Table of Contents

Question 1

The Fashion MNIST dataset consists of Zalando's article images. There are 60000 training and 10000 test 28 x 28 greyscale images, each pixel value ranging from 0 to 255 indicating the lightness or darkness of the pixel. The datasets have 785 columns, the first corresponding to the label (0-9 for 10 classes) and the rest representing the 784 (28 x 28) pixel values. The following steps are performed to load and process the dataset.

1. The dataset is downloaded from the source https://www.kaggle.com/datasets/zalando-research/fashionmnist?resource=download and stored in the location /content/drive/MyDrive/UMD/DATA 604/fashion_mnist_data/
2. The training and test datasets are loaded from the respective CSV files as dataframes and combined into a single dataframe.
3. The dataframe is then sorted by the labels of the images to obtain images of each class sequentially. Now, the dataframe contains 70000 images with 7000 images in each class.

Question 2

In order to estimate the computer's capability to perform numerical computations, a simple for loop performing two O(1) operations in each iteration is performed and it requires 136 µs. Since this for loop is of order O(n) and n = 1000, the time complexity for a single arithmetic operation is 0.098 µs. The brute force K-Nearest Neighbours classification algorithm is of testing time complexity O(kNd), where k is the number of nearest neighbors, N is the number of samples in the training set and d is the number of dimensions of the dataset. Hence, the worst-case time complexity for the entire dataset, i.e., N = 70000, is 107.565 seconds.

Question 3

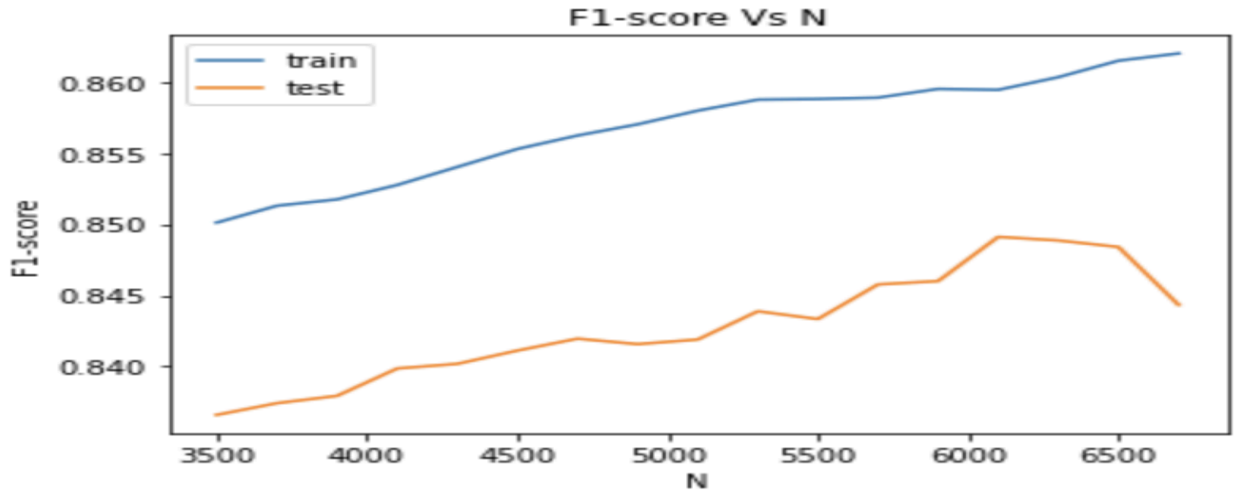
On the basis of the above calculation, an experiment is designed for evaluating the optimal value of N (the number of training samples in each class), we apply the brute force K-Nearest Neighbours algorithm for different values of N. The following steps are followed for studying the train-test split for classification:

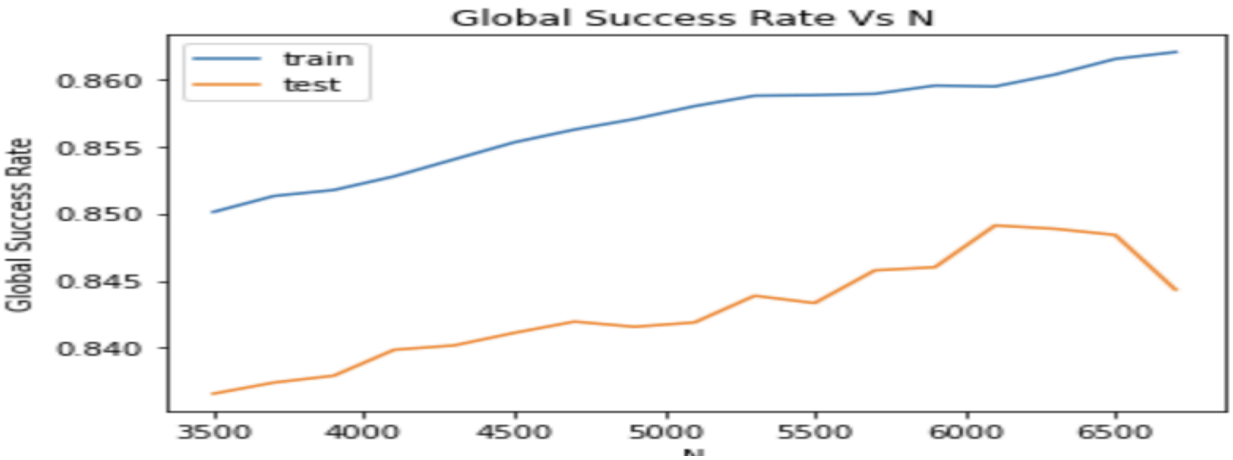
1. The split_train_test(df, N) function splits the dataset into training and testing datasets for each value of N. For the images corresponding to class i, the first N images are added to the training dataset, and the remaining 7000-N images are included in the test dataset.
2. The train_knn(X_train, y_train) function creates a KNN model with k = 20 using the brute force algorithm and standard euclidean metric. It fits the model on the given training set and returns the model. It also computed the time required for training the model.
3. The test_knn(knn_model, X_test, y_test) function uses the fitted KNN model to predict the class label for the test dataset. It returns the predicted labels as well as computes the time required for testing.
4. The above 3 steps are performed for 17 different values of N ranging from 3500 to 6700. Along with fitting the knn model and predicting the class labels for test datasets with different values of N, the F1 score and global success rate or accuracy are stored for each value of N.

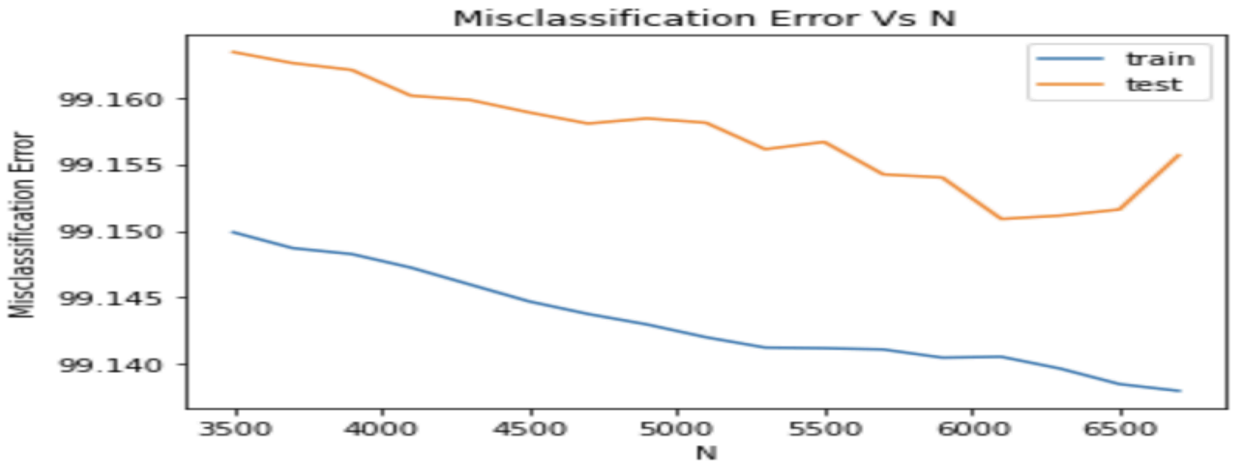
Since, the brute force algorithm is O(1) in training and O(kNd) in testing (for each test datapoint), for 10N training samples and 10*(7000-N) testing samples, we will require testing time complexity proportional to 20 * 10N * 784. Since the time required in the worst case is 107.565 and 17 values of N ranging from 3500 to 6700 are considered, the approximate expected time required for this experiment is 1828.605 seconds.

The following observations and conclusions are made from the given experiment:

1. It takes 0.521 seconds for training with the highest value for N = 6700, whereas the testing time for the same is 13.984 seconds (slightly lesser than the calculation done before).
2. It takes 3774.761 seconds (slightly more than expected) for performing the experiment.
3. The accuracy and F1-score follow an increasing trend as N increases till N = 6100. The misclassification error is reported as 100 - accuracy. The metrics are plotted as follows:







4. Hence, with the highest testing F1-score and accuracy score of 0.849, and the lowest misclassification error, N = 6100 is the optimal value for the number of training samples in each class. The accuracy score or the global success rate doesn't capture the robustness of the classification model, i.e., the global success rate may be high even if the

success rates for some classes are significantly low or it may not be able to capture if the model is giving a high number of false positives/negatives. Hence, the F1-score which is a harmonic mean of precision and recall is also used to evaluate the optimality of the chosen value of N.

Question 4

To study the structure of the train-test split for classification, a new method is proposed for choosing N samples from each class for training:

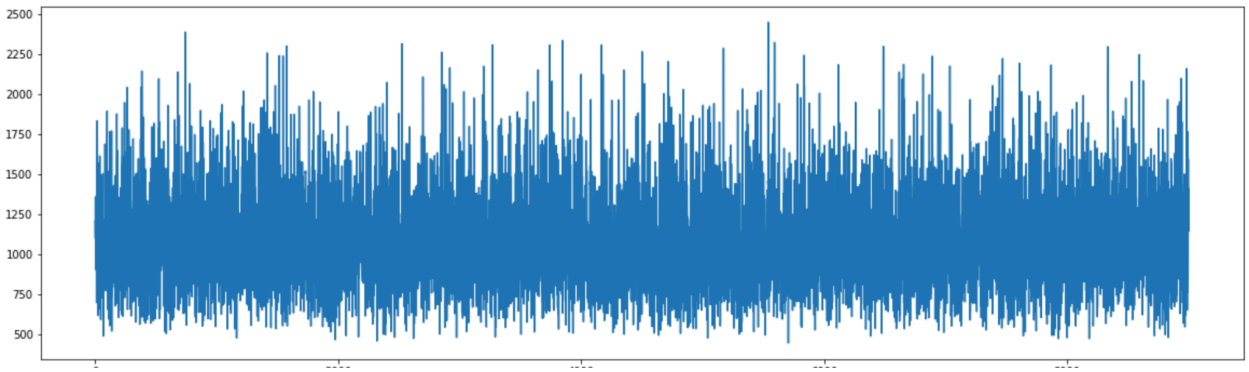
1. The split_train_test_2(df, N) splits the dataframe into training and testing datasets by randomly choosing N samples from each class for training and the remaining ones for testing. It uses the 'stratify' option to avoid class imbalance.
2. Furthermore, a KNN model is fitted on the training set obtained for N = 6100 (optimal N obtained in the last question) and used for predictions on the test dataset.

An F-1 score and accuracy score of 0.862 is obtained for the given model on the test dataset. These metrics are slightly higher than the previous method for choosing N training samples from each class. Although the percentage change (1.5%) is not very significant, a random selection of training samples proves to build a more robust and generalized KNN model. Hence, the second method is chosen for further experiments.

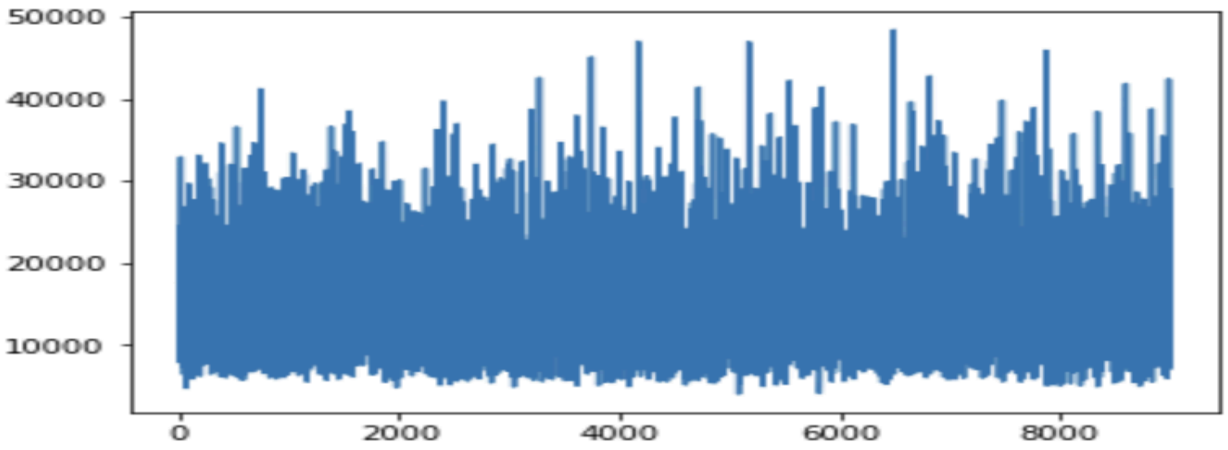
Question 5

To analyze the role of the metric in classification, the following steps are performed:

1. The Euclidean distances and indices of the 20 nearest neighbors for the data points in the test dataset are calculated using the KNN model fitted in the last question.
2. The average Euclidean distances over 20 neighboring data points are calculated for each data point in the test dataset. A plot of these averages is obtained:



3. A KNN model is fitted on the training dataset using the 'l1' norm or 'Manhattan' distance metric. The train_knn_l1(X_train, y_train) function returns this KNN model. It is further used to predict the class labels for the test dataset.
4. Similarly, Manhattan distances and indices of the 20 nearest neighbors for the data points in the test dataset.
5. The average Manhattan distances over 20 neighboring data points are calculated for each data point in the test dataset. A plot of these averages is obtained:



Although the F1-score and accuracy score obtained with an 'l1' norm metric is the same as that of the Euclidean distance metric (0.862), the Manhattan distances are much greater than Euclidean distances as observed from the plots. This observation agrees with the mathematical proof as follows:

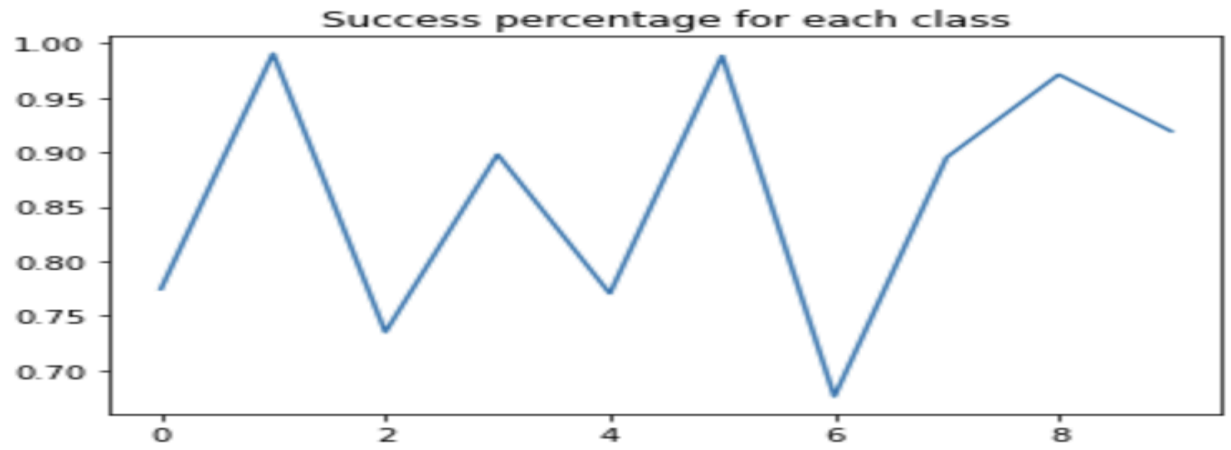
$$||x||^2_2 = \sum^N_{i=1}|x_i|^2 \leq (\sum^N_{i=1}|x_i|^2 + 2 \cdot \sum_{i,j,\, i<j} |x_i||x_j|) = ||x||^2_1$$

Question 6

Since, the performance of KNN models with Euclidean metric and Manhattan distance metric is the same, the success rates for individual classes are evaluated on the KNN model trained using the Manhattan distance. The success rates for individual classes as the diagonal elements of the confusion matrix after normalizing each element by dividing it by the total number of test data points in that class. The success rates for individual classes are as follows:

Class	Success Rate (%)
0	77.44
1	99.08

Class	Success Rate (%)
2	73.5
3	89.82
4	77.04
5	98.87
6	67.59
7	89.54
8	97.16
9	91.95



Class with the label 1 has the highest success rate (99.08 %) and class with label 6 has the lowest success rate (67.59 %). The global success rate is 100 * accuracy score = 86.2 %.

Code

The Python code for all the experiments discussed above is attached below: