ML assignment 2

Group 55

Members :
Vinit Raj (19CS10065)
Aryan Agarwal (19CS30005)

<u>Building a K nearest neighbour classifier on the spam-ham email dataset</u>

To run, install the dependencies mentioned in the requirments.txt file and then enter python3 questions.py in any terminal

We have used the knn algorithm(both weighted and unweighted) to build a classifier for the dataset

File structure:

knn_model.py

- We declare the knn model class in this file along with the classification algorithm.

utils.py

-here we have all the utility functions that help in reading the data and interacting with the model.

questions.py

-here we run the model with different types of the distance measures for all values of k and plot them.

Implementation details

function *read_data*

- reads the mail dataset and then vectorizes the text part of the mail using the TfIdf vectorizer

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

The length of the vector obtained is the count of distinct words across all documents

function cosine_similarity

- calculates the cosine similarity as dot product divided by product of norms.

function *euclidean_distance*

- calculates the *euclidean_distance as the norm of the difference of the two vectors.*

function *manhattan_distance*

- calculates *manhattan_distance as the abs sum of elements of the difference of the two vectors.*

function split_data

- shuffles the dataset and makes a 80:20 split for the train and test set

class *knnClassifier*

- Defines the main model for the classification of the dataset. It contains all the functions needed for finding the neighbors and assigning the classes.

function *classify*

- given a test sample it tries to classify the sample into either spam or ham category. First it orders the neighbours by corresponding distances then finds the majority class in the first k neighbours and assigns that to the sample for all k.

function *classify_weighted*

- similar to previous but uses weighted classification proportional to the inverse of distance.

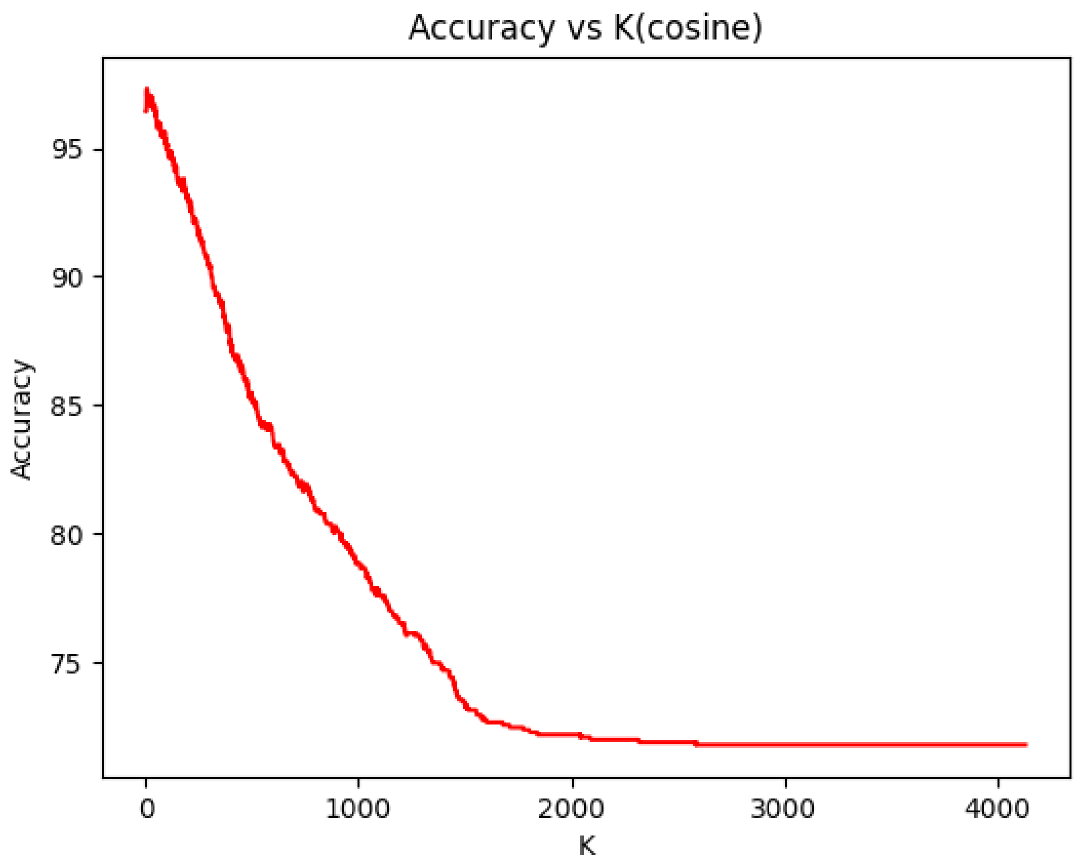
function *accuracy*

- it calls classify multiple times to find the classification for all test samples and calculates the accuracy.

<u>(The answers are given for a run of the program and since the splits are random the answer might vary on repeated executions.)</u>

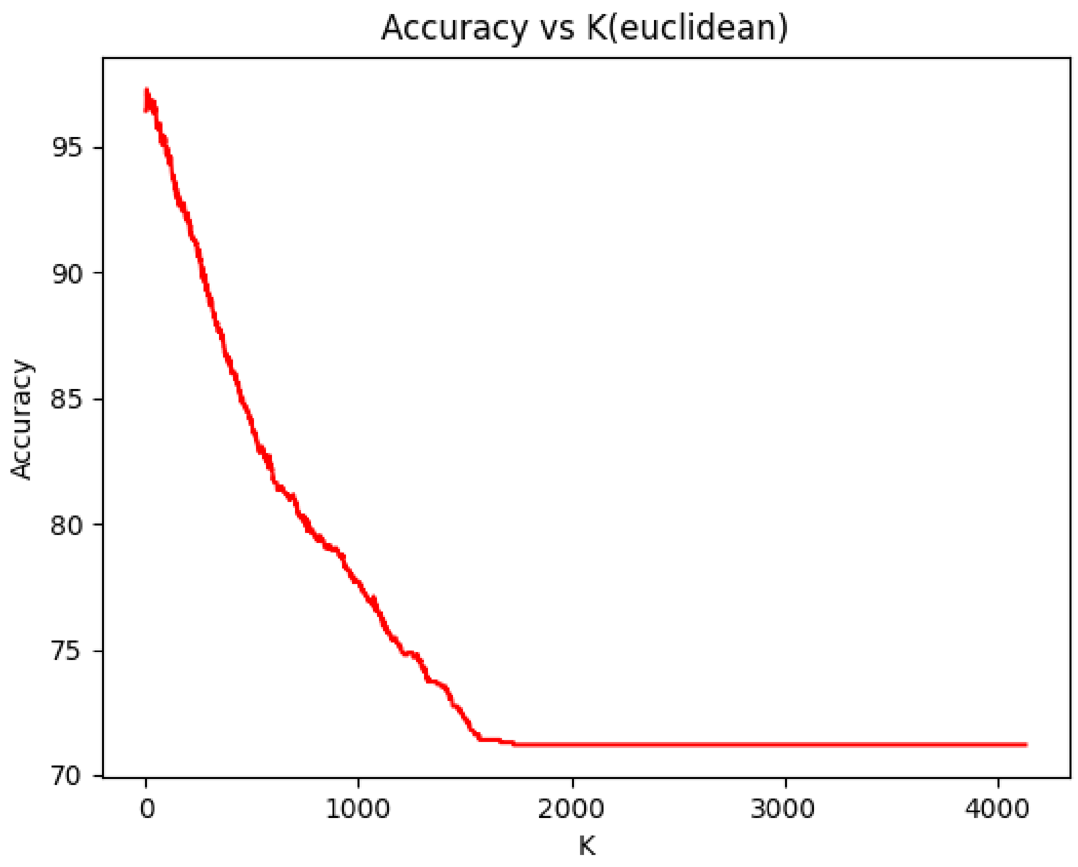
<u>Result Analysis:</u>

1. Cosine Similarity as a distance measure



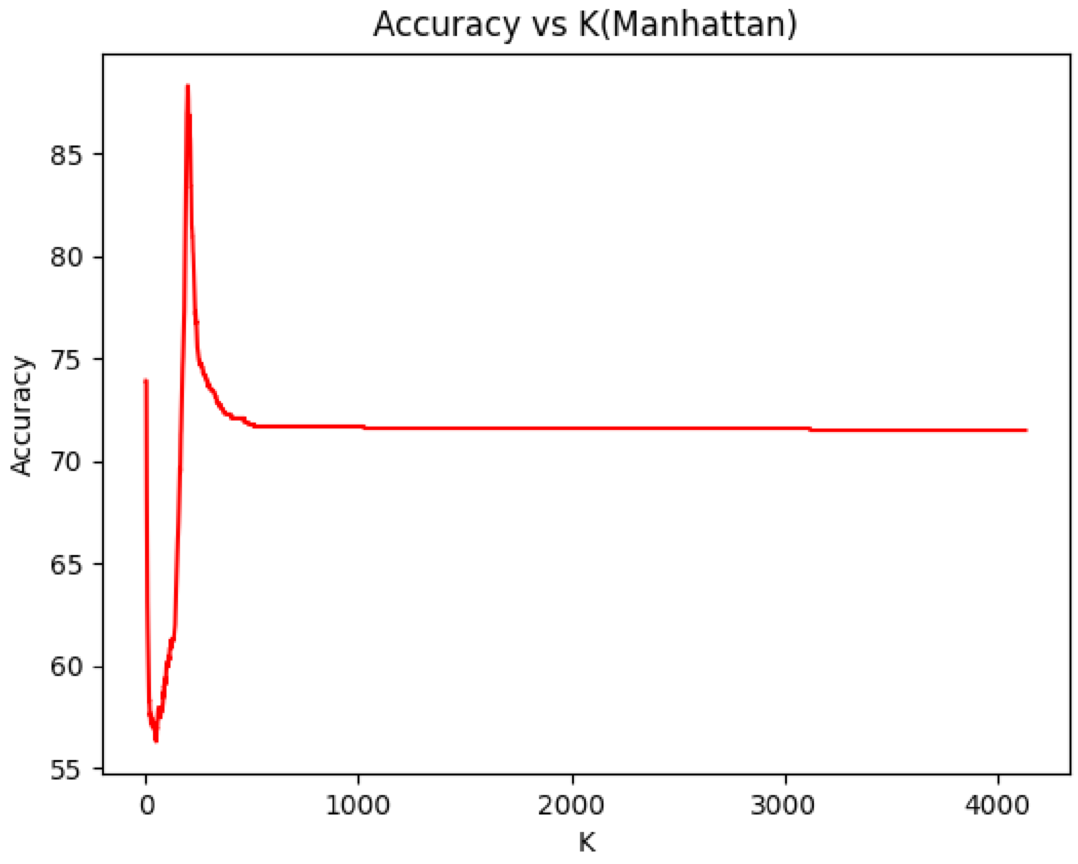
As we can see the method works really well for smaller values of k. The best result is obtained for k=3 which gives 96% accuracy on classification. As the value of k increases more neighbours are considered, since the number of spam examples are less, ultimately the ham examples overpower the classifier and lead to misclassification of all spam samples.

2. Euclidean Distance as a distance measure



In this method as well we see good accuracy for smaller values of k, The accuracy first increases slightly and then decreases with larger values of k. The best value of k is again 3 with accuracy of 97%. This model also suffers from similar problems with large values of k hence smaller values are optimal.

3. Manhattan Distance as a distance measure



The values in this case are more varied. The Manhattan distance shows large variance compared to other metrics. The best value of k comes to 197 with accuracy of 88%. Here we see a fluctuation in the values in the beginning because manhattan distance is not being able to measure feature distance properly.

Finally in all cases we see accuracy tends to 71% as k becomes large. This is because for large k all test samples are classified as 0(ham) hence accuracy is ratio of ham samples.

Distance Function	Best_K	Best_Accuracy
Cosine_Similarity	3	96
Euclidean	3	97
Manhattan	197	88

So the Euclidean distance comes out to be the best distance function for this particular scenario.