Assignment No. 1

Problem Statement: Exploring data analysis (Various operations on dataset).

Objective: To perform Exploratory Data Analysis (EDA) and Preprocessing on a dataset to understand its structure, detect anomalies, and prepare it for machine learning models. The process includes handling missing data, analyzing correlations, applying encoding techniques, and visualizing data using charts and heatmaps.

Prerequisite :

1. A Python environment set up with libraries like pandas, xml.etree.ElementTree, and requests (for web access).
2. Internet connection (for reading datasets from the web).
3. Text editor and basic knowledge of python and EDA

Theory :

Exploratory Data Analysis (EDA) and Preprocessing

To build a well-performing machine learning model, it is essential to thoroughly explore and preprocess the dataset. The following steps ensure data quality, mitigate potential issues, and prepare the dataset for modeling.

1. **Understanding the Dataset**

Before proceeding with any modifications, it is important to analyze the dataset's structure and characteristics. This helps identify potential inconsistencies and decide on necessary preprocessing techniques.

- **Dataset Dimensions**
  1. The .shape function provides the total number of rows (samples) and columns (features).
  2. If the dataset is too large, feature selection techniques might be required to prevent overfitting. Conversely, smaller datasets may need augmentation strategies.

- **Data Types of Columns**
  1. Each column may contain numerical (integer/float) or categorical (string/object) values.
  2. The .info() function provides a summary of the data types, helping determine whether encoding is necessary.

- **Missing Values**
  1. Missing data can introduce biases, affecting model accuracy.
  2. The .isnull().sum() function helps count missing values per column, indicating whether imputation or removal is needed.

- **Basic Statistical Measures**
  1. Summary statistics such as mean, median, and standard deviation (via .describe()) provide an overview of data distribution.
  2. If distributions are skewed, applying transformations like log-scaling may be beneficial.

2. **Handling Missing Data**

Missing values should be addressed to ensure data completeness and prevent biased model training. Common approaches include:

- **Removing Missing Data**

1. If a feature contains more than 50-60% missing values, it may be dropped due to insufficient information.
2. Rows with missing values can also be removed if their count is minimal and does not significantly impact the dataset.

- **Imputation Techniques**
  1. **Numerical Data:**
     i. Mean imputation (for normally distributed data).
     ii. Median imputation (for skewed data).

  2. **Categorical Data:**
     i. Mode imputation (replacing with the most frequent category).

3. **Correlation Analysis**

Analyzing relationships between numerical features helps identify redundant variables, reducing the risk of multicollinearity.

- **Methods to Analyze Correlation:**

- **Pearson's Correlation Coefficient:**

    1. Measures the strength of linear relationships between numerical variables.
    2. Values range from **-1 to +1**, where:
        - ii. **+1** indicates a strong positive correlation.
        - iii. **-1** indicates a strong negative correlation.
        - iv. **0** means no correlation.

- **Heatmap Visualization:**

    1. A heatmap visually highlights highly correlated features, helping determine which ones to remove or merge.

4. Encoding Categorical Features

Since machine learning models operate on numerical data, categorical variables must be converted appropriately.

- **Common Encoding Techniques:**

    - **Label Encoding:**
        - i. Assigns a unique integer to each category.
        - ii. Suitable for ordinal variables (e.g., low < medium < high).

    - **One-Hot Encoding (OHE):**
        - i. Creates separate binary columns for each category.
        - ii. Ideal for nominal variables (e.g., gender, city names).

5. Data Visualization

Visualization helps in understanding patterns, distributions, and relationships within the data.

- **Commonly Used Plots:**

    - **Histograms:** Show the frequency distribution of numerical variables.
    - **Boxplots:** Help detect outliers in the dataset.
    - **Scatter Plots:** Illustrate relationships between two numerical variables.

6. **Feature Scaling and Normalization**

Scaling numerical features ensures uniformity and improves model performance.

- **Scaling Techniques:**

  - **Standardization (Z-score Normalization)**

    - Converts data to a standard distribution with zero mean and unit variance.
    - Useful for models like **linear regression, logistic regression, and PCA**.
    - **Formula:**

$$X' = \frac{X - \mu}{\sigma}$$

  - **Min-Max Scaling**

    - Scales values between 0 and 1.
    - Suitable for models such as **KNN and neural networks**.
    - **Formula:**

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

  - **Robust Scaling**

    - Uses the median and interquartile range (IQR) to handle outliers.
    - Recommended for datasets containing extreme values.
    - Formula:

$$X' = \frac{X - \text{Median}}{\text{IQR}}$$

Code & Output :

```
import pandas as pd
df= pd.read_csv("C:/Users/vishal/MachineLearning/Datasets/Titanic-Dataset.csv")
```

```
print(df.head())
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

```
# shape of the data
df.shape
```

```
(891, 12)
```

```python
df.tail(10)
```

```python
#data information
df.info()
```

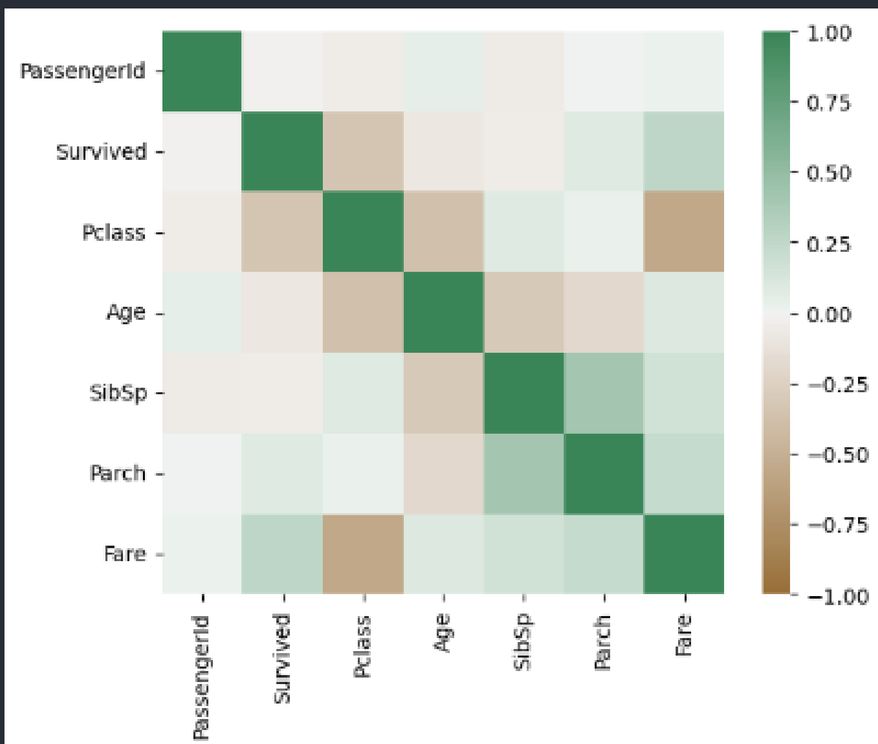
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    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
```

```python
# describing the data
df.describe()
```

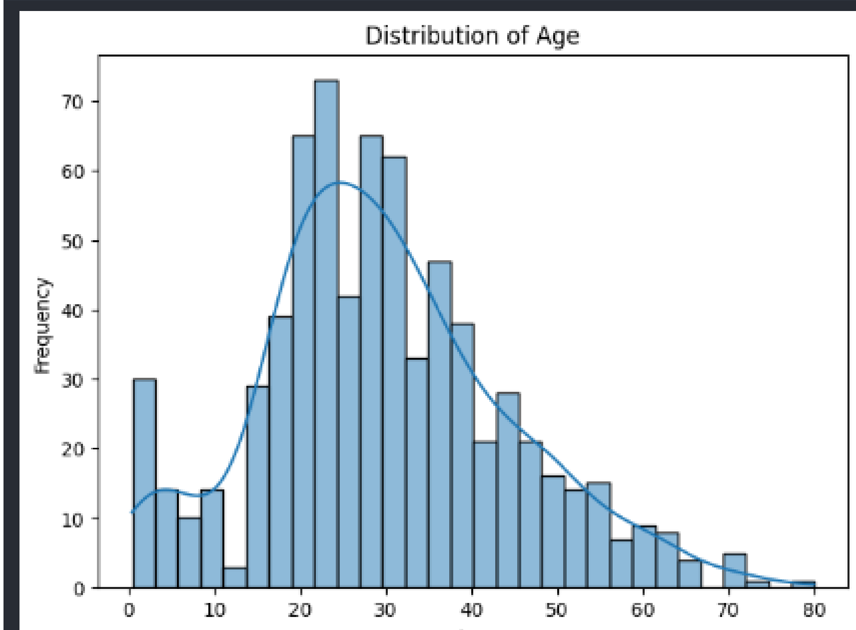
```python
Corr_Matrix = round(df.select_dtypes(include=[float, int]).corr(), 2)
print(Corr_Matrix)
```

```
             PassengerId  Survived  Pclass   Age  SibSp  Parch  Fare
PassengerId         1.00     -0.01   -0.04  0.04  -0.06  -0.00  0.01
Survived           -0.01      1.00   -0.34 -0.08  -0.04   0.08  0.26
Pclass             -0.04     -0.34    1.00 -0.37   0.08   0.02 -0.55
Age                 0.04     -0.08   -0.37  1.00  -0.31  -0.19  0.10
SibSp              -0.06     -0.04    0.08 -0.31   1.00   0.41  0.16
Parch              -0.00      0.08    0.02 -0.19   0.41   1.00  0.22
Fare                0.01      0.26   -0.55  0.10   0.16   0.22  1.00
```

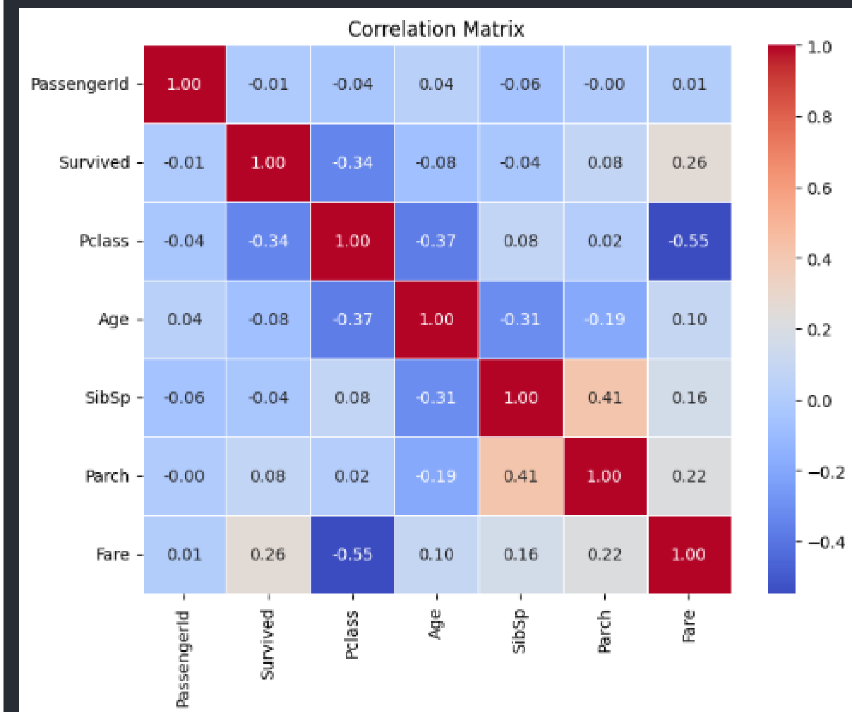
```python
import matplotlib.pyplot as plt
import seaborn as sns

axis_corr = sns.heatmap(
Corr_Matrix,
vmin=-1, vmax=1, center=0,
cmap=sns.diverging_palette(50, 500, n=500),
square=True
)

plt.show()
```



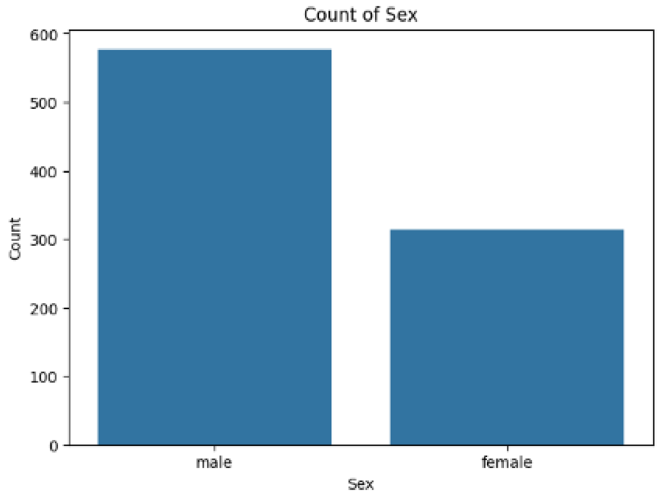
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Distribution of Age
plt.figure(figsize=(7, 5))
sns.histplot(df['Age'], kde=True, bins=30)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()

# Distribution of Fare
plt.figure(figsize=(7, 5))
sns.histplot(df['Fare'], kde=True, bins=30)
plt.title('Distribution of Fare')
plt.xlabel('Fare')
plt.ylabel('Frequency')
plt.show()
```

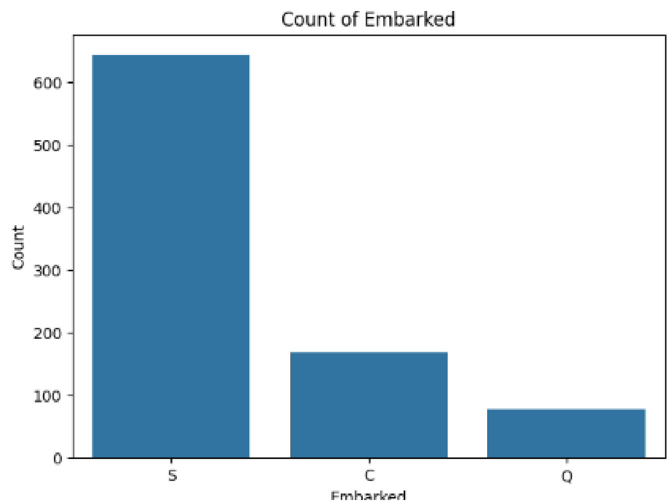


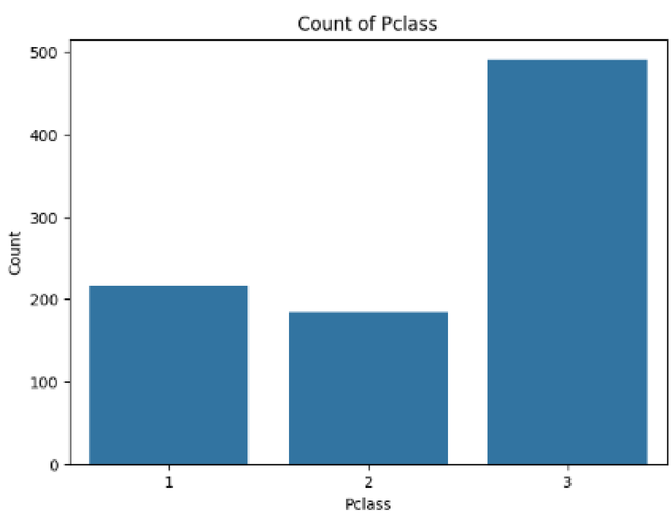
```python
# Correlation Matrix (only numerical columns)
corr_matrix = df.select_dtypes(include=[float, int]).corr()

plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```



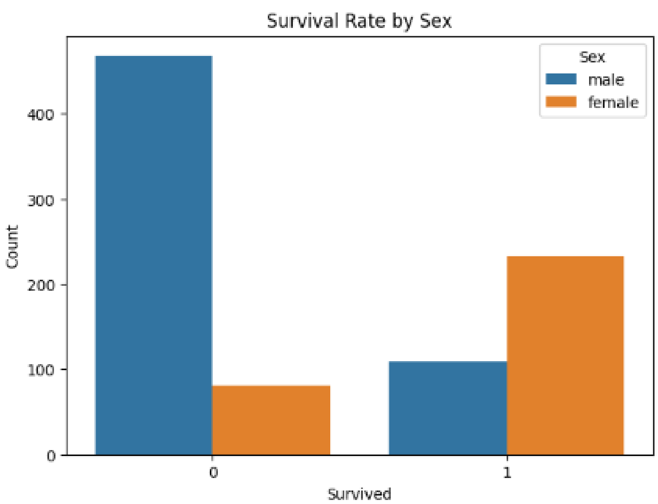

```python
# Count plot for Sex
plt.figure(figsize=(7, 5))
sns.countplot(x='Sex', data=df)
plt.title('Count of Sex')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.show()

# Count plot for Embarked
plt.figure(figsize=(7, 5))
sns.countplot(x='Embarked', data=df)
plt.title('Count of Embarked')
plt.xlabel('Embarked')
plt.ylabel('Count')
plt.show()

# Count plot for Pclass
plt.figure(figsize=(7, 5))
sns.countplot(x='Pclass', data=df)
plt.title('Count of Pclass')
plt.xlabel('Pclass')
plt.ylabel('Count')
plt.show()
```

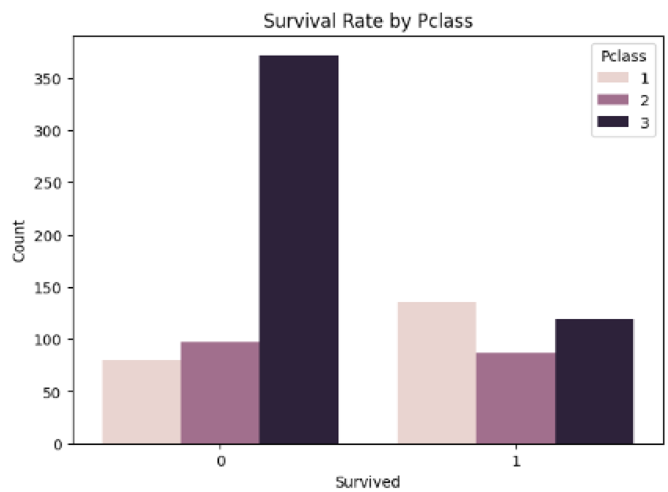

Count of Sex
Count
600
500
400
300
200
100
0
male
female
Sex

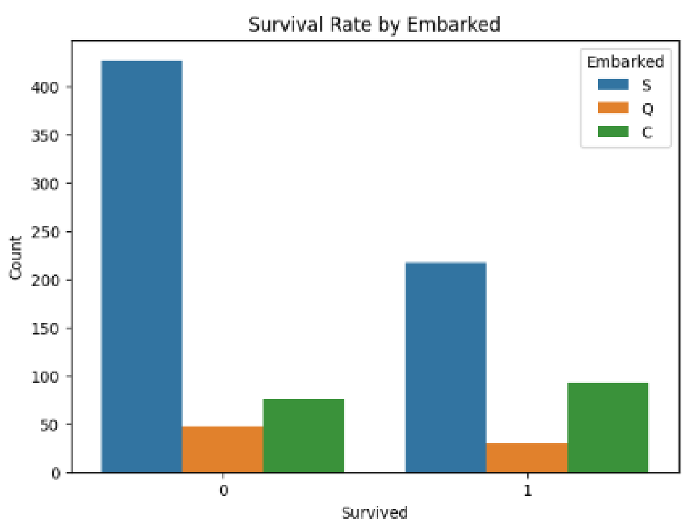

Count of Embarked
Count
600
500
400
300
200
100
0
S
C
Q
Embarked


Count of Pclass
Count
500
400
300
200
100
0
1
2
3
Pclass

```python
# Survival Rate by Sex
plt.figure(figsize=(7, 5))
sns.countplot(x='Survived', hue='Sex', data=df)
plt.title('Survival Rate by Sex')
plt.xlabel('Survived')
plt.ylabel('Count')
plt.show()

# Survival Rate by Pclass
plt.figure(figsize=(7, 5))
sns.countplot(x='Survived', hue='Pclass', data=df)
plt.title('Survival Rate by Pclass')
plt.xlabel('Survived')
plt.ylabel('Count')
plt.show()

# Survival Rate by Embarked
plt.figure(figsize=(7, 5))
sns.countplot(x='Survived', hue='Embarked', data=df)
plt.title('Survival Rate by Embarked')
plt.xlabel('Survived')
plt.ylabel('Count')
plt.show()
```







```python
# sum of missing values:
df.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
# Calculate the percentage of missing values for each column
missing_percentage = df.isnull().mean() * 100
print(missing_percentage)
```

```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

```python
#checking duplicate values
df.nunique()
```

```
PassengerId    891
Survived         2
Pclass           3
Name           891
Sex              2
Age             88
SibSp            7
Parch            7
Ticket         681
Fare           248
Cabin          147
Embarked         3
dtype: int64
```

```python
# Fill missing values in 'Age' with the median of the column
df['Age'] = df['Age'].fillna(df['Age'].median())

# Drop 'Cabin' as it has too many missing values and we don't have enough data to fill them
df.drop(columns=['Cabin'], inplace=True, errors='ignore')

# Fill missing values in 'Embarked' with the mode of the column
df['Embarked'] = df['Embarked'].fillna(df['Embarked'].mode()[0])
```

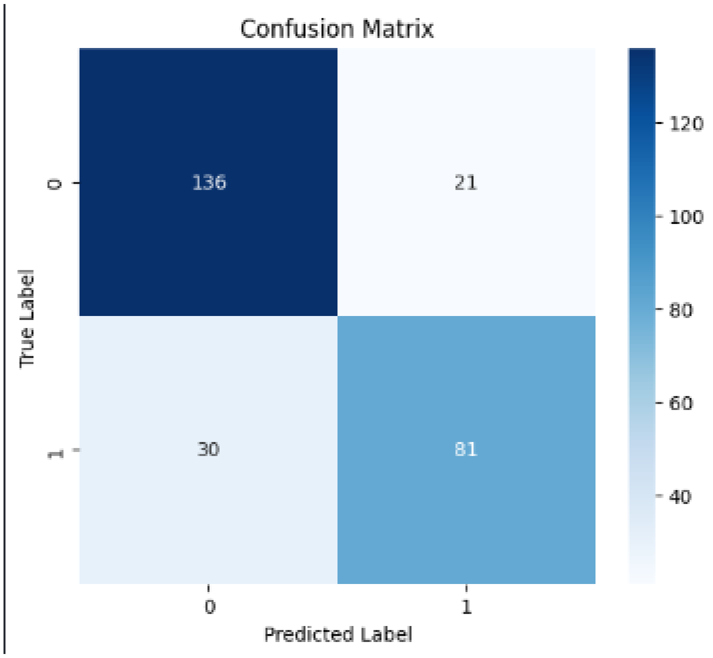
```python
# Convert categorical columns 'Sex, Embarked into numerical format using get_dummies
df = pd.get_dummies(df, columns=['Sex', 'Embarked'], drop_first=True)
```

```python
# Drop non-feature columns
X = df.drop(columns=['Survived', 'Name', 'Ticket', 'PassengerId'])
y = df['Survived']

# Split the dataset into 70% training and 30% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```python
# Training and testing
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```python
model = LogisticRegression(max_iter=1000)

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix:\n{conf_matrix}")
print(f"Classification Report:\n{class_report}")
```

```python
#Confusion Matrix Heatmap
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

conf_matrix = np.array([[136, 21], [30, 81]])
class_names = ['0', '1']

plt.figure(figsize=(6, 5))
sns.heatmap(conf_matrix, annot=True, fmt='g', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

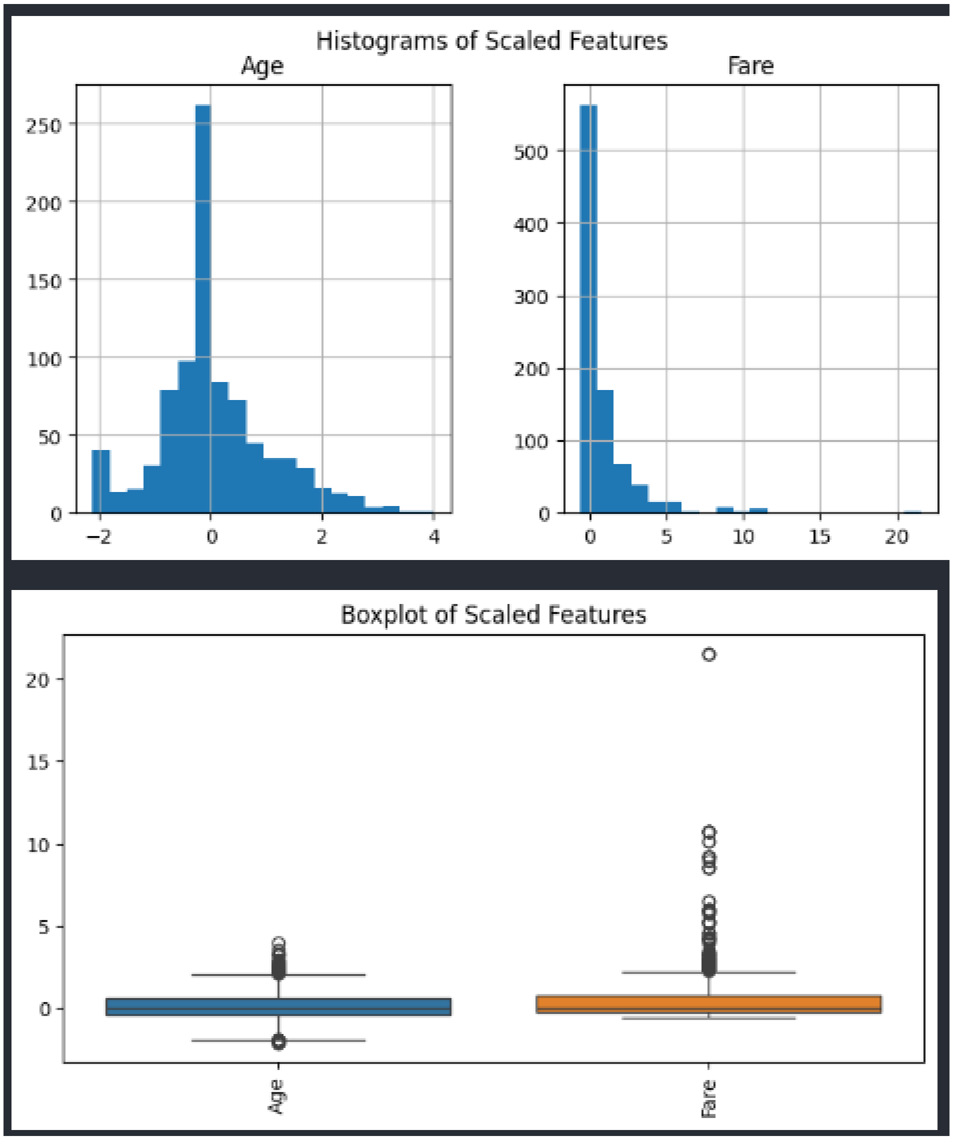



```python
from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler

# Standardization
scaler_standard = StandardScaler()
df[['Age', 'Fare']] = scaler_standard.fit_transform(df[['Age', 'Fare']])

# Min-Max Scaling
scaler_minmax = MinMaxScaler()
df[['Age', 'Fare']] = scaler_minmax.fit_transform(df[['Age', 'Fare']])

# Robust Scaling
scaler_robust = RobustScaler()
df[['Age', 'Fare']] = scaler_robust.fit_transform(df[['Age', 'Fare']])
```

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Check Descriptive Statistics
print("Descriptive Statistics after Scaling:\n", df[['Age', 'Fare']].describe())

# 2. Check Mean and Standard Deviation
print("\nMean Values:\n", df[['Age', 'Fare']].mean())
print("\nStandard Deviation:\n", df[['Age', 'Fare']].std())

# 3. Check Minimum and Maximum Values
print("\nMinimum Values:\n", df[['Age', 'Fare']].min())
print("\nMaximum Values:\n", df[['Age', 'Fare']].max())

# 4. Check Data Distribution Using Histograms
df[['Age', 'Fare']].hist(figsize=(8, 4), bins=20)
plt.suptitle("Histograms of Scaled Features")
plt.show()

# 5. Check Outliers Using Boxplots
plt.figure(figsize=(8, 4))
sns.boxplot(data=df[['Age', 'Fare']])
plt.xticks(rotation=90)
plt.title("Boxplot of Scaled Features")
plt.show()
```

```
Descriptive Statistics after Scaling:
              Age        Fare
count  891.000000  891.000000
mean     0.104737    0.768745
std      1.001515    2.152200
min     -2.121538   -0.626005
25%     -0.461538   -0.283409
50%      0.000000    0.000000
75%      0.538462    0.716591
max      4.000000   21.562738

Mean Values:
 Age     0.104737
Fare    0.768745
dtype: float64

Standard Deviation:
 Age     1.001515
Fare    2.152200
dtype: float64

Minimum Values:
 Age    -2.121538
Fare   -0.626005
dtype: float64
...
Maximum Values:
 Age      4.000000
Fare    21.562738
dtype: float64
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...



Github : **https://github.com/Vishalgodalkar**

Conclusion:

This EDA process involved inspecting the dataset, addressing missing values, encoding categorical variables, analyzing correlations, and applying feature scaling. Missing data was managed effectively, and duplicate features were identified. However, improper overwriting led to data corruption during scaling. To ensure accurate results, careful application of transformations is required. Overall, the preprocessing steps significantly improved data quality and prepared the dataset for machine learning applications.