PATTERN RECOGNITION ASSIGNMENT 1

CO-324

2021 Semester VI

Submitted to: Sh. Anurag Goel

Submitted By: Saksham Gera (2K18/EE/179) & Shivam Shaurya(2K18/EE/194)

Answer 1

Required Libraries:

Numpy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

Documentation Link: https://numpy.org/doc/stable/contents.html (https://numpy.org/doc/stable/contents.html)

Pandas

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Documentation Link: https://pandas.pydata.org/docs/ (https://pandas.pydata.org/docs/)

Sci-Kit Learn

Scikit-learn is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.

Documentation Link: https://scikit-learn.org/stable/ (https://scikit-learn.org/stable/)

Tensorflow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

Documentation Link: https://www.tensorflow.org/ (https://www.tensorflow.org/)

ANSWER 1: MNIST DIGIT CLASSIFICATION

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from tensorflow.keras.datasets import mnist
from sklearn import metrics
```

For Classification Between 0 and 1

In [5]:
```
(x_train1,y_train1),(x_test1,y_test1)=mnist.load_data()
x_train1=x_train1.reshape(60000,784)
x_test1=x_test1.reshape(10000,784)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz)
11493376/11490434 [==============================] - 0s 0us/step
```

In [6]:
```
x_train=[]
y_train=[]
for i in range(60000):
  if y_train1[i]==0 or y_train1[i]==1:
    temp=[]
    for j in range(784):
      temp.append(x_train1[i][j])
    x_train.append(temp)
    y_train.append(y_train1[i])
```

In [7]:
```
print(len(x_train[0]))
```

```
784
```

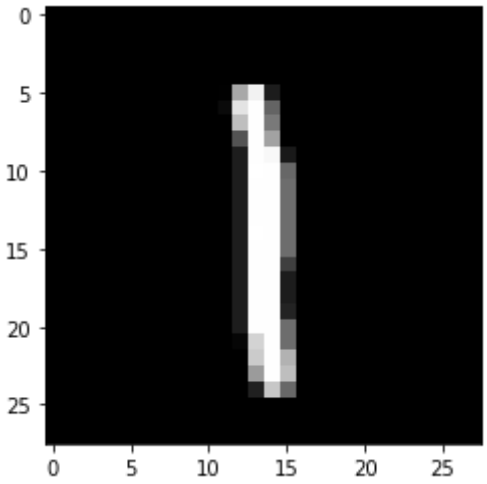
In [8]:
```
x_test=[]
y_test=[]
for i in range(10000):
  if y_test1[i]==0 or y_test1[i]==1:
    temp=[]
    for j in range(784):
      temp.append(x_test1[i][j])
    x_test.append(temp)
    y_test.append(y_test1[i])
```

In [9]:
```
print(len(x_test))
```

```
2115
```

In [10]:
```
x_train=np.matrix(x_train)
y_train=np.array(y_train)
x_test=np.matrix(x_test)
y_test=np.array(y_test)
# x_train.reshape((12665,784))
```

In [11]:
```
plt.imshow(x_train[4].reshape((28,28)),cmap='gray')
plt.show()
```



In [12]:
```
nb_model=GaussianNB()
```

In [13]:
```
y_train
```

Out[13]:
```
array([0, 1, 1, ..., 1, 0, 1], dtype=uint8)
```

In [14]:
```
fit_nb=nb_model.fit(x_train,y_train)
```

In [15]:
```
prediction=fit_nb.predict(x_test)
con_matrix=confusion_matrix(y_test,prediction)
print(con_matrix)
```

```
[[ 976    4]
 [  22 1113]]
```

In [16]:
```
def diagonal_sum(con_matrix):
  sum=0
  for i in range(2):
    for j in range(2):
      if i==j:
        sum+=con_matrix[i][j]
  return sum
```

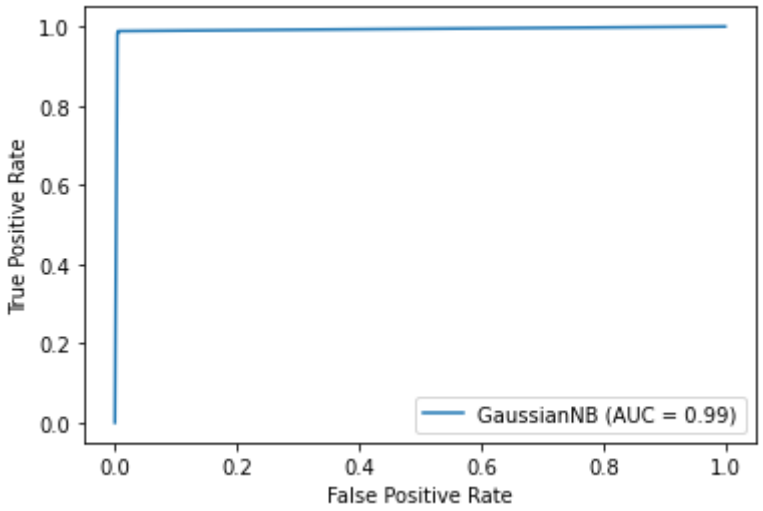
In [17]:
```
sum=diagonal_sum(con_matrix)
print(sum)
```

```
2089
```

In [18]:
```
print(f'Accuracy % : {sum/2115}')
```

```
Accuracy % : 0.9877068557919622
```

In [19]:
```
metrics.plot_roc_curve(fit_nb, x_test, y_test)
plt.show()
```



For Classification Between 3 and 8

In [20]:
```
(x_train1,y_train1),(x_test1,y_test1)=mnist.load_data()
x_train1=x_train1.reshape(60000,784)
x_test1=x_test1.reshape(10000,784)
```

In [21]:
```
x_train=[]
y_train=[]
for i in range(60000):
  if y_train1[i]==3 or y_train1[i]==8:
    temp=[]
    for j in range(784):
      temp.append(x_train1[i][j])
    x_train.append(temp)
    y_train.append(y_train1[i])
```

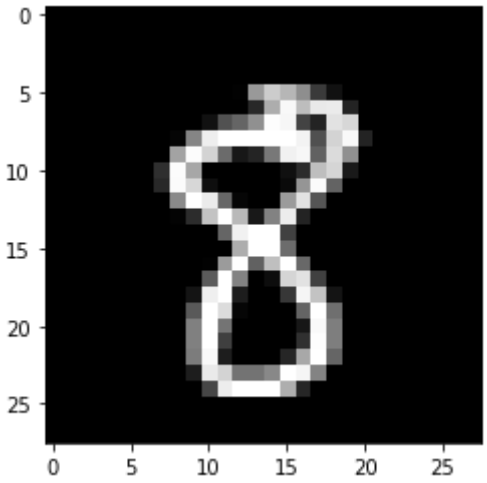
In [22]:
```
print(len(x_train))
```

```
11982
```

In [23]:
```
x_test=[]
y_test=[]
for i in range(10000):
  if y_test1[i]==3 or y_test1[i]==8:
    temp=[]
    for j in range(784):
      temp.append(x_test1[i][j])
    x_test.append(temp)
    y_test.append(y_test1[i])
```

In [24]:
```
x_train=np.matrix(x_train)
y_train=np.array(y_train)
x_test=np.matrix(x_test)
y_test=np.array(y_test)
```

In [25]:
```
plt.imshow(x_train[9].reshape((28,28)),cmap='gray')
plt.show()
```



In [26]:
```
nb_model=GaussianNB()
```

In [27]:
```
fit_nb=nb_model.fit(x_train,y_train)
```

In [28]:
```
prediction=fit_nb.predict(x_test)
con_matrix=confusion_matrix(y_test,prediction)
print(con_matrix)
```

```
[[435 575]
 [ 22 952]]
```

In [29]:
```
def diagonal_sum(con_matrix):
  sum=0
  totalSum=0
  for i in range(2):
    for j in range(2):
      totalSum+=con_matrix[i][j]
      if i==j:
        sum+=con_matrix[i][j]
  return sum,totalSum
```

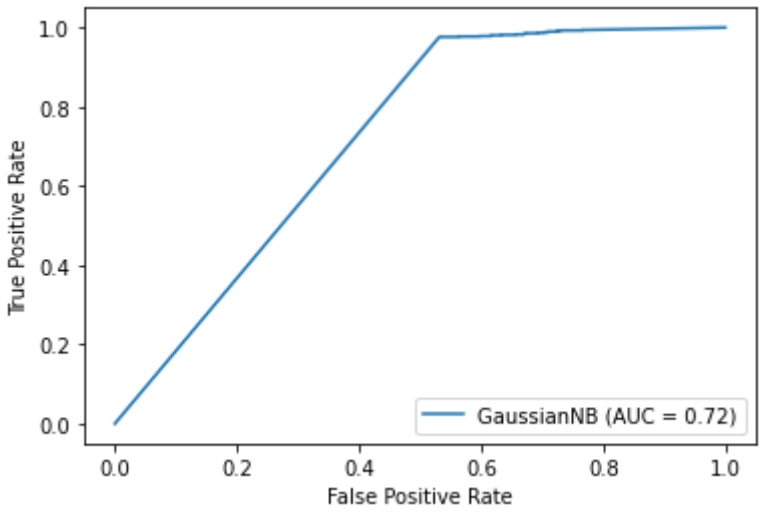
In [30]:
```
sum,totalSum=diagonal_sum(con_matrix)
print(sum,totalSum)
```

```
1387 1984
```

In [31]:
```
print(f'Accuracy % : {sum/totalSum}')
```

```
Accuracy % : 0.6990927419354839
```

In [33]:
```
metrics.plot_roc_curve(fit_nb, x_test, y_test)
plt.show()
```



CONCLUSION & DISCUSSION

We see a Drastic Drop in Accuracy in Case 2 because the digits 3 and 8 are very similar in nature as compared to digits 1 and 0. This confuses our classifier and it tends to misclassify the digit.