Name : Marwan Mohamed Abd Elmonem ID : 20190513

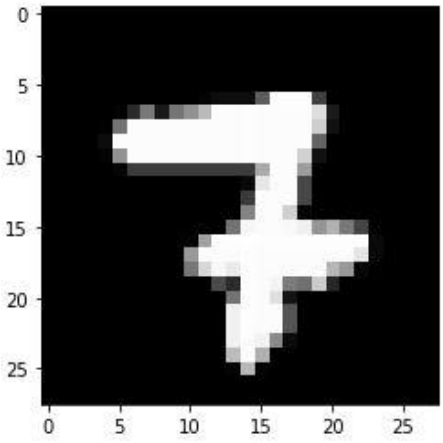
Report for MNIST dataset Assignment

we are required to implement a machine learning algorithm for classification of the handwritten digits (MNIST) using KNN classifier

we used 60,000 sample from the dataset for training and 10,000 for testing and since its saved in a 1-dimentional array of size 784 so we will reshape it in form of 2-dimentional array with size of 28*28 to be easily manipulated and visualized, we will divide the photo into grids to help in feature extraction

```
def display_img(mnist_index):
    """
        this function takes the mnist image index and plotting it
    """
    image = mnist_index
    image = np.array(image, dtype='float')
    pixels = image.reshape((28, 28))
    plt.imshow(pixels, cmap='gray')
    plt.show()
```

```
display_img(train_images[4832])
```



then in feature extraction phase we used the center of mass (centroid) of each grid, and we save them in feature vector so that each number has many feature vectors

but close in value so that we can compare a feature vector of a test sample with the trained ones.

```
def get_centroid(img):

    """
        return feature vector for current image

    """

    feature_vector = []

    for grid in imaged_grid(img , 7 , 7 ) :

        Xc = 0
        Yc = 0
        sum = 0

        for index, x in np.ndenumerate(grid):
          sum+= x
          Xc += x * index[0]
          Yc += x * index[1]

        if sum != 0 :
            feature_vector.append( Xc/ sum )
            feature_vector.append(Yc/ sum )
        else :
             feature_vector.append(0)
             feature_vector.append(0)


    return np.array(feature_vector)
```

After training we test the model using KNN classifier by computing the square distance between each feature vector in test samples with the trained samples and we classify according to the minimum distance.

We conclude that using k with value 8 is the optimal value for highest accuracy ... decreasing the value decreases the accuracy and increasing it resulting in nonuniform scatter plotting but not increasing over k = 8

```
def KNN(train_features, test_features, train_labels):
    """
        return predicted lablels to be compared with the test label
    """
    knn = KNeighborsClassifier(5, metric='euclidean')
    knn.fit(train_features, train_labels)  # fit train data
    prediction = knn.predict(test_features)  # test data
    return prediction
```

```
Knn_prediction = KNN(train_features, test_features , train_labels )
```

```
print("Accuracy Score =", accuracy_score(test_labels, Knn_prediction) * 100, "%")
```

Also, we conclude that using rectangular grid instead of squared grid increases the accuracy by a quite difference relatively were using a grid of 7*14 scored the highest accuracy of 92.56%

Additionally, we used the model with 10,000 sample for training and 1000 for testing for a fast result but its downside that it doesn't score high accuracy comparing to the whole dataset