**k-Nearest Neighbors**

Riya Srivastava

**Introduction**

This is the first assignment of the course Predictive Analytics in which we learnt about the basics of Predictive Analytics and the k-Nearest Neighbor Classification Learning algorithm. In this project, we are given with an image classification dataset called Fashion-MNIST from the MNIST database. Our goal is to use a k-Nearest Neighbors classifier to classify the clothing items in the dataset.

The Fashion-MNIST data set consists of a training set of 60,000 instances and a test set of 10,000 instances. The first column in both training and testing dataset consists of the class labels representing the article of clothing. The rest of the columns contain the pixel values. Each instance is a 28x28 grayscale image that is associated with a label from 10 classes. The classes are from 0-9 namely T-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag and ankle boot respectively.

Each image is 28 pixels in both height and width with total 784 pixels for each image. The pixel value is an integer between 0 to 255. The higher the value of pixel, darker is the brightness of the pixel.

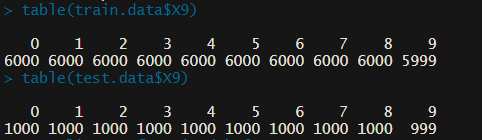
We usually perform feature selection for large datasets but in this case we cannot omit columns because each column represents a pixel which is important for the formation of an image.

**Analysis**

**Step 1:** Import all the necessary libraries. The required libraries for this project are class, gmodels and caret. The library class is used for classification problems. Since, k-NN is a classification algorithm, it is important to import this library.

**Step 2:** Import the dataset. For this project, we have been given the dataset of Fashion-MNIST which has two .csv files namely fashion\_train and fashion\_test. It is always a good practice to use the function file.choose() while importing a dataset as it allows the user to choose the files from their local machine.

**Step 3:** Speculating and viewing the structure of data. In this step, we get to know our datasets a little more in detail by viewing the structure and the attributes of the data files using functions ‘str()’ and ‘View()’ in R. We will also use the ‘table()’ function to list the values of column ‘X9’ which is the first column in both the training and testing dataset and gives the labels of clothing items. The below output snippet from R shows the results for each label. In training data, there are 6000 items each for labels 0-8 and 5,999 for label 9 totaling to 59,999 which is the total number of observations in our training dataset. Below that, we can see the output for testing data which has 1000 items each for labels 0-8 and 999 for label 9 adding to 9,999 which is the total number of observations in testing dataset.



**Step 4:** Normalization of values. Since the data varies in values, we normalize the data values so that they are rescaled to fit a particular range. For this, we have defined a normalize vector and used it on both the training and testing dataset using the ‘lapply’ function. Further, we have created data frames from the resultant and named them as ‘train\_n’ and ‘test\_n’.

**Step 5**: Find K and create vectors. The next step is to find the value of K which is the number of neighbors, in this case is an odd number near to the square root of the size of training dataset that is square root of 59,999 which is approximately 245 when rounded off. We will then create vectors of our normalized testing and training data and store them in ‘MyTestData’ and ‘MyTrainData’ respectively. Similarly, we will also create vectors of labels column from our original dataset and store them in ‘MyTrainData.labels’ and ‘MyTestData.labels’.

**Step 6:** Apply k-NN and create Cross Table. We have now arrived at the most important step of applying the k-NN algorithm to our dataset. The function ‘knn’ in R requires us to define the training and testing dataset in the form of dataframes, describe the true classifications of training set and define the value of k (here, =245) which is the number of neighbors considered. We will now create a cross table of test data labels for the values predicted by knn(Appendix A) which are stored in ‘knn\_predict’. As shown in the screenshot, the legend gives us information about the table. Let’s consider an example from the table for the square depicting the values in MyTestData.labels 0 and knn\_predict 0. First is N that is number of images for label 0 i.e T-shirt/Top, this means there are 846 T-shirts and tops after applying knn. The next number is N/Row total = 846/1000 = 0.846. Next is N/Col Total = 846/1169 = 0.724 and the last is N/Table Total = 846/9999 = 0.085 as there are 9,999 observations in the test data. This gives us an in-depth analysis into our results. Similarly, for other squares we have the values as shown in the output snapshot.

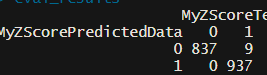
**Step 7:** Create of Z-score standardisation method. Now, we will use the Z-score standardisation method for rescaling of the data. Z score transformation rescales all the values by subtracting mean from each value and then diving the resultant by standard deviation. The trnasformed values will now have a mean of 0 and a standard deviation of 1.First, we will rescale the values of training data and store them in ‘MyZScoreData’ and the values of testing data in ‘MyZScoreData2’ as dataframes. Next, we created Z-score vectors for training and testing sets namely ‘MyZScoreTrainData’ and ‘MyZScoreTestData’ respectively. After this, we created Z-score vectors for training and testing data labels and stored them in ‘MyZScoreTrainData.labels’ and ‘MyZScoreTestData.labels’ respectively.

**Step 8:** Applying k-NN to Z-score predicted and creation of Cross Table. In this tep, we will again apply the function ‘knn()’ to the z-score predicted values and store them in ‘MyZScorePredictedData’. Here, we will use ‘MyZScoreTrainData’ as the training set, ‘MyZScoreTestData’ as the testing set and ‘MyZScoreTrainData.labels’ as the true classification of training set with 245 as the value of k.

In the cross table created from ‘MyZScoreTestData.labels’ and ‘MyZScorePredictedData’, the interpretation of values will be the same as the output 1.

**Step 9:** Evaluating Results. To evaluate the results, we will create a vector called ‘eval-results’ which is a table storing results of ‘MyZScorePredictedData’ vs ‘MyZScoreTestData.labels’. Now, we will calculate the true negative(TN), false positive(FP), false negative(FN) and true positive(TP) as per figure 1.

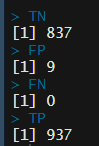
From the output 3 screenshot we can see that a number of values are given but for our analysis we will consider the first four values for the explanation of confusion matrox and calculation of accuracy. To make it more clear, below are the values from the grid that have been considered for calculation:

TN = eval\_results[1,1]

FP = eval\_results[1,2]

FN = eval\_results[2,1]

TP = eval\_results[2,2]

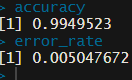
In this case, the values are depicted below:

**Step 10:** Calculation of accuracy and error rate. Now, we will calculate the accuracy of our k-NN model and it’s error rate with the above values. The formula for accuracy and error rate used is:

**Accuracy = (TN + TP) / (TN + TP + FP + FN)**

**Error Rate = 1 - Accuracy**

The value of accuracy and error rate is:



**Findings:**

To prove the accuracy and further verify our results, we will check if the values predicted by our k-NN algorithm vary from the testing data or not. For this, we have exported the results of ‘MyZScorePredictedData’ and ‘test.data$X9’ (which is the labels column for test data) into two .csv files to make a comparison between them and combined the results further into a file named Comparison.csv. We can see that the 18th image has a different result from the test set. In the test set, it was labelled number 4(coat) but our results show that it is number 6(shirt). Therefore, these differences further prove that our results are valid.

**Final Conclusion:**

To conclude we can say that our model is 99.4% accurate with an error rate of 0.005%. The values of accuracy and error rate depends on the values chosen by us for TP, TN, FP and FN. The goal of applying k-NN algorithm was achieved but this was just one of the ways of performing the task. Another way to perform the given task was to slice the dataset i.e reduce the number of observations in the dataset and perform the same task. I did perform the task with sliced data and did not get satisfactory results, therefore, decided to consider the entire dataset.

A screenshot of a cell phone

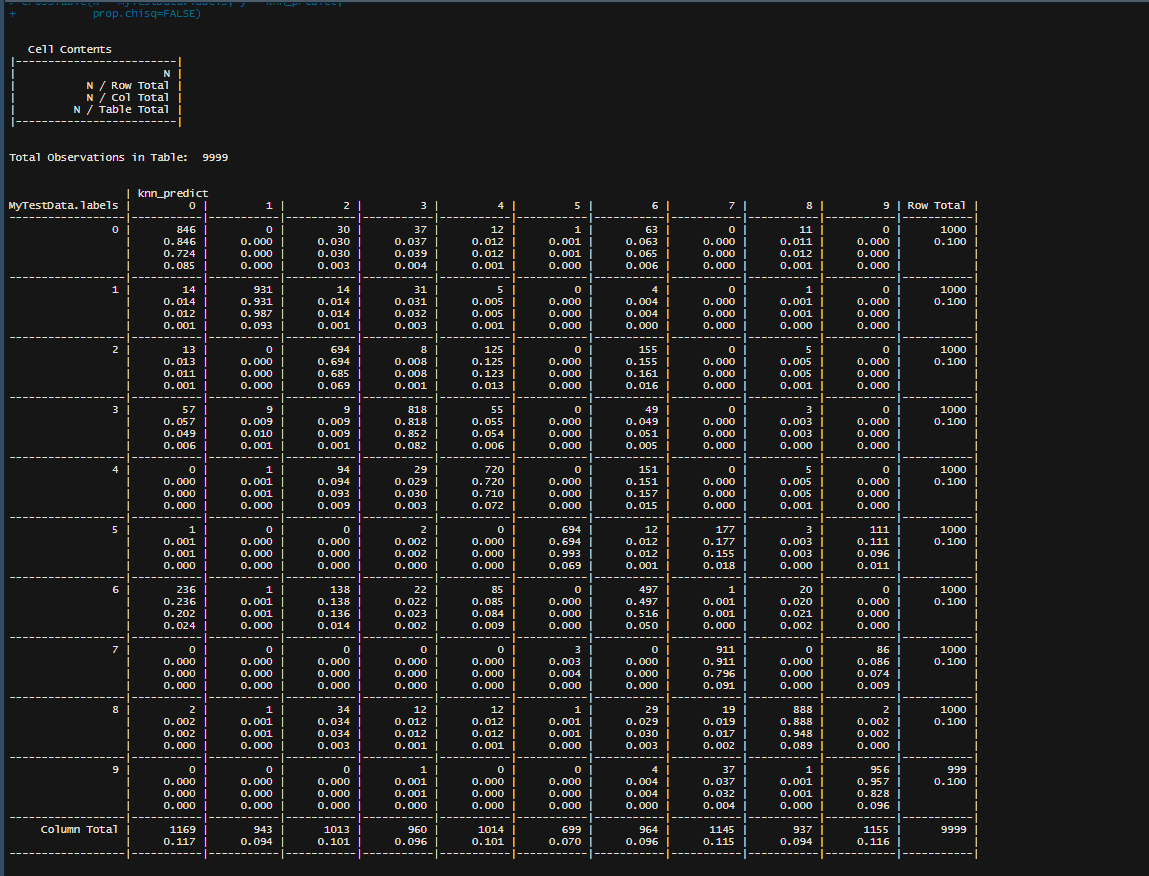
Description automatically generatedThe size of the sliced dataset was training set = 39,999 observations and testing set = 4,999 observations.The accuracy achieved with sliced dataset was 99.5% and the graph for ‘k vs accuracy’ and ‘k vs error rate’ are as shown below:

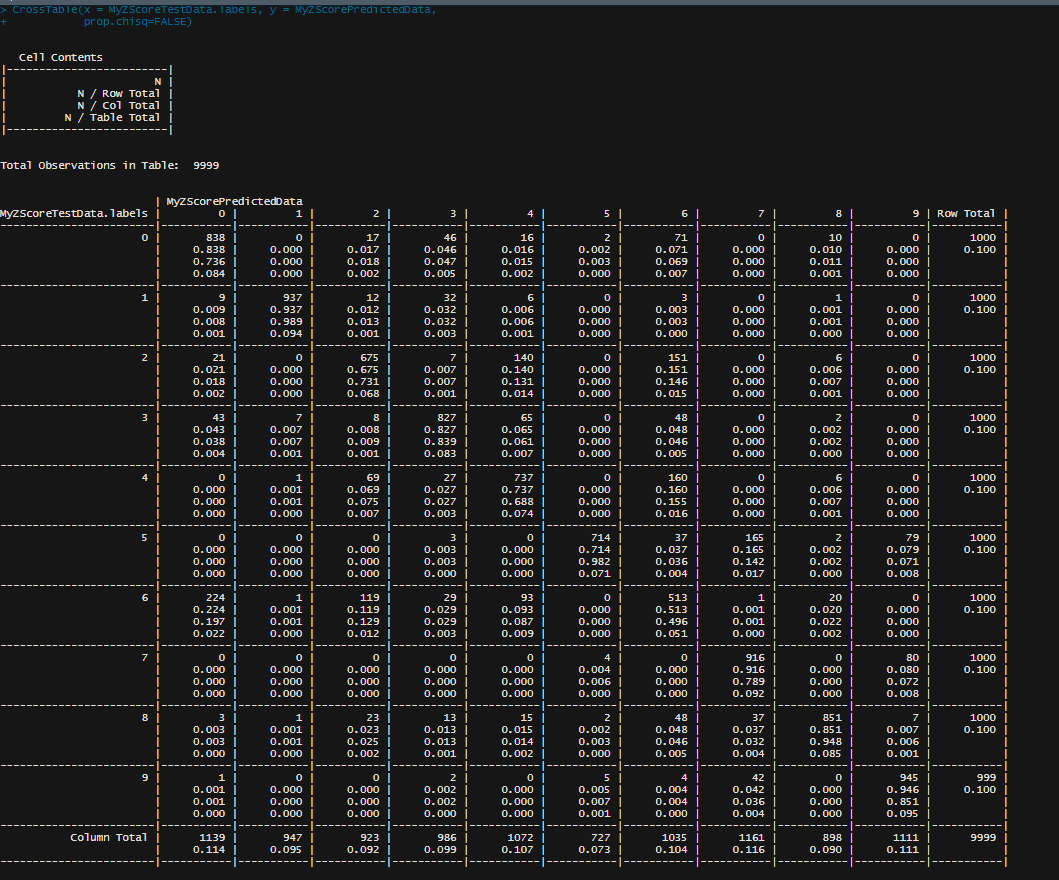
A close up of a mans face

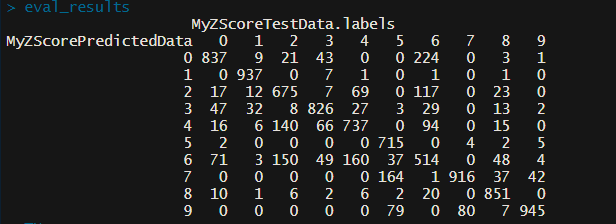
Description automatically generatedThe pros of using a sliced dataset is that it gives faster results but a disadvantage is that the results could be biased in some cases. It depends on the analyst how he wants to study the data and also on the hardware available. If a CPU with higher computational power is avaliable, a larger dataset can give better and faster results.

**Appendix A**

Output 1: Cross Table showing the results of MyTestData.labels and knn\_predict



Output 2: Cross Table showing the results of MyZScoreTestData.labels and MyZScorePredictedData

Output 3: table showing the evaluation results of MyZScorePredictedData and MyZScoreTestData.labels

**References**

[1] Data Science by Vijay Kotu and Bala Deshpande, Second Edition

[2] Class Presentation and Lecture, Example codes

[3] Han Xiao, Kashif Rasul, and Roland Vollgraf, "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms," 2017

[4] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Technical report, University of Toronto, 2009.

[5] Shobit Bhatnagar, Deepanway Ghosal, and Maheshkumar H. Kolekar, “Classification of Fashion Article Image