Breast Cancer Prediction Using K-Nearest Neighbor: Application of K-Nearest Neighbor

Lab Exercise 1

Introduction to Machine Learning

Domain: Health/Life/Medical

Leader: Angeles, Angelic

Members:

Quintos, Maria Nikki

Urbano, Rommel Jr.

I. Introduction

Health is essential to a human's life and well-being; it is at the center of every individual's life. Every part of your life depends on having a good health. Although health can also be compromised by various infections, diseases, some cancers, and other conditions. Cancer is a major problem of disease in the medical or health domain. Breast cancer is one of the cancer killers in the world. According to cancer.org, breast cancer is the second leading cause of cancer death in women. Using machine learning, we can now reduce the chances of dying from breast cancer. Machine learning is widely used in the field of bioinformatics, specifically in breast cancer diagnosis. Early detection of breast cancer is important because when it is found early, it may be easier to cure and better chances of saving lives. One of the most popular algorithms that we will used is K-Nearest Neighbors (K-NN) which is a supervised learning method. We will used K-NN algorithm to predict the breast cancer in women whether she is having cancer (Malignant Tumor) or (Benign Tumor). The accuracy of the results relies on the distance and the different values of the parameter "k" which represents the number of the nearest neighbors. The goal was to develop a better procedure of predicting breast cancer without the need of a surgical biopsy which is more convenient for both medical professionals and patients.

II. Data Description and Classification Problem

Utilizing the Data Set provided by Dr. William H. Wolberg, General Surgery Dept. University of Wisconsin, W. Nick Street, Computer Sciences Dept. University of Wisconsin, and Olvi L. Mangasarian, Computer Sciences Dept. University of Wisconsin during 1992, due to change of time, knowledge and technology we may have better alternatives on addressing the problem, however, this activity simply shows that Machine Learning can be of great help to our Medical Professionals and Patients alike, we focus on the Medical Processes of determining if a patient has Breast Cancer and how Machine Learning can assist our professionals in determining a patient's status. The problem that we aim to understand and help solve is how can we determine is a patient is a Breast Cancer patient with the use of existing data. Attributes of 10 are used to represent factors that contribute to the possibility of a patient having Breast Cancer with their corresponding scale (1-10):

- Clump Thickness 1 - 10
- Uniformity of Cell Size 1 - 10
- Uniformity of Cell Shape 1 - 10
- Marginal Adhesion 1 - 10
- Single Epithelial Cell Size 1 - 10
- Bare Nuclei 1 - 10
- Bland Chromatin 1 - 10
- Normal Nucleoli 1 - 10
- Mitoses 1 - 10
- Class: (2 for Benign, 4 for Malignant)

The utilized columns are 9 in total, and the records used are 699, therefore producing a DataFrame with dimensions of 699 by 9. Each parameter presented revolves on the observation of

cells in the patient's body, the lower the rate per each attribute, the less abnormal it is. The "Class" attribute is the corresponding result of each rows in the dataset. The Data Set portrays a Benign Tumor to be as 2 and a Malignant Tumor to be as 4. A Malignant Tumor is a tumor in the body that persists and could be a sign that a patient has bodily abnormalities, most commonly associated with cancer, while a Benign Tumor is a tumor that can be removed surgically and will not resurface/persist in the patient's body. The Class distribution in this Data Set is 65.5% Benign (458 records) and 34.5% Malignant (241 records).

III. Data Pre-Processing

3.1 Preprocessing of Non-Normalized Data

For preprocessing of non-normalized data, we used the drop cell function. Since we are dealing with the non-normalize data there is no need for much data cleaning, as most of the attributes are essential deciding factor aside from the ID number. We performed the following preprocessing steps:

1. Drop the ID number attribute from the dataset.

3.2 Preprocessing of Normalized Data

For preprocessing of normalized data, we also used the drop cell function and performed the following preprocessing steps:

1. Drop the Id number and Class attribute.
2. Create a "refreshData" variable that loads again the data.
3. Extracting the Class attribute using the "refreshData" variable and save the data as a data frame.
4. Dropping the Class on the dataset is a way to only normalized the needed attributes.
5. Once normalized, retrieved the class attribute, and placed the attribute back on both data frames.

3.3 Features

To find the feature that has a relationship with each other and that would contribute most to the result. We performed the feature selection by finding the correlation of paired features.

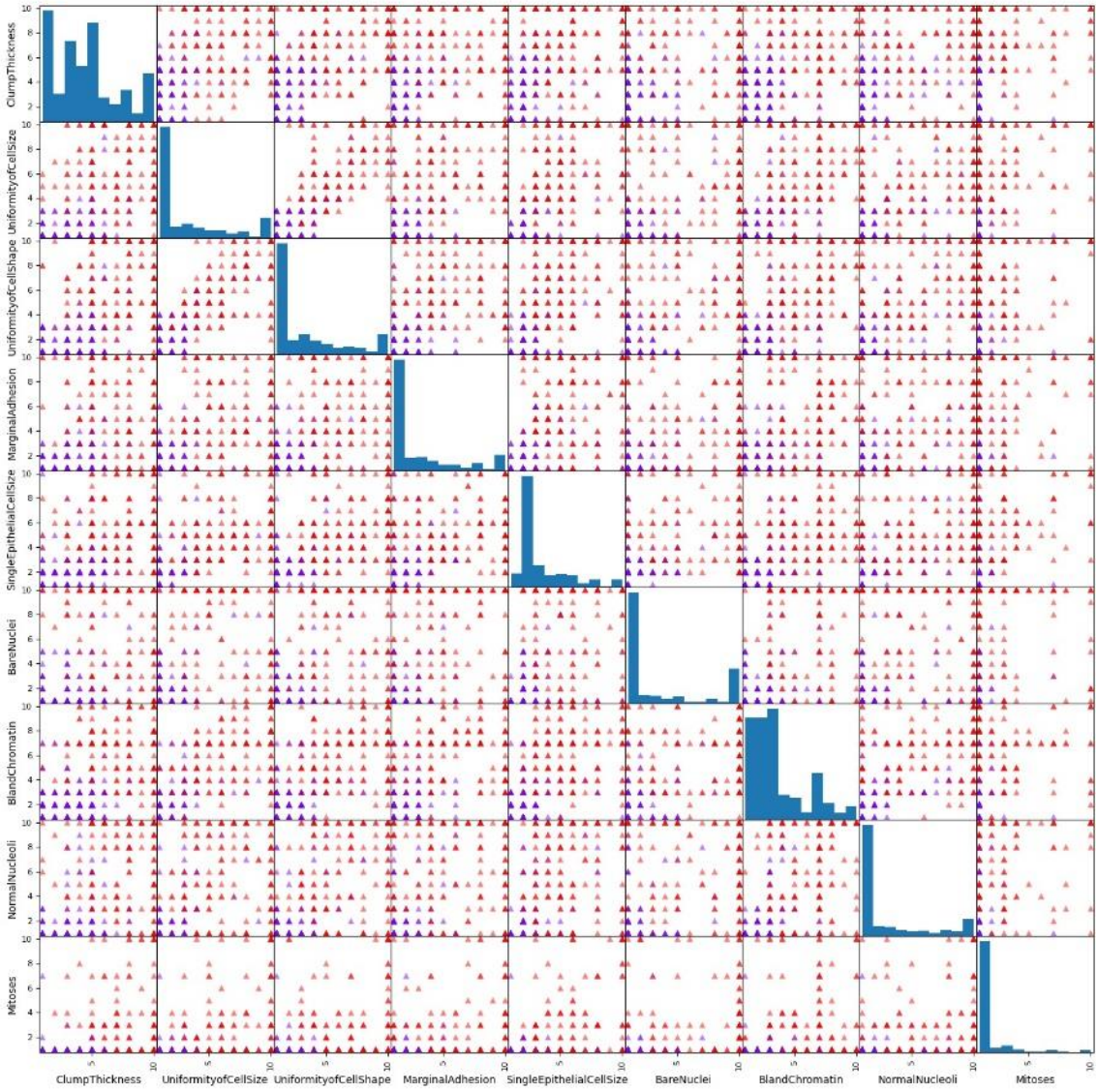

Figure 1: *Visualization on the result of each paired features.*

Figure 1 shows the result when each attribute is paired with the other attributes. The scatter plot creates an x and y axis to determine the movement of each attribute depending on the corresponding data.

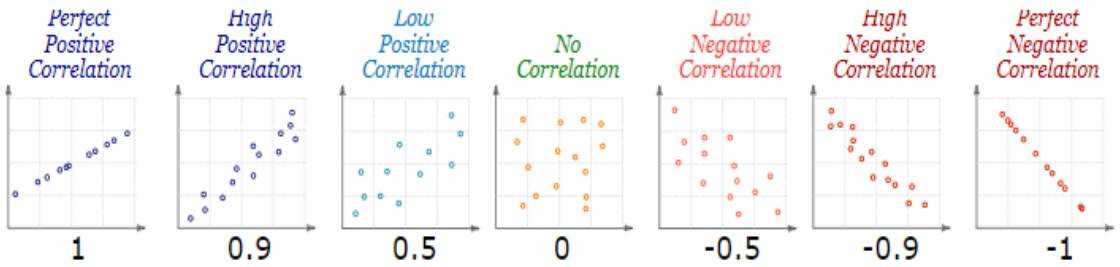

Figure 2: *Classifications of Correlation.*

Based on the article in towards data science each classification shows how good the correlation is. Positive correlations mean that if, feature A increases then feature B also increases. A negative correlation means that if feature A increases then feature B decreases. No correlation simply means that there is no relationship between those two attributes.

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Where:

- Σ is Sigma, the symbol for "sum up"
- $(x_i - \bar{x})$ is each x-value minus the mean of x (called "a" above)
- $(y_i - \bar{y})$ is each y-value minus the mean of y (called "b" above)

Figure 3: *The Formula used on Finding how good the relationship between features.*

Figure 3 was used to compute for the correlation of each features. Using the formula, we were able to see on which classification of correlation is better fit for the result of the paired feature.

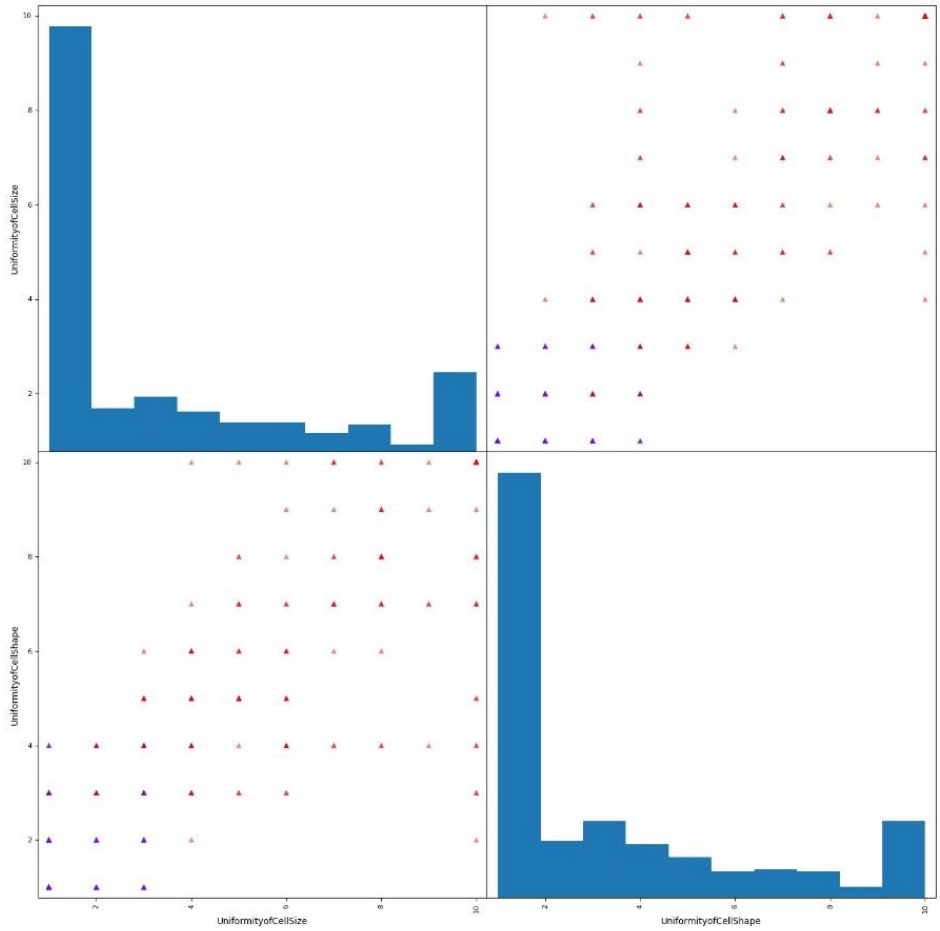

Figure 4: *Positive Correlation from the Features*

From the observation made on Figure 4, the distribution of the data on the plot and how it is formed, shows that only the uniformity of cell size and uniformity of cell shape has a relationship. Base on figure 2 classifications of correlation.

	UniformityofCellSize	A-AVE	UniformityofCellShape	B-AVE	AXB	A2	B2
0	1.0	-2.134478	1	-2.207439	4.711730	4.555996	4.872788
1	4.0	0.865522	4	0.792561	0.685979	0.749129	0.628153
2	1.0	-2.134478	1	-2.207439	4.711730	4.555996	4.872788
3	8.0	4.865522	8	4.792561	23.318311	23.673306	22.968639
4	1.0	-2.134478	1	-2.207439	4.711730	4.555996	4.872788

Figure 5: *Result of the Uniformity of cell size and uniformity of cell shape.*

Finding the Correlation
0.91

Figure 5.1: *Overall Result.*

Using the formula on figure 3, we were able to solve the data of uniformity of cell shape and size as shown on figure 5. To get the needed data we used excel were in the data is located, and we come up to the result shown on figure 5.1. Upon solving for the correlation that has a positive correlation, we can now classify on which correlation it lands base on the result on figure 5.1. The result from figure 5.1, shows that the uniformity of cell size and uniformity of cell shape has a high positive correlation base on figure 2.

The importance of the correlation in our data is to be able to know if we have a perfect positive correlation or a perfect negative correlation. The reason is, based on the article in towards data science if the dataset has perfectly positive or negative attributes there is a high chance of multicollinearity and that leads to misleading results. The only way to solve this is to delete one perfectly correlated feature or deduct the dimension using the principle component analysis.

IV. Answer to Questions

a. What is the highest prediction accuracy that the K-NN algorithm can generate using the raw dataset?

Raw Dataset

K Value	Accuracy
K=2	0.93
K=3	0.95
K=4	0.95
K=5	0.96
K=6	0.95
K=7	0.96
K=8	0.95
K=9	0.95

Table 1: *Accuracy of K-NN depending on the value of k.*

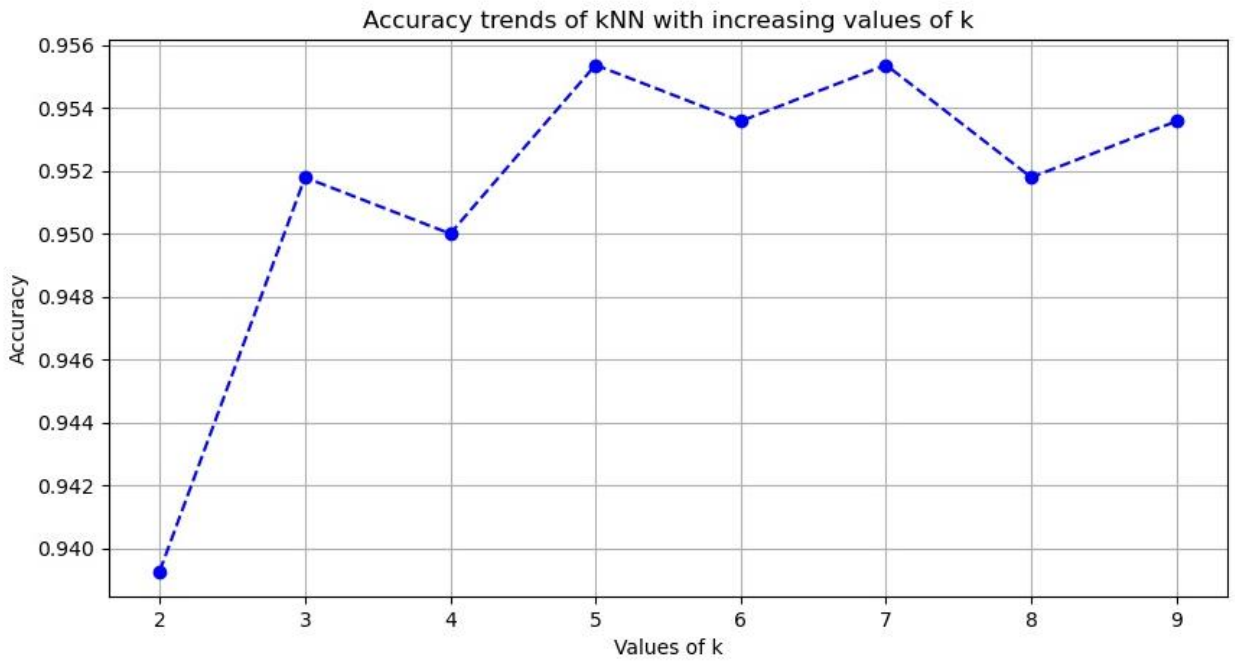

Figure 6: *Visual Representation of the result of accuracy with increasing value of k.*

The highest accuracy that we obtained with the increasing value of k on the raw data set is when k=5 and k=7, with the accuracy of 0.96 obtained as shown on Table 1 and shown on the visualization when k is increasing on Figure 6.

b. Will applying normalization on the dataset improve the prediction accuracy of the K-NN algorithm? What is the performance difference of K-NN with raw dataset vs. K-NN with normalized dataset?

Normalized Dataset		Raw Dataset	
K Value	Accuracy	K Value	Accuracy
K=2	0.88	K=2	0.93
K=3	0.88	K=3	0.95
K=4	0.88	K=4	0.95
K=5	0.89	K=5	0.96
K=6	0.91	K=6	0.95
K=7	0.91	K=7	0.96
K=8	0.90	K=8	0.95
K=9	0.91	K=9	0.95

Table 2: *Accuracy Comparison of Normalized and Raw Dataset*

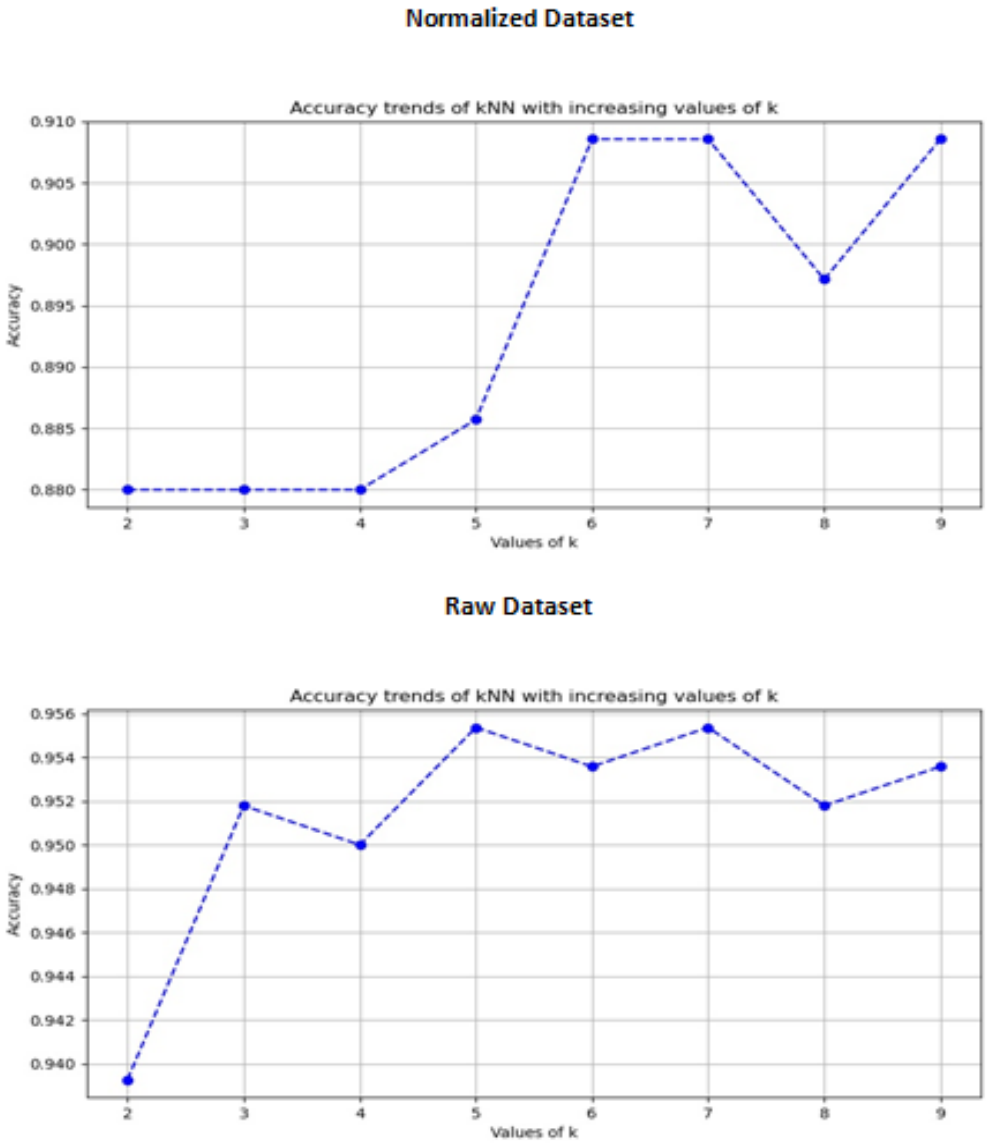

Figure 7: *Visualization on the accuracy result of raw dataset and normalized dataset*

Based on the result shown in Table 2, applying normalization does not make the accuracy improved, by observing the raw and normalized dataset you can see the behavior of the result on the graph shown in Figure 7, on which the raw dataset has the higher result than the normalized dataset. The accuracy decreases when the normalization is applied. The performance of the raw dataset based on our observation, when k reaches an odd number, the accuracy increases and when the k reaches an even number the accuracy decreases. While the performance of the normalized dataset seems to be different, as shown on the visualization on the result on Figure 7, when k = 2 to 4 it has the same value or behavior and when it reaches a certain value it increases. From the visualization, we concluded that it does not have a distinct pattern unlike the raw dataset.

c. How sensitive is K-NN classification accuracy to the choice of the 'k' value? What is the best value for the parameter ‘k’ for your Dataset?

The number of nearest neighbors is one of the main factors that can affect the classification accuracy of the K-NN algorithm. There are no criteria on finding the optimal value of k. It still depends on the dataset and algorithm that you will use which will have a better fit.

c.1 Non-Normalized Data

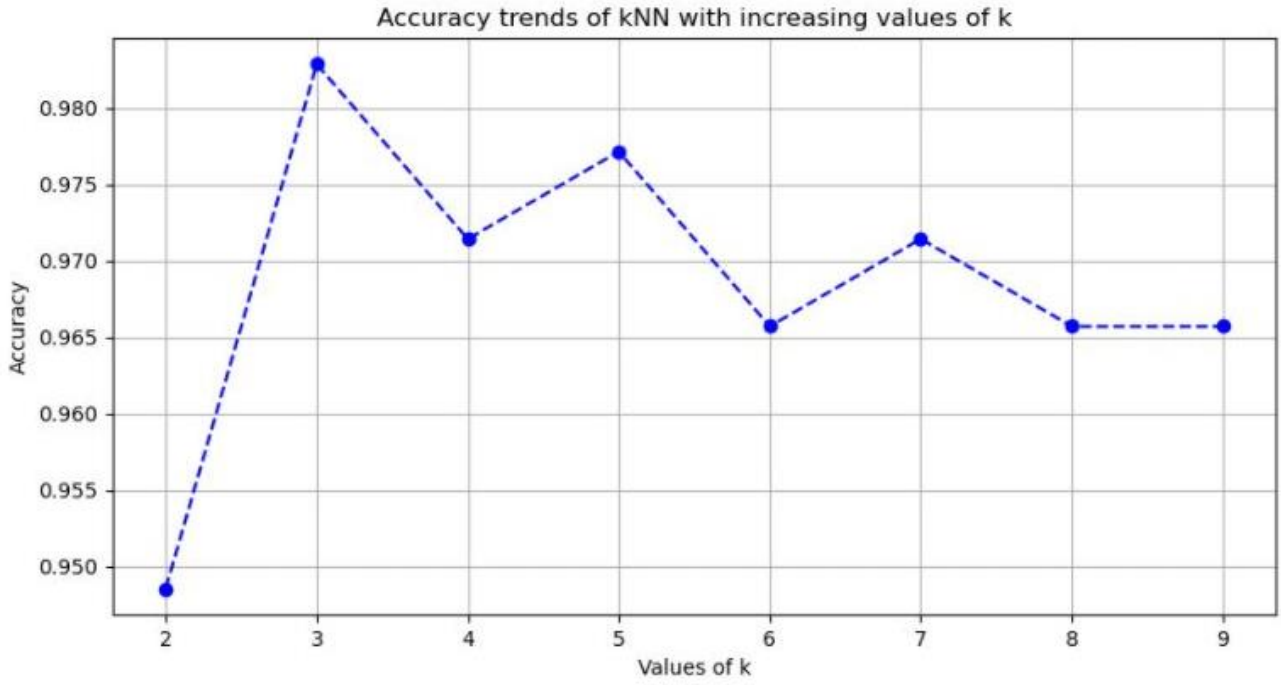

***Figure 8.1**: Graph Accuracy Trends of K-NN with respect to k (Non-Normalized)*

The graph shows the accuracy trends of K-NN in an unnormalized data with increasing values of parameter ‘k’ which represents the number of nearest neighbors. The classification accuracy of the K-NN algorithm drastically increased from 0.949 (where the value of k is 2) to 0.983 (where the value of k is 3) then the accuracy decreased at 0.971 (where the value of k is 4) then the accuracy increased again at 0.977 (where the value of k is 5). The graph is alternatively increasing and decreasing from values of k (2 to 8), remained constant from values of k (8 to 9). Based on my observation, when the value of k is an odd number, the accuracy also increases but when the value of k is an even number, the accuracy decreases. So therefore, we can conclude that the K-NN classification accuracy is sensitive on some values of k (2 to 8) and as the values of k increased more than 8 it became stable which means that larger values of k are less sensitive since there is a wide range of neighbors to depend on and are smoother between the class boundaries. In addition, the best value for the parameter ‘k’ of our dataset is an odd value of k to avoid the ties in case your distance measure achieves the same score or same number of frequencies for various classes that will lead to confusion in the algorithm. In this dataset, the optimal value of ‘k’ would

be 5 because there is no significant increase in the accuracy when we increase the value of k more than 5.

c.2 Normalized Data

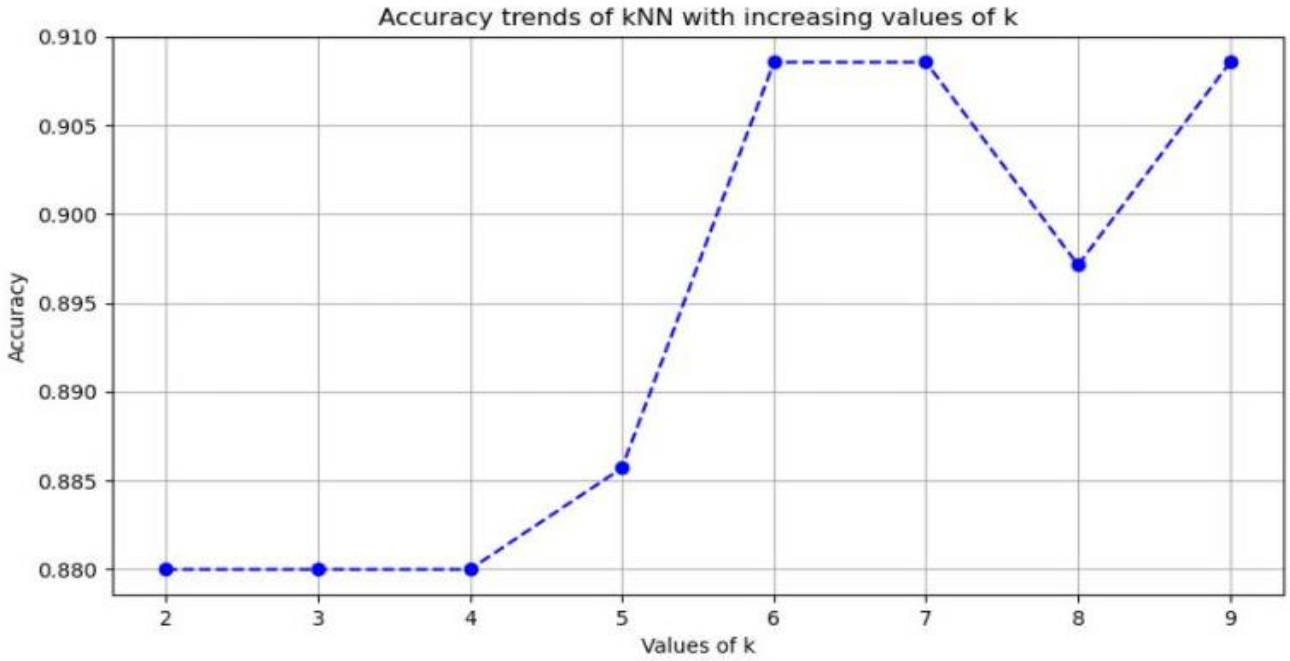

***Figure 8.2**: Graph Accuracy Trends of K-NN with respect to k (Normalized)*

The graph shows the accuracy trends of K-NN in a normalized data with increasing values of parameter 'k' which represents the number of nearest neighbors. The classification accuracy of the K-NN algorithm remained constant on values of k (2 to 4) and the accuracy slightly increased at 0.886 (where the value of k is 5) and drastically increased from 0.886 to 0.909 (where the value of k is 6) then remained constant up until 7 then decreased at 0.897 (where the value of k is 8) then increased again at the same accuracy of the value of k (6 and 7) which is 0.909 (where the value of k is 9). Compare to the unnormalized data, the normalized data has more constant values because when we normalize the data, we change the values into a common scale. In addition to that the accuracy drastically decreased (0.880, 0.885, 0.890, …, 0.910) compare to the accuracy of the unnormalized data which are (0.950, 0.955, 0.960, …, 0.980). So therefore, we can conclude that the K-NN classification accuracy using a normalized data is less sensitive compare to the unnormalized since the data are now reduced and some attributes are eliminated but the classification accuracy drops when using normalized data. That does not mean that it is less sensitive, the better the accuracy will be. In addition, the best value for the parameter 'k' of our dataset would be 6 since there is no significant increase in the accuracy, it just remains constant from values of k (6, 7, and 9).

d. Does adjusting the train/test split proportion also affect the performance of K-NN? Based on your observations, how sensitive is K-NN classification accuracy to this ratio?

K-Nearest Neighbors (K-NN) is a supervised machine learning algorithm which means that it maps a labeled training data to its corresponding output. The goal is to find the relationship between x and y. In order to find their relationship, we must split the dataset into training data and testing data. So that we can train the model and test the accuracy of it. The optimal splitting proportion depends on the size of the dataset and classification accuracy.

d.1 Non-Normalized Data

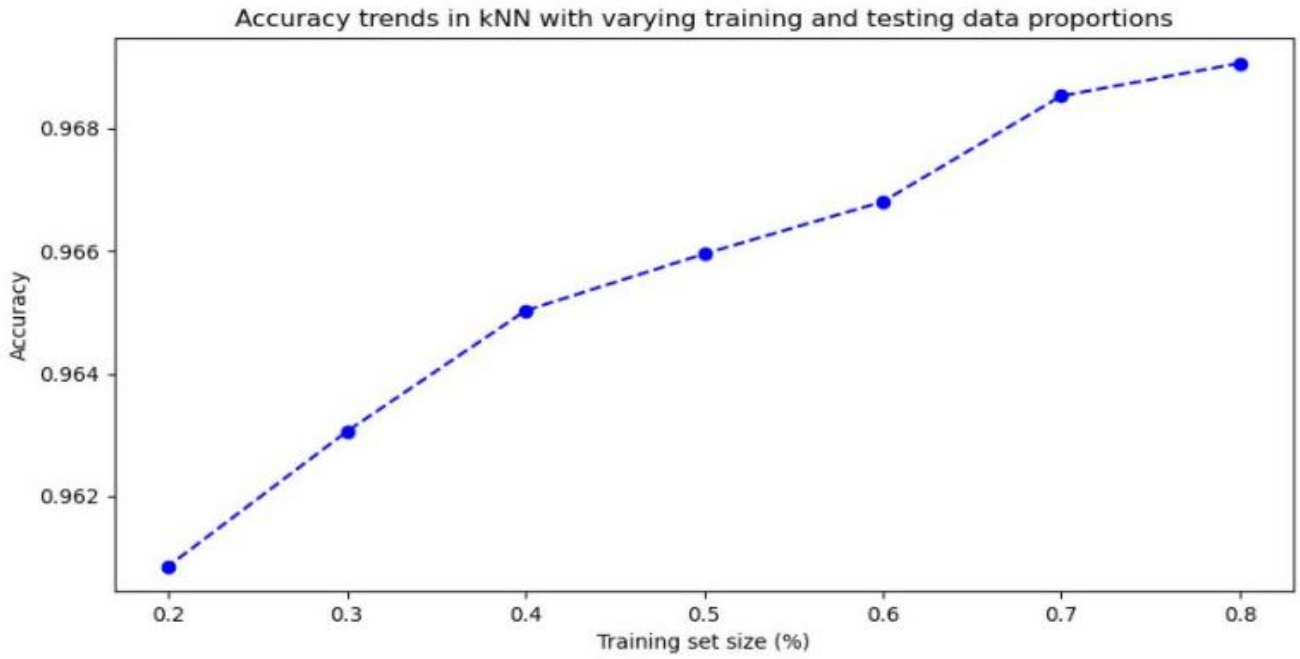

Figure 9.1*: Graph of Accuracy Trends in K-NN with varying training and testing data proportions (Non-Normalized)*

Figure 9.1 shows the accuracy trends of K-NN in a non-normalized data with varying training and testing data proportions. There was a gradual rise in the accuracy of the K-NN algorithm as the training set size increases. Overall, the classification accuracy went up from 0.960 to 0.969. Based on my observations, there is a high accuracy obtained using the non-normalized data. From 0.2 to 0.3 training set size, there was a substantial growth from 0.960 to 0.963 accuracy. Then from 0.3 to 0.4 training set size, there was a significant increase from 0.963 to 0.965 accuracy. Then from 0.4 to 0.5 training set size, the accuracy increased considerably from 0.965 to 0.966. Then from 0.5 to 0.6 training set size, the accuracy increased again from 0.966 to 0.967. Then from 0.6 to 0.7 training set size, there is a substantial growth again from 0.967 to 0.968. Last but not the least, from 0.7 to 0.8 training set size, there was a slight rise in the accuracy from 0.968 to 0.969.

d.2 Normalized Data

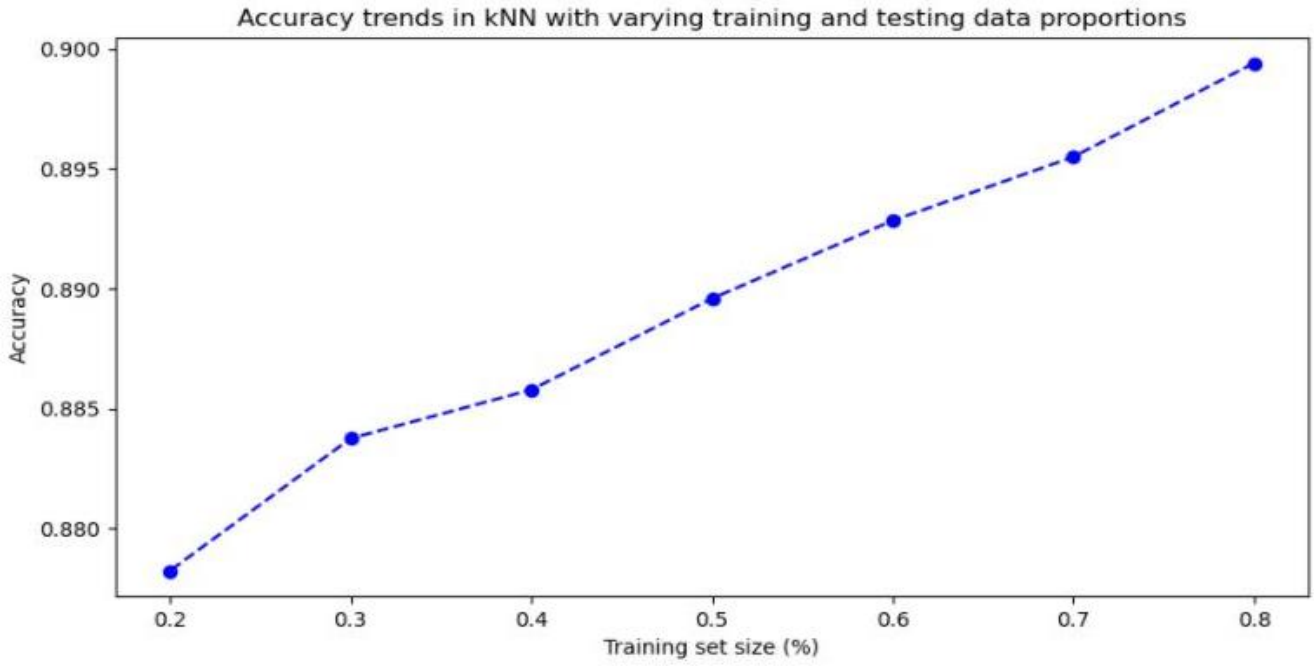

***Figure 9.2**: Graph of Accuracy Trends in K-NN with varying training and testing data proportions (Normalized)*

Figure 9.2 shows the accuracy trends of K-NN in a normalized data with varying training and testing data proportions. There was a moderate rise in the accuracy of the K-NN algorithm as the training set size increases. Overall, the classification accuracy went up from 0.876 to 0.899. Based on my observations, there is a low accuracy obtained using the normalized data. From 0.2 to 0.3 training set size, there was a significant increase from 0.876 to 0.884 accuracy. Then from 0.3 to 0.4 training set size, the accuracy slightly rises from 0.884 to 0.885. Then from 0.4 to 0.5 training set size, the accuracy increased considerably from 0.885 to 0.890. Then from 0.5 to 0.6 training set size, there is an increased again from 0.890 to 0.893 accuracy. Then from 0.6 to 0.7 training set size, there is a steady growth from 0.893 to 0.895. Last but not the least, from 0.7 to 0.8 training set size, there was a substantial increase in the accuracy from 0.895 to 0.899.

So therefore, we can conclude that adjusting the training and testing dataset size will greatly impact the accuracy performance of the K-NN algorithm in both non-normalized and normalized datasets. In addition, the classification accuracy is sensitive with respect to the training and testing data proportions since we can see that as the training set size increases, it also increases accuracy. The only difference between non-normalized and normalized is that there is a high accuracy (0.960 to 0.969) when using a non-normalized data compared to a normalized data which has a low accuracy (0.876 to 0.899). Optimal data proportioning is essential to prevent large bias when estimating the classification accuracy.

e. Try to measure the time it takes for each version to run and chart your observations. Is there a significant time difference in running the different version of K-NN? By how much?

Approaching the time complexity of the K-NN algorithm, we have observed both the Non-Normalized and Normalized Data to further observe if Normalization has enhanced the performance of the K-NN algorithm. The Testing and Training Proportions used for this test are as follows:

- Training Data:80% Testing Data:20%
- Training Data:50% Testing Data:50%
- Training Data:20% Testing Data:80%

These will be the parameters of Training and Testing data for this experiment, accompanied by a range of values of k from 2 to 9. Let us first observe the Time Complexity (in seconds) of the Non-Normalized Data.

- Non-Normalized Data

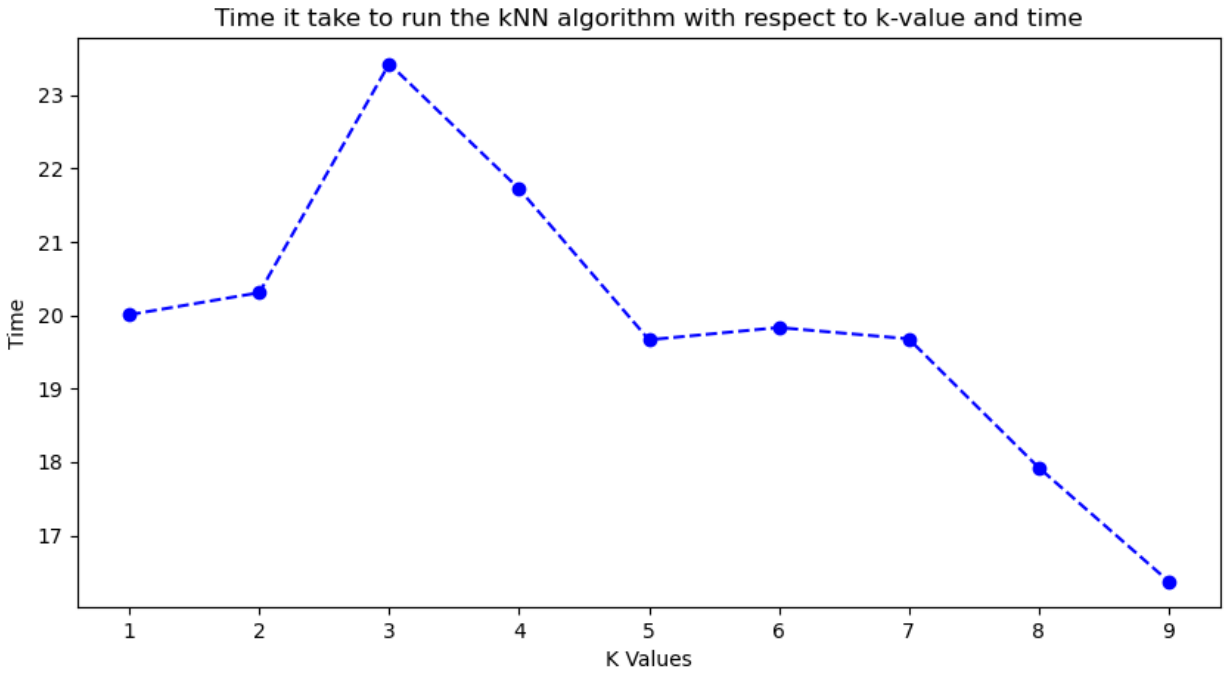

Figure 10.1: *Time Complexity of K-NN algorithm with Testing Data of 20% and Training Data of 80% with respect to the series of k (Non-Normalized)*

For Figure 10.1, we are using the standard 80% Training Data and 20% Testing Data, as we iterate through k-values of 2-9, we can see a decrease of time consumption, almost resembling a linear curve. Inconsistencies are found when our k-value is 3, we observe a spike in time as this is the value of k where the K-NN algorithm performs the proper classification, following the general rule that k-values should revolve between 3-10, from 3 onwards, a decrease has been

observed as our k-value becomes greater, this may be an indication that as our k-value becomes larger, more factors come into play and results in a faster decision making of the algorithm.

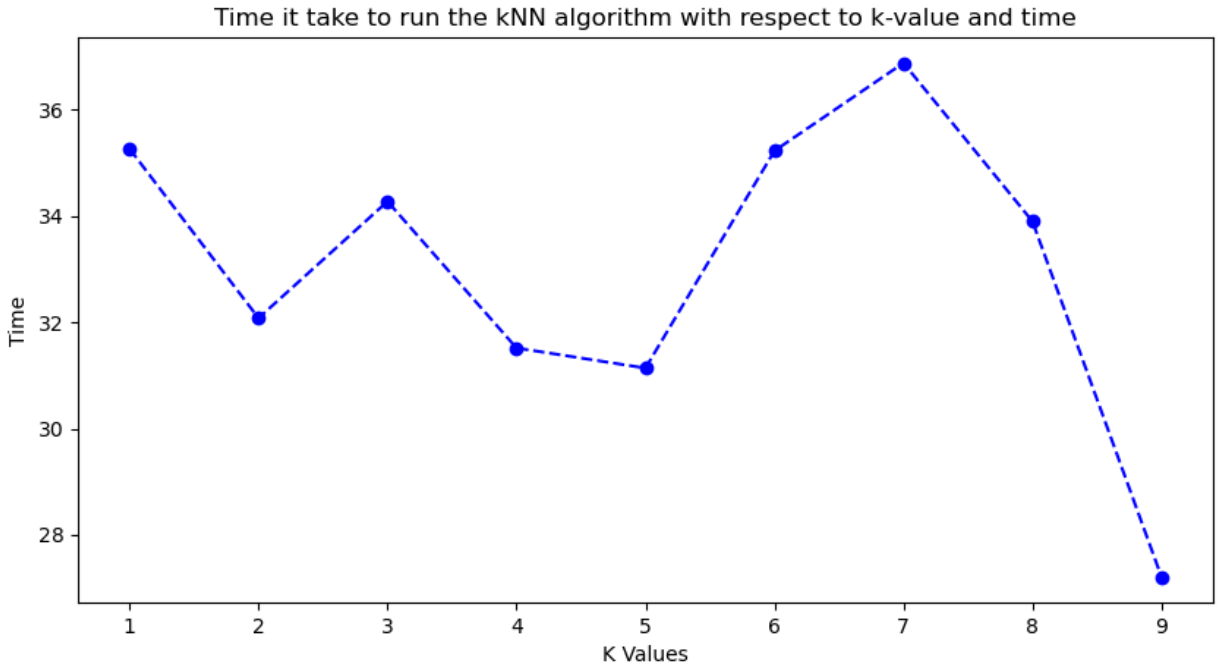

Figure 10.2: *Time Complexity of K-NN algorithm with Testing Data of 50% and Training Data of 50% with respect to the series of k (Non-Normalized)*

For Figure 10.2, we utilize 50% of the data for Training, and 50% of the remaining data for Testing. We can see inconsistent time consumption as we iterate through the values of k. Observing the general rule that our k-values should be between 3-10, we can see a massive spike in time as we reach the k-values of 6 and 7, and observe a significant drop in time as we reach the k-values of 8 and 9. We can say that as we “train” (memorization) the K-NN algorithm less, the effect of k-values over time lessens.

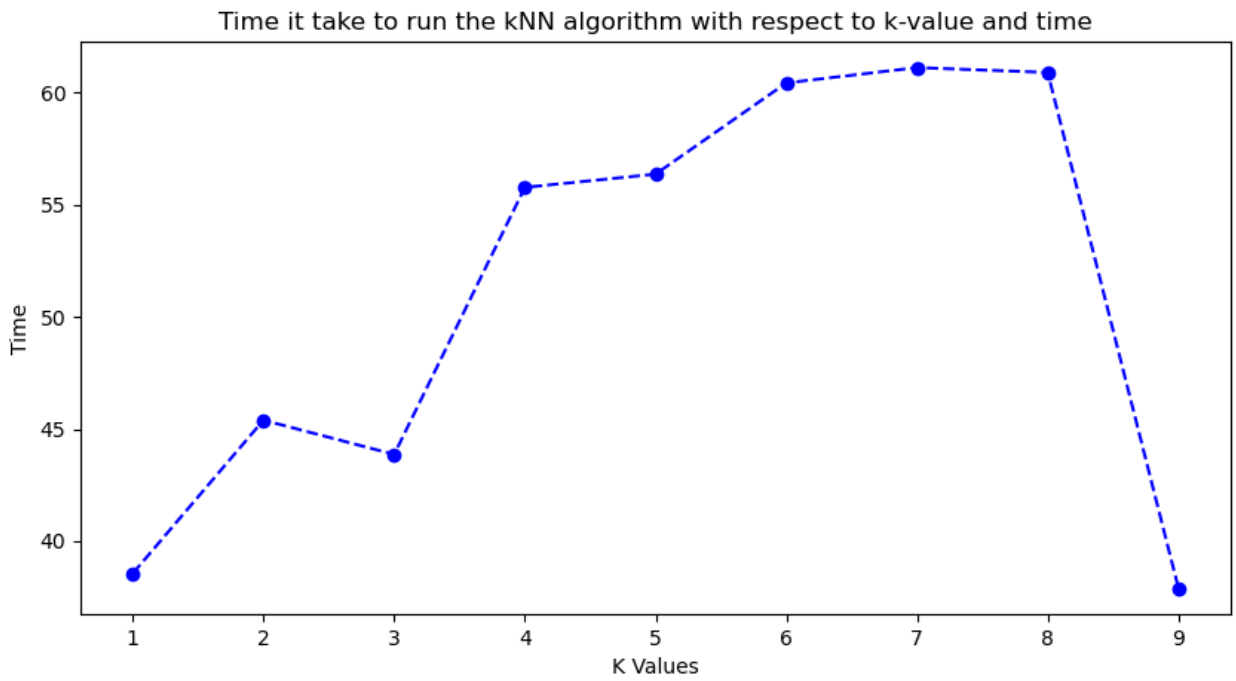

Figure 10.3: *Time Complexity of K-NN algorithm with Testing Data of 80% and Training Data of 20% with respect to the series of k (Non-Normalized)*

Figure 10.3 shows the time graph of the K-NN runtime with Training Data of 20% and Testing Data of 20%. With relation to the observations of Figure 10.2, the less we train the K-NN algorithm, the factor or k-values with time lessens, and it is more prominent here with Figure 10.3. The algorithm's "learning/memorization" does not have much information to have a proper classification, even if we try to find the "best" k-value, the decision making of the algorithm with 20% Training Data will have a hard time distinguishing between parameters, therefore, having to allocate more time in classification, there is also a significant drop in time consumption when we reach the k-value of 9, this is an indication that as we test through the data, the algorithm memorizes each test and will have a much easier decision making, due to more information gathered.

- Normalized Data

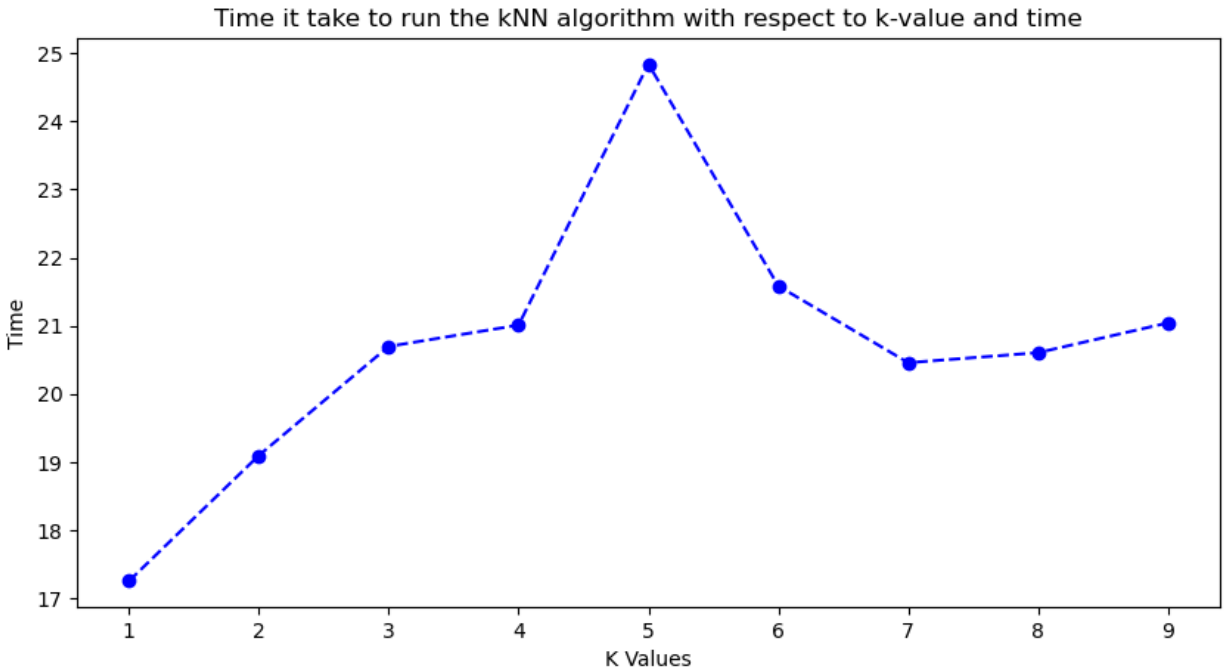

Figure 10.4: *Time Complexity of K-NN algorithm with Testing Data of 20% and Training Data of 80% with respect to the series of k (Normalized)*

Figure 10.4 show the time consumption per run of the K-NN algorithm with respect to the value of k. We are utilizing the standard 80% Training Data and 20% Testing Data, as per the usual Train-Test Split. Observing the general rule of that the k-values should generally be in between 3 and 10, we can start seeing a spike in time when we use the k-value of 5, and drastically drop down in time with the following k-values of 6-9. k-values of 3,6,7,8,9 generally revolves on the same time span of between 21-22 seconds. There exists a spike in k-value of 5, this may be the cause of as we progress through the values of k, in k-value of 5, there comes in more and new factors in the classification of the algorithm that affects the decision making of the algorithm, therefore allocating more time in classification with the new found test parameters. Afterwards k-value of 5, we can see that the algorithm has now “memorized” the new data parameters of testing, further optimizing decision making on the algorithm, therefore lowering time consumption.

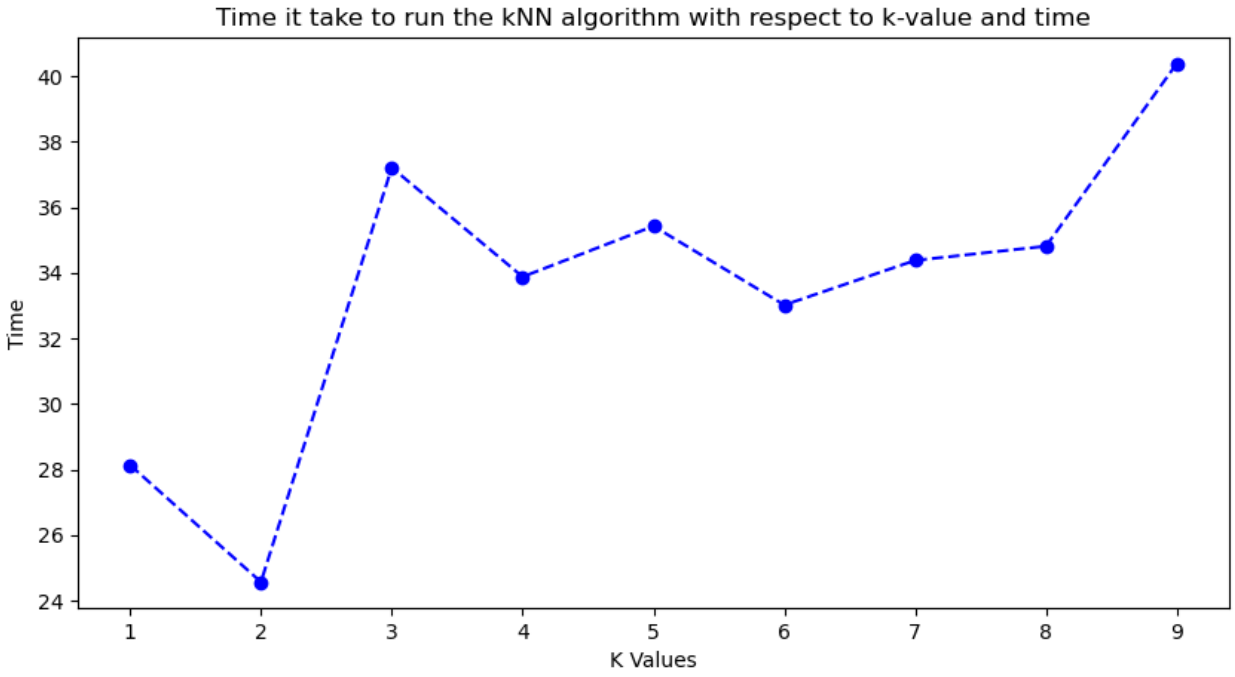

Figure 10.5: *Time Complexity of K-NN algorithm with Testing Data of 50% and Training Data of 50% with respect to the series of k (Normalized)*

Figure 10.5 showcases the time consumption of running the K-NN algorithm while utilizing the 50-50 ratio of Test-Train Split. In comparison with Figure 10.4, as we iterate through k-values between 1-9, we can see a resemblance of a linear graph, gaining time consumption as our k-values gets larger, with k-value of 3-8, we can see inconsistent records of time, revolving around 33-38, and as we get to k-value of 9, a spike of 40 seconds is observed. These inconsistencies in runtime is a result of having less training allocation of data, making for a more difficult classification.

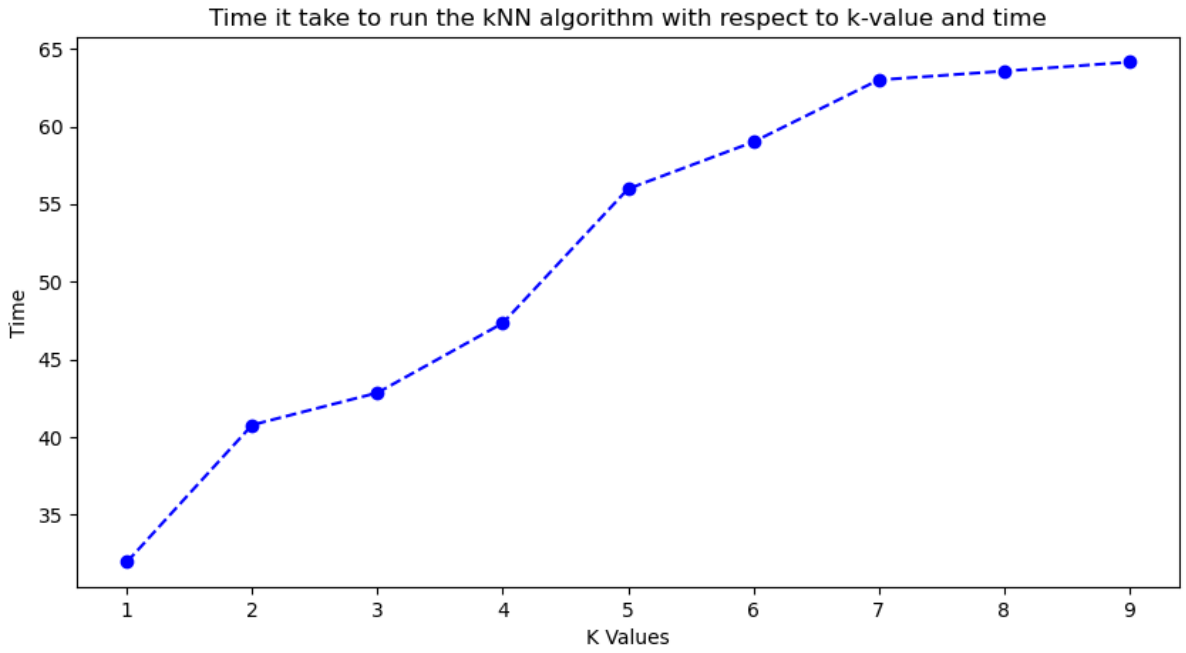

Figure 10.6: *Time Complexity of K-NN algorithm with Testing Data of 80% and Training Data of 20% with respect to the series of k (Normalized)*

Figure 10.6 shows a linear graph with time increasing as we iterate through k-values. In this graph we are utilizing the Train-Test Split with the ratio of 2:8, 20% for Training and 80% for testing. As k-values increase the time the KNN algorithm needs to take grows larger, this is due to less "training" for the algorithm making decision making consume more time as our k-value grows. With this Train-Test Split, we can say that lesser training size makes the algorithm have less information to compare to, therefore increasing time, and as more "factors" come in (k-values), the more prominent the lack of information becomes.

- Normalized Data vs Non-Normalized Data

Main findings on the Non-Normalized graph shows that as our k increases, the time we need to run the kNN algorithm decreases, this is the result of having our data in whole numbers, therefore increasing the distance of each parameter, making decision making much more easier.(Figure 1 and Figure 4). For the Non-Normalized Data, we can say that both Training Size and k-values contribute to the runtime of the algorithm.

Normalized Data, with Train-Test Split of 8:2, 80% Training Data, 20% Testing, we can see that there is almost a "plateau" with run time (with the exception of k-value 5), this is due to the parameter data being between 0 and 1, making distance much more shorter, therefore consuming more time in decision making. A linear graph is present in Normalized Data Test Split of 50-50 and 20-80, this is due to the lack of Training Data for the algorithm to memorize and compare parameters to, therefore increasing time as factors (k-values) increase.

Now we ask the question, does k and training size correlate and affect the performance of the algorithm?

K-values and training size go hand in hand in dictating the runtime of each iteration of the K-NN algorithm, Training Size handles on how much information the algorithm holds and compares to, k-values holds how many "votes" should be considered in making the prediction. Having less training size results in longer runtimes regardless of k-values and having to iterate k-values optimizes the algorithm more, making runtime become either consistent, or decrease.

V. Summary and Conclusion

The study focuses on medical or health domain specifically in cancer. According to the World Health Organization (WHO), cancer is a leading cause of death around the world and one of the most common cancer is breast cancer having approximately 2.09 million cases. The aim of this study is to easily predict or determine the type of breast cancer (Malignant or Benign) does a female patient have without undergoing surgery and several procedures. In this way, we can help patients and medical professionals make their work easier and more convenient for them. We gathered a labelled dataset from Computer Sciences Department, University of Wisconsin during 1992. Then we conducted the experiment on optimal utilization of the K-Nearest Neighbors (K-NN) Algorithm to solve the problem for the health domain. Based on our observations from the data that we analyzed, here are the findings we can conclude:

- The highest accuracy obtained by testing the k from ranges 2-9, has an accuracy of 0.96 when k = 5 and k = 7.
- After testing the normalized and raw dataset we can see that it has a certain pattern in the raw dataset, on which the accuracy increases when k is equal to an odd number, and it decreases when k is equal to an even number. While the normalized dataset does not have any pattern and showing a lower accuracy with 0.91 as the highest when k = 6, k = 7 and k = 9 which means that applying normalization on the dataset won't improve the prediction accuracy of the K-NN algorithm and would only make it worse compared to the raw dataset with 0.96 as the highest accuracy when k = 5 and k = 7.
- K-NN classification accuracy using a normalized data is less sensitive compare to the non-normalized but the classification accuracy drops when using normalized compared to non-normalized which has a higher accuracy. In addition, the best value for the parameter 'k' for non-normalized dataset is an odd value of k which is 5 because there is no significant increase in the accuracy when we increase the value of k further. On the other hand, the optimal value of 'k' for normalized dataset is 6 since there is no significant increase in the accuracy, it just remains constant from values of k (6, 7, and 9).
- Adjusting the training and testing dataset size has a significant impact on the accuracy performance of the K-NN algorithm in both non-normalized and normalized datasets. In addition, the classification accuracy is sensitive with respect to the training and testing data proportions since we can see that as the training set size increases, it also increases accuracy. The only difference between non-normalized and normalized is that there is high accuracy (0.960 to 0.969) when using non-normalized data compared to normalized data which has low accuracy (0.876 to 0.899). Optimal data proportioning is essential to prevent large bias when estimating the classification accuracy.

- K-values and Training Size go hand in hand in determining the runtime of the algorithm. As shown with series of Figure 10, as the Training Data gets less, the time to run several instances of k-values increases. The more training data is processed, the less time it takes for the algorithm to run several instances of k-values, if not less, the runtime difference is minimal.