Guided Assignment On Supervised Learning

Name: **Shreya Shrivastava**

Course: **Artificial Intelligence And Machine Learning**

Batch Four

Duration: **12 Months**

Problem statement: Use MNIST dataset to **create a classifier** for all the 10 digits. First implement the classifier by squeezing the image into a vector and then using a **MLP**. Now, try the same task using a different machine learning classifier such as an **SVM** to **check the gain in performance** by using perceptrons as compared to conventional machine learning techniques.

Prerequisites:

The libraries as well as things required in order for the program to work:

I. **Python 3.6** : The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic. Second 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/

II. **SCIKIT-LEARN** **:** You will also need to download and install scikit learn after you install either python or anaconda from the steps

above. If you have chosen to install python 3.6,then run the following commands in command prompt/terminal to install these packages :

SCIKIT LEARN: pip install -U scikit-learn

If using Anaconda then run the following commands in anaconda prompt to install these packages:

SCIKIT LEARN: conda install -c scikit-learn

III. METHODS USED:

A. MULTI LAYERED PERCEPTRON

B. SUPPORT VECTOR MACHINES

THE PROJECT :

1. Importing the libraries and loading the MNIST dataset. We then proceed to vectorise the images and split the data into training and testing set respectively.

```
from sklearn.datasets import load_digits
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.neural_network import MLPClassifier as mlp
digits = load_digits()
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.5, shuffle=False)
```

2. Then we fit a MLP classifier on the training set and calculate the accuracy on the test dataset. Displaying the results.

```
clf_mlp = mlp(hidden_layer_sizes=(150,90,40), max_iter=300,activation = 'relu',
              solver='adam',random_state=1)
clf_mlp.fit(X_train, y_train)
y_pred_mlp = clf_mlp.predict(X_test)

print(f"Classification report for classifier {clf_mlp}:\n"
      f"{classification_report(y_test, y_pred_mlp)}\n")
```

```
Classification report for classifier MLPClassifier(hidden_layer_sizes=(150, 90, 40), max_iter=300, random_state=1):
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        88
           1       0.94      0.96      0.95        91
           2       0.96      0.99      0.97        86
           3       0.97      0.85      0.91        91
           4       0.98      0.96      0.97        92
           5       0.95      0.96      0.95        91
           6       0.97      0.99      0.98        91
           7       0.97      0.96      0.96        89
           8       0.91      0.88      0.89        88
           9       0.86      0.96      0.91        92

    accuracy                           0.95       899
   macro avg       0.95      0.95      0.95       899
weighted avg       0.95      0.95      0.95       899
```

3. Then we fit a SVM model on the training set and calculate the accuracy on the test dataset. Displaying the results.

```
clf_svm = SVC(gamma=0.001)
clf_svm.fit(X_train, y_train)

# Predict the value of the digit on the test subset
predicted_svm = clf_svm.predict(X_test)
print(f"Classification report for classifier {clf_svm}:\n"
      f"{classification_report(y_test, predicted_svm)}\n")
```

```
Classification report for classifier SVC(gamma=0.01):
              precision    recall  f1-score   support

           0       1.00      0.65      0.79        88
           1       1.00      0.74      0.85        91
           2       1.00      0.64      0.78        86
           3       1.00      0.64      0.78        91
           4       1.00      0.55      0.71        92
           5       0.93      0.98      0.95        91
           6       1.00      0.68      0.81        91
           7       1.00      0.49      0.66        89
           8       0.25      1.00      0.40        88
           9       1.00      0.61      0.76        92

    accuracy                           0.70       899
   macro avg       0.92      0.70      0.75       899
weighted avg       0.92      0.70      0.75       899
```