Jacob Blumsack

COMP 4220 Machine Learning Final: Classification Set

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:

```
#Load dataset
creditCard = pd.read_csv("creditcard.csv")

#Set X and Y
X = creditCard.drop('Class', axis = 1)
y = creditCard['Class']

#display set
creditCard.head()
```

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0

5 rows × 31 columns

In [4]:

```
#Now we want to split the data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.25, random_state = 0)
```

In [5]:

```
#Now for some preprocessing
from sklearn.preprocessing import StandardScaler
standard_scaler = StandardScaler()
X_train = standard_scaler.fit_transform(X_train)
X_test = standard_scaler.transform(X_test)
```

In [6]:

```
#First use logistic regression
from sklearn.linear_model import LogisticRegression
LogReg = LogisticRegression(random_state = 0)
LogReg.fit(X_train, y_train)
```

Out[6]:

```
LogisticRegression(random_state=0)
```

In [7]:

```
#predict
y_pred = LogReg.predict(X_test)
```

In [8]:

```
#show confusion matrix
from sklearn.metrics import confusion_matrix
ConfusionMatrix = confusion_matrix(y_test, y_pred)
print(ConfusionMatrix)
```

```
[[71071    11]
 [   41    79]]
```

In [9]:

```
#calculate the metrics
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score

AccuracyScore = accuracy_score(y_test, y_pred)
PrecisionScore = precision_score(y_test, y_pred)
RecallScore = recall_score(y_test, y_pred)
```

```
F1Score = f1_score(y_test, y_pred)

print ("Accuracy Score is", AccuracyScore)
print ("Precision Score is", PrecisionScore)
print ("Recall Score is", RecallScore)
print ("F1 Score is", F1Score)
```

```
Accuracy Score is 0.9992696834358585
Precision Score is 0.8777777777777778
Recall Score is 0.6583333333333333
F1 Score is 0.7523809523809525
```

In [10]:

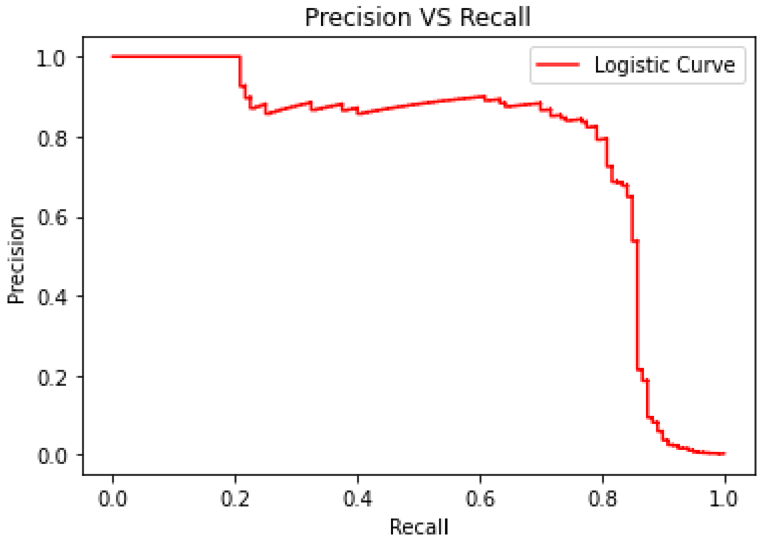
```
#Now for the precision recall curve
from sklearn.metrics import precision_recall_curve, average_precision_s
core
pred_prob = LogReg.predict_proba(X_test)
y_score = pred_prob[:,1]
average_precision = average_precision_score(y_test, y_score)
precision, recall, thresholds = precision_recall_curve(y_test, y_score)
```

In [20]:

```
#plot precision vs recall
plt.step(recall, precision, color = 'red', label = 'Logistic Curve')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = 'upper right')
plt.title('Precision VS Recall')
```

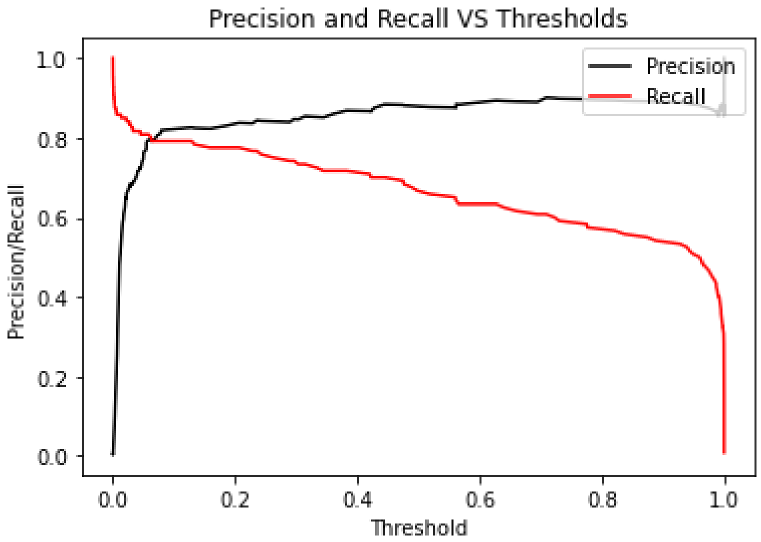
Out[20]:

```
Text(0.5, 1.0, 'Precision VS Recall')
```



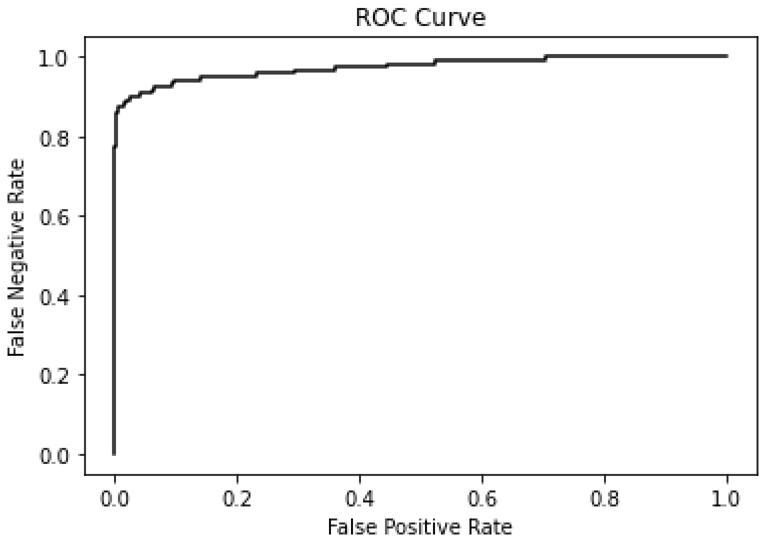
In [12]:

```
#Now to plot the precision/recall vs threshold
plt.plot(thresholds, precision[:-1], color = 'black', label = 'Precision')
plt.plot(thresholds, recall[:-1], color = 'red', label = 'Recall')
plt.xlabel('Threshold')
plt.ylabel('Precision/Recall')
plt.legend(loc = 'upper right')
plt.title("Precision and Recall VS Thresholds")
plt.show()
```



In [13]:

```
#compute the ROC curve for TPR and FPR purposes
from sklearn import metrics
FPR, TPR, threshold = metrics.roc_curve(y_test, y_score)
plt.plot(FPR, TPR, color = 'black')
plt.xlabel("False Positive Rate")
plt.ylabel("False Negative Rate")
plt.title("ROC Curve")
plt.show()
```



In [14]:
```
#computing the AUC score
AUCScore = metrics.auc(FPR, TPR)
print (AUCScore)
```

```
0.9738448786847116
```

In [15]:
```
#now lets try random forest, first train the random forest
from sklearn.ensemble import RandomForestClassifier
Random_Forest = RandomForestClassifier(criterion = 'entropy', random_st
ate = 0)
Random_Forest.fit(X_train, y_train)
```

Out[15]:
```
RandomForestClassifier(criterion='entropy', random_state=0)
```

In [16]:
```
#now set up the prediction
y_pred = Random_Forest.predict(X_test)
```

In [17]:
```
#now lets make a confusion matrix
Confusion_Matrix = confusion_matrix(y_test, y_pred)
print(Confusion_Matrix)
```

```
[[71076     6]
 [   26    94]]
```

In [18]:

```
#calculate the metrics
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score

AccuracyScore = accuracy_score(y_test, y_pred)
PrecisionScore = precision_score(y_test, y_pred)
RecallScore = recall_score(y_test, y_pred)
F1Score = f1_score(y_test, y_pred)

print ("Accuracy Score is", AccuracyScore)
print ("Precision Score is", PrecisionScore)
print ("Recall Score is", RecallScore)
print ("F1 Score is", F1Score)
```

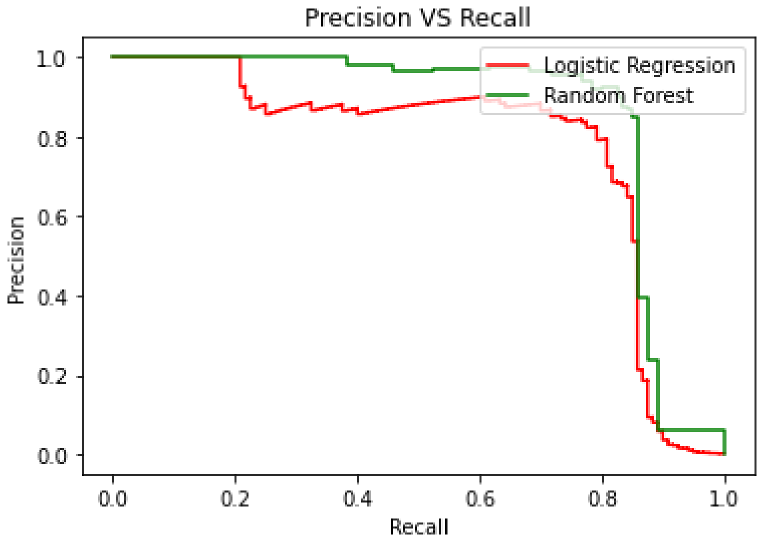
```
Accuracy Score is 0.9995505744220669
Precision Score is 0.94
Recall Score is 0.7833333333333333
F1 Score is 0.8545454545454546
```

In [22]:

```
#now lets compare the precision and recall to that of logistic regressi
on
pred_prob = Random_Forest.predict_proba(X_test)
Random_Forest_score = pred_prob[:,1]
Random_Forest_precision = average_precision_score(y_test, Random_Forest
_score)
Random_Forest_precision, Random_Forest_recall, Random_Forest_thresholds
= precision_recall_curve(y_test, Random_Forest_score)
plt.step(recall, precision, color = 'red', label = 'Logistic Regressio
n')
plt.step(Random_Forest_recall, Random_Forest_precision, color = 'green'
, label = 'Random Forest')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = 'upper right')
plt.title('Precision VS Recall')
```

Out[22]:

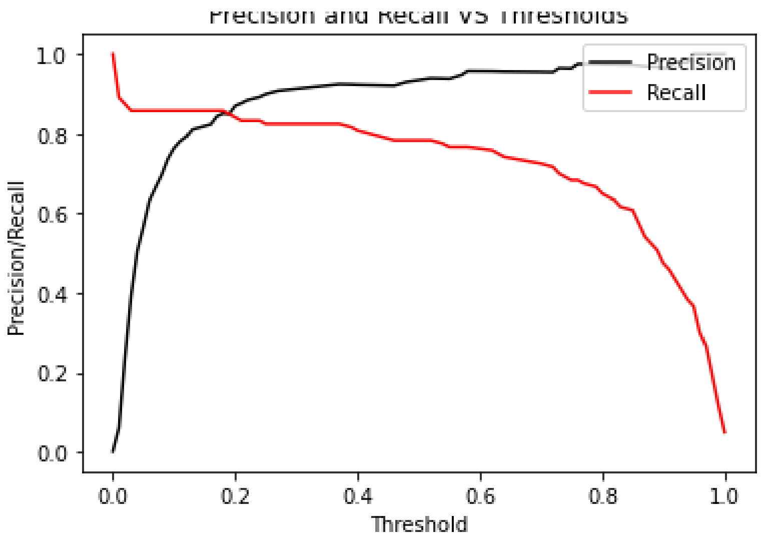
```
Text(0.5, 1.0, 'Precision VS Recall')
```



In [23]:

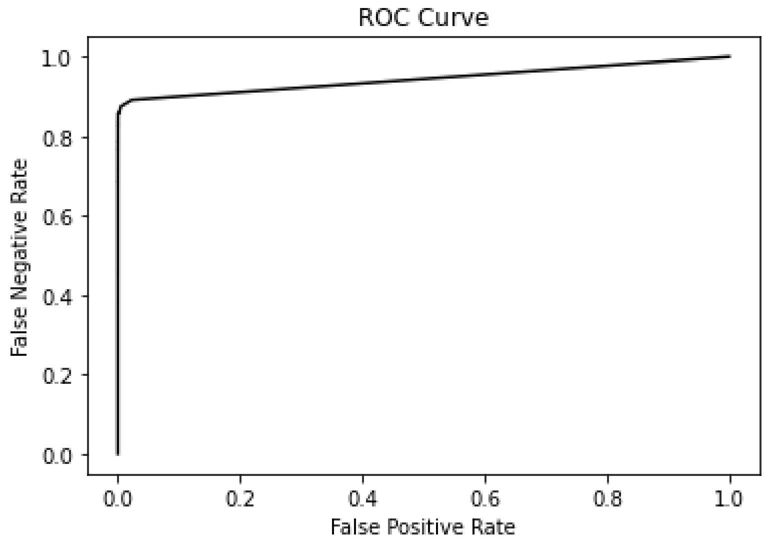
```
#Now to plot the precision/recall vs threshold
plt.plot(Random_Forest_thresholds, Random_Forest_precision[:-1], color = 'black', label = 'Precision')
plt.plot(Random_Forest_thresholds, Random_Forest_recall[:-1], color = 'red', label = 'Recall')
plt.xlabel('Threshold')
plt.ylabel('Precision/Recall')
plt.legend(loc = 'upper right')
plt.title("Precision and Recall VS Thresholds")
plt.show()
```

Precision and Recall VS Thresholds



In [25]:

```
#compute the ROC curve for TPR and FPR purposes
Random_Forest_FPR, Random_Forest_TPR, Random_Forest_threshold = metrics
.roc_curve(y_test, Random_Forest_score)
plt.plot(Random_Forest_FPR, Random_Forest_TPR, color = 'black')
plt.xlabel("False Positive Rate")
plt.ylabel("False Negative Rate")
plt.title("ROC Curve")
plt.show()
```



In [26]:

```
#computing the AUC score
AUCScore = metrics.auc(Random_Forest_FPR, Random_Forest_TPR)
print (AUCScore)
```

```
0.9442650155219793
```

In [27]:

```
#now lets try the neural network, first get the needed libraries
import tensorflow as tf
from sklearn.compose import ColumnTransformer
import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

In [40]:

```
#the data has already been split and preprocessed with standard scaler
#Time to make the ANN
classifier = Sequential()
```

In [41]:

```
#add the imput layer and first hidden layer
classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activat
ion = 'relu', input_dim = 30))
```

In [42]:

```
#now add the second hidden layer
classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activat
ion = 'relu'))
```

In [43]:

```
#finally add the output layer
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activat
ion = 'sigmoid'))
```

In [44]:

```
#time to compile it
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', me
trics = ['accuracy'])
```

In [45]:

```
#now the data comes in, the ANN is fit to the training set
classifier.fit(X_train, y_train)
```

```
6676/6676 [==============================] - 5s 601us/step - loss: 0.07
83 - accuracy: 0.9979
```

Out[45]:

```
<tensorflow.python.keras.callbacks.History at 0x1788fd1ddc0>
```

In [53]:

```
#now predict the test set results
yPred = classifier.predict(X_test)
yPred = (yPred > 0.1)
```

In [54]:

```
#construct the confusion matrix
confusionMatrix = confusion_matrix(y_test, yPred)
print(confusionMatrix)
```

```
[[71047    35]
 [   20   100]]
```

In [55]:

```
#calculate the metrics
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
```

```
AccuracyScore = accuracy_score(y_test, yPred)
PrecisionScore = precision_score(y_test, yPred)
RecallScore = recall_score(y_test, yPred)
F1Score = f1_score(y_test, yPred)

print ("Accuracy Score is", AccuracyScore)
print ("Precision Score is", PrecisionScore)
print ("Recall Score is", RecallScore)
print ("F1 Score is", F1Score)
```

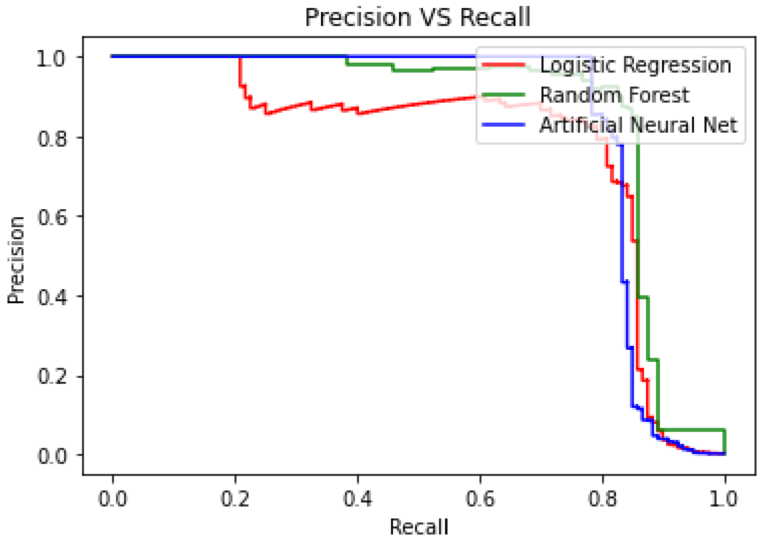
```
Accuracy Score is 0.9992275497879273
Precision Score is 0.7407407407407407
Recall Score is 0.8333333333333334
F1 Score is 0.7843137254901961
```

In [59]:

```
#now lets compare the precision and recall to that of logistic regressi
on
pred_prob = classifier.predict_proba(X_test)
classifier_score = pred_prob
classifier_precision = average_precision_score(y_test, classifier_score
)
classifier_precision, classifier_recall, classifier_thresholds = precis
ion_recall_curve(y_test, classifier_score)
plt.step(recall, precision, color = 'red', label = 'Logistic Regressio
n')
plt.step(Random_Forest_recall, Random_Forest_precision, color = 'green'
, label = 'Random Forest')
plt.step(classifier_recall, classifier_precision, color = 'blue', label
= 'Artificial Neural Net')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.legend(loc = 'upper right')
plt.title('Precision VS Recall')
```

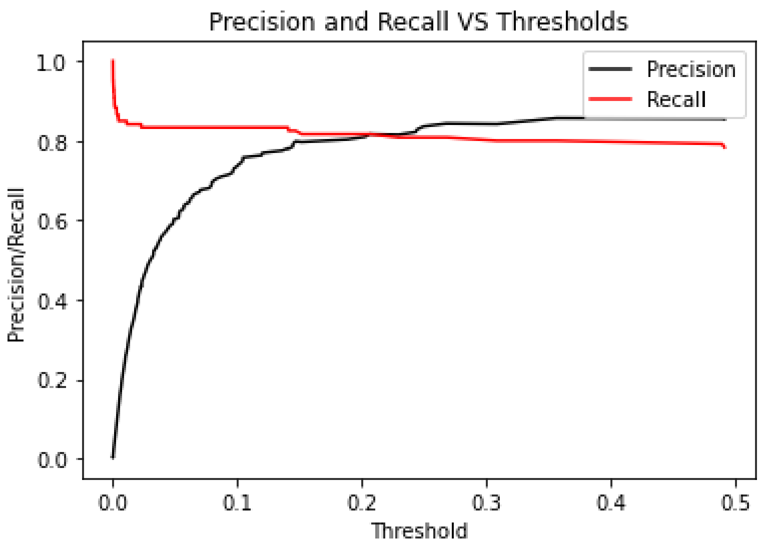
Out[59]:

```
Text(0.5, 1.0, 'Precision VS Recall')
```



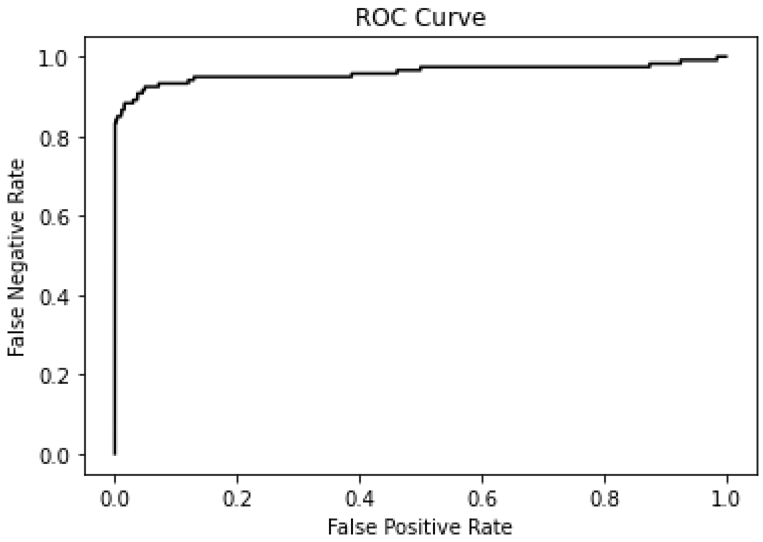
In [60]:

```
#Now to plot the precision/recall vs threshold
plt.plot(classifier_thresholds, classifier_precision[:-1], color = 'black', label = 'Precision')
plt.plot(classifier_thresholds, classifier_recall[:-1], color = 'red', label = 'Recall')
plt.xlabel('Threshold')
plt.ylabel('Precision/Recall')
plt.legend(loc = 'upper right')
plt.title("Precision and Recall VS Thresholds")
plt.show()
```



In [61]:

```
#compute the ROC curve for TPR and FPR purposes
classifier_FPR, classifier_TPR, classifier_threshold = metrics.roc_curv
e(y_test, classifier_score)
plt.plot(classifier_FPR, classifier_TPR, color = 'black')
plt.xlabel("False Positive Rate")
plt.ylabel("False Negative Rate")
plt.title("ROC Curve")
plt.show()
```



In [62]:

```
#computing the AUC score
AUCScore = metrics.auc(classifier_FPR, classifier_TPR)
print (AUCScore)
```

```
0.9606009022443562
```

In [ ]: