Exploratory Data Analysis

Dataset: Titanic

Dataset Link: https://github.com/RimshaIshfaq/Statistics-Assignments/blob/main/titanic_train.csv

Prepared by: Rimsha Ishfaq

1. Introduction

The analysis aimed to uncover patterns and relationships within the dataset, handle missing values, prepare data for modeling, and apply statistical methods to draw meaningful insights. The following steps were taken systematically:

Data Overview

- The dataset includes 12 columns, such as **PassengerID**, **Survived** (target variable), **Pclass**, **Sex**, **Age**, **SibSp**, **Parch**, **Fare**, **Cabin**, and **Embarked**.
- The structure, data types, and missing values were analyzed to guide preprocessing.

Distribution Analysis

- **Numerical Features**: Histograms and boxplots revealed the spread, outliers, and skewness in columns like Age and Fare.
- **Categorical Features**: Bar plots were used to analyze frequency distributions for variables like Pclass and Embarked.

Correlation Analysis

- A heatmap visualized missing values, identifying significant gaps in the Cabin and Age columns.
- A correlation matrix heatmap highlighted relationships between features, such as Pclass, Fare, and Survived, aiding feature selection.

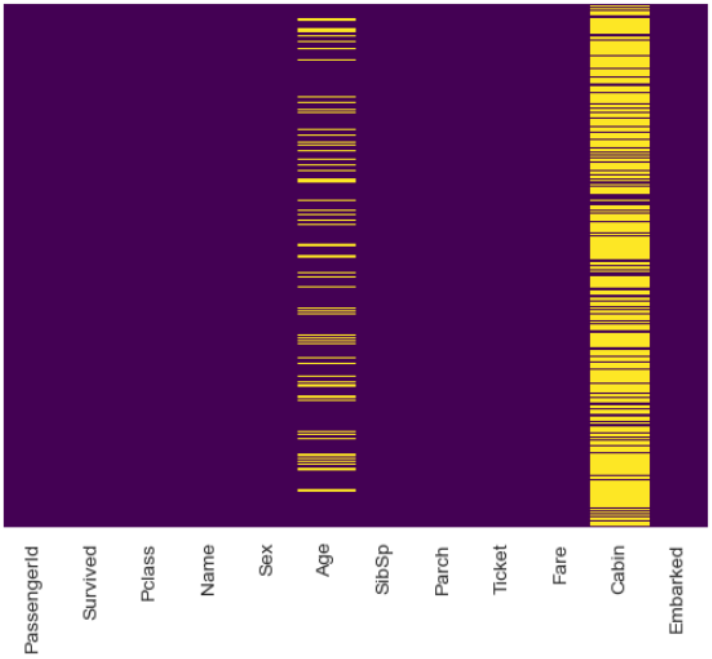

Figure 1: Heatmap of dataset to visualize missing values present

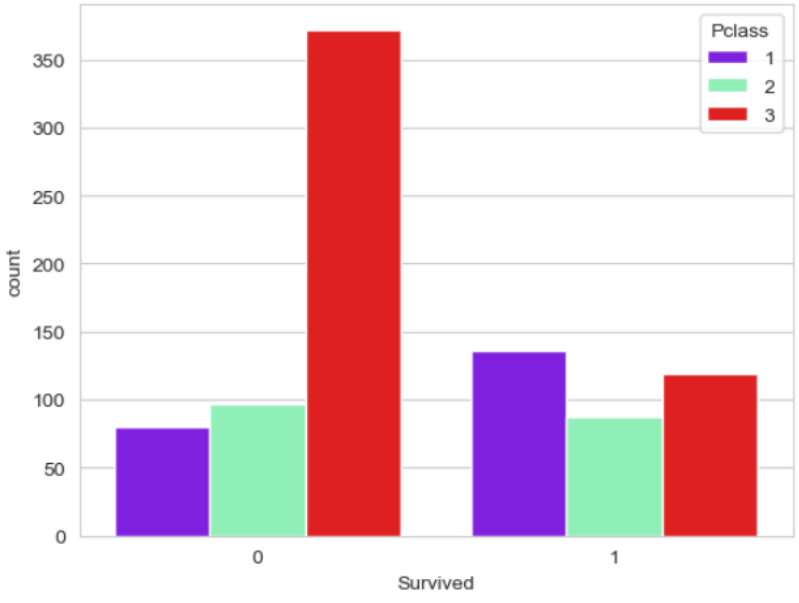

Figure 2: Count plot between Survived and Pclass

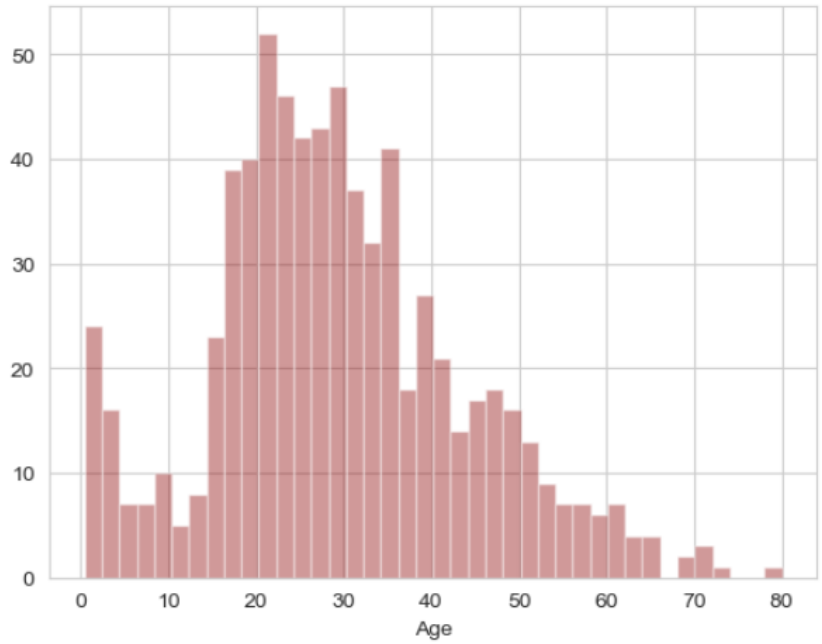

Figure 3: Age Distribution of Passengers

2. Handling Missing Values

To ensure the dataset's integrity and prevent biased results, missing values were addressed systematically:

Numerical Features

- **Columns with Missing Values**: The dataset had missing values in the Cabin and Age columns.
- **Cabin Column**: The Cabin column contained a significant number of missing values. Since it was not critical for the analysis, this column was dropped to reduce noise.
- **Age Column**: Missing values in the Age column were imputed instead of being dropped. To determine how to fill these missing values, a relationship between Pclass (passenger class) and Age was explored using a **box plot**.

  The box plot revealed:

  - **Outliers** in the Age distribution for each passenger class.
  - A clear relationship between Pclass and Age, indicating that passengers in higher classes tended to be older on average.

  Imputation Strategy:

  - Missing Age values were replaced with the mean age of passengers in the corresponding Pclass. This approach ensured that the imputed values reflected the underlying pattern in the data and maintained its consistency.

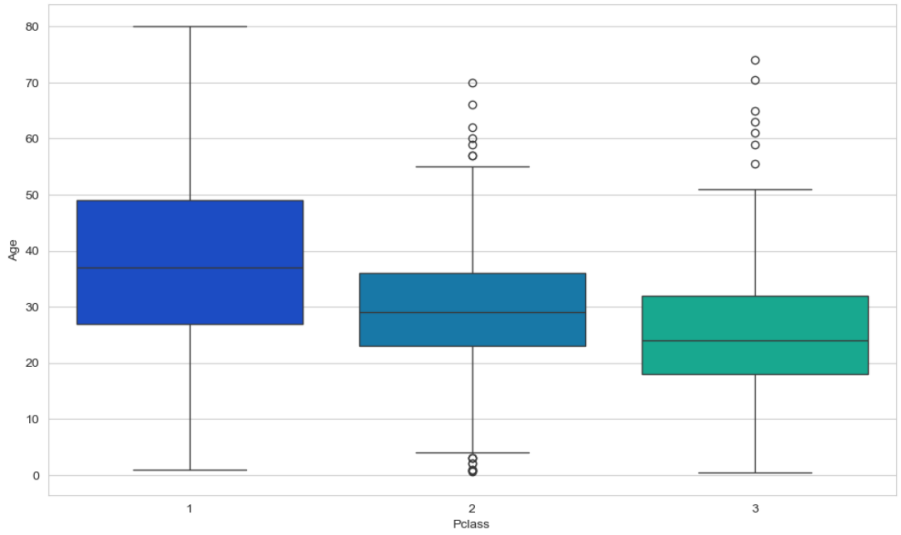

Figure 4: Heat-Map of Pclass by Age of Passengers

- **By using this approach, I** preserved the maximum amount of information by imputing instead of discarding rows, leveraged the relationship between Pclass and Age for more accurate imputations and dropped the highly incomplete Cabin column to streamline the dataset.

4. Feature Engineering

To ensure compatibility with machine learning algorithms, the categorical features in the dataset were transformed into numerical representations using the following steps:

One-Hot Encoding

- **Purpose**: Convert categorical variables into a numerical format to be used in machine learning models.
- **Method**: One-hot encoding was applied to categorical variables, creating separate binary columns for each unique category. For example:
    - The Embarked column was transformed into binary columns Q and S, representing two of the categories (with one dropped to avoid multicollinearity due to the drop first=True parameter).
    - Similarly, the Sex column was converted into a binary column to distinguish between the categories (e.g., male/female).

```
#using one_hot encoding to convert it into dummy variable
pd.get_dummies(train['Embarked'], drop_first=True).head()
```

	Q	S
0	False	True
1	False	False
2	False	True
3	False	True
4	False	True

Figure 5: One-Hot Encoding Technique (Feature Engineering)

This approach ensures that the machine learning model treats each category as distinct, preventing biased outcomes.

Cleanup of Unnecessary Columns

- After encoding, original categorical columns (Sex, Embarked) and irrelevant columns (Name, Ticket) were dropped to streamline the dataset.
- The final dataset retained only the necessary features for modeling, ensuring minimal noise and optimal readiness for machine learning algorithms.

5. Logistic Regression Model

Model Training:

- The dataset was split into **training** and **test sets** to ensure the model's performance could be evaluated on unseen data.
- Logistic regression was chosen due to its interpretability, efficiency, and suitability for binary classification tasks.
- During initial model fitting, a **ConvergenceWarning** was encountered because the maximum number of iterations was reached before convergence. To address this:
    - The max_iter parameter was increased to 10,000, allowing the solver sufficient iterations to converge.

Model Predictions:

- Predictions were made on the test dataset using the trained logistic regression model.

Model Evaluation:

- The model's performance was assessed using the following metrics:
    - **Accuracy Score:** The model achieved an accuracy of **77.6%**, indicating that approximately three-fourths of the predictions were correct.
    - **Confusion Matrix:**
        - The confusion matrix provided a breakdown of predictions:
            - **True Positives (135):** Correctly predicted survivors.
            - **True Negatives (73):** Correctly predicted non-survivors.
            - **False Positives (19):** Predicted survivors who did not survive.
            - **False Negatives (41):** Predicted non-survivors who actually survived.
        - This analysis highlights the balance between the model's sensitivity (recall for survivors) and specificity (recall for non-survivors).

```
#okay, let's try increasing no. of iterations
from sklearn.linear_model import LogisticRegression

logmodel = LogisticRegression(max_iter=10000)  # Increase max_iter from the default 100
logmodel.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=10000)
```

```
predictions = logmodel.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix
```

```
accuracy=confusion_matrix(y_test,predictions)
accuracy
```

```
array([[135,  19],
       [ 41,  73]], dtype=int64)
```

Figure 6: Logistic Regression

Feature Coefficients:

- Feature coefficients from the logistic regression model offered insights into the impact of each variable on the target outcome. Features with larger coefficients (in absolute value) had a stronger influence on the likelihood of survival.

6. Inferential Statistics

Statistical techniques were applied to validate relationships and refine the model. A random sampling method was used to extract 10% of the data as a representative sample. Correlation analysis was conducted to assess the relationship between the features **Survived** and **Fare**:

- **Sampling Approach:**
    - Random sampling ensured the sample was unbiased and representative of the dataset's characteristics. The resulting sample consisted of 89 rows and included only the relevant columns, **Survived** and **Fare**.
- **Correlation Analysis:**
    - The Pearson correlation coefficient was calculated between the **Survived** and **Fare** columns.
    - Results from the correlation matrix:
        - **Correlation between Survived and Fare:** 0.3476
        - This indicates a moderate positive correlation, suggesting that individuals who paid higher fares were more likely to survive.

```
# Calculate the correlation between two numeric columns in the sample
correlation = sample_1.corr()

# Display the correlation matrix
print(correlation)
```

```
          Survived      Fare
Survived  1.000000  0.347596
Fare      0.347596  1.000000
```

Figure 7: Correlation Matrix

Key Findings:

- Pclass and Fare strongly influenced survival outcomes.
- One-hot encoding effectively transformed categorical variables into usable features for modeling.
- Logistic regression was a suitable model for this binary classification task, providing meaningful insights into the factors influencing survival.