School of Mathematical and Statistical Sciences

Arizona State University

Student: Mason Manning

Other students: Ashley Barth, Nicholas Chmielewski, Chein-Wen Pan, Ouse Sheblak

Course: APM 505

Program: Mathematics MA

Instructor: Malena Espanol

Date: Fall 2019

APM505 Final Project Report

Team 2: Mason Manning, Ashley Barth, Nicholas Chmielewski, Chein-Wen Pan, Ouse Sheblak

Introduction:

Diabetes is one of the most prominent diseases in the world. In 2015, approximately 30.3 million American had diabetes, and of those 30.3 million, 7.2 million were undiagnosed ("Diabetes"). Diagnosing a person with diabetes is not always accurate, depending on the method of diagnosis used. Some methods can lead to false positive, where a person is diagnosed as diabetic when they are not, or false negative when they are not diagnosed with diabetes when they are in fact diabetic. The two methods currently used to confirm diagnosis are testing the A1C, which produces an average blood glucose level for the past 2 or 3 months, and testing for the presence of IAA, IA-2A, ICA and GAD antibodies, which attack the insulin-producing cells of the pancreas. We will use kNN and weighted kNN using physiological characteristics to predict diagnosis.

Classification is a very common method used in data analysis. It assigns a new data point, vector, or matrix to a group or a 'class' by comparing it to already classified data. Classification can be done using many methods. K nearest neighbors, or kNN, is one of the methods or algorithms used to classify data.This method finds the nearest classified points to the unclassified data point, and compares the number of points belonging to one class over the other(s). K is the number of points considered in this classification. If the majority of the points belong to one class, then the unclassified data point belongs to that class k is the number of data points considered when ranking the distances.

Weighted kNN does not depend on the number of closer points as seen with the regular kNN. It assigns more weight to the distances between the unclassified data point and the classified data points. This method classifies a data point by calculating the distance between it and the nearest data points. These distances are then ranked from smallest to largest, where the smallest distances indicate the closest data points to the unclassified data point. If the unclassified data point has more points of one class closer to it than the other, it is then classified as the former. This gives a more reliable result as the nearest neighbors could possibly vary in distances, which would affect the result if the regular kNN method was used. These distances are calculated using norms. Different norms can also affect the classification as they might give different results or distances.

For this project, the classifications or results obtained using regular kNN and the weight kNN are compared to see which are more accurate. The data set used was obtained from Kaggle.com and was collected by the National Institute of Diabetes and Digestive and Kidney Diseases as a part of the Pima Indian Diabetes database (Jagadish). All the data points were also obtained from females of age 21 and above. The two classes used in this classification are 'diabetic' and 'non-diabetic'. The data set uses eight attributes or characteristics to make this classification; Pregnancy, Glucose Levels, Blood Pressure, Skin Thickness, Insulin Levels, BMI, BMI, and Age. It also includes the true classification of each data point, which will be compared to the result of the kNN and weighted kNN to test for their accuracy.

Method:

In this project, classification accuracy in standard kNN and weighted kNN algorithm using different k values and norms are evaluated. Each algorithm was implemented into a function and an analysis code was used to visualize the accuracy under different conditions.

kNN function takes four inputs: 'Data' is the matrix of classified data. In this project, it is a 693*9 matrix where 693 means the training sample size and 9 indicates the eight attributes mentioned in introduction plus one column of true classification. 'Newpoint' is the matrix of unclassified data. 'k' is the integer representing the number of nearest-neighbor to observe. In the nearest-neighbor (k = 1) approach, the resulting decision boundary is composed of hyperplanes that form perpendicular bisector pairs of points from different classes. 'p' is the norm for calculating distance. In the project, 1, 2, and inf norm were used. Then the norm of each data point is calculated according to the assigned p-norm, and is stored in a vector B in an increasing order. K nearest classified data point will then be used to help deciding the class of the unclassified point.

The wKNN function calculated the prediction for a set of new points using the weighted KNN algorithm. It takes as input the training data called train data, which includes the outcome variable as the last column. It also takes as input a matrix of test data, called testdata,which is the new points to be predicted on. Then to define the algorithm it also takes as parameters k, which is the number of closest neighbors to consider, and normtype which is a numeric value that either calls the p-norm for finite integers or inf norm in the case that Inf is passed for this parameter. For each row in the test dataset, the function looks at the distance between that point and all the rows in the training data (minus the last column). It then identifies the k closest points and it's associated distance. Then the weight for the point i is given by w_i=(1/d_i)/sumk(1/d_k) where d_i is the distance between the chosen point i and the point we are fitting and the sum is over all k distances. This weight ensures the sum of the weights is equal to 1. Then, the predicted score is given by summing over the weights multiplied by the outcome variables (a vector of 0's and 1's). This gives a number between 0 and 1, inclusive. The final prediction is decided to be 1 if the score is greater than or equal to 0.5. Otherwise the final prediction is decided to be 1. After completing this calculation for all rows in the training data, the wKNN function then returns a column vector of 0's and 1's corresponding to the outcome of the data from each row in the training data.

Before the data is classified we will normalize the data set. This is done because for instance Glucose is around 80 or more and pregnancies are 20 or less, if we want to compare these when classifying we must first normalize the data such that every value is between zero and one, essentially we would like all variables to have the same weight. To normalize the data we will find the mean and standard deviation of each column in the data set, then for each element in the column we take the absolute value of the entry minus the mean of the column, we then divide by the standard deviation and thus we have a normalized data set. We then split the data such that we remove ten percent which is used as our unclassified data or test data, we will

store the true values of the test data in a separate vector to check accuracy later. The results of this can be found in the paragraphs and figures below.

Results:

The visualization plots below show the true values for each data point in column zero (first column of the plot) sorted such that the zeros (red) are grouped together as well as the ones (blue). Along the x-axis are different values of k, and the y-axis is the ten percent of the data that was removed. Essentially each dot (not in the first column) is a data point that was classified using the specified norm and the k value represented on the x-axis with red representing one and blue representing zero.

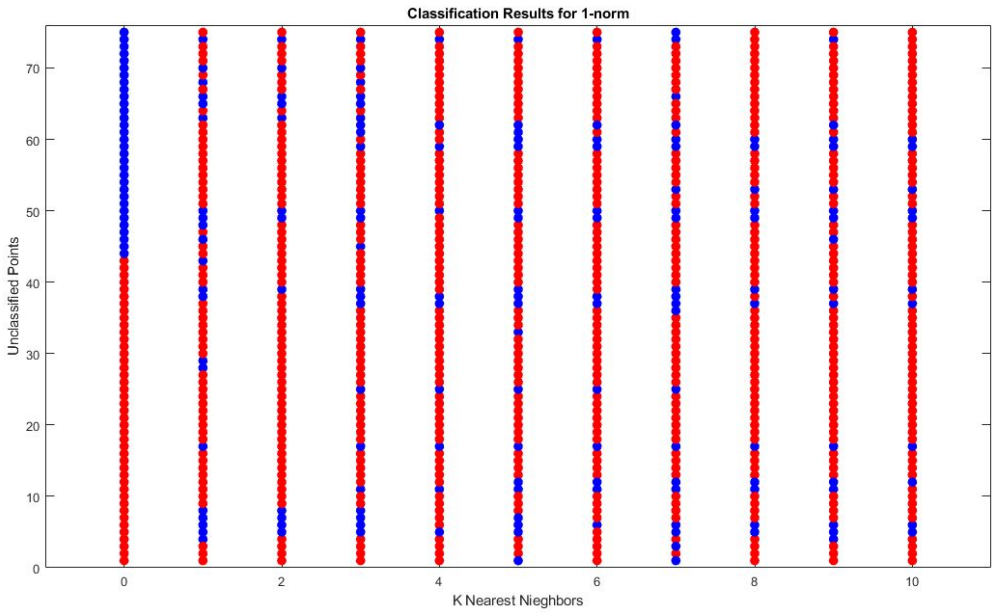


Figure 1: Standard KNN 1-norm Results

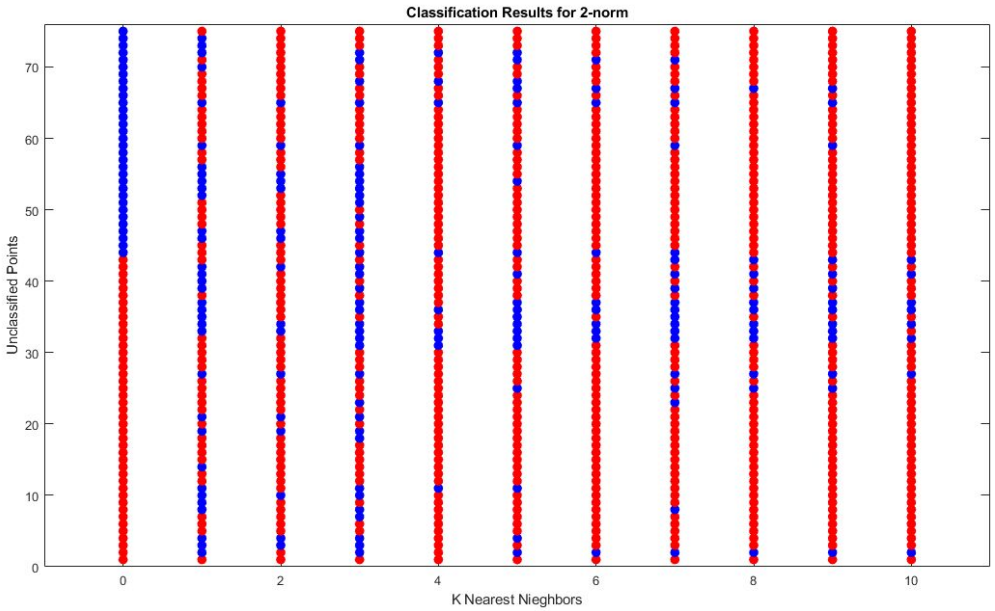


Figure 2: Standard KNN 2-norm Results

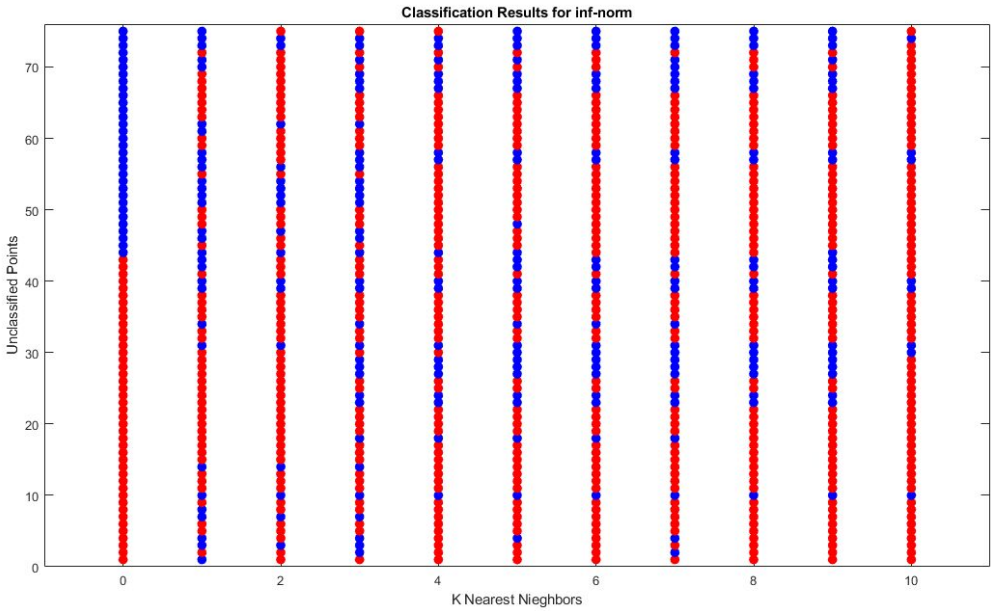


Figure 3: Standard KNN inf-norm Results

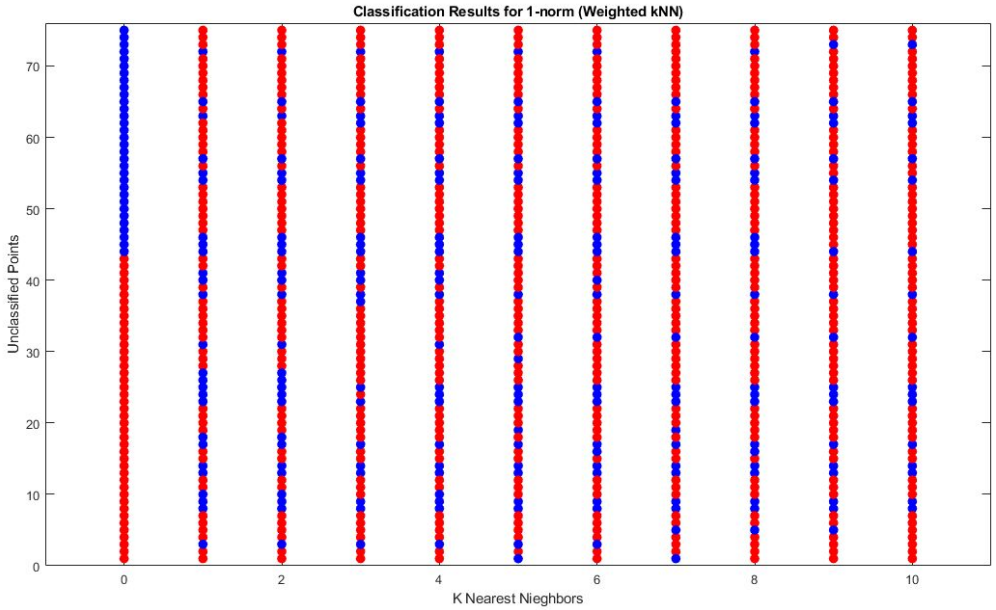


Figure 4: Weighted KNN 1-norm Results

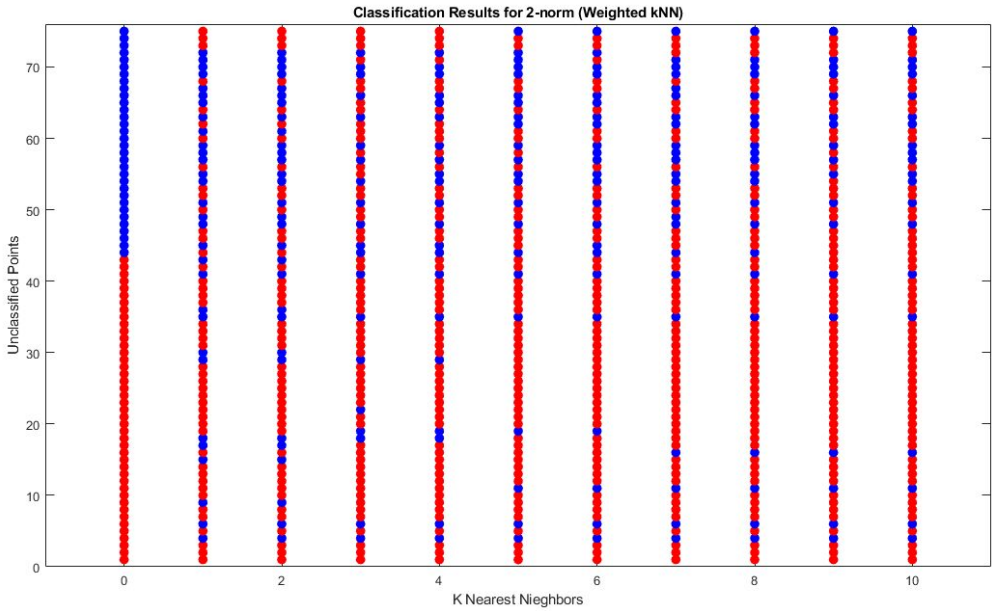


Figure 5: Weighted KNN 2-norm Results

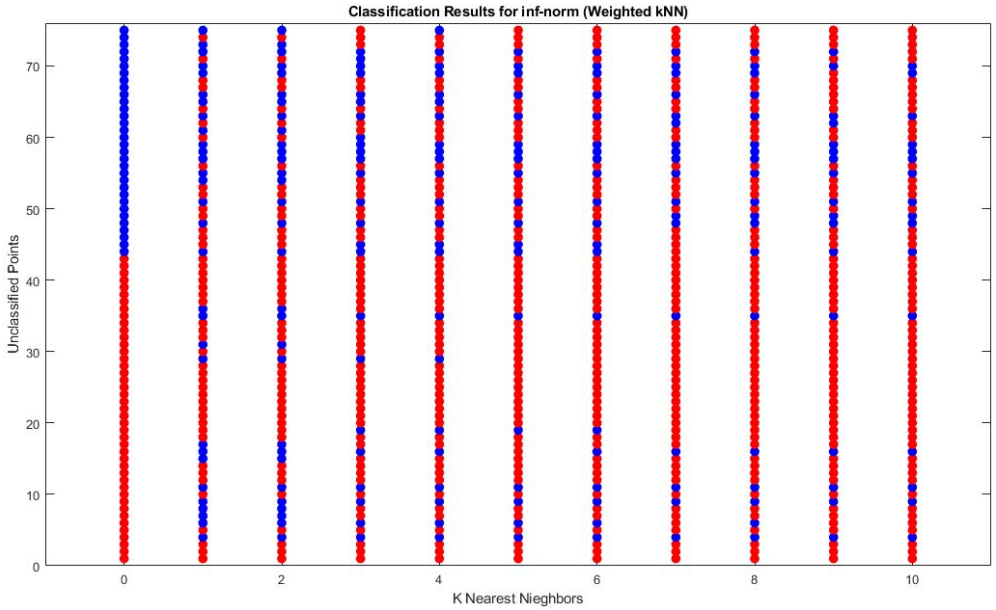


Figure 6: Weighted KNN inf-norm Results

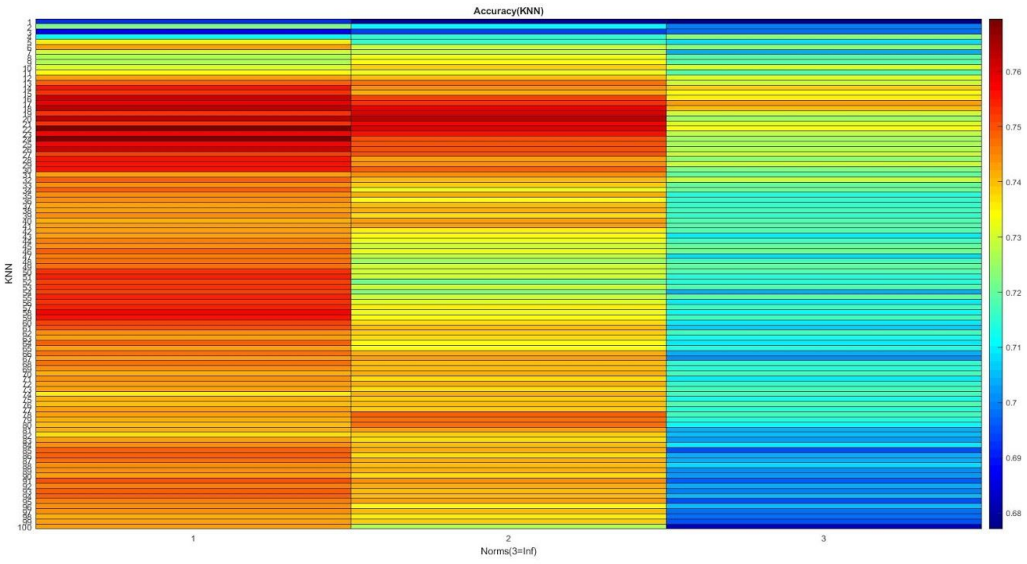


Figure 7: Standard KNN Results

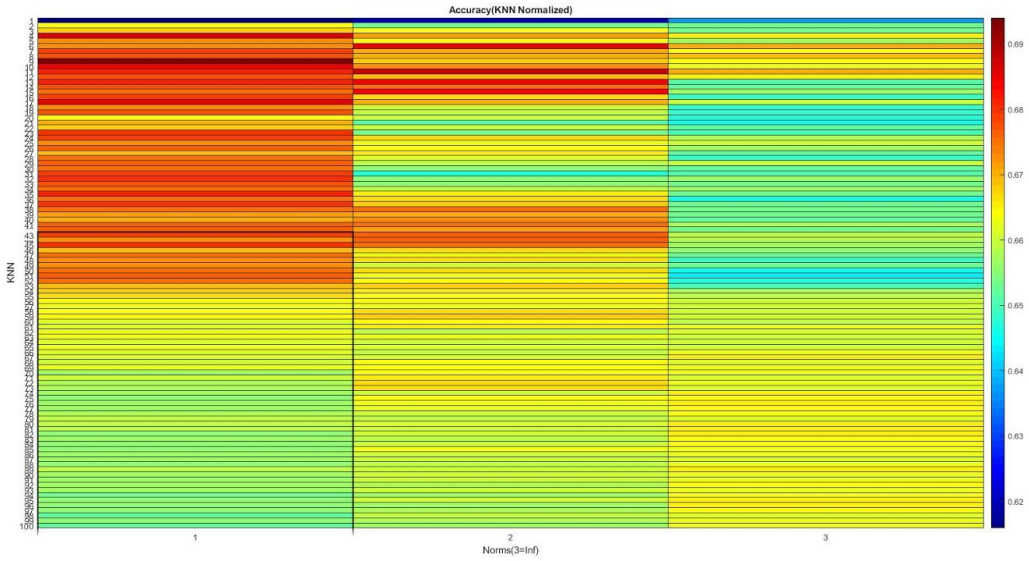


Figure 8: Standard KNN Results with normalized data

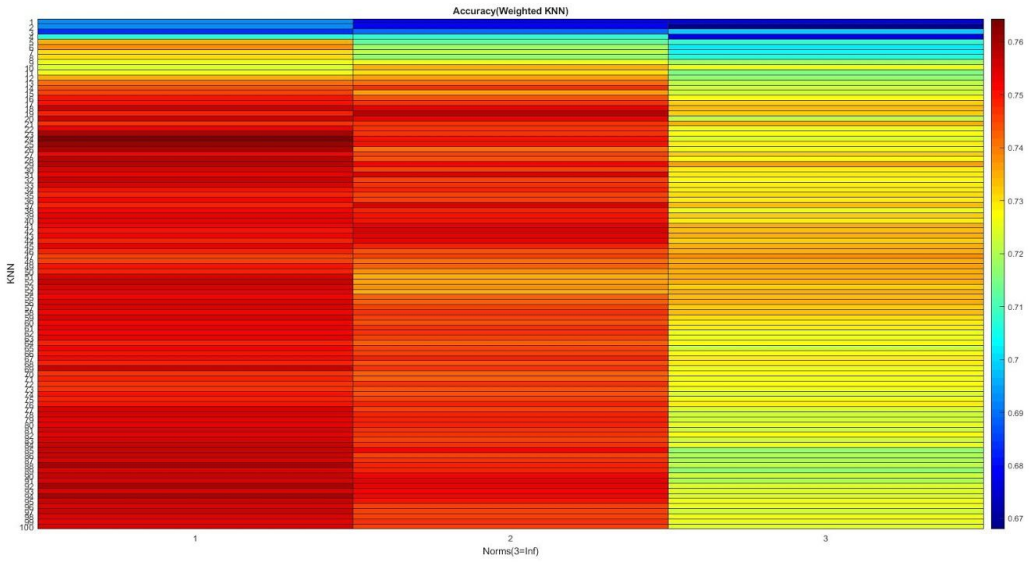


Figure 9: Weighted KNN Results

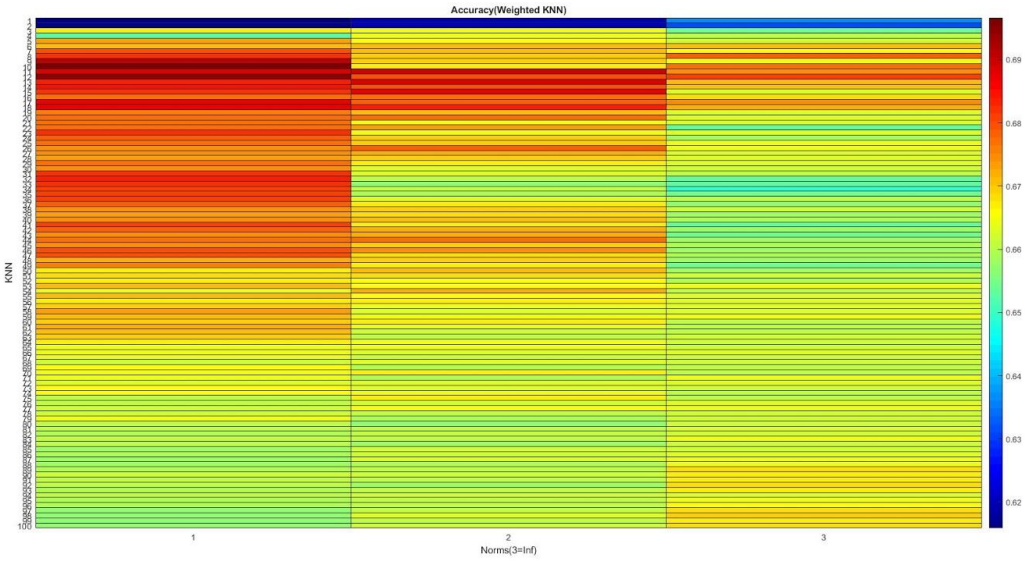


Figure 10: Weighted KNN Results with normalized data

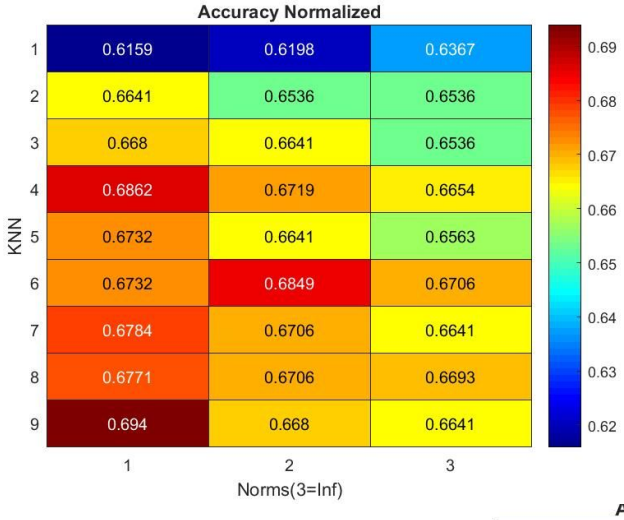
Figure 7-10 gives us a heat map for accuracy. Using said heat map, we were able to nicely represent three-dimensions. Those dimensions being the norms that were used to classify the new point - which is our x-axis(x=3 is are infinite norm) -, the number of k-nearest neighbors - we want the function to consider which is our y-axis - and the accuracy of our model as a shade of color where red means relatively high accuracy and blue means relatively low accuracy. Note the previous figure shows blue and red dots for diabetes and not diabetes whereas these figures present accuracy; meaning was the new point classified right or it was classified wrong? 700+ data points were tested and their accuracies averaged.

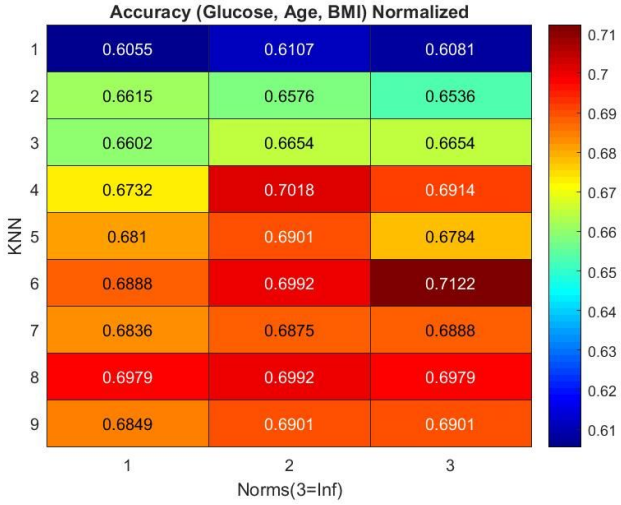
Interestingly, Figures 1-3 have better accuracy then Figures 2-4. The reason for this is because the data for Figures 2-4 was normalized. This normally seems like it would beneficial. However, this is only true if you know that every classifying variable is equally important for determining how a new point should be classified.

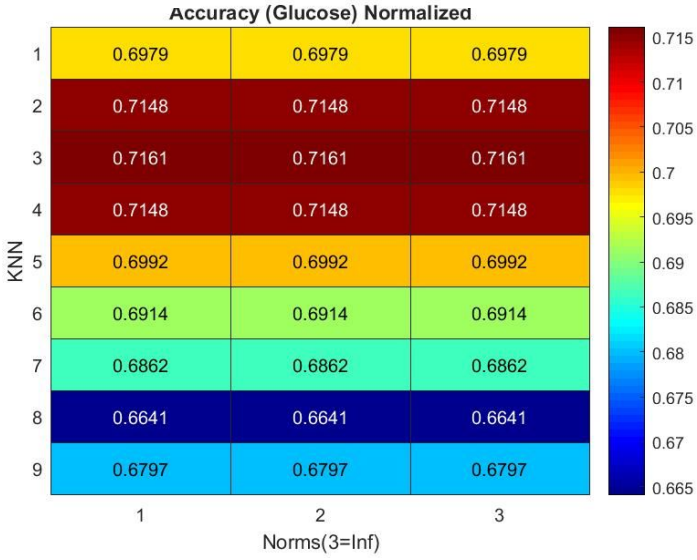
For example, if we want to classify a fruit as either an orange or an apple, the variables we use to do so are **weight**, **color** and "**does it have seeds?".** In this case, **weight** and **color** seem to help for classifying and could be considered equally important. Although if we treat "**does it have seeds"** as equally important our results will be heavily flawed since both apple and oranges have seeds.

With some further testing on which parameters seem to give the best results, it was found that glucose, age, and BMI were the only parameter, by themselves, to do better than just

randomly guessing. When combined, they perform better than when you consider all the parameters as seen in the figures below.







Conclusion:

Using a data set of over seven hundred points each of which contains eight variables and a class, we used both the standard and weighted k-nearest neighbors algorithms across several norms to classify a set of unknown points. Upon our first tests, which considered all eight variables, we found results that were mostly lackluster. We were seeing results just over half correct which is really no better than just flipping a coin. As we began to remove variables from our model the results got much better, with our best results being just over 70 percent by using just glucose for low values of k, and results yielding 78 percent accuracy by using age, glucose and BMI for high values of k. What this tells us is that your best bet in predicting whether or not a patient is diebetic would be to observe their glucose levels, BMI and to keep in mind the age of

the patient. The standard and weighted k-nearest neighbors algorithms are just two of many classification algorithms, it may be that for this data another algorithm may be a better match yielding a better model with a higher accuracy. Therefore, we did some literature review to compare our models with other algorithms. In 2008, researchers used the least squares support vector machine (LS-SVM) for classification diabetic dataset and got 82% accuracy [2]. Another group got a 96.68% accuracy by first using k-means clustering to identify and eliminate incorrectly classified instances. Then they used generic algorithm and correlation based feature selection (GA-CFS) for relevant features selection. And finally applied kNN using features subsets from the second stage as inputs for classification [3]. One other group got an accuracy of 78% by using support vector machine (SVM) in 2013 [4]. Although our best model gives us a high 70% accuracy in classifying diabetics, many other performances could be done to improve the classification result.

References:

1. Jagadish, Kandi. "Diabetics Prediction Using Logistic Regression." *Kaggle*, 7 May 2019, https://www.kaggle.com/kandij/diabetes-dataset.
2. K. Polat et al. \Expert Systems with Applications 34 (2008) 482–487
3. A.G. Karegowda et al. IJEAT (2012)
4. V.A. Kumari et al. IJERA (2013)

Dropbox Link to Our Matlab Code:

All code written to produce the results in this report can be found in the following dropbox folder. https://www.dropbox.com/sh/3ikl9bg9gkkvaib/AAD7P-d40gBl6NRhgo4ev4zba?dl=0