Handwritten Digit Classification

Name - Amith Korada

Course - AI & ML
(Batch - 4)

Duration - 12 Months

Problem Statement -Use MNIST dataset to create a classifier for all the 10 digits.

Prerequisites -

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has the latest version of python. The following URL https://www.python.org/downloads/ can be referred to as download python.

The second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this URL https://www.anaconda.com/download/ You will also need to download and install the below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run the below commands in command prompt/terminal to install these packages pip install -U sci-kit-learn pip install NumPy pip install scipy if you have chosen to install anaconda then run the below commands in anaconda prompt to install these packages conda install -c sci-kit-learn conda install -c anaconda numpy conda install -c anaconda scipy.

1. Importing necessary libraries-

```
from keras.datasets import mnist
from keras.utils import np_utils
import matplotlib.pyplot as plt
import numpy as np
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

2. Loading the dataset-

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

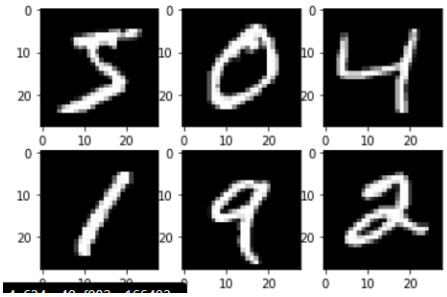
```
x_train.shape
```

```
(60000, 28, 28)
```

```
x_test.shape
```

```
(10000, 28, 28)
```

```
fig, ((ax1, ax2,ax3), (ax4, ax5, ax6)) = plt.subplots(2, 3)
ax1.imshow(x_train[0], cmap='gray')
ax2.imshow(x_train[1], cmap='gray')
ax3.imshow(x_train[2], cmap='gray')
ax4.imshow(x_train[3], cmap='gray')
ax5.imshow(x_train[4], cmap='gray')
ax6.imshow(x_train[5], cmap='gray')
plt.show()
```



3. Converting image to a vector-

```
X_train = x_train.reshape((x_train.shape[0], x_train.shape[1] * x_train.shape[2]))
X_test = x_test.reshape((x_test.shape[0], x_train.shape[1] * x_train.shape[2]))
```

```
X_train.shape
```

```
(60000, 784)
```

4. Normalising the data-

```
X_train = X_train / 255
X_test = X_test / 255
```

```
y_train_cat = np_utils.to_categorical(y_train)
y_test_cat = np_utils.to_categorical(y_test)
```

5. Classification using MLP -

```
mlp.fit(X_train,y_train_cat)
```

```
C:\Users\amith\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:614: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
```

```
MLPClassifier(hidden_layer_sizes=(15,))
```

```
mlp_pred= mlp.predict(X_test)
```

```
print(classification_report(y_test_cat, mlp_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       980
           1       0.97      0.98      0.97      1135
           2       0.95      0.89      0.92      1032
           3       0.94      0.92      0.93      1010
           4       0.94      0.93      0.94       982
           5       0.93      0.93      0.93       892
           6       0.96      0.96      0.96       958
           7       0.95      0.93      0.94      1028
           8       0.92      0.91      0.91       974
           9       0.91      0.92      0.91      1009

   micro avg       0.94      0.93      0.94     10000
   macro avg       0.94      0.93      0.94     10000
weighted avg       0.94      0.93      0.94     10000
 samples avg       0.92      0.93      0.93     10000
```

```
print(accuracy_score(y_test_cat, mlp_pred))
```

```
0.9058
```

6. Classification using SVM-

```
svm = SVC()
```

```
svm.fit(X_train, y_train)
```

```
SVC()
```

```
svm_pred = svm.predict(X_test)
```

```
print(classification_report(y_test, svm_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.98      0.97      0.98      1032
           3       0.97      0.99      0.98      1010
           4       0.98      0.98      0.98       982
           5       0.99      0.98      0.98       892
           6       0.99      0.99      0.99       958
           7       0.98      0.97      0.97      1028
           8       0.97      0.98      0.97       974
           9       0.97      0.96      0.97      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

```
print(accuracy_score(y_test, svm_pred))
```

```
0.9792
```