Research on K-nearest neighbor

Subhrajyoti Pradhan — Elijah Mallaby — Astrid Gamoneda

***Abstract*—Classification of objects is an important area of research and has possibilities of application in a variety of fields. Possessing knowledge of, underlying probabilities, Bayes decision theory gives optimal error rates, and in these cases where this information is not present, many algorithms make use of distance or similarity among samples as a means of classification. The K-nearest neighbor decision rule has often been used in these pattern recognition problems. In this project, we try to implement the k-nearest neighbor algorithm on a big data set on python.**

I. Introduction

THE Big Data is a very broad area of study working with large volume and complex data sets. One of the major problems we face while working on such huge amount of data, is classifying untagged/ unclassified data. Hence, in an attempt to understand this data, we employ the use of classification algorithms. These methods take unstructured datas and convert them into organized labeled datasets, such that users can access the required data easily. Classification algorithms primarily have two parts -

- A training step which provides training data
- A testing part which labels the unlabeled data.

While researching about various approaches to classifications, we came across some interesting methods such as the Naive Bayes and Decision Trees. However, we were fascinated by the k-nearest neighbors algorithms because of how simple, yet efficient it was when dealing with classifying large quantities of complex data[1]. K-nearest neighbors (k-NN) algorithm is a non-parametric method used for classification The input consists of the k closest training examples in the feature space. The output produced is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor [2].

An example dataset that classification may be applied to is the mnist dataset, which tries to identify hand drawn digits.

II. Experiment

We worked with two examples of a binary datasets - Amazon NASDAQ (Table.I) and stock and breast Cancer prediction (Table.II). We also used one example of a multilabel dataset which was a type of cancer prediction (Table.III). We ran our algorithm using a random 50/50 split of each dataset into training and seting sets, and generated confusion matrices from the results (displayed below).

III. The Implementation

This algorithm was implemented on python. The algorithm had several methods, a knn worker, knn and a testknn. In the knnWorker method we used a python built in called heapq which is used to find the n smallest elements of dataset.We find the k closest samples to p, the point we want to predict.We find the count of classifications among the k samples.We select the classification with the highest count. The knn method broadcasts the training dataset to all the worker nodes and maps the kNN worker over the testing dataset, passing in the broadcast training dataset. [3] The testKnn method Takes in a classified dataset. We split the dataset by the provided split value. We strip the classification out of the testing dataset. We run the kNN on the training and stripped testing dataset. Finally, we join the true classifications from the testing dataset with the computed classification from kNN.

TABLE I
Amazon Stock Data

Date	Open	High	Low	Last Close	Result
2002-01-03	11.13	11.94	11.05	11.9	up
2002-01-04	12.02	12.4	11.95	12.25	up
2002-01-07	12.08	12.51	12.08	12.34	up
2002-01-08	12.27	12.32	11.75	11.85	down
2002-01-09	11.96	12.17	11.3	11.53	down
2002-01-10	11.66	11.67	10.87	11.04	down
2002-01-11	11.03	11.34	10.93	11.03	down
2002-01-14	10.8	10.83	10.09	10.11	down
2002-01-15	10.34	10.42	10.16	10.29	up
2002-01-16	10.01	10.01	9.03	9.13	down
2002-01-17	9.82	9.85	9.4	9.74	up
2002-01-18	9.6	10.55	9.39	10.16	up
2002-01-22	12.74	12.79	12.14	12.6	up
2002-01-23	12.49	12.6	11.9	12.47	down
2002-01-24	12.75	14.06	12.71	14.01	up
2002-01-25	13.56	15.39	13.39	14.44	up
2002-01-28	14.86	15.77	14.76	15.5	up
2002-01-29	15.54	15.55	13.97	14.22	down
2002-01-30	13.98	14.25	13.2	13.9	down
2002-01-31	14.12	14.49	13.41	14.19	up

IV. Observation

For our binary datasets we saw reasonable performance- the vast majority of samples were classified correctly, as can be clearly in both of their confusion matrices. With the breast cancer data we discovered a bias towards benign tumors, which is perhaps for the best but does skew our results somewhat. With the multilabel data we were less successful, although the confusion matrix still shows good results in many categories.

V. Conclusion and Future Work

Altogether, for the simplicity of the implementation and lack of an explicit training step, kNN seems to be a good algorithm to have on hand for for generating quick predictions. However,

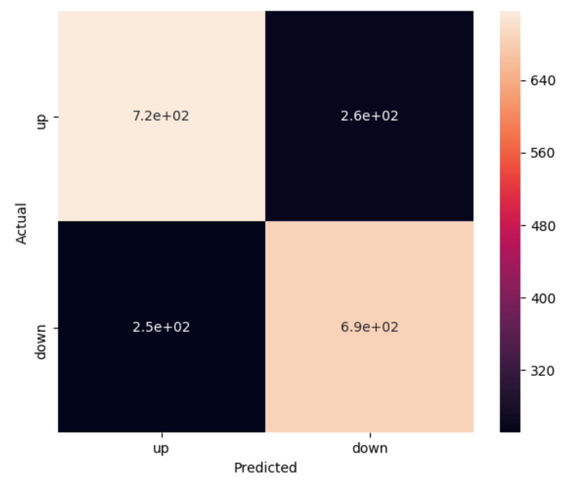

Fig. 1. Confusion Matrix for Amazon Stock Data

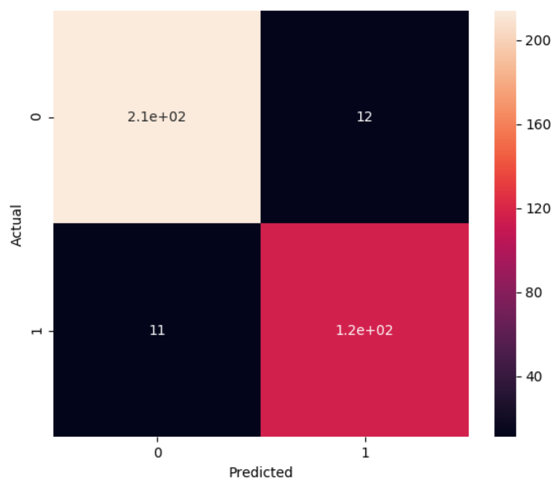

Fig. 2. Confusion Matrix of Breast Cancer Patient Data

TABLE II
BREAST CANCER MALIGNANCY DATA

ID	Clump Thickness	Cell Size	Type
158	1	2	0
499	1	1	0
396	1	1	0
155	5	5	1
321	1	1	0
212	1	1	0
234	3	2	0
289	6	6	1
300	4	10	1
356	3	3	1
672	1	1	0
328	10	3	1
199	1	1	0
78	1	1	0
598	1	1	0
569	10	8	1
446	1	1	0
506	10	10	1
626	6	6	1
603	4	6	1

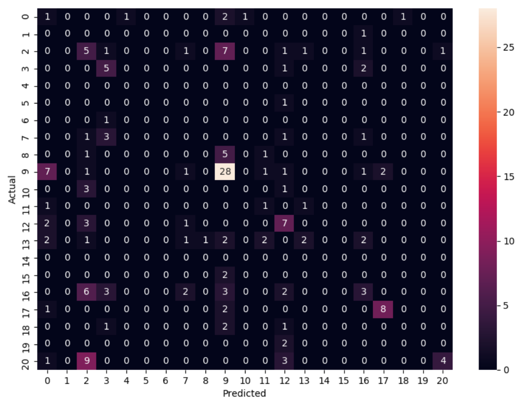

Fig. 3. Confusion Matrix of Tumor Data

we in the future having tests on much larger multilabel datasets would be valuable.

There are also some extensions to our algorithm which could improve its runtime and effectiveness. Firstly, preprocessing the training data into a tree datastructure by locality would allow us to reduce the load on individual nodes. A 2d example of such a datastructure would be a quadtree, which recursively breaks the sample space into quadrants. Making a couple assumptions about the density of quadrants would allow us to consult a much smaller section of the training data for samples when determining the k nearest neighbors. These sections could then be broadcast to separate nodes, limiting the amount of training data any one worker node needs to keep in memory. Another extension would be to add a fuzzy variation of kNN. Fuzzy kNN algorithms take into account the "typicalness" of the samples in the training dataset to help minimize the effect of outliers on results.

TABLE III
SAMPLE OF TUMOR DATA COLUMNS

primary class	age	sex	histologic type	degree of diffe	bone	marrow
lung	< 30	male	?	poorly	no	no
lung	< 30	male	?	poorly	no	no
lung	< 30	female	adeno	poorly	yes	no
lung	< 30	female	?	poorly	yes	no
lung	< 30	female	?	poorly	yes	no
lung	< 30	female	?	poorly	yes	no
lung	30-59	male	epidermoid	well	yes	no
lung	30-59	male	epidermoid	well	yes	no
lung	30-59	male	epidermoid	well	no	no
lung	30-59	male	epidermoid	fairly	yes	no
lung	30-59	male	epidermoid	poorly	yes	no
lung	30-59	male	epidermoid	poorly	yes	no
lung	30-59	male	epidermoid	poorly	yes	no
lung	30-59	male	epidermoid	poorly	yes	no
lung	30-59	male	epidermoid	poorly	no	no
lung	30-59	male	epidermoid	?	no	no
lung	30-59	male	epidermoid	?	no	no
lung	30-59	male	adeno	fairly	no	no
lung	30-59	male	adeno	poorly	yes	no

Acknowledgments

We would like to acknowledge our professor Dr. Zhang for her constant support and help. We faced many challenges while performing the project, but she guided us with meaningful insights which helped us achieve better results.

References

[1] Z. Yamak, “Modern machine learning algorithms: Strengths and weaknesses,” *elitedatascience.com/machine-learning-algorithms*, 2018.

[2] N. Castle, “Regression vs. classification algorithms. oracle + datascience.com,” in *www.datascience.com/blog/regression-and-classification-machine-learning-algorithms.* IEEE, 2016, pp. 55–62.

[3] R. Gandhi, “Introduction to machine learning algorithms: Logistic regression,” *Hacker Noon, Hacker Noon*, 2018.