Big Data Home Assignment 1 Report

Experimental Data:

A file named 'HeartDisease.csv' has been used for performing the experiment. This csv data consists of 270 instances(rows) and 3 features(columns). Feature 1 and feature 3 are continuous variables(varying from 29 to 77 and 94 to 200 accordingly) whereas, feature 2 is a categorical variable with two categories as 0 and 1. This data is picked up from UCI Machine Learning repository(https://archive.ics.uci.edu/ml/datasets/Heart+Disease). The original CSV data consists of 75 features which were not required in the experiment and has been trimmed down to 3 features according to the requirements.

Imputation methods used: 1-NN, K-NN, Weighted K-NN

1. **1-NN(One Nearest Neighbor)**:

It's a traditional method to find a value of X. Take the **closest neighbor**(Y) of X and assign Y's value to X. To find the nearest neighbor of X, we use different distance measures. In attached code, **oneNN** is the function name used for calculation where distances are calculated by Euclidean, Manhattan and Minkowski distance methods

2. **K-NN(K-Nearest Neighbor)**:

The same as 1-NN, instead of 1 nearest neighbor, we search for K nearest neighbors and for continuous feature, the **mean of k values** will be the value of X and for categorical feature, the most frequent value will be the X value. For this, **kNN** is the function name written in the attached code.

3. **Weighted K-NN (Weighted K-Nearest Neighbor)**:

It is an improved version of K-NN. K-NN might perform inaccurate if nearest neighbor varies widely in their distances or if k is too small or too large. In this method, k neighbors are given a weight which is nothing but inverse of the calculated distance. For continuous features, the value of the X is calculated by taking the **mean of weighted values** (The weighted values are weights multiplied with the values of those k points). Whereas, for categorical features, weights are summed up according to the category of k points and whichever category gives the highest weight sum will be the category of x. The idea behind this Weighted k-NN method is to give more weight to the nearest points rather than the points which are far away. For weighted k-NN method, **weight_kNN** function name is used in attached code.

Distance measures used: Euclidean method, Manhattan method, Minkowski method

1. **Euclidean method**:

It is one of the distance measures for calculating the straight-line distance between two points. Euclidean distance can be calculated with following formula.

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

where i is the number of features, $x=(x_1,x_2,x_3,\ldots,x_n)$ & $y=(y_1,y_2,y_3,\ldots,y_n)$ are the two point vectors in which we have to calculate the distance. This method can be found with **euclidean_dist** function name in attached code.

2. **Manhattan method**:

This method is an alternative to Euclidean distance for calculating the distance between two points. It can be calculated as the sum of horizontal and vertical distances between two points on a grid. Manhattan formula is as below:

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

where i is the number of features, $x=(x_1,x_2,x_3,\ldots,x_n)$ & $y=(y_1,y_2,y_3,\ldots,y_n)$ are the two point vectors in which we have to calculate the distance. This method can be found with **manhattan_dist** function name in attached code.

3. **Minkowski method**:

This distance method is a generalization of both Euclidean and Manhattan distance. It can be computed as below:

$$d(x, y) = \sqrt[1/p]{\sum_{i=1}^{n}|(y_i - x_i)|^p}$$

where i is the number of features, $x=(x_1,x_2,x_3,\ldots,x_n)$ & $y=(y_1,y_2,y_3,\ldots,y_n)$ are the two point vectors in which we have to calculate the distance. p can be any value, by default it will be 1. This method can be found with **minkowski_dist** function name in attached code.

Feature scaling methods used: Mean Normalization, Min-Max Normalization

1. **Mean Normalization**:

Mean Normalization is a feature scaling method which can be calculated as below:

$$\text{Mean Normalization} = (x - mean(x))/(max(x) - min(x))$$

Where $x=(x_1,x_2,x_3,\ldots,x_n)$ is feature vector where each value of feature will be normalized. In attached code, **MeanNormalize** is the Mean Normalization function name.

2. **Min-Max Normalization**:

Min-Max Normalization is a technique which mostly brings the data towards the mean and can be formulated as below:

$$\text{Min-Max Normalization} = (\boldsymbol{x} - \boldsymbol{min}(\boldsymbol{x}))/(\boldsymbol{max}(\boldsymbol{x}) - \boldsymbol{min}(\boldsymbol{x}))$$

Where $x=(x_1,x_2,x_3,\ldots,x_n)$ is feature vector where each value of feature will be normalized. In attached code, Min-Max Normalization function is named as **normalize**.

Imputation accuracy measure used: Mean Absolute Accuracy Continuous, Mean Absolute Accuracy Categorical

1. **Mean Absolute Accuracy Continuous**:

This accuracy measure is used for calculating the accuracy for continuous features. It can be computed as below:

$$\text{Mean Absolute Accuracy} = \left|\boldsymbol{mean}\left(\mathbf{1} - \left|\frac{\boldsymbol{actual}-\boldsymbol{predicted}}{\boldsymbol{actual}}\right|\right)\right|$$

Where | (actual-predicted)/actual | is an absolute error value for continuous feature. It can be found in attached code with **calculateAccuracyContinuous** function name.

2. **Mean Absolute Accuracy Categorical**:

This accuracy measure is used in categorical feature scaling and can be computed as below:

$$\text{Mean Absolute Accuracy} = |\mathbf{1} - \boldsymbol{mean}(|(\boldsymbol{actual} - \boldsymbol{predicted}|)|$$

Where |(actual-predicted)| is an absolute error value for categorical feature. **calculateAccuracyCategorical** is the function name for this accuracy method in attached code.

Tools and libraries used for implementation:

I have used **R (3.6.1 version)** language with the **RStudio (1.2.5001 version)** IDE. No external libraries have been used in the implementation of code.

Installation instructions for R and RStudio:

To install R:

1. Open an internet browser and go to www.r-project.org.
2. Click the "download R" link under the section named "Getting Started."
3. Save the .exe file for windows and start installing it.

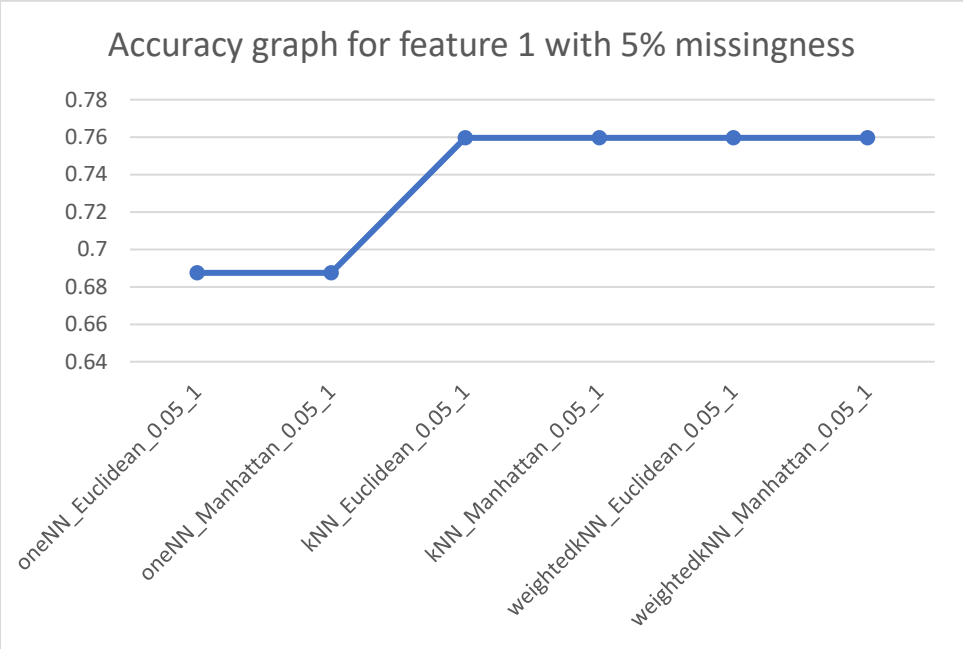
To install RStudio:

1. Go to www.rstudio.com and click on the "Download RStudio" button.
2. Click on "Download RStudio Desktop."
3. Click on the version recommended for your system.

Results: (**Note: Accuracy results are not in percentages or multiplied with 100.**)

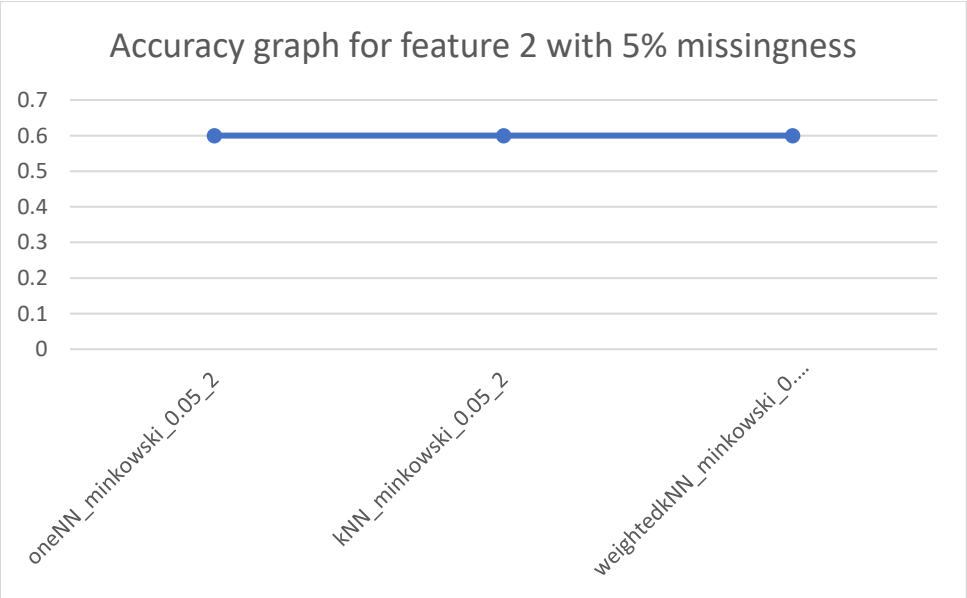
Feature1(Continuous) imputation accuracy for 5% missingness:

Continuous feature 1 with 5% missing values, 1NN has the low but same accuracy with both Euclidean and Manhattan distances that is around 0.6874. Whereas, KNN and Weighted KNN has the highest same accuracy with both Euclidean and Manhattan distance measures around 0.76.



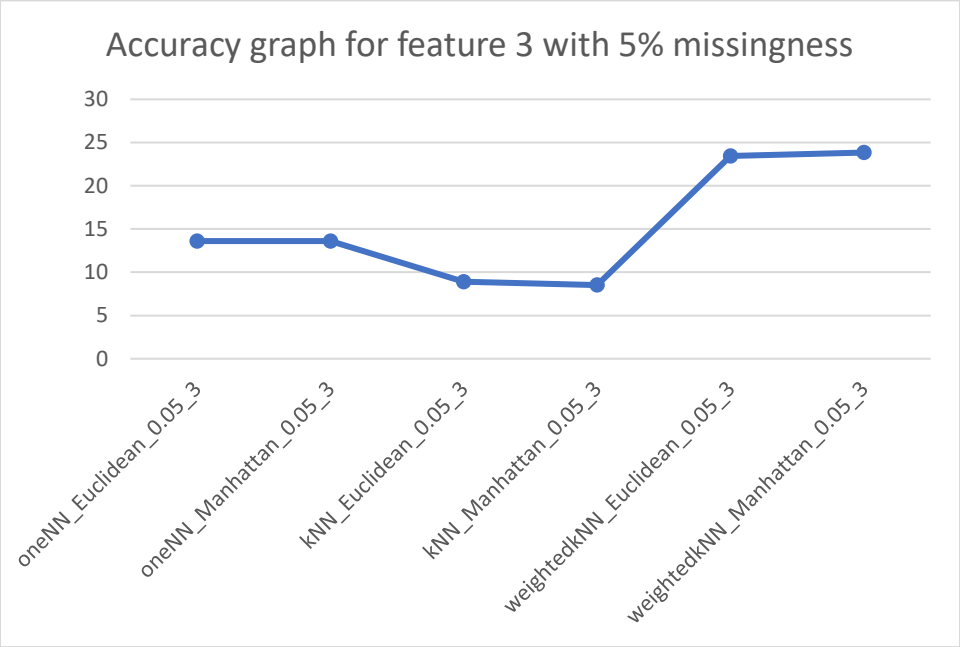
Feature2(Categorical) imputation accuracy for 5% missingness:

For categorical feature 2 with 5% missing values, we get the same accuracy over all the imputation methods (1NN, k-NN, weighted k-NN) with Minkowski distance measure (0.6).



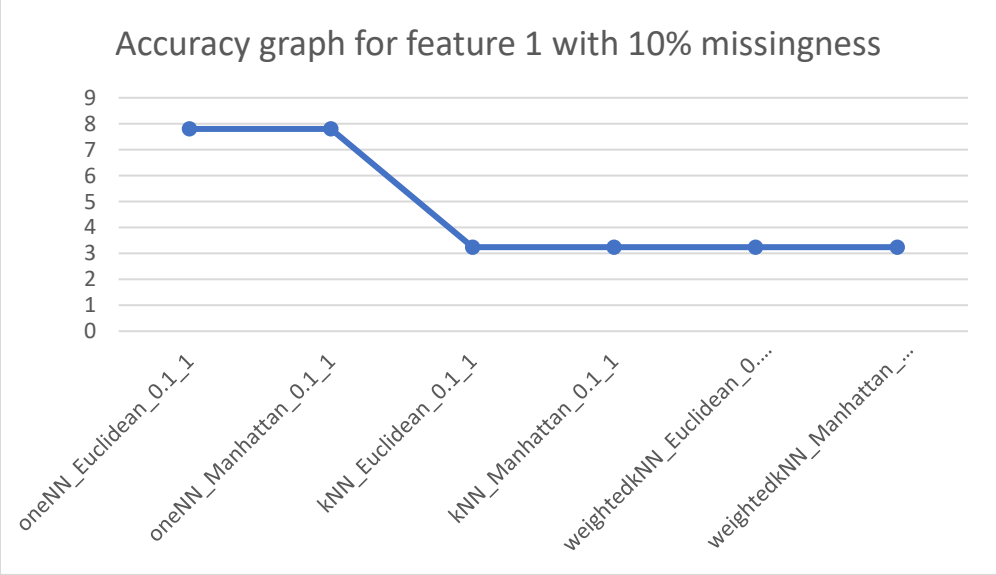
Feature 3(Continuous) imputation accuracy for 5% missingness:

For continuous feature 3 with 5% missingness, You can see the accuracy going above 1, As the data is not normalized, and this accuracy is with the raw data where we can see that k-NN has the lowest accuracy and weighted k-NN gives us higher accuracy. Yet again, we can see same accuracies over Euclidean and Manhattan distance measures.



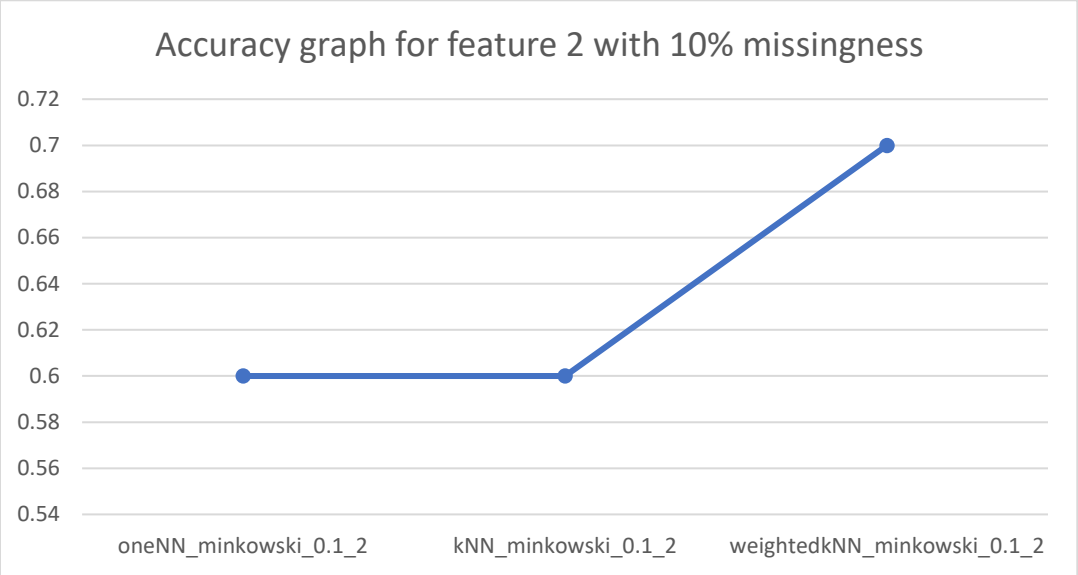
Feature1(Continuous) imputation accuracy for 10% missingness:

With 10% missingness, feature 1 shows different behavior than 5% missingness, Here we are getting high accuracy for 1NN (7.8) and low but same accuracy for k-NN and weighted k-NN(3.24) imputation methods.



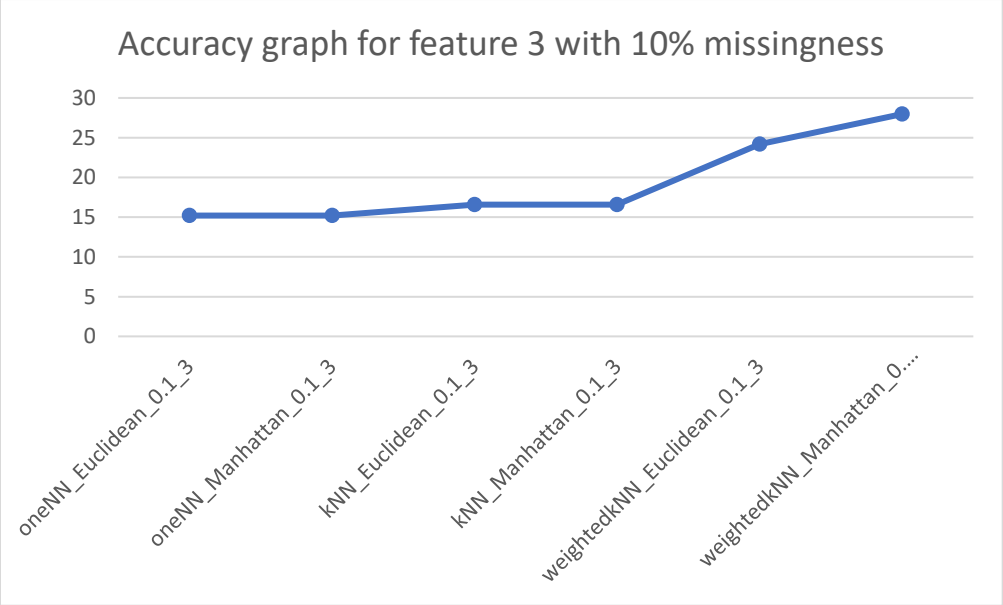
Feature2(Categorical) imputation accuracy for 10% missingness:

For 5% missingness in categorical feature, we got the same accuracy across all imputation methods (0.6), whereas here for 10% missingness, we are getting higher accuracy for weighted k-NN method with Minkowski distance measure (0.7).



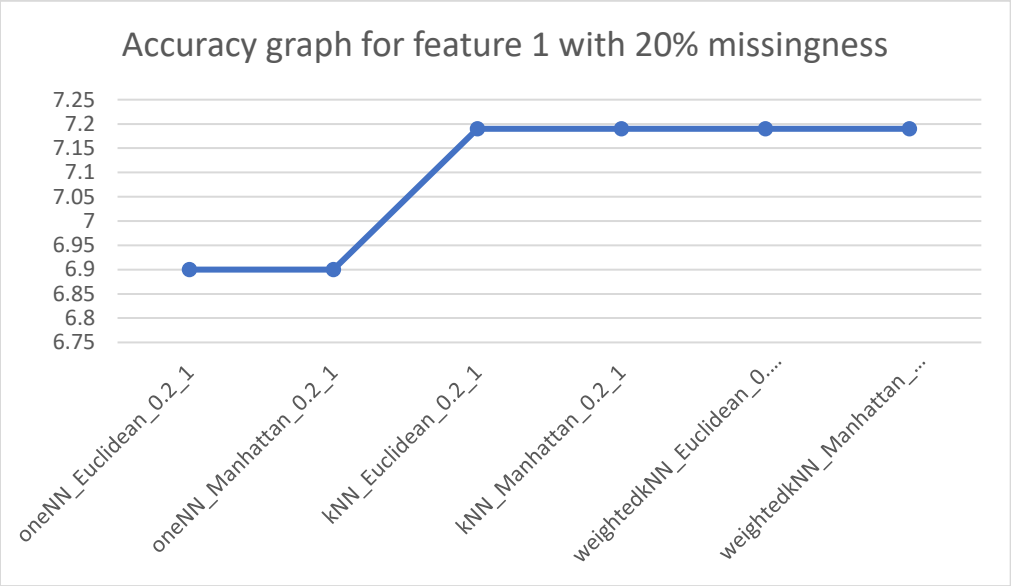
Feature 3(Continuous) imputation accuracy for 10% missingness:

With respect to 5% missingness, we got the highest accuracy in weighted k-NN imputation method, but it was same for both Euclidean and Manhattan distance measure. Whereas, for 10% missingness, we are getting the highest accuracy for weighted k-NN method Manhattan distance measure(27.97). And, here k-NN has the improved accuracy than 1NN which was not the case in 5% missingness.



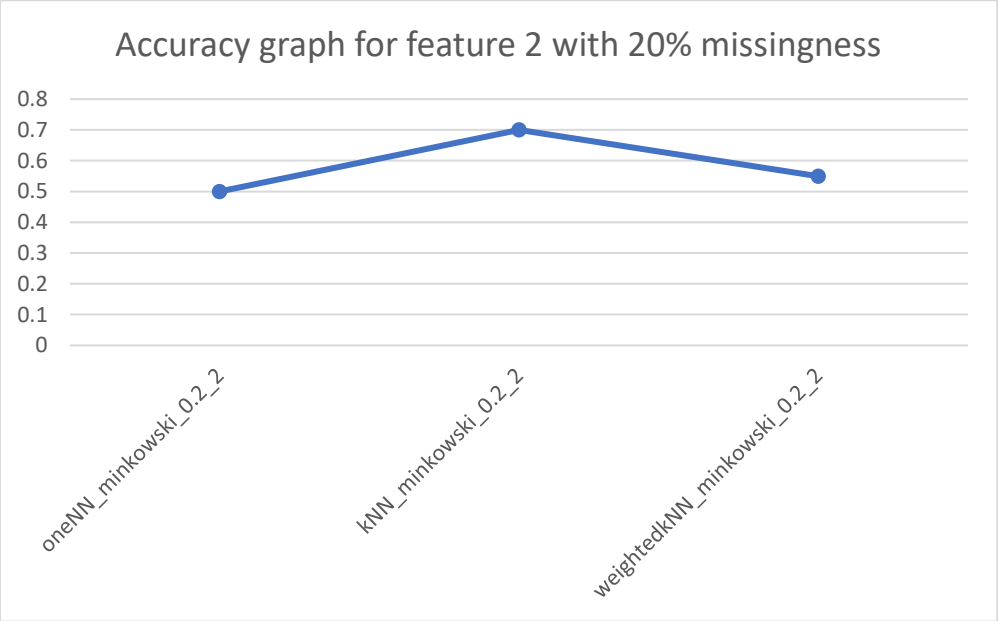
Feature 1(Continuous) imputation accuracy for 20% missingness:

With 20% missingness, k-NN and weighted k-NN imputation methods gives the same accuracy (7.19). Here, k-NN and weighted k-NN performs better than 10% missingness. Original values are used in raw format, which are resulting in accuracy more than 1.



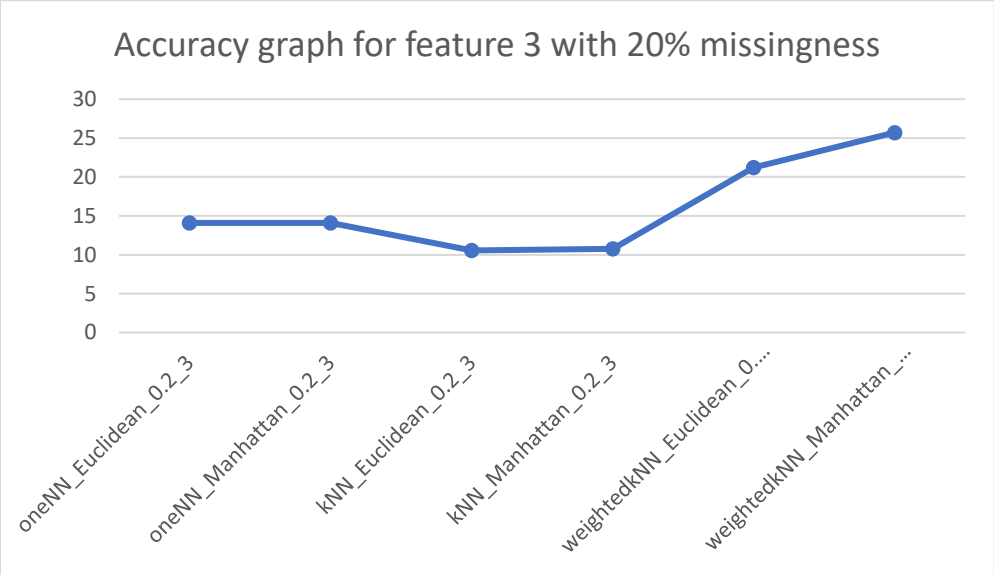
Feature2(Categorical) imputation accuracy for 20% missingness:

For categorical feature, graph shows that k-NN imputation method performs better than other imputation methods as well as k-NN performs better than 10% and 5% missingness. Plus, weighted k-NN in 10% and k-NN in 20% gives the same accuracy (0.7).



Feature 3(Continuous) imputation accuracy for 20% missingness:

For continuous feature 3, weighted k-NN with Manhattan distance measure has performed better than other imputation methods which was the same case in 10% missingness. Whereas, here we can see the same trend as 5% missingness.



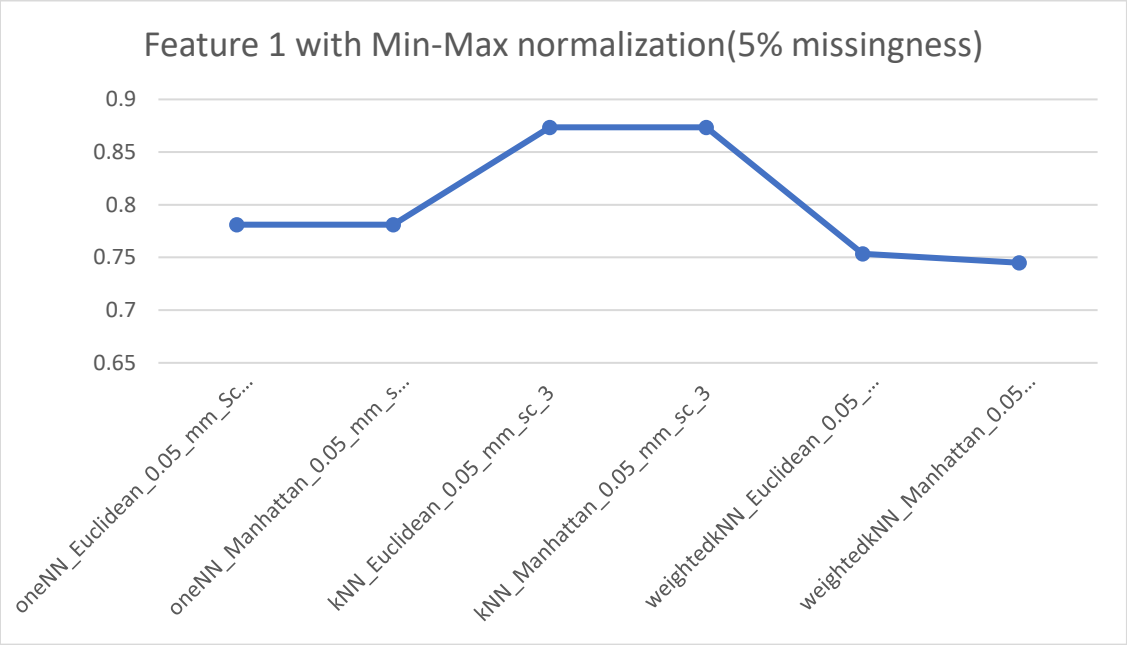
Feature 1(Continuous) imputation accuracy for 5%, 10%, 20% missingness with Min-Max Normalization and Mean Normalization:

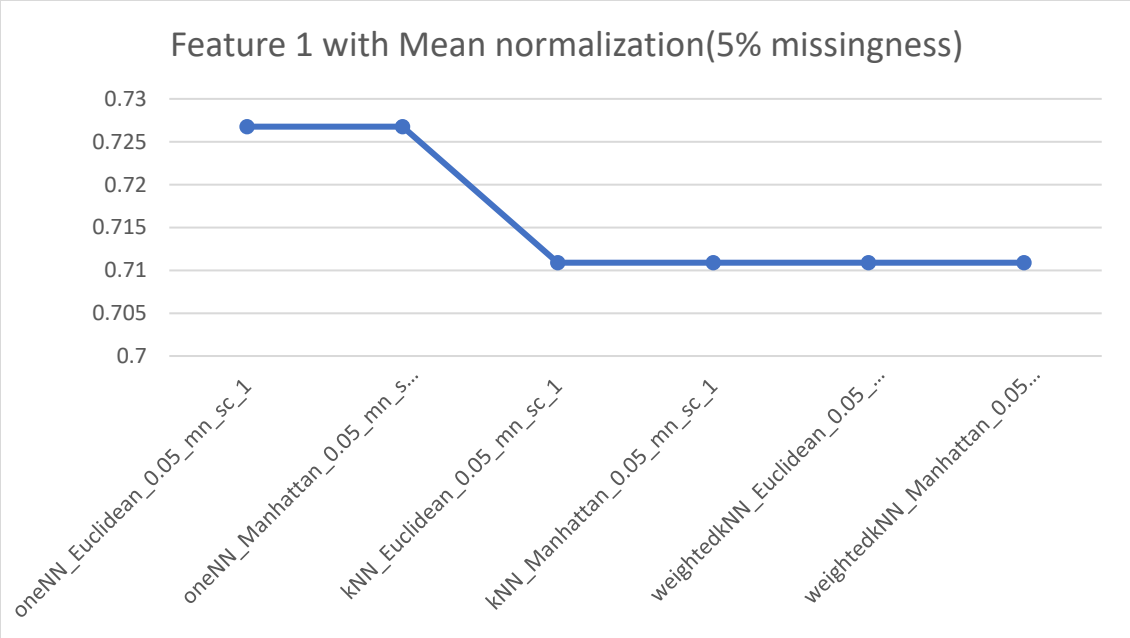
For continuous feature 1 with all percentage missingness, we get two different leanings for two different normalization method.

For Min-Max Normalization with 5%, 10% and 20% missingness, k-NN has performed pretty good than other two imputation methods and it is following the same accuracy trend in all missingness. While, for Mean Normalization feature scaling method with 5% missingness give totally contradictory results to 10% and 20% missingness. In 5% missingness, 1-NN performs better than other imputation methods, however 10% and 20% missingness follows the same trend by giving out higher accuracy for k-NN and weighted k-NN imputation method. All imputation methods have performed with Euclidean and Manhattan distance measure.

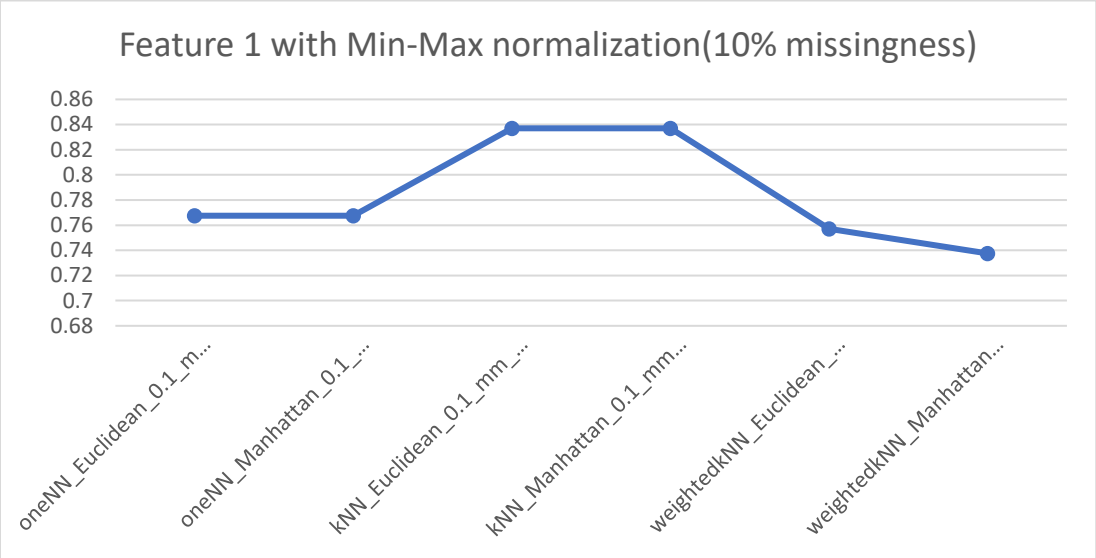
And we can see that, after normalizing the original data and finding distances with scaled values have achieved better accuracy than using the original values for distance measurement.

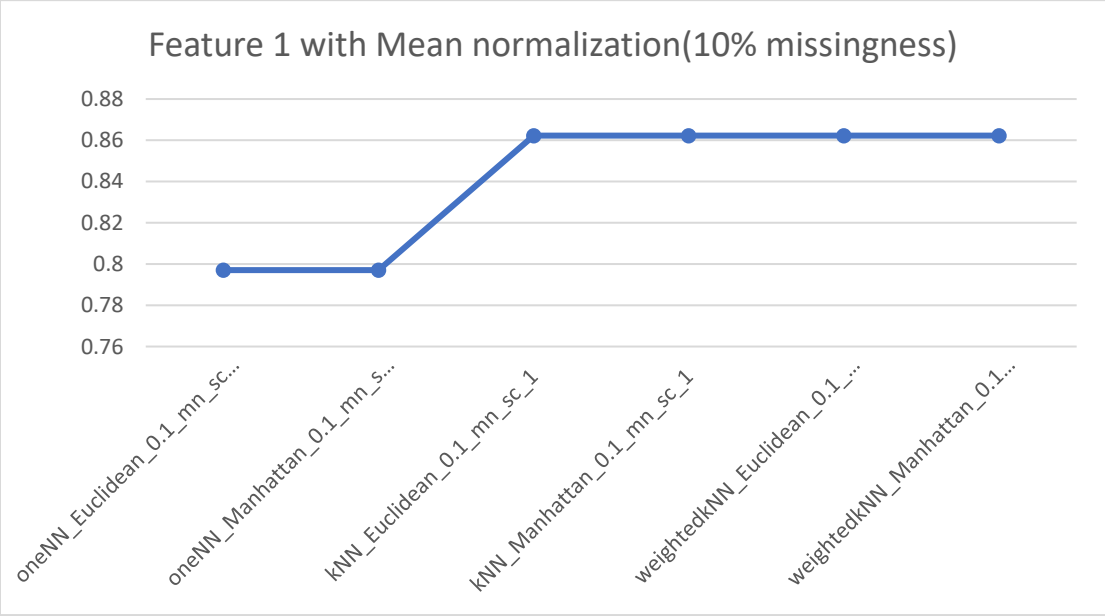
Feature Scaling for 5% missingness for continuous feature 1:



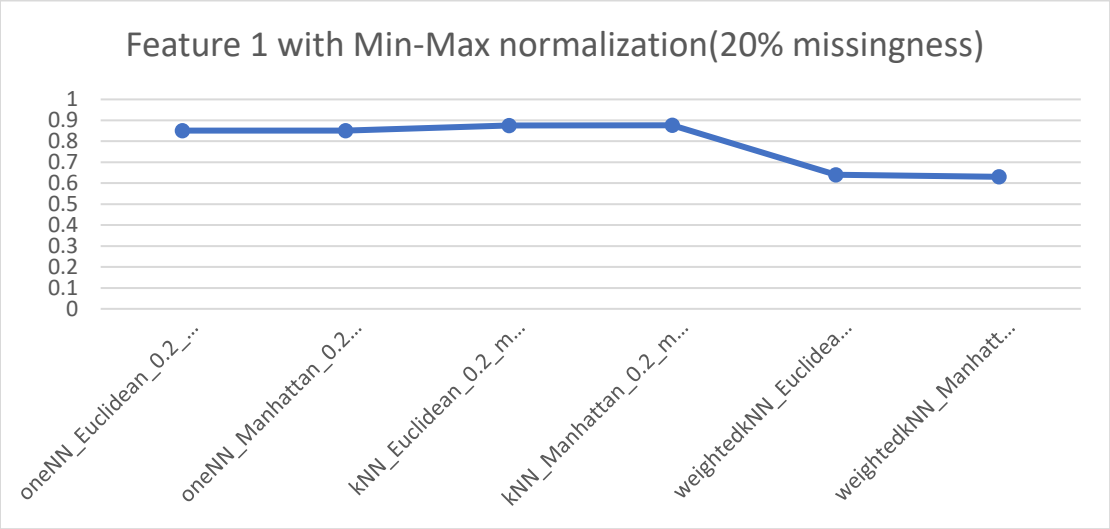


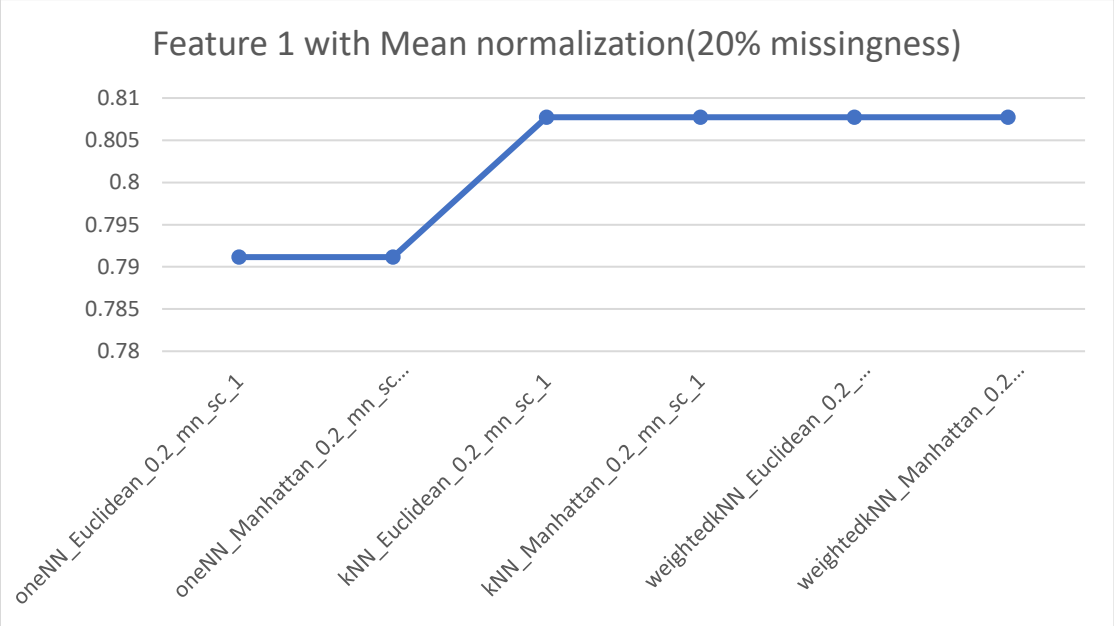
Feature scaling for 10% missingness for continuous feature 1:





Feature scaling for 20% missingness for continuous feature 1:





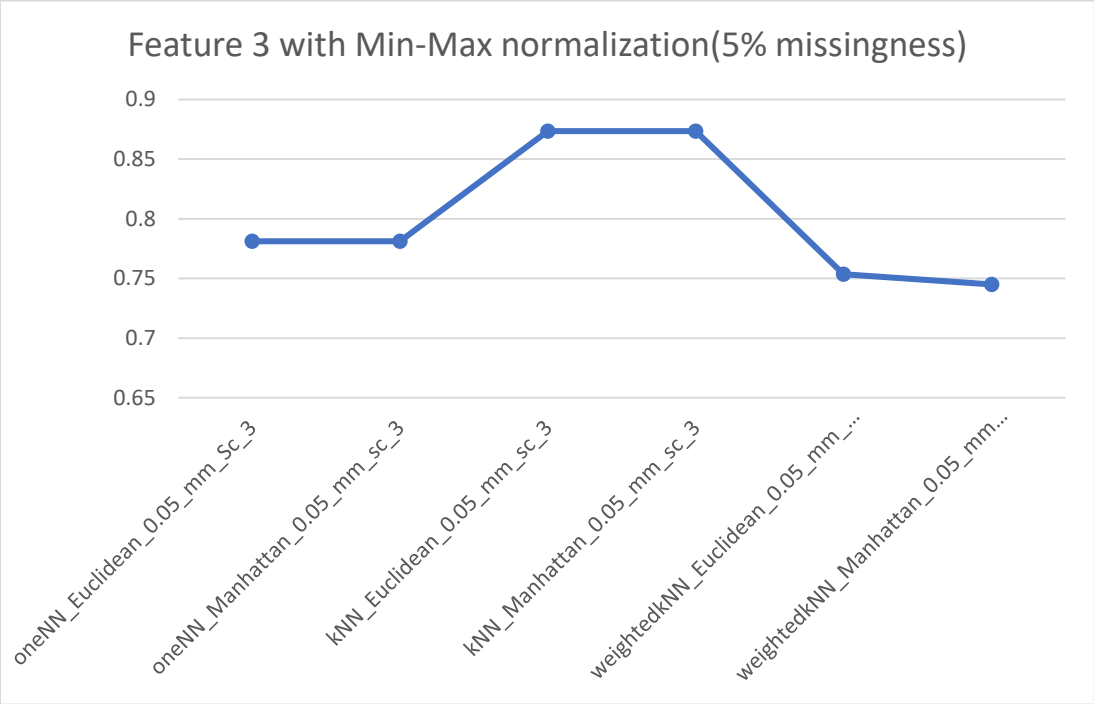
Feature 3(Continuous) imputation accuracy for 5%, 10%, 20% missingness with Min-Max Normalization and Mean Normalization:

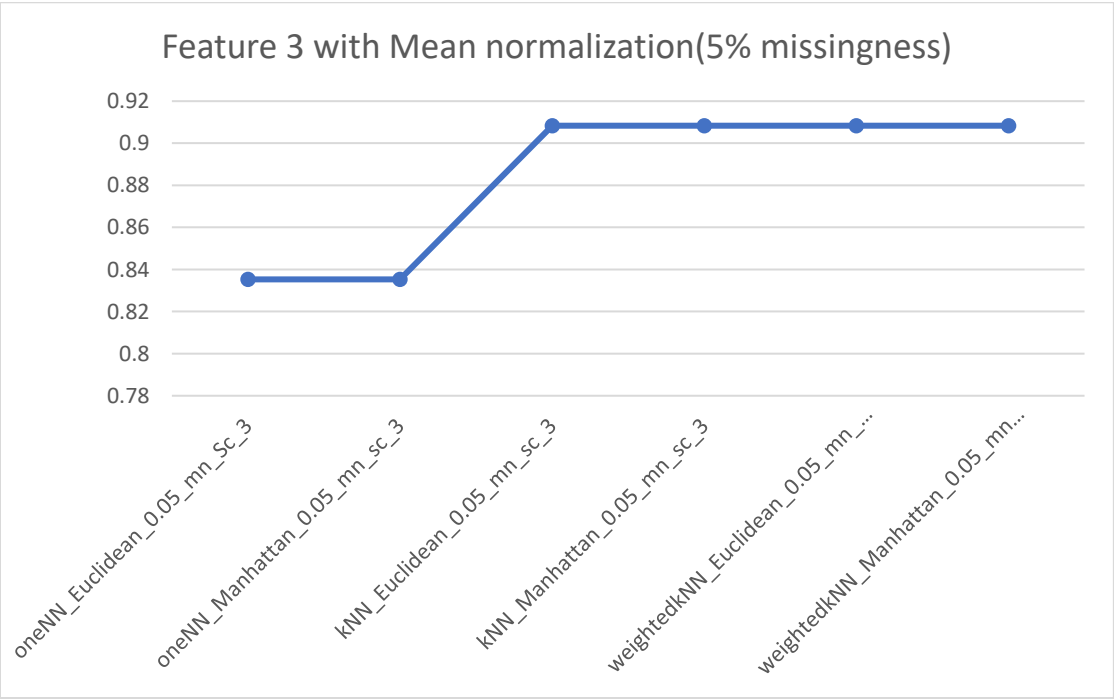
For continuous feature 3, we can see that it has the same accuracy curve as feature 1 for both scaling normalizations. But, here for two different normalizations, there is a same trend throughout all missingness.

k-NN imputation method has performed better in Min-Max Normalization while, k-NN and weighted k-NN imputation method has shown better and same performance in Mean

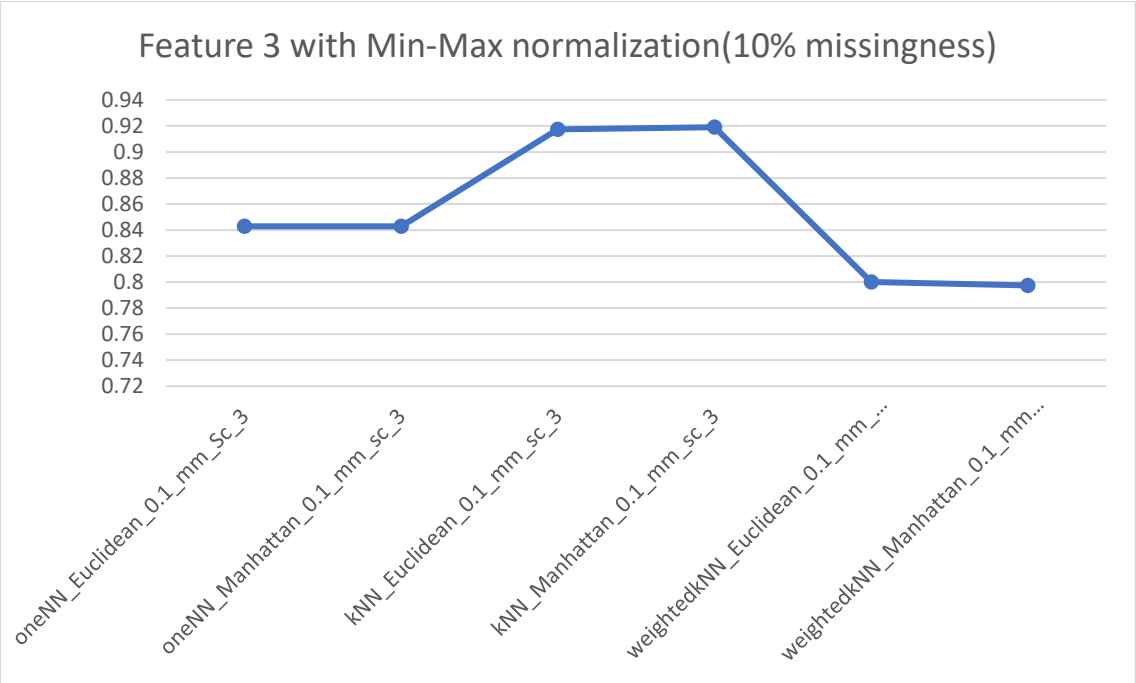
Normalization. 1-NN and weighted k-NN shows the lowest accuracy in Mean Normalization and Min-Max Normalization feature scaling method respectively.

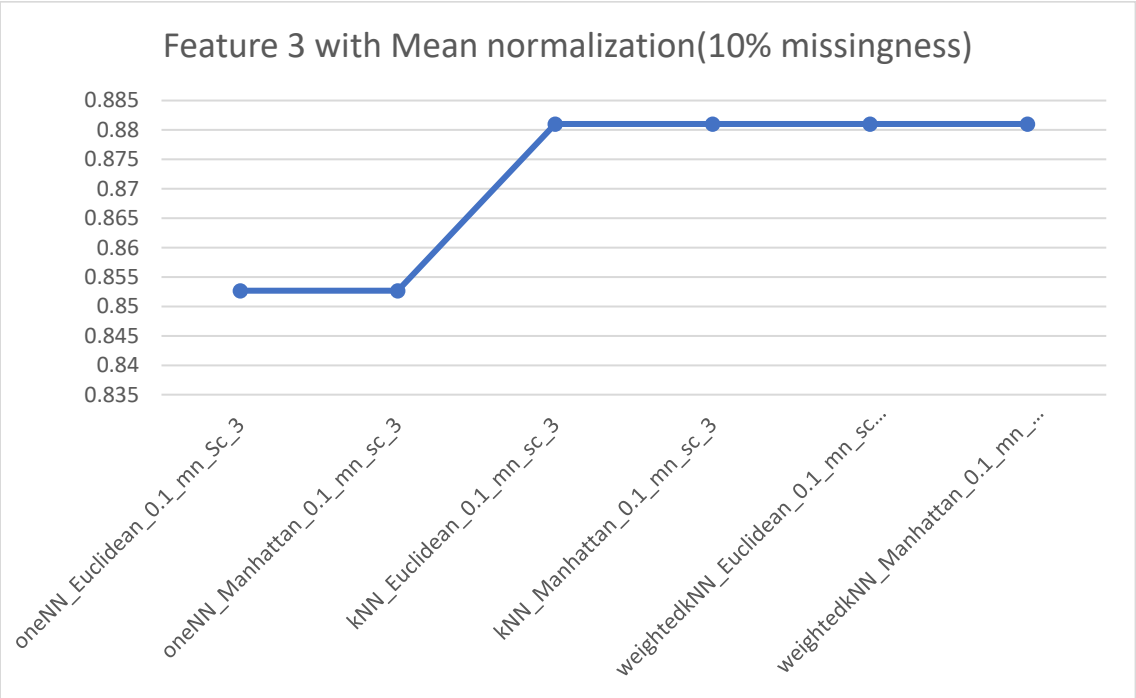
Feature scaling for 5% missingness for continuous feature 3:



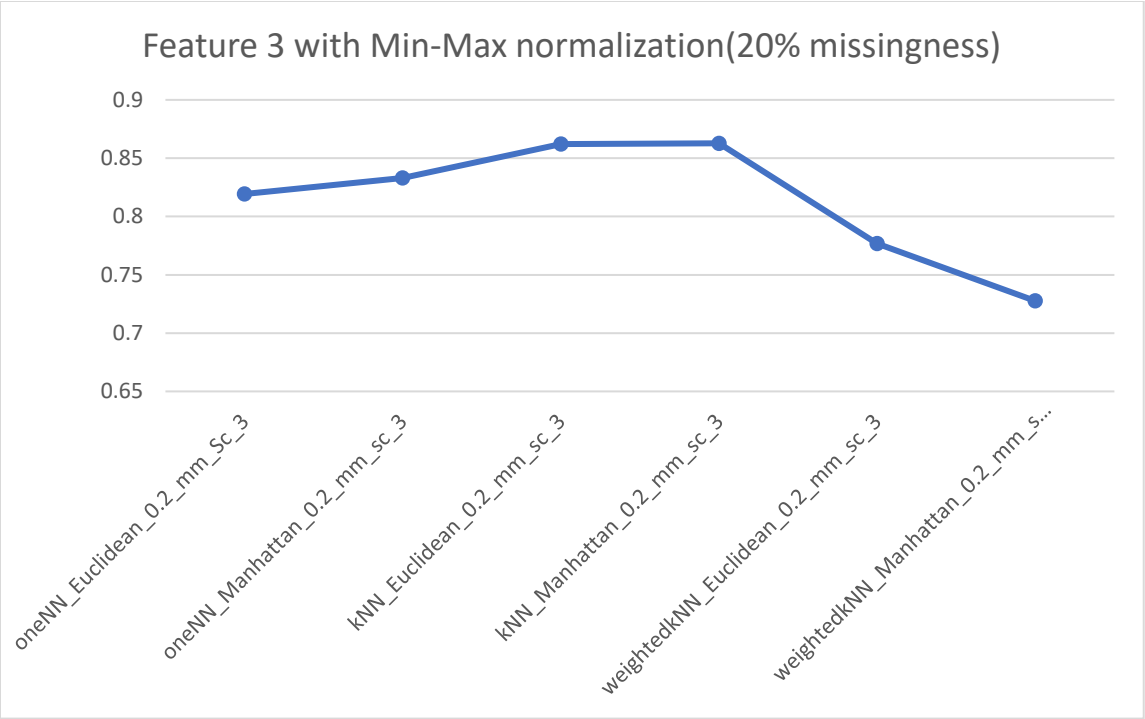


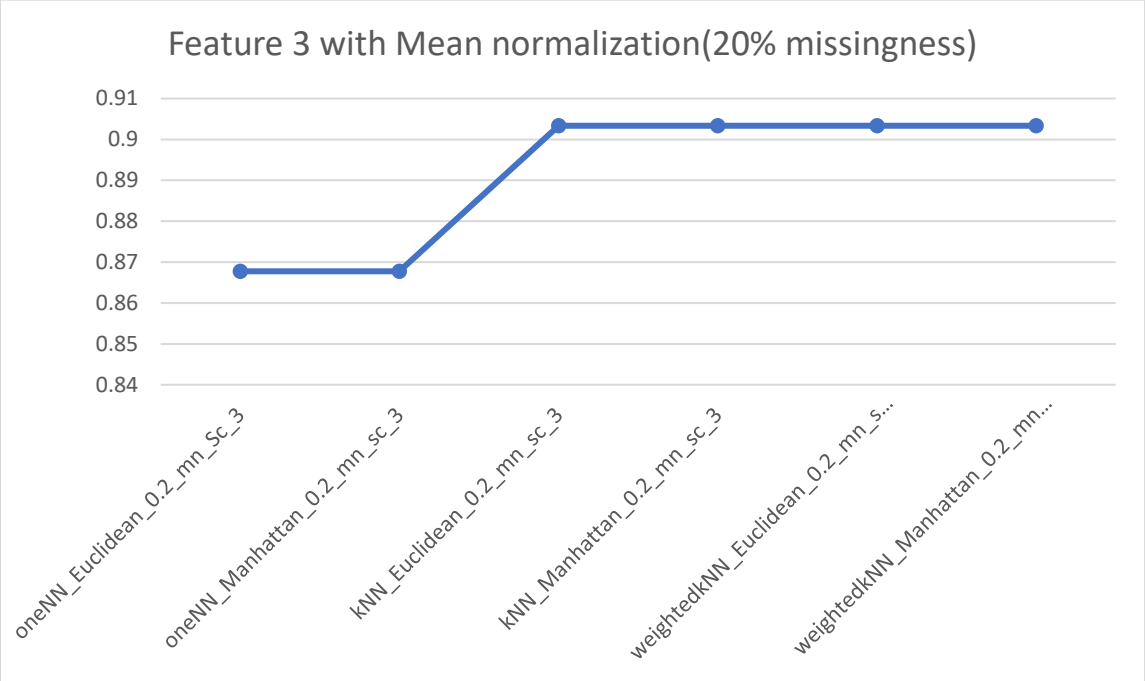
Feature scaling for 10% missingness for continuous feature 3:





Feature scaling for 20% missingness for continuous feature 3:





How to run an attached program:

1. Kindly Enter an absolute file path for downloaded csv file.
2. To run the program: kindly run each line till the end which can be done by using “Ctrl+Enter” on each line.
3. You don't have to change your cursor position while using this shortcut.
4. Just keep on pressing “Ctrl+Enter” at the same time.
5. Result accuracy values may vary, as random missing values are generated while running a program every-time.