Contents

Project 2: K Nearest Neighbor

Jaudat Raza JAUDAT.RAZA@JHU.EDU
Master of Computer Science
John Hopkins University
Baltimore, MD 21218

Editor: Jaudat Raza

Abstract

This document will cover the implementation process of K Nearest Neighbor, which is a non-Parametric algorithm. The resource used will be linked below in the resource section of the document. The document will follow a simple pattern for which a less than 5-minute video will be submitted as well. The Jupyter Notebook starts with getting the data into Pandas Data from the link provided in the data source. After that, first the data will be processed through K Nearest Neighbor after checking best K Value. Then it will include Edited K Nearest Neighbor and Condensed Nearest Neighbor to show the difference of how KNN is impacted if the data is either edited or Condensed, which will be explained further in the document.

1 Problem Statement

The purpose of this assignment is to give you a chance to get some hands-on experience implementing a nonparametric algorithm to perform classification and regression. We will learn that using KNN classifier and regressor. We will run the data from the UCI data set to see the result of the algorithm and then implement condensed and edited KNN to see how it has impacted to the result time and accuracy wise.

2 Algorithm Studied

In this project we are covering K Nearest Neighbor and two subcategories of the method, which are Edited K Nearest Neighbor and Condensed K Nearest Neighbor.

2.1 K Nearest Neighbor

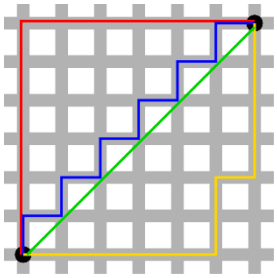
KNN is a non-parametric algorithm, so it does not make any strong assumption about the type of form it is dealing with, which lead to less restriction on the algorithm.

KNN is a very simple technique. In simple words, all KNN algorithm does is find the closest data points to the point provided. After finding the neighbors around a data point, the algorithm returns the classification that is found the most in the neighbors. There are some basic steps, which included find the Neighbor and then making the prediction.

2.1.1 Finding Neighbor

Finding Neighbor could mean a lot of thing and there are a lot of ways of doing it. First step is to realize what is the goal. Do you want to find the closes neighbor from distance perspective or want to find a neighbor that is already in a similar field as you? For this project, we are only dealing with numbers so in the implementation we use distance to find our neighbor. Then, the next step is analysis the distance you are looking from in the data set compare to what you are giving. There is Manhattan Distance, Euclidian Distance, and max-coordinate distance.

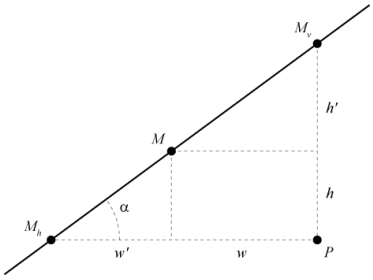
2.1.1.1 Manhattan Distance



Manhattan Distance is calculated using the equation below. It is Minkowski method L1 norm. Instead of going in a straight line. It has to take the possibility of the path allowed as well as shown in the picture above.

$$f(x) = |x_1 - x_2| + |y_1 - y_2|$$

2.1.1.2 Euclidian Distance



Euclidian Distance is calculated using the equation below. It is Minkowski method L2 norm. This is what we are using in our implementation to find neighbors

$$f(x) = \sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$$

2.1.2 Make Prediction

After you have picked a way to find your neighbors. The first thing to consider is how many neighbors you want to analyze to make prediction. Depending on the number of neighbors you pick, it will impact your prediction. Let say that there are 9 data points in your set and 2 of them are way to far apart, but you chose to find 9 neighbor, then you prediction would take the border cases into consideration and decrease your accuracy. On the other side, if you pick not enough, then your prediction might start looking at class right next to you but is less distance from you. For my implementation what I have done is that I run my prediction with 2 to 10 as K and see which would return the best accuracy rate. Then, that is the K used for rest of the prediction.

So now we know the K we want, then we use Euclidian Distance to find K number of neighbor and the class we see the most in our neighbors we pick that as our prediction. We train on 5 data sets and the result of each is added and return the mean to show the mean accuracy.

2.2 Edited K Nearest Neighbor

Edited KNN is sort of a subcategory to KNN. Basically, the goal is removing the data from the data set that is not helping make the prediction or improving the accuracy rate. It seems bad that in

general more data means more data to learn from, but then if the data is hurting more then helping, this is where it is good to remove the unhelpful data. Show below are two ways of doing it.

1) First, for every example, we will classify that example using the rest of the data. Then we will compare the classification returned to the label associated with that point. If the classes do not agree, then we will remove the point from our training set. And we will repeat this process as long as performance on a separate validation data set continues to improve or until a no further points are removed.
2) First, again, for every data point in our data set, we classify using the rest of the data, Compare the classes, and this time, if they agree, we will remove the point from X.

2.3 Condensed K Nearest Neighbor

Condensed KNN is sort of a subcategory to KNN. Basically, the goal here is same as Edited, to provide a data set to the algorithm that does not include noise and error. Not have the data points that hurt more then they help. We do this by instead of removing data set. We create a brand-new Data, let us call it Z, we only include the data that will try to decrease the error.

The basic approach to finding Z is through a greedy search process as follows. First, initialize Z to empty and then we enter a loop. Now we consider each the point on our data set in random order. And we will find a point within our set Z that is the minimum distance to each point. If the classes do not agree then we will add that point to Z

3 Tuning

This is the section that describes all the data and how it was updated to make it work with the Algorithm. This will be described in the video as well.

3.1 Abalone

1) The only change made in the data was the “Sex” Column. The infants were changed to 1, Female were changed to 2, and Male were changed to 3.

3.2 Glass

1) No Data Change just made sure it was processed correctly.

3.3 Forest Fires

1) The Month column was updated from 1 to 12, where 1 is January and 12 was December.
2) The Day column was updated from 1 to 7, where 1 was Monday and then 7 was Sunday.

3.4 Segmentation

1) The column name was updated
2) Dead elements were dropped out of the data

3.5 House Vote

1) The Missing Data was replaced with 2, which show no vote given for the class
2) One Hot Encoding on Class for demo and Republican
3) Rest of the attributes y is 1 and no was 0

3.6 Machine

1) Model column in the data is serving the unique identifier code purpose, so all of them were updated from their name to integer from 1 to 208.
2) Vendor column had 29 models in it, so they were changed to 1 to 29.
3) Vendor column was set as the last column of the data.

4 Results

The following result are from one instance of running the whole script at one time. The percent of accuracy during transferring to the documentation was only kept accurate till the decimal point. The time was also running in that instance to show the time it took to execute the full KNN with different data sets. It also shows the how much data was dropped, where lower time would mean less data to process. For all the Charts X label is percent and Y Label is the fold. All the data set were run on 5 Folds. Listed below are the Best and the Worst K for the related data, which were generated using the BestK and WorstK function.

Data	Best K	Worst K
Glass	2	8
Fire	9	1
Abalone	5	1
Vote	4	2
Machine	1	6
Image Segmentation	4	6

4.1 Abalone

4.1.1 KNN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
23.83%	21.55%	22.39%	23.26%	28.02%	23.83%	93.62s

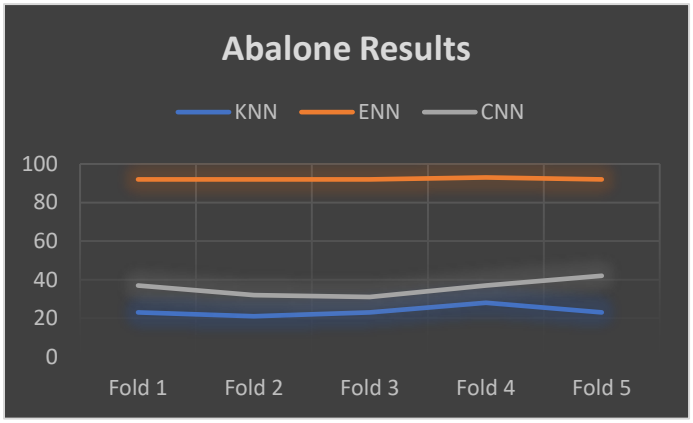
4.1.2 ENN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
92.13%	92.71%	92.13%	93.63%	92.45%	92.62%	35.27s

4.1.3 CNN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
37.13%	32.71%	31.13%	37.26%	42.02%	35.83%	67.62s

4.1.4 Chart



4.2 Glass

4.2.1 KNN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
95.24%	100.0%	100.0%	92.86%	100.0%	97.62%	0.28s

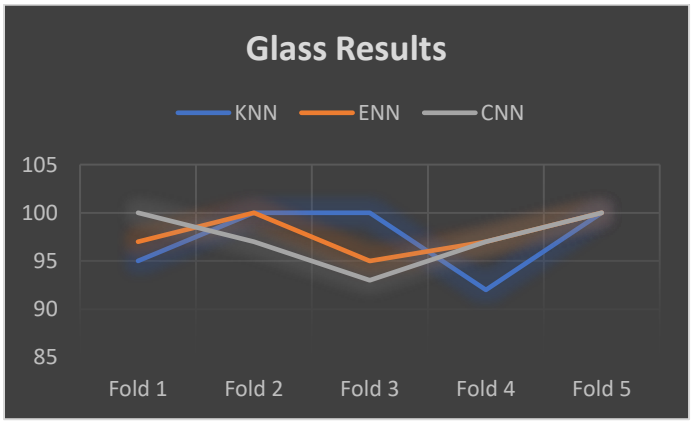
4.2.2 ENN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
97.62%	100.0%	95.23%	97.62%	100.0%	98.05%	0.22s

4.2.3 CNN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
100.0%	97.62%	93.23%	97.62%	100.0%	98.03%	0.22s

4.2.4 Chart



4.3 Forest Fires

4.3.1 KNN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
52.43%	55.34%	38.83%	43.69%	42.72%	46.60%	1.85s

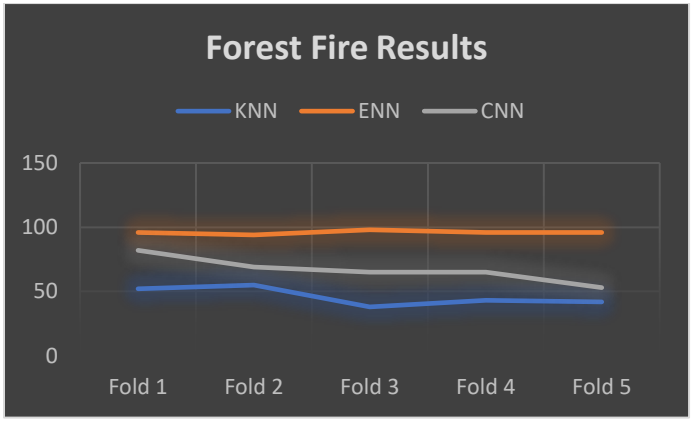
4.3.2 ENN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
96.07%	94.11%	98.04%	96.08%	96.07%	96.07%	0.23

4.3.3 CNN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
82.07%	69.11%	65.04%	65.08%	53.07%	68.07%	0.33

4.3.4 Chart



4.4 Segmentation

4.4.1 KNN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
2.44%	0.0%	0.0%	7.32%	7.32%	3.41%	0.35s

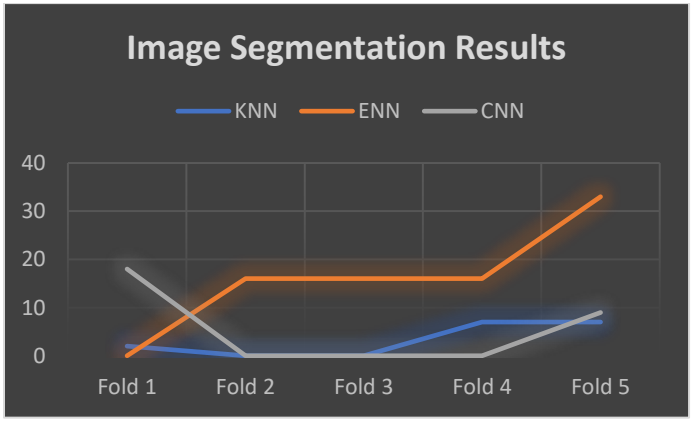
4.4.2 ENN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
0.0%	16.67%	16.67%	16.67%	33.34%	16.67%	0.21

4.4.3 CNN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
18.0%	0.0%	0.0%	0.0%	9.34%	5.47%	0.33

4.4.4 Chart



4.5 House Vote

4.5.1 KNN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
93.02%	94.19%	90.70%	91.86%	91.86%	92.32%	1.70s

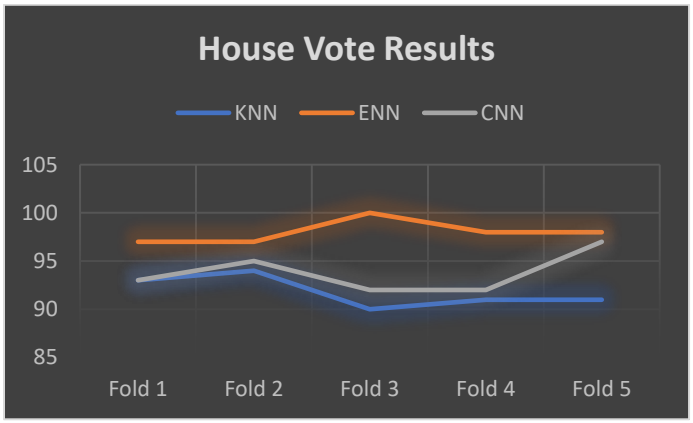
4.5.2 ENN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
97.53%	97.53%	100.0%	98.76%	98.76%	98.52%	1.53

4.5.3 CNN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
93.53%	95.53%	92.0%	92.76%	97.76%	94.52%	1.34

4.5.4 Chart



4.6 Machine

4.6.1 KNN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
41.46%	60.98%	34.14%	46.34%	56.10%	47.80	0.25s

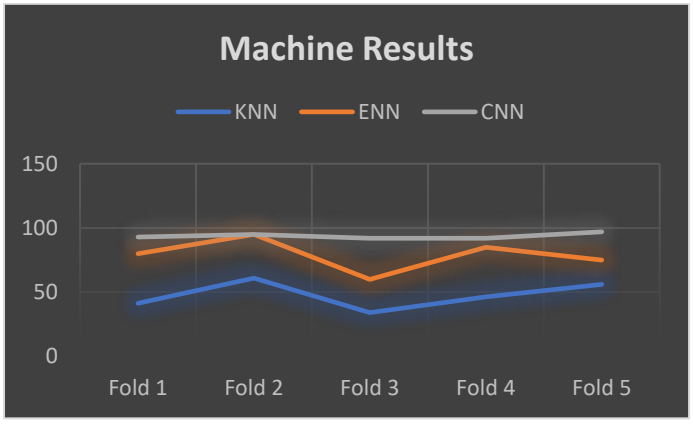
4.6.2 ENN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
80.0%	95.0%	60.0%	85.0%	75.0%	79.0%	0.06s

4.6.3 CNN

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Avg	Time
93.0%	95.0%	92.0%	92.0%	97.0%	94.3%	0.08s

4.6.4 Chart



5 Conclusion

5.1 Abalone

Abalone Best K for finding the best accuracy was 5, which was unexpected. Knowing the size and the number of features in the data, it was expected to see less neighbors into consideration return better result. There are four weight feature that are very closely related, which could cause confusion picking up the neighbor. Using the regular KNN with the given data, the algorithm achieved 23.83%. Then after going through the data and created Edited data from ENN, the result went from 23.83% to 92.3%. This was a great improvement. But as we see that the time in general dropped a lot as well, which shows that ENN and CNN dropped a lot of data which was present in the original set.

5.2 Glass

Glass Data was very well sorted, so I guess that really did help with making prediction. The Best K was 2, which also made sense because it only wanted to look at some thing very close to it to make the prediction, because it was already sorted for all the features. The 97% accuracy on KNN was already very good, but ENN made sure that it performed better then KNN every single time. For example, in the result provided from the instance in the report. Glass KNN return 97% accuracy so the ENN return 98% or above as the result. As expected, form the result that the time also dropped each time as well in ENN compare to KNN

5.3 Forest Fire

Forest Fire needed some data cleanup before getting the result. Removing Days and month impacted the accuracy in negative manner. Adding them back in the data were proven to be helpful after hot encoded to their relative integer. Forest Fire really took benefit of ENN. There were a lot of data in it that was leading to the same result, which was not helping with classification. The best K for Forest Fire was 9, which kind of made sense because during data analysis, I did notice that the standard deviation of all the features was very small except DMC and DC feature. The results show that the accuracy from KNN to ENN was doubled. KNN 46% to almost 96%.

5.4 Machine

Machine Data Set was one of the unique data set where the Condensed Nearest Neighbor helped more then Edited Nearest Neighbor. For all 5 folds it performed better. This data improved the most from ENN and CNN from KNN. Where I think the model and vendor caused the best k to be 1, which really limited it, I think. In ENN after dropping some of the copy vendors with different Models really helped the data set. It started with 208 instances, but the ENN only had 74 in it by the time it was done dropping any more instances. As we can see from the results that KNN was 47% where CNN was 94.3%

5.5 Image Segmentation

This was just hard to learn because of so many features in the data. I think most of them did not help. Just condensing and editing the did not really help. I think with more domain knowledge. This would have required some of the features to be dropped as well. In the model , during testing, I tried dropping all the wrong prediction column, which is the opposite of what I did for the other data set, It let to zero remaining data point after the 3 round of filtering. So, it showed that even after the first or second round of filtering the prediction was not improving leading me to believe that this require feature updates.

5.6 House Vote

House Vote did very well across all run. The CNN and ENN did perform better every time with no doubt, but with initial run of the data KNN was returning very low number below 30%, when I was removing all the time people did not vote. So, I decided to keep that information and marked the for 2. So, each time vote was not given, the data marked it as 2 which meant undecided. That improved the results from low 30s to the 90s, which are recorded in the result section. This led me to play a bit with other data set as well. Where I thought may be sorting will help the data algo run better, which it did not.

5.7 Overall

What we learned from this project is that implementing KNN is not as complicated. Just knowing what you are expecting from your actions is step 1. Also, for this data set, we only used Euclidian Distance. It would be kind of interesting to see how other distance calculation would have worked with the algorithm. There are two main steps of algo which are to find neighbors and then just make prediction based on similarity. CNN and ENN are very good example to show how more data is not always good. Sometime trimming extra bad data might be good. We see the result all across the board, that ENN and CNN performed better, and they also decreased time, which was expected because the data process was less then what it started with after trimming the waste.

6 Resources

- “K Nearest Neighbor.” *K Nearest Neighbor - an Overview | ScienceDirect Topics*, 2020, www.sciencedirect.com/topics/biochemistry-genetics-and-molecular-biology/k-nearest-neighbor.
- Black, Paul E. *Manhattan Distance*, 11 Feb. 2019, xlinux.nist.gov/dads/HTML/manhattanDistance.html.
- pcp21599Check out this Author's contributed articles., et al. “Measures of Distance in Data Mining.” *GeeksforGeeks*, 3 Feb. 2020, www.geeksforgeeks.org/measures-of-distance-in-data-mining/.