Step 1: Importing Libraries and figuring out the data

```
import pandas as pd

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

data.head()
```

```
   Unnamed: 0  PassengerId  Survived  Pclass  \
0           0            1       0.0       3
1           1            2       1.0       1
2           2            3       1.0       3
3           3            4       1.0       1
4           4            5       0.0       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
```

```
data.shape
```

```
(1309, 13)
```

```
data.isnull().sum()
```

```
Unnamed: 0       0
PassengerId      0
Survived       418
Pclass           0
Name             0
Sex              0
```

```
Age              263
SibSp              0
Parch              0
Ticket             0
Fare               1
Cabin           1014
Embarked           2
dtype: int64
```

```
data.describe()
```

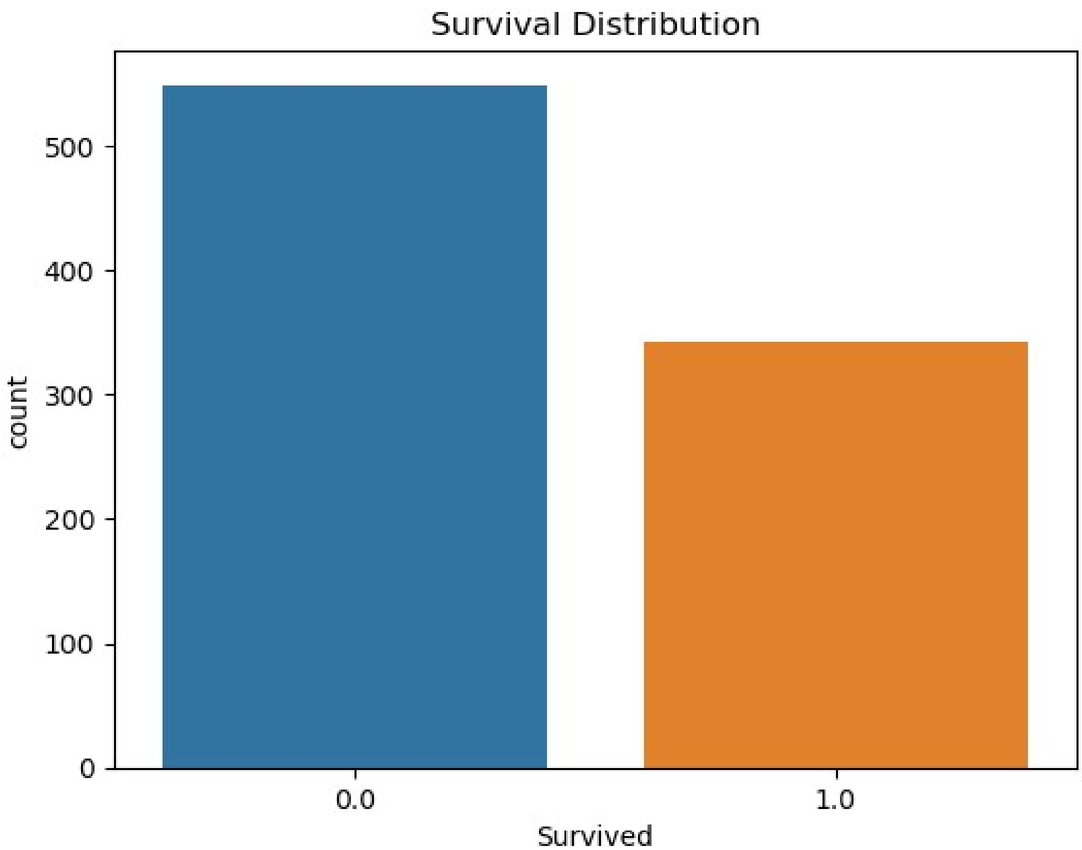
```
       Unnamed: 0  PassengerId    Survived       Pclass          Age
\
count  1309.000000  1309.000000  891.000000  1309.000000  1046.000000

mean    654.000000   655.000000    0.383838     2.294882    29.881138

std     378.020061   378.020061    0.486592     0.837836    14.413493

min       0.000000     1.000000    0.000000     1.000000     0.170000

25%     327.000000   328.000000    0.000000     2.000000    21.000000

50%     654.000000   655.000000    0.000000     3.000000    28.000000

75%     981.000000   982.000000    1.000000     3.000000    39.000000

max    1308.000000  1309.000000    1.000000     3.000000    80.000000


             SibSp        Parch         Fare
count  1309.000000  1309.000000  1308.000000
mean      0.498854     0.385027    33.295479
std       1.041658     0.865560    51.758668
min       0.000000     0.000000     0.000000
25%       0.000000     0.000000     7.895800
50%       0.000000     0.000000    14.454200
75%       1.000000     0.000000    31.275000
max       8.000000     9.000000   512.329200
```

Step 2: EDA process

2.1 Survival Count Plot: Understand the distribution of the target variable.

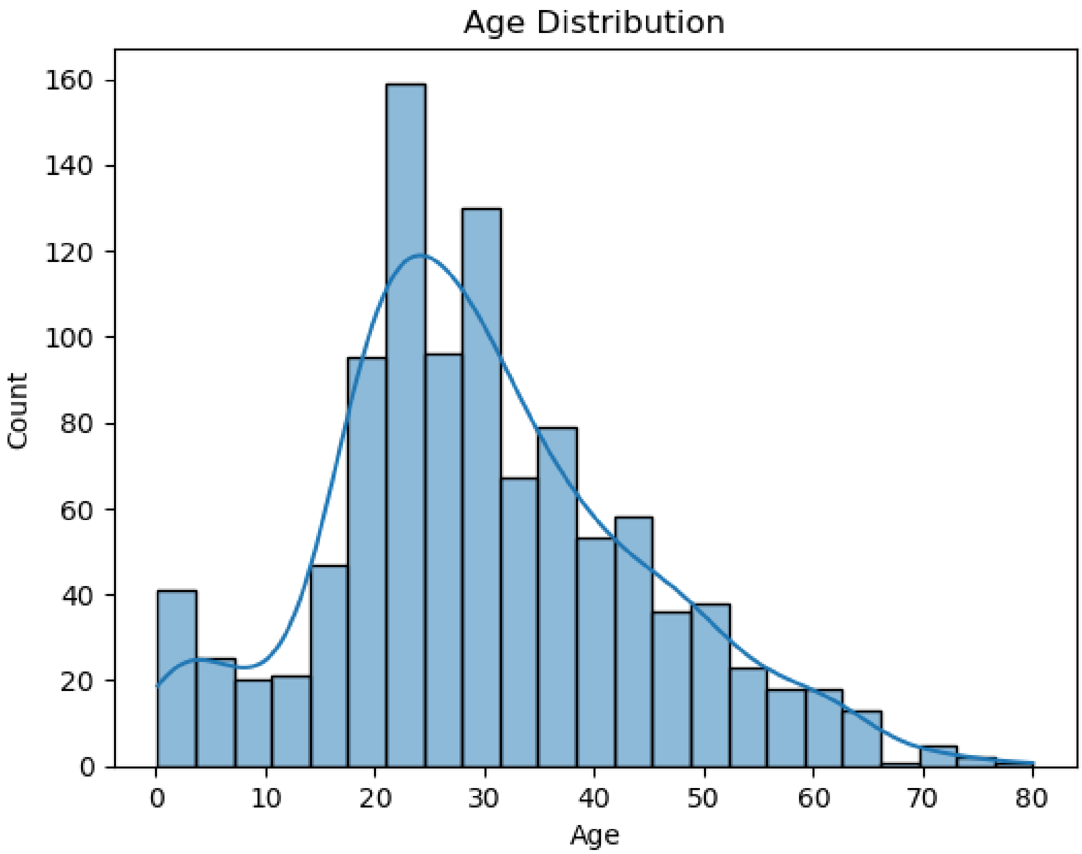
```
import seaborn as sns
import matplotlib.pyplot as plt
```

```python
# Countplot of Survived
sns.countplot(x='Survived', data=data)
plt.title('Survival Distribution')
plt.show()
```

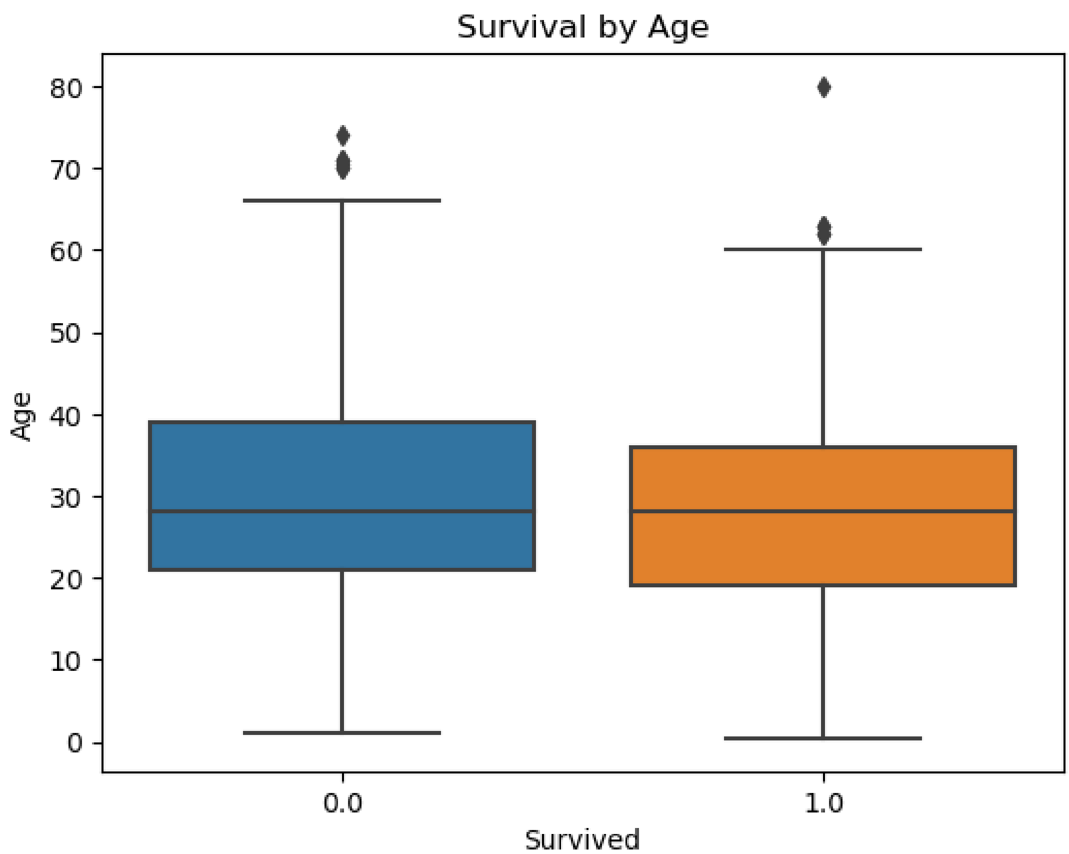


2.2 Age Distribution: Explore the distribution of Age and its relationship to survival.

```python
sns.histplot(data['Age'], kde=True)
plt.title('Age Distribution')
plt.show()

# Relationship between Age and Survival
sns.boxplot(x='Survived', y='Age', data=data)
plt.title('Survival by Age')
plt.show()
```

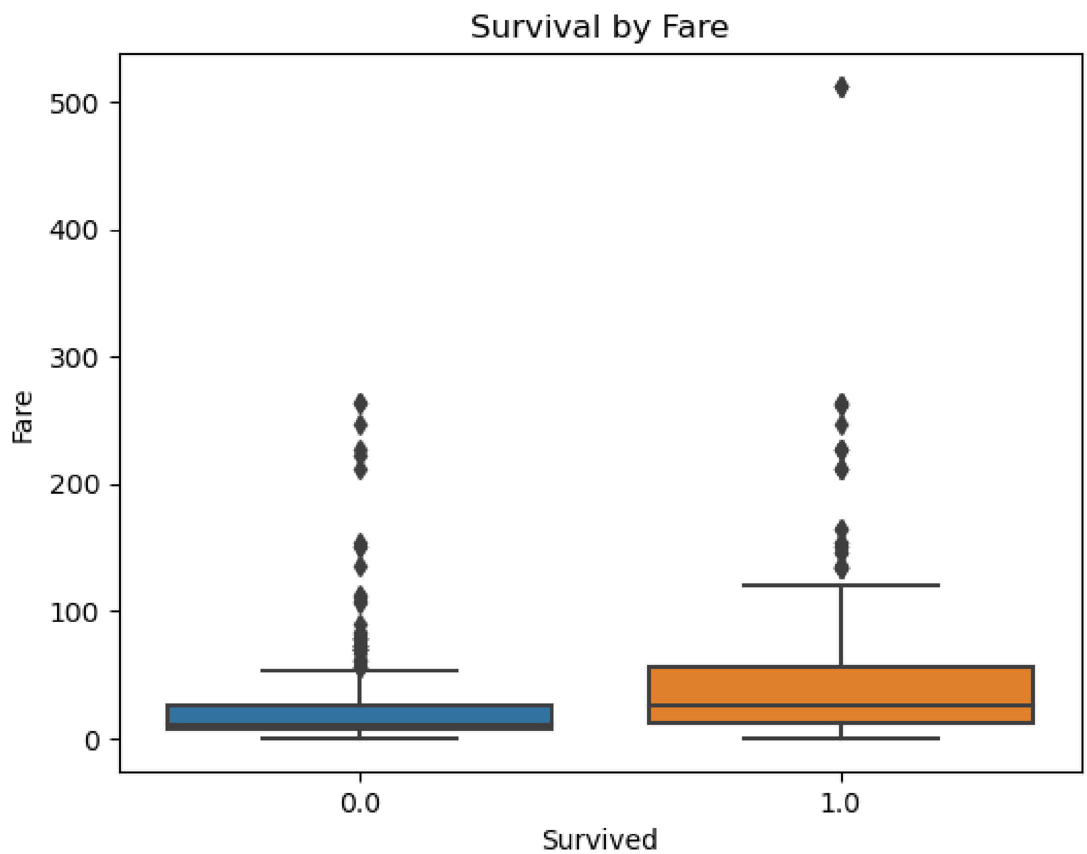


Age Distribution
Count
Age
0
20
40
60
80
100
120
140
160
0
10
20
30
40
50
60
70
80



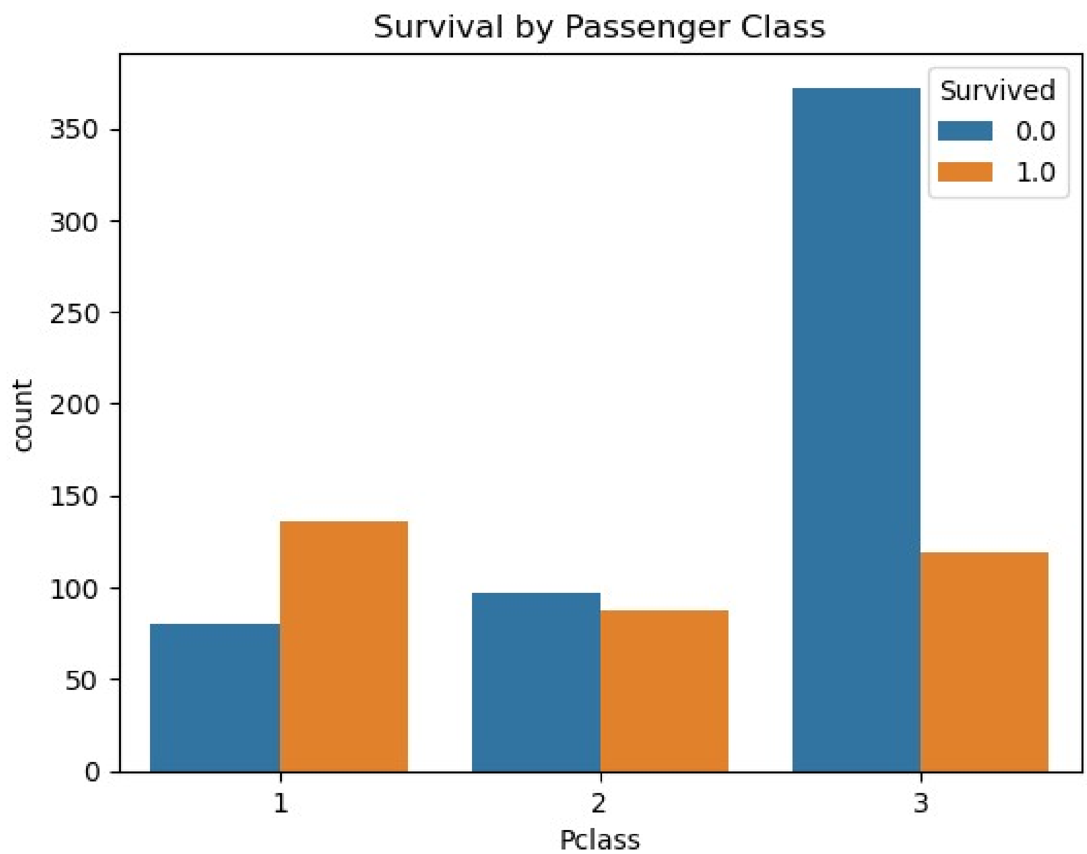
2.3 Fare Distribution by Survival: Explore how ticket fare affects survival.

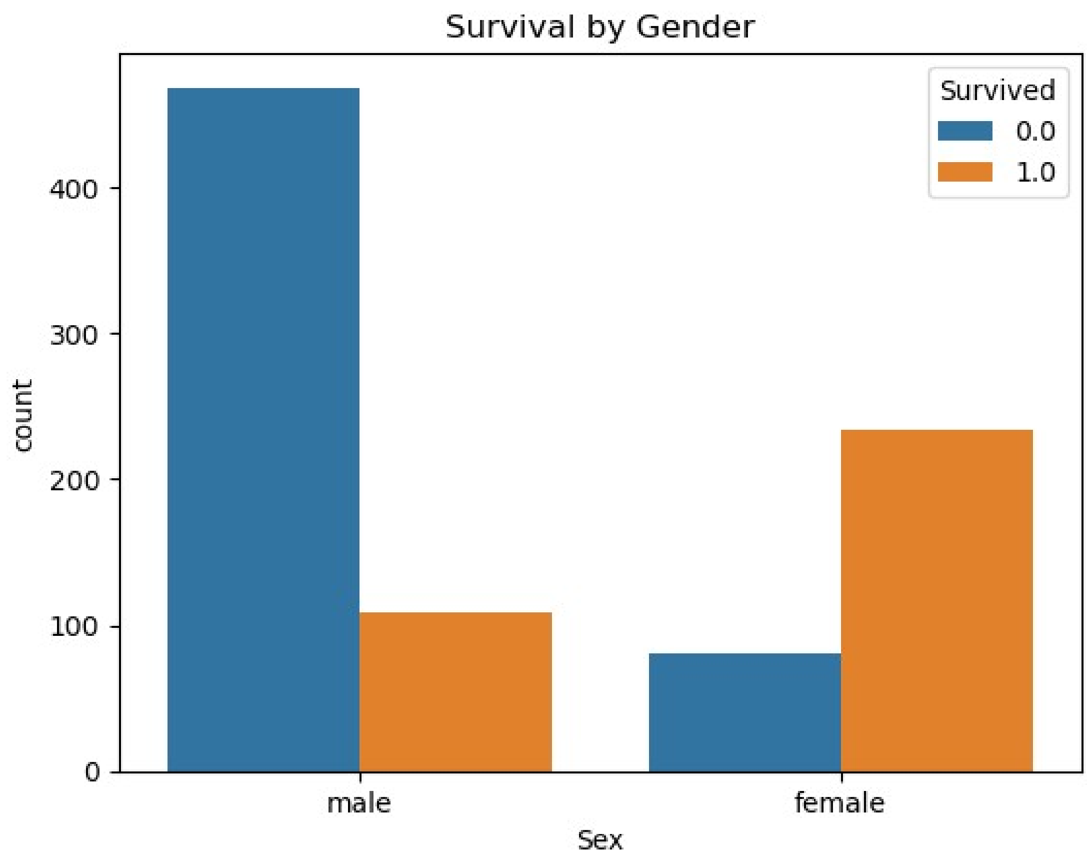
```
sns.boxplot(x='Survived', y='Fare', data=data)
plt.title('Survival by Fare')
plt.show()
```



2.4 Categorical Variables vs Survival: For Pclass, Sex, and Embarked.

```
sns.countplot(x='Pclass', hue='Survived', data=data)
plt.title('Survival by Passenger Class')
plt.show()

sns.countplot(x='Sex', hue='Survived', data=data)
plt.title('Survival by Gender')
plt.show()
```


Survival by Passenger Class
Survived
0.0
1.0
count
350
300
250
200
150
100
50
0
1
2
3
Pclass



Step 3: Data Preprocessing

3.1 Handling Missing Values

```
data = data.drop(columns=['Unnamed: 0'])

# Fill missing Age with median
data['Age'].fillna(data['Age'].median(), inplace=True)

# Fill missing Embarked with mode
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)

# Fill missing Fare (if any)
data['Fare'].fillna(data['Fare'].median(), inplace=True)
```

3.2 Encoding Categorical Variables

```
# One-hot encode Embarked
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
```

```
# Encode Sex column
data['Sex'] = data['Sex'].map({'male': 1, 'female': 0})
```

3.3 Feature Scaling

```
# You may want to standardize features like Fare and Age.

from sklearn.preprocessing import StandardScaler

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

Step 4: Feature Selection (Lasso Regularization)

Lasso can be used for feature selection by penalizing less important features.

```
# Drop columns that won't be used in the model
data.drop(columns=['Name', 'Ticket', 'Cabin'], inplace=True)
```

plt.figure(figsize=(10, 8)) # Set the figure size sns.heatmap(data, annot=True, cmap='viridis', fmt='g') # 'annot' displays values on the heatmap plt.title('Heatmap of DataFrame') plt.xlabel('Columns') plt.ylabel('Rows') plt.show()

```
''' We when checked above knew there are missing values in the target
y(survived) so we
need to deal with it first '''

# Fill missing Survived values with the most frequent value (mode)

data['Survived'].fillna(data['Survived'].mode()[0], inplace=True)

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LassoCV
import numpy as np

# Split data into X and y
X = data.drop(columns=['Survived'])
y = data['Survived']

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Apply Lasso to perform feature selection
lasso = LassoCV(cv=5, random_state=0).fit(X_train, y_train)
```

```
# Get selected features
coef = np.where(lasso.coef_ != 0)[0]
selected_features = X.columns[coef]
print("Selected features by Lasso:", selected_features)
```

```
Selected features by Lasso: Index(['Unnamed: 0', 'PassengerId',
'Pclass', 'Sex'], dtype='object')
```

```
# Split data into X and y
X = data[['PassengerId', 'Pclass', 'Sex']]
y = data['Survived']

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Step 5: Model Building

```
from sklearn.linear_model import LogisticRegression

# Create the logistic regression model
model = LogisticRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

# Predicting the results for test set
y_pred = model.predict(X_test)
```

Step 6: Evaluation Metrics

6.1 Accuracy Scores

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.8282442748091603
```

6.2 Precision, Recall, and F1-Score:

```
from sklearn.metrics import precision_score, recall_score, f1_score

precision = precision_score(y_test, y_pred)
```

```
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

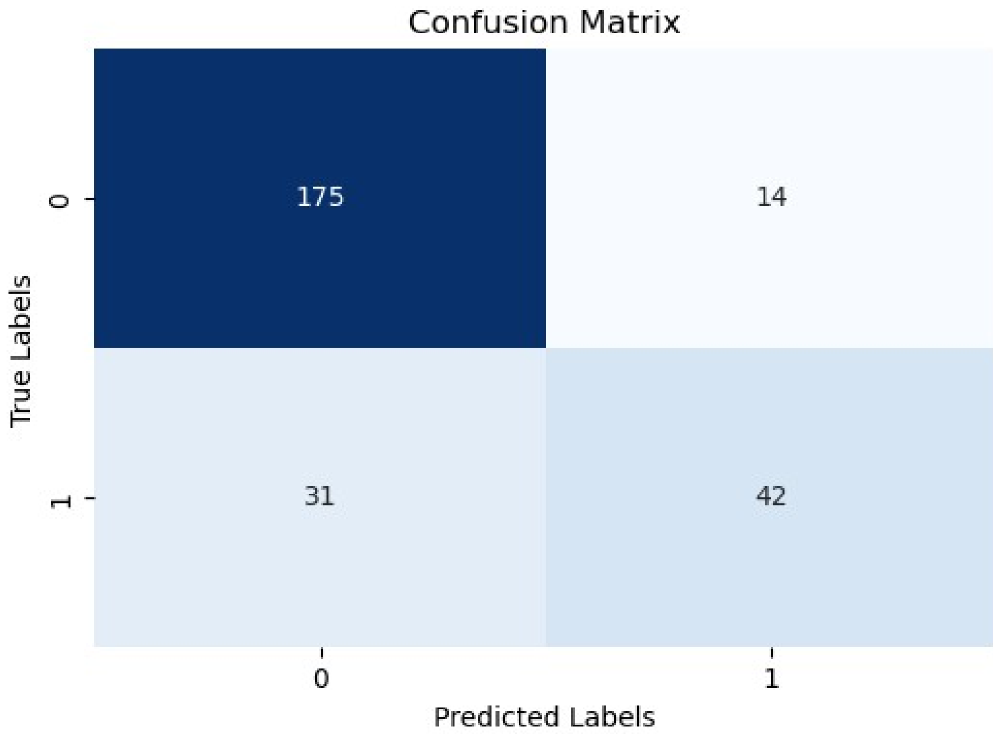
```
Precision: 0.75
Recall: 0.5753424657534246
F1 Score: 0.6511627906976744
```

6.3 Confusion Matrix:

```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix: \n", cm)
```

```
Confusion Matrix:
 [[175  14]
 [ 31  42]]
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix as a heatmap
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)

# Add labels and titles
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')

# Show the plot
plt.show()
```



Step 7: Model Interpretation

```
coefficients = pd.DataFrame({'Feature': X_train.columns,
'Coefficient': model.coef_[0]})
print(coefficients)

        Feature  Coefficient
0   Unnamed: 0    -1.145734
1  PassengerId     1.142903
2       Pclass    -0.806022
3          Sex    -2.118951

import statsmodels.api as sm

X_train_sm = sm.add_constant(X_train)
logit_model = sm.Logit(y_train, X_train_sm)
result = logit_model.fit()
print(result.summary())

Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.395503
         Iterations: 35
                           Logit Regression Results

======================================================================
========
```

```
Dep. Variable:                   Survived   No. Observations:
1047
Model:                              Logit   Df Residuals:
1042
Method:                               MLE   Df Model:
4
Date:                    Tue, 24 Sep 2024   Pseudo R-squ.:
0.3059
Time:                            22:44:13   Log-Likelihood:
-414.09
converged:                          False   LL-Null:
-596.60
Covariance Type:                nonrobust   LLR p-value:
1.006e-77
======================================================================
=========
                  coef    std err          z      P>|z|      [0.025
0.975]
----------------------------------------------------------------------
---------
const           2.3793        nan        nan        nan         nan
nan
Unnamed: 0     -1.2741        nan        nan        nan         nan
nan
PassengerId     1.2712        nan        nan        nan         nan
nan
Pclass         -0.8391      0.104     -8.075      0.000      -1.043
-0.635
Sex            -2.2140      0.185    -11.956      0.000      -2.577
-1.851
======================================================================
=========

/Users/prakashpandey/Documents/anaconda3/lib/python3.9/site-packages/
statsmodels/base/model.py:607: ConvergenceWarning: Maximum Likelihood
optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

P>|z|: The p-values for Pclass and Sex are both 0.000, which is highly significant, suggesting that these features are statistically significant predictors of survival.

```python
# Create a DataFrame for predictions
predictions_df = pd.DataFrame({
    'PassengerId': data.loc[X_test.index, 'PassengerId'],  # Use the
indices to match
    'Predicted_Survived': y_pred
})
print(predictions_df)
```

```python
# Merge predictions with the original DataFrame
final_df = data.merge(predictions_df, on='PassengerId', how='inner')

# Save the final DataFrame to a CSV file
final_df.to_csv('titanic_with_predictions.csv', index=False)

# Optional: View the columns in the final DataFrame
print(final_df[['PassengerId', 'Survived',
'Predicted_Survived']].head())
```

```
      PassengerId  Predicted_Survived
1148         1149                 0.0
1049         1050                 0.0
982           983                 0.0
808           809                 0.0
1195         1196                 0.0
...           ...                 ...
572           573                 0.0
140           141                 1.0
1182         1183                 0.0
312           313                 1.0
199           200                 1.0

[262 rows x 2 columns]
   PassengerId  Survived  Predicted_Survived
0           24       1.0                 1.0
1           30       0.0                 0.0
2           32       1.0                 1.0
3           33       1.0                 1.0
4           44       1.0                 1.0
```

```python
final_df.tail()
```

```
     Unnamed: 0  PassengerId  Survived  Pclass  Sex       Age  SibSp
Parch  \
257        1292         1293       0.0       2    1  0.658652      1
0
258        1293         1294       0.0       1    0 -0.581628      0
1
259        1295         1296       0.0       1    1  1.046240      1
0
260        1298         1299       0.0       1    1  1.588862      1
1
261        1301         1302       0.0       3    0 -0.116523      0
0

         Fare  Embarked_Q  Embarked_S  Predicted_Survived
257 -0.237445           0           1                 0.0
258  0.504989           0           0                 0.0
259 -0.107504           0           0                 0.0
```

260	3.445726	0	0	0.0
261	-0.493624	1	0	0.0