Logistic Regression - Heart Data

import libraries

In [65]:
```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report, roc_curve, roc_auc_score

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
warnings.filterwarnings("ignore")
```

Data Gathering

In [2]:
```
df = pd.read_csv("heart.csv")
df
```

Out[2]:

	age	gender	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

EDA

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   gender    303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [4]:
```
df['target'].value_counts()
```

Out[4]:
```
target
1    165
0    138
Name: count, dtype: int64
```

In [5]:
```
165/303
```

Out[5]: 0.5445544554455446

In [ ]:

Feature Selection

In [ ]:
```
1. Filter Method:

2. Wrapper Method
3. Embedded Method:
    1. Lasso Regression
```

Model Training

Train Test Split

In [6]:
```
x = df.drop('target', axis = 1)
y = df['target']
# y
```

In [7]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=10, stratify = y)
y_train.value_counts()
```

Out[7]:
```
target
1    132
0    110
Name: count, dtype: int64
```

In [8]:
```
132/242
```

Out[8]: 0.5454545454545454

In [9]:
```
# x_train
```

In [10]:
```
# x_test
```

In [11]:
```
log_clf = LogisticRegression()
log_clf.fit(x_train, y_train)
```

Out[11]:
```
LogisticRegression()
```

Evaluation

In [12]:
```
y_pred_test_prob = log_clf.predict_proba(x_test)
y_pred_test_prob[:6]

# [0.07054224, 0.92945776], >> Class1
# [0.26737039, 0.73262961], >> Class1
# [0.82159417, 0.17840583], >> Class0
# [0.98835195, 0.01164805], >> Class0
# [0.84237389, 0.15762611   >> Class0
# [0.01919849, 0.98080151], >> Class1
```

Out[12]:
```
array([[0.07054224, 0.92945776],
       [0.26737039, 0.73262961],
       [0.82159417, 0.17840583],
       [0.98835195, 0.01164805],
       [0.84237389, 0.15762611],
       [0.01919849, 0.98080151]])
```

In [13]:
```
y_pred_test = log_clf.predict(x_test)
y_pred_test[50:55]
```

Out[13]:
```
array([0, 1, 1, 0, 1], dtype=int64)
```

In [14]:
```
y_test[50:55]
```

Out[14]:
```
165    0
149    1
278    0
298    0
90     1
Name: target, dtype: int64
```

In [15]:
```
confusion_matrix(y_test, y_pred_test)
```

Out[15]:
```
array([[23,  5],
       [ 3, 30]], dtype=int64)
```

In [16]:
```
y_test.value_counts()
```

Out[16]:
```
target
1    33
0    28
Name: count, dtype: int64
```

In [17]:
```
y_test
```

Out[17]:
```
94     1
99     1
280    0
174    0
176    0
      ..
115    1
101    1
219    0
253    0
243    0
Name: target, Length: 61, dtype: int64
```

In [18]:
```
(23+30)/61
```

Out[18]:
```
0.8688524590163934
```

In [19]:
```
accuracy_score(y_test, y_pred_test)
```

Out[19]:
```
0.8688524590163934
```

In [20]:
```
log_clf.score(x_test, y_test)
```

Out[20]:
```
0.8688524590163934
```

Training Data Evaluation

In [21]:
```
y_pred_train = log_clf.predict(x_train)
y_pred_train[60:65]
```

Out[21]:
```
array([0, 1, 1, 1, 1], dtype=int64)
```

In [22]:
```
y_train[60:65]
```

Out[22]:
```
247    0
37     1
293    0
114    1
19     1
Name: target, dtype: int64
```

In [23]:
```
log_clf.predict_proba(x_train)[60:65]
```

Out[23]:
```
array([[0.78384075, 0.21615925],
       [0.42148032, 0.57851968],
       [0.4989611 , 0.5010389 ],
       [0.11473814, 0.88526186],
       [0.0618728 , 0.9381272 ]])
```

In [24]:
```
confusion_matrix(y_train, y_pred_train)
```

Out[24]:
```
array([[ 83,  27],
       [ 10, 122]], dtype=int64)
```

In [25]:
```
(83+122)/242
```

Out[25]:
```
0.8471074380165289
```

In [26]:
```
122/132
```

Out[26]:
```
0.9242424242424242
```

In [27]:
```
TP = 122
FP = 27
Precision = TP/(TP+FP)
Precision
```

Out[27]:
```
0.8187919463087249
```

In [55]:
```
clf_report = classification_report(y_train, y_pred_train)
print("Classification Report :\n", clf_report)
```

```
Classification Report :
               precision    recall  f1-score   support

           0       0.89      0.75      0.82       110
           1       0.82      0.92      0.87       132

    accuracy                           0.85       242
   macro avg       0.86      0.84      0.84       242
weighted avg       0.85      0.85      0.85       242
```

In [29]:
```
clf_report = classification_report(y_test, y_pred_test)
print("Classification Report :\n", clf_report)
```

```
Classification Report :
               precision    recall  f1-score   support

           0       0.88      0.82      0.85        28
           1       0.86      0.91      0.88        33

    accuracy                           0.87        61
   macro avg       0.87      0.87      0.87        61
weighted avg       0.87      0.87      0.87        61
```

In [ ]:
```

```

In [42]:
```
y_pred_train_prob = log_clf.predict_proba(x_train)
fpr, tpr, thresh = roc_curve(y_train, y_pred_train_prob[:,1])
```

In [44]:
```
fpr
```

Out[44]:
```
array([0.        , 0.        , 0.        , 0.00909091, 0.00909091,
       0.01818182, 0.01818182, 0.02727273, 0.02727273, 0.03636364,
       0.03636364, 0.04545455, 0.04545455, 0.05454545, 0.05454545,
       0.06363636, 0.06363636, 0.07272727, 0.07272727, 0.08181818,
       0.08181818, 0.09090909, 0.09090909, 0.09090909, 0.09090909,
       0.1       , 0.1       , 0.11818182, 0.11818182, 0.12727273,
       0.12727273, 0.13636364, 0.13636364, 0.14545455, 0.14545455,
       0.16363636, 0.16363636, 0.17272727, 0.17272727, 0.21818182,
       0.21818182, 0.22727273, 0.22727273, 0.23636364, 0.23636364,
       0.26363636, 0.26363636, 0.28181818, 0.28181818, 0.30909091,
       0.30909091, 0.37272727, 0.37272727, 0.39090909, 0.39090909,
       0.42727273, 0.42727273, 0.47272727, 0.47272727, 0.5       ,
       0.5       , 0.64545455, 0.64545455, 1.        ])
```

In [46]:
```
tpr
```

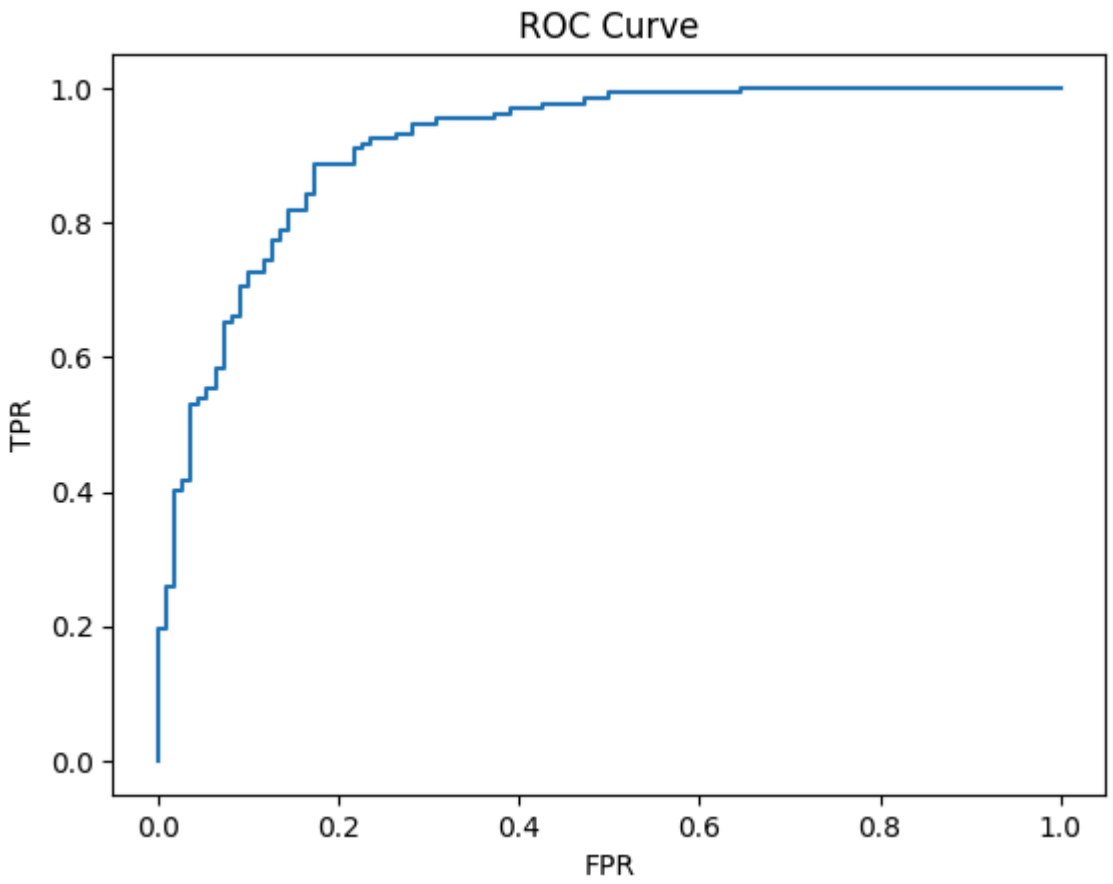
Out[46]:
```
array([0.        , 0.00757576, 0.1969697 , 0.1969697 , 0.25757576,
       0.25757576, 0.40151515, 0.40151515, 0.41666667, 0.41666667,
       0.53030303, 0.53030303, 0.53787879, 0.53787879, 0.5530303 ,
       0.5530303 , 0.58333333, 0.58333333, 0.65151515, 0.65151515,
       0.65909091, 0.65909091, 0.68181818, 0.6969697 , 0.70454545,
       0.70454545, 0.72727273, 0.72727273, 0.74242424, 0.74242424,
       0.77272727, 0.77272727, 0.78787879, 0.78787879, 0.81818182,
       0.81818182, 0.84090909, 0.84090909, 0.88636364, 0.88636364,
       0.90909091, 0.90909091, 0.91666667, 0.91666667, 0.92424242,
       0.92424242, 0.93181818, 0.93181818, 0.9469697 , 0.9469697 ,
       0.95454545, 0.95454545, 0.96212121, 0.96212121, 0.96969697,
       0.96969697, 0.97727273, 0.97727273, 0.98484848, 0.98484848,
       0.99242424, 0.99242424, 1.        , 1.        ])
```

In [48]:
```
thresh
```

Out[48]:
```
array([       inf, 0.99448755, 0.95321669, 0.95289286, 0.9378883 ,
       0.93209381, 0.88684339, 0.88657238, 0.88466559, 0.88273015,
       0.8191005 , 0.81674828, 0.81637076, 0.81158425, 0.8095993 ,
       0.80711642, 0.7850566 , 0.78488461, 0.76050925, 0.75211307,
       0.74879297, 0.74767768, 0.73921676, 0.73666853, 0.73457792,
       0.72893221, 0.70343623, 0.70122651, 0.68876316, 0.68636715,
       0.66671936, 0.6630252 , 0.6459502 , 0.64170872, 0.62897988,
       0.61801926, 0.60482762, 0.6046812 , 0.53592203, 0.52755954,
       0.51433146, 0.50748671, 0.50261164, 0.50184913, 0.50122575,
       0.49420772, 0.48432043, 0.45922821, 0.4445082 , 0.4118362 ,
       0.40306403, 0.25813904, 0.2455269 , 0.24301206, 0.22919007,
       0.21615925, 0.21538947, 0.14613663, 0.13687115, 0.11101719,
       0.10812541, 0.05359264, 0.05058366, 0.00170894])
```

In [52]:
```
plt.plot(fpr,tpr)
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC Curve")
```

Out[52]:
```
Text(0.5, 1.0, 'ROC Curve')
```



In [54]:
```
px.line(x = fpr, y = tpr, labels = {'x' : "FPR", 'y':"TPR"})
```

In [57]:
```
clf_report = classification_report(y_train, y_pred_train)
print("Classification Report :\n", clf_report)
```

```
Classification Report :
               precision    recall  f1-score   support

           0       0.89      0.75      0.82       110
           1       0.82      0.92      0.87       132

    accuracy                           0.85       242
   macro avg       0.86      0.84      0.84       242
weighted avg       0.85      0.85      0.85       242
```

In [59]:
```
np.where(tpr >= 0.94)
```

Out[59]:
```
(array([48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63],
       dtype=int64),)
```

In [61]:
```
tpr[48]
```

Out[61]:
```
0.946969696969697
```

In [63]:
```
thresh[48]
```

Out[63]: 0.4445081983336496

In [64]:
```
fpr[48]
```

Out[64]: 0.2818181818181818

In [67]:
```
roc_auc_score(y_train, y_pred_train_prob[:,1])
```

Out[67]: 0.9151515151515152

In [68]:
```
prob = y_pred_train_prob[:,1]
```

In [70]:
```
logit_values = np.log(prob/(1-prob))
```

In [ ]: