TITANIC SURVIVAL PREDICTION

1. Load the dataset and view some rows.

```
[13] #JaiShreeRam

     import pandas as pd


     data = pd.read_csv('tested.csv')


     print(data.head())
```

```
   PassengerId  Survived  Pclass  \
0          892         0       3
1          893         1       3
2          894         0       2
3          895         0       3
4          896         1       3

                                          Name     Sex   Age  SibSp  Parch  \
0                              Kelly, Mr. James    male  34.5      0      0
1              Wilkes, Mrs. James (Ellen Needs)  female  47.0      1      0
2                     Myles, Mr. Thomas Francis    male  62.0      0      0
3                              Wirz, Mr. Albert    male  27.0      0      0
4  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1      1

   Ticket     Fare Cabin Embarked
0  330911   7.8292   NaN        Q
1  363272   7.0000   NaN        S
2  240276   9.6875   NaN        Q
3  315154   8.6625   NaN        S
4 3101298  12.2875   NaN        S
```

2. Handle the missing values.

```
[22] missing_values = data.isnull().sum()
     print(missing_values)
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

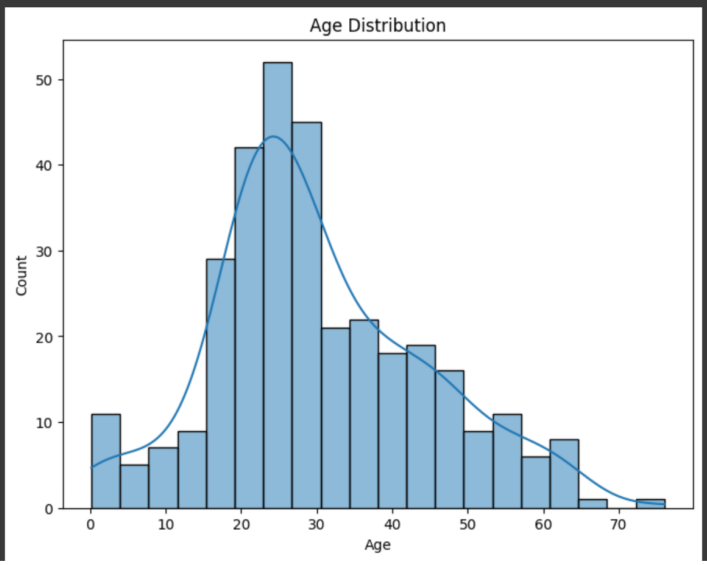
```
     data['Age'].fillna(data['Age'].median(), inplace=True)
     data['Fare'].fillna(data['Fare'].median(), inplace=True)
```

```
[24] data.drop(columns=['Cabin'], inplace=True)
```

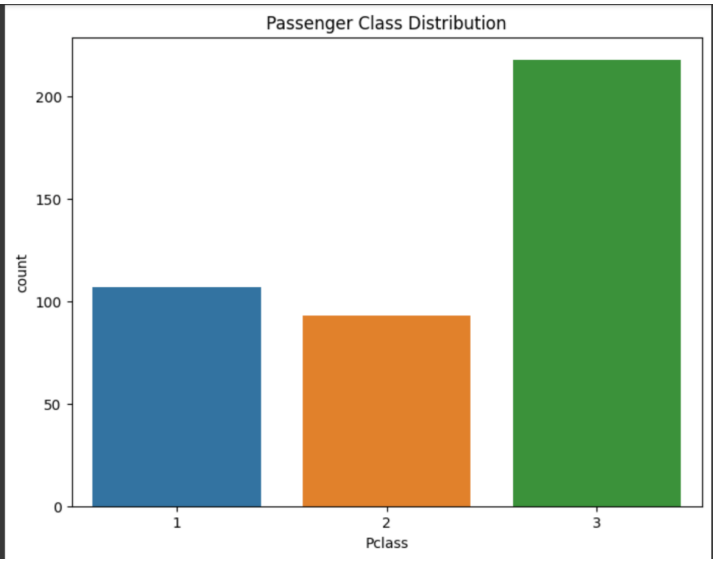
```
[25] data = pd.get_dummies(data, columns=['Sex', 'Embarked'], drop_first=True)
```

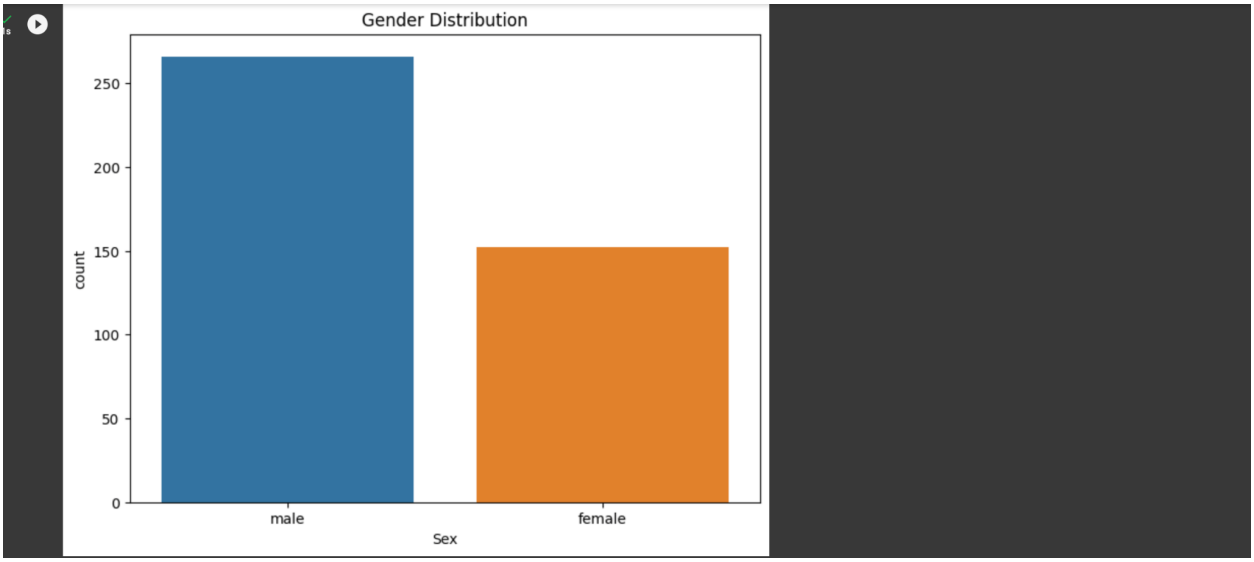
3. Perform EDA.

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


plt.figure(figsize=(8, 6))
sns.histplot(data['Age'], bins=20, kde=True)
plt.title('Age Distribution')
plt.show()
```

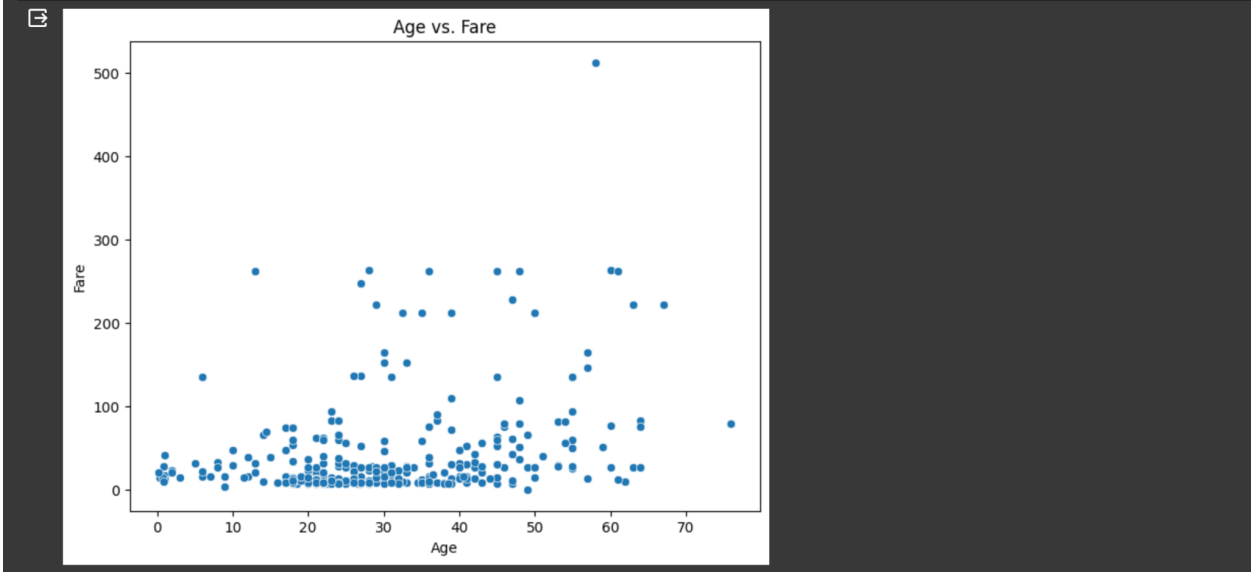



```python
plt.figure(figsize=(8, 6))
sns.countplot(data=data, x='Pclass')
plt.title('Passenger Class Distribution')
plt.show()



# Count plot for 'Sex' (Gender)
plt.figure(figsize=(8, 6))
sns.countplot(data=data, x='Sex')
plt.title('Gender Distribution')
plt.show()
```



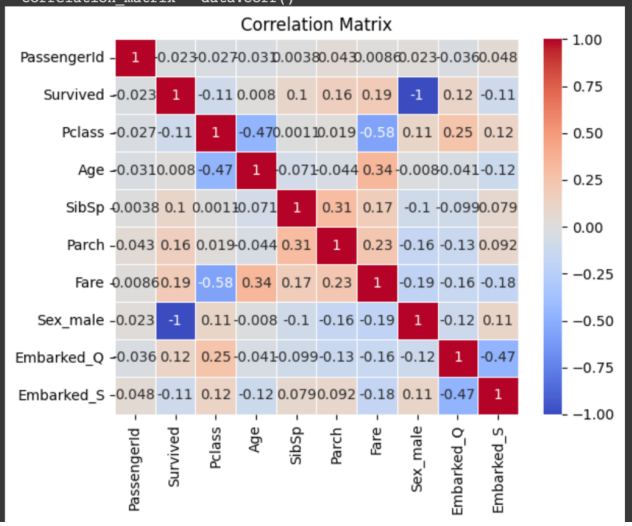



```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Age', y='Fare', data=data)
plt.title('Age vs. Fare')
plt.show()
```



```
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```

```
<ipython-input-26-609fdfdc245b>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, i
  correlation_matrix = data.corr()
```



4. Scale the feature.

```
[27] data['Title'] = data['Name'].str.extract(' ([A-Za-z]+)\.')
```

```
[28] bins = [0, 18, 35, 60, 120]
     labels = ['Child', 'Young Adult', 'Adult', 'Senior']
     data['AgeGroup'] = pd.cut(data['Age'], bins=bins, labels=labels)
```

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler


scaler = StandardScaler()
data['Fare_Standardized'] = scaler.fit_transform(data[['Fare']])


minmax_scaler = MinMaxScaler()
data['Age_Normalized'] = minmax_scaler.fit_transform(data[['Age']])
```

5. Perform encoding.

```
from sklearn.preprocessing import OneHotEncoder


encoder = OneHotEncoder()


name_encoded = encoder.fit_transform(data[['Name']])

feature_names = encoder.get_feature_names_out(['Name'])


data = data.join(pd.DataFrame(name_encoded.toarray(), columns=feature_names))


data.drop(columns=['Name'], inplace=True)
```

6. Model selection, data split and model training.

```
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder()

name_encoded = encoder.fit_transform(data[['Name']])
feature_names = encoder.get_feature_names_out(['Name'])

data = data.join(pd.DataFrame(name_encoded.toarray(), columns=feature_names))

data.drop(columns=['Name'], inplace=True)
```

7. Model evaluation.

```
[43] from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

Accuracy: 1.00
```

8. Regularization.

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(max_depth=10, min_samples_split=5)

model.fit(X_train, y_train)
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

Accuracy: 1.00
```

9. Survival Prediction.

```
passenger_ids = X_test['PassengerId'].values

survival_predictions = model.predict(X_test)
results = pd.DataFrame({'PassengerId': passenger_ids, 'Survival_Prediction': survival_predictions})

print(results)
```

```
    PassengerId  Survival_Prediction
0          1213                    0
1          1216                    1
2          1280                    0
3           948                    0
4          1045                    1
..          ...                  ...
79          949                    0
80         1018                    0
81          916                    1
82          909                    0
83          958                    1

[84 rows x 2 columns]
```