K-NN Algorithm
Using Spam Mail Dataset

Group F:
Abhinandan De(19CS10069)
Yashica Patodia(19CS10067)

1. Introduction

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. It assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. The algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

2. Experimental Observation

We have used the K-NN Algorithm for designing the classifier.The data used for training was provided in .csv format.a. We considered the split of 80:20 a Train and Test set respectively.We have tried implementing all these things mentioned in the previous section, in python with and without using in-built library files of python.

2.1. Step-Wise Description of Implementation of Major Functions

2.1.1. data_handling.py

This file handles our input dataset.It is also responsible for splits, preprocessing and vectorization.

1) **preprocess:** Here we make some minor changes in the email messages.
2) **vectorize:** Convert the text to a matrix form with floating point values using Tfidf vectorizer as it combines all the options of CountVectorizer and TfidfTransformer in a single model.We ignore english stop words to improve accuracy.
3) **gen_test_and_validation_set:** Generates training data and test data in the ratio 80:20.

2.1.2. distance_functions.py

This python file contains the major functions which contain all the distance and similarity functions will be put here

1) **cosine_similarity:** Returns the cosine similarity between two vectors X and Y. Here

$$d = 1 - cos\theta$$

$$cos\theta = \frac{X.Y}{|X||Y|}$$

.

2) **manhattan_distance:** Returns the manhattan distance between two vectors X and Y. Here

$$d = \sum_{i=1}^{n} |x_i - y_i|$$

3) **euclidian_distance:** Returns the euclidian distance between two vectors X and Y. Here

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

2.1.3. knn_model.py

This is the classifier model. First we initialize with the dataset. Then we find the nearest neighbors.After that we make predictions by varying k (no of neighbors).
We have used weighted K-NN where weights are inversely proportional to d^2, where d is the distance calculated between the vector instances.
After finding the k nearest neighbors, we simply take a weighted mean of our k nearest neighbors to classify our data.

$$p(x) = \frac{\sum_{i=1}^{k} w_i x_i}{\sum_{i=1}^{k} w_i}$$

Here x is our new instance, w_i and x_i represent the weight and classification of the i^{th} nearest neighbor.

1) **predict:** Here, we compute a 2-d array of predictions for all instances in X_test. Then we vary the number of neighbors and predict the class value. This leads to a better time complexity.
2) **compute_accuracy:** Here we compute accuracy values by varying no of neighbors from 1 to len(X_train)After that we return a list containing the accuracy values and the corresponding number of neighbors considered.

2.1.4. solve.py

This is the main file which contains the implementations of all the parts mentioned in the assignment.

2.1.5. Part 1.Construction of K-NN Model and Part 2. Calculating accuracy

Contained in the **run_model** method.This will run the model with the different difference/similarity measures.Then it will also plot the different accuracy values.

2.1.6. Part 3. Plotting the graphs

Contained in the **plot** method. This will be called from the run model and the vary_num_neighbours functions. This will plot and save in the output_files folder.

2.2. Detailed Data Analysis

In this section, we have shown the classification accuracy of the k-nn model for the following three distance metrics

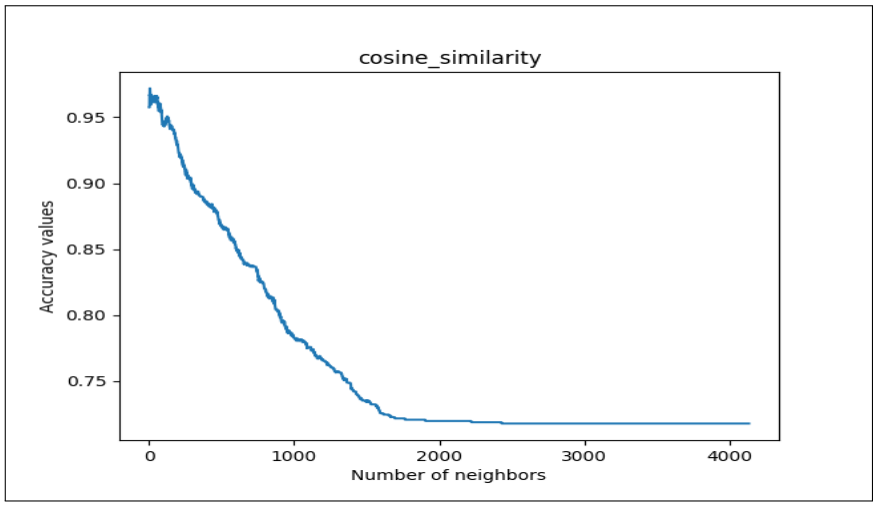
1) Cosine Similarity
2) Euclidean Distance
3) Manhattan Distance

1) **Data Analysis of K-NN model using Cosine Similarity:**

a) Following were the accuracy values observed for without using sklearn.

Accuracy for various values of K using Cosine	
Accuracy(%)	No of neighbours
0.967	1
0.975	**11**
0.944	101
0.924	201
0.898	301
0.884	401
0.866	501
0.850	601
0.837	701
0.818	801
0.799	901
0.782	1001
0.775	1101
0.765	1201
0.757	1301
0.743	1401
0.735	1501
0.725	1601
0.721	1701
0.720	1801
0.720	1901
0.719	2001
0.719	2101
0.719	2201
0.718	2301
0.718	2401
0.717	2501
0.717	2601
0.717	2701
0.717	2801
0.717	2901
0.717	3001
0.717	3101
0.717	3201
0.717	3301
0.717	3401
0.717	3501
0.717	3601
0.717	3701
0.717	3801
0.717	3901
0.717	4001
0.717	4101

b) No of neighbours v/s accuracy plot



The graph is decreasing.The accuracy of the classification decreases with increase in the number of neighbours and then it becomes constant for very large values of K.

Obtained best accuracy 97.48% for 11 neighbors

c) Following was the classification report (for the most accurate classification) generated using sklearn

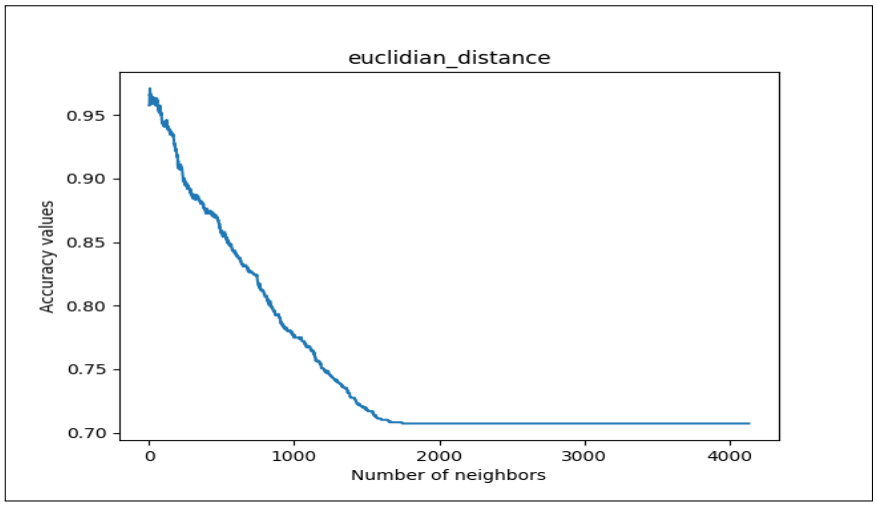
	precision	recall	f1-score	support
Ham	0.97	0.99	0.98	740
Spam	0.98	0.93	0.95	295
accuracy			0.97	1035
macro avg	0.98	0.96	0.97	1035
weighted avg	0.97	0.97	0.97	1035

2) **Data Analysis of K-NN model using Euclidean Distance:**

a) Following were the accuracy values observed for without using sklearn.

Accuracy for various values of K using Euclidean Distance	
Accuracy(%)	No of neighbours
0.957	1
0.974	**11**
0.942	101
0.908	201
0.885	301
0.873	401
0.857	501
0.839	601
0.827	701
0.806	801
0.790	901
0.775	1001
0.768	1101
0.750	1201
0.740	1301
0.727	1401
0.717	1501
0.710	1601
0.708	1701
0.707	1801
0.707	1901
0.707	2001
0.707	2101
0.707	2201
0.707	2301
0.707	2401
0.707	2501
0.707	2601
0.707	2701
0.707	2801
0.707	2901
0.707	3001
0.707	3101
0.707	3201
0.707	3301
0.707	3401
0.707	3501
0.707	3601
0.707	3701
0.707	3801
0.707	3901
0.707	4001
0.707	4101

b) No of neighbours v/s accuracy plot



The graph is decreasing.The accuracy of the classification decreases with increase in the number of neighbours.And then it becomes constant for very large values of K similar to the one with cosine similarity as the distance measure.

Obtained best accuracy 97.39% for 11 neighbors

c) Following was the classification report (for the most accurate classification) generated using sklearn

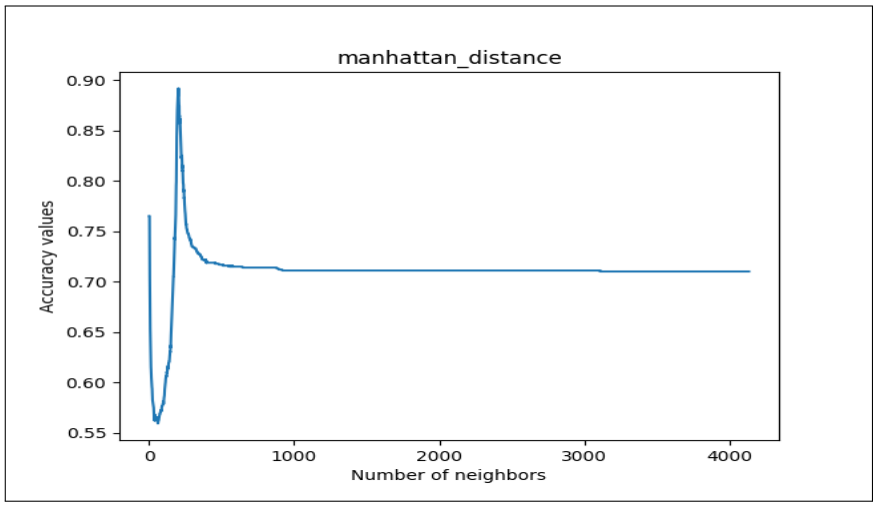
	precision	recall	f1-score	support
Ham	0.97	0.99	0.98	740
Spam	0.98	0.93	0.95	295
accuracy			0.97	1035
macro avg	0.98	0.96	0.97	1035
weighted avg	0.97	0.97	0.97	1035

3) **Data Analysis of K-NN model using Manhattan Distance:**

a) Following were the accuracy values observed for without using sklearn.

Accuracy for various values of K using Manhattan Distance	
Accuracy(%)	No of neighbours
0.765	1
0.582	101
0.890	201
0.915	**210**
0.734	301
0.718	401
0.716	501
0.714	601
0.714	701
0.714	801
0.712	901
0.711	1001
0.711	1101
0.711	1201
0.711	1301
0.711	1401
0.711	1501
0.711	1601
0.711	1701
0.711	1801
0.711	1901
0.711	2001
0.711	2101
0.711	2201
0.711	2301
0.711	2401
0.711	2501
0.711	2601
0.711	2701
0.711	2801
0.711	2901
0.711	3001
0.711	3101
0.710	3201
0.710	3301
0.710	3401
0.710	3501
0.710	3601
0.710	3701
0.710	3801
0.710	3901
0.7101	4001
0.7101	4101

b) No of neighbours v/s accuracy plot



The graph obtained using Manhattan Distance is peculiar as it first decrease steeply then it increases and attends a maxima of 91.49% at 210 neighbours then decreases and stabilizes for large values of K. The sharp decrease can be attributed to the bias over the training set which results in an overall decrease in accuracy in the initial phase.

Obtained best accuracy 91.49% for 210 neighbors

c) Following was the classification report (for the most accurate classification) generated using sklearn

	precision	recall	f1-score	support
Ham	0.97	0.90	0.93	740
Spam	0.79	0.92	0.85	295
accuracy			0.91	1035
macro avg	0.88	0.91	0.89	1035
weighted avg	0.92	0.91	0.91	1035

2.3. Steps to Run The Code

- Download the code into your local machine.
- Ensure all the necessary dependencies with required version and latest version of Python3 are available (verify with requirements.txt) **pip3 install -r requirements.txt**
- Go into the src directory
- Run the solve.py python code using the following command: **python3 solve.py**

3. Conclusion

After doing this assignment we understand that significance of value of K in K-NN algorithm. The following conclusions were drawn:

Maximum accuracy for the three distance functions	
Distance Function	Accuracy(%)
Cosine Similarity	97.48
Euclidean Distance	97.39
Manhattan Distance	91.49

1) The maximum accuracy is observed for cosine similarity with 97.48% and is the most optimal distance function for this dataset.
2) Euclidian distance also provided an almost identical plot with a maximum accuracy of 97.39%. The classification reports for both these distance measures are also identical
3) However, manhattan distance proved to be the least optimal classifier for the given dataset.
4) For all cases, the accuracy becomes almost constant at the end because when we increase the number of neighbors, the distribution of the nearest-neighbor set resembles the initial training set. The training set initially had approximately 70% ham mails and 30% spam mails. Since the ham mails become a majority, it becomes the predicted value for a large fraction of the test samples. Hence we are right approximately 70% of the time, which explains the stabilisation of the accuracy at the end for each distance function.

Acknowledgements

We are very grateful to Professor Jayanta Mukhopadhyay for giving us an opportunity to learn and apply the topics we have studied on some real-life problem. His guidance, support and teaching helped us a lot for making the skeleton of the project.