```
19SE02IT058                                                  SEIT4031
```

Practical-09

In [1]:

```
import numpy as np import
pandas as pd import
matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder,StandardScaler,binarize
from sklearn.model_selection import train_test_split from
sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,roc_curve,auc,roc_auc_score,accuracy
import os
```

In [2]:

```
df = pd.read_csv('framingham.csv').dropna()
with pd.option_context('display.max_rows',6):
display(df)
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	d
0	1	39	4.0	0	0.0	0.0	0	0	
1	0	46	2.0	0	0.0	0.0	0	0	
2	1	48	1.0	1	20.0	0.0	0	0	
...	...	...	...	...	...	...	...	...	
4233	1	50	1.0	1	1.0	0.0	0	1	
4234	1	51	3.0	1	43.0	0.0	0	0	
4237	0	52	2.0	0	0.0	0.0	0	0	

3656 rows × 16 columns

In [3]:

```
# Scaling x = df.loc[:,df.columns !=
'TenYearCHD'] y = df.TenYearCHD

scaler = StandardScaler()

x = scaler.fit_transform(x)

# Checking the correlation
temp_df = pd.DataFrame(x,columns=df.columns[:-1])
temp_df['TenYearCHD'] = y scaled_df =
temp_df.dropna()
```

```
# with pd.option_context('display.max_rows',6):
#     display(temp_df)
target_corr_q3 = np.quantile(np.abs(scaled_df.corr()['TenYearCHD']),0.75)
target_corr = scaled_df.corr().loc['TenYearCHD']

# Get the most important feature(s) : by value of more than q3 of the correlation
selected_features = target_corr[np.abs(target_corr) > target_corr_q3].index[:-
1]. print(f'Selected Features {selected_features}')
```

```
Selected Features ['male' 'age' 'totChol']
```

In [4]:
```
x = scaled_df[selected_features].values
y = scaled_df.TenYearCHD.values
```

In [5]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0
```

In [6]:
```
model = LogisticRegression()
```

In [7]:
```
model.fit(x_train,y_train)
```

Out[7]:
```
LogisticRegression()
```

In [8]:
```
y_predict = model.predict(x_test)
```

In [9]:
```
print(f'Accuracy Score {model.score(x_test,y_test)}')
```

```
Accuracy Score 0.8537360890302067
```

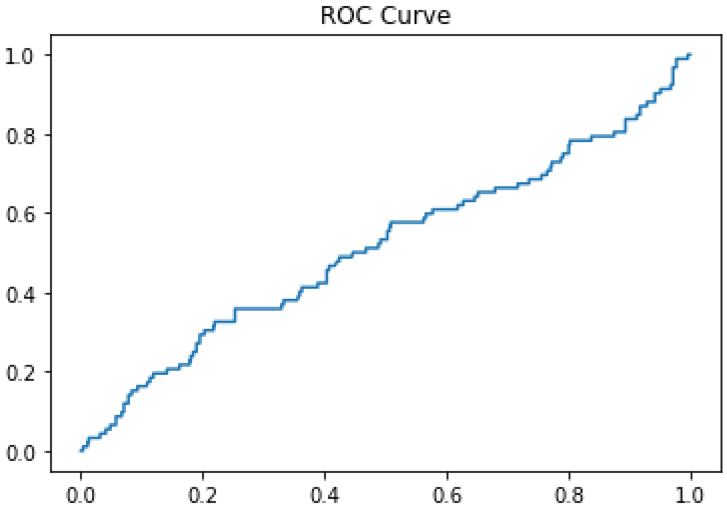
In [10]:
```
print(f'Accuracy Score {accuracy_score(y_test,y_predict)}')
```

```
Accuracy Score 0.8537360890302067
```

In [11]:
```
probs = model.predict_proba(x_test)
probs = probs[:,1]
y_pred = model.predict(x_test)
fpr, tpr, thresholds = roc_curve(y_test,probs)
```

In [12]:

```
plt.plot(fpr,tpr)
plt.title('ROC Curve')
plt.show()
```



In [13]:

```
auc_score = auc(fpr,tpr)
print(f'AUC Score {auc_score}')
```

```
In
          AUC Score 0.5128026070763501
```

```
In [14]: proba = model.predict_proba(x_test)
```

```


acc_ts = 0

for t in np.linspace(0,1,100):
    y_bin = binarize(proba,threshold=t)
    y_pred_t = y_bin[:,1]

    acc_t = accuracy_score(y_test,y_pred_t)
    if acc_t > acc_ts:
        ts = t
        acc_ts = acc_t

print(ts,acc_ts)
```

```
0.20202020202020204 0.8537360890302067
```

```
[15]: ts = 0
```

```
In [19]: x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0
```

```
In [20]: model = LogisticRegression()
```

```
In [21]: model.fit(x_train,y_train)
```

```
Out[21]: LogisticRegression()
```

```
In [22]: y_pred = model.predict(x_test)
```

```
In [23]: print(f'Accuracy Score : {accuracy_score(y_test,y_pred)}')
```

```
In [16]: y_bin_02 = binarize(proba,threshold=0.2)
```

```
In [17]: y_pred_02 = y_bin_02[:,1]
```

```
In [18]: x = scaled_df.loc[:,scaled_df.columns != 'TenYearCHD']
         y= scaled_df.TenYearCHD
```

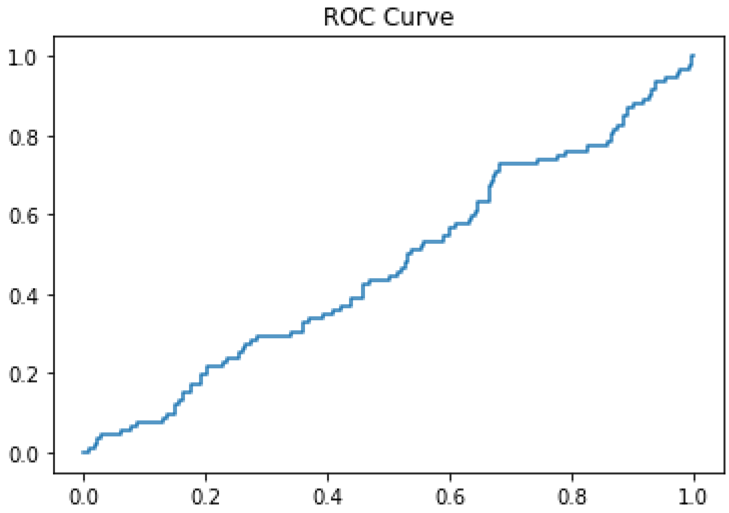
```
Accuracy Score : 0.8537360890302067
```

In [24]:

```
probs = model.predict_proba(x_test)
probs = probs[:,1]
y_pred = model.predict(x_test)
fpr, tpr, thresholds = roc_curve(y_test,probs)
```

In [25]:

```
plt.plot(fpr,tpr)
plt.title('ROC Curve')
plt.show()
```



In [26]:

```
print(f'AUC Score : {auc(fpr,tpr)}')
```

```
AUC Score : 0.4720063152781151
```