CSCI 4144

Using KNN Algorithm to Detect Parkinson Disease

Group 27

Prepared for April 4, 2022

Ryan McInroy B00798564

Justin Timmins B00795685

1. Introduction

Parkinson's Disease is a nervous system brain disorder that negatively affects one's muscles and body movements. Symptoms of the disease include shaking, stiffness, unbalancedness, difficulty walking, and more. Parkinson's is a very hard disease to diagnose, especially in the early stages. Parkinson's is best treated in the early stages as early treatment can reduce and relieve effects of symptoms. Researchers and scientists are still unable to determine what causes someone to develop Parkinson's disease and how it can be detected early. Many different studies and papers have been published to highlight different findings in an effort to answer these questions. The problem we aim to solve is to be able to train a model to be capable of identifying if a person has Parkinson's Disease based on the given data/symptoms.

The experiment conducted within this paper is an implementation of an experiment conducted by Ma, Yang, and Cheng (2013) in "How the Parameters of K-nearest Neighbor Algorithm Impact on the Best Classification Accuracy: In Case of Parkinson Dataset". The steps of the original experiment will be followed in order to produce similar results. Results will then be able to be compared between the 2 separate experiments, allowing for trends and findings to then be discussed.

2. Algorithm

2.1 Optimal K

The KNN algorithm, which stands for k-nearest neighbor, has been used in studies similar to our own as it is a simple and efficient classification algorithm. The KNN algorithm can be given a point, and then by using a data set, categorize the given point by looking at categories of points nearby. KNN makes use of a K value which is set by the user or generated by a program. The K value is an integer and represents a count of the number of nearest neighbors a data point can use to classify itself. It is up to the user or program to determine what the K value should be, but this is not a straightforward task as different K values yield different classification accuracies.

2.2 Distance Metric

Along with finding the optimal K value, we also aim to explore the effects the distance metric and data normalization methods have on the accuracy of our models. The 2 distance metrics we will experiment with are the Euclidean distance and the Manhattan distance. These metrics are used to calculate the distance between 2 points.

Euclidean distance is quite simple as it is the direct length between 2 points.:

$$d(x, y) = \sqrt{\sum_{i=1}^{22} (x_i - y_i)^2}$$

Manhattan distance is the distance between 2 points measured along the axes' using right angles only.

$$d(x, y) = \sum_{i=1}^{22} |x_i - y_i|$$

These measures are used to calculate the distances of neighboring points. Resulting accuracies will vary between the 2 distance metrics which is why we aim to experiment to find the one which yields the most accurate results.

2.3 Normalization Method

The accuracy of the KNN model also depends on how balanced the data is. Some attributes in the set are calculated in Hz while others are just a percentage. When computing distance, we don't want 1

attribute to have a greater influence than another just because it is computed using a different scale. To solve this issue, we can normalize our data which generally involves scaling all values down to a specific range of numbers. We will explore a variety of normalization methods, such as Min-Max, Z-score, absolute max normalization, and normalization by decimal scaling. Details of these normalization methods are listed below.

Min-max normalization: Linear transformation of the raw data. Every value gets transformed into a decimal value between 0 and 1.

$$v' = \frac{v_{max} - v_{min}}{v \quad - v_{min}}$$

Z-score normalization: values are normalized using the mean and standard. The mean of all values is 0 and the standard deviation is 1.

$$v' = \frac{v - v_{avg}}{v_{std}}$$

Max normalization: values are normalized by dividing value V by the maximum attribute Vmax.

$$v' = \frac{v}{v_{max}}$$

Normalization by decimal scaling: normalizes values by moving their decimal points. The number of decimal points to move will depend on the maximum absolute value.

$$v' = \frac{v}{10^j}$$

The various data normalization methods will affect our models accuracy in different ways so we seek to find the best method for our data set.

2.4 Percentage of Data Used for Training

On top of these factors, our model's accuracy will also depend on how the data set is split, as the KNN algorithm utilizes a labeled data set that gets split into 2 groups. The first group is training data which is used to train our model. The second group of data consists of testing data which we will use to verify how accurate our model that was trained with the training data is. In this paper, we will discuss how the KNN algorithm can be implemented, and how results can change depending on the parameter of the model, data normalization method used, and percentage of data used for training.

2.4 Our Implementation

The particular algorithm we will implement first involves normalizing our data set with 4 different methods then storing our normalized data sets along with our raw data to be used later. Next, starting at using 5% of the data to train our model, we will generate a K Nearest Neighbor Classifier(KNN) model with a number of different parameters. For each KNN model we generate, we must first find the k value that performs best on the given set of training data and use that value as k in our model. The first parameter we will explore is the distance metric to use for our KNN model. We will generate a KNN model using the Euclidean distance metric, and then using the Manhattan distance metric. For each distance metric, we will test the KNN model on our 5 different data sets(Raw Data, Min Max Normalized Data, Z-Score Normalized Data, Absolute Max Normalized Data, and Decimal Normalized Data). Then, for each data set we use, we will generate 100 models and take the average accuracy, standard deviation and most common k value used from all 100 models. We then displayed the accuracy, standard deviation and most used k value for each distance metric, on each data set while using

5% of our data to train the model. After this we will increase our training % by 5% and then carry out the same process outlined above for 10% training data. This process continues until we reach 95% training data and have completed our table.

3. Data Preparation and Algorithm Implementation

3.1 Data Preparation

I was able to download the Parkinsons.csv data set from the UCI machine learning repository at https://archive.ics.uci.edu/ml/datasets/parkinsons. The data in the set consists of different biomedical voice measurements from 31 people, where 23 of them are diagnosed with Parkinson's Disease. There are 195 tuples in the data set with each representing a voice recording from a particular individual. Each tuple consists of 23 attribute's which each represents a particular voice measure excluding 1 attribute for the participants name and 1 attribute for the participants health status. Namely, the health status attribute is 1 if the person is diagnosed with Parkinson's Disease and 0 if the person is not diagnosed with Parkinson's Disease.

When processing the data, first remove the name of each participant as it is irrelevant because we only care if the person has Parkinson's Disease or not, not who they are. On top of this, the name attribute is the only one which does not involve a numerical measure so dropping it from the dataset will make the dataset easier to work with. Next I assigned a variable "X" to represent the entire data set excluding the "status" attribute as it represents our class labels for this problem. Finally, I assigned variable "y" to hold only the "status" attribute for each row of data, which will represent our class labels.

After reading in the Parkinsons data and assigning "X" to the data and "y" to the class labels, the data was normalized using 4 different methods. I used the X variable to normalize my data using min max normalization, z-score normalization, absolute max normalization and decimal normalization. All normalized data sets along with the original raw data are stored in an array to be tested with later, which marks the end of data preparation for the algorithm.

3.2 Algorithm Implementation

When implementing the algorithm, we chose to use python as we were most comfortable with it and provided simple methods for data manipulation. The program is mainly built from 2 primary functions knn(X, y, metric, trainingPer) and repeatKnn(X, y, metric, trainingPer).

The knn(X, y, metric, trainingPer) function and repeatKnn(X, y, metric, trainingPer) take in:

X: The data which is used to predict if someone has Parkinson's Disease

y: The data which has the class labels for each data sample in X,

metric: An int(1 or 2) which represents the distance metric used in the K nearest neighbors classifier

trainingPer: A decimal value used to represent the percentage of data used to train the model.

The knn() function is used to compute and return the accuracy of a K Nearest Neighbor classifier, as well as the optimal k value used for the model. It first splits the data into training and testing data based on the inputted training data percentage. The knn() function then computes a temporary K Nearest Neighbors classifier and runs a gridSearchCV with 5 fold cross evaluation on the n_neighbors parameter to find the optimal k value to use for the training data at hand. That optimal k value is then used to generate a new K Nearest Neighbors classifier where the accuracy of that classifier is returned along with the k value used.

The repeatKnn() function is used to generate the knn() function 100 times and return the average accuracy, standard deviation and most common k values used from all the models generated. Since the

function repeatKnn() can cause the program runtime to be very large, I reduced the function to only generate knn() 5 times as it allows the program to execute in a reasonable amount of time, but at the cost of less accurate accuracy values. In order to uphold the integrity of the program, the table found in the experimental results section below was computed by generating knn() 100 times inside of repeatKnn() so we can correctly compare results. However, I changed it back with the submitted program so it can be executed in a reasonable amount of time. The repeatKnn() function is crucial to our program as it strengthens the results we compute for each training percentage used. If we are computing knn() with 10% of data used for training, our algorithm will take a random 10% of the data and mold the model to work best for that 10% of data. That 10% may represent the data as a whole well or it may be full of outliers and be very different from the rest of the data. This is the rationale for repeating knn 100 times and using the average of those 100 times as it will give the best representation of what taking any 10% of the data to train a model will look like, not just 1 particular case of taking a random 10% of samples to train the model.

To bring it all together, the program uses a nested loop to display the table below. The outer loop iterates through each training percentage going from 5% to 95% and increases by 5% for each iteration. The inner loop iterates through each distance metric to use for the model, consisting of the Manhattan distance metric and the Euclidean distance metric. Inside each iteration of the inner loop, repeatKnn() will be executed for each data set including the raw data, min max normalized data, z-score normalized data, absolute max normalized data and decimal normalized data. For each data set, the average accuracy, standard deviation and most common k value will be printed to the table.

Experimental Results

When looking at the results of the algorithm, Table 1, many observations can be made about how factors like the k value, distance metric, data normalization method and percentage of data used for training affects our models accuracy. More so, how "good" our model is at determining if someone has Parkinson's Disease based on a voice recording.

In terms of the k values effect on our models accuracy, it's clear that the majority of models found k=1 to be the optimal k value and produce the best results. This means that our model is able to classify a sample by simply looking at its single closest neighbor and using its label. Because the data set we are working with has multiple recordings of the same people, there is a good chance that there are at least 2 recordings of the same people in each training set(at least as the training sets get larger) and knn can therefore use the single closest neighbor to classify a sample as it most likely comes from the same person. I suspect this is the reason that k values tend to be larger when with smaller training sets as there is a greater chance that there is only 1 recording a person in the set and the algorithm must consider more neighbors to classify a sample. On the contrary, k values also seem to increase as training sets grow very large which we suspect is due to other the fact that the more people you have in a dataset, the more people you have that are similar to each other and the algorithm must then increase the number of neighbors to account for and give less influence to outliers.

The distance metric seems to affect our results primarily based on the data normalization method. It seems that the Euclidean distance metric produces slightly better accuracies on the raw data and decimal normalized data. While the Manhattan distance metric seems to outperform euclidian on the min max, z-score and absolute max normalized data sets. On top of this, it can be observed that Euclidean distance models have a much higher chance of having a k value other than 1 compared to manhattan.

Every data normalization method seems to improve models accuracy, but certain normalization methods seem to perform much better than others with the Parkinsons data set. In particular, the min max, z-score and absolute max normalization methods seem to perform the best and were the only data sets able to achieve an average accuracy higher than 90%, with most of them hitting 90% at 50% training data. The min max normalization method is evidently the strongest normalization method to use for the parkinsons data set.

Finally, it is very clear that as the model uses more data for training, it becomes more accurate. This is an obvious observation but also a very important one as it is arguably the most important factor. Every single data set, distance metric and k value will perform better when a higher percentage of data is used to train the model as it exposes the model to more situations. For example, if the model uses 5% training data and all 5% came from 1 person with Parkinson's disease. The model will assume that someone with Parkinsons must sound exactly like the person in the training set. This is terrible training for our model, so having a higher training percentage allows the model to look at the similarities between people with Parkinsons instead and therefore become more accurate when predicting if someone has parkinsons that isn't in the data set.

I would first like to point out that although our results are similar to the results observed in the research paper, they can almost never be the exact same, even when using the same data set as the exact data samples being allocated to training is completely random, even when the training percentage is the same. However, when comparing our results to the results achieved in the research paper, many similarities can be observed. The first being that k values in the research paper, are also peaking when training percentages are extremely low and extremely high. Along with this, distance metrics seem to affect our results in the same way as the research paper results. Our normalization methods perform similarly as min max is still the top performing method in the research paper, however decimal normalization seems to perform much better in the research papers results compared to ours. Obviously as training data percentage increases so does model accuracy. And Finally, our results were able to determine the same best performing model as the research papers. The best performing model uses min max data normalization with the euclidean distance metric, with 95% of data allocated to training the model and a k value of 1. Although our exact values differ(research papers being 96.73 while ours is 97.0) this is expected as our models most likely wouldn't have pulled the exact same 95% of data for training. It provided us with the information we sought after which is to determine what factors to use to generate the most accurate model.

Training(%)	Metric	Raw Data	Min-Max	Z-Score	Max	Decimal
5.0	Euclidian	71.46+-9.55(3)	76.16+-6.13(1)	75.61+-5.07(1)	76.46+-6.06(1)	72.48+-9.27(3)
	Manhattan	73.72+-6.59(1)	74.70+-7.22(1)	75.82+-4.67(1)	76.19+-5.31(1)	73.35+-8.79(1)
10.0	Euclidian	74.85+-5.17(3)	78.43+-4.92(1)	79.76+-4.55(1)	79.73+-4.58(1)	73.66+-5.63(1)
	Manhattan	75.64+-5.05(1)	79.99+-4.57(1)	79.39+-4.73(1)	80.45+-4.40(1)	75.68+-5.10(1)
15.0	Euclidian	76.61+-3.98(1)	82.37+-4.85(1)	82.26+-4.42(1)	81.15+-4.34(1)	76.50+-3.83(1)
	Manhattan	77.75+-4.44(1)	82.09+-4.73(1)	82.89+-4.63(1)	82.15+-4.46(1)	77.96+-4.07(1)
20.0	Euclidian	78.35+-3.66(3)	84.64+-3.90(1)	84.32+-3.86(1)	83.89+-3.46(1)	77.93+-4.59(1)
	Manhattan	78.60+-3.47(1)	85.36+-3.82(1)	84.36+-4.15(1)	84.19+-4.11(1)	78.62+-3.50(1)
25.0	Euclidian	78.60+-3.08(1)	85.76+-4.61(1)	85.22+-4.41(1)	85.00+-3.62(1)	78.57+-3.68(3)
	Manhattan	79.65+-3.07(1)	86.65+-3.34(1)	86.12+-3.18(1)	85.57+-3.49(1)	79.86+-3.13(1)
30.0	Euclidian	79.72+-3.52(1)	87.34+-3.73(1)	86.77+-3.77(1)	86.53+-4.12(1)	80.24+-2.96(1)
	Manhattan	79.82+-3.15(1)	87.51+-3.78(1)	87.23+-3.77(1)	87.47+-3.24(1)	80.09+-3.29(1)
35.0	Euclidian	80.43+-3.06(1)	89.02+-3.48(1)	88.02+-3.05(1)	87.66+-3.73(1)	79.79+-3.20(1)
	Manhattan	81.23+-2.68(1)	88.61+-3.36(1)	88.61+-3.36(1)	87.54+-3.80(1)	81.22+-2.68(1)
40.0	Euclidian	80.74+-3.23(1)	89.64+-3.93(1)	89.48+-3.55(1)	88.69+-3.68(1)	81.30+-3.00(1)
	Manhattan	81.15+-2.91(1)	90.03+-2.84(1)	89.00+-3.58(1)	88.67+-3.48(1)	80.56+-3.02(1)
45.0	Euclidian	81.10+-3.12(1)	91.10+-3.58(1)	89.60+-3.87(1)	89.81+-3.13(1)	81.24+-3.10(1)
	Manhattan	80.82+-3.40(1)	91.39+-3.06(1)	90.29+-3.77(1)	89.79+-3.42(1)	81.06+-3.10(1)
50.0	Euclidian	80.90+-3.49(1)	92.17+-3.29(1)	90.38+-3.61(1)	90.04+-3.55(1)	81.47+-3.46(3)
	Manhattan	82.24+-3.11(1)	91.59+-2.86(1)	90.59+-3.75(1)	89.78+-2.96(1)	81.93+-3.05(1)
55.0	Euclidian	82.14+-3.46(3)	91.89+-3.76(1)	90.35+-3.35(1)	90.38+-3.98(1)	80.76+-3.24(1)
	Manhattan	82.06+-3.70(1)	91.83+-2.85(1)	90.78+-3.24(1)	90.57+-2.57(1)	81.75+-2.96(1)
60.0	Euclidian	81.82+-3.31(3)	92.94+-2.86(1)	91.51+-3.23(1)	91.62+-3.36(1)	82.27+-3.37(3)
	Manhattan	81.79+-3.19(1)	92.38+-3.40(1)	91.54+-3.41(1)	90.83+-3.16(1)	82.19+-3.78(1)
65.0	Euclidian	82.20+-3.88(3)	93.36+-2.98(1)	92.12+-3.41(1)	92.28+-3.28(1)	82.36+-3.57(1)
	Manhattan	82.71+-3.93(1)	92.96+-2.84(1)	92.07+-3.35(1)	91.26+-3.08(1)	82.58+-3.96(1)
70.0	Euclidian	82.22+-4.08(3)	94.19+-2.79(1)	92.69+-3.15(1)	92.56+-3.46(1)	82.17+-4.41(3)
	Manhattan	82.81+-3.98(1)	92.54+-2.89(1)	92.24+-3.43(1)	91.53+-3.42(1)	83.31+-3.86(1)
75.0	Euclidian	83.06+-5.70(3)	94.84+-3.07(1)	92.57+-3.81(1)	93.00+-3.80(1)	82.80+-5.07(3)
	Manhattan	83.47+-5.07(1)	93.61+-2.98(1)	92.39+-3.57(1)	91.67+-3.20(1)	82.10+-4.50(1)
80.0	Euclidian	82.67+-4.96(3)	94.77+-3.22(1)	92.87+-4.75(1)	93.08+-4.05(1)	82.26+-5.59(3)
	Manhattan	83.46+-4.87(1)	93.10+-3.51(1)	92.38+-4.11(1)	91.74+-3.74(1)	84.59+-4.80(1)
85.0	Euclidian	83.73+-6.36(5)	94.80+-4.06(1)	93.77+-4.41(1)	92.83+-4.51(1)	82.93+-5.70(5)
	Manhattan	84.10+-5.55(1)	93.37+-4.51(1)	91.97+-4.55(1)	92.33+-4.18(1)	83.23+-6.51(1)
90.0	Euclidian	83.75+-7.56(5)	95.65+-4.93(1)	92.75+-5.89(1)	93.35+-4.75(1)	83.50+-8.38(5)
	Manhattan	83.90+-8.53(1)	93.85+-5.14(3)	92.60+-4.92(1)	92.70+-4.82(1)	85.30+-6.99(1)
95.0	Euclidian	84.40+-10.03(5)	97.00+-6.08(1)	94.40+-6.83(1)	92.30+-9.04(1)	84.40+-12.03(5)
	Manhattan	81.10+-11.82(5)	94.20+-6.51(3)	91.80+-7.79(1)	92.00+-8.49(1)	82.60+-9.86(1)

Table 1

Training (%)	Metric	Raw data	Min-Max	Z-score	Max	Decimal
5.0	Euclidean	75.43±0.27 (7)	77.78±6.69 (1)	76.96±5.52 (1)	77.92±5.58 (1)	79.84±6.66 (1)*
	Manhattan	75.62±3.67 (5)	78.61±5.40 (1)	78.68±5.52 (1)	77.30±6.78 (1)	78.75±6.63 (1)
10.0	Euclidean	76.06±4.44 (5)	80.91±4.83 (1)	81.71±4.03 (1)	81.30±4.72 (1)	83.14±4.43 (1)
	Manhattan	78.03±5.23 (3)	83.37±3.24 (1)	82.50±3.71 (1)	82.64±4.13 (1)	83.45±3.97 (1)*
15.0	Euclidean	78.26±3.91 (3)	84.34±3.56 (1)	84.08±3.88 (1)	84.58±3.04 (1)	86.23±3.45 (1)
	Manhattan	79.26±4.27 (5)	85.15±3.35 (1)	84.64±3.97 (1)	85.19±3.77 (1)	86.83±3.35 (1)*
20.0	Euclidean	79.38±3.30 (1)	87.12±3.13 (1)	86.64±3.10 (1)	85.90±3.44 (1)	87.28±2.98 (1)
	Manhattan	80.85±3.39 (1)	87.17±3.41 (1)	87.21±2.86 (1)	86.53±3.17 (1)	87.94±2.90 (1)*
25.0	Euclidean	80.43±3.58 (1)	88.30±2.92 (1)	87.17±2.83 (1)	87.71±3.08 (1)	88.44±3.14 (1)
	Manhattan	81.73±2.88 (1)	88.86±3.04 (1)	88.43±2.93 (1)	87.82±2.89 (1)	90.23±3.15 (1)*
30.0	Euclidean	80.23±3.79 (1)	89.57±2.63 (1)	88.63±2.83 (1)	88.26±3.17 (1)	89.59±2.55 (1)
	Manhattan	82.57±2.91 (1)	89.80±2.88 (1)	89.40±2.96 (1)	89.29±2.61 (1)	91.13±2.93 (1)*
35.0	Euclidean	81.84±3.47 (1)	90.87±2.52 (1)	89.89±2.84 (1)	89.25±2.78 (1)	91.44±2.95 (1)
	Manhattan	83.21±3.17 (1)	91.01±2.50 (1)	90.59±2.78 (1)	89.83±2.71 (1)	92.34±2.68 (1)*
40.0	Euclidean	82.25±3.38 (1)	91.81±2.75 (1)	90.53±3.06 (1)	90.35±2.82 (1)	91.72±2.52 (1)
	Manhattan	83.60±3.02 (1)	91.64±2.81 (1)	90.97±2.82 (1)	90.34±2.96 (1)	92.96±2.56 (1)*
45.0	Euclidean	83.10±2.98 (1)	92.14±2.80 (1)	90.64±3.07 (1)	90.69±2.72 (1)	92.78±2.23 (1)
	Manhattan	84.35±3.12 (1)	91.84±2.73 (1)	91.49±2.97 (1)	90.62±2.79 (1)	93.41±2.38 (1)*
50.0	Euclidean	82.93±3.07 (1)	92.89±2.67 (1)	91.35±2.74 (1)	91.65±2.55 (1)	93.18±2.60 (1)
	Manhattan	84.74±3.36 (1)	92.85±2.57 (1)	91.84±2.74 (1)	90.95±2.43 (1)	94.08±2.39 (1)*
55.0	Euclidean	83.07±3.50 (1)	93.08±2.52 (1)	92.21±3.01 (1)	92.30±2.56 (1)	93.61±2.44 (1)
	Manhattan	85.15±3.45 (1)	92.52±2.67 (1)	91.67±2.80 (1)	91.54±2.24 (1)	94.09±2.28 (1)*
60.0	Euclidean	83.63±3.24 (3)	93.74±2.42 (1)	92.35±3.08 (1)	93.00±2.79 (1)	93.55±2.46 (1)
	Manhattan	84.49±3.16 (1)	93.05±3.03 (1)	92.50±2.83 (1)	91.56±2.69 (1)	94.65±2.19 (1)*
65.0	Euclidean	83.63±3.72 (3)	94.34±2.76 (1)	93.24±3.05 (1)	93.18±2.52 (1)	93.96±2.40 (1)
	Manhattan	85.25±3.34 (1)	93.79±2.96 (3)	92.63±2.89 (1)	92.04±2.96 (1)	94.90±2.30 (1)*
70.0	Euclidean	83.90±4.59 (1)	94.33±2.82 (1)	93.79±3.14 (1)	92.72±3.11 (1)	94.33±2.79 (1)
	Manhattan	85.36±3.94 (1)	93.62±3.07 (1)	93.00±2.77 (1)	92.22±2.58 (1)	94.74±2.56 (1)*
75.0	Euclidean	84.21±4.29 (3)	95.00±3.26 (1)	93.00±3.79 (1)	93.00±3.55 (1)	94.54±3.13 (1)
	Manhattan	85.88±4.24 (1)	94.75±3.13 (3)	93.71±3.18 (1)	92.02±3.26 (1)	95.10±2.78 (1)*
80.0	Euclidean	85.38±4.23 (5)	95.36±2.98 (1)	94.31±4.04 (1)	93.51±3.63 (1)	94.36±3.80 (1)
	Manhattan	85.21±4.59 (1)	94.13±3.70 (3)	93.05±3.88 (3)	92.72±3.69 (1)	95.87±3.05 (1)*
85.0	Euclidean	85.69±5.29 (5)	95.93±3.48 (1)*	94.17±4.72 (1)	93.59±3.89 (1)	95.38±3.90 (1)
	Manhattan	85.28±5.32 (7)	94.28±4.47 (3)	93.62±3.90 (1)	92.24±4.37 (3)	95.79±3.78 (1)
90.0	Euclidean	84.84±6.95 (5)	96.68±4.14 (1)*	94.68±4.06 (1)	94.58±5.04 (1)	95.95±4.09 (1)
	Manhattan	85.89±7.66 (3)	94.74±5.45 (3)	94.21±4.76 (3)	91.53±5.98 (3)	96.47±3.82 (1)
95.0	Euclidean	85.84±9.53 (5)	96.73±5.97 (1)*	95.75±6.82 (1)	94.55±6.95 (1)	96.51±6.81 (1)
	Manhattan	85.4±11.16 (1)	94.99±6.75 (1)	93.79±7.75 (1)	93.14±7.98 (3)	96.30±5.71 (1)
LOOCV	Euclidean	86.15 (5)	96.41 (1)*	95.38 (1)	94.36 (1)	95.9 (1)
	Manhattan	85.64 (3)	94.87 (5)	93.33 (1)	92.82 (1)	95.9 (1)

Table 2.

Conclusion

In this paper, the problem we aimed to solve is to be able to train a model to be capable of identifying if a person has Parkinson's Disease based on the given data/symptoms. We have explored the use of the KNN algorithm and how it is important to determine an optimal k value, normalization method, distance metric and percentage of training data. Our experiment was performed on a dataset of biomedical voice measurements from 31 people, 23 with Parkinson's disease. The goal was to find the best classification accuracy to determine if someone has Parkinson's disease or not. As discussed in the results, we have determined the algorithm to have the highest average accuracy when the training data is set to 95%, k = 1, uses min-max normalization, and euclidean distance metric. Determining the optimal k value, normalization method, distance metric and percentage of training data is not an easy task, however it is a crucial part in having success with classification algorithms and in our case, training the most accurate model for predicting if a person has Parkinson's Disease.

References

- Ma, C.-M., Yang, W.-S., & Cheng, B.-W. (2014, February 1). How the parameters of k-nearest neighbor algorithm impact on the best classification accuracy: In case of parkinson dataset. Science Alert: Research Papers, Journals, Authors, Subscribers, Publishers. Retrieved April 11, 2022, from https://scialert.net/abstract/?doi=jas.2014.171.176
- Harrison, O. (2018, September 10). Machine learning basics with the K-nearest neighbors algorithm. Medium. Retrieved April 11, 2022, from https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761