Logistic Regression

Objective: “What sorts of people were more likely to survive the sinking of the Titanic?”

https://www.kaggle.com/competitions/titanic/data

[1]:

```
import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:

```
t_df = pd.read_csv('titanic_data.csv', index_col='PassengerId')
```

[3]:

```
print(t_df.shape)

# Count null values per column
# null_counts = t_df.isnull().sum()
# print(null_counts)

#Drop all rows containing at least one null value
# t_df = t_df.dropna()
# print(t_df.shape)
```

```
(891, 11)
```

[10]:

```
#Data Cleanup



cols_to_drop = ['Name', 'Cabin', 'Ticket']
t_df.drop(columns=[col for col in cols_to_drop if col in t_df.columns], inplace=True)



t_df = t_df.dropna()



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

[11]:

```
X = t_df.drop(columns=['Survived'])
y = t_df['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

[12]:

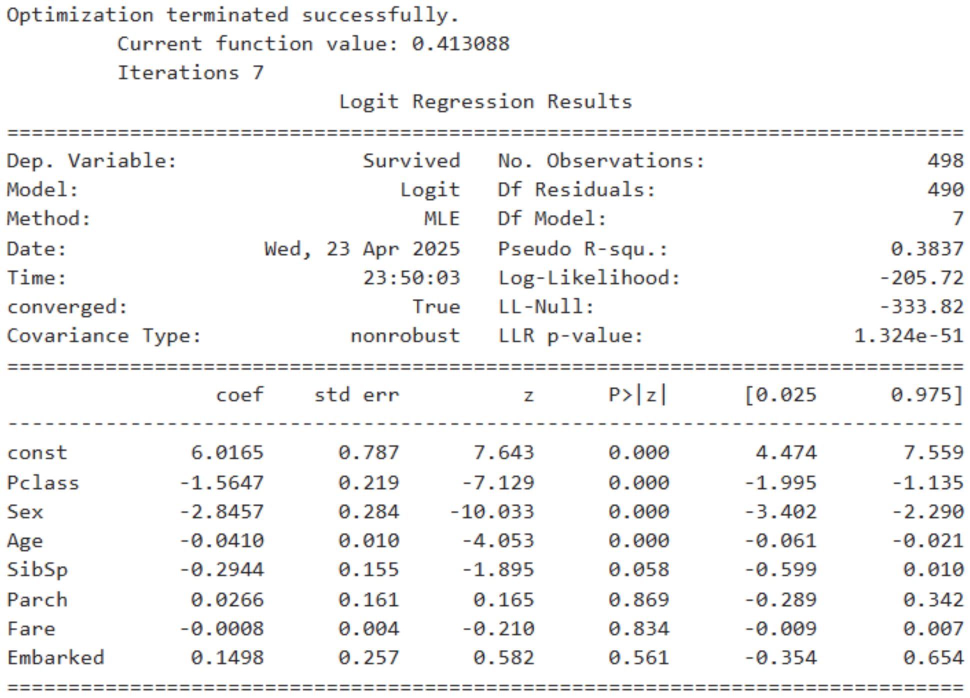
```

X_train_sm = sm.add_constant(X_train)

X_train_sm = X_train_sm.dropna()
y_train_cleaned = y_train.loc[X_train_sm.index]

logmodel = sm.Logit(y_train_cleaned, X_train_sm).fit()

print(logmodel.summary())
```



```
Optimization terminated successfully.
         Current function value: 0.413088
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:               Survived   No. Observations:                  498
Model:                          Logit   Df Residuals:                      490
Method:                           MLE   Df Model:                            7
Date:                Wed, 23 Apr 2025   Pseudo R-squ.:                  0.3837
Time:                        23:50:03   Log-Likelihood:                -205.72
converged:                       True   LL-Null:                       -333.82
Covariance Type:            nonrobust   LLR p-value:                 1.324e-51
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          6.0165      0.787      7.643      0.000       4.474       7.559
Pclass        -1.5647      0.219     -7.129      0.000      -1.995      -1.135
Sex           -2.8457      0.284    -10.033      0.000      -3.402      -2.290
Age           -0.0410      0.010     -4.053      0.000      -0.061      -0.021
SibSp         -0.2944      0.155     -1.895      0.058      -0.599       0.010
Parch          0.0266      0.161      0.165      0.869      -0.289       0.342
Fare          -0.0008      0.004     -0.210      0.834      -0.009       0.007
Embarked       0.1498      0.257      0.582      0.561      -0.354       0.654
==============================================================================
```

[15]:

```
X_test_sm = sm.add_constant(X_test)
X_test_sm = X_test_sm.dropna()
y_test_cleaned = y_test.loc[X_test_sm.index]


pred_probs = logmodel.predict(X_test_sm)


predictions = [1 if p > 0.5 else 0 for p in pred_probs]


from sklearn.metrics import accuracy_score, confusion_matrix
print("Accuracy:", accuracy_score(y_test_cleaned, pred_binary))
print("Confusion Matrix:\n", confusion_matrix(y_test_cleaned, pred_binary))
```

```
Accuracy: 0.780373831775701
Confusion Matrix:
 [[105  17]
 [ 30  62]]
```

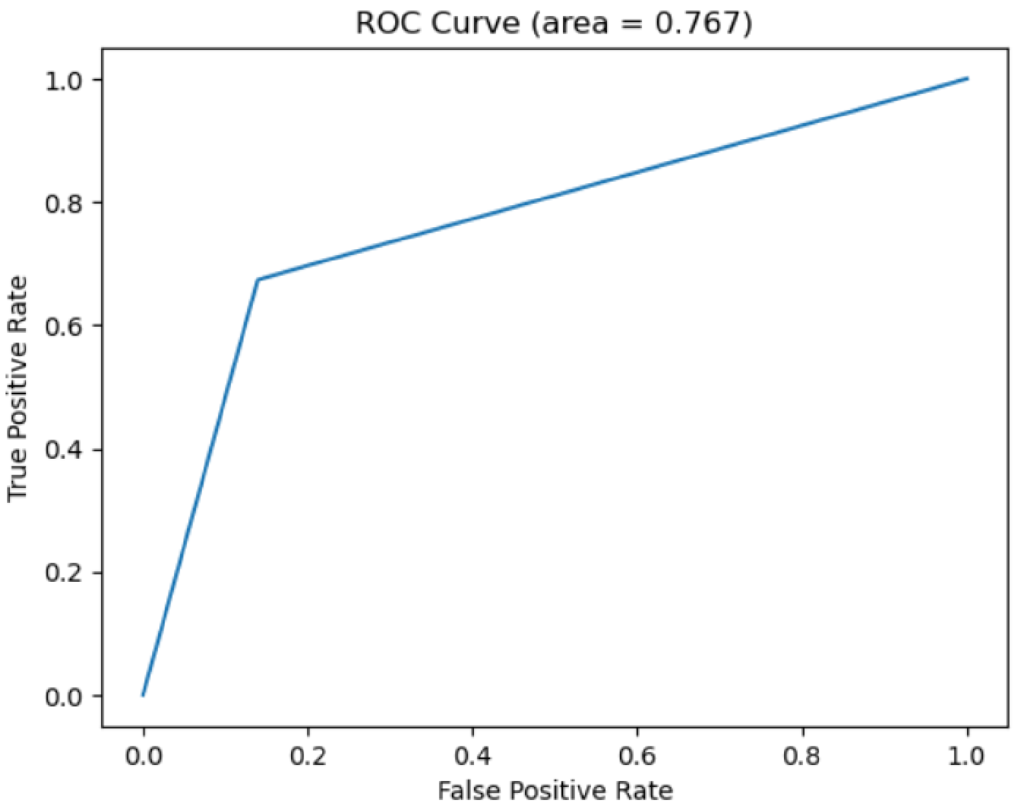
[16]:

```
fpr, tpr, thresholds = roc_curve(y_test, predictions)
roc_auc = roc_auc_score(y_test, predictions)

plt.plot(fpr, tpr, label='ROC Curve (area = %0.3f)' % roc_auc)
plt.title('ROC Curve (area = %0.3f)' % roc_auc)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

[16]:

```
Text(0, 0.5, 'True Positive Rate')
```



Task 1: Rearrange the code so that the cabin column is removed BEFORE the null values. How does this change the confusion matrix and overall accuracy?

```
[17]: df = pd.read_csv('titanic_data.csv', index_col='PassengerId')
```

```
[18]: df.drop(columns=['Cabin', 'Name', 'Ticket'], inplace=True)
```

```
[19]: df = df.dropna()
```

```
[21]: df['Sex'] = df['Sex'].replace({'male': 1, 'female': 0}).astype(int)
      df['Embarked'] = df['Embarked'].replace({'S': 0, 'C': 1, 'Q': 2}).astype(int)
```

```
[22]: X = df.drop(columns='Survived')
      y = df['Survived']
```

```
[23]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

[24]:

```
import statsmodels.api as sm
X_train_sm = sm.add_constant(X_train).dropna()
y_train_sm = y_train.loc[X_train_sm.index]
logmodel = sm.Logit(y_train_sm, X_train_sm).fit(disp=False)
```

[25]:

```
X_test_sm = sm.add_constant(X_test).dropna()
y_test_cleaned = y_test.loc[X_test_sm.index]
pred_probs = logmodel.predict(X_test_sm)
predictions = [1 if p > 0.5 else 0 for p in pred_probs]
```

[26]:

```
from sklearn.metrics import accuracy_score, confusion_matrix
print("Confusion Matrix:\n", confusion_matrix(y_test_cleaned, predictions))
print("Accuracy: %.2f%%" % (accuracy_score(y_test_cleaned, predictions) * 100))
```

```
Confusion Matrix:
 [[105  17]
 [ 30  62]]
Accuracy: 78.04%
```