Ismet Okan Celik CWID: 10472265

I pledge my honor that I have abided by the Stevens Honor System.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split
```

In [2]:
```
data=pd.read_csv('dataSet_2.csv',header=None)
data
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	...	914	915	916	9
0	1.85050	0.44413	-0.26072	-0.92156	-0.52009	2.73230	-1.5232	-1.2479	1.4870	-0.52449	...	-0.403160	-0.555010	-0.590640	-0.6319
1	2.13930	0.67815	-0.34487	-0.93527	-0.52009	3.08680	-1.5549	-1.2048	1.4753	-0.41132	...	-1.182500	1.527000	0.267590	2.7987
2	2.19670	0.63249	-0.28908	-0.89806	-0.52009	3.08970	-1.5624	-1.1764	1.5184	-0.37138	...	0.258950	-0.537660	1.899500	0.5685
3	2.21590	0.68735	-0.31650	-0.90001	-0.52009	3.15650	-1.5627	-1.0686	1.5353	-0.34309	...	1.780200	2.820300	0.268560	-0.1508
4	2.26450	0.69541	-0.31650	-0.93331	-0.52009	3.09840	-1.5769	-1.1232	1.4557	-0.32478	...	-0.991740	-0.520880	-0.426940	0.5063
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5917	-0.49460	0.25691	1.60670	1.64210	1.92240	0.24546	1.9545	1.6000	1.6307	2.14820	...	-0.132260	-0.004755	-0.801210	0.0420
5918	-0.48429	0.29911	1.63510	1.67340	1.92240	0.26870	1.9817	1.6059	1.6307	2.15980	...	1.373600	-0.004755	0.982660	0.0949
5919	-0.43715	0.28530	1.66250	1.68120	1.92240	0.29775	2.0059	1.6073	1.6424	2.20810	...	-0.898300	-0.004755	-0.693790	1.1471
5920	-0.47545	0.31983	1.60670	1.71850	1.92240	0.26289	2.0425	1.6059	1.6816	2.20140	...	-0.971120	-0.004755	0.982660	-0.9870
5921	-0.46514	0.28339	1.60670	1.73220	1.92240	0.26579	2.0516	1.6146	1.6660	2.23140	...	0.003648	-0.004755	0.000621	-0.0011

5922 rows × 924 columns

In [3]:
```
predict=data.iloc[:,0:923]
response=data[923]
```

In [4]:
```
x_train, x_test, y_train, y_test=train_test_split(predict, response, test_size=0.24,random_state=0,shuffle=False)
print('X_Train--->>(Number of Row, Number of Column)=',np.shape(x_train))
print('Y_Train--->>(Number of Row, Number of Column)=',np.shape(x_test))
print('X_Test--->>(Number of Row, Number of Column)=',np.shape(y_train))
print('Y_Test--->>(Number of Row, Number of Column)=',np.shape(y_test))
```

```
X_Train--->>(Number of Row, Number of Column)= (4500, 923)
Y_Train--->>(Number of Row, Number of Column)= (1422, 923)
X_Test--->>(Number of Row, Number of Column)= (4500,)
Y_Test--->>(Number of Row, Number of Column)= (1422,)
```

In [5]:
```
#Training with Original Matrix

model=LogisticRegression(solver='liblinear',C=1.0,random_state=0)
model_fit=model.fit(x_train,y_train)
```

In [6]:
```
prob_pred_train=model_fit.predict_proba(x_train)[:,1]
prob_pred_train
```

Out[6]:
```
array([0.99997895, 0.99932807, 0.99476401, ..., 0.36858563, 0.20719681,
       0.25312582])
```

In [7]:
```
FP_Rate_Train,TP_Rate_Train,_=metrics.roc_curve(y_train,prob_pred_train)
AUC_Train=metrics.roc_auc_score(y_train,prob_pred_train)
```

In [8]:
```
#Testing with Original Matrix

prob_pred_test=model_fit.predict_proba(x_test)[:,1]
prob_pred_test
```

Out[8]:
```
array([1.64375495e-01, 1.89381346e-03, 7.82597861e-04, ...,
       1.69012276e-06, 2.84864545e-06, 2.49816978e-02])
```

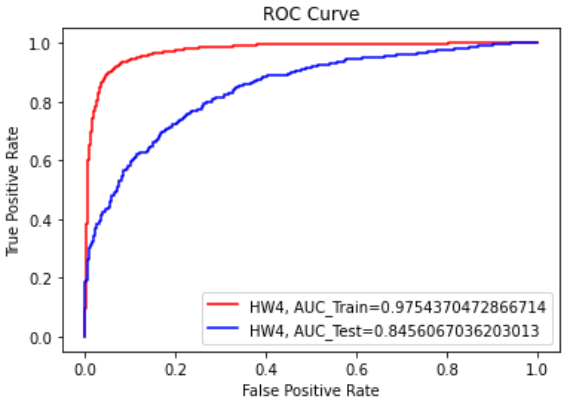
In [9]:
```
FP_Rate_Test,TP_Rate_Test,_=metrics.roc_curve(y_test,prob_pred_test)
AUC_Test=metrics.roc_auc_score(y_test,prob_pred_test)
```

In [10]:
```
# Plotting ROC for Training and Testing with Original Matrix

difference1=AUC_Train-AUC_Test
print('Differance Between AUC_Train and AUC_Test=',difference1)
plt.plot(FP_Rate_Train,TP_Rate_Train,label='HW4, AUC_Train='+str(AUC_Train),color='r')
plt.plot(FP_Rate_Test,TP_Rate_Test,label='HW4, AUC_Test='+str(AUC_Test),color='b')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc=4)
plt.show
```

```
Differance Between AUC_Train and AUC_Test= 0.12983034366637003
```

Out[10]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [19]:
```
# Generating new predictor matrix by using [X(t)- X(t-1)]/X(t-1) and adding new matrix to the original predictor m

new_predictor_train=np.subtract(x_train[2:4500],x_train[1:4499])
new_predictor_train=np.divide(new_predictor_train,x_train[1:4499])
all_predictor_train=pd.concat([x_train[2:4500],new_predictor_train],axis=1,ignore_index=True)

new_response_train=y_train[2:4500]

new_predictor_test=np.subtract(x_test[2:1422],x_test[1:1421])
new_predictor_test=np.divide(new_predictor_test,x_test[1:1421])
all_predictor_test=pd.concat([x_test[2:1422],new_predictor_test],axis=1,ignore_index=True)

new_response_test=y_test[2:4500]

print('Shape of all Predictor Train Matrix')
print('(Number of Row, Number of Column)=',np.shape(all_predictor_train),'\n')

print('Shape of all Predictor Test Matrix')
print('(Number of Row, Number of Column)=',np.shape(all_predictor_test))
```

```
Shape of all Predictor Train Matrix
(Number of Row, Number of Column)= (4498, 1846)

Shape of all Predictor Test Matrix
(Number of Row, Number of Column)= (1420, 1846)

C:\Users\okanc\AppData\Local\Temp/ipykernel_6096/2643719114.py:1: FutureWarning: Calling a ufunc on non-aligned D
ataFrames (or DataFrame/Series combination). Currently, the indices are ignored and the result takes the index/co
lumns of the first DataFrame. In the future , the DataFrames/Series will be aligned before applying the ufunc.
Convert one of the arguments to a NumPy array (eg 'ufunc(df1, np.asarray(df2)') to keep the current behaviour, or
align manually (eg 'df1, df2 = df1.align(df2)') before passing to the ufunc to obtain the future behaviour and si
lence this warning.
  new_predictor_train=np.subtract(x_train[2:4500],x_train[1:4499])
C:\Users\okanc\AppData\Local\Temp/ipykernel_6096/2643719114.py:2: FutureWarning: Calling a ufunc on non-aligned D
ataFrames (or DataFrame/Series combination). Currently, the indices are ignored and the result takes the index/co
lumns of the first DataFrame. In the future , the DataFrames/Series will be aligned before applying the ufunc.
Convert one of the arguments to a NumPy array (eg 'ufunc(df1, np.asarray(df2)') to keep the current behaviour, or
align manually (eg 'df1, df2 = df1.align(df2)') before passing to the ufunc to obtain the future behaviour and si
lence this warning.
  new_predictor_train=np.divide(new_predictor_train,x_train[1:4499])
C:\Users\okanc\AppData\Local\Temp/ipykernel_6096/2643719114.py:7: FutureWarning: Calling a ufunc on non-aligned D
ataFrames (or DataFrame/Series combination). Currently, the indices are ignored and the result takes the index/co
lumns of the first DataFrame. In the future , the DataFrames/Series will be aligned before applying the ufunc.
Convert one of the arguments to a NumPy array (eg 'ufunc(df1, np.asarray(df2)') to keep the current behaviour, or
align manually (eg 'df1, df2 = df1.align(df2)') before passing to the ufunc to obtain the future behaviour and si
lence this warning.
  new_predictor_test=np.subtract(x_test[2:1422],x_test[1:1421])
C:\Users\okanc\AppData\Local\Temp/ipykernel_6096/2643719114.py:8: FutureWarning: Calling a ufunc on non-aligned D
ataFrames (or DataFrame/Series combination). Currently, the indices are ignored and the result takes the index/co
lumns of the first DataFrame. In the future , the DataFrames/Series will be aligned before applying the ufunc.
Convert one of the arguments to a NumPy array (eg 'ufunc(df1, np.asarray(df2)') to keep the current behaviour, or
align manually (eg 'df1, df2 = df1.align(df2)') before passing to the ufunc to obtain the future behaviour and si
lence this warning.
  new_predictor_test=np.divide(new_predictor_test,x_test[1:1421])
```

In [12]:
```
# Training with Original Matrix and New Generated Features

model_new=LogisticRegression(solver='liblinear',C=1.0,random_state=0)
model_fit_new=model.fit(all_predictor_train,new_response_train)
```

```
C:\Users\okanc\anaconda3\lib\site-packages\sklearn\svm\_base.py:985: ConvergenceWarning: Liblinear failed to conv
erge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

In [13]:
```
prob_pred_train_new=model_fit_new.predict_proba(all_predictor_train)[:,1]
prob_pred_train_new
```

Out[13]:
```
array([4.54035439e-01, 4.41674647e-01, 4.73837673e-01, ...,
       4.44390432e-01, 3.33885937e-01, 1.44116681e-10])
```

In [14]:
```
FP_Rate_Train_new,TP_Rate_Train_new,_=metrics.roc_curve(new_response_train,prob_pred_train_new)
AUC_Train_new=metrics.roc_auc_score(new_response_train,prob_pred_train_new)
```

In [15]:
```
# Testing with Original Matrix and New Generated Features

prob_pred_test_new=model_fit_new.predict_proba(all_predictor_test)[:,1]
prob_pred_test_new
```

Out[15]:
```
array([0.33725649, 0.57306177, 0.48530598, ..., 0.32160023, 0.39569653,
       0.88338005])
```

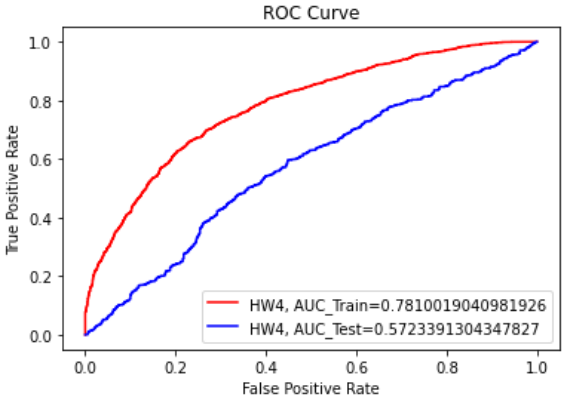
In [16]:
```
FP_Rate_Test_new,TP_Rate_Test_new,_=metrics.roc_curve(new_response_test,prob_pred_test_new)
AUC_Test_new=metrics.roc_auc_score(new_response_test,prob_pred_test_new)
```

In [17]:
```
# Plotting ROC - Training and Testing for Original Matrix and New Generated Features

differance_new=AUC_Train_new-AUC_Test_new
print('Difference Between New AUC_Train and AUC_Test=',differance_new)
plt.plot(FP_Rate_Train_new,TP_Rate_Train_new,label='HW4, AUC_Train='+str(AUC_Train_new),color='r')
plt.plot(FP_Rate_Test_new,TP_Rate_Test_new,label='HW4, AUC_Test='+str(AUC_Test_new),color='b')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc=4)
plt.show
```

```
Difference Between New AUC_Train and AUC_Test= 0.2086627736634099
```

Out[17]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



We can see how the new generated matrix new features affected the AUC_Train and AUC_Test. The difference between them got bigger and the AUC_Train value decreased drastically

In [18]:
```
# Feature Selection by Usin Recursive Feature Elimination (RFE)

rfe_selector=RFE(estimator=LogisticRegression(solver='liblinear',C=1.0,random_state=0),n_features_to_select=10,ste
rfe_selector=rfe_selector.fit(all_predictor_train,new_response_train);
rfe_support=rfe_selector.get_support();
rfe_feature=all_predictor_train.loc[:,rfe_support].columns.tolist();
```

```
Fitting estimator with 101 features.
Fitting estimator with 96 features.
Fitting estimator with 91 features.
Fitting estimator with 86 features.
Fitting estimator with 81 features.
Fitting estimator with 76 features.
Fitting estimator with 71 features.
Fitting estimator with 66 features.
Fitting estimator with 61 features.
Fitting estimator with 56 features.
Fitting estimator with 51 features.
Fitting estimator with 46 features.
Fitting estimator with 41 features.
Fitting estimator with 36 features.
Fitting estimator with 31 features.
Fitting estimator with 26 features.
Fitting estimator with 21 features.
Fitting estimator with 16 features.
Fitting estimator with 11 features.
```

In [20]:
```
print('Selected Features=',rfe_feature)
```

```
Selected Features= [18, 25, 63, 71, 78, 80, 279, 347, 348, 824]
```

There is no feature among the selected features from the newly generated matrix. As we can see above the last column number is 824, and newly generated matrix starts after column number 923. Also, I have tried differend stepsize for fast computation and if I apply step size as 1 it takes too long to compute and at the and the function selects the same features. That's why I set step size as 5 for fast computation

In [21]:
```
selected_features_train=all_predictor_train[rfe_feature]

selected_features_test=all_predictor_test[rfe_feature]

print('Shape of New Training Matrix=',np.shape(selected_features_train))
print('Shape of New Testing Matrix=',np.shape(selected_features_test))
```

```
Shape of New Training Matrix= (4498, 10)
Shape of New Testing Matrix= (1420, 10)
```

In [22]:
```
# Training With Selected Features

model2=LogisticRegression(solver='liblinear',C=1.0,random_state=0)
model_fit2=model2.fit(selected_features_train,new_response_train)
```

In [23]:
```
prob_pred_train2=model_fit2.predict_proba(selected_features_train)[:,1]
prob_pred_train2
```

Out[23]:
```
array([0.78242616, 0.78960513, 0.8955788 , ..., 0.85858887, 0.58608702,
       0.47043029])
```

In [24]:
```
FP_Rate_Train_new1,TP_Rate_Train_new1,_=metrics.roc_curve(new_response_train,prob_pred_train2)
AUC_Train_new1=metrics.roc_auc_score(new_response_train,prob_pred_train2)
```

In [25]:
```
# Testing With Selected Features

prob_pred_test2=model_fit2.predict_proba(selected_features_test)[:,1]
prob_pred_test2
```

Out[25]:
```
array([5.01386235e-02, 2.22654306e-01, 3.76914404e-01, ...,
       2.15651614e-03, 7.54085210e-05, 4.46022862e-04])
```

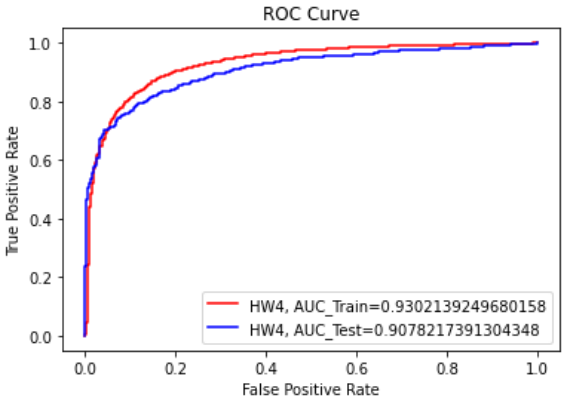
In [26]:
```
FP_Rate_Test_new1,TP_Rate_Test_new1,_=metrics.roc_curve(new_response_test,prob_pred_test2)
AUC_Test_new1=metrics.roc_auc_score(new_response_test,prob_pred_test2)
```

In [27]:
```
# Plotting ROC for Selected Features

difference2=AUC_Train_new1-AUC_Test_new1
print('Differance Between New AUC_Train and AUC_Test=',difference2)
plt.plot(FP_Rate_Train_new1,TP_Rate_Train_new1,label='HW4, AUC_Train='+str(AUC_Train_new1),color='r')
plt.plot(FP_Rate_Test_new1,TP_Rate_Test_new1,label='HW4, AUC_Test='+str(AUC_Test_new1),color='b')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc=4)
plt.show
```

```
Differance Between New AUC_Train and AUC_Test= 0.022392185837580958
```

Out[27]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



As we can see on the graph, the difference between AUC_Train and AUC_Test is 0.022392185837580958; it is really small after feature selection.

The new predictor matrix hasn't affected the feature selection because the feature selection function chooses the features from the original matrix. However, we used ten columns to run Logistic Regression compared with 923 columns, and we got better results in terms of testing, and our AUC_Train value is so close to the AUC_Train value compared with the graph before the feature selection.

I also tried selecting 5,10,15, and 20 features, and I observed that all the selected columns index numbers are smaller than 824. That means all the features selected from the original matrix. When I increased the selected feature numbers from 10 to the 20 ROC and AUC values didn't change much, the difference between AUC_Train and AUC_Test was 0.02677145617815324. That is a little bigger than the difference above.

I had tried different feature selection methods from Sklearn Library such as SelectFromModel, SelectKBest and in that experiment, I saw when the features were selected from the newly generated matrix. AUC values were around 0.50, and performance were bad.

In [ ]:
```
```