In [ ]:

```python
# Tyler Boudreau

# Import Required packages as needed throughout
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt

# Location of Dataset must be set
df = pd.read_csv('C:\\Users\\Tyler\\Downloads\\Heart_disease_cleveland_new.csv')
print(df)
df.head(10)
```

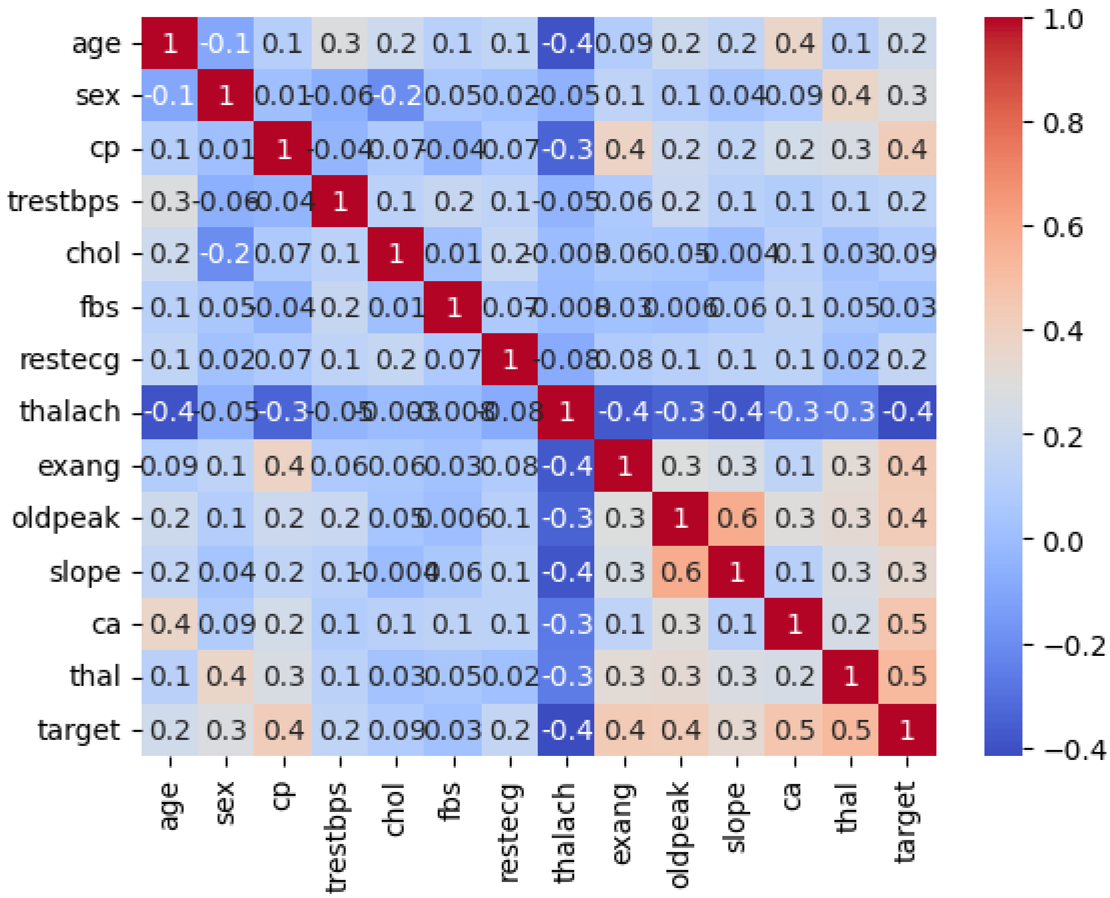
```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0     63    1   0       145   233    1        2      150      0      2.3   
1     67    1   3       160   286    0        2      108      1      1.5   
2     67    1   3       120   229    0        2      129      1      2.6   
3     37    1   2       130   250    0        0      187      0      3.5   
4     41    0   1       130   204    0        2      172      0      1.4   
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...   
298   45    1   0       110   264    0        0      132      0      1.2   
299   68    1   3       144   193    1        0      141      0      3.4   
300   57    1   3       130   131    0        0      115      1      1.2   
301   57    0   1       130   236    0        2      174      0      0.0   
302   38    1   2       138   175    0        0      173      0      0.0   

     slope  ca  thal  target  
0        2   0     2       0  
1        1   3     1       1  
2        1   2     3       1  
3        2   0     1       0  
4        0   0     1       0  
..     ...  ..   ...     ...  
298      1   0     3       1  
299      1   2     3       1  
300      1   1     3       1  
301      1   1     1       1  
302      0   0     1       0  

[303 rows x 14 columns]
```

Out[ ]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	ta
0	63	1	0	145	233	1	2	150	0	2.3	2	0	2	
1	67	1	3	160	286	0	2	108	1	1.5	1	3	1	
2	67	1	3	120	229	0	2	129	1	2.6	1	2	3	
3	37	1	2	130	250	0	0	187	0	3.5	2	0	1	
4	41	0	1	130	204	0	2	172	0	1.4	0	0	1	
5	56	1	1	120	236	0	0	178	0	0.8	0	0	1	
6	62	0	3	140	268	0	2	160	0	3.6	2	2	1	
7	57	0	3	120	354	0	0	163	1	0.6	0	0	1	
8	63	1	3	130	254	0	2	147	0	1.4	1	1	3	
9	53	1	3	140	203	1	2	155	1	3.1	2	0	3	

In [ ]:

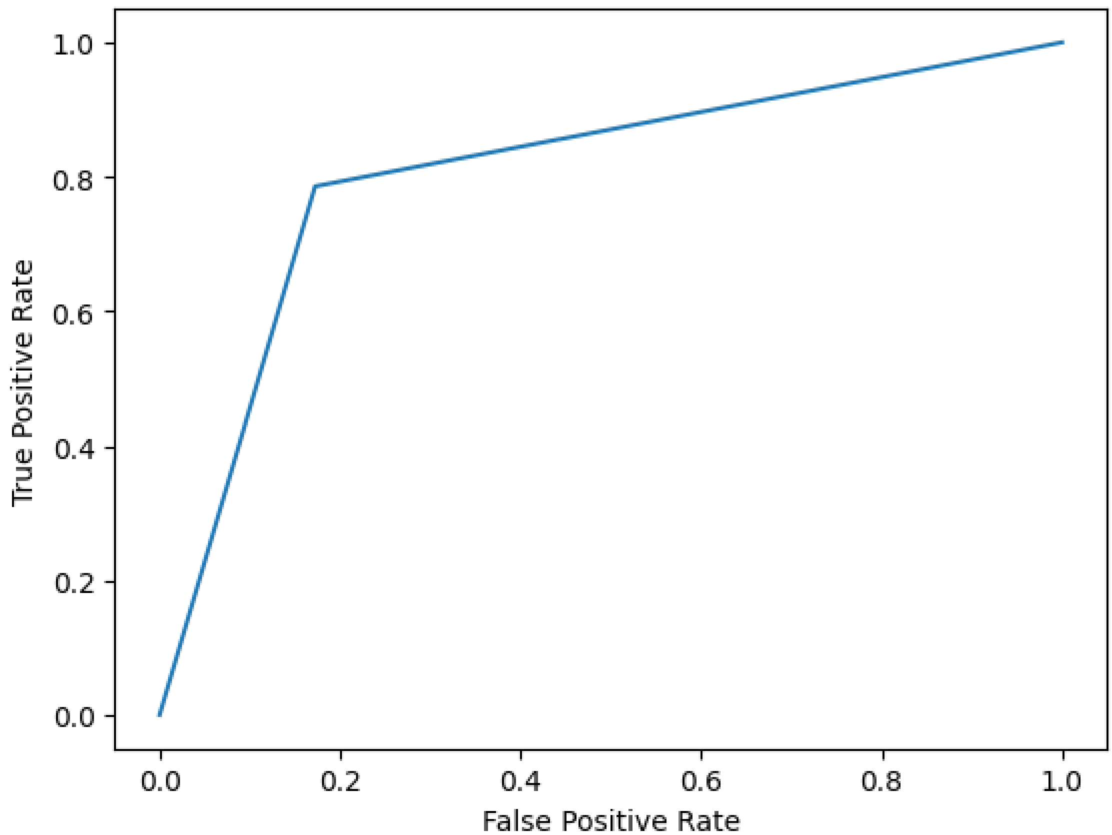
```
# Create Correlation Matrix to check for Collinearity
CorrMatrix1 = df.corr()

sb.heatmap(CorrMatrix1, cmap="coolwarm", annot=True, fmt=".1g")
plt.show()
```



In [ ]:
```python
X=df.iloc[:,0:13]
X
y=df['target']

# Create Supervised Train and Unsupervised Test Partitions
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, roc_curve
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=.33, random_state

# Count number 0 and 1 prediction values for Heart Disease, 0 being absent, 1 being
y.value_counts()

def plot_roc_curve(true_y,y_predt):
    fpr, tpr, thresholds = roc_curve(true_y,y_predt)
    plt.plot(fpr,tpr)
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
```

In [ ]:
```python
# Logistic Regression Model
from sklearn.linear_model import LogisticRegression
logModel=LogisticRegression(max_iter=10000)
logModel.fit(X_train, y_train)
pred_y = logModel.predict(X_test)
from sklearn.metrics import accuracy_score
print('Logistic Regression Model Accuracy: {0:0.4f}'.format(accuracy_score(y_test,p
predresult1 = pd.DataFrame({"Actual" : y_test, "Predicted" : pred_y})
print(predresult1)

plot_roc_curve(y_test,pred_y)
print(f"Logistic Regression AUC Score: {roc_auc_score(y_test,pred_y)}")
```

```
Logistic Regression Model Accuracy: 81.0000 %
     Actual  Predicted
166       0          0
182       0          1
292       1          1
22        1          0
179       0          1
..      ...        ...
41        0          0
282       1          1
200       0          0
174       1          1
18        0          0

[100 rows x 2 columns]
Logistic Regression AUC Score: 0.8066502463054187
```



In [ ]:

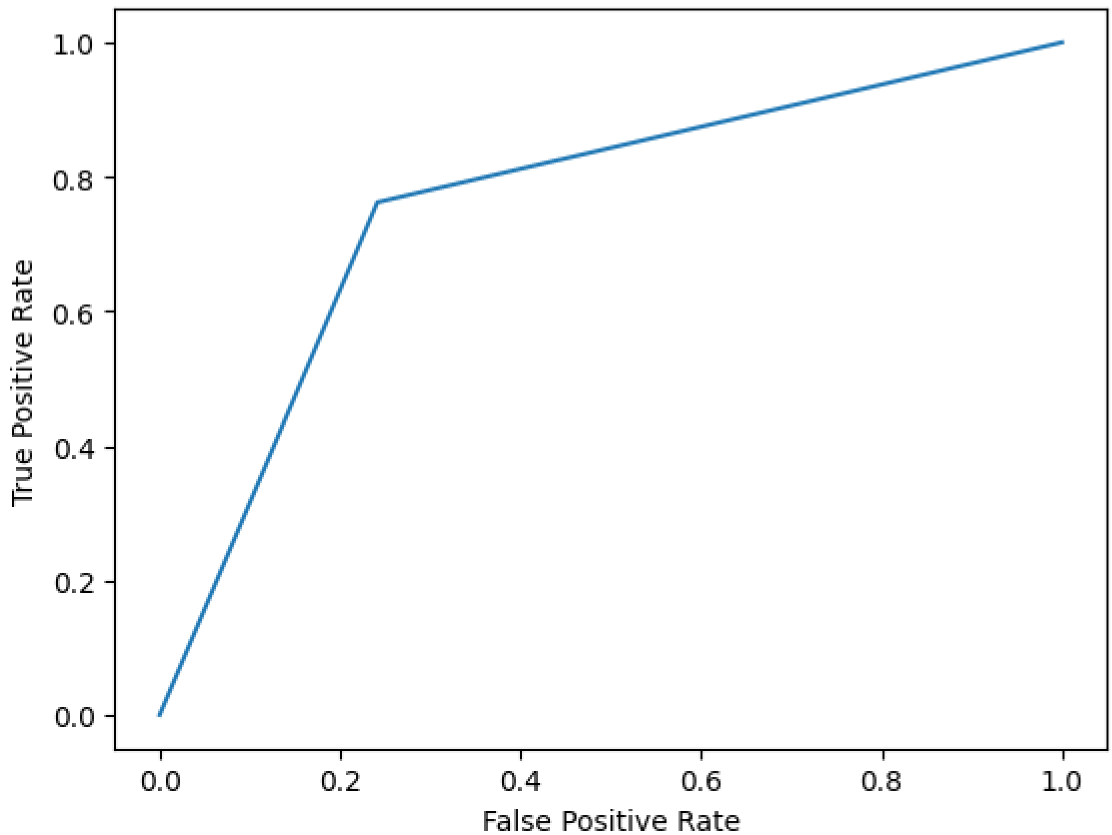
```
# Testing Logistic Regression model on example data
XTestValues1 = pd.DataFrame(np.array([[63,1,0,145,233,1,2,150,0,2.3,2,0,2]]), colum
XTestValues2 = pd.DataFrame(np.array([[67,1,3,160,286,0,2,108,1,1.5,1,3,1]]), colum
Logmodelprediction1 = logModel.predict(XTestValues1)
Logmodelprediction2 = logModel.predict(XTestValues2)
print(Logmodelprediction1)
print(Logmodelprediction2)
```

```
[0]
[1]
```

In [ ]:

```
# LightGBM Model
import lightgbm as lgb
clf = lgb.LGBMClassifier()
clf.fit(X_train, y_train)
y_pred=clf.predict(X_test)
accuracy=accuracy_score(y_pred, y_test)
print('LightGBM Model Accuracy: {0:0.4f}'.format(accuracy_score(y_test, y_pred)*100
predresult2 = pd.DataFrame({"Actual" : y_test, "Predicted" : y_pred})
print(predresult2)

LightGBMPred1 = clf.predict(XTestValues1)
LightGBMPred2 = clf.predict(XTestValues2)

print(LightGBMPred1)
print(LightGBMPred2)

plot_roc_curve(y_test,y_pred)
print(f"LightGBM Model AUC Score: {roc_auc_score(y_test,y_pred)}")
```

```
LightGBM Model Accuracy: 76.0000 %
     Actual  Predicted
166       0          0
182       0          1
292       1          1
22        1          0
179       0          1
..      ...        ...
41        0          0
282       1          1
200       0          0
174       1          1
18        0          0

[100 rows x 2 columns]
[0]
[1]
LightGBM Model AUC Score: 0.7602627257799671
```

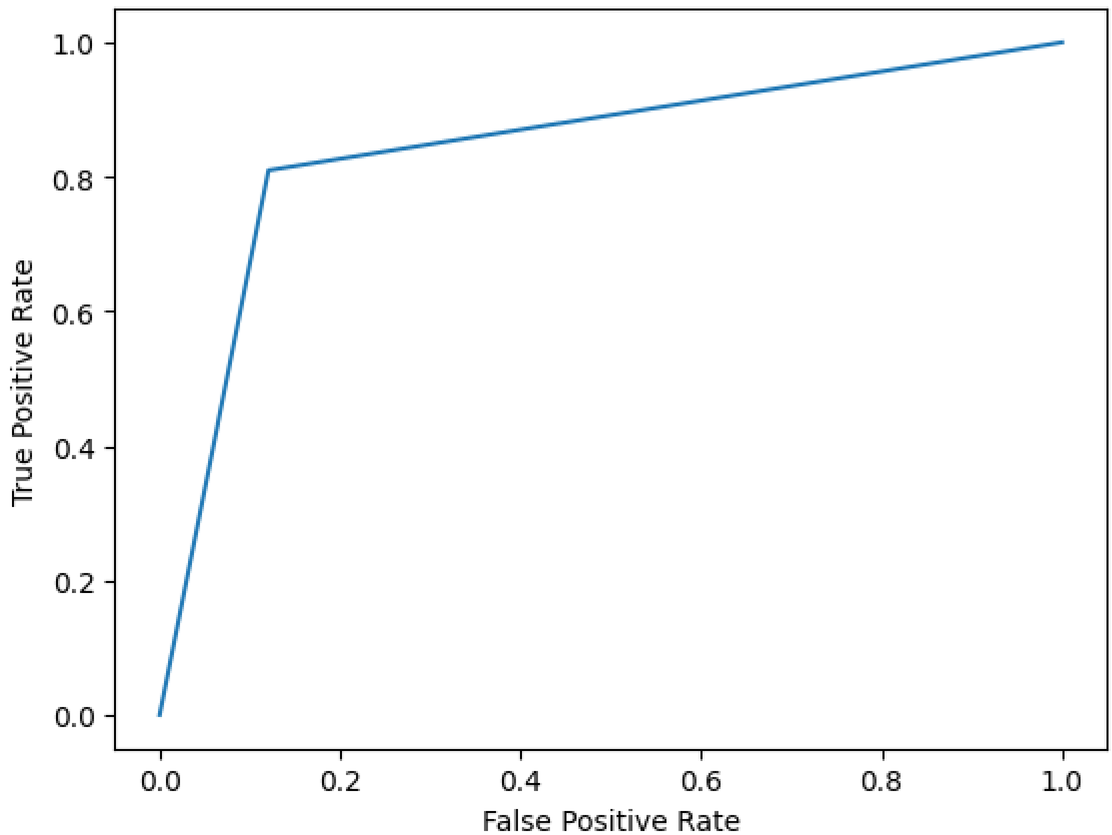


In [ ]:

```python
# Random Forest Model
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators = 100)

# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
clf.fit(X_train, y_train)

# performing predictions on the test dataset
y_pred8 = clf.predict(X_test)
 # metrics are used to find accuracy or error
```

```python
from sklearn import metrics
print()

# using metrics module for accuracy calculation
print("Random Forest Accuracy:",metrics.accuracy_score(y_test, y_pred8)*100,"%")

predresult3 = pd.DataFrame({"Actual" : y_test, "Predicted" : y_pred8})
print(predresult3)
RandomForestPred1 = clf.predict(XTestValues1)
RandomForestPred2 = clf.predict(XTestValues2)

print(RandomForestPred1)
print(RandomForestPred2)


plot_roc_curve(y_test,y_pred8)
print(f"Random Forest AUC Score: {roc_auc_score(y_test,y_pred8)}")
```

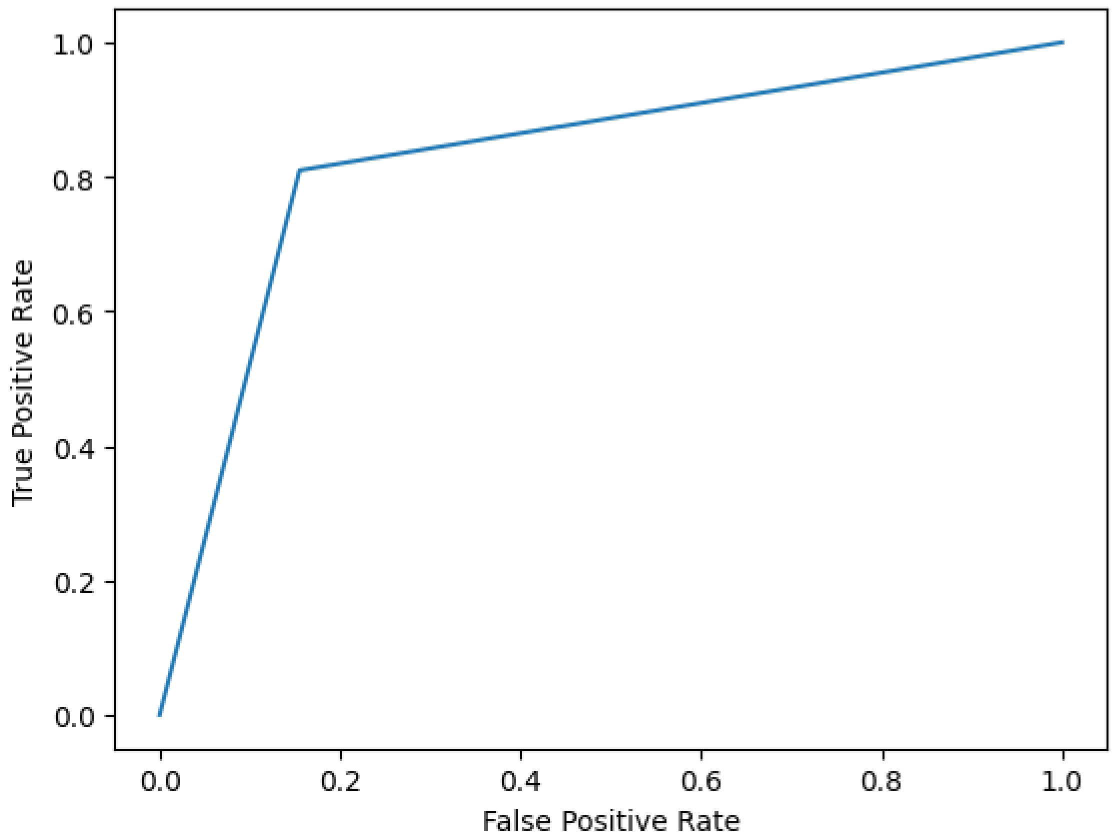
```
Random Forest Accuracy: 85.0 %
     Actual  Predicted
166       0          0
182       0          0
292       1          1
22        1          0
179       0          1
..      ...        ...
41        0          0
282       1          1
200       0          0
174       1          1
18        0          0

[100 rows x 2 columns]
[0]
[1]
Random Forest AUC Score: 0.8444170771756978
```



In [ ]:

```
# ExtraTree Model
from sklearn.ensemble import ExtraTreesClassifier
clf = ExtraTreesClassifier(n_estimators=100,max_depth=6,min_samples_split=2,min_wei
clf.fit(X_train, y_train)
print("ExtraTree Classifier Accuracy:",clf.score(X_test, y_test)*100,"%")
y_pred9 = clf.predict(X_test)
predresult4 = pd.DataFrame({"Actual" : y_test, "Predicted" : y_pred9})
print(predresult4)
ExtraTreePred1 = clf.predict(XTestValues1)
ExtraTreePred2 = clf.predict(XTestValues2)

print(ExtraTreePred1)
print(ExtraTreePred2)

plot_roc_curve(y_test,y_pred9)
print(f"ExtraTree AUC Score: {roc_auc_score(y_test,y_pred9)}")
```

```
ExtraTree Classifier Accuracy: 83.0 %
     Actual  Predicted
166       0          0
182       0          1
292       1          1
22        1          0
179       0          1
..      ...        ...
41        0          1
282       1          1
200       0          0
174       1          1
18        0          0

[100 rows x 2 columns]
[0]
[1]
ExtraTree AUC Score: 0.827175697865353
```

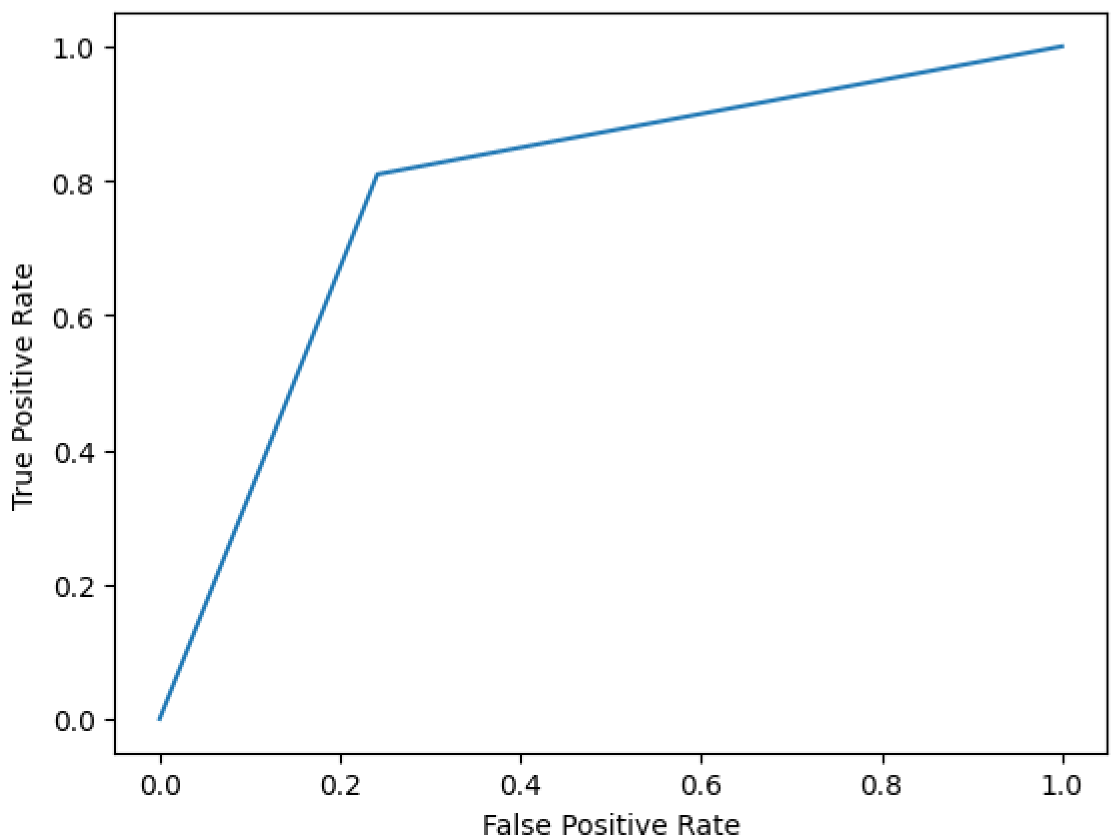


In [ ]:

```
# XGBoost Model
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
model = XGBClassifier(eval_metric='mlogloss')
model.fit(X_train, y_train)
y_pred1 = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred1)
print("XGBoost Accuracy:",accuracy*100,"%")
predresult5 = pd.DataFrame({"Actual" : y_test, "Predicted" : y_pred1})
print(predresult5)
XGBoostPred1 = model.predict(XTestValues1)
```

```python
XGBoostPred2 = model.predict(XTestValues2)

print(XGBoostPred1)
print(XGBoostPred2)

plot_roc_curve(y_test,y_pred1)
print(f"XGBoost Model AUC Score: {roc_auc_score(y_test,y_pred1)}")
```

```
XGBoost Accuracy: 78.0 %
     Actual  Predicted
166       0          0
182       0          1
292       1          1
22        1          0
179       0          1
..      ...        ...
41        0          0
282       1          1
200       0          0
174       1          1
18        0          0

[100 rows x 2 columns]
[0]
[1]
XGBoost Model AUC Score: 0.784072249589491
```



In [ ]:
```python
# Setup TensorFLow Model
from tensorflow.keras.models import Sequential #Helps to create Forward and backwar
from tensorflow.keras.layers import Dense #Helps to create neurons in ANN
```

In [ ]:

```
# Continue TensorFlow Setup
classifier=Sequential()
classifier.add(Dense(units=11,activation='relu'))
classifier.add(Dense(units=7,activation='relu'))
classifier.add(Dense(units=6,activation='relu'))
## Adding the output layer
classifier.add(Dense(units=1,activation='sigmoid'))
classifier.compile(optimizer='adam',loss="binary_crossentropy",metrics=["accuracy"]
#classifier.compile(optimizer=opt,loss="binary_crossentropy",metrics=["accuracy"])
```

In [ ]:

```
# TensorFlow continued setup
import tensorflow as tf
early_stopping=tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    min_delta=0.0001,
    patience=20,
    verbose=1,
    mode="auto",
    baseline=None,
    restore_best_weights=False,
)
```

In [ ]:

```
# Runs TensorFlow model up to 1000 iterations or until optimal value is found
model_history=classifier.fit(X_train,y_train,validation_split=0.30,batch_size=10,ep
```

```
Epoch 1/1000
15/15 [==============================] - 1s 14ms/step - loss: 23.0328 - accuracy: 0.
5141 - val_loss: 14.5817 - val_accuracy: 0.5410
Epoch 2/1000
15/15 [==============================] - 0s 5ms/step - loss: 9.8461 - accuracy: 0.50
70 - val_loss: 3.1870 - val_accuracy: 0.5082
Epoch 3/1000
15/15 [==============================] - 0s 5ms/step - loss: 2.1434 - accuracy: 0.59
15 - val_loss: 2.3211 - val_accuracy: 0.5902
Epoch 4/1000
15/15 [==============================] - 0s 4ms/step - loss: 1.5824 - accuracy: 0.59
15 - val_loss: 1.1114 - val_accuracy: 0.6557
Epoch 5/1000
15/15 [==============================] - 0s 5ms/step - loss: 1.0223 - accuracy: 0.66
20 - val_loss: 0.9098 - val_accuracy: 0.5410
Epoch 6/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.8873 - accuracy: 0.65
49 - val_loss: 0.8776 - val_accuracy: 0.6066
Epoch 7/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.8215 - accuracy: 0.68
31 - val_loss: 0.8475 - val_accuracy: 0.5574
Epoch 8/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.7283 - accuracy: 0.66
90 - val_loss: 0.7466 - val_accuracy: 0.5410
Epoch 9/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.6991 - accuracy: 0.65
49 - val_loss: 0.7034 - val_accuracy: 0.5738
Epoch 10/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.6971 - accuracy: 0.66
90 - val_loss: 0.7558 - val_accuracy: 0.5738
Epoch 11/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.6733 - accuracy: 0.57
04 - val_loss: 0.7038 - val_accuracy: 0.5902
Epoch 12/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.6858 - accuracy: 0.64
08 - val_loss: 0.7925 - val_accuracy: 0.6230
Epoch 13/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.6528 - accuracy: 0.64
08 - val_loss: 0.7183 - val_accuracy: 0.5902
Epoch 14/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.6279 - accuracy: 0.69
72 - val_loss: 0.7558 - val_accuracy: 0.6230
Epoch 15/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.7016 - accuracy: 0.64
79 - val_loss: 0.6787 - val_accuracy: 0.6557
Epoch 16/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.6466 - accuracy: 0.62
68 - val_loss: 0.6925 - val_accuracy: 0.5902
Epoch 17/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.6003 - accuracy: 0.64
79 - val_loss: 0.6398 - val_accuracy: 0.6557
Epoch 18/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.6213 - accuracy: 0.70
42 - val_loss: 0.7500 - val_accuracy: 0.6393
Epoch 19/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.6425 - accuracy: 0.64
```

```
79 - val_loss: 0.6919 - val_accuracy: 0.6721
Epoch 20/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.6049 - accuracy: 0.67
61 - val_loss: 0.6762 - val_accuracy: 0.5902
Epoch 21/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.5845 - accuracy: 0.71
13 - val_loss: 0.6627 - val_accuracy: 0.5738
Epoch 22/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5913 - accuracy: 0.71
83 - val_loss: 0.6779 - val_accuracy: 0.5738
Epoch 23/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.5842 - accuracy: 0.72
54 - val_loss: 0.6596 - val_accuracy: 0.6230
Epoch 24/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.6205 - accuracy: 0.65
49 - val_loss: 0.6174 - val_accuracy: 0.6885
Epoch 25/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.5806 - accuracy: 0.67
61 - val_loss: 0.6400 - val_accuracy: 0.6393
Epoch 26/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.5669 - accuracy: 0.71
83 - val_loss: 0.6502 - val_accuracy: 0.6393
Epoch 27/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.5689 - accuracy: 0.68
31 - val_loss: 0.6461 - val_accuracy: 0.6066
Epoch 28/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5445 - accuracy: 0.73
24 - val_loss: 0.6229 - val_accuracy: 0.6230
Epoch 29/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5558 - accuracy: 0.74
65 - val_loss: 0.6302 - val_accuracy: 0.6066
Epoch 30/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5543 - accuracy: 0.68
31 - val_loss: 0.6563 - val_accuracy: 0.6557
Epoch 31/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5637 - accuracy: 0.71
83 - val_loss: 0.6424 - val_accuracy: 0.6885
Epoch 32/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5568 - accuracy: 0.73
24 - val_loss: 0.7102 - val_accuracy: 0.6066
Epoch 33/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.5718 - accuracy: 0.64
08 - val_loss: 0.6541 - val_accuracy: 0.7213
Epoch 34/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5684 - accuracy: 0.70
42 - val_loss: 0.6402 - val_accuracy: 0.6557
Epoch 35/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5279 - accuracy: 0.71
13 - val_loss: 0.6263 - val_accuracy: 0.7049
Epoch 36/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5555 - accuracy: 0.71
13 - val_loss: 0.6069 - val_accuracy: 0.6557
Epoch 37/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5352 - accuracy: 0.72
54 - val_loss: 0.5869 - val_accuracy: 0.6721
Epoch 38/1000
```

```
15/15 [==============================] - 0s 5ms/step - loss: 0.5213 - accuracy: 0.72
54 - val_loss: 0.6157 - val_accuracy: 0.6885
Epoch 39/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5111 - accuracy: 0.73
94 - val_loss: 0.5871 - val_accuracy: 0.6721
Epoch 40/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5157 - accuracy: 0.75
35 - val_loss: 0.5900 - val_accuracy: 0.6885
Epoch 41/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5450 - accuracy: 0.71
13 - val_loss: 0.5810 - val_accuracy: 0.7377
Epoch 42/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5073 - accuracy: 0.73
94 - val_loss: 0.5753 - val_accuracy: 0.6885
Epoch 43/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5150 - accuracy: 0.71
13 - val_loss: 0.5817 - val_accuracy: 0.7541
Epoch 44/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4993 - accuracy: 0.76
06 - val_loss: 0.5750 - val_accuracy: 0.6885
Epoch 45/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4897 - accuracy: 0.75
35 - val_loss: 0.5647 - val_accuracy: 0.7377
Epoch 46/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5014 - accuracy: 0.76
06 - val_loss: 0.5948 - val_accuracy: 0.7049
Epoch 47/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.4857 - accuracy: 0.73
94 - val_loss: 0.5575 - val_accuracy: 0.7377
Epoch 48/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.4992 - accuracy: 0.76
76 - val_loss: 0.6704 - val_accuracy: 0.6885
Epoch 49/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5335 - accuracy: 0.73
24 - val_loss: 0.5659 - val_accuracy: 0.6721
Epoch 50/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4934 - accuracy: 0.76
06 - val_loss: 0.5918 - val_accuracy: 0.6721
Epoch 51/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5183 - accuracy: 0.75
35 - val_loss: 0.6116 - val_accuracy: 0.7049
Epoch 52/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5402 - accuracy: 0.70
42 - val_loss: 0.5399 - val_accuracy: 0.7213
Epoch 53/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.4888 - accuracy: 0.77
46 - val_loss: 0.6229 - val_accuracy: 0.6885
Epoch 54/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5252 - accuracy: 0.71
13 - val_loss: 0.6171 - val_accuracy: 0.6393
Epoch 55/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5079 - accuracy: 0.70
42 - val_loss: 0.5856 - val_accuracy: 0.7049
Epoch 56/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4865 - accuracy: 0.73
94 - val_loss: 0.5385 - val_accuracy: 0.7049
```

```
Epoch 57/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4953 - accuracy: 0.78
17 - val_loss: 0.6043 - val_accuracy: 0.7213
Epoch 58/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5219 - accuracy: 0.73
94 - val_loss: 0.6085 - val_accuracy: 0.6885
Epoch 59/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4799 - accuracy: 0.77
46 - val_loss: 0.5982 - val_accuracy: 0.7049
Epoch 60/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4747 - accuracy: 0.76
76 - val_loss: 0.6155 - val_accuracy: 0.6393
Epoch 61/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4805 - accuracy: 0.76
76 - val_loss: 0.6110 - val_accuracy: 0.7049
Epoch 62/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4787 - accuracy: 0.76
76 - val_loss: 0.5754 - val_accuracy: 0.7213
Epoch 63/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4567 - accuracy: 0.76
76 - val_loss: 0.6291 - val_accuracy: 0.6885
Epoch 64/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.5146 - accuracy: 0.72
54 - val_loss: 0.5468 - val_accuracy: 0.6721
Epoch 65/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4845 - accuracy: 0.73
94 - val_loss: 0.5438 - val_accuracy: 0.7049
Epoch 66/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.4509 - accuracy: 0.78
87 - val_loss: 0.5303 - val_accuracy: 0.7213
Epoch 67/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4800 - accuracy: 0.69
72 - val_loss: 0.5572 - val_accuracy: 0.7213
Epoch 68/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4720 - accuracy: 0.73
24 - val_loss: 0.5262 - val_accuracy: 0.7213
Epoch 69/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4689 - accuracy: 0.76
76 - val_loss: 0.5403 - val_accuracy: 0.7377
Epoch 70/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4575 - accuracy: 0.76
76 - val_loss: 0.5731 - val_accuracy: 0.7049
Epoch 71/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4451 - accuracy: 0.77
46 - val_loss: 0.6366 - val_accuracy: 0.6393
Epoch 72/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4904 - accuracy: 0.71
13 - val_loss: 0.6235 - val_accuracy: 0.6393
Epoch 73/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4936 - accuracy: 0.76
06 - val_loss: 0.5438 - val_accuracy: 0.7377
Epoch 74/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4488 - accuracy: 0.76
76 - val_loss: 0.5533 - val_accuracy: 0.7213
Epoch 75/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4506 - accuracy: 0.76
```

```
76 - val_loss: 0.5604 - val_accuracy: 0.7213
Epoch 76/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4613 - accuracy: 0.78
87 - val_loss: 0.5126 - val_accuracy: 0.7541
Epoch 77/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.4607 - accuracy: 0.78
87 - val_loss: 0.5123 - val_accuracy: 0.7541
Epoch 78/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.4849 - accuracy: 0.72
54 - val_loss: 0.5052 - val_accuracy: 0.7541
Epoch 79/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4308 - accuracy: 0.76
06 - val_loss: 0.5391 - val_accuracy: 0.7541
Epoch 80/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4468 - accuracy: 0.79
58 - val_loss: 0.5189 - val_accuracy: 0.7213
Epoch 81/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4297 - accuracy: 0.76
06 - val_loss: 0.5091 - val_accuracy: 0.7377
Epoch 82/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4333 - accuracy: 0.77
46 - val_loss: 0.5363 - val_accuracy: 0.7541
Epoch 83/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4267 - accuracy: 0.76
76 - val_loss: 0.5109 - val_accuracy: 0.7541
Epoch 84/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4213 - accuracy: 0.79
58 - val_loss: 0.5240 - val_accuracy: 0.7377
Epoch 85/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4349 - accuracy: 0.77
46 - val_loss: 0.5281 - val_accuracy: 0.6885
Epoch 86/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4590 - accuracy: 0.76
06 - val_loss: 0.5482 - val_accuracy: 0.7049
Epoch 87/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4394 - accuracy: 0.78
87 - val_loss: 0.5247 - val_accuracy: 0.7049
Epoch 88/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4272 - accuracy: 0.76
76 - val_loss: 0.5528 - val_accuracy: 0.7377
Epoch 89/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4405 - accuracy: 0.75
35 - val_loss: 0.4980 - val_accuracy: 0.7377
Epoch 90/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4422 - accuracy: 0.76
06 - val_loss: 0.4937 - val_accuracy: 0.7705
Epoch 91/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4430 - accuracy: 0.73
24 - val_loss: 0.5165 - val_accuracy: 0.7541
Epoch 92/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4293 - accuracy: 0.78
17 - val_loss: 0.6073 - val_accuracy: 0.6557
Epoch 93/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4302 - accuracy: 0.76
76 - val_loss: 0.4974 - val_accuracy: 0.7213
Epoch 94/1000
```

```
15/15 [==============================] - 0s 5ms/step - loss: 0.4016 - accuracy: 0.80
28 - val_loss: 0.5582 - val_accuracy: 0.7377
Epoch 95/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4324 - accuracy: 0.78
87 - val_loss: 0.5145 - val_accuracy: 0.7049
Epoch 96/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4264 - accuracy: 0.80
99 - val_loss: 0.4718 - val_accuracy: 0.7377
Epoch 97/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4667 - accuracy: 0.76
76 - val_loss: 0.5409 - val_accuracy: 0.8033
Epoch 98/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.4833 - accuracy: 0.77
46 - val_loss: 0.5259 - val_accuracy: 0.7377
Epoch 99/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4439 - accuracy: 0.76
06 - val_loss: 0.5535 - val_accuracy: 0.7213
Epoch 100/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4473 - accuracy: 0.76
76 - val_loss: 0.5170 - val_accuracy: 0.7541
Epoch 101/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4514 - accuracy: 0.78
87 - val_loss: 0.5274 - val_accuracy: 0.7049
Epoch 102/1000
15/15 [==============================] - 0s 7ms/step - loss: 0.4080 - accuracy: 0.78
87 - val_loss: 0.4964 - val_accuracy: 0.7213
Epoch 103/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4181 - accuracy: 0.80
28 - val_loss: 0.5721 - val_accuracy: 0.7049
Epoch 104/1000
15/15 [==============================] - 0s 7ms/step - loss: 0.4285 - accuracy: 0.81
69 - val_loss: 0.4845 - val_accuracy: 0.7541
Epoch 105/1000
15/15 [==============================] - 0s 7ms/step - loss: 0.4071 - accuracy: 0.80
28 - val_loss: 0.5080 - val_accuracy: 0.7377
Epoch 106/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4056 - accuracy: 0.80
28 - val_loss: 0.4930 - val_accuracy: 0.7705
Epoch 107/1000
15/15 [==============================] - 0s 7ms/step - loss: 0.4070 - accuracy: 0.80
28 - val_loss: 0.4975 - val_accuracy: 0.7705
Epoch 108/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.3984 - accuracy: 0.79
58 - val_loss: 0.5242 - val_accuracy: 0.7541
Epoch 109/1000
15/15 [==============================] - 0s 6ms/step - loss: 0.4207 - accuracy: 0.78
87 - val_loss: 0.5056 - val_accuracy: 0.7377
Epoch 110/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4191 - accuracy: 0.78
87 - val_loss: 0.5221 - val_accuracy: 0.7377
Epoch 111/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4417 - accuracy: 0.79
58 - val_loss: 0.5589 - val_accuracy: 0.6721
Epoch 112/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4685 - accuracy: 0.76
06 - val_loss: 0.5340 - val_accuracy: 0.7541
```

```
Epoch 113/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4166 - accuracy: 0.78
17 - val_loss: 0.4983 - val_accuracy: 0.7377
Epoch 114/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4154 - accuracy: 0.78
17 - val_loss: 0.5686 - val_accuracy: 0.7049
Epoch 115/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4362 - accuracy: 0.78
17 - val_loss: 0.4873 - val_accuracy: 0.7541
Epoch 116/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.3958 - accuracy: 0.80
28 - val_loss: 0.4711 - val_accuracy: 0.7213
Epoch 117/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4150 - accuracy: 0.78
87 - val_loss: 0.5120 - val_accuracy: 0.7541
Epoch 118/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.4017 - accuracy: 0.79
58 - val_loss: 0.4754 - val_accuracy: 0.7213
Epoch 119/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.3904 - accuracy: 0.80
99 - val_loss: 0.4985 - val_accuracy: 0.7705
Epoch 120/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4001 - accuracy: 0.79
58 - val_loss: 0.5012 - val_accuracy: 0.7377
Epoch 121/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4029 - accuracy: 0.78
87 - val_loss: 0.4798 - val_accuracy: 0.7541
Epoch 122/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.3948 - accuracy: 0.79
58 - val_loss: 0.4982 - val_accuracy: 0.7377
Epoch 123/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.4004 - accuracy: 0.79
58 - val_loss: 0.4658 - val_accuracy: 0.7377
Epoch 124/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4144 - accuracy: 0.82
39 - val_loss: 0.5498 - val_accuracy: 0.7049
Epoch 125/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4085 - accuracy: 0.78
87 - val_loss: 0.4751 - val_accuracy: 0.7705
Epoch 126/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4073 - accuracy: 0.77
46 - val_loss: 0.5113 - val_accuracy: 0.7705
Epoch 127/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.3762 - accuracy: 0.80
99 - val_loss: 0.4993 - val_accuracy: 0.7705
Epoch 128/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4071 - accuracy: 0.79
58 - val_loss: 0.4788 - val_accuracy: 0.7213
Epoch 129/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.3799 - accuracy: 0.81
69 - val_loss: 0.4728 - val_accuracy: 0.7377
Epoch 130/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.3854 - accuracy: 0.80
99 - val_loss: 0.4762 - val_accuracy: 0.7541
Epoch 131/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.4061 - accuracy: 0.80
```

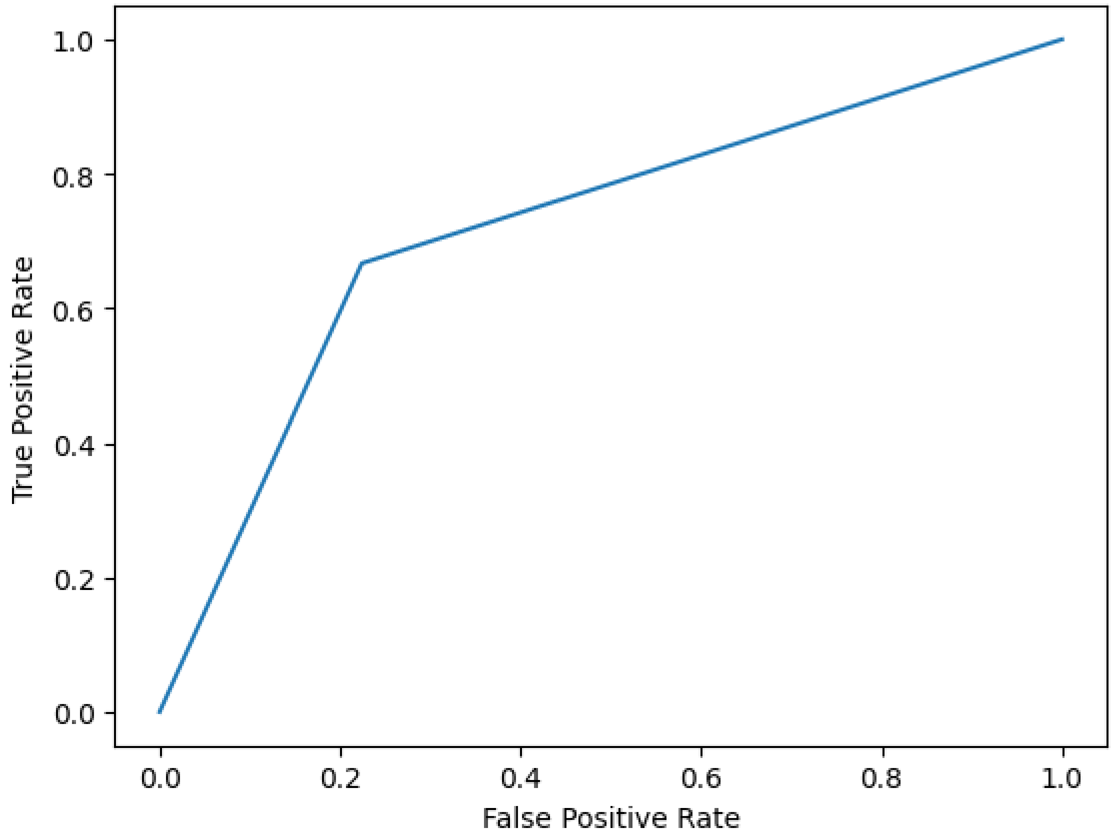
```
99 - val_loss: 0.4694 - val_accuracy: 0.7541
Epoch 132/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.3843 - accuracy: 0.81
69 - val_loss: 0.4921 - val_accuracy: 0.7541
Epoch 133/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.3709 - accuracy: 0.80
99 - val_loss: 0.5842 - val_accuracy: 0.6885
Epoch 134/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.4365 - accuracy: 0.80
99 - val_loss: 0.4980 - val_accuracy: 0.7705
Epoch 135/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4232 - accuracy: 0.79
58 - val_loss: 0.5672 - val_accuracy: 0.7049
Epoch 136/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4036 - accuracy: 0.78
87 - val_loss: 0.4915 - val_accuracy: 0.7869
Epoch 137/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.3986 - accuracy: 0.81
69 - val_loss: 0.4946 - val_accuracy: 0.7705
Epoch 138/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.3735 - accuracy: 0.82
39 - val_loss: 0.5121 - val_accuracy: 0.7213
Epoch 139/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.3784 - accuracy: 0.85
92 - val_loss: 0.4923 - val_accuracy: 0.7541
Epoch 140/1000
15/15 [==============================] - 0s 4ms/step - loss: 0.3793 - accuracy: 0.83
10 - val_loss: 0.4773 - val_accuracy: 0.7869
Epoch 141/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.3914 - accuracy: 0.81
69 - val_loss: 0.4869 - val_accuracy: 0.7213
Epoch 142/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.3835 - accuracy: 0.82
39 - val_loss: 0.4794 - val_accuracy: 0.7705
Epoch 143/1000
15/15 [==============================] - 0s 5ms/step - loss: 0.4157 - accuracy: 0.80
28 - val_loss: 0.5044 - val_accuracy: 0.7541
Epoch 143: early stopping
```

In [ ]:
```
# Make predictions with model on test set
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5) # If greater than .5 then model returns True or present for
```

```
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
```

In [ ]:
```
# Calculate the Accuracy
from sklearn.metrics import accuracy_score
score=accuracy_score(y_pred,y_test)
print("TensorFLow Accuracy:",score*100,"%")

TensorFlowPred1 = classifier.predict(XTestValues1)
TensorFlowPred2 = classifier.predict(XTestValues2)

plot_roc_curve(y_test,y_pred)
print(f"TensorFlow AUC Score: {roc_auc_score(y_test,y_pred)}")
```

```
TensorFLow Accuracy: 73.0 %
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
TensorFlow AUC Score: 0.7212643678160918
```



In [ ]:

```
# Make predictions with example test values
print(TensorFlowPred1)
print(TensorFlowPred2)
```

```
[[0.10073901]]
[[0.9455779]]
```

In [ ]:

```
from sklearn.model_selection import RepeatedKFold
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge

# Define K-Fold Cross Validation
#cv = RepeatedKFold(n_splits=203,n_repeats=3,random_state=1)

# Define predictor and target variables
X = df[["age",'sex','cp','trestbps','chol','fbs','restecg','thalach','exang','oldpe
y = df["target"]

# Linear Regression
LinearModel1 = LinearRegression().fit(X_train, y_train)

LinearModel1.predict(X_test)
accuracy = LinearModel1.score(X_test,y_test)

print('The predicted accuracy for Linear Regression is: {0:0.4f}'.format((accuracy*
```

```python
# Ridge Model
RidgeModel1 = Ridge(alpha=10)

RidgeModel1.fit(X_train,y_train)

accuracy = RidgeModel1.score(X_test,y_test)

print('The Predicted accuracy for the Ridge Model is: {0:0.4f}'.format((accuracy*10
```

```
The predicted accuracy for Linear Regression is: 44.6637 %

The Predicted accuracy for the Ridge Model is: 45.1017 %
```