Math156 Hw2

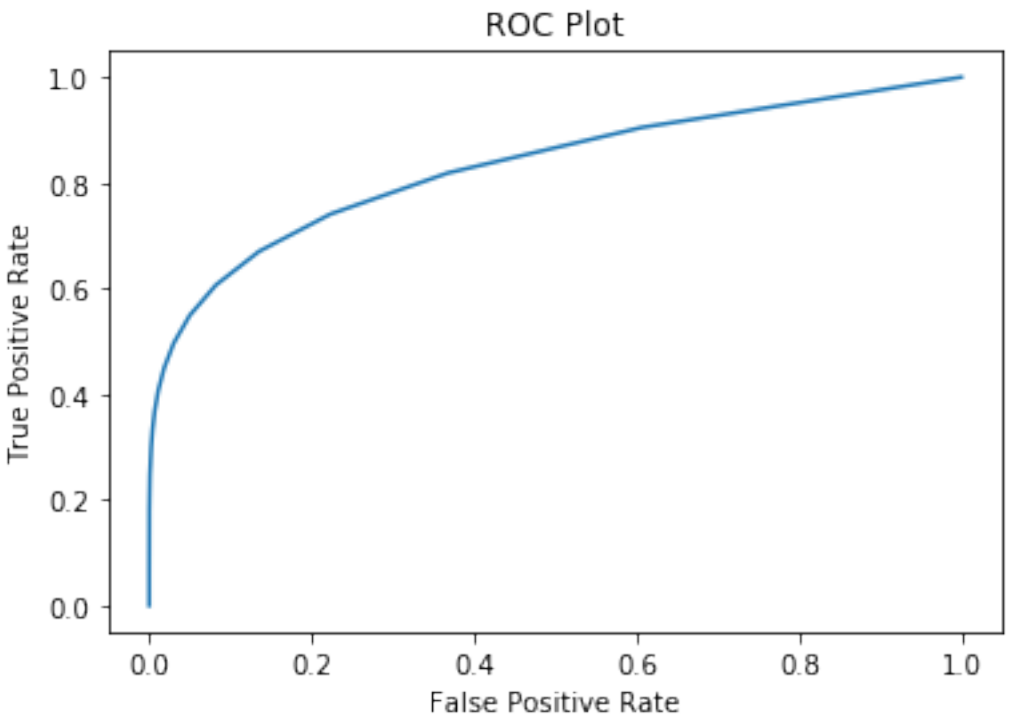
May 5, 2019

In [1]:
```
# Math 156 HW2
## Sijia Hua
## May 6th, 2019

# imports
import numpy as np
import matplotlib.pyplot as plt
import math
import tensorflow as tf
from sklearn import datasets, linear_model
from sklearn.linear_model import LinearRegression
import time
#from sklearn import preprocessing
```

In [3]:
```
## 2(b) plot PD against PFA
t=[t*0.1 for t in range(500)]   #setup possible t values
P_D =[]  #True positive
P_FA =[] # False negative
for i in range (500):
    P_D.append(math.exp(-5*t[i]))
    P_FA.append(math.exp(-1*t[i]))

plt.plot(P_D,P_FA,'-')
plt.title("ROC Plot")
plt.ylabel("True Positive Rate")
plt.xlabel("False Positive Rate")
plt.show()
```



In [5]:
```
##3
# import mnist data
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
s1 = 5000  # size of training set
s2 = 1500  # size of testing set

xtrain1 = x_train[5000:s1+5000]
ytrain1= y_train[5000:s1+5000]
xtest1 = x_test[5000:s2+5000]
ytest1 = y_test[5000:s2+5000]

##### 3.(a)###### Linear Regression

# construct one hot encoding matrix
oh_ytrain1 = np.zeros((s1,10))
oh_ytest1 = np.zeros((s2,10))
for i in range(s1):
    oh_ytrain1[i][ytrain1[i]] = 1
for i in range(s2) :
    oh_ytest1[i][ytest1[i]] = 1

# Construct training matrix
x_trainm = np.zeros((s1,784))  # matrix for training set
x_testm = np.zeros((s2,784))   # matrix for testing set
```

```
for i in range (s1):
    x_trainm[i]=xtrain1[i,:].flatten()
for i in range(s2):
    x_testm[i]= xtest1[i,:].flatten()

# Starting time of model 1
st1 = time.time()
# Create linear regression object
lm = LinearRegression()
# Train the model using the training set
lm.fit(x_trainm,oh_ytrain1)
pred_test = lm.predict(x_testm)
pred_train = lm.predict(x_trainm)

# Print running time of model 1
print"Running time of Linear Regression is : "
print("--- %s seconds ---" % (time.time() - st1))

#return the one hot encoding
ytrain_result = np.zeros((s1,1))
ytest_result = np.zeros((s2,1))
for i in range(s1):
    ytrain_result[i] = np.argmax(pred_train[i,:])
for i in range(s2):
    ytest_result[i] = np.argmax(pred_test[i,:])

# Train and Test Error
print"Training Error of model1 is : ", np.mean((ytrain_result - ytrain1)**2)
print"Testing Error of model1 is : ", np.mean((ytest_result - ytest1)**2)

def accuracy(pred,real,size):
    sum=0
    for i in range(size):
        if pred[i] == real[i]:
            sum = sum +1
    return sum*1.0/(size)

print"Training Accuracy of model1 is: ", accuracy(ytrain_result,ytrain1,s1)
print"Testing Accuracy of model1 is: ", accuracy(ytest_result,ytest1,s2)


##### 3.(b) ###### Linear Discriminant Analysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Starting time of model 2
st2 = time.time()
# name the model
lda = LinearDiscriminantAnalysis()
```

```
# fit the model to the training set
m2 = lda.fit(x_trainm,ytrain1.ravel())
# Predict values
plda_train = m2.predict(x_trainm)
plda_test = m2.predict(x_testm)

# Print running time of model 2
print"\nRunning time of Linear Discriminant Analysis is : "
print("--- %s seconds ---" % (time.time() - st2))

# test mse
print"Training Error of model1 is : ", np.mean((plda_train - ytrain1)**2)
print"Testing Error of model1 is : ", np.mean((plda_test - ytest1)**2)

# test accuracy
print"Training Accuracy of model1 is: ", accuracy(plda_train,ytrain1,s1)
print"Testing Accuracy of model1 is: ", accuracy(plda_test,ytest1,s2)


##### 3.(c) #####  Logistic Regression

from sklearn.linear_model import LogisticRegression

# Starting time of model 3
st3 = time.time()
# name the model
lr = LogisticRegression()
# fit the model to the training set
m3 = lr.fit(x_trainm,ytrain1.ravel())
# predict values
plr_train = m3.predict(x_trainm)
plr_test = m3.predict(x_testm)

# Print running time of model 3
print"\nRunning time of Logistic Regression is : "
print("--- %s seconds ---" % (time.time() - st3))

# test mse
print"Training Error of model1 is : ", np.mean((plr_train - ytrain1)**2)
print"Testing Error of model1 is : ", np.mean((plr_test - ytest1)**2)

# test accuracy
print"Training Accuracy of model1 is: ", accuracy(plr_train,ytrain1,s1)
print"Testing Accuracy of model1 is: ", accuracy(plr_test,ytest1,s2)



##### 3.(d) ##### Random Forest
```

```
from sklearn.ensemble import RandomForestClassifier

# Starting time of model 4
st4 = time.time()
# name the model
rf = RandomForestClassifier(n_estimators=500,random_state=0)
# fit the model to the training set
m4 = rf.fit(x_trainm,ytrain1)
# prediction values
prf_train = m4.predict(x_trainm)
prf_test = m4.predict(x_testm)

# Print running time of model 4
print"\nRunning time of Random Forest is : "
print("--- %s seconds ---" % (time.time() - st4))

# test mse
print"Training Error of model1 is : ", np.mean((prf_train - ytrain1)**2)
print"Testing Error of model1 is : ", np.mean((prf_test - ytest1)**2)

# test accuracy
print"Training Accuracy of model1 is: ", accuracy(prf_train,ytrain1,s1)
print"Testing Accuracy of model1 is: ", accuracy(prf_test,ytest1,s2)

##### 3.(e) ##### SVM
from sklearn.svm import SVC

## Linear SVM
# Starting time of model 5
st5=time.time()
# name the model
lsvm = SVC(kernel ='linear')
# fit the model to the training set
m5 = lsvm.fit(x_trainm,ytrain1)
# preidction values
plsvm_train = m5.predict(x_trainm)
plsvm_test = m5.predict(x_testm)

# Print running time of model 5
print"\nRunning time of Linear SVM is : "
print("--- %s seconds ---" % (time.time() - st5))

# test mse
print"Training Error of model1 is : ", np.mean((plsvm_train - ytrain1)**2)
print"Testing Error of model1 is : ", np.mean((plsvm_test - ytest1)**2)

# test accuracy
print"Training Accuracy of model1 is: ", accuracy(plsvm_train,ytrain1,s1)
```

```
        print"Testing Accuracy of model1 is: ", accuracy(plsvm_test,ytest1,s2)

        ## Gaussian SVM
        # Starting time of model 6
        st6=time.time()
        # name the model
        gsvm = SVC(C=1, kernel='rbf', gamma = 5.5*(1e-7))
        # fit the model to the training set
        m6 = gsvm.fit(x_trainm,ytrain1)
        # preidction values
        pgsvm_train = m6.predict(x_trainm)
        pgsvm_test = m6.predict(x_testm)

        # Print running time of model 6
        print"\nRunning time of Gaussian SVM is : "
        print("--- %s seconds ---" % (time.time() - st6))

        # test mse
        print"Training Error of model1 is : ", np.mean((pgsvm_train - ytrain1)**2)
        print"Testing Error of model1 is : ", np.mean((pgsvm_test - ytest1)**2)

        # test accuracy
        print"Training Accuracy of model1 is: ", accuracy(pgsvm_train,ytrain1,s1)
        print"Testing Accuracy of model1 is: ", accuracy(pgsvm_test,ytest1,s2)
```

```
Running time of Linear Regression is :
--- 0.370328903198 seconds ---
Training Error of model1 is :  17.02438696
Testing Error of model1 is :  17.080010666666666
Training Accuracy of model1 is:  0.897
Testing Accuracy of model1 is:  0.862

Running time of Linear Discriminant Analysis is :
--- 1.28804206848 seconds ---
Training Error of model1 is :  1.582
Testing Error of model1 is :  1.766
Training Accuracy of model1 is:  0.9148
Testing Accuracy of model1 is:  0.872666666667

Running time of Logistic Regression is :
--- 45.3065629005 seconds ---
Training Error of model1 is :  0.0058
Testing Error of model1 is :  2.242
Training Accuracy of model1 is:  0.9996
Testing Accuracy of model1 is:  0.872666666667

Running time of Random Forest is :
--- 12.5346992016 seconds ---
```

```
Training Error of model1 is :  0.0
Testing Error of model1 is :  0.72
Training Accuracy of model1 is:  1.0
Testing Accuracy of model1 is:  0.954

Running time of Linear SVM is :
--- 12.8206408024 seconds ---
Training Error of model1 is :  0.0
Testing Error of model1 is :  1.0786666666666667
Training Accuracy of model1 is:  1.0
Testing Accuracy of model1 is:  0.932666666667

Running time of Gaussian SVM is :
--- 33.4751429558 seconds ---
Training Error of model1 is :  0.0328
Testing Error of model1 is :  0.6053333333333333
Training Accuracy of model1 is:  0.999
Testing Accuracy of model1 is:  0.971333333333
```

In [11]:
```
## compare and conclude
# confusion matrix
from sklearn.metrics import confusion_matrix
# find maximum error rate
def error(ytest_result,ytest1,s2):
    m=np.zeros((10,10))
    r=np.zeros((10,10))
    for i in range(s2):
        m[int(ytest_result[i]),int(ytest1[i])] = m[int(ytest_result[i]),int(ytest1[i])
    for i in range(10):
        for j in range(10):
            r[i,j] = m[i,j]/np.sum(m,axis=0)[j]
    for i in range(10): r[i,i]=0
    for i in range(10):
        for j in range(10):
           if r[i,j] == np.max(r):
               print "Predict "+str(j)+" as "+str(i)+" with probability: "+str((r[i][j


# for Linear Regression
print("The most probably error for Linear Regression to make is : ")
error(ytest_result,ytest1,s2)

# for LDA
print("\nThe most probably error for LDA to make is : ")
error(plda_test,ytest1,s2)

# for Logistic Regression
```

```
print("\nThe most probably error for Logistic Regression to make is : ")
error(plr_test,ytest1,s2)


# for Random Forest
print("\nThe most probably error for Random Forest to make is : ")
error(prf_test,ytest1,s2)


# for Linear SVM
print("\nThe most probably error for Linear SVM to make is : ")
error(plsvm_test,ytest1,s2)

# for SVM with Gaussian radial basis kernel
print("\nThe most probably error for Gaussian SVM to make is : ")
error(pgsvm_test,ytest1,s2)
```

```
The most probably error for Linear Regression to make is :
Predict 5 as 3 with probability: 0.17647058823529413

The most probably error for LDA to make is :
Predict 5 as 3 with probability: 0.1323529411764706

The most probably error for Logistic Regression to make is :
Predict 3 as 8 with probability: 0.11409395973154363

The most probably error for Random Forest to make is :
Predict 5 as 3 with probability: 0.09558823529411764

The most probably error for Linear SVM to make is :
Predict 5 as 3 with probability: 0.08088235294117647

The most probably error for Gaussian SVM to make is :
Predict 5 as 3 with probability: 0.04411764705882353
```

Summary:

Linear Regression has the lowest running time but with the highest error rate and highest MSE.LDA is also fast and the accuracy has improved around 1% compared to linear regression. Logistic Regression takes thelongest time and larger MSE, same accuracy compared with LDA. Random Forest, Linear SVM, Gaussian SVM all takes 10 more seconds and a accuracy of 90%+. Even though Gaussian SVM has accuracy 97%, it takes three times as long as linear svm and random forest. In these cases, I prefer to use Random Forest.

For most Prediction methods, they have confusion about 5 and 3 as the code results shown above. Hence in the future, we can add more conditions or parameters to take care of this case, especially how does the language read number 3.

In [ ]: