Data Science Lab

CSEL-42--

Assignment on Exploratory Data Analysis

Submitted by

NISHAT MAHMUD
ID: B190305003

FAHIM HASAN
ID: B190305029

MD. WALIUL ISLAM RAYHAN
ID: B190305034

Submitted To

MD. MANOWARUL ISLAM, PHD
Associate Professor
Dept. of CSE
Jagannath University, Dhaka - 1100

The Titanic dataset is a classic dataset widely used in data science and machine learning for exploring relationships between features and outcomes. It provides detailed information about passengers on the Titanic, including demographic details, travel class, and survival status.

The primary goal of this assignment is to perform Exploratory Data Analysis (EDA) to uncover meaningful insights about the dataset. In addition, data preprocessing steps are included to handle missing values and transform features to enable more effective analysis. Key visualizations such as histograms, bar plots, and heatmaps are used to highlight trends and correlations in the data.

This document provides a detailed explanation of each code section, ensuring a clear understanding of the EDA process and its findings.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
%matplotlib inline
```

This section imports all the necessary libraries for data processing, visualization, and modeling. `%matplotlib inline` ensures that plots are displayed directly within the Jupyter Notebook.

```
# Load Titanic dataset
df = sns.load_dataset('titanic')
```

Loads the Titanic dataset from Seaborn's built-in datasets.

```
# Display the first five rows of the dataset
print(df.head())
```

Displays the first five rows of the dataset to preview its structure and contents.

```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

```
# Get a concise summary of the DataFrame
print(df.info())
```

Provides a concise summary of the dataset, including column data types and the presence of null values.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
None
```

```
# Display summary statistics for numerical columns
print(df.describe())
```

Displays statistical summaries such as mean, median, and standard deviation for numerical columns.

```
       survived      pclass         age       sibsp       parch        fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```

```python
# Check for missing values
print(df.isnull().sum())
```

Counts the number of missing values in each column to identify issues with incomplete data.

```
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

```python
# Impute missing values
imputer_median = SimpleImputer(strategy='median')
df['age'] = imputer_median.fit_transform(df[['age']])[:, 0]  # Flattened to 1D

imputer_mode = SimpleImputer(strategy='most_frequent')
df['embarked'] = imputer_mode.fit_transform(df[['embarked']])[:, 0]  # Flattened
df['embark_town'] = imputer_mode.fit_transform(df[['embark_town']])[:, 0]  # Flattened
```

Fills missing values in the 'age' column with the median value using the `SimpleImputer`. Also, replaces missing values in 'embarked' and 'embark_town' columns with the most frequently occurring value.

```python
# Drop 'deck' column due to excessive missing values
df.drop(columns=['deck'], inplace=True)
```

Drops the 'deck' column because it contains too many missing values, making it unreliable for analysis.

```
# Drop rows with missing 'alive' values
df.dropna(subset=['alive'], inplace=True)
```

Removes rows with missing values in the 'alive' column to maintain consistency.

```
# Feature Engineering: Add Family Size
df['family_size'] = df['sibsp'] + df['parch'] + 1
```

Creates a new feature 'family_size' by summing the number of siblings/spouses ('sibsp') and parents/children ('parch') on board, adding 1 for the passenger.

```
# Label Encoding for binary categorical variables
label_encoder = LabelEncoder()
df['sex'] = label_encoder.fit_transform(df['sex'])
```

Encodes the 'sex' column into numeric values (0 for female, 1 for male) using label encoding.

```
# One-hot encode categorical variables
df = pd.get_dummies(df, columns=['embarked', 'class', 'who', 'embark_town', 'alive'], drop_first=True)
```

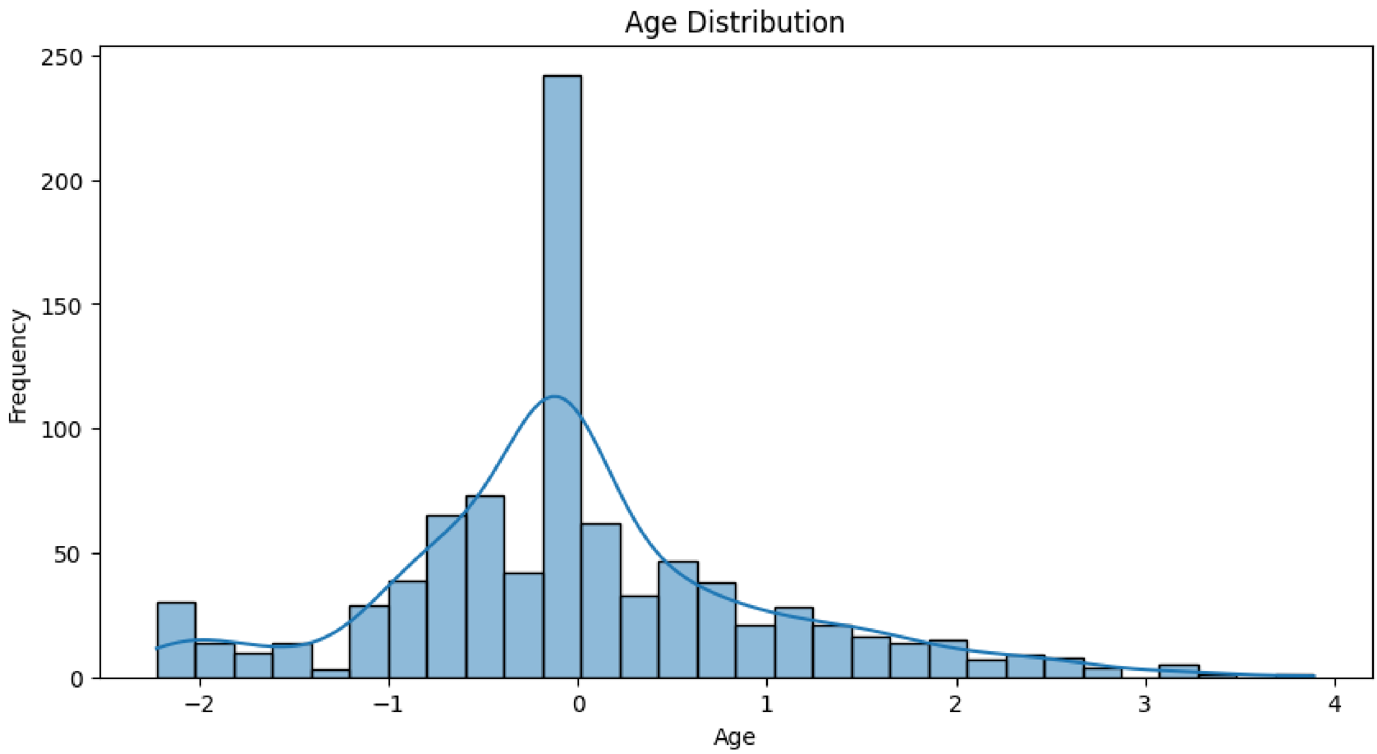
Performs one-hot encoding on categorical columns to convert them into numerical format, dropping the first category to avoid redundancy.

```
# Standardize numerical features
scaler = StandardScaler()
df[['age', 'fare']] = scaler.fit_transform(df[['age', 'fare']])
```

Standardizes the 'age' and 'fare' columns to have a mean of 0 and a standard deviation of 1, improving model performance.

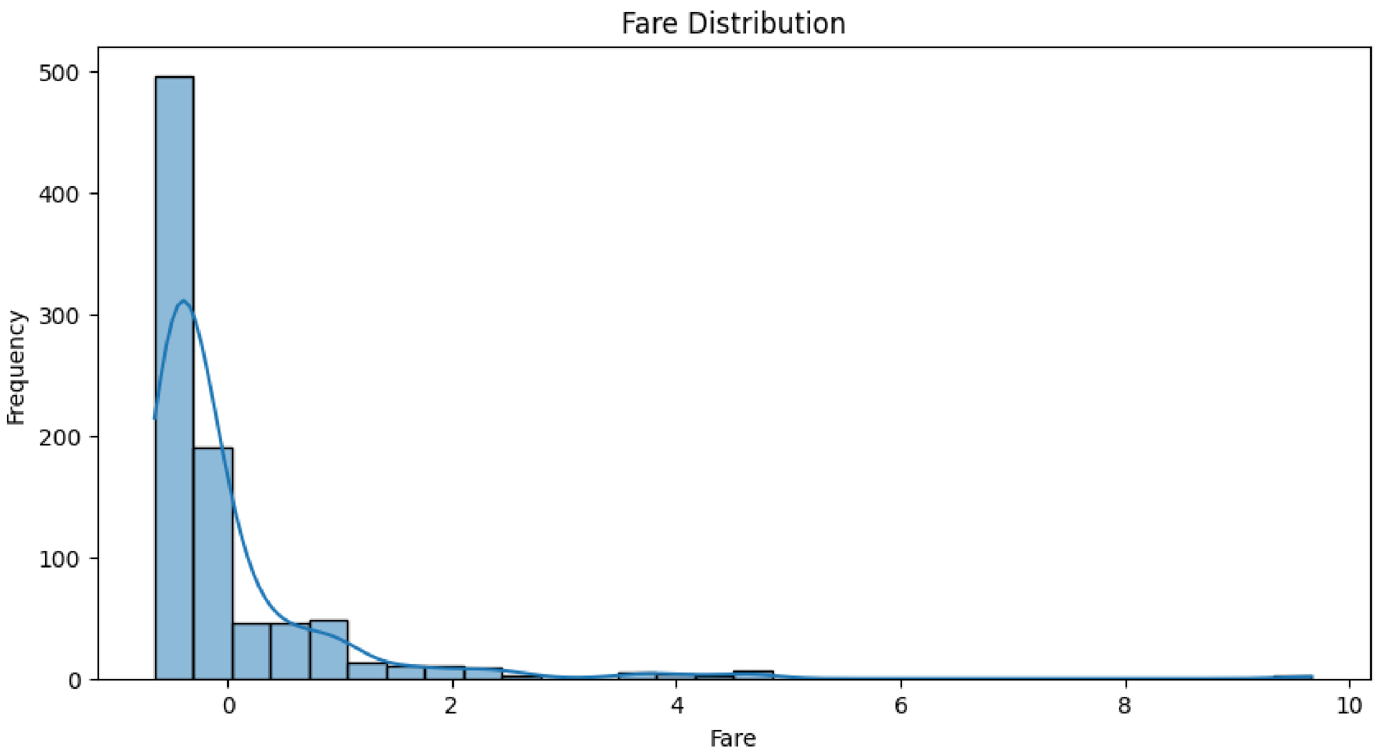
```
# EDA: Age distribution
plt.figure(figsize=(10, 5))
sns.histplot(df['age'], bins=30, kde=True)
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

Plots the distribution of 'age' using a histogram with a kernel density estimate (KDE) curve to visualize the spread of ages in the dataset.



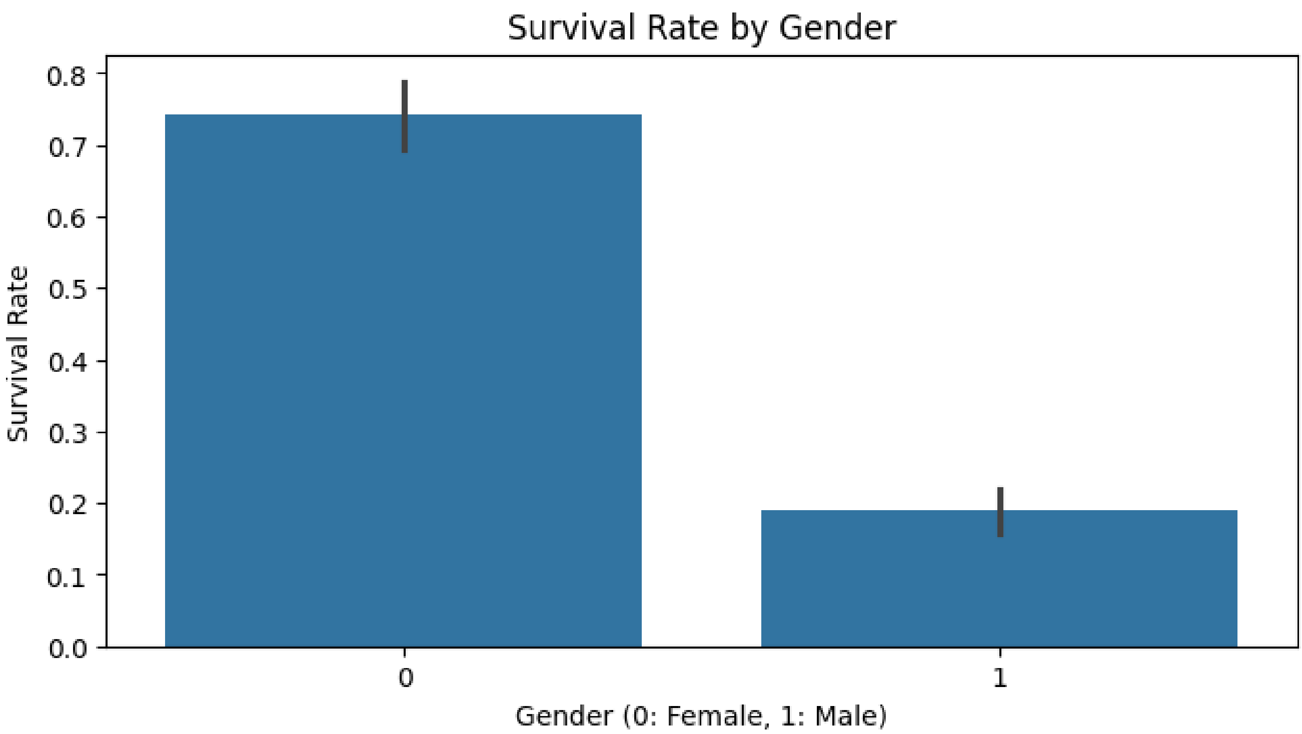

```
# EDA: Fare distribution
plt.figure(figsize=(10, 5))
sns.histplot(df['fare'], bins=30, kde=True)
plt.title('Fare Distribution')
plt.xlabel('Fare')
plt.ylabel('Frequency')
plt.show()
```

Plots the distribution of 'fare' to examine its spread and detect potential outliers.



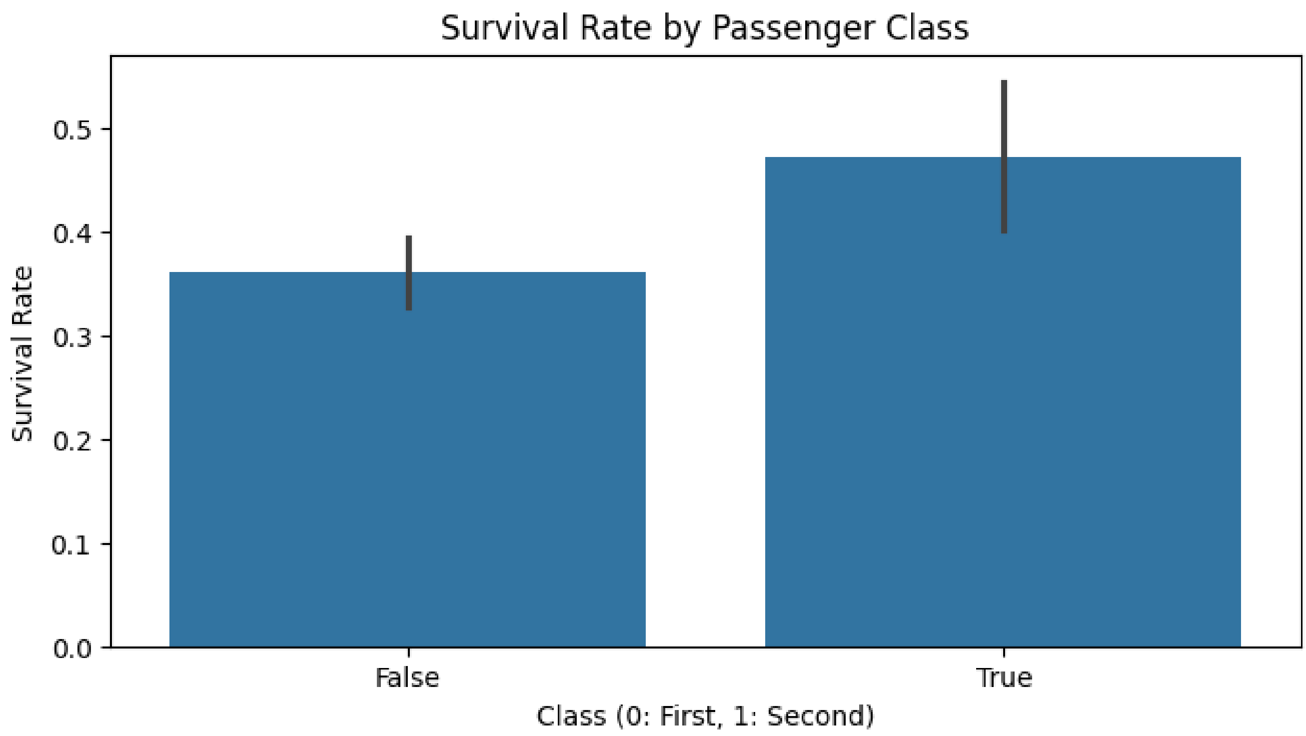

```
# EDA: Survival rate by gender
plt.figure(figsize=(8, 4))
sns.barplot(x='sex', y='survived', data=df)
plt.title('Survival Rate by Gender')
plt.xlabel('Gender (0: Female, 1: Male)')
plt.ylabel('Survival Rate')
plt.show()
```

Creates a bar plot to show survival rates for different genders.



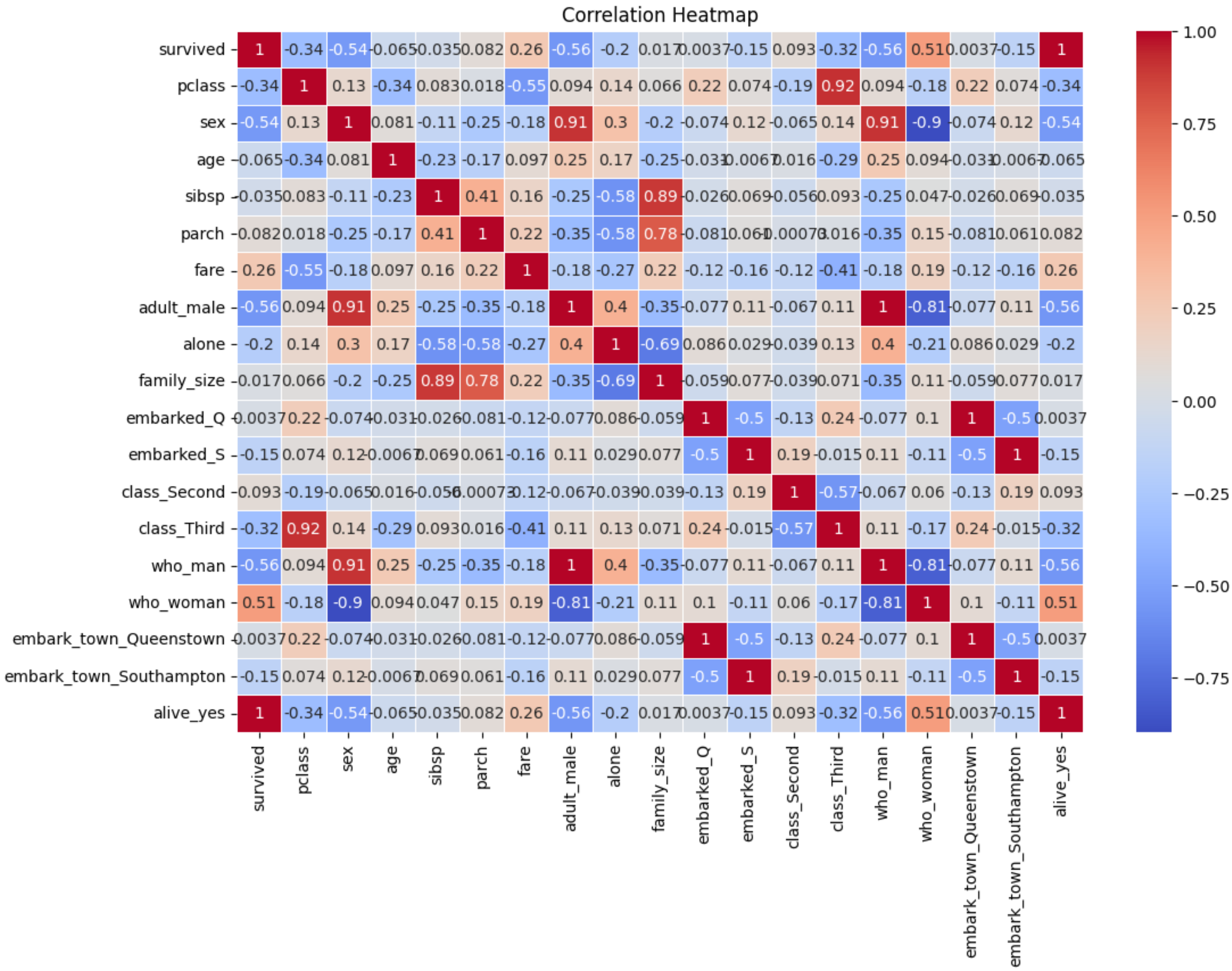
```python
# EDA: Survival rate by Passenger Class
plt.figure(figsize=(8, 4))
sns.barplot(x='class_Second', y='survived', data=df)
plt.title('Survival Rate by Passenger Class')
plt.xlabel('Class (0: First, 1: Second)')
plt.ylabel('Survival Rate')
plt.show()
```

Visualizes survival rates by passenger class to observe how class influenced survival.



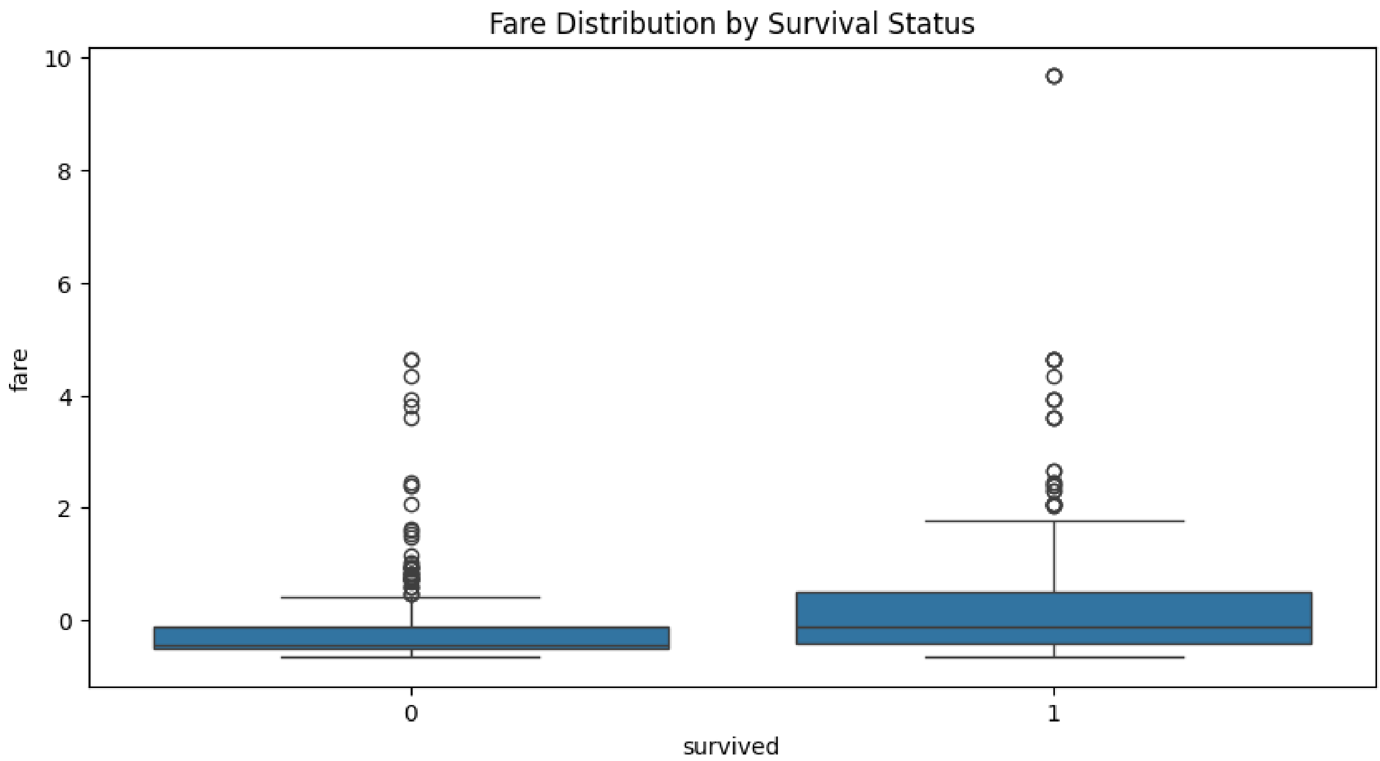

```python
# EDA: Correlation Heatmap
corr_matrix = df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

Displays a heatmap to visualize the correlation between features in the dataset.




```python
# EDA: Boxplot for Outliers in Fare
plt.figure(figsize=(10, 5))
sns.boxplot(data=df, x='survived', y='fare')
plt.title('Fare Distribution by Survival Status')
plt.show()
```

Uses a box plot to detect outliers in the 'fare' column, categorized by survival status.




```
# Modeling: Prepare data for Logistic Regression
X = df.drop('survived', axis=1)
y = df['survived']
```

Separates features (X) and target variable (y) for logistic regression.

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Splits the dataset into training (70%) and testing (30%) sets for model evaluation.

```
# Model Training
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

Initializes and trains a logistic regression model with a maximum of 1000 iterations for convergence.

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

```python
# Predictions
y_pred = model.predict(X_test)
```

Uses the trained model to predict survival outcomes for the test set.

```python
# Model Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

Evaluates model performance using accuracy, a classification report, and a confusion matrix.

```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       157
           1       1.00      1.00      1.00       111

    accuracy                           1.00       268
   macro avg       1.00      1.00      1.00       268
weighted avg       1.00      1.00      1.00       268

Confusion Matrix:
[[157   0]
 [  0 111]]
```

Through the Exploratory Data Analysis (EDA) of the Titanic dataset, we uncovered valuable insights about survival rates and their relationship with features such as age, gender, fare, and passenger class.

The analysis also showcased the importance of data preprocessing, including handling missing values and feature engineering, to enable accurate and meaningful exploration. These insights can serve as a foundation for further predictive modeling or decision-making.