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 :

Steps for EDA and Preprocessing

1. **Understanding the Dataset**

Before performing any analysis, it is crucial to explore the dataset structure and its contents. This helps in identifying potential issues and determining the necessary preprocessing steps. The key aspects to check include:

- **Number of Rows and Columns**
  1. The dataset size is checked using .shape, which gives the count of rows (samples) and columns (features).
  2. A large dataset may need feature selection to avoid overfitting, while a small dataset may require augmentation techniques.

- **Data Types of Columns**
  1. Different columns may have numerical (integer/float) or categorical (string/object) values.
  2. The .info() function provides an overview of data types, which helps determine if encoding is required.

- **Missing Values**
  1. Missing values can cause biases in model predictions.
  2. They are detected using .isnull().sum(), which counts the number of missing values per column.

- **Basic Statistical Measures**
    1. Measures like mean, median, and standard deviation (.describe()) provide insights into data distribution.
    2. Skewness in distributions may indicate the need for transformations such as log scaling.

2. **Handling Missing Data**
Missing values must be addressed to prevent biased model training. There are two main strategies:

- **Removal of Missing Data**
    1. If a column has more than **50-60% missing values**, it may be dropped as it lacks sufficient information.
    2. Rows with missing values may also be removed, but only if their number is small.

- **Imputation Techniques**

    1. **Numerical Data:** Replace missing values with:
        a. **Mean** (if data is normally distributed).
        b. **Median** (if data is skewed).
    2. **Categorical Data:** Replace with the **mode** (most frequent category).

3. Correlation Analysis
Correlation measures the relationship between numerical features. It helps in identifying redundant features that may lead to **multicollinearity**, negatively impacting model performance.

1. **Pearson's Correlation Coefficient**
   Values range from **-1 to +1**:
    - **+1**: Strong positive correlation (as one increases, the other increases).
    - **-1**: Strong negative correlation (as one increases, the other decreases).
    - **0**: No correlation.

2. **Heatmap Visualization**
   A heatmap helps identify highly correlated features, which can be removed or merged.

4.Encoding Categorical Features
Since machine learning models only work with numerical data, categorical features must be converted into numerical representations.

- **Encoding Techniques**
    1. **Label Encoding:** Assigns an integer to each category. Used for **ordinal** data (e.g., low < medium < high).

2. **One-Hot Encoding (OHE):** Creates binary columns for each category. Suitable for **nominal** data (e.g., gender, cities).

5.Data Visualization

- Key Types of Plots for EDA
    1. **Histograms:** Show the distribution of numerical variables.
    2. **Boxplots:** Identify outliers.
    3. **Scatter plots:** Show relationships between two numerical variables.

6. Feature Scaling and Normalization

Feature scaling ensures uniformity in numerical features, improving model performance.

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

    1. Transforms values to **zero mean** and **unit variance**.
    2. Formula: X′=X−μσX' = \frac{X - \mu}{\sigma}X′=σX−μ
    3. Suitable for models like **linear regression, logistic regression, and PCA**.

- **Min-Max Scaling**

    1. Scales values between **0 and 1**.
    2. Formula: X′=X−XminXmax−XminX' = \frac{X - X_{min}}{X_{max} - X_{min}}X′=Xmax−XminX−Xmin
    3. Used for models like **KNN and neural networks**.

- **Robust Scaling**

    1. Uses **median and IQR** to handle outliers.
    2. Formula: X′=X−MedianIQRX' = \frac{X - Median}{IQR}X′=IQRX−Median
    3. Best for **datasets with extreme values**.

Code & Output :

```
import pandas as pd
df= pd.read_csv("C:/Users/dnyan/ML Assignments/Dataset/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
```

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

```
(891, 12)
```

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

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
881	882	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.8958	NaN	S
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.5167	NaN	S
883	884	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000	NaN	S
884	885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	NaN	S
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
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

```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
```

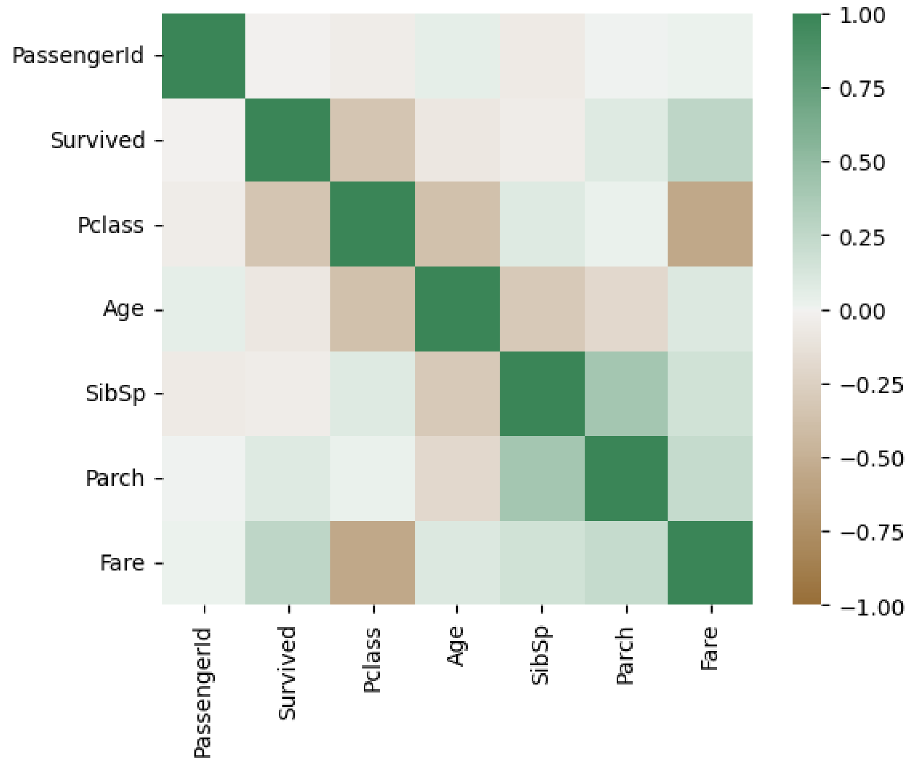
```python
# describing the data
df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

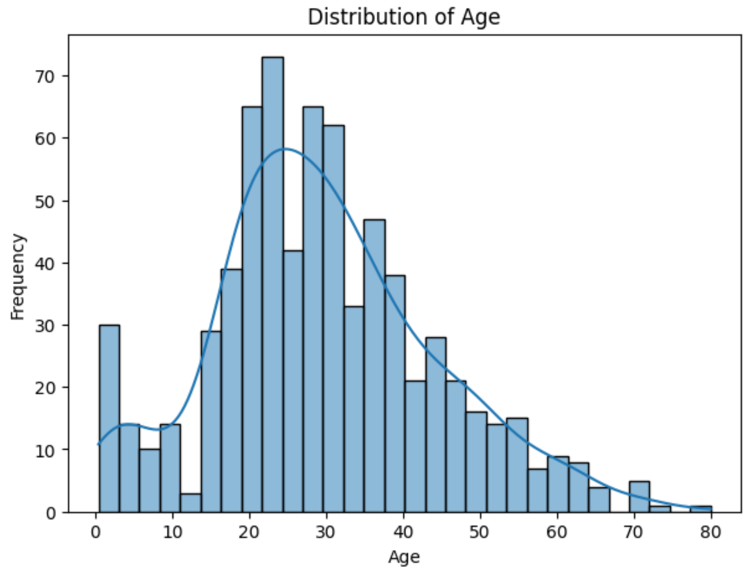
```python
Corr_Matrix = round(df.select_dtypes(include=[float, int]).corr(), 2)
print(Corr_Matrix)
```

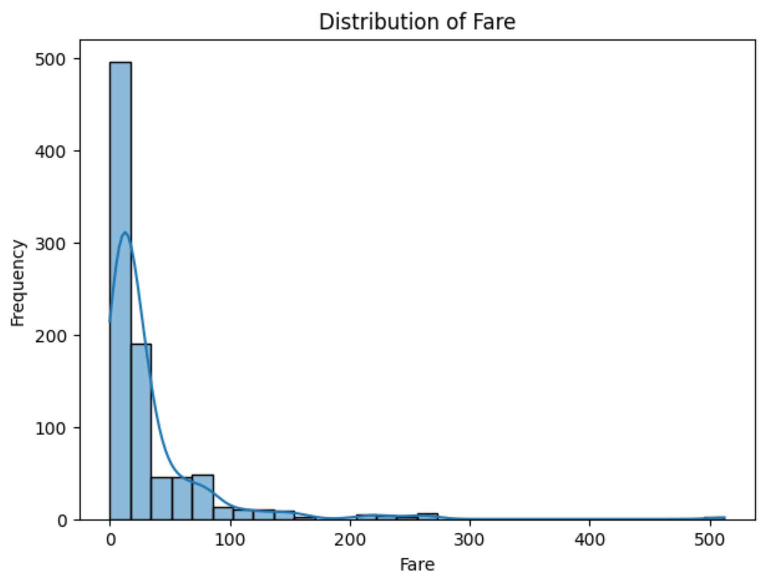
```
             PassengerId  Survived  Pclass   Age  SibSp  Parch  Fare
PassengerId         1.00     -0.01   -0.04  0.04  -0.06  -0.00  0.01
Survived           -0.01      1.00   -0.34 -0.08  -0.04   0.08  0.26
Pclass             -0.04     -0.34    1.00 -0.37   0.08   0.02 -0.55
Age                 0.04     -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()
```

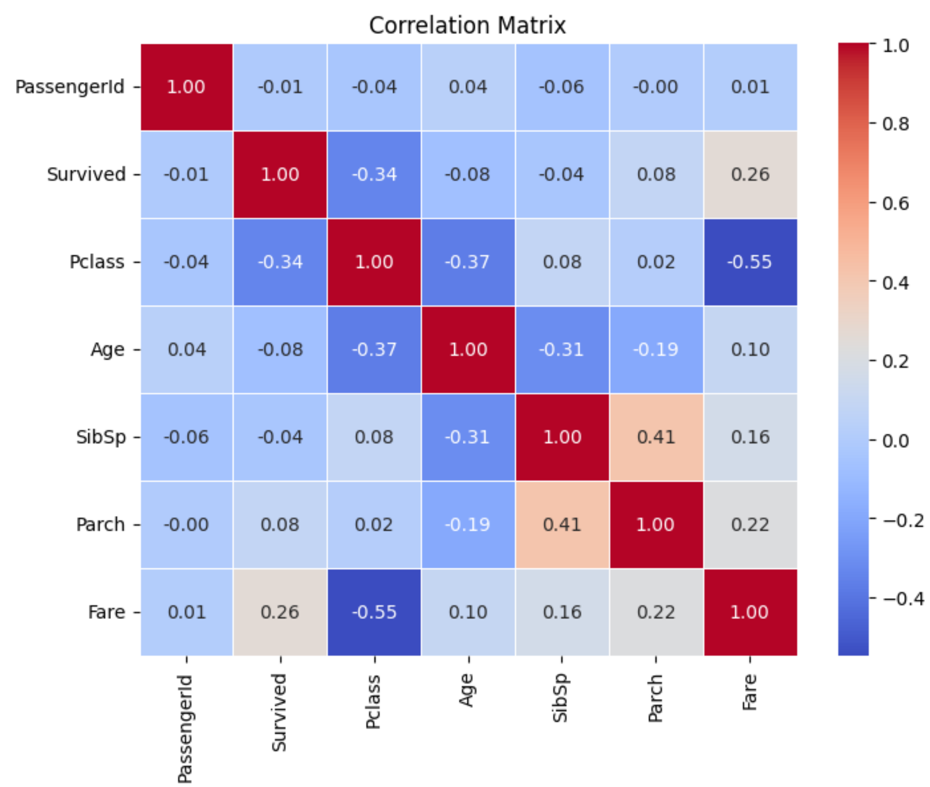



```python
import seaborn as sns
import matplotlib.pyplot as plt

# 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()
```

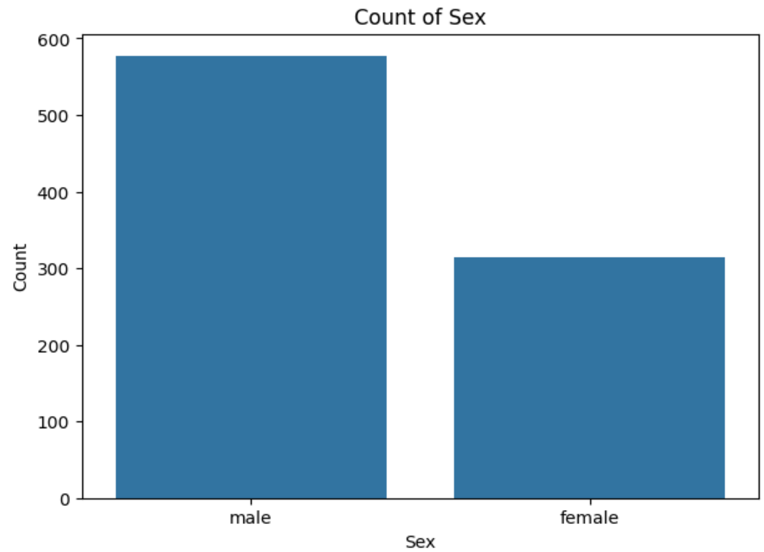


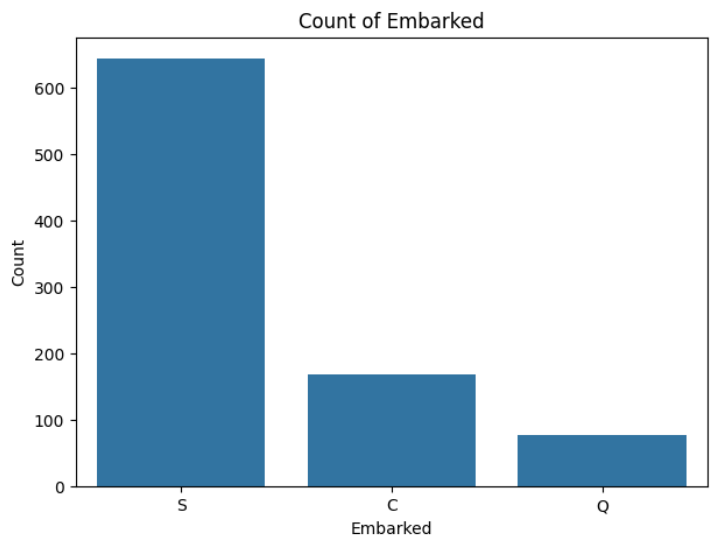


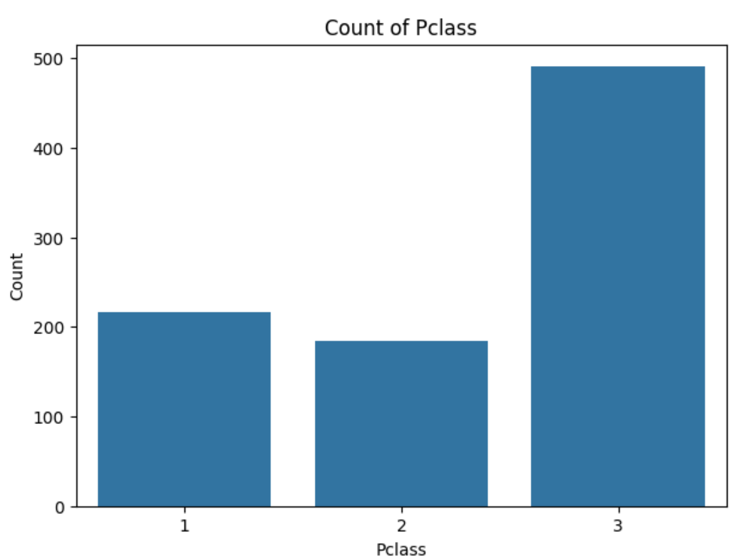

```
# Correlation Matrix (only numerical columns)
corr_matrix = df.select_dtypes(include=[float, int]).corr()

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



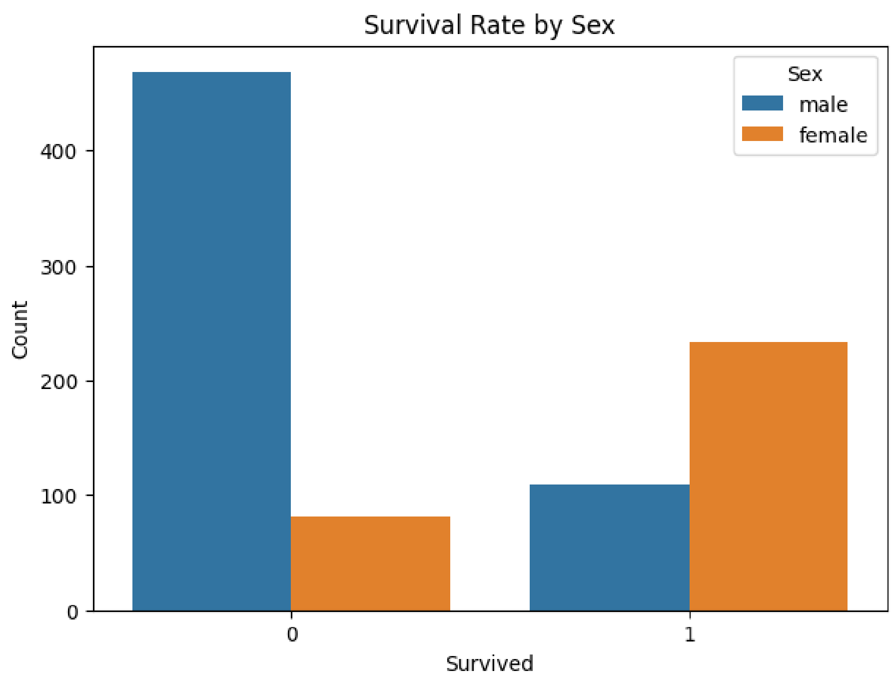

```python
# Count plot for Sex
plt.figure(figsize=(7, 5))
sns.countplot(x='Sex', data=df)
plt.title('Count of Sex')
plt.xlabel('Sex')
plt.ylabel('Count')
plt.show()

# 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()
```

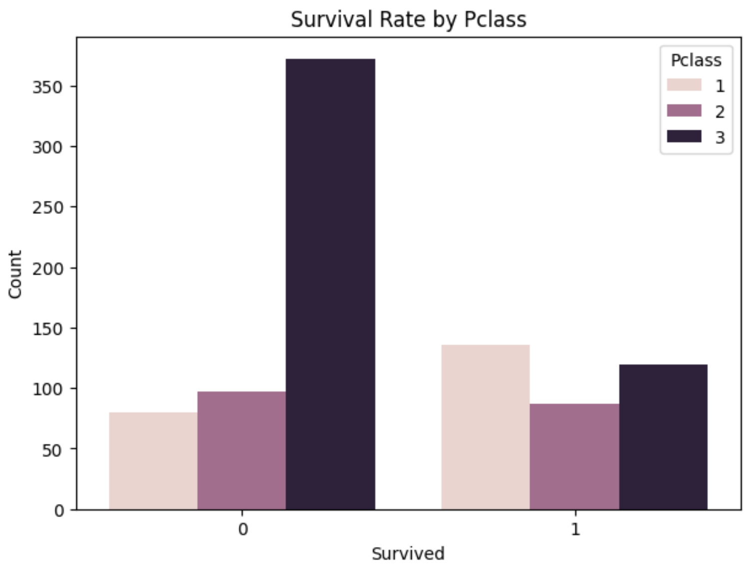

Count of Sex
Count
600
500
400
300
200
100
0
male
female
Sex

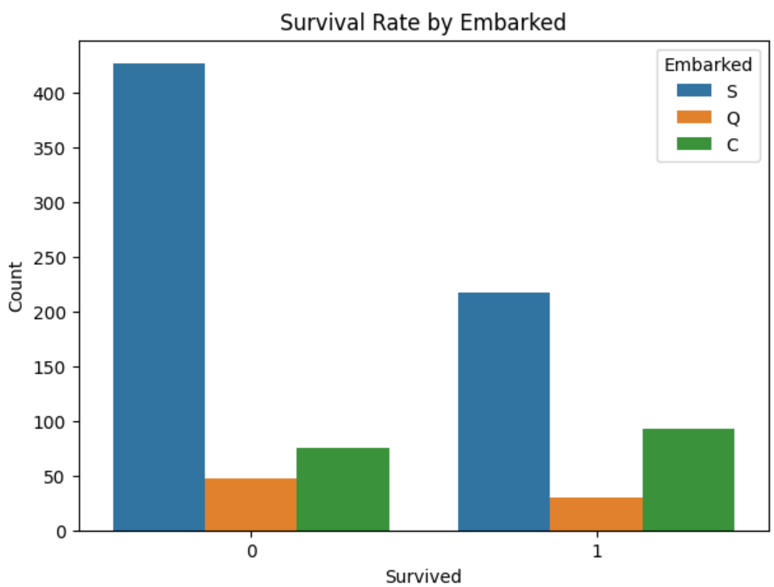

Count of Embarked
Count
600
500
400
300
200
100
0
S
C
Q
Embarked


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()
```








```
# 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
#we can drop the cabin column because it has too much missing values, more than 70%
```

```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
df.isnull().sum()
```

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 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          891 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  Embarked     891 non-null    object 
dtypes: float64(2), int64(5), object(4)
memory usage: 76.7+ KB
```

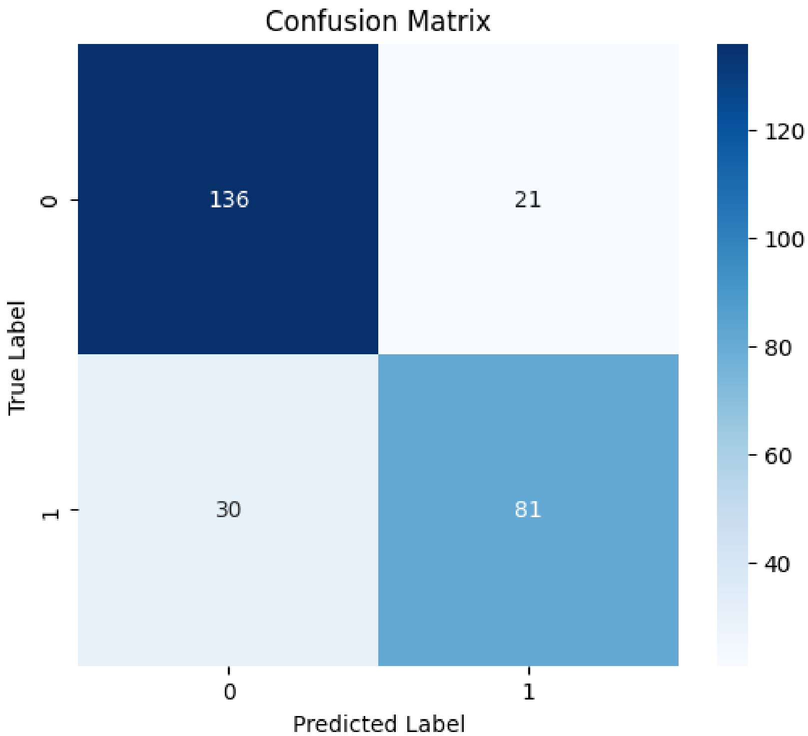
```
# Convert categorical columns 'Sex, Embarked into numerical format using get_dummies
df = pd.get_dummies(df, columns=['Sex', 'Embarked'], drop_first=True)
```

```
# 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)
```

```
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}")
```

```
Accuracy: 0.8097014925373134
Confusion Matrix:
[[136  21]
 [ 30  81]]
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.87      0.84       157
           1       0.79      0.73      0.76       111

    accuracy                           0.81       268
   macro avg       0.81      0.80      0.80       268
weighted avg       0.81      0.81      0.81       268
```

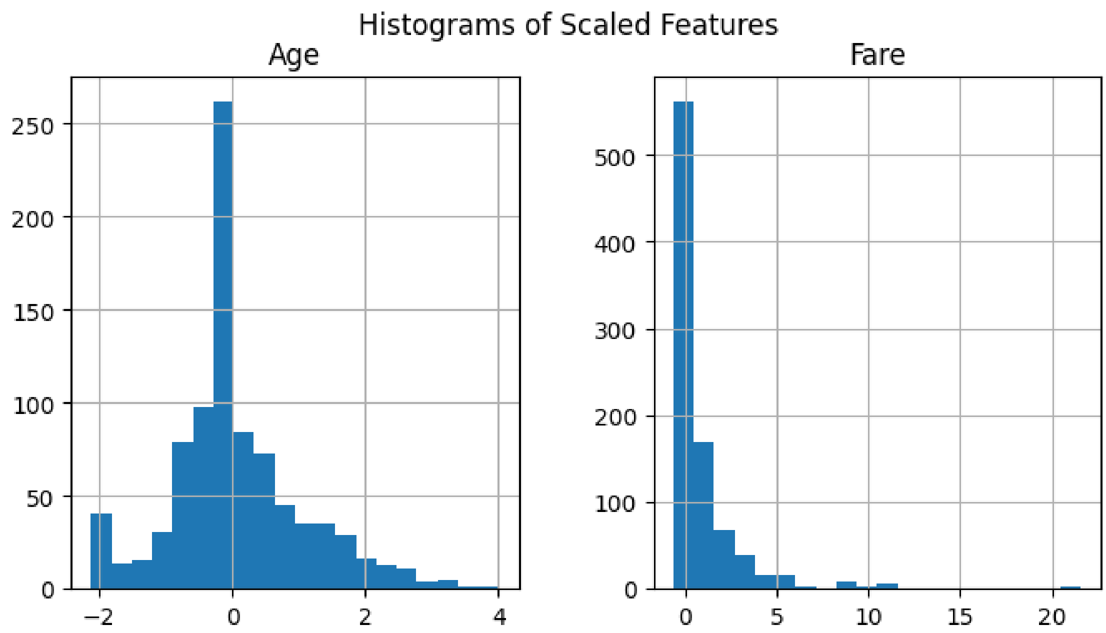
```python
#Confusion Matrix Heatmap
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

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()
```

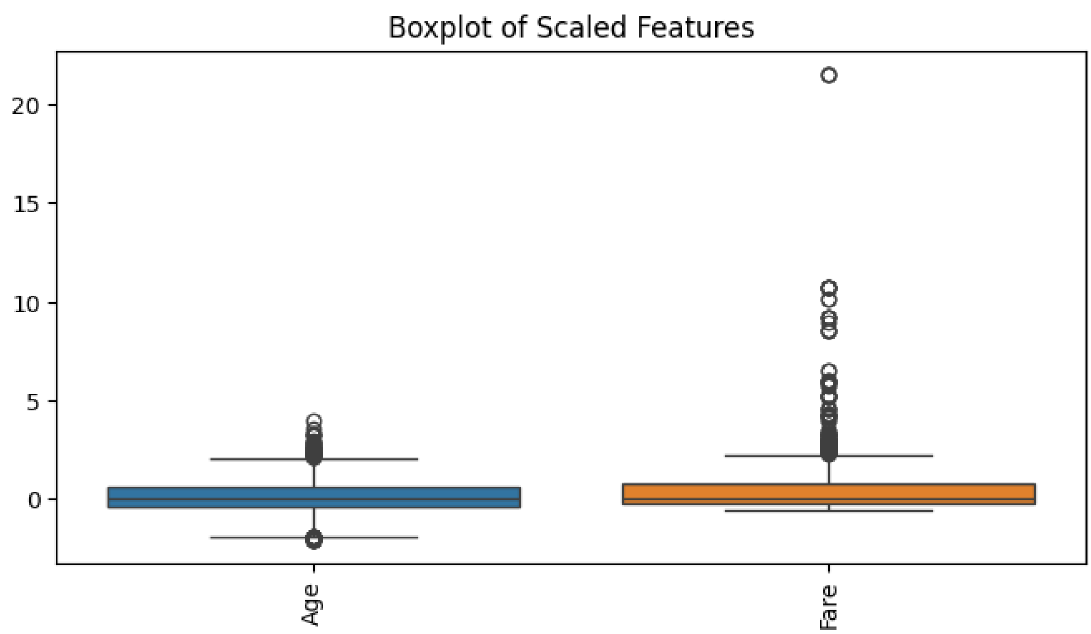



```python
from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler

# 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
```





Github :- https://github.com/sahilb-official/machinelearninglab

Conclusion:

This EDA task covered data inspection, missing value handling, categorical feature encoding, correlation analysis, and feature scaling. Missing values were handled, categorical variables were encoded, and duplicate features were detected. Various scaling methods were used, but incorrect overwriting corrupted the results. Proper transformations are necessary for improved model performance. The process generally enhanced data quality and got it machine learning ready.