Machine Learning
Assignment 1
Aakanksha Darekar
202200733
A1 09

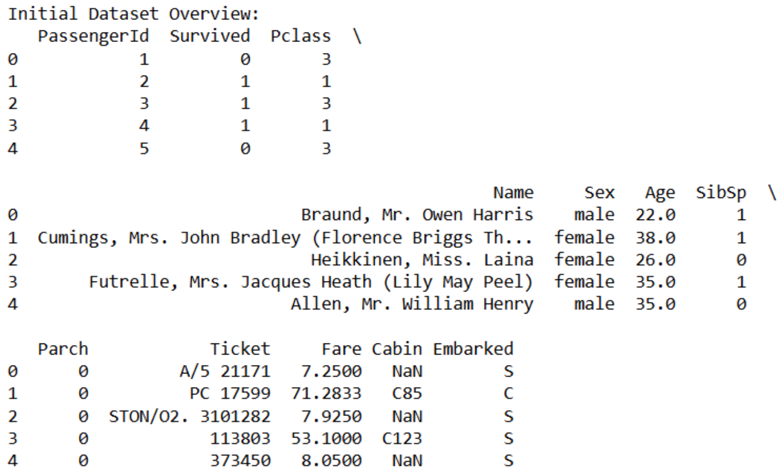
Problem Statement
Preprocess an unclean dataset (Titanic) by performing the following steps:

- **Identify and handle missing values.**
- **Encode categorical variables.**
- **Normalize/standardize numerical data.**
- **Identify and remove duplicate records.**
- **Perform exploratory data analysis (EDA) to understand the dataset.**

Last 2 points not done

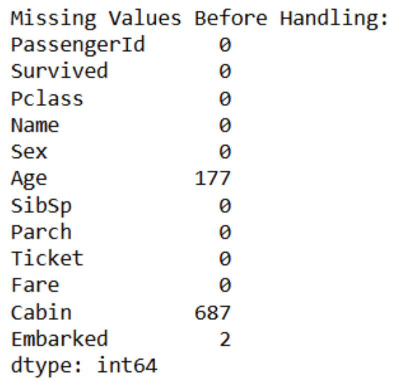
Python code:

```
import pandas as pd
import numpy as np
import seaborn as sns

file_path = ("C:/Users/Maithili/Downloads/titanic.csv")

data = pd.read_csv(file_path)

print("Initial Dataset Overview:")
print(data.head())
```



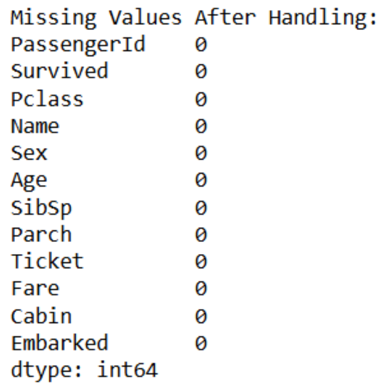
```
Initial Dataset Overview:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                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
```

```
print("\nMissing Values Before Handling:")
print(data.isnull().sum())
```



```
Missing Values Before Handling:
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
```

```
data['Age'].fillna(data['Age'].median(), inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
data['Cabin'].fillna('Unknown', inplace=True)

print("\nMissing Values After Handling:")
print(data.isnull().sum())
```

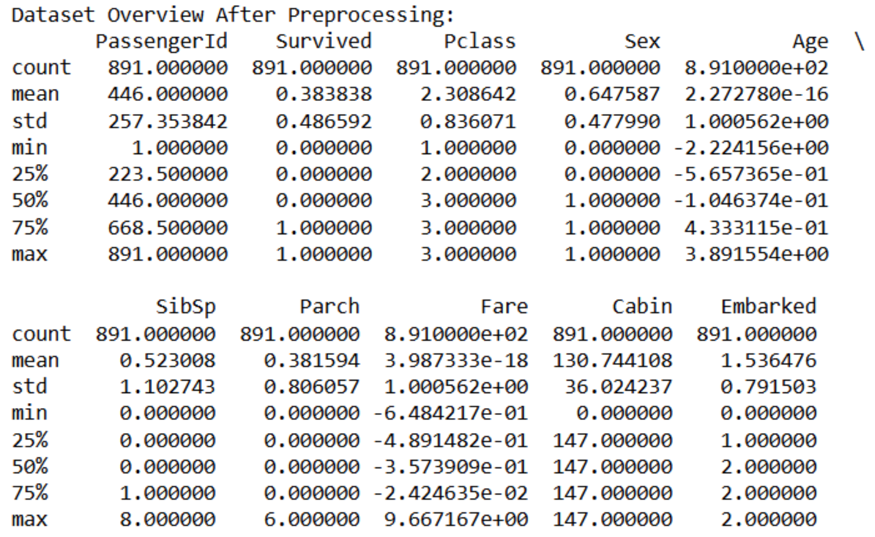


```
Missing Values After Handling:
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

```
categorical_columns = ['Sex', 'Embarked', 'Cabin']
label_encoders = {}

for col in categorical_columns:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

numerical_columns = ['Age', 'Fare']
scaler = StandardScaler()
data[numerical_columns] = scaler.fit_transform(data[numerical_columns])
```

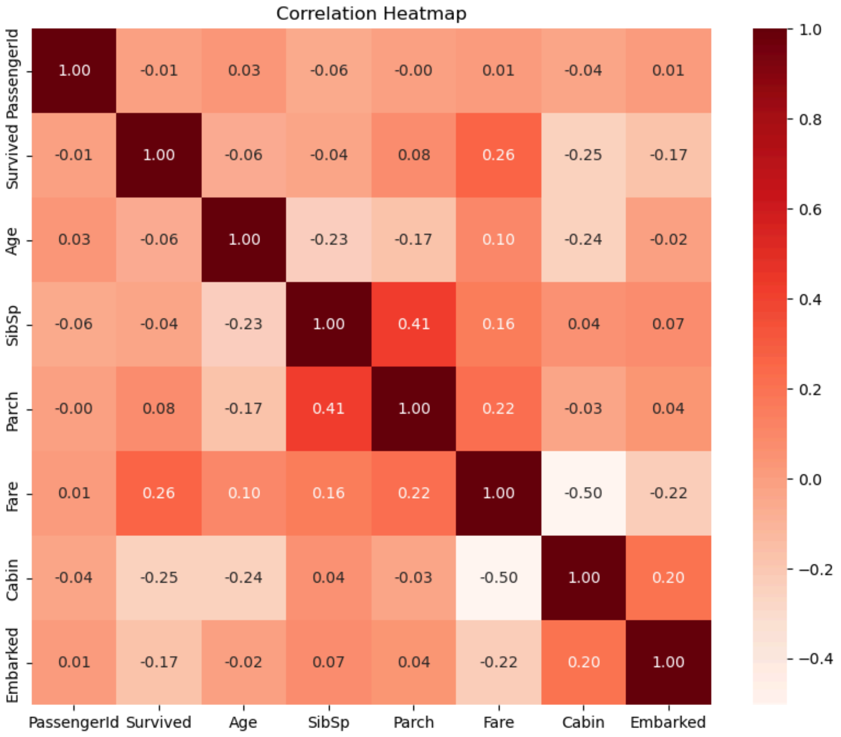
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the Titanic dataset (replace 'titanic.csv' with
your actual file path)
df = pd.read_csv('titanic.csv')

# Display basic dataset information
print("Initial Dataset Info:")
print(df.info())

# Identify and remove duplicate records
print("\nChecking for duplicates...")
duplicates = df.duplicated().sum()
print(f"Number of duplicate records: {duplicates}")

# Remove duplicates
df = df.drop_duplicates()
print("Duplicates removed.")

# Perform Exploratory Data Analysis (EDA)
print("\nSummary Statistics:")
print(df.describe())

# Count missing values
print("\nMissing Values:")
print(df.isnull().sum())

# Visualizing missing values
plt.figure(figsize=(10, 6))
sns.heatmap(df.isnull(), cmap='viridis', cbar=False)
plt.title("Missing Values Heatmap")
plt.show()

# Visualize survival distribution
plt.figure(figsize=(6, 4))
sns.countplot(x='Survived', data=df,
palette='coolwarm')
plt.title("Survival Count")
plt.show()
```

```
print("\nDataset Overview After Preprocessing:")
print(data.describe())
```

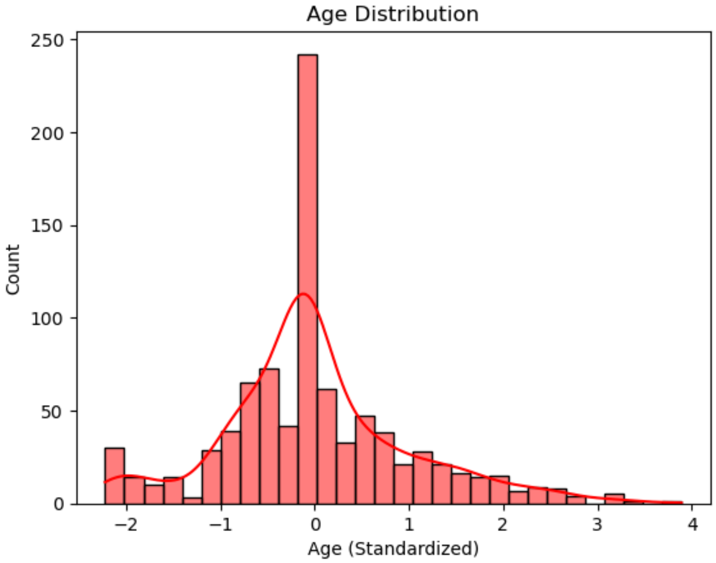


```
Dataset Overview After Preprocessing:
       PassengerId    Survived      Pclass         Sex           Age  \
count   891.000000  891.000000  891.000000  891.000000  8.910000e+02
mean    446.000000    0.383838    2.308642    0.647587  2.272780e-16
std     257.353842    0.486592    0.836071    0.477990  1.000562e+00
min       1.000000    0.000000    1.000000    0.000000 -2.224156e+00
25%     223.500000    0.000000    2.000000    0.000000 -5.657365e-01
50%     446.000000    0.000000    3.000000    1.000000 -1.046374e-01
75%     668.500000    1.000000    3.000000    1.000000  4.333115e-01
max     891.000000    1.000000    3.000000    1.000000  3.891554e+00

            SibSp       Parch          Fare       Cabin    Embarked
count  891.000000  891.000000  8.910000e+02  891.000000  891.000000
mean     0.523008    0.381594  3.987333e-18  130.744108    1.536476
std      1.102743    0.806057  1.000562e+00   36.024237    0.791503
min      0.000000    0.000000 -6.484217e-01    0.000000    0.000000
25%      0.000000    0.000000 -4.891482e-01  147.000000    1.000000
50%      0.000000    0.000000 -3.573909e-01  147.000000    2.000000
75%      1.000000    0.000000 -2.424635e-02  147.000000    2.000000
max      8.000000    6.000000  9.667167e+00  147.000000    2.000000
```

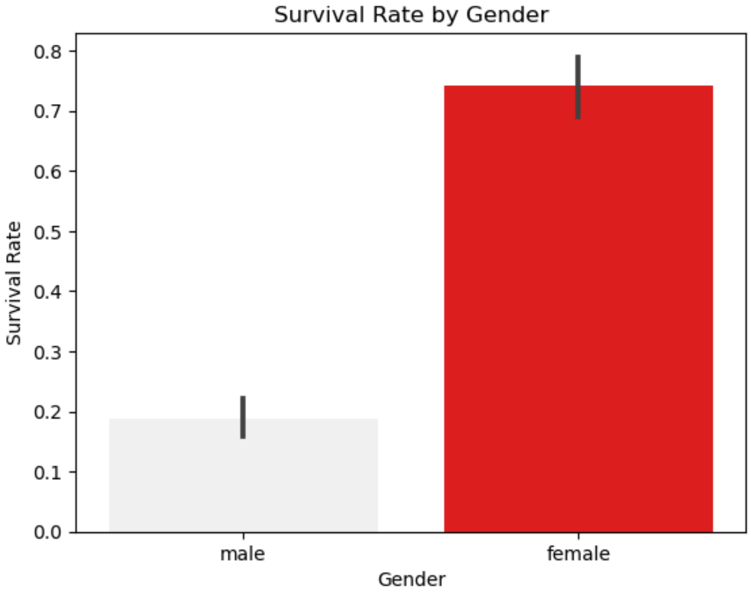
```
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



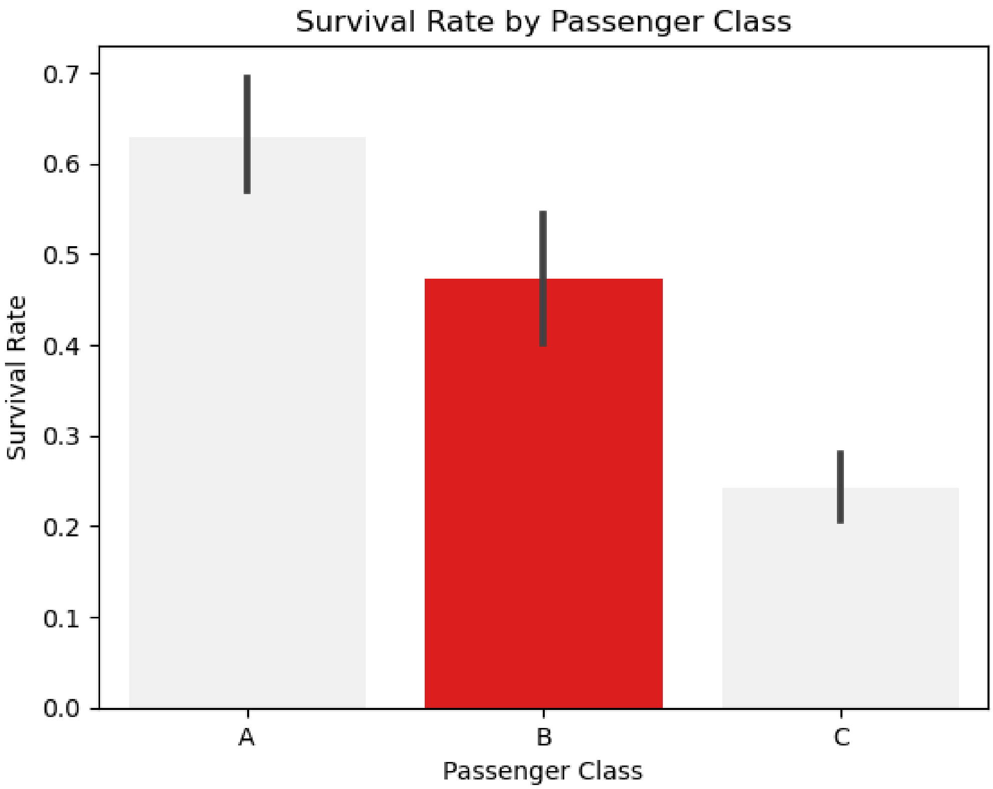
```
sns.histplot(data['Age'], kde=True, bins=30)
plt.title('Age Distribution')
plt.xlabel('Age (Standardized)')
plt.show()
```



```
data['Sex'] = data['Sex'].replace({1: 'male', 0: 'female'})

sns.barplot(x='Sex', y='Survived', data=data)
plt.title('Survival Rate by Gender')
plt.xlabel('Gender')
plt.ylabel('Survival Rate')
plt.show()
```



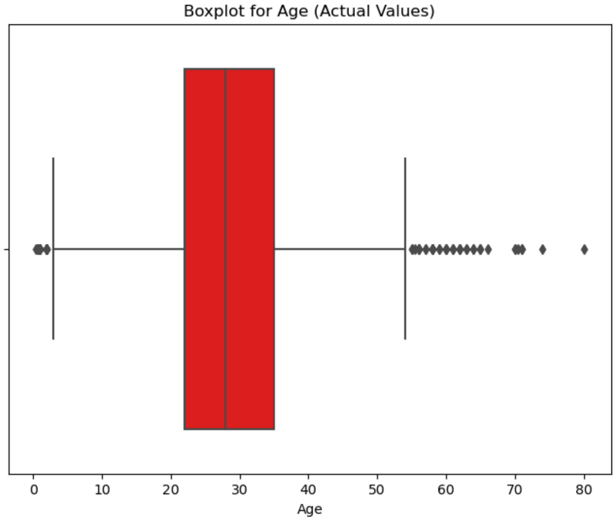
```
data['Pclass'] = data['Pclass'].replace({1: 'A', 2: 'B', 3: 'C'})


sns.barplot(x='Pclass', y='Survived', data=data, order=['A', 'B', 'C'])
plt.title('Survival Rate by Passenger Class')
plt.xlabel('Passenger Class')
plt.ylabel('Survival Rate')
plt.show()
```



Finding outliers

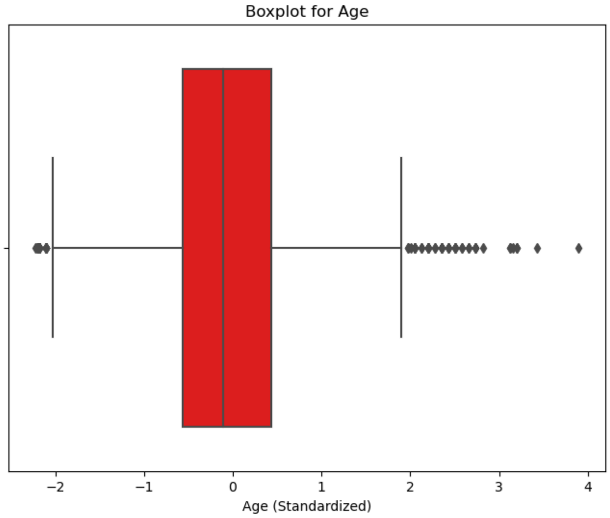
Before standardization

```
plt.figure(figsize=(8, 6))
sns.boxplot(x=data['Age'] * scaler.scale_[0] + scaler.mean_[0], color =”red”)
# Reverse standardization
plt.title('Boxplot for Age (Actual Values)')
plt.xlabel('Age')
plt.show()
```



After standardization

```
plt.figure(figsize=(8, 6))
sns.boxplot(x=data['Age'])
plt.title('Boxplot for Age')
plt.xlabel('Age (Standardized)')
plt.show()
```


Boxplot for Age
Age (Standardized)