Machine learning for Data Science
Assignment no 1

Implement and analyze K-NN classifier

Muhammad Waleed Usman
i19-2140 MSDS-A

Submitted to:
Ma’am Amina Asif

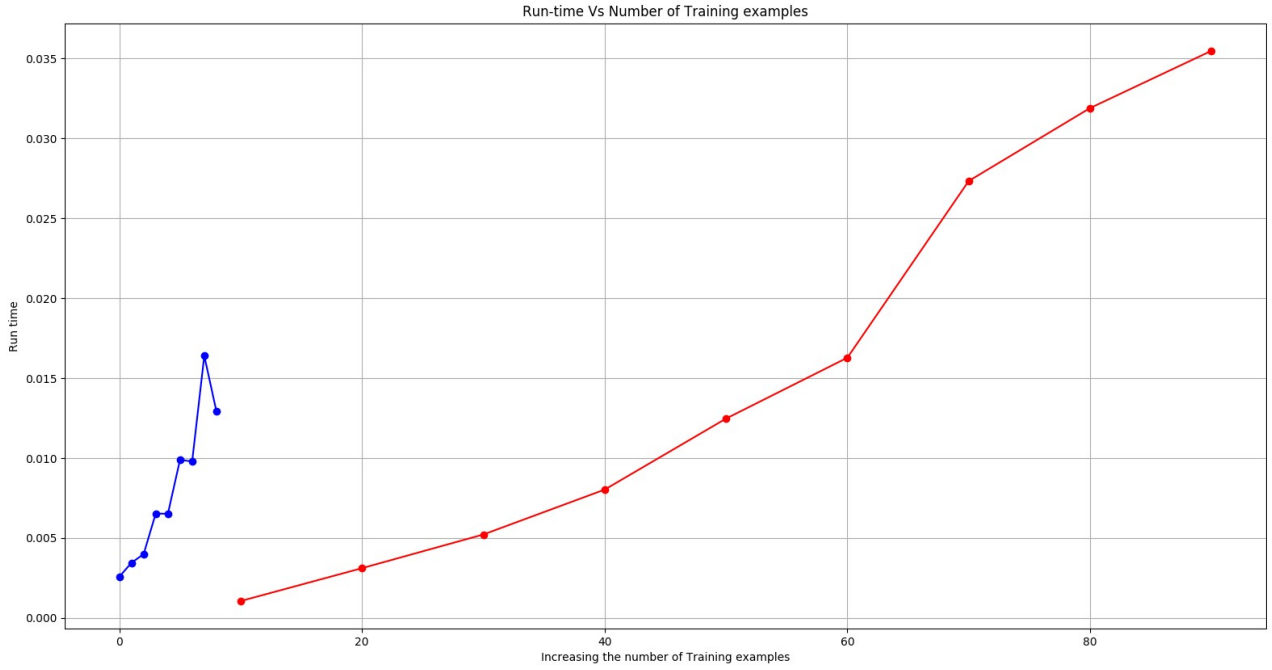
Question No.1. Modify the predict method in NN class such that it takes K as an input and performs K-NN classification instead of 1-NN.

Solution: Firstly finding the least distances in the array of our neighbors. Then after that finding the labels for each shortest distance. After that finding the max occurring label and appending it to our testing labels.
Here is the code:

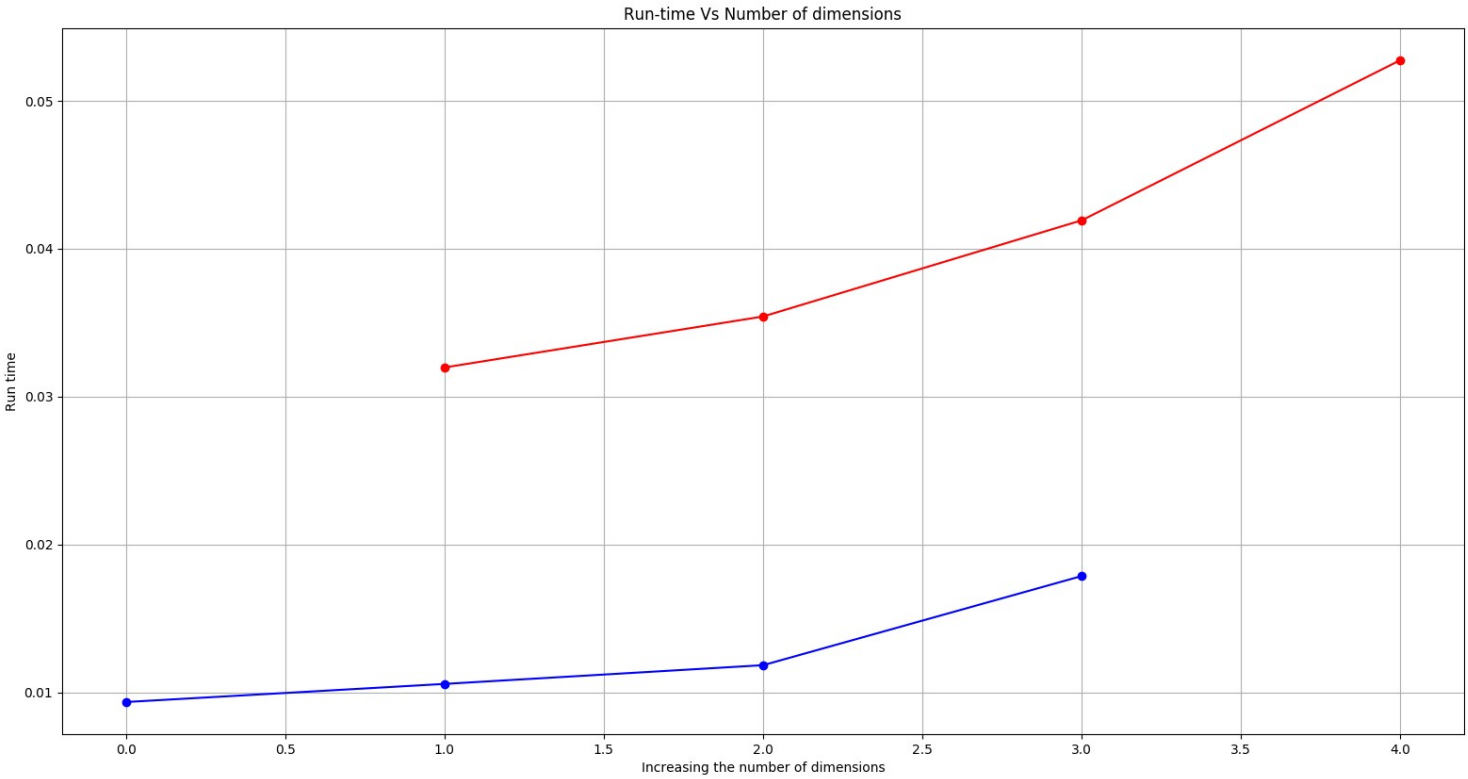
```
def predict(self, Xts, k):
    Yts=[]
    Xts=np.array(Xts)
    for t in self.Xtr:
        distances=np.sqrt(np.sum(np.power((Xts-t), 2), axis=1))
        dist = list(zip(self.Xtr,self.Ytr,distances.tolist())) # List of training data with labels and their distances
        dist.sort(key=lambda tup: tup[2]) # sorting the distances using the distance metric
        neighbors = list()
        for i in range(k): # Findind the k neighbors using the least k distances
            neighbors.append(dist[i][1])
        prediction = max(set(neighbors), key=neighbors.count) # finding the maximum occuring label in the neighbors
        Yts.append(prediction)
    return Yts
```

Question No.2. Using your implementation of K-NN perform the following analysis for toy dataset. Also, Compare the performance of your implementation with SKlearn implementation.

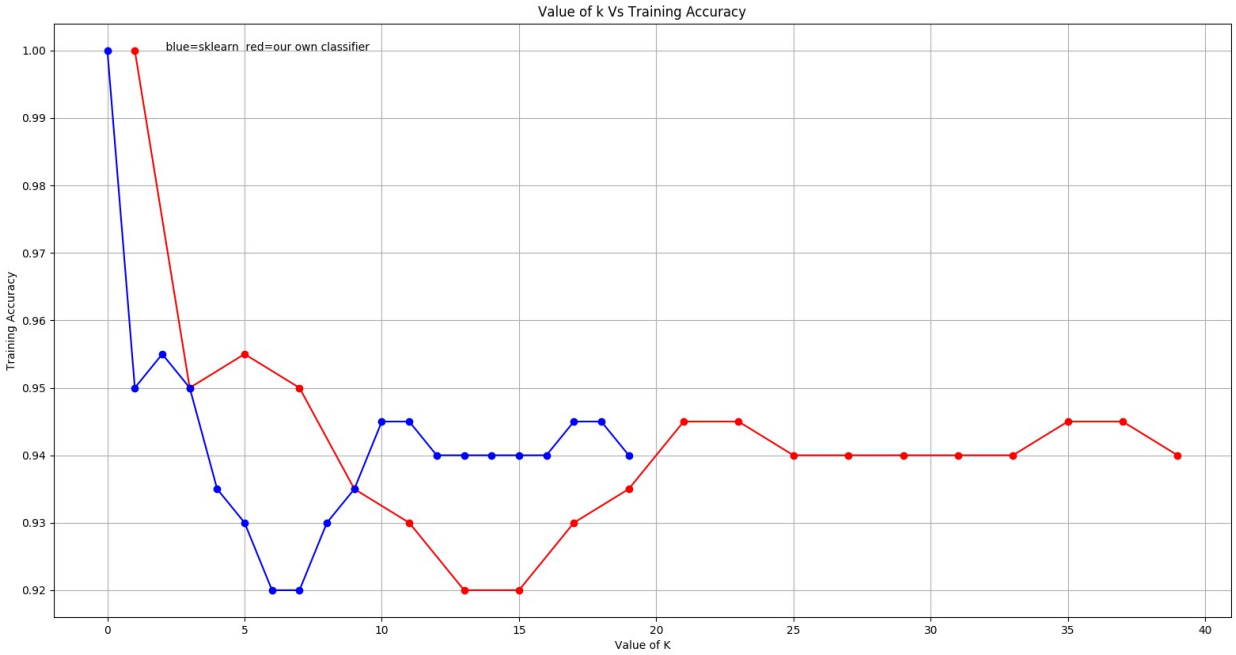
Part A. How is run-time affected by increase in number of training examples?
Solution: Run time is the total time taken by the program to run and show results. So when we increase the training examples, this actually increases the run-time. Because the KNN algorithm will take more examples and that's why the time taken will be more if we increase training examples.
Here is the plot which show when we increase the training data the run time also increase:
(blue line indicates the sklearn implementations and red line shows our own classifier.)
The important thing that run time of our classifier tends to increase more because of low optimization or distance metric, but the sklearn one is not that much increasing (small change).



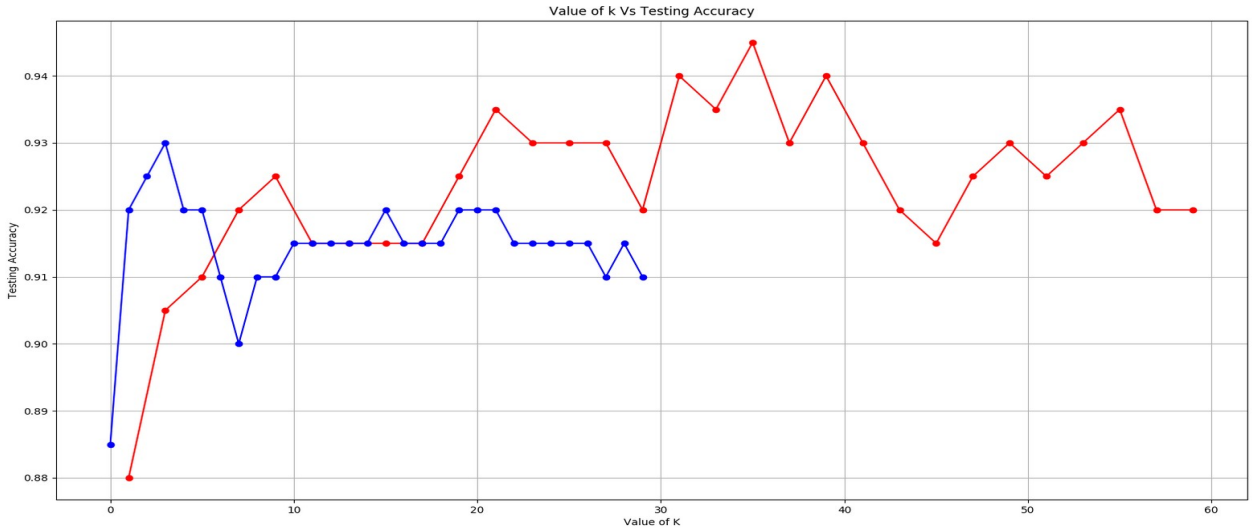
Part B. How is run-time affected by increase in dimensionality?
Solution: Run time is the total time taken by the program to run and show results. So when we increase the dimensionality, this actually increases the run-time. Because the KNN algorithm will execute more dimentions of data and that's why the time taken will be more.
Here is the plot which show when we increase the dimentions the run time also increase:
(blue line indicates the sklearn implementations and red line shows our own classifier)



Part C. How is training accuracy affected by change in K?
Solution: When we increase the number of k neighbors, the training accuracy decreases but there is a stage from where the accuracy become minimum and don't decrease any further.
The training accuracy starts from 100 percent when we have k=1 and tends to decrease. But at the end it become no more effective.
Here is the plot which show what is happening to our training accuracy when we increase the k:
(blue line indicates the sklearn implementations and red line shows our own classifier)



Part D. How is testing accuracy affected by change in K?
Solution: When we increase the number of k neighbors, the testing accuracy increases but there is a stage from where the accuracy become maximum and don't increase any further.
The training accuracy starts from minimum value when we have k=1 and tends to increase. But at the end it become no more effective.
Here is the plot which show what is happening to our testing accuracy when we increase the k:
(blue line indicates the sklearn implementations and red line shows our own classifier)



Question No.3. Give some ideas on how you think you can make your implementation more efficient?

Explanation:

KNN can be modeled using the three metrics, which are:

1) The distance function.

2) The number of neighbors k.

3) The weight of neighbors (or instances).

Distance is one of the important factor in the improvement of the KNN classifier. The KNN is improved using the best and optimum distance equation. In distance functions you must decide which is the best alternative. There are some ways you can introduce other techniques in order to improve the results. For example, using evolutionary computation to calculate the weights of instances, or using cross-validation to know a good approximation for the number k.

You can also improve efficiency (instead of accuracy), by using some **alternative data structures** which will provide you approximate solutions (i.e., KD-Tree, hash tables, etc).