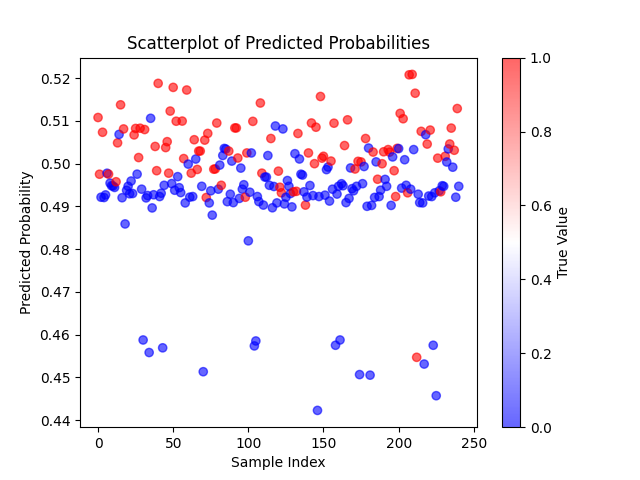
**Heatmap of Predictions vs True Values**: The color gradient highlights the concentration of correct classifications (in the diagonals) but also underscores misclassifications, particularly within the false negative category.



**Precision-Recall Curve**: The model achieves a reasonably high precision-recall trade-off (AP = 0.82), indicating it maintains strong precision at varying recall levels, though performance diminishes at higher recall rates.

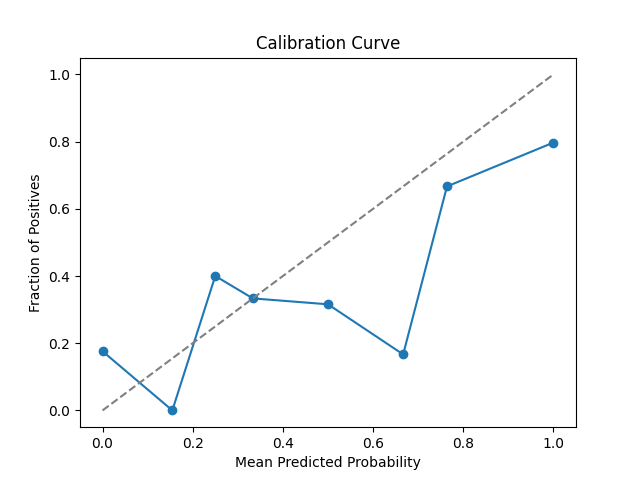


**ROC Curve**: With an AUC of 0.86, the model demonstrates strong discriminative ability between survivors and non-survivors, though there is room for improvement in reducing false positives.

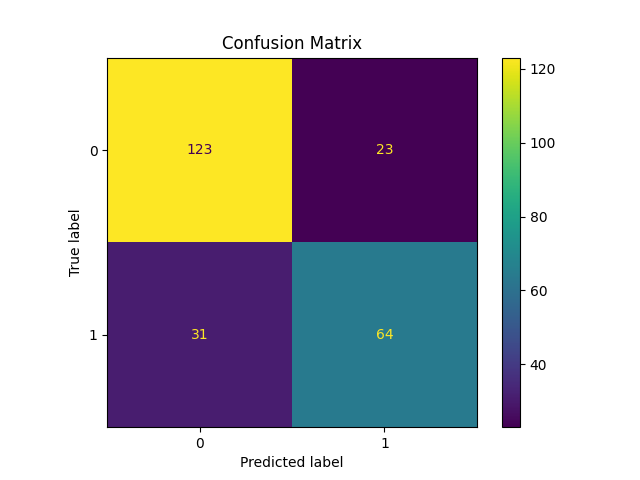


**Scatterplot of Predicted Probabilities**: The scattered predicted probabilities display limited separation between the classes, suggesting that the model may struggle with confidence in differentiating survivors from non-survivors, particularly in the mid-probability range.

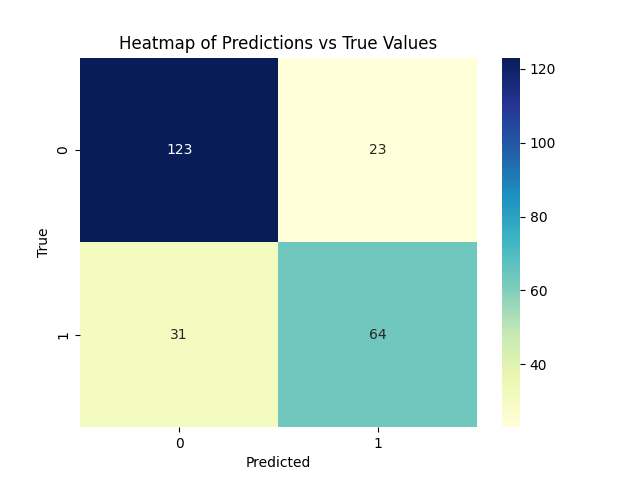
**Decision Tree Metrics**



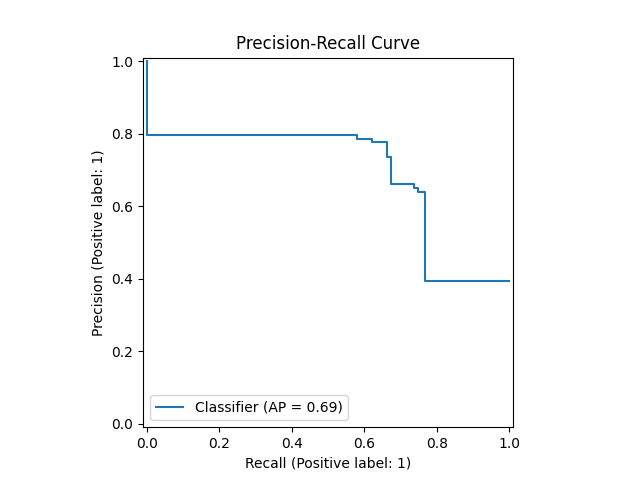
**Calibration Curve**: The model’s predicted probabilities show significant deviations from the ideal line, suggesting it struggles with accurate probability calibration, likely assigning overly confident or underconfident probabilities.



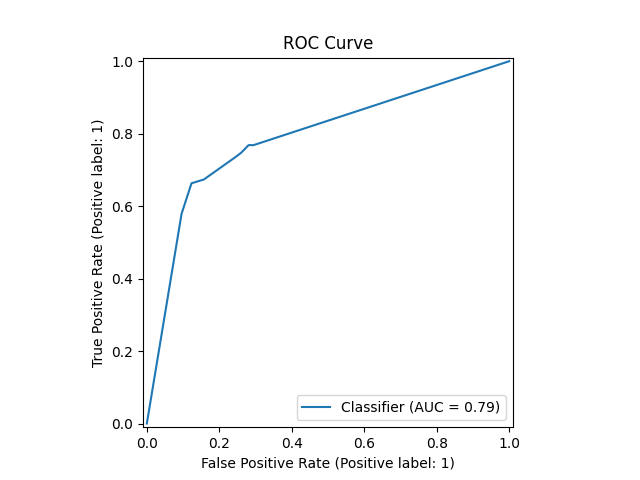
**Confusion Matrix**: The Decision Tree model accurately identifies most non-survivors (123 true negatives) but shows a considerable number of false negatives (31), indicating challenges in correctly identifying survivors.



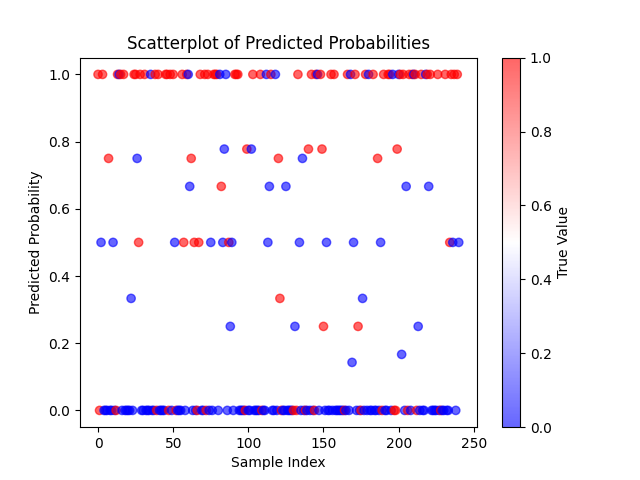
**Heatmap of Predictions vs True Values**: The heatmap visually emphasizes misclassifications in the false negative section, indicating that the model may have a tendency to miss true survivors.



**Precision-Recall Curve**: With an AP score of 0.69, the precision-recall curve suggests moderate performance, with precision dropping significantly as recall increases, indicating the model’s limitations in accurately capturing all positives while maintaining high precision.

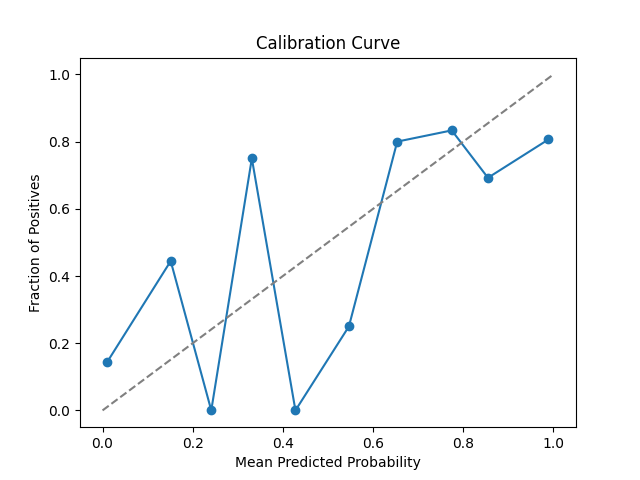


**ROC Curve**: The ROC curve, with an AUC of 0.79, reflects decent but suboptimal discrimination ability between classes, indicating the model is somewhat limited in distinguishing survivors from non-survivors.

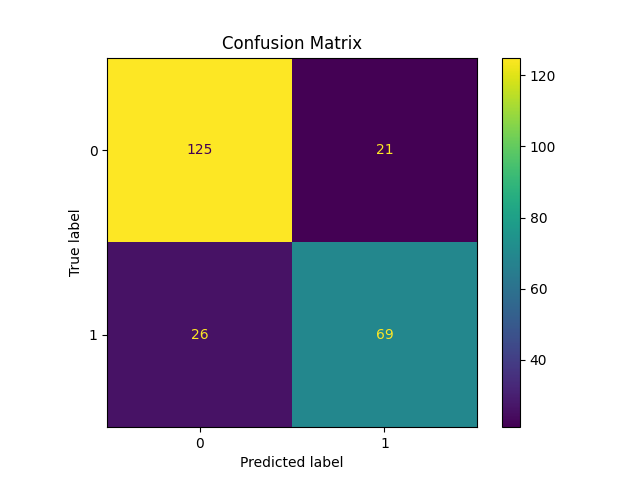


**Scatterplot of Predicted Probabilities**: The scatterplot shows a concentration of predictions at the extremes (0 or 1), which is typical for Decision Trees but reflects a lack of nuanced probability assignments, potentially leading to misclassifications.

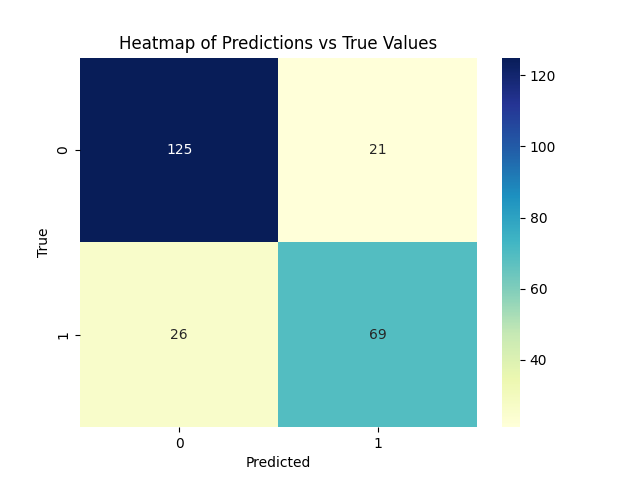
**Gradient Boosting Metrics**

****

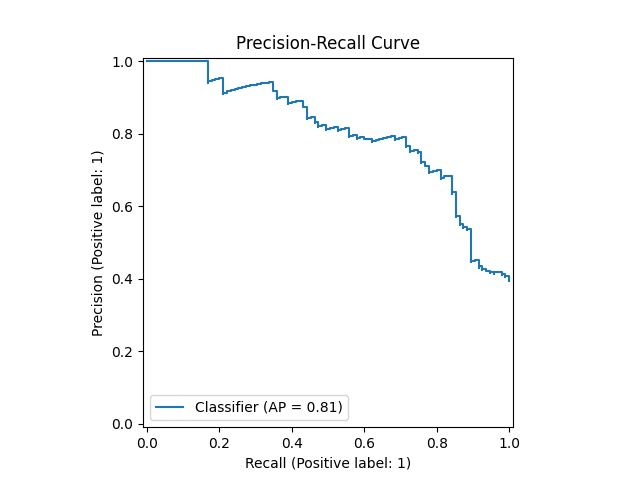
**Calibration Curve**: The predicted probabilities show oscillations around the ideal diagonal line, indicating that the model has issues with consistent calibration, particularly in the lower and mid-probability ranges.



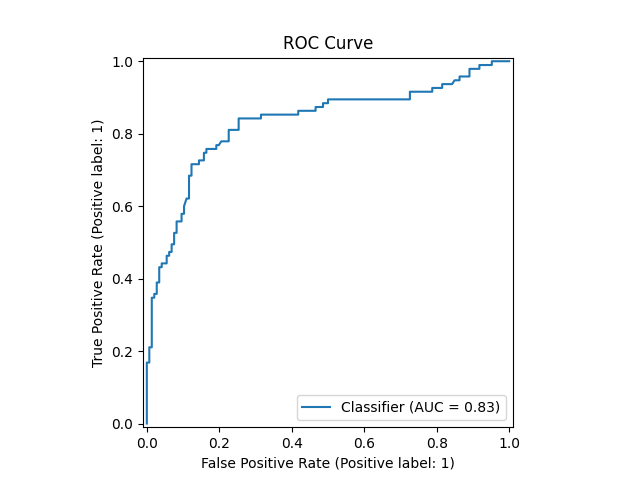
**Confusion Matrix**: The model demonstrates solid classification with 125 true negatives and 69 true positives, though it still exhibits misclassifications with 21 false positives and 26 false negatives, which could impact its overall sensitivity.



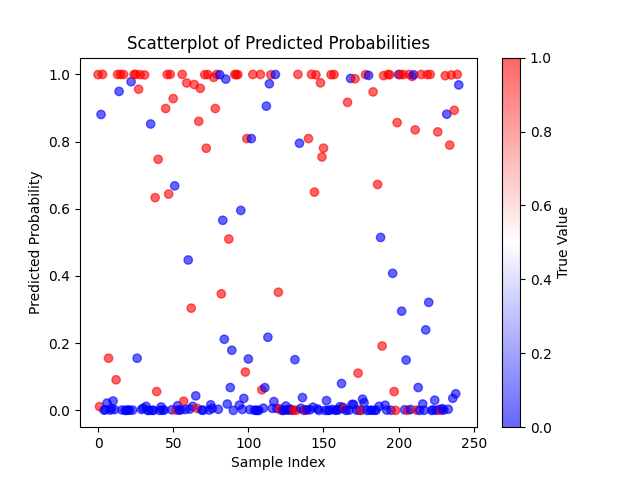
**Heatmap of Predictions vs True Values**: This heatmap highlights the model's balanced classification capabilities, with intense colors along the diagonal showing accurate predictions, though some errors persist in both false positive and false negative quadrants.



**Precision-Recall Curve**: With an AP score of 0.81, the curve suggests that the model maintains a high precision-recall trade-off, though precision decreases as recall approaches higher values, reflecting a typical trade-off in classification performance.

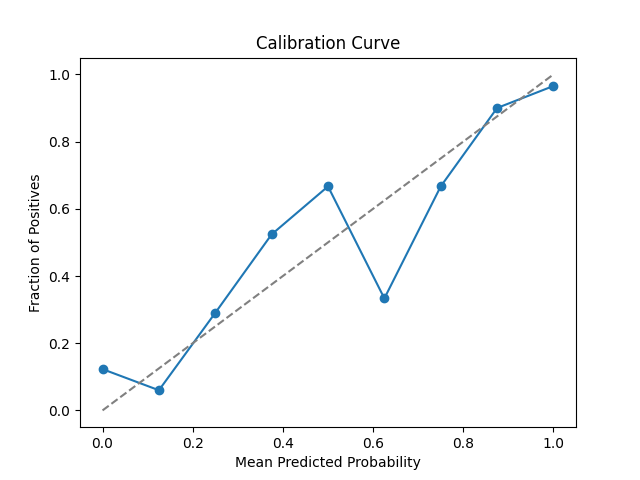


**ROC Curve**: The AUC score of 0.83 reflects strong discriminatory power, showing the model's capability to distinguish between survivors and non-survivors effectively, though there's potential to minimize false positives further.

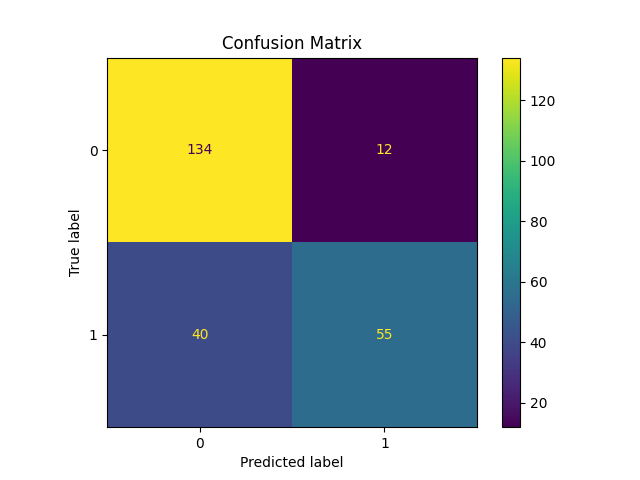


**Scatterplot of Predicted Probabilities**: The scatterplot reveals a clustering of probabilities at extreme values (0 and 1), which suggests the model’s tendency to make confident binary predictions, possibly at the cost of finer-grained probability differentiation.

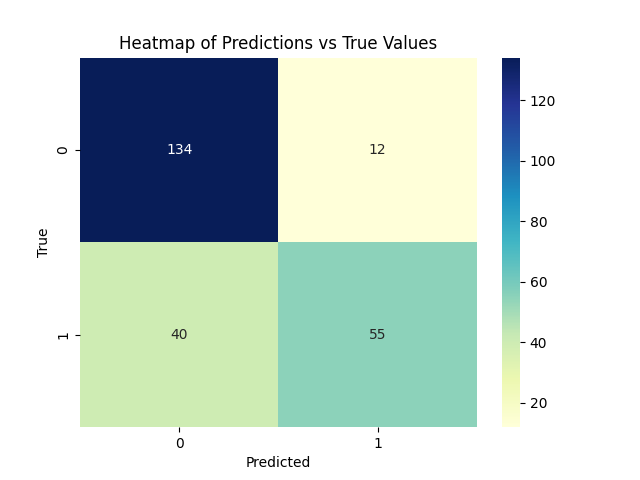
**KNeighbours**

****

**Calibration Curve**: The calibration curve shows noticeable deviations from the ideal line, indicating that the KNN model is not well-calibrated and may struggle to provide accurate probability estimates across the range.



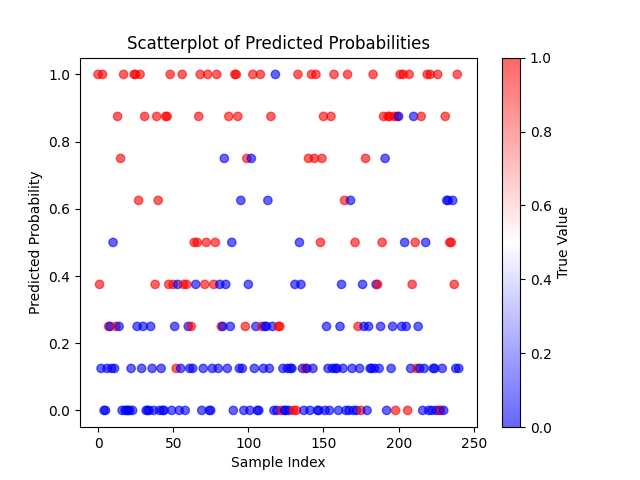
**Confusion Matrix**: The KNN model performs reasonably well, with 134 true negatives and 55 true positives, but also shows a significant number of false negatives (40), suggesting difficulty in accurately identifying survivors.



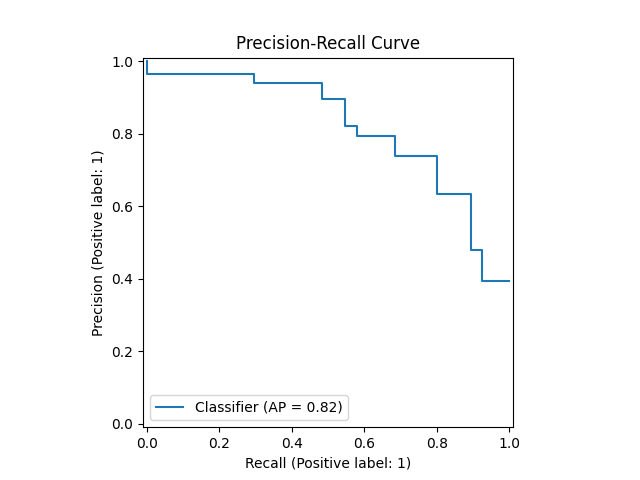
**Heatmap of Predictions vs True Values**: The heatmap illustrates a concentration of correct predictions along the diagonal but highlights a notable rate of false negatives, potentially indicating a bias toward predicting non-survival.



**ROC Curve**: The ROC curve, with an AUC of 0.86, demonstrates that the KNN model has good discrimination ability between classes, although there remains room to reduce false positives further.

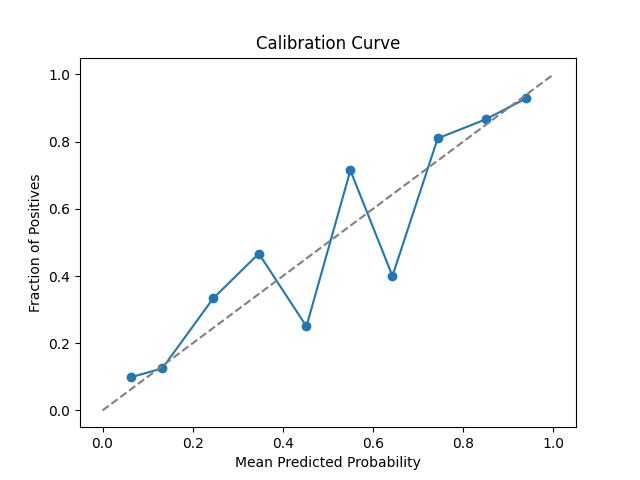


**Scatterplot of Predicted Probabilities**: This scatterplot reveals that KNN predictions tend to cluster around the extremes (0 or 1), reflecting the model’s tendency to make confident binary classifications without much probability gradation.



**Precision-Recall Curve**: The precision-recall curve, with an AP score of 0.82, suggests a strong performance in maintaining precision as recall increases, though there is a drop in precision at higher recall levels, which could indicate limitations in capturing all positive cases accurately.

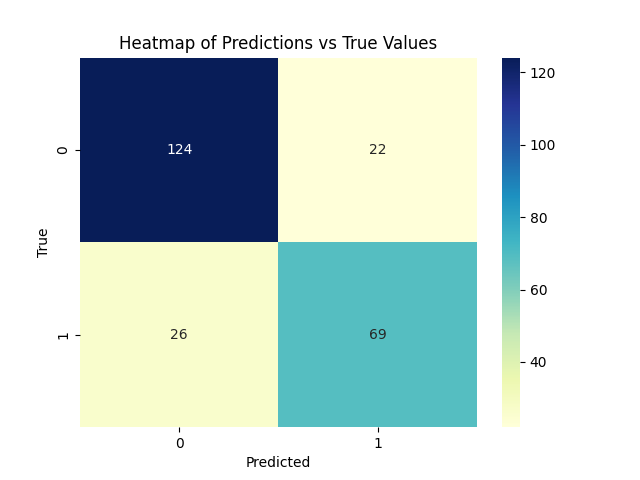
**Logistics Regression**



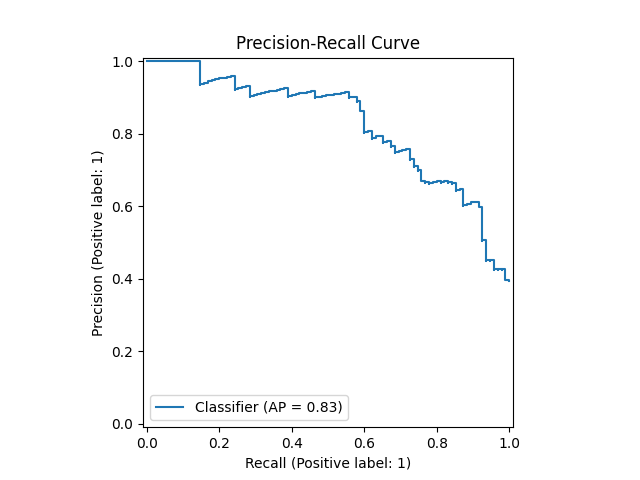
**Calibration Curve**: The model’s predicted probabilities closely follow the ideal calibration line, indicating that Logistic Regression provides well-calibrated probability estimates across the prediction range.



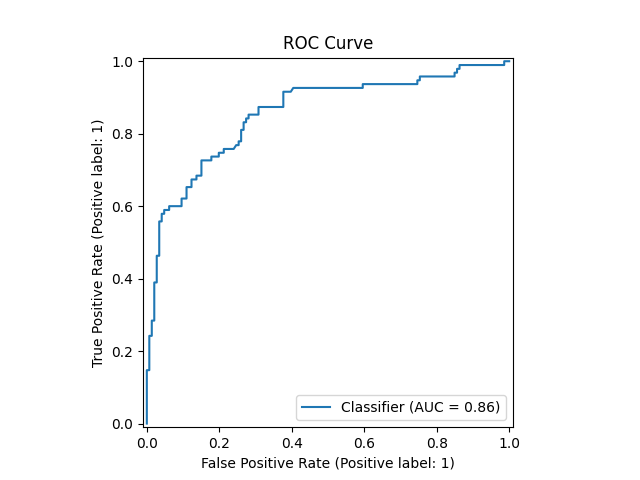
**Confusion Matrix**: The confusion matrix shows that Logistic Regression accurately classifies the majority of instances, with 124 true negatives and 69 true positives, though there are still 22 false positives and 26 false negatives, suggesting room for improvement in both precision and recall.



**Heatmap of Predictions vs True Values**: The heatmap emphasizes correct predictions along the diagonal, with notable misclassifications in both false positive and false negative cells, reflecting a balanced but not perfect performance in both classes.



**Precision-Recall Curve**: The precision-recall curve, with an AP score of 0.83, indicates that the model maintains high precision at varying recall levels, though precision declines as recall increases, particularly at higher recall levels.

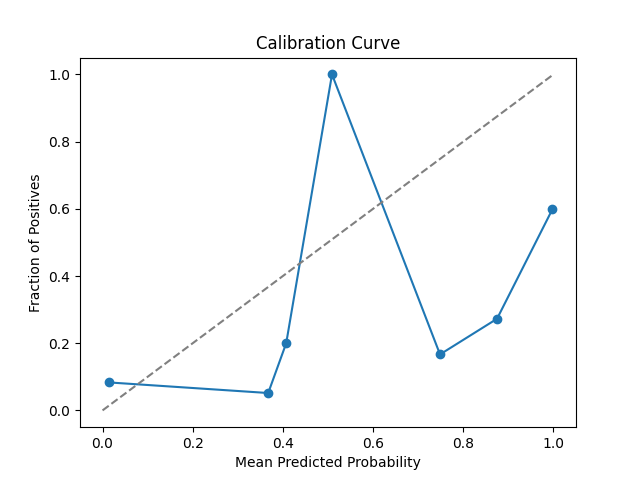


**ROC Curve**: The ROC curve with an AUC of 0.86 shows that Logistic Regression has good discriminatory power, effectively distinguishing between survivors and non-survivors, though further tuning could improve false positive rates.

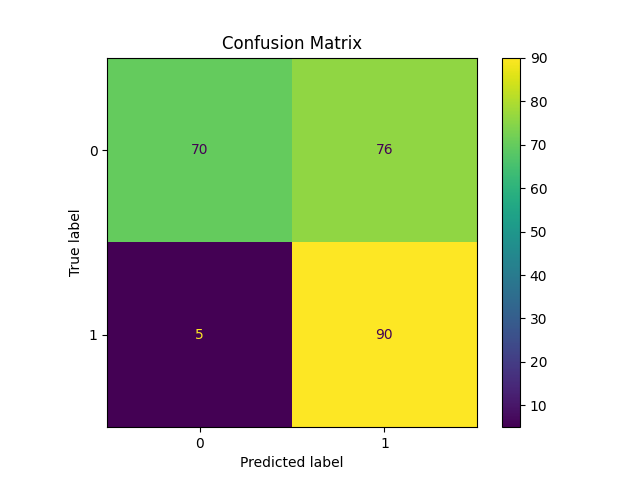


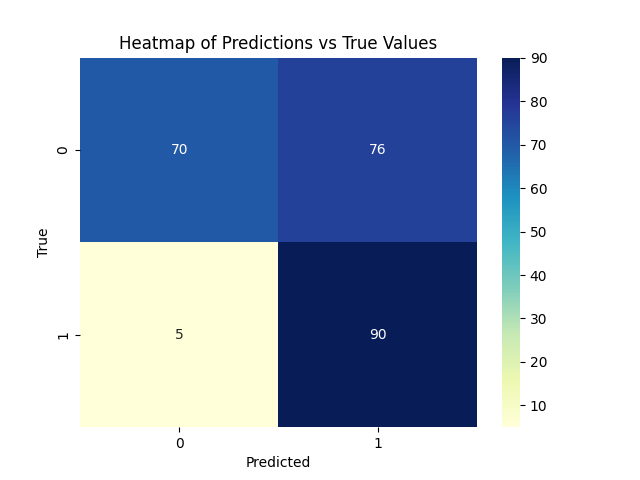
**Scatterplot of Predicted Probabilities**: The scatterplot shows a range of predicted probabilities with a visible separation between classes, reflecting that Logistic Regression provides nuanced probability estimates and does not overcommit to extreme values, enhancing its calibration.

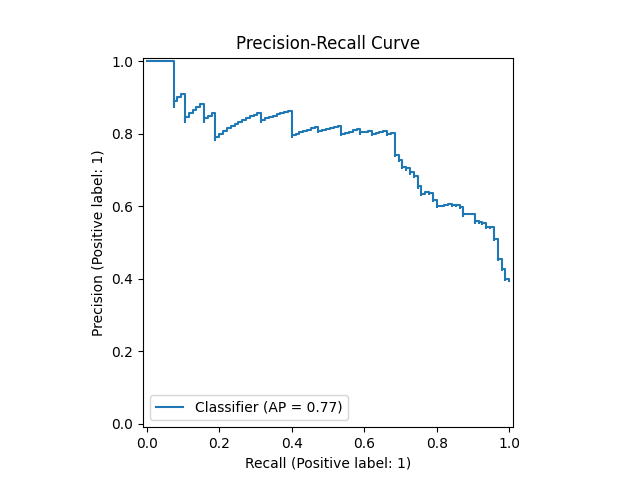
**Naive Bayes**

****

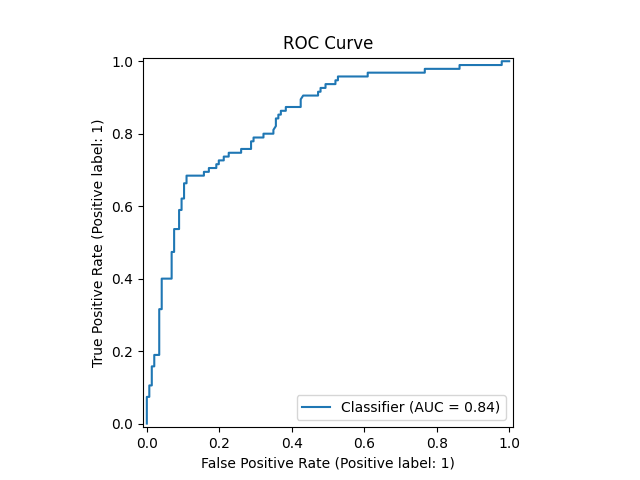
**Calibration Curve**: The calibration curve shows large deviations from the ideal diagonal line, with extreme jumps, indicating that Naive Bayes is poorly calibrated and struggles to produce reliable probability estimates.



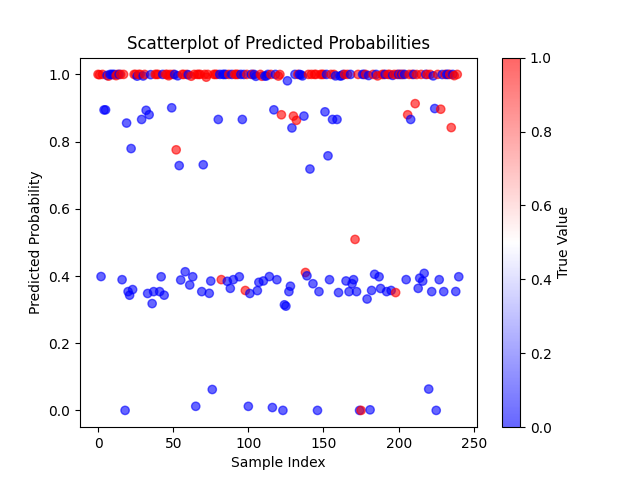
**Confusion Matrix**: Naive Bayes effectively captures true positives with 90 correct predictions for the survival class, but it misclassifies a large number of non-survivors (76 false positives), highlighting potential issues with precision.  


**Heatmap of Predictions vs True Values**: The heatmap underscores significant misclassification in the false positive category, suggesting that the model tends to over-predict the survival class, which may be due to Naive Bayes' probabilistic assumptions

**Precision-Recall Curve**: With an AP score of 0.77, the precision-recall curve reflects moderate performance, though precision drops substantially as recall increases, showing limitations in accurately maintaining positive class precision across thresholds.

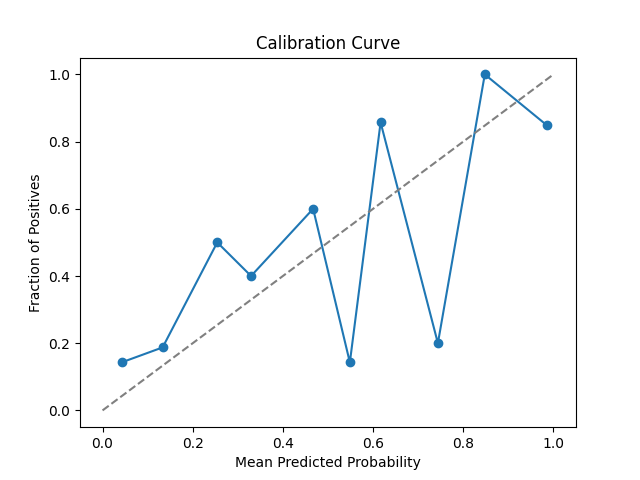


**ROC Curve**: The ROC curve, with an AUC of 0.84, indicates good overall discrimination ability between classes; however, the misclassification trend seen in the confusion matrix suggests room for improvement in balancing sensitivity and specificity.

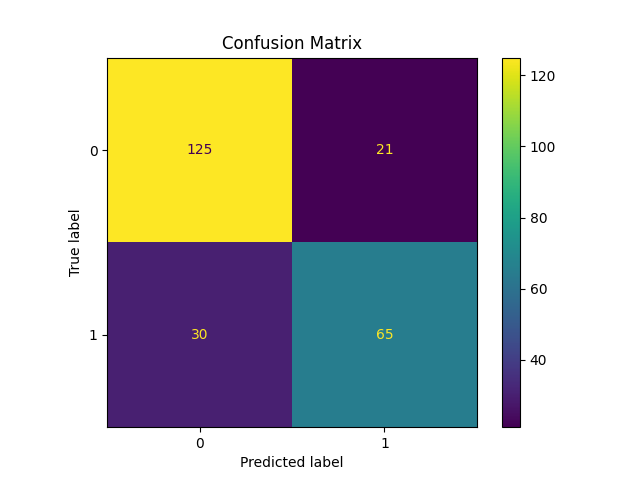


**Scatterplot of Predicted Probabilities**: The scatterplot reveals that Naive Bayes assigns probabilities clustered around 0 and 1, with a bias toward extreme values, but the lack of calibrated estimates may lead to overconfident predictions that are less accurate.

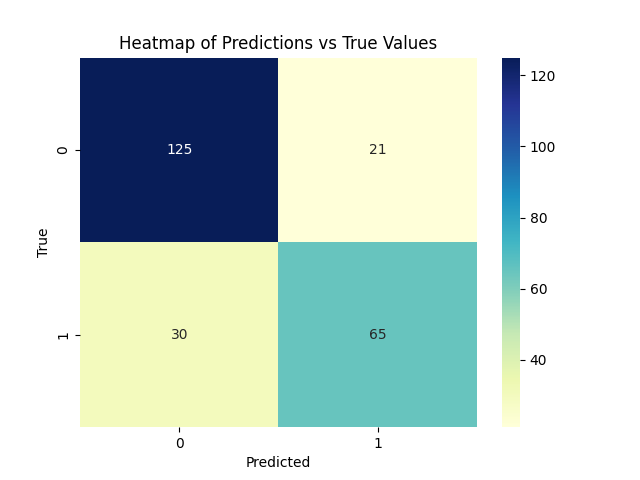
**Neural Network**

****

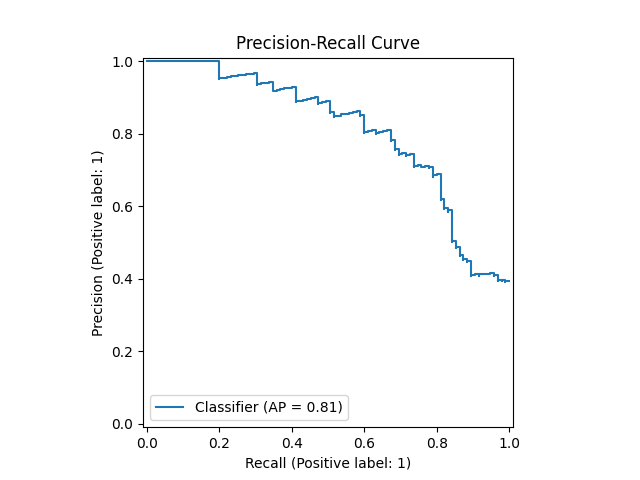
**Calibration Curve**: The calibration curve shows significant oscillations, with points deviating from the ideal diagonal line, indicating that the Neural Network model may struggle with probability calibration and tends to assign probabilities inconsistently across different ranges.



**Confusion Matrix**: The model demonstrates solid classification, with 125 true negatives and 65 true positives, though it has 21 false positives and 30 false negatives, suggesting balanced but moderate performance with room for improvement in both precision and recall.



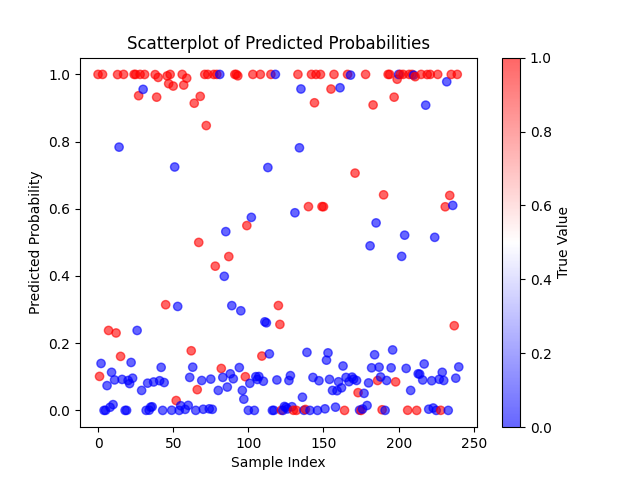
**Heatmap of Predictions vs True Values**: The heatmap emphasizes correct predictions along the diagonal, with visible misclassifications in both the false positive and false negative categories, which may impact the model's overall accuracy.



**Precision-Recall Curve**: With an AP score of 0.81, the precision-recall curve indicates that the Neural Network maintains a high precision-recall trade-off at various thresholds, though precision drops as recall increases, reflecting limitations in accurately capturing all positives.

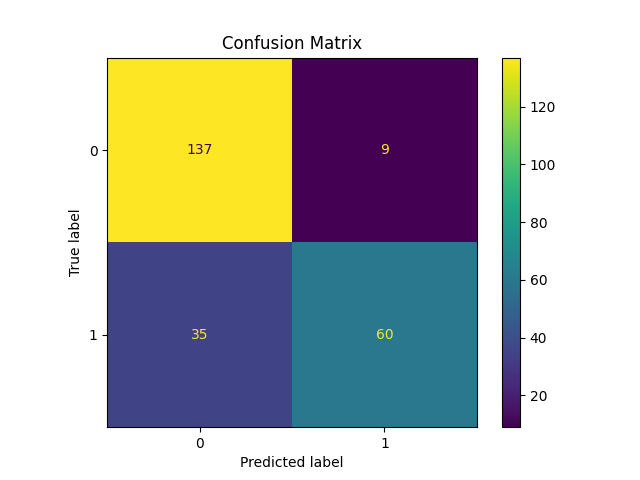


**ROC Curve**: The ROC curve, with an AUC of 0.82, demonstrates that the model has good discriminatory power between survivors and non-survivors, although further tuning could help improve its sensitivity and specificity.

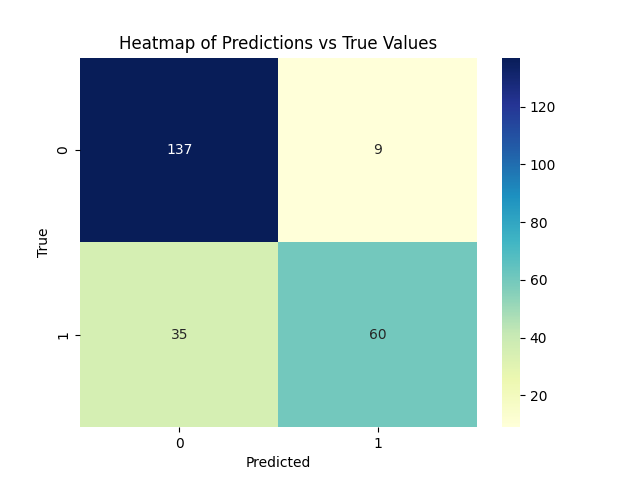


**Scatterplot of Predicted Probabilities**: The scatterplot displays a range of predicted probabilities, with clusters at both extremes and mid-range values, showing that the Neural Network provides nuanced probability estimates, though inconsistent calibration may reduce its reliability.

**SVC**

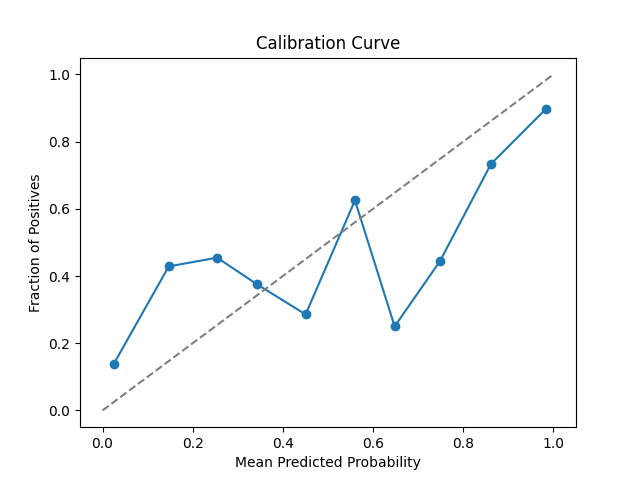


**Confusion Matrix**: The SVC model performs well in classifying non-survivors, with 137 true negatives and only 9 false positives. However, it has 35 false negatives, suggesting a tendency to miss survivors, which could impact sensitivity.



**Heatmap of Predictions vs True Values**: The heatmap visually emphasizes the SVC's strength in correctly predicting non-survivors (shown by the intense color in the top-left cell) but highlights its relative weakness in identifying survivors, as seen by the lower correct predictions in the bottom-right cell. This indicates a possible class imbalance or a conservative approach favoring non-survivor classification.

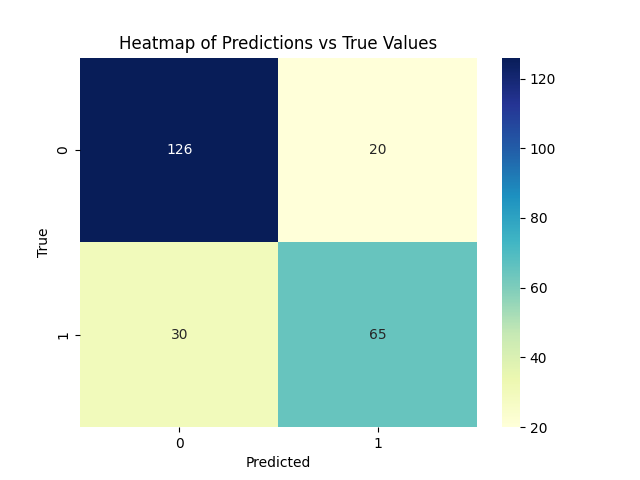
**XGBoost**

****

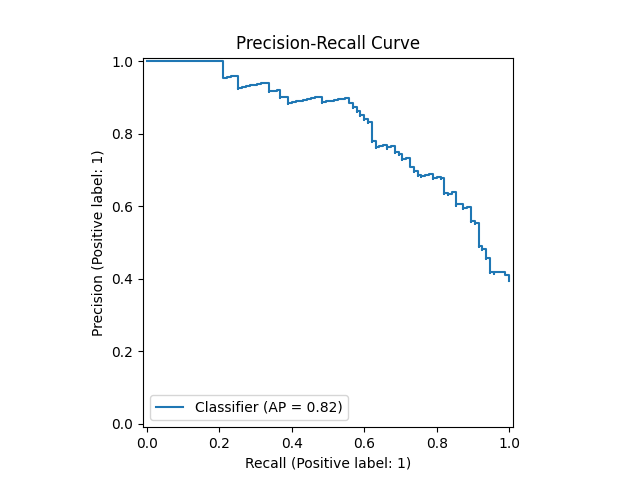
**Calibration Curve**: The calibration curve for XGBoost shows deviations from the ideal line, suggesting that while the model provides reasonable probability estimates, there may be calibration issues, especially in the mid-probability range.



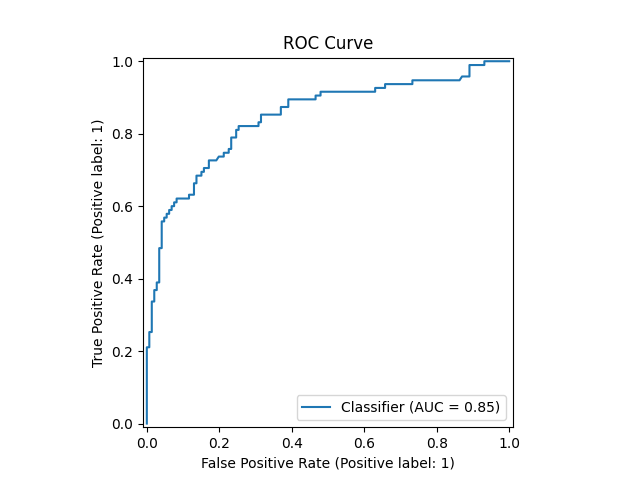
**Confusion Matrix**: The confusion matrix indicates that XGBoost performs well in identifying non-survivors, with 126 true negatives and 65 true positives, while misclassifying 30 instances as false negatives and 20 as false positives. This balance highlights moderate sensitivity and specificity.

****

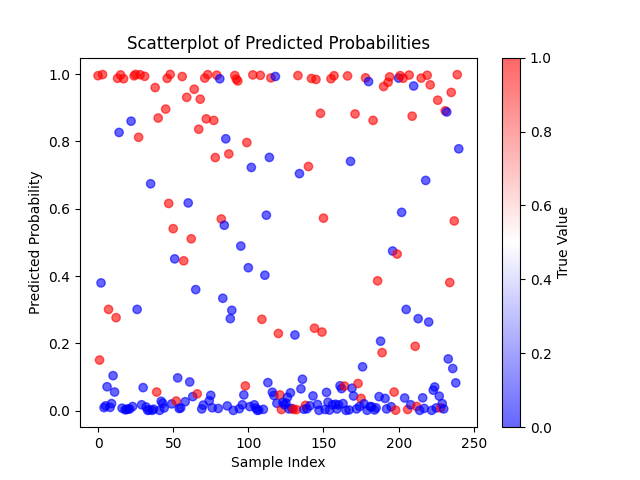
**Heatmap of Predictions vs True Values**: The heatmap emphasizes the accuracy of predictions along the diagonal cells, with some misclassifications in both false positive and false negative areas, suggesting that XGBoost achieves balanced performance but could still improve.



**Precision-Recall Curve**: With an AP score of 0.82, the precision-recall curve shows that XGBoost maintains a high level of precision at varying recall levels, though precision declines as recall approaches higher values, reflecting a typical trade-off in positive class identification.



**ROC Curve**: The ROC curve with an AUC of 0.85 indicates that XGBoost has strong discriminatory power in distinguishing survivors from non-survivors, though further improvements could help minimize false positives.



**Scatterplot of Predicted Probabilities**: The scatterplot displays a range of predicted probabilities, with clusters near the extremes and some dispersion in the mid-range, reflecting that XGBoost can produce nuanced probability estimates but may require better calibration for consistent reliability.