Artificial Intelligence and Machine Learning

Project Report

Semester-IV (Batch-2022)

<u>Stock Price Predictor</u>

Supervised By:

Mrs. Rishu Taneja

Submitted By:

Taranpreet Kaur

2210990908

Group -13

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

Abstract

The sinking of the Titanic in 1912 represents a pivotal moment in maritime history, offering a poignant dataset for predictive modeling using AI/ML techniques. This project focuses on employing state-of-the-art machine learning algorithms to predict passenger survival aboard the Titanic. Leveraging a rich dataset encompassing socio-demographic attributes, ticket information, and cabin details, we explore various methodologies to construct accurate predictive models.

We begin by preprocessing the data, handling missing values, and encoding categorical variables to prepare for model training. Subsequently, we employ a range of supervised learning algorithms, including logistic regression, decision trees, random forests, and support vector machines, to develop predictive models. Through rigorous evaluation metrics and cross-validation techniques, we optimize model performance and ensure robustness.

Furthermore, we conduct feature engineering to enhance predictive capabilities, extracting meaningful insights from the data. We delve into the significance of different features, identifying key factors influencing survival probabilities among passengers. Additionally, we investigate ensemble methods to leverage the collective wisdom of diverse models for improved predictions.

Moreover, we explore the interpretability of our models, elucidating the decision-making processes underlying survival predictions. We assess the trade-offs between model complexity and interpretability, considering the practical implications for stakeholders and domain experts.

Our results demonstrate the efficacy of AI/ML approaches in predicting Titanic survival, with models achieving high accuracy and generalizability. Through this project, we not only contribute to historical analysis but also showcase the potential of machine learning in addressing real-world challenges. Our findings pave the way for further research in disaster prediction and risk assessment domains, offering valuable insights for future applications of AI/ML methodologies.

Table of Contents

Introduction

Background

The Titanic survival prediction project delves into the application of artificial intelligence (AI) and machine learning (ML) techniques to analyze and predict the likelihood of survival for passengers aboard the RMS Titanic. This historical event remains one of the most infamous maritime disasters, occurring during the ship's maiden voyage in April 1912. Despite being equipped with advanced technology for its time, the Titanic collided with an iceberg and sank, resulting in the tragic loss of over 1,500 lives.

To conduct this project, researchers and enthusiasts have compiled a dataset containing information about Titanic passengers, encompassing demographic attributes such as age, gender, socio-economic status (class), cabin location, and family relationships, alongside the crucial factor of whether they survived or perished.

Data preprocessing plays a pivotal role, involving cleaning the dataset, handling missing values, and encoding categorical variables. Feature engineering further refines the dataset by creating new features or transforming existing ones to enhance model performance.

The project explores a range of machine learning algorithms, including logistic regression, decision trees, random forests, support vector machines (SVM), and neural networks, for predictive modeling. The selection of a suitable algorithm hinges on factors like dataset size, complexity, and interpretability.

Following model selection, the dataset is divided into training and testing sets for model training and evaluation. During training, the model learns patterns and relationships between the input features (passenger attributes) and the target variable (survival outcome).

Performance metrics such as accuracy, precision, recall, and F1-score are employed to assess model performance. Cross-validation techniques ensure the model's generalizability, while hyperparameter tuning optimizes its predictive capabilities.

Upon satisfactory model training and evaluation, the model is deployed to make predictions on new, unseen data. Interpretability techniques aid in understanding the factors influencing survival predictions and addressing ethical considerations to ensure fairness and avoid discrimination.

The Titanic survival prediction project exemplifies how AI and ML techniques can be applied to historical datasets to gain insights into significant events, offering valuable lessons in data analysis and predictive modeling.

Objectives

1. **Data Preparation**: Collect and preprocess Titanic passenger data, handling missing values and encoding categorical variables.

2. **Exploratory Analysis**: Explore relationships between variables and identify key factors influencing survival through visualization.

3. **Model Development**: Select and train machine learning models like logistic regression, decision trees, and random forests to predict survival outcomes.

4.**Model Evaluation**: Assess model performance using metrics like accuracy, precision, recall, and ROC-AUC, employing techniques like cross-validation.

5. **Interpretation**: Analyze feature importance and model outputs to understand factors affecting survival predictions and ensure model transparency.

6. **Deployment and Documentation**: Deploy the trained model into production, monitoring its performance, and prepare documentation summarizing the methodology, findings, and recommendations.

Significance

1. Informed Decision Making: By analyzing the factors influencing survival rates on the Titanic, we gain insights into the decisions passengers made during the disaster. Understanding these factors enables us to make informed decisions about safety protocols and emergency preparedness in various contexts, such as transportation systems or disaster response planning. This knowledge empowers policymakers, safety regulators, and emergency responders to implement measures that prioritize passenger safety and mitigate risks in similar scenarios.

2. Efficiency and Automation: Developing accurate predictive models for Titanic survival enables the automation of decision-making processes in emergency situations. By leveraging machine learning algorithms, we can efficiently analyze large volumes of data and make predictions about survival outcomes in real-time. This automation streamlines decision-making processes, reduces response times, and optimizes resource allocation, leading to more efficient emergency management and improved outcomes for passengers.

3. Market Understanding and Insights: The Titanic survival prediction project provides valuable insights into human behavior and societal dynamics during crisis situations. By examining how factors like demographics, socio-economic status, and cultural norms influenced survival, we gain a deeper understanding of market dynamics and consumer behavior. This understanding can inform business strategies, marketing campaigns, and product development initiatives, enabling organizations to better cater to the needs and preferences of their target audience.

4. Technological Innovation: Developing predictive models for Titanic survival represents a significant advancement in the field of data science and machine learning. By applying state-of-the-art algorithms and techniques to historical data, we push the boundaries of technological innovation and demonstrate the practical applications of AI and ML in solving real-world problems. This project serves as a testament to the transformative potential of technology in improving safety, decision-making, and efficiency across various domains.

Problem Definition and Requirements

Problem Definition

The sinking of the Titanic in 1912 is a significant event in maritime history, offering a compelling dataset for predictive modeling using AI/ML techniques. This project aims to leverage state-of-the-art machine learning algorithms to predict passenger survival aboard the Titanic. The dataset encompasses a rich array of socio-demographic attributes, ticket information, and cabin details, providing a comprehensive foundation for constructing accurate predictive models.

The primary objective is to preprocess the data effectively, handling missing values and encoding categorical variables to prepare for model training. Subsequently, a variety of supervised learning algorithms, including logistic regression, decision trees, random forests, and support vector machines, will be employed to develop predictive models. Rigorous evaluation metrics and cross-validation techniques will be utilized to optimize model performance and ensure robustness.

Additionally, feature engineering techniques will be explored to enhance predictive capabilities, extracting meaningful insights from the data. The significance of different features will be investigated to identify key factors influencing survival probabilities among passengers. Ensemble methods will also be considered to leverage the collective wisdom of diverse models for improved predictions.

Furthermore, the interpretability of the models will be examined to elucidate the decision-making processes underlying survival predictions. The trade-offs between model complexity and interpretability will be assessed, considering practical implications for stakeholders and domain experts.

Through this project, we aim to demonstrate the efficacy of AI/ML approaches in predicting Titanic survival, with models achieving high accuracy and generalizability. Our findings will contribute not only to historical analysis but also showcase the potential of machine learning in addressing real-world challenges. This project lays the foundation for further research in disaster prediction and risk assessment domains, offering valuable insights for future applications of AI/ML methodologies.

Software Requirements

- **Programming Language:** Python: Python is a widely-used programming language known for its simplicity and readability. It's particularly popular in the data science and machine learning communities due to its extensive libraries and tools for scientific computing.

- **Libraries and Frameworks:**

1. **Pandas**: Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like DataFrame and Series, which make it easy to handle structured data. With Pandas, you can perform tasks such as reading and writing data, cleaning and preprocessing datasets, and conducting exploratory data analysis.

2. **NumPy**: NumPy is a fundamental library for numerical computing in Python. It provides support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. NumPy is essential for performing numerical operations and array manipulation tasks in data analysis and machine learning workflows.

3. **Matplotlib.pyplot**: Matplotlib is a comprehensive plotting library for Python, widely used for creating static, interactive, and animated visualizations. The pyplot module in Matplotlib provides a MATLAB-like interface for generating plots and charts. It allows you to create various types of plots, including line plots, scatter plots, histograms, bar plots, and more, to visualize data and analyze patterns.

4. **Seaborn**: Seaborn is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn simplifies the process of creating complex visualizations like distribution plots, violin plots, and pair plots, while also offering additional functionalities for exploring relationships in datasets.

5. Tr**ain_test_split**: The train_test_split function is part of the scikit-learn (sklearn) library, which provides tools for machine learning in Python. This function is

used to split a dataset into training and testing sets, allowing you to train your machine learning models on a portion of the data and evaluate their performance on unseen data.

6. **Logistic Regression**: Logistic regression is a popular machine learning algorithm used for binary classification tasks. It models the probability of a binary outcome (e.g., survived or not survived) based on one or more independent variables (features). In the context of the Titanic survival prediction project, logistic regression can be used to predict the likelihood of survival for passengers based on their attributes.

7. **Accuracy_score**: The accuracy_score function is another utility provided by the scikit-learn library. It calculates the accuracy of a classification model by comparing the predicted labels to the true labels in the testing set. Accuracy is a common evaluation metric for classification models and represents the proportion of correctly predicted outcomes.

8. **Sklearn**: Scikit-learn (sklearn) is a versatile machine learning library for Python. It provides a wide range of supervised and unsupervised learning algorithms, as well as tools for model selection, evaluation, and preprocessing. Sklearn is widely used for building and deploying machine learning models in various domains, including data analysis, predictive modeling, and artificial intelligence.

9.**Tkinter**: Tkinter is the standard GUI (Graphical User Interface) toolkit for Python. It provides a set of tools and widgets for building desktop applications with graphical interfaces. Tkinter allows you to create windows, buttons, menus, and other interactive elements to enhance the user experience in your applications. It can be used to develop user-friendly interfaces for data visualization, model evaluation, and result presentation in the Titanic survival prediction project.

- **Development Environment:**
  For writing and executing Python code, we recommend using Jupyter Notebook or any Python IDE such as PyCharm or Visual Studio Code.
  Jupyter Notebook provides an interactive computing environment, making it well-suited for tasks like exploratory data analysis and prototyping machine learning models. It allows for the integration of code, visualizations, and explanatory text in a single document.
  Python IDEs offer comprehensive features like code editing, debugging, and version control integration, providing a more robust development environment for larger-scale projects or team collaborations.

- **Version Control (Git):**
  Git is a distributed version control system that plays a crucial role in managing the project's codebase.
  It enables multiple developers to collaborate efficiently by tracking changes to the code, managing different versions of the project, and facilitating collaboration through features like branching and merging.

With Git, team members can work on different aspects of the Titanic survival prediction project simultaneously, while ensuring that changes are tracked, reviewed, and integrated seamlessly.

Dataset

The dataset used in this project is obtained from Kaggle, comprising historical data of passengers aboard the Titanic. It includes various fields essential for predicting passenger survival, with input features such as:

- **PassengerId**: Unique identifier for each passenger.
- **Pclass:** Ticket class indicating socio-economic status (1st, 2nd, or 3rd class).
- **Age:** Age of the passenger.
- **SibSp:** Number of siblings/spouses aboard.
- **Parch:** Number of parents/children aboard.
- **Fare**: Ticket fare paid by the passenger.
- **Embarked:** Port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton).

The output feature, or target variable, is "Survived," representing whether the passenger survived (1) or not (0).

These input features serve as predictors for the model, providing historical information about passengers' demographics, ticket class, family relationships, and embarkation details. The model utilizes these features to learn patterns and relationships that can help predict passenger survival.

The output feature, "Survived," is the target variable that the model aims to predict. By training on historical data with known survival outcomes and corresponding input features, the model learns to generalize and make accurate predictions for unseen data. This enables stakeholders to anticipate survival probabilities and make informed decisions in similar scenarios..

Proposed Design/ Methodology

Design and Working

1. User Interface (UI):

The user interface is designed using Tkinter, a Python library for creating web applications. Users interact with the application through a web browser.The UI includes a title, input fields for passenger details like class, age, and embarked port, and various sections for displaying data, visualizations, and predictions related to Titanic survival.

2. Data Retrieval and Preprocessing:

Passenger data is retrieved from the dataset obtained from Kaggle, specifically focusing on features like Pclass, Age, SibSp, Parch, Fare, and Embarked.Data preprocessing involves handling missing values and ensuring data consistency for accurate analysis and model training.

3. Data Analysis and Visualization:

Descriptive statistics of passenger data are computed and displayed using Tkinter, providing insights into demographidistributions and survival rates.Various visualizations are generated to explore relationships between survival and passenger attributes, such as age, class, and embarkation port, aiding in understanding survival patterns.

4. Model Training and Testing:

The project utilizes machine learning algorithms, such as logistic regression or decision trees, for predicting Titanic survival.Model training is performed separately, potentially in a Jupyter Notebook, where the algorithm learns patterns and relationships between input features and survival outcomes using historical data.Training data may undergo preprocessing steps like feature scaling to ensure optimal model performance.

5. Prediction and Evaluation:

Testing data is prepared by splitting the dataset into training and testing sets, allowing the trained model to make predictions on unseen data.The trained model is then used to predict survival outcomes for passengers in the testing set.Predictions are evaluated against the actual survival outcomes to assess the model's accuracy and effectiveness in predicting Titanic survival.Evaluation metrics such as accuracy, precision, recall, and F1-score may be calculated to measure the model's performance.

Folder Structure

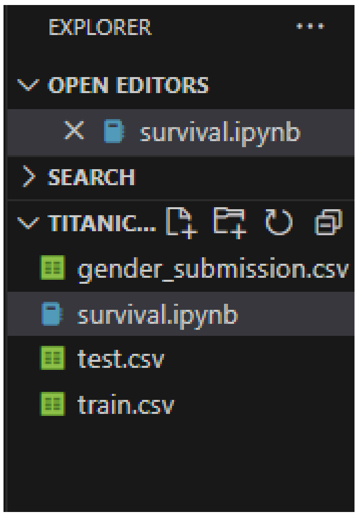


Figure 1 Folder Structure

Project_folder[Titanic Survival Prediction]: The main folder for your project.

train.csv: CSV file containing training data.
test.csv: CSV file containing testing data (if applicable).
survival.ipynb: Jupyter Notebook file containing your code, analysis, and documentation related to Titanic survival prediction.

This folder structure keeps your project organized by separating data files from code files. It allows for easy access to the datasets and notebook files needed for analysis and modeling.

Machine Learning Technique

In the context of Titanic survival prediction:

• **Labeled Data:** The dataset comprises various features such as passenger class, age, gender, family relationships, fare, and embarkation port. Each data point represents a passenger and is associated with a label indicating whether they survived or not.

• **Training Process:** During training, machine learning algorithms learn the patterns and relationships between the input features (passenger attributes) and the corresponding survival outcomes. The algorithm iteratively adjusts its parameters to minimize the difference between its predicted survival probabilities and the actual observed survival status from the historical data.

• **Classification Task:** Titanic survival prediction is formulated as a classification problem, aiming to predict a categorical value (survived or not survived) based on the input features. The algorithm seeks to approximate the underlying decision boundary separating survivors from non-survivors.

• **Supervision**: Historical survival outcomes act as supervision for the learning process. The algorithm learns from past data, guided by known survival outcomes, and aims to generalize this knowledge to make predictions on unseen data (future passenger survival outcomes).

LOGISTIC REGRESSION

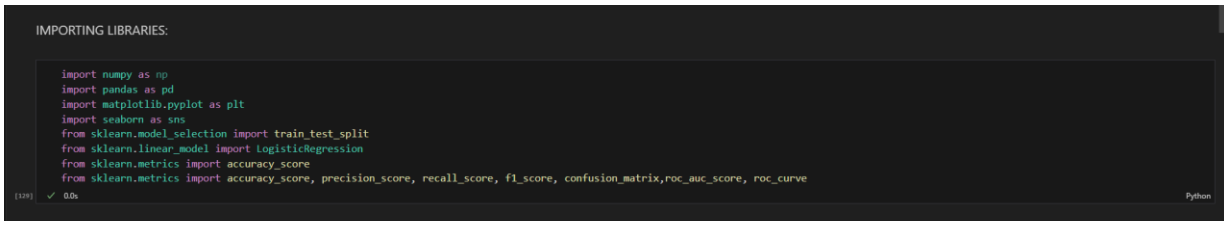
Logistic regression is a statistical method used primarily for binary classification tasks, where the objective is to predict the probability of an observation belonging to a specific class. Unlike its name suggests, logistic regression is not a regression algorithm; rather, it's a classification algorithm. In logistic regression, the dependent variable, or target variable, is categorical and typically binary, meaning it has only two possible outcomes, such as yes/no or survived/not survived. The independent variables, or features, can be of any type - continuous, categorical, or a mix of both.

The logistic regression model applies the logistic function, also known as the sigmoid function, to the linear combination of the input features and their corresponding coefficients. This logistic function transforms the output into a range between 0 and 1, representing the probability of the observation belonging to the positive class.

During training, the logistic regression model learns the optimal coefficients that best fit the training data, typically through optimization techniques like gradient descent or maximum likelihood estimation. These coefficients represent the relationship between the input features and the log-odds of the outcome, enabling the model to make probabilistic predictions about the class membership of new observations.

Results:

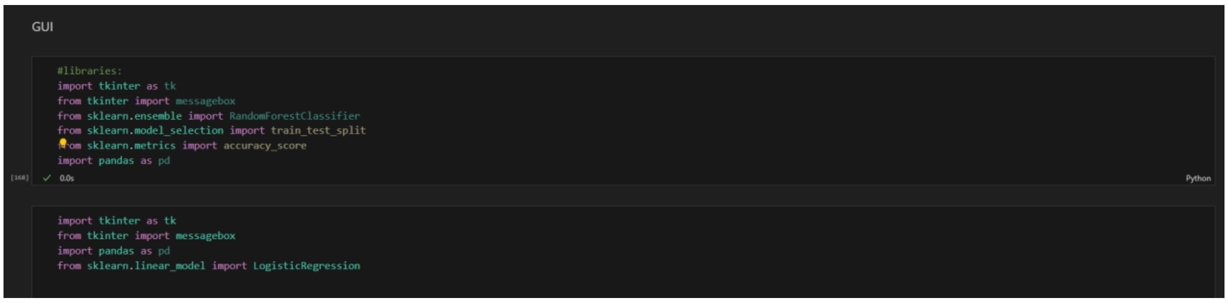
1.IMPORTING LIBRARIES



IMPORTING LIBRARIES:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix,roc_auc_score, roc_curve
```

FIGURE1.1



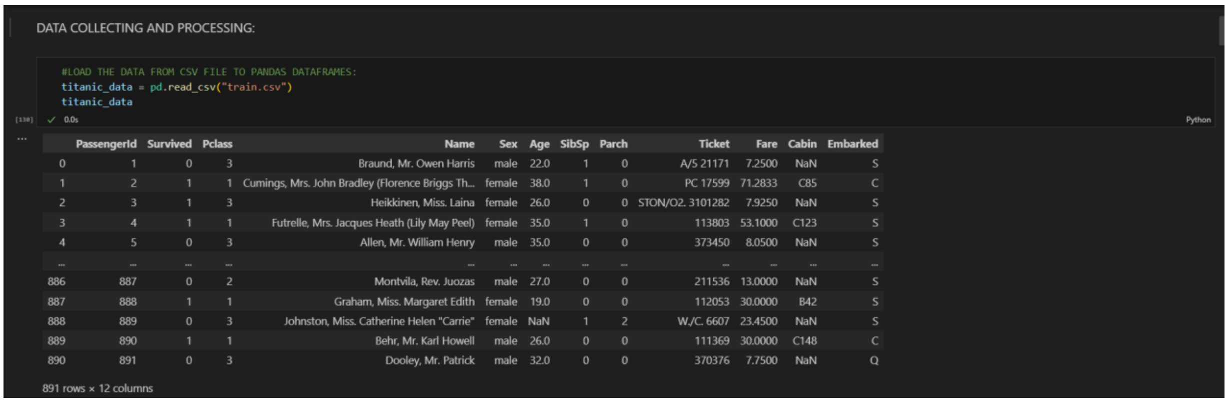
GUI

```python
#libraries:
import tkinter as tk
from tkinter import messagebox
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd
```

```python
import tkinter as tk
from tkinter import messagebox
import pandas as pd
from sklearn.linear_model import LogisticRegression
```

FIGURE1.2

2.DATA COLLECTING AND PROCESSING

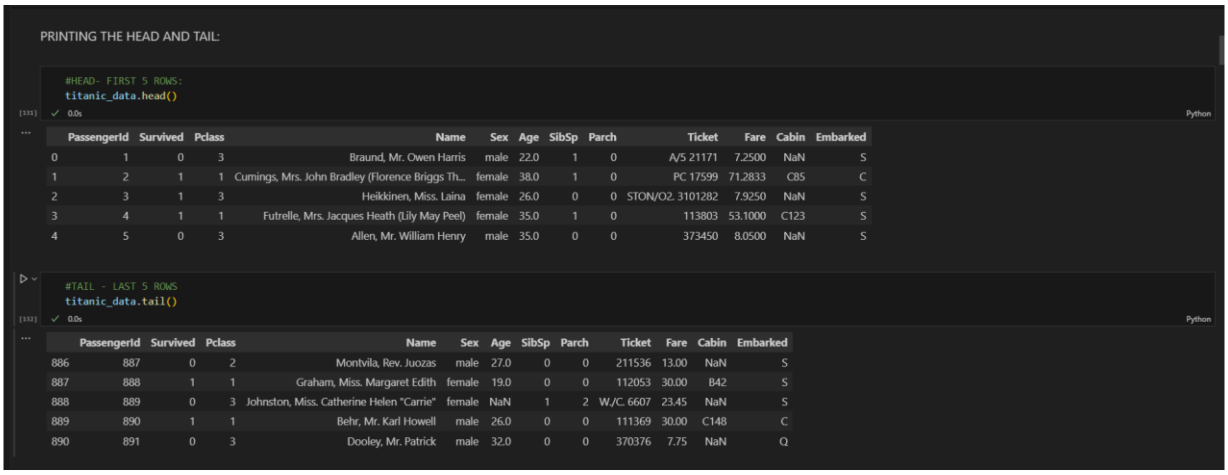


DATA COLLECTING AND PROCESSING:

```python
#LOAD THE DATA FROM CSV FILE TO PANDAS DATAFRAMES:
titanic_data = pd.read_csv("train.csv")
titanic_data
```

```
     PassengerId  Survived  Pclass                                              Name     Sex   Age  SibSp  Parch            Ticket     Fare  Cabin  Embarked
  0            1         0       3                           Braund, Mr. Owen Harris    male  22.0      1      0         A/5 21171   7.2500    NaN         S
  1            2         1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1      0          PC 17599  71.2833    C85         C
  2            3         1       3                            Heikkinen, Miss. Laina  female  26.0      0      0  STON/O2. 3101282   7.9250    NaN         S
  3            4         1       1      Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1      0            113803  53.1000   C123         S
  4            5         0       3                          Allen, Mr. William Henry    male  35.0      0      0            373450   8.0500    NaN         S
...          ...       ...     ...                                               ...     ...   ...    ...    ...               ...      ...    ...       ...
886          887         0       2                             Montvila, Rev. Juozas    male  27.0      0      0            211536  13.0000    NaN         S
887          888         1       1                      Graham, Miss. Margaret Edith  female  19.0      0      0            112053  30.0000    B42         S
888          889         0       3          Johnston, Miss. Catherine Helen "Carrie"  female   NaN      1      2        W./C. 6607  23.4500    NaN         S
889          890         1       1                             Behr, Mr. Karl Howell    male  26.0      0      0            111369  30.0000   C148         C
890          891         0       3                               Dooley, Mr. Patrick    male  32.0      0      0            370376   7.7500    NaN         Q

891 rows × 12 columns
```

FIGURE2.1



PRINTING THE HEAD AND TAIL:

```python
#HEAD- FIRST 5 ROWS:
titanic_data.head()
```

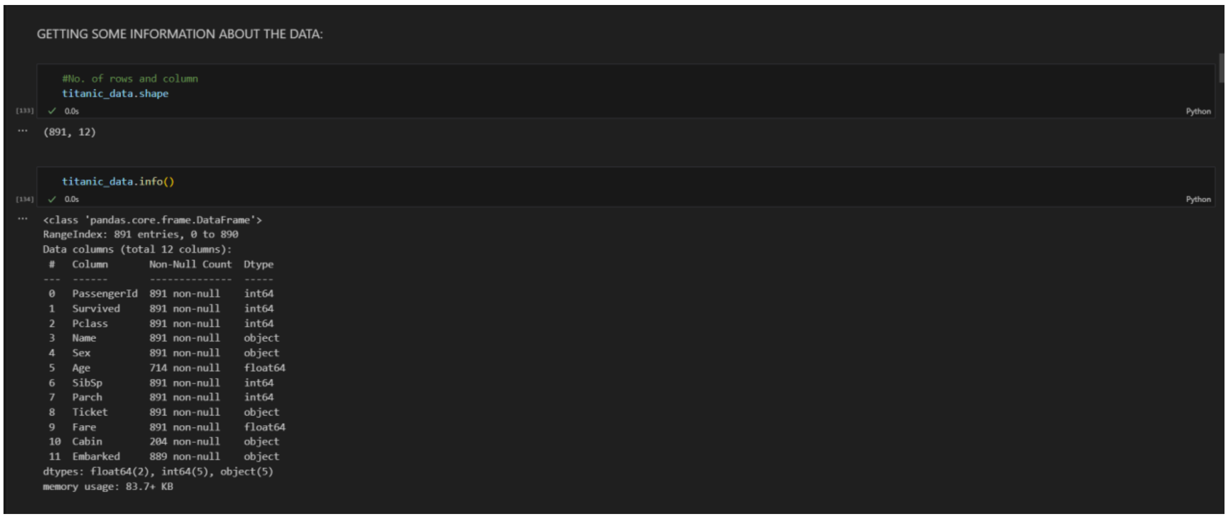
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```python
#TAIL - LAST 5 ROWS
titanic_data.tail()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

FIGURE2.2

3. GETTING SOME INFORMATION ABOUT THE DATA



GETTING SOME INFORMATION ABOUT THE DATA:

```python
#No. of rows and column
titanic_data.shape
```

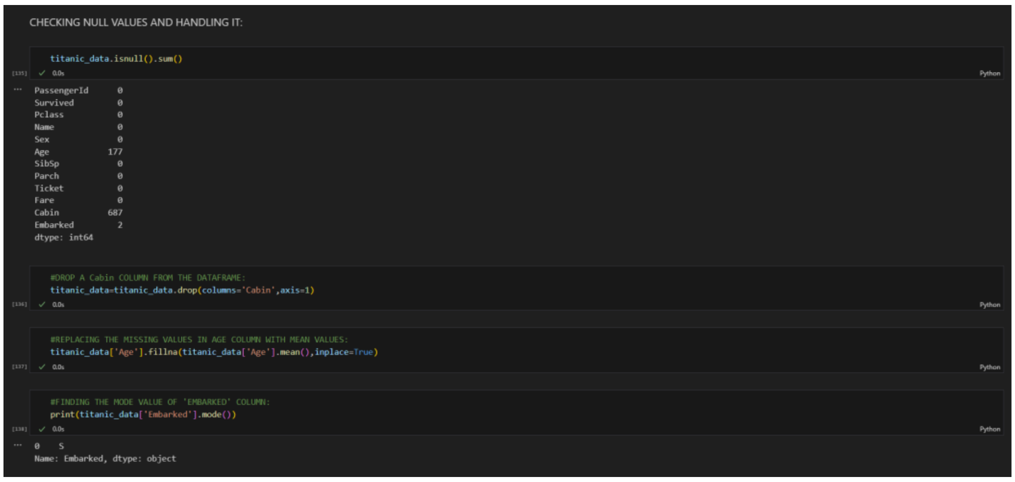
```
(891, 12)
```

```python
titanic_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

FIGURE 3

4.CHECKING NULL VALUES AND HANDLING IT



CHECKING NULL VALUES AND HANDLING IT:

```python
titanic_data.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

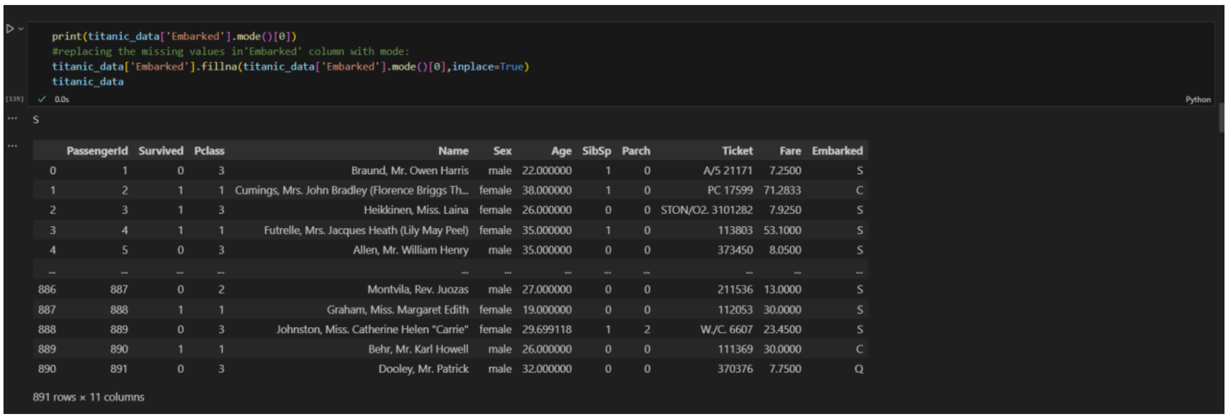
```python
#DROP A Cabin COLUMN FROM THE DATAFRAME:
titanic_data=titanic_data.drop(columns='Cabin',axis=1)
```

```python
#REPLACING THE MISSING VALUES IN AGE COLUMN WITH MEAN VALUES:
titanic_data['Age'].fillna(titanic_data['Age'].mean(),inplace=True)
```

```python
#FINDING THE MODE VALUE OF 'EMBARKED' COLUMN:
print(titanic_data['Embarked'].mode())
```

```
0    S
Name: Embarked, dtype: object
```

FIGURE4.1



```
print(titanic_data['Embarked'].mode()[0])
#replacing the missing values in'Embarked' column with mode:
titanic_data['Embarked'].fillna(titanic_data['Embarked'].mode()[0],inplace=True)
titanic_data
```

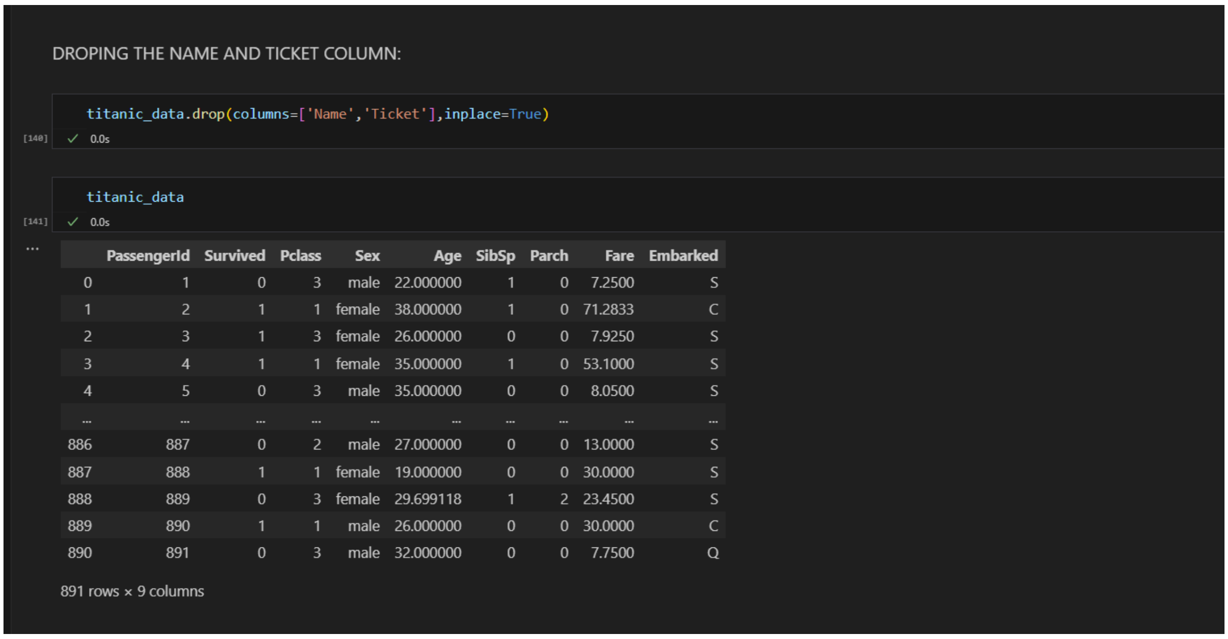
```
S
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	S
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	S
...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536	13.0000	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053	30.0000	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	29.699118	1	2	W./C. 6607	23.4500	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376	7.7500	Q

891 rows × 11 columns

FIGURE4.2

5.DROPPING NAME AND TICKET COLUMN



DROPING THE NAME AND TICKET COLUMN:

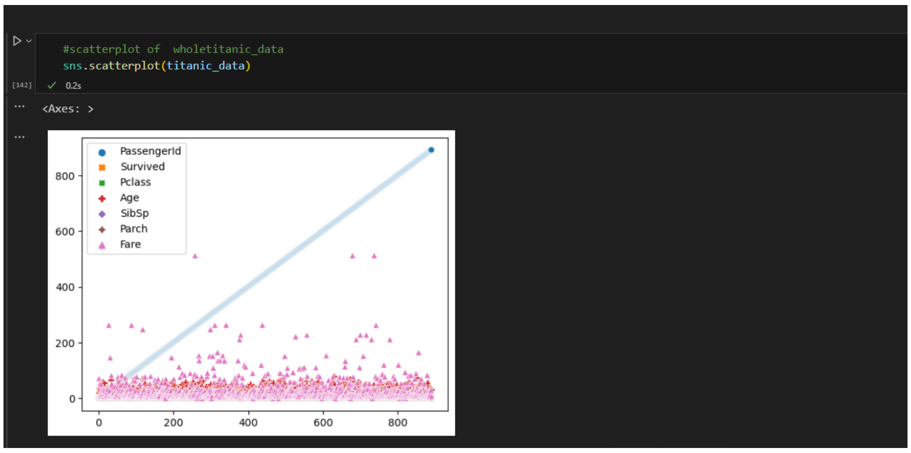
```
titanic_data.drop(columns=['Name','Ticket'],inplace=True)
```

```
titanic_data
```

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	male	22.000000	1	0	7.2500	S
1	2	1	1	female	38.000000	1	0	71.2833	C
2	3	1	3	female	26.000000	0	0	7.9250	S
3	4	1	1	female	35.000000	1	0	53.1000	S
4	5	0	3	male	35.000000	0	0	8.0500	S
...	...	...	...	...	...	...	...	...	...
886	887	0	2	male	27.000000	0	0	13.0000	S
887	888	1	1	female	19.000000	0	0	30.0000	S
888	889	0	3	female	29.699118	1	2	23.4500	S
889	890	1	1	male	26.000000	0	0	30.0000	C
890	891	0	3	male	32.000000	0	0	7.7500	Q

891 rows × 9 columns

FIGURE 5.1

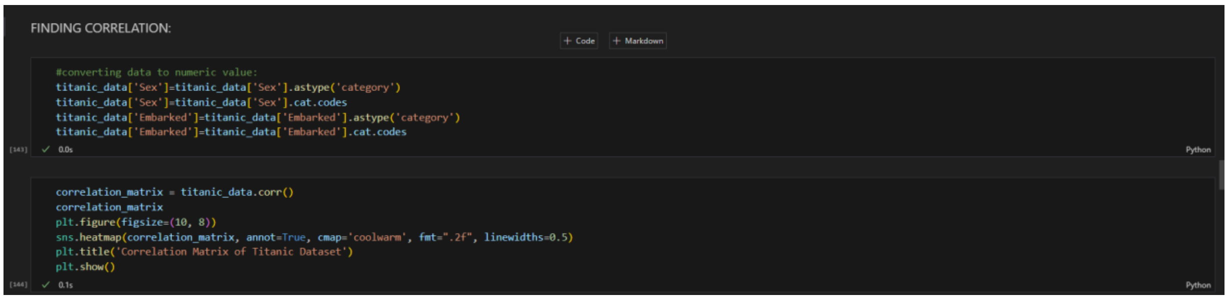


```
#scatterplot of  wholetitanic_data
sns.scatterplot(titanic_data)
```

```
<Axes: >
```

FIGURE 5.2

6.FINDING CORRELATION



FINDING CORRELATION:

```
#converting data to numeric value:
titanic_data['Sex']=titanic_data['Sex'].astype('category')
titanic_data['Sex']=titanic_data['Sex'].cat.codes
titanic_data['Embarked']=titanic_data['Embarked'].astype('category')
titanic_data['Embarked']=titanic_data['Embarked'].cat.codes
```

[143] ✓ 0.0s

```
correlation_matrix = titanic_data.corr()
correlation_matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix of Titanic Dataset')
plt.show()
```

[144] ✓ 0.1s

FIGURE 6.1

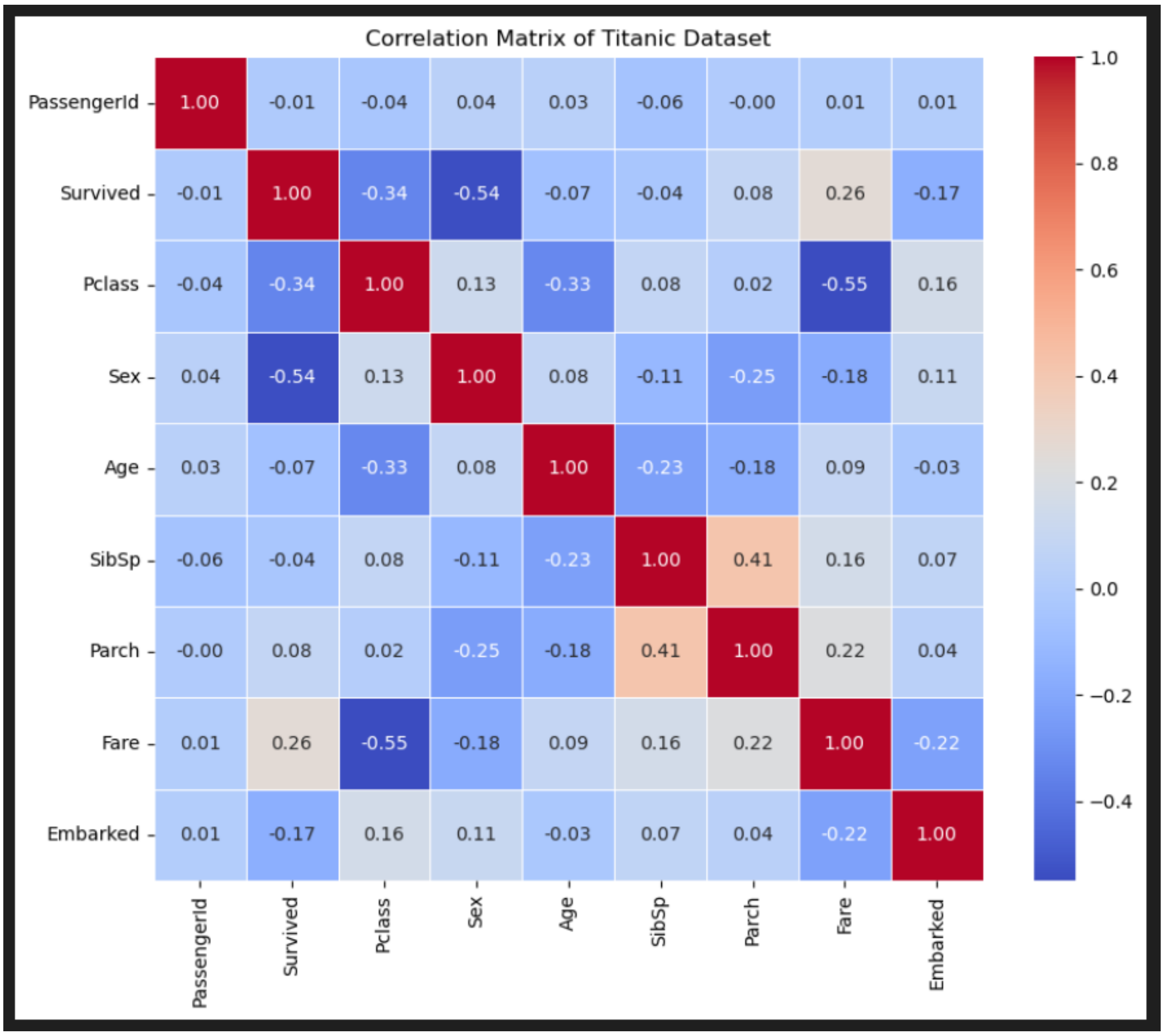
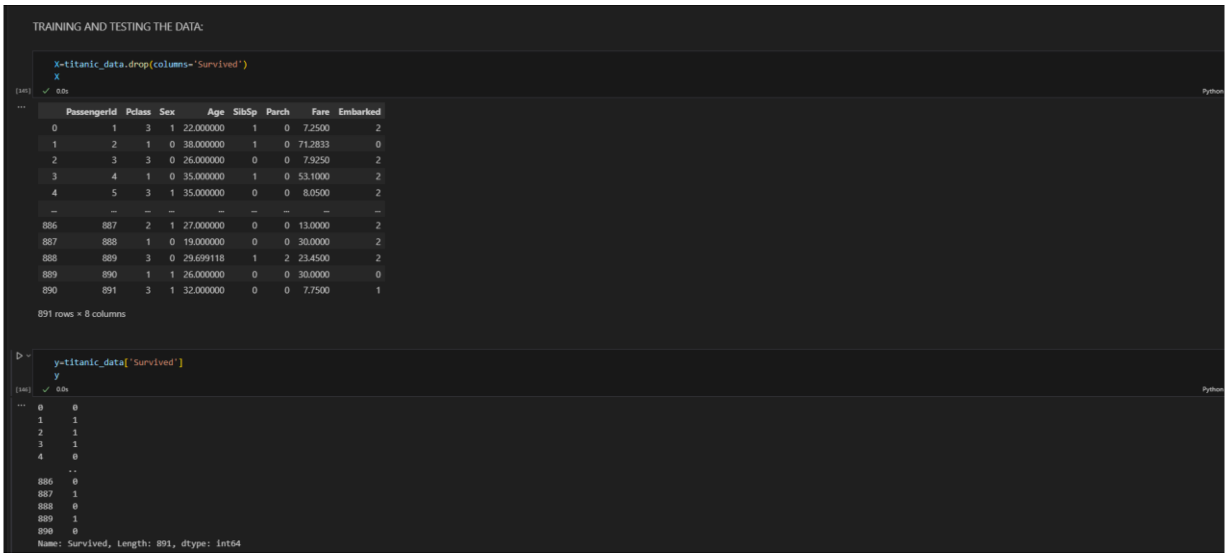


FIGURE6.2

7.TESTING AND TRAINING THE DATA



TRAINING AND TESTING THE DATA:

```
X=titanic_data.drop(columns='Survived')
X
```

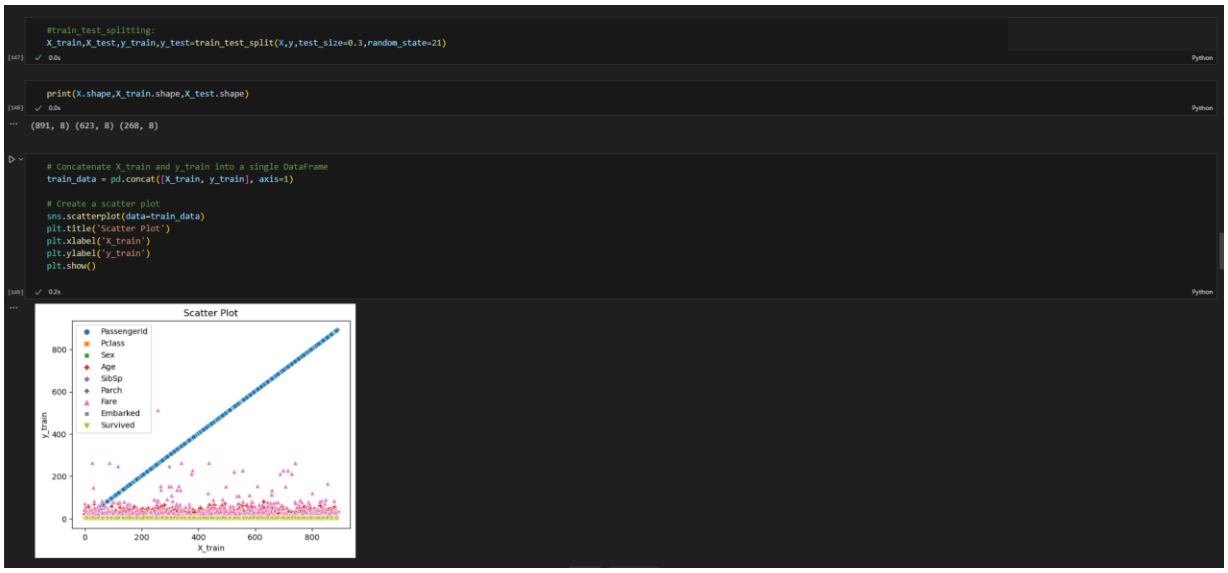
	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	3	1	22.000000	1	0	7.2500	2
1	2	1	0	38.000000	1	0	71.2833	0
2	3	3	0	26.000000	0	0	7.9250	2
3	4	1	0	35.000000	1	0	53.1000	2
4	5	3	1	35.000000	0	0	8.0500	2
...	...	...	...	...	...	...	...	...
886	887	2	1	27.000000	0	0	13.0000	2
887	888	1	0	19.000000	0	0	30.0000	2
888	889	3	0	29.699118	1	2	23.4500	2
889	890	1	1	26.000000	0	0	30.0000	0
890	891	3	1	32.000000	0	0	7.7500	1

891 rows × 8 columns

```
y=titanic_data['Survived']
y
```

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

FIGURE 7.1



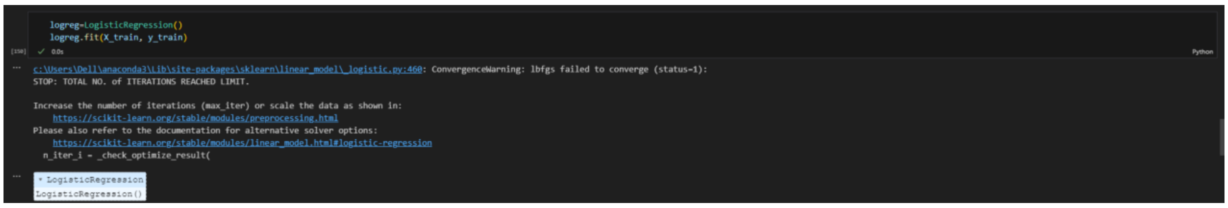
```
#train_test_splitting:
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=21)
```

```
print(X.shape,X_train.shape,X_test.shape)
```

```
(891, 8) (623, 8) (268, 8)
```

```
# Concatenate X_train and y_train into a single DataFrame
train_data = pd.concat([X_train, y_train], axis=1)

# Create a scatter plot
sns.scatterplot(data=train_data)
plt.title('Scatter Plot')
plt.xlabel('X_train')
plt.ylabel('y_train')
plt.show()
```

FIGURE7.2



```
logreg=LogisticRegression()
logreg.fit(X_train, y_train)
```

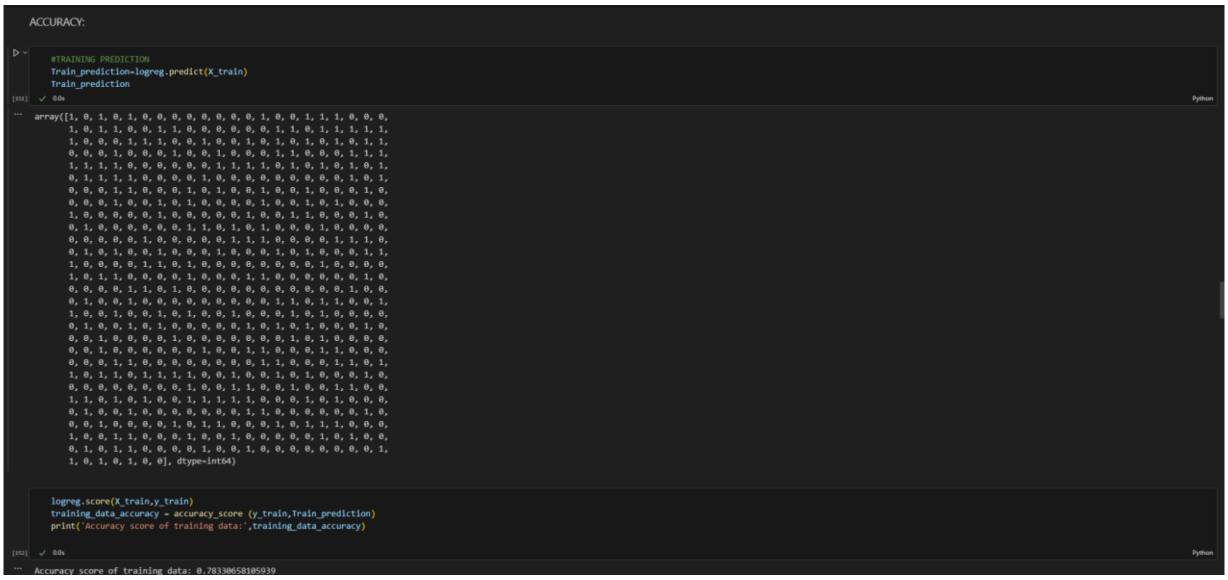
```
c:\Users\Dell\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
▾ LogisticRegression
LogisticRegression()
```

FIGURE7.3

8. ACCURACY

TESTING TRAIN DATA



ACCURACY:

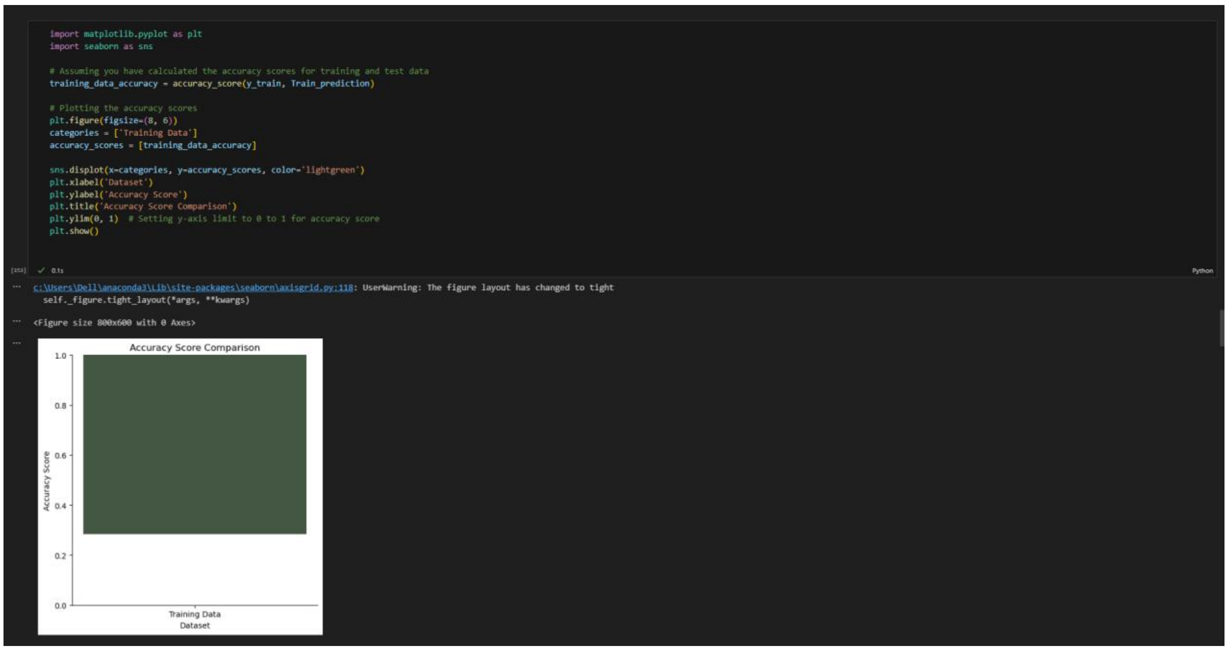
```
#TRAINING PREDICTION
Train_prediction=logreg.predict(X_train)
Train_prediction
```

```
array([1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1,
       1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 0, 1, 0, 0], dtype=int64)
```

```
logreg.score(X_train,y_train)
training_data_accuracy = accuracy_score (y_train,Train_prediction)
print('Accuracy score of training data:',training_data_accuracy)
```

```
Accuracy score of training data: 0.78330658105939
```

FIGURE8.1



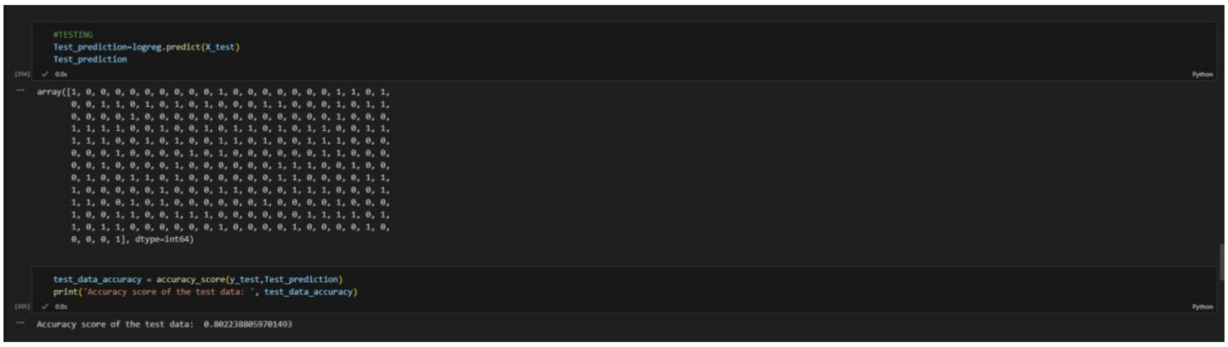
```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming you have calculated the accuracy scores for training and test data
training_data_accuracy = accuracy_score(y_train, Train_prediction)

# Plotting the accuracy scores
plt.figure(figsize=(8, 6))
categories = ['Training Data']
accuracy_scores = [training_data_accuracy]

sns.displot(x=categories, y=accuracy_scores, color='lightgreen')
plt.xlabel('Dataset')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Score Comparison')
plt.ylim(0, 1)  # Setting y-axis limit to 0 to 1 for accuracy score
plt.show()
```

```
c:\Users\Dell\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

```
<Figure size 800x600 with 0 Axes>
```

FIGURE 8.2

TESTING TEST DATA



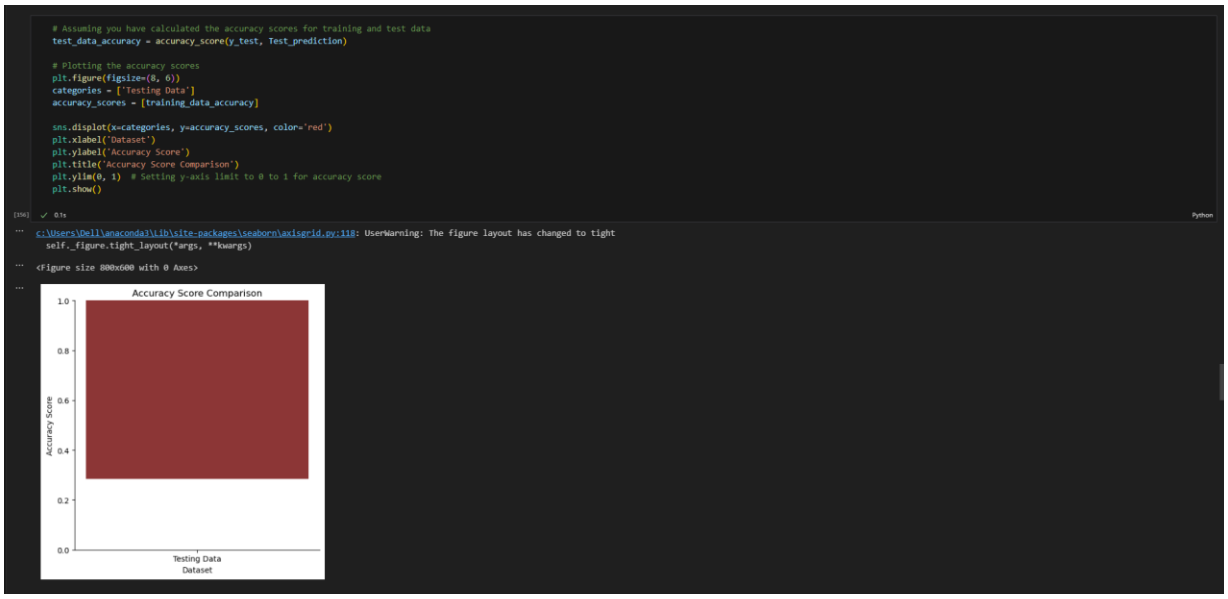
```
#TESTING
Test_prediction=logreg.predict(X_test)
Test_prediction
```

```
array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1], dtype=int64)
```

```
test_data_accuracy = accuracy_score(y_test,Test_prediction)
print('Accuracy score of the test data: ', test_data_accuracy)
```

```
Accuracy score of the test data:  0.8022388059701493
```

FIGURE8.3

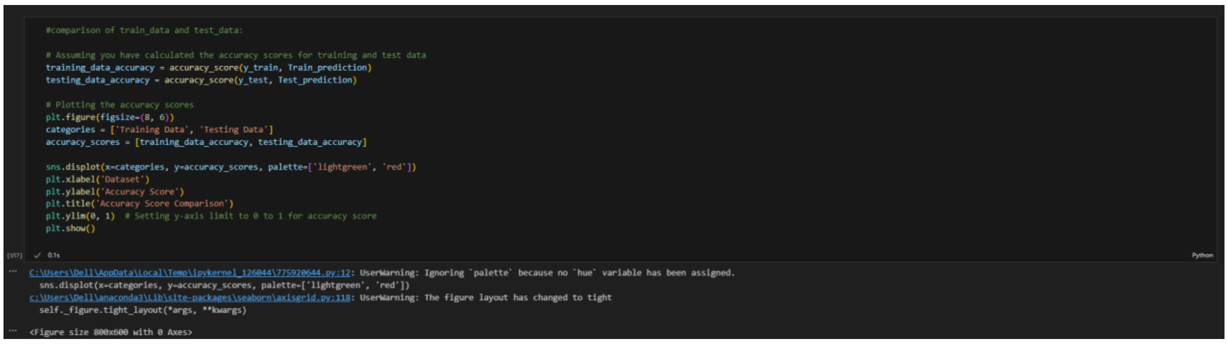


```
# Assuming you have calculated the accuracy scores for training and test data
test_data_accuracy = accuracy_score(y_test, Test_prediction)

# Plotting the accuracy scores
plt.figure(figsize=(8, 6))
categories = ['Testing Data']
accuracy_scores = [training_data_accuracy]

sns.displot(x=categories, y=accuracy_scores, color='red')
plt.xlabel('Dataset')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Score Comparison')
plt.ylim(0, 1)  # Setting y-axis limit to 0 to 1 for accuracy score
plt.show()
```

```
c:\Users\Dell\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)

<Figure size 800x600 with 0 Axes>
```

FIGURE8.4

COMPARISON BETWEEN TRAIN DATA AND TEST DATA



```
#comparison of train_data and test_data:

# Assuming you have calculated the accuracy scores for training and test data
training_data_accuracy = accuracy_score(y_train, Train_prediction)
testing_data_accuracy = accuracy_score(y_test, Test_prediction)

# Plotting the accuracy scores
plt.figure(figsize=(8, 6))
categories = ['Training Data', 'Testing Data']
accuracy_scores = [training_data_accuracy, testing_data_accuracy]

sns.displot(x=categories, y=accuracy_scores, palette=['lightgreen', 'red'])
plt.xlabel('Dataset')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Score Comparison')
plt.ylim(0, 1)  # Setting y-axis limit to 0 to 1 for accuracy score
plt.show()
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_126044\775920644.py:12: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  sns.displot(x=categories, y=accuracy_scores, palette=['lightgreen', 'red'])
c:\Users\Dell\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)

<Figure size 800x600 with 0 Axes>
```

FIGURE8.5

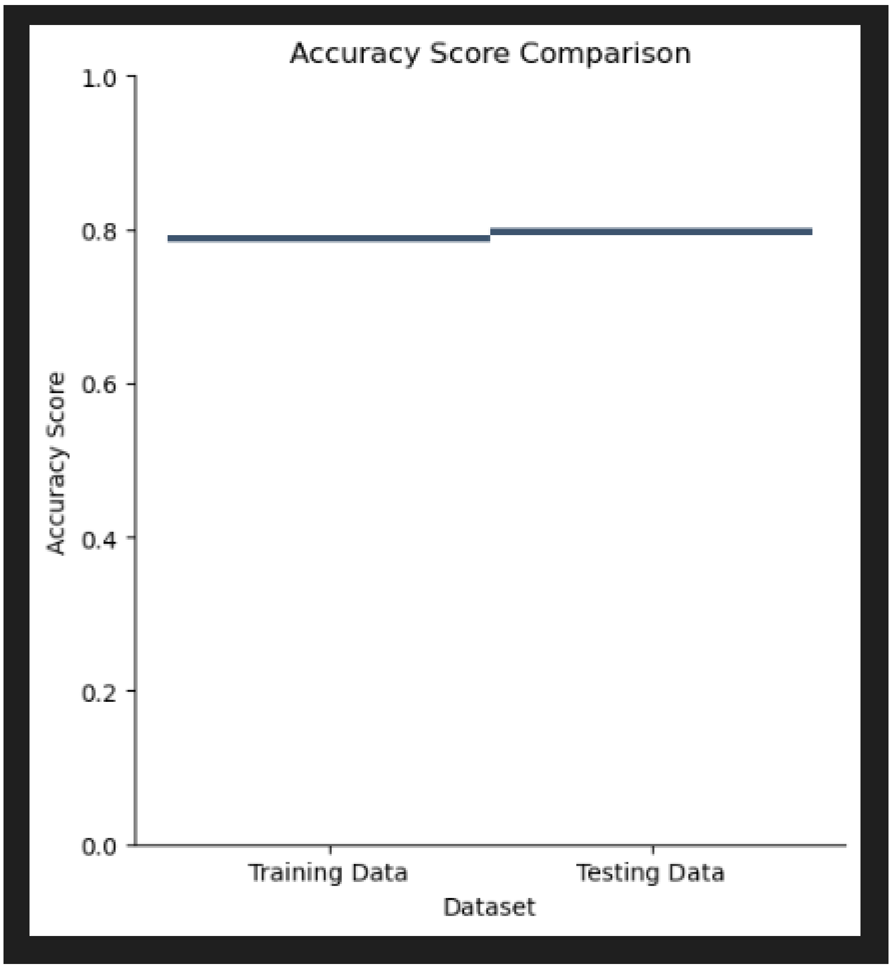
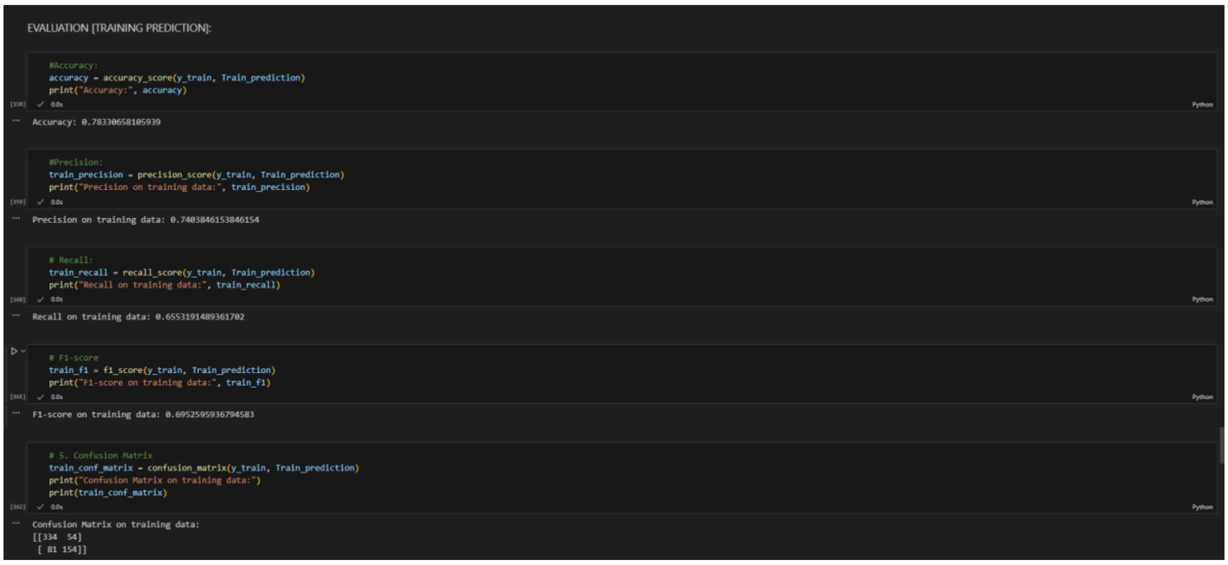


FIGURE8.6

9.EVALUATION

TRAINING DATA



EVALUATION [TRAINING PREDICTION]:

```
#Accuracy:
accuracy = accuracy_score(y_train, Train_prediction)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.78330658105939
```

```
#Precision:
train_precision = precision_score(y_train, Train_prediction)
print("Precision on training data:", train_precision)
```

```
Precision on training data: 0.7403846153846154
```

```
# Recall:
train_recall = recall_score(y_train, Train_prediction)
print("Recall on training data:", train_recall)
```

```
Recall on training data: 0.6553191489361702
```

```
# F1-score
train_f1 = f1_score(y_train, Train_prediction)
print("F1-score on training data:", train_f1)
```

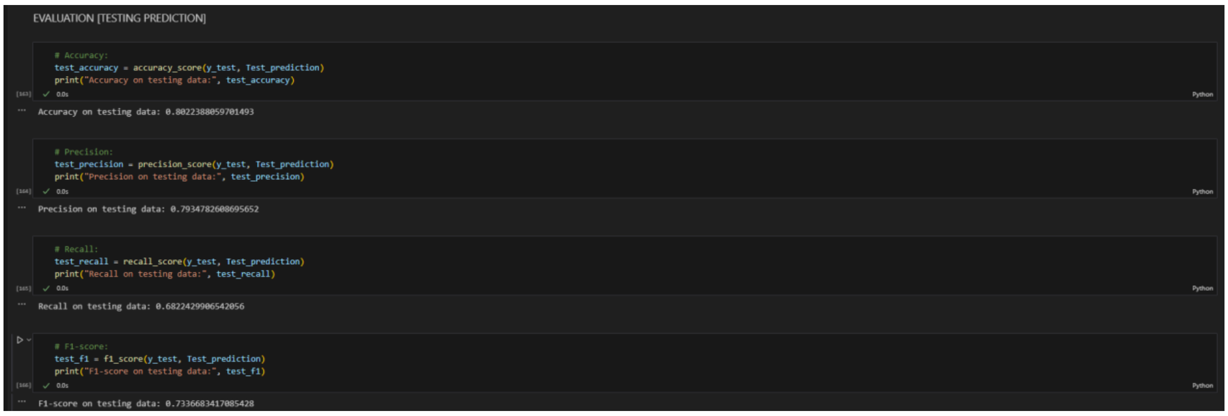
```
F1-score on training data: 0.6952595936794583
```

```
# 5. Confusion Matrix
train_conf_matrix = confusion_matrix(y_train, Train_prediction)
print("Confusion Matrix on training data:")
print(train_conf_matrix)
```

```
Confusion Matrix on training data:
[[334  54]
 [ 81 154]]
```

FIGURE9.1

TESTING DATA



EVALUATION [TESTING PREDICTION]

```
# Accuracy:
test_accuracy = accuracy_score(y_test, Test_prediction)
print("Accuracy on testing data:", test_accuracy)
```

```
Accuracy on testing data: 0.8022388059701493
```

```
# Precision:
test_precision = precision_score(y_test, Test_prediction)
print("Precision on testing data:", test_precision)
```

```
Precision on testing data: 0.7934782608695652
```

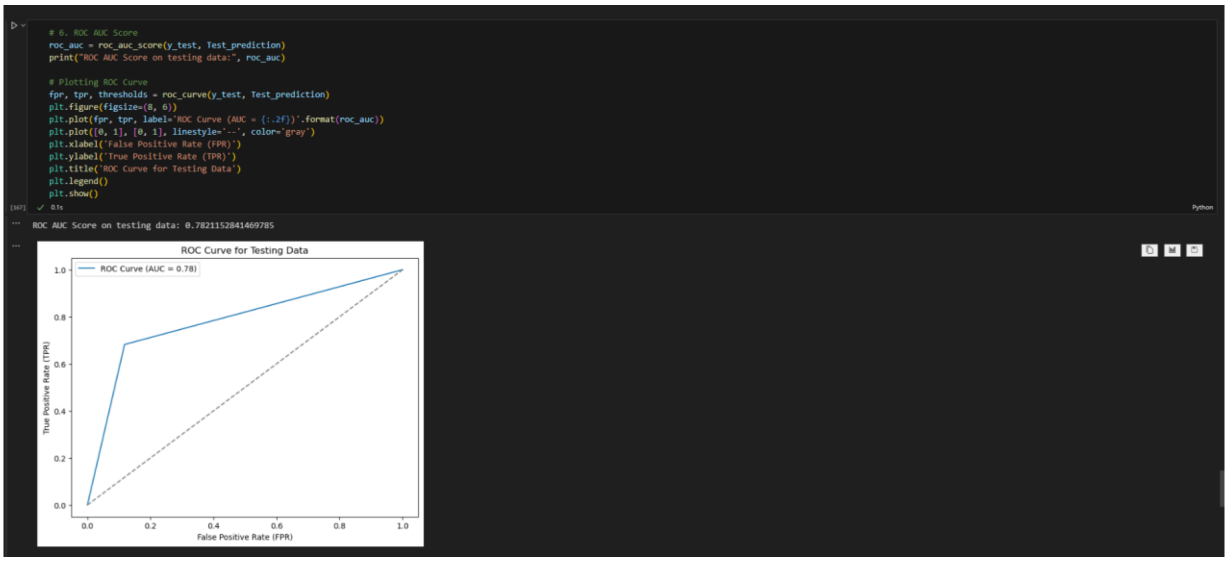
```
# Recall:
test_recall = recall_score(y_test, Test_prediction)
print("Recall on testing data:", test_recall)
```

```
Recall on testing data: 0.6822429906542056
```

```
# F1-score:
test_f1 = f1_score(y_test, Test_prediction)
print("F1-score on testing data:", test_f1)
```

```
F1-score on testing data: 0.7336683417085428
```

FIGURE9.2

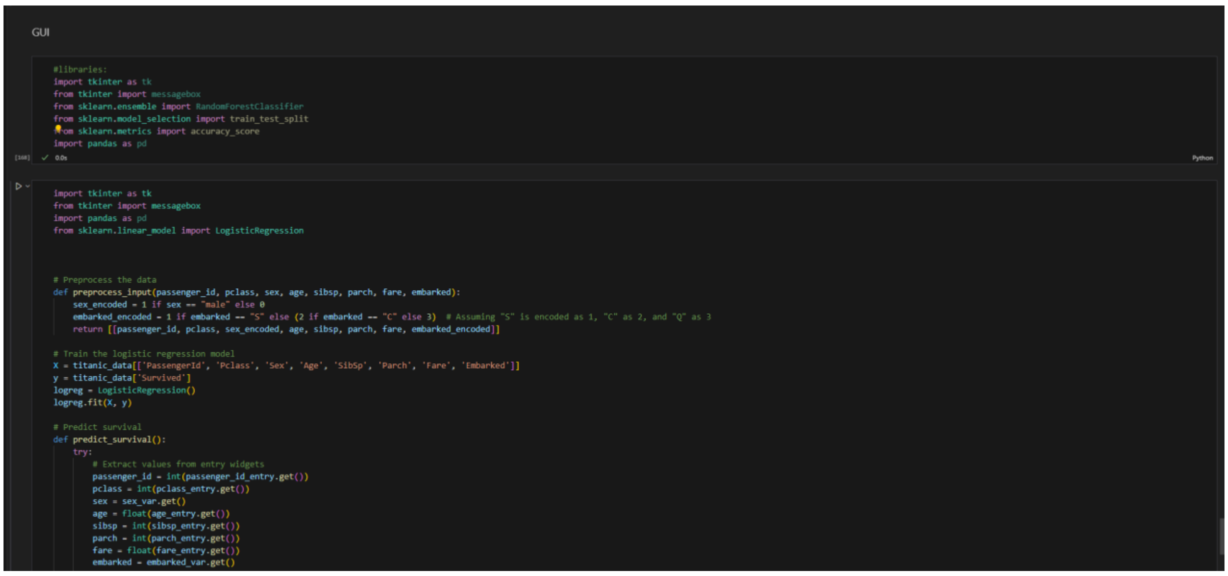


```
# 6. ROC AUC Score
roc_auc = roc_auc_score(y_test, Test_prediction)
print("ROC AUC Score on testing data:", roc_auc)

# Plotting ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, Test_prediction)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label='ROC Curve (AUC = {:.2f})'.format(roc_auc))
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve for Testing Data')
plt.legend()
plt.show()
```

```
ROC AUC Score on testing data: 0.7821152841469785
```

FIGURE 9.3

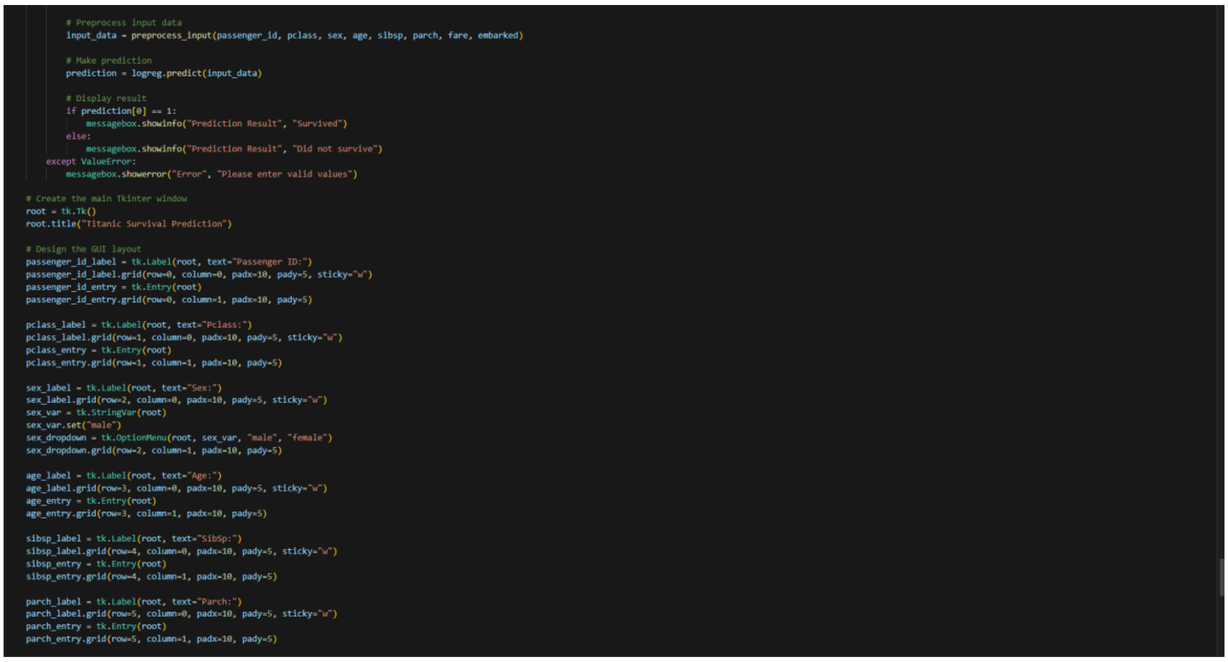
10.GUI [GRAPHICAL USER INTERFACE]



```python
#libraries:
import tkinter as tk
from tkinter import messagebox
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pandas as pd
```

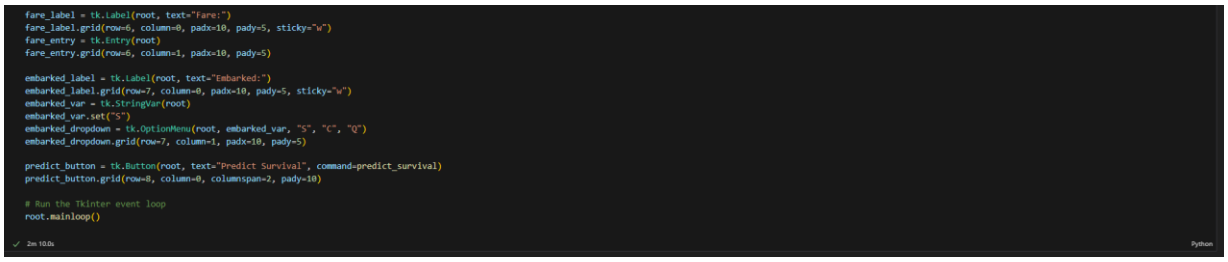
```python
import tkinter as tk
from tkinter import messagebox
import pandas as pd
from sklearn.linear_model import LogisticRegression



# Preprocess the data
def preprocess_input(passenger_id, pclass, sex, age, sibsp, parch, fare, embarked):
    sex_encoded = 1 if sex == "male" else 0
    embarked_encoded = 1 if embarked == "S" else (2 if embarked == "C" else 3)  # Assuming "S" is encoded as 1, "C" as 2, and "Q" as 3
    return [[passenger_id, pclass, sex_encoded, age, sibsp, parch, fare, embarked_encoded]]

# Train the logistic regression model
X = titanic_data[['PassengerId', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']]
y = titanic_data['Survived']
logreg = LogisticRegression()
logreg.fit(X, y)

# Predict survival
def predict_survival():
    try:
        # Extract values from entry widgets
        passenger_id = int(passenger_id_entry.get())
        pclass = int(pclass_entry.get())
        sex = sex_var.get()
        age = float(age_entry.get())
        sibsp = int(sibsp_entry.get())
        parch = int(parch_entry.get())
        fare = float(fare_entry.get())
        embarked = embarked_var.get()
```

FIGURE10.1



```python
        # Preprocess input data
        input_data = preprocess_input(passenger_id, pclass, sex, age, sibsp, parch, fare, embarked)

        # Make prediction
        prediction = logreg.predict(input_data)

        # Display result
        if prediction[0] == 1:
            messagebox.showinfo("Prediction Result", "Survived")
        else:
            messagebox.showinfo("Prediction Result", "Did not survive")
    except ValueError:
        messagebox.showerror("Error", "Please enter valid values")

# Create the main Tkinter window
root = tk.Tk()
root.title("Titanic Survival Prediction")

# Design the GUI layout
passenger_id_label = tk.Label(root, text="Passenger ID:")
passenger_id_label.grid(row=0, column=0, padx=10, pady=5, sticky="w")
passenger_id_entry = tk.Entry(root)
passenger_id_entry.grid(row=0, column=1, padx=10, pady=5)

pclass_label = tk.Label(root, text="Pclass:")
pclass_label.grid(row=1, column=0, padx=10, pady=5, sticky="w")
pclass_entry = tk.Entry(root)
pclass_entry.grid(row=1, column=1, padx=10, pady=5)

sex_label = tk.Label(root, text="Sex:")
sex_label.grid(row=2, column=0, padx=10, pady=5, sticky="w")
sex_var = tk.StringVar(root)
sex_var.set("male")
sex_dropdown = tk.OptionMenu(root, sex_var, "male", "female")
sex_dropdown.grid(row=2, column=1, padx=10, pady=5)

age_label = tk.Label(root, text="Age:")
age_label.grid(row=3, column=0, padx=10, pady=5, sticky="w")
age_entry = tk.Entry(root)
age_entry.grid(row=3, column=1, padx=10, pady=5)

sibsp_label = tk.Label(root, text="SibSp:")
sibsp_label.grid(row=4, column=0, padx=10, pady=5, sticky="w")
sibsp_entry = tk.Entry(root)
sibsp_entry.grid(row=4, column=1, padx=10, pady=5)

parch_label = tk.Label(root, text="Parch:")
parch_label.grid(row=5, column=0, padx=10, pady=5, sticky="w")
parch_entry = tk.Entry(root)
parch_entry.grid(row=5, column=1, padx=10, pady=5)
```

FIGURE10.2



```python
fare_label = tk.Label(root, text="Fare:")
fare_label.grid(row=6, column=0, padx=10, pady=5, sticky="w")
fare_entry = tk.Entry(root)
fare_entry.grid(row=6, column=1, padx=10, pady=5)

embarked_label = tk.Label(root, text="Embarked:")
embarked_label.grid(row=7, column=0, padx=10, pady=5, sticky="w")
embarked_var = tk.StringVar(root)
embarked_var.set("S")
embarked_dropdown = tk.OptionMenu(root, embarked_var, "S", "C", "Q")
embarked_dropdown.grid(row=7, column=1, padx=10, pady=5)

predict_button = tk.Button(root, text="Predict Survival", command=predict_survival)
predict_button.grid(row=8, column=0, columnspan=2, pady=10)

# Run the Tkinter event loop
root.mainloop()
```

FIGURE10.3

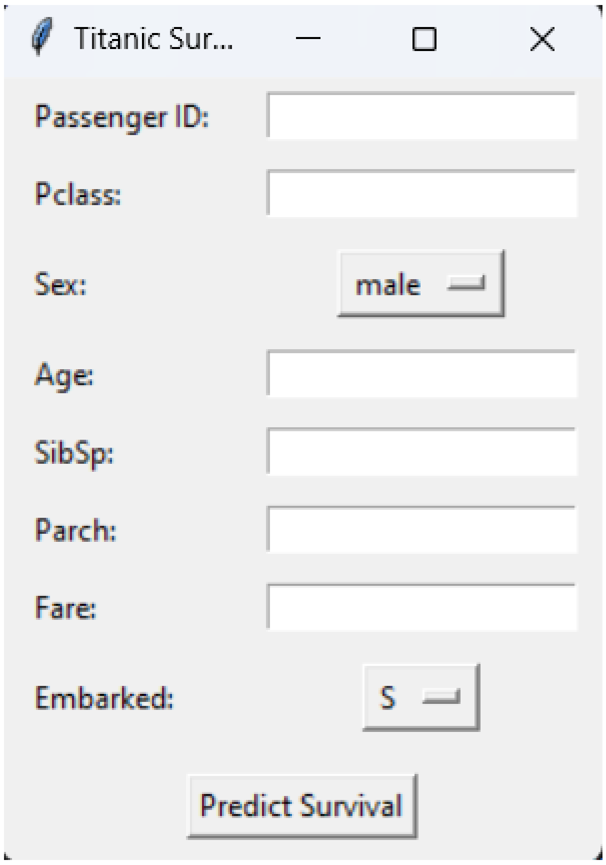
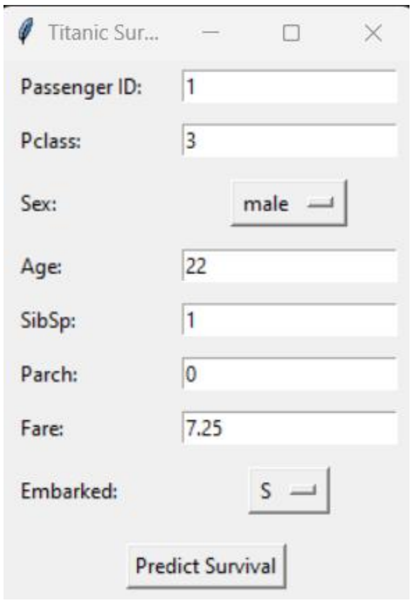


FIGURE 10.4



Titanic Sur...

Passenger ID: 1

Pclass: 3

Sex: male

Age: 22

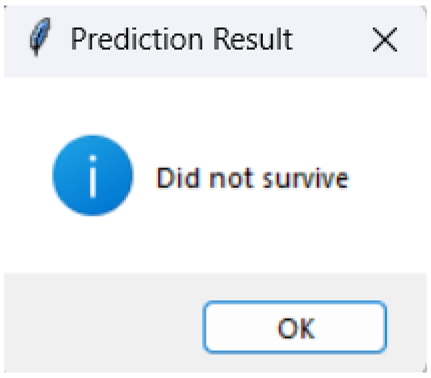
SibSp: 1

Parch: 0

Fare: 7.25

Embarked: S

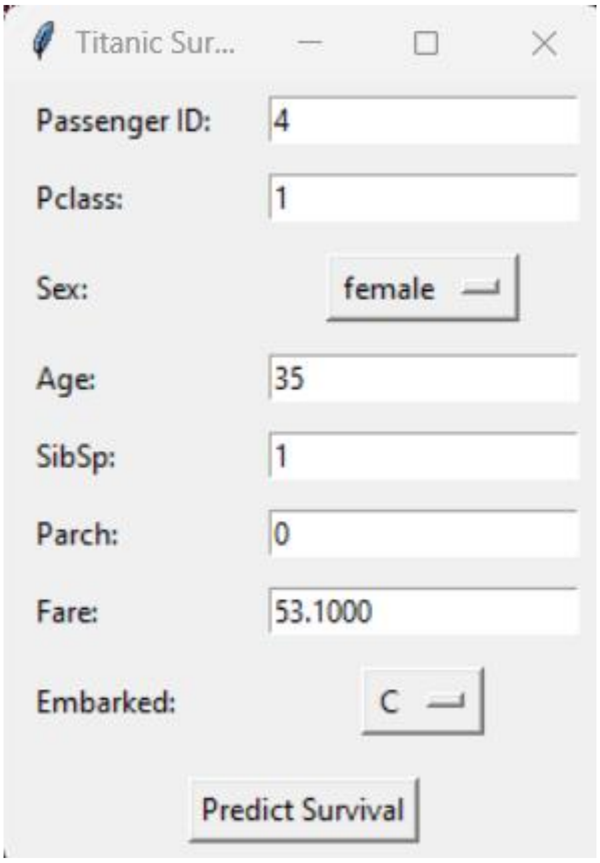
FIGURE 10.5



Prediction Result

Did not survive

FIGURE10.6



Titanic Sur...

Passenger ID: 4

Pclass: 1

Sex: female

Age: 35

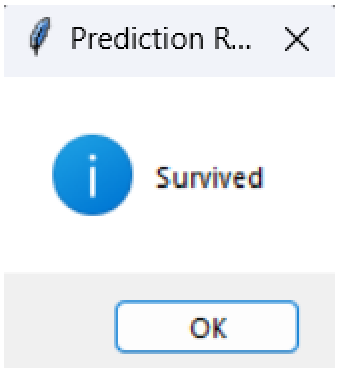
SibSp: 1

Parch: 0

Fare: 53.1000

Embarked: C

FIGURE 10.7



Prediction R...

Survived

FIGURE 10.8