# INCOMPLETE CASE K NEAREST NEIGHBOUR IMPUTATION [ICKNNI] AND COMPLETE CASE K NEAREST NEIGHBOUR IMPUTATION [CCKNNI]

Sariat Sultana (110028032), Arulpiruthiviraj Arunthavaraja (105212573), Baranidharan Pasupathi (110007279)

**Abstract –** The project aims to evaluate the imputed missingness of ICKNNI and CCKNNI and to give an outset of ICKNNI and its importance over traditional CCKNNI where imputations occur with minimum missing values. It is observed that including some incomplete instances for imputation would provide persuasive unbiased solutions for analysis. This report explains the steps and complexity computation of both imputation techniques and their implementations.

**Notation Usage** - Input dataset $X$; instances $(x_i = x_{i1},\ x_{i2}, \dots, x_{in})$; variables $m\&i$ similarly $n\&j$ are used interchangeably; missing variables $x_i^{mis}$; observed variables $x_i^{obs}$). Let $C$ set of complete instances $(C = \{\, x_i^{com} \in X\}$, where $x_i^{com}$ are complete instances; $M$ set of incomplete instances $(M = \{\, x_i^{incom} \in X\}$, where $x_i^{incom}$ is incomplete instances; $(M = X \backslash C)$; $d_k$ the shortest distance between $x_i^{incom}, x_i^{com}$; $\hat{X}$ modified/output dataset.

## 1. INTRODUCTION

The real-world datasets always come with missing values that create a huge problem in analyzing the data. It is nearly impossible to impute the actual data with any algorithms, therefore, the appropriate algorithms based on the pattern and volume of the missingness are being implemented to obtain the missing values closest to the actual data. Moreover, regression techniques cannot be applied to the data set with missing values, so it is crucial to impute the missing values and then apply regression models to deal with the data.

There are several techniques to obtain missing values. The simplest method is Listwise Deletion (LD), where all instances with missing values are eliminated to secure a complete dataset. This method poses a great drawback, it loses its significant information and analysis results in a biased incomplete manner which is intolerable [1].

There is other numerous alternatives method for imputation, one of which is *k* nearest neighbor imputation (KNNI). KNNI is a type of instance-based learning mechanism which stores all available cases and defines new cases based on similarity measures, such as distance function. There are several ways to implement KNNI, one of them mainly deals with the missing values based on the complete cases in the dataset, which is known as Complete Case *k* Nearest Neighbor Imputation (CCKNNI). This results in biased imputation at times and causes a significant problem where the volume of missingness is high. There it loses its integrity and information as it deals with only complete cases. The alternative method is Incomplete Case *k* Nearest Neighbor Imputation (ICKNNI). This mechanism deals with all the instances, both complete and incomplete, and imputes the missing values based on the nearest distance. The major issue with this method is that it is quite complex and is not readily available yet [1].

## 2. K NEAREST NEIGHBOR IMPUTATION (KNNI)

KNNI is considered to be the lazy learning algorithm as it tries to generalize the data before receiving queries. It is a non-parametric method used for classification and regression. It predicts the class to which the output variable belongs

by computing the local probability. KNNI predicts the value of the output variable by using a local average. In numerical dataset the shortest distances are measured using three methods [3]:

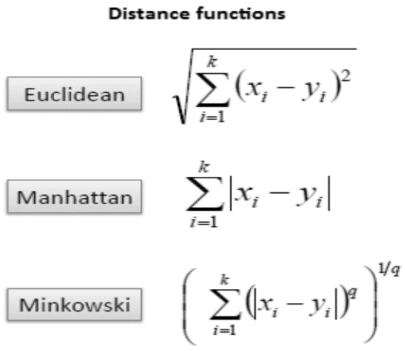



Fig.1. Distance functions to measure the shortest distance between neighboring values [3].

The Euclidean distance method is mostly used for CCKNNI and ICKNNI.

## 3. COMPLETE CASE K NEAREST NEIGHBOR IMPUTATION (CCKNNI)

In CCKNNI, the missing values are imputed based on the complete cases of the datasets. Firstly, the complete cases are determined relative to the missing instance. Secondly, a set of *k* nearest neighbors is found. Thirdly, the distance between the *k* nearest neighbors and the complete instances are measured by using the Euclidean distance formula. Fourthly, each missing value is imputed by taking the average of the k nearest neighbor with the shortest distances. Finally, the missing values are replaced by the imputed values, thus returning an imputed complete dataset.

## 4. MAIN STEPS OF COMPLETE CASE K NEAREST NEIGHBOUR IMPUTATION

**Step-1:** Initialize $x_i^{incom}$ from $(x_i = x_{i1},\ x_{i2}, \dots, x_{in})$ and load it to $M$ i.e., $x_i^{incom} \in M$[1].

**Step-2:** Initialize $x_i^{com}$ to $C$ and find $d_k = d(x_i^{incom}, x_i^{com})$;

$d_k = \sqrt{\sum_{o=1}^{nobs}(x_{io}^{incomobs} - x_{io}^{comobs})\text{^}2}$; $x_i^{incomobs}$ and $x_i^{comobs}$ are observed variables of incomplete and complete instances; $o = 1,..,nobs$; $nobs$ represents the number of observed variables in an instance;

**Step-3:** Find complete nearest neighbor $K = |K_i|$;

$$K_i = \{x_i^{com} \in C\ for\ d_k \leq d_q \forall x_q \notin K_i\}$$

**Step-4:** For $m = 1,..,nmiss$; $nmiss$ is number of missing variables in an instance; $\hat{x}_{im}^{incommiss}$ imputes missing values;

$\hat{x}_{im}^{incommiss} = k_i^{-1} \sum_{x_p \epsilon K_i} x_{pm}^{miss}$;

**Step-5:** Imputed values $\hat{x}_{im}^{incommiss}$ and $x_{io}^{incomobs}$ observed are stored in a new instance $\hat{x}_i^{incom}$;

$\hat{x}_i^{incom} = (x_{i1}^{incommiss}, \dots, x_{im}^{incommiss}, x_{i1}^{incomobs}, \dots,\ x_{io}^{incomobs})$;

**Step-6:** $\hat{x}_i^{incom}$ is then equated to the variable $\widehat{M}$;

$\widehat{M} = \{\hat{x}_i^{incom}\}$, since $x_i^{incom} \in M$. Finally, $\hat{X} = C \cup \widehat{M}$

## 5. INCOMPLETE CASE K NEAREST NEIGHBOR IMPUTATION (ICKNNI)

The ICKNNI algorithm is performed in all nearest neighbors. It is relaxed in the sense of completeness of the attributes. Firstly, the nearest neighbors are determined that have the same subset as the observed attribute. Then a set of k nearest neighbors is found. Therefore, each missing value has a different set of k nearest neighbors. Finally, after determining

the nearest neighbors, the steps are carried out in the same manner as the CCKNNI. Therefore, the only difference is in the step of determining the set of nearest neighbors.

## 6. MAIN STEPS OF INCOMPLETE CASE K NEAREST NEIGHBOUR IMPUTATION

**Step-1:** Initialize $x_i^{incom}$ from $(x_i = x_{i1},\ x_{i2}, \dots, x_{in})$ and load it to $M$ i.e., $x_i^{incom} \in M$; Compute $x_{im}^{incommiss}$, $m = 1, .., nmiss$; [2]

**Step-2:** Compute $O_i = \{x_1^{obs}, x_2^{obs} \dots, x_i^{obs}\}$; $O_i$ is a set of observed values of all instances. $T_{im}^{incommiss} = \{x_i \in X | \{O_i^{incom} \cup x_m\} \subset O_i\}$; $T_{im}^{incommiss}$ is a library that
holds all possible neighbor instances of $x_i^{incom}$, later their distances $d_i$ is computed.

**Step-3:** Since $T_{im}^{incommiss}$ holds all instances obviously $\forall x_i \in T_{im}^{incommiss}$; Find $d_i = d\left(x_i^{incom}, x_i\right)$;

$$d_i = \sqrt{\sum_{o=1}^{nobs} (x_{io}^{incomobs} - x_{io})\text{^}2};$$

**Step-4:** Find complete nearest neighbor $K = |K_{im}|$;

$$K_{im} = \{x_i^{com} \in C\ for\ d_k \leq d_q \forall x_q \notin K_{im}\}$$

**Step-5:** $\hat{x}_{im}^{incommiss}$ predicts values of missing one; $\hat{x}_{im}^{incommiss} = k_i^{-1} \sum_{x_p \epsilon K_i} x_{pm}^{miss}$; [1]

**Step-6:** Step-5 and Step-6 of Complete case K nearest neighbor is followed to derive a complete dataset.

## 7. EXPERIMENTAL DESIGNS

This section demonstrates the detailed experimental results of both ICKNNI and CCKNNI algorithms. The real-world datasets are used to impute missing values and then compared with the originally given values. The process has been repeated several times to obtain the most accurate imputed results.

### *I. COMPLEXITY ANALYSIS*

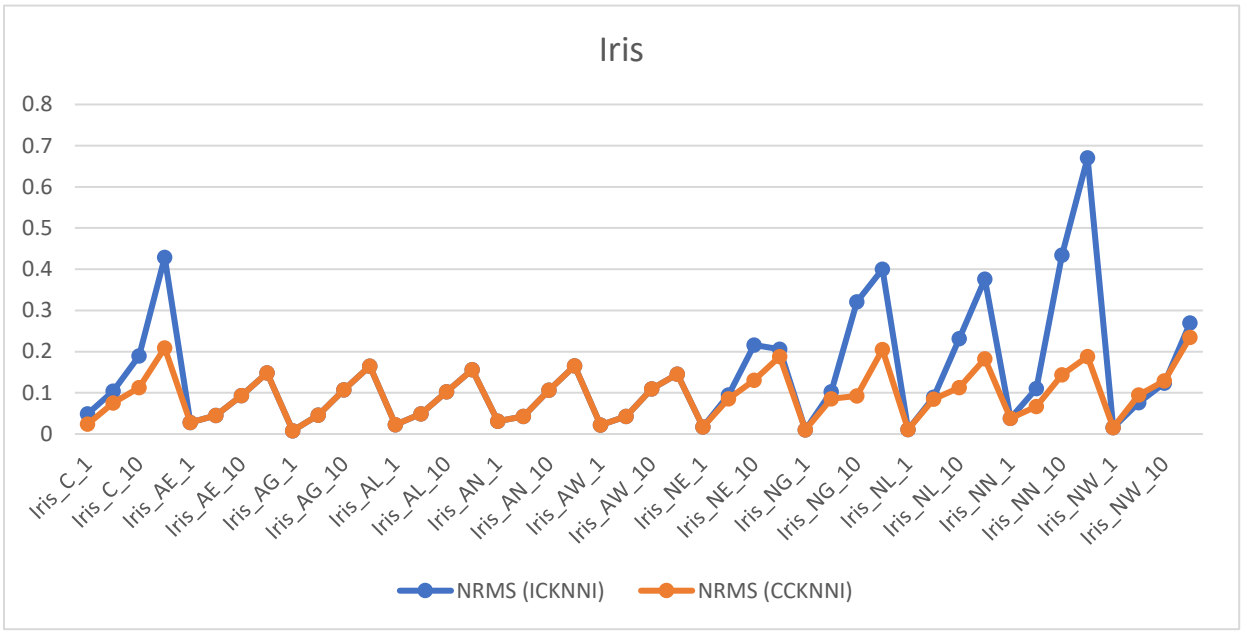
Considering dataset X with $mXn$ records & attributes. The computational complexity of KNN impute method is approximate $O(m^2 n)$, assuming $m \gg k$. The execution of both cases will be almost similar since they follow the KNN procedure. In ICKNNI where the maximum percent of missing cases are considered, there might be a slightly longer execution is expected while in CCKNNI it's not. In-reality software empirical analysts use list deletion LD in CCKNNI datasets to avoid biased output which will make the execution time much faster when compared with ICKNNI.

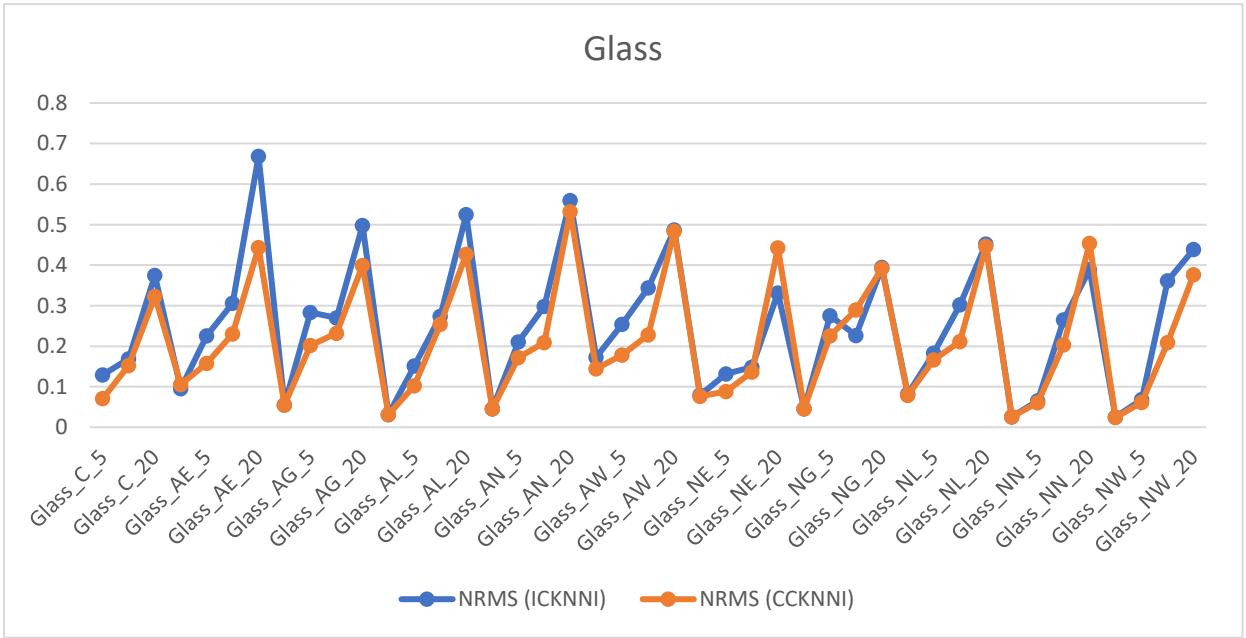
### *II. PERFORMANCE MEASURE*

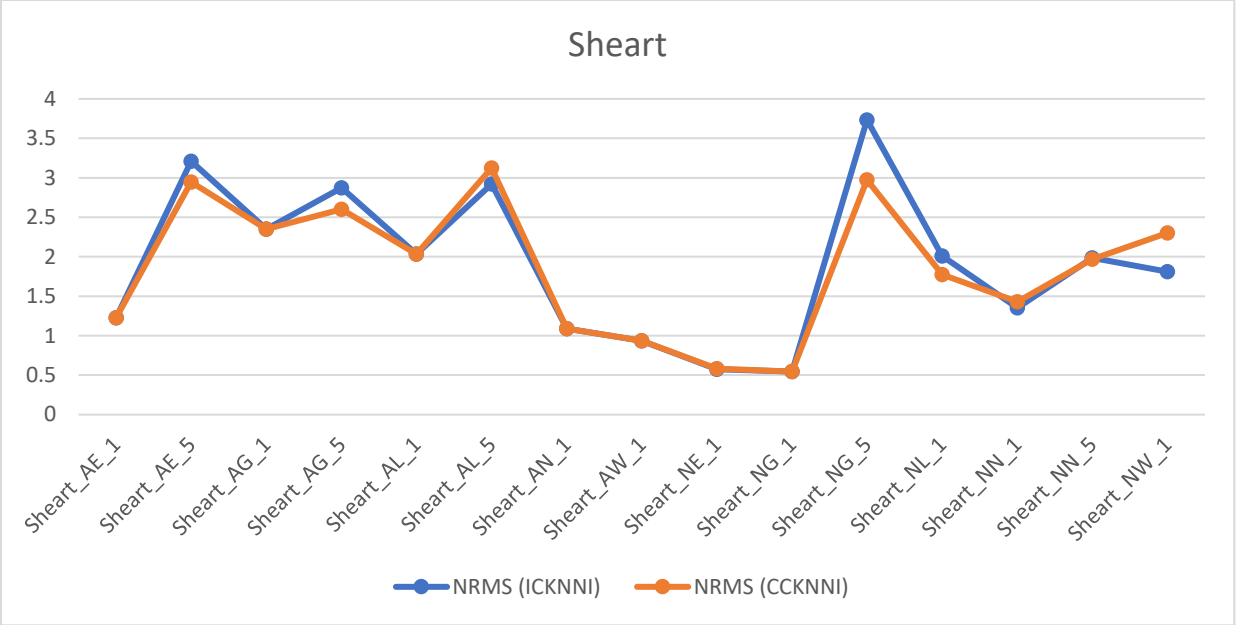
The effectiveness and the imputation performance are measured utilizing NRMS which stands for Normalized Root Mean Square. It is a transformation of the coefficient of variance. The purpose of normalizing the data is to allow a direct comparison of two variables that exist on different scales so that the magnitude of errors can be compared more meaningfully. The NRMS values for both ICKNNI and CCKNNI are measured for each dataset to compare its implementation for varying volumes of missingness. Moreover, average ICKNNI NRMS and CCKNNI NRMS are measured to compare the overall utilization of the algorithms.

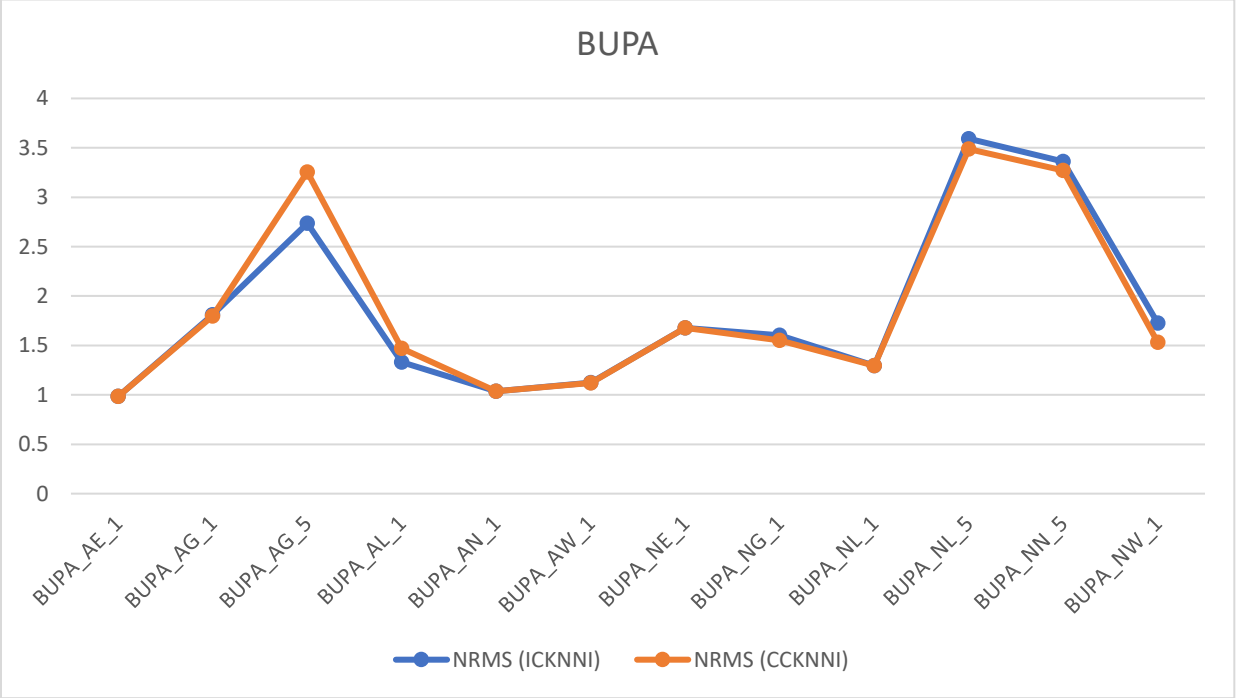
## 8. RESULTS

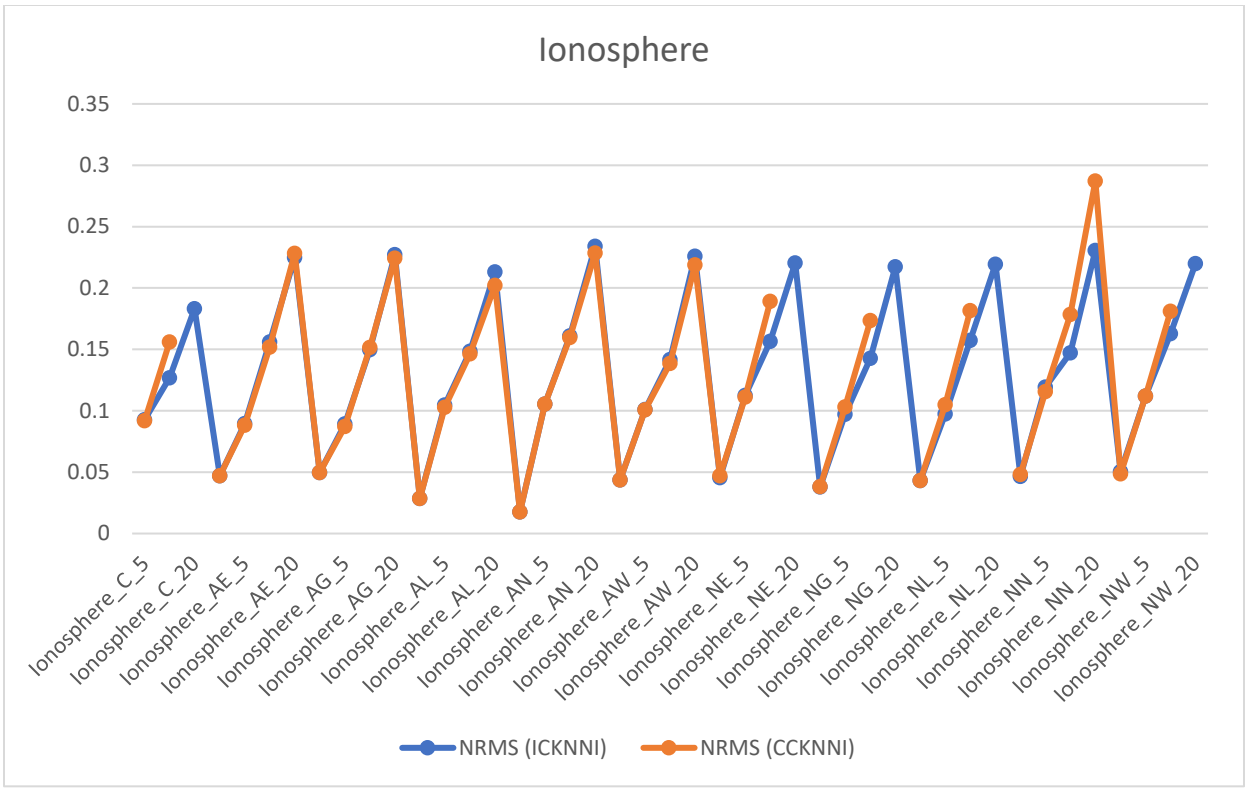
All the results are provided in this section. Figure 2 shows the imputation comparison between CCKNNI and ICKNNI for every numerical dataset. The *x*-axis represents the parameters and the *y*-axis represents the values of the NRMS. The orange and blue lines with the markers show the values of NRMS for CCKNNI and ICKNNI respectively. Figure 3 demonstrates the effectiveness of the imputation algorithms by comparing the average NRMS of both ICKNNI and CCKNNI in a clustered bar chart.

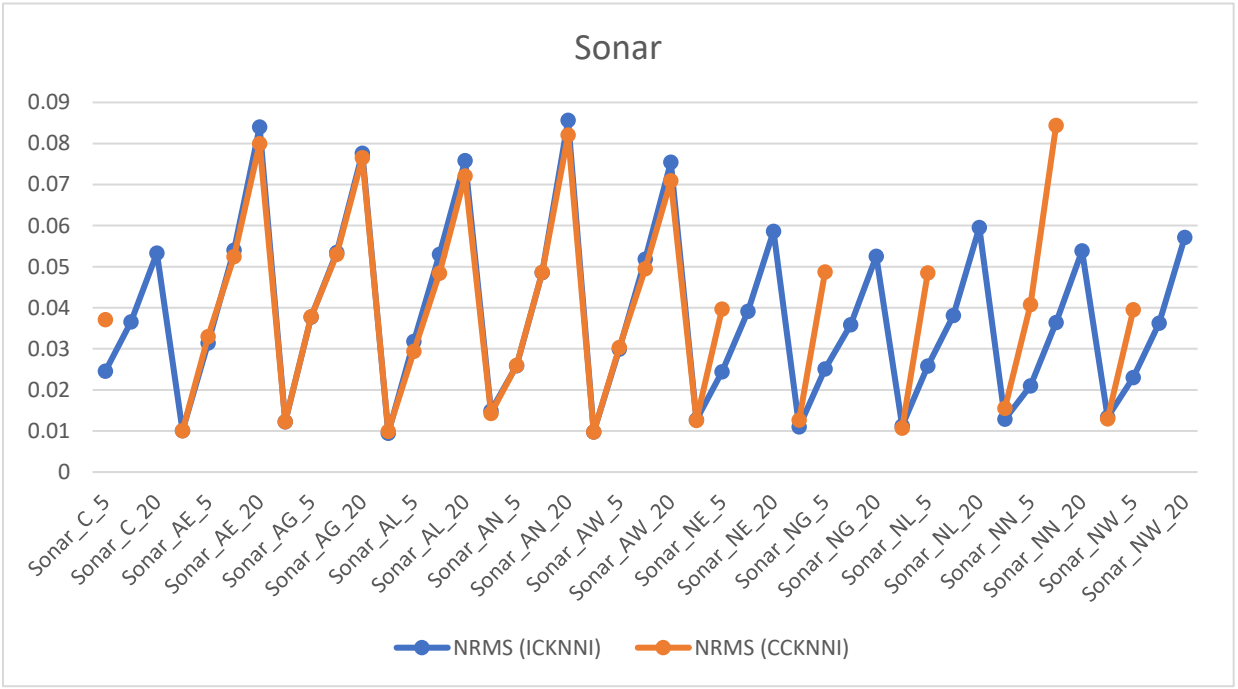


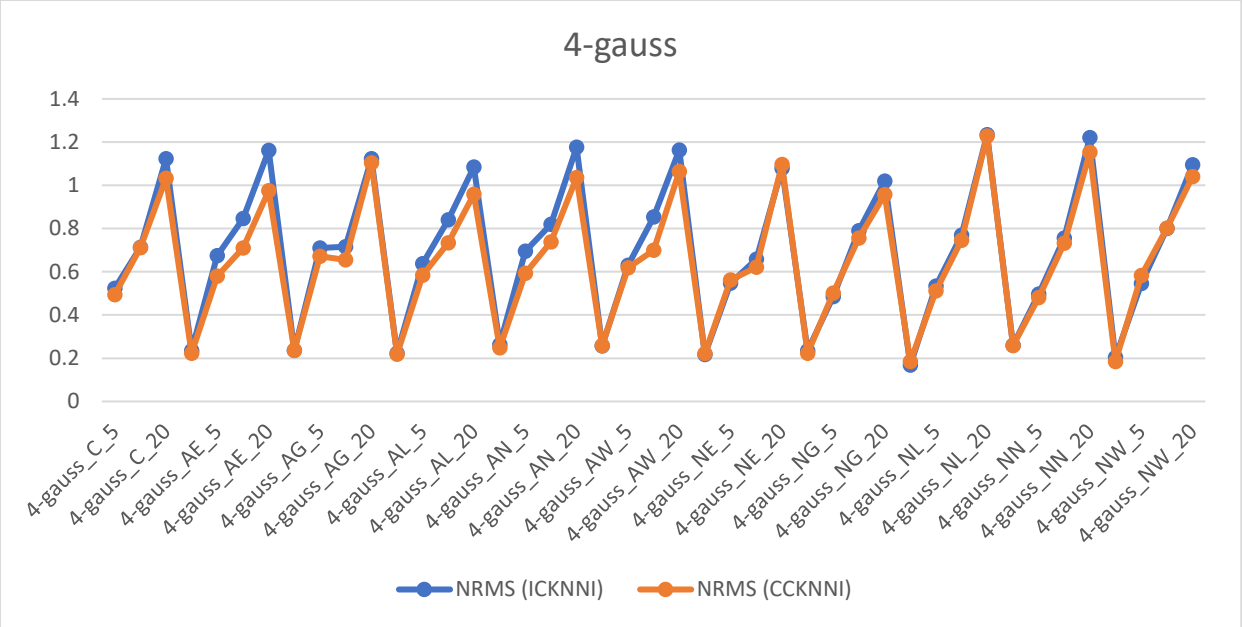



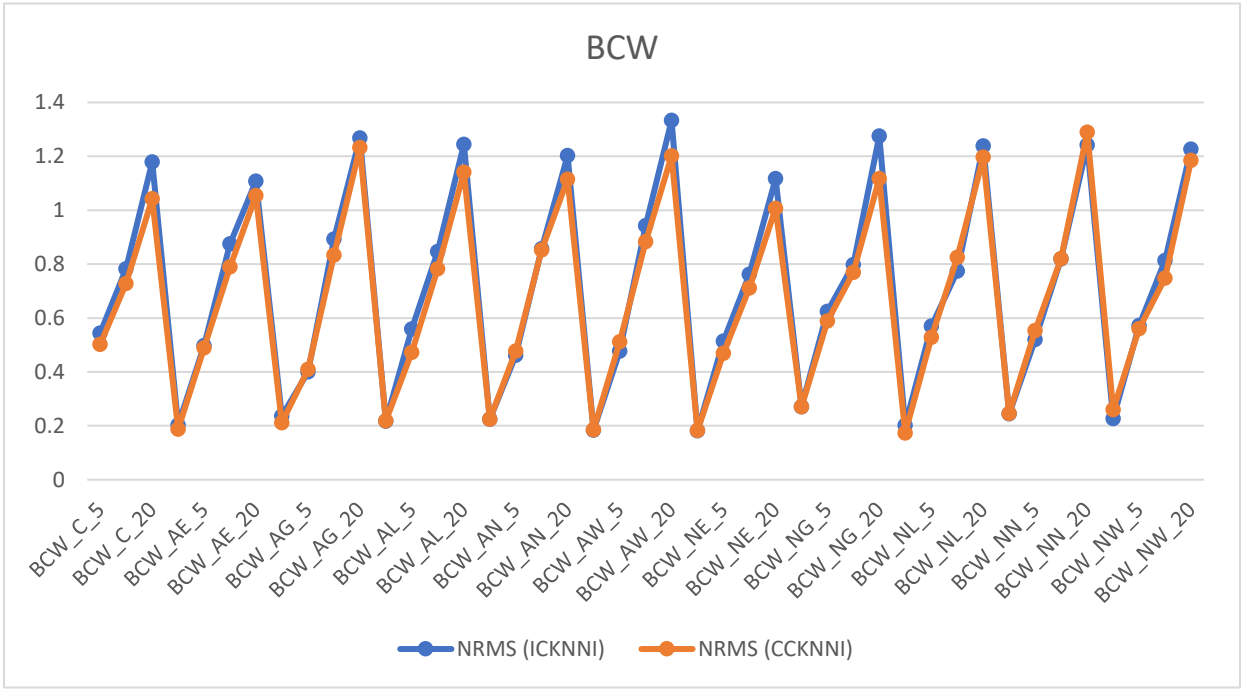

Sheart
4
3.5
3
2.5
2
1.5
1
0.5
0
Sheart_AE_1
Sheart_AE_5
Sheart_AG_1
Sheart_AG_5
Sheart_AL_1
Sheart_AL_5
Sheart_AN_1
Sheart_AW_1
Sheart_NE_1
Sheart_NG_1
Sheart_NG_5
Sheart_NL_1
Sheart_NN_1
Sheart_NN_5
Sheart_NW_1
NRMS (ICKNNI)
NRMS (CCKNNI)

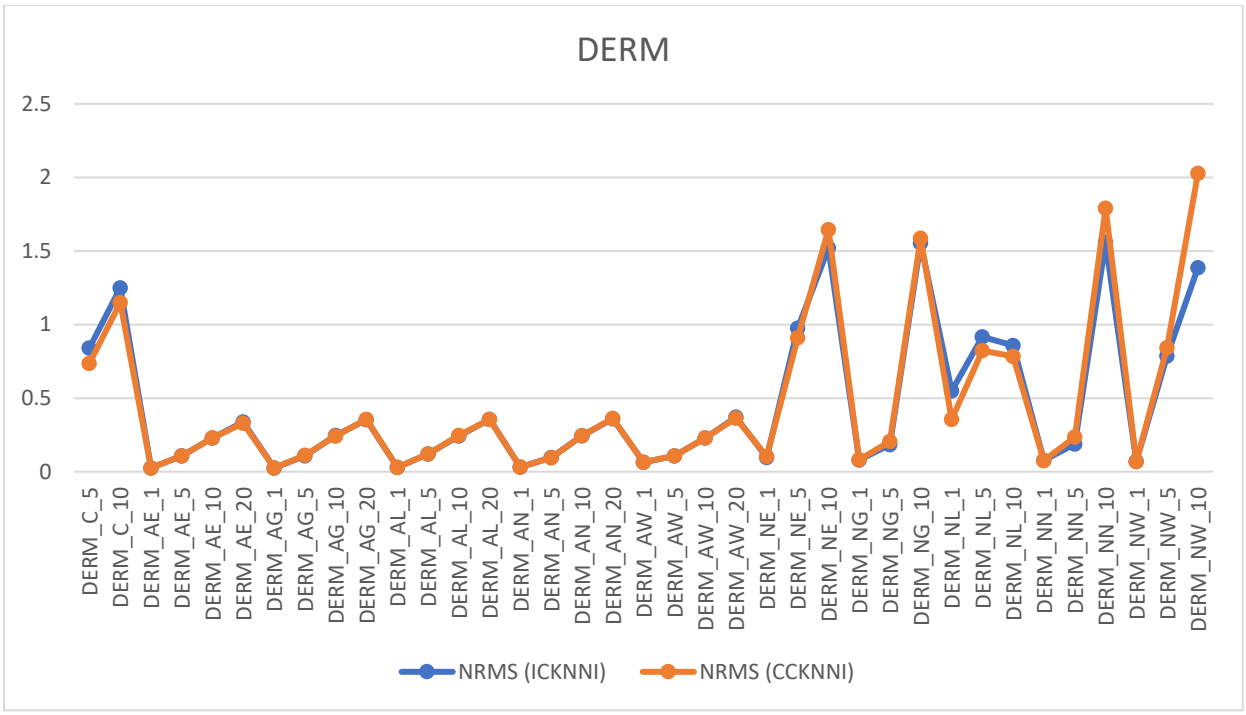

BUPA
4
3.5
3
2.5
2
1.5
1
0.5
0
BUPA_AE_1
BUPA_AG_1
BUPA_AG_5
BUPA_AL_1
BUPA_AN_1
BUPA_AW_1
BUPA_NE_1
BUPA_NG_1
BUPA_NL_1
BUPA_NL_5
BUPA_NN_5
BUPA_NW_1
NRMS (ICKNNI)
NRMS (CCKNNI)

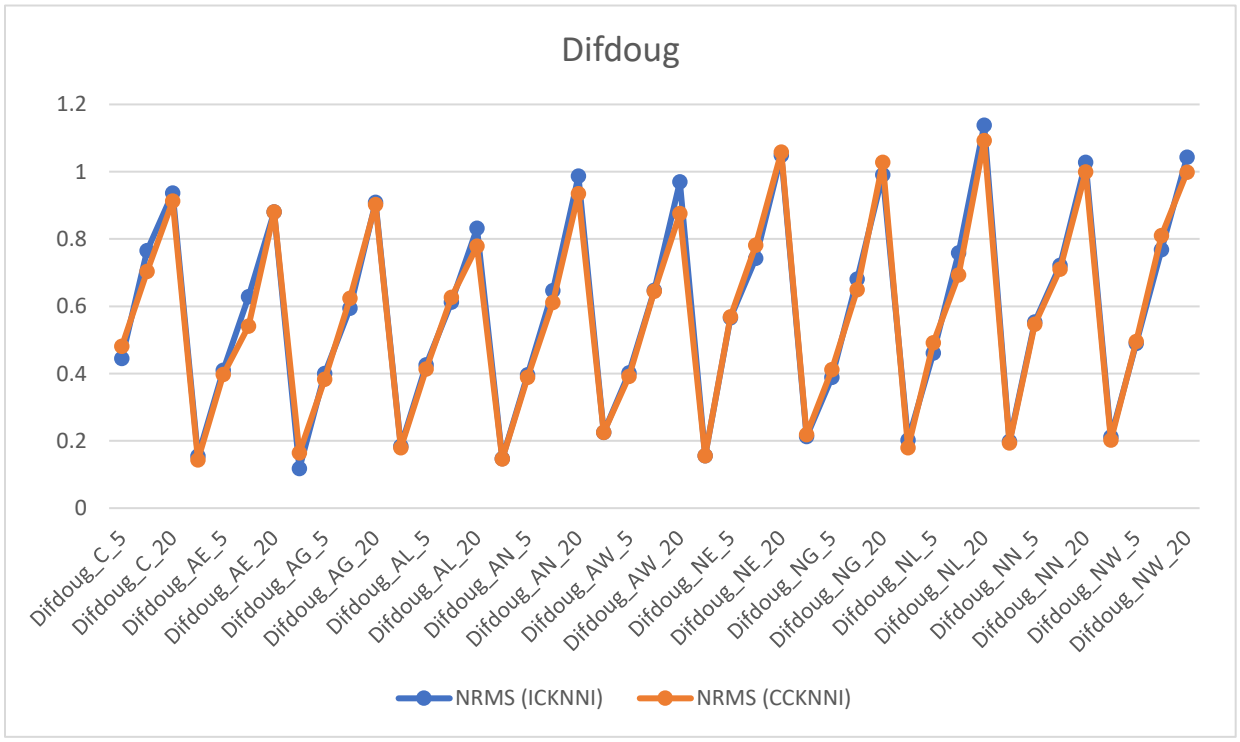

Ionosphere
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
Ionosphere_C_5
Ionosphere_C_20
Ionosphere_AE_5
Ionosphere_AE_20
Ionosphere_AG_5
Ionosphere_AG_20
Ionosphere_AL_5
Ionosphere_AL_20
Ionosphere_AN_5
Ionosphere_AN_20
Ionosphere_AW_5
Ionosphere_AW_20
Ionosphere_NE_5
Ionosphere_NE_20
Ionosphere_NG_5
Ionosphere_NG_20
Ionosphere_NL_5
Ionosphere_NL_20
Ionosphere_NN_5
Ionosphere_NN_20
Ionosphere_NW_5
Ionosphere_NW_20
NRMS (ICKNNI)
NRMS (CCKNNI)

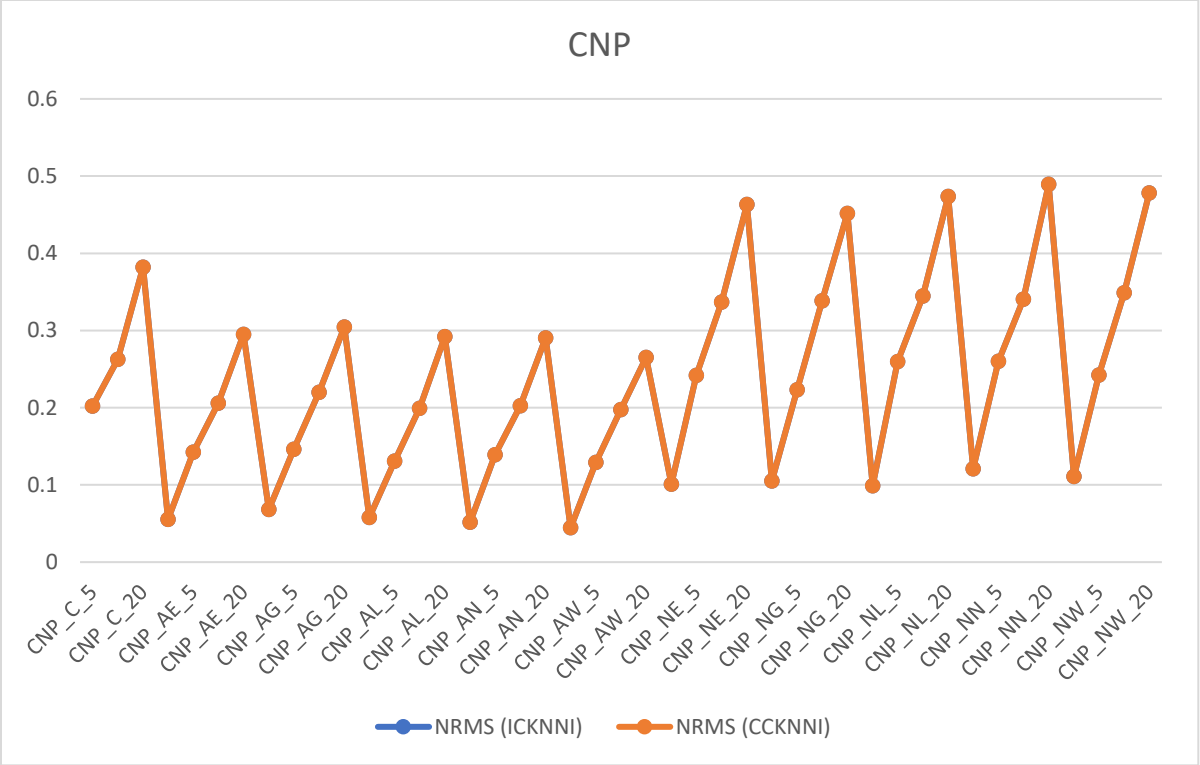

Sonar
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
Sonar_C_5
Sonar_C_20
Sonar_AE_5
Sonar_AE_20
Sonar_AG_5
Sonar_AG_20
Sonar_AL_5
Sonar_AL_20
Sonar_AN_5
Sonar_AN_20
Sonar_AW_5
Sonar_AW_20
Sonar_NE_5
Sonar_NE_20
Sonar_NG_5
Sonar_NG_20
Sonar_NL_5
Sonar_NL_20
Sonar_NN_5
Sonar_NN_20
Sonar_NW_5
Sonar_NW_20
NRMS (ICKNNI)
NRMS (CCKNNI)


4-gauss
1.4
1.2
1
0.8
0.6
0.4
0.2
0
4-gauss_C_5
4-gauss_C_20
4-gauss_AE_5
4-gauss_AE_20
4-gauss_AG_5
4-gauss_AG_20
4-gauss_AL_5
4-gauss_AL_20
4-gauss_AN_5
4-gauss_AN_20
4-gauss_AW_5
4-gauss_AW_20
4-gauss_NE_5
4-gauss_NE_20
4-gauss_NG_5
4-gauss_NG_20
4-gauss_NL_5
4-gauss_NL_20
4-gauss_NN_5
4-gauss_NN_20
4-gauss_NW_5
4-gauss_NW_20
NRMS (ICKNNI)
NRMS (CCKNNI)


BCW
1.4
1.2
1
0.8
0.6
0.4
0.2
0
BCW_C_5
BCW_C_20
BCW_AE_5
BCW_AE_20
BCW_AG_5
BCW_AG_20
BCW_AL_5
BCW_AL_20
BCW_AN_5
BCW_AN_20
BCW_AW_5
BCW_AW_20
BCW_NE_5
BCW_NE_20
BCW_NG_5
BCW_NG_20
BCW_NL_5
BCW_NL_20
BCW_NN_5
BCW_NN_20
BCW_NW_5
BCW_NW_20
NRMS (ICKNNI)
NRMS (CCKNNI)


DERM
2.5
2
1.5
1
0.5
0
DERM_C_5
DERM_C_10
DERM_AE_1
DERM_AE_5
DERM_AE_10
DERM_AE_20
DERM_AG_1
DERM_AG_5
DERM_AG_10
DERM_AG_20
DERM_AL_1
DERM_AL_5
DERM_AL_10
DERM_AL_20
DERM_AN_1
DERM_AN_5
DERM_AN_10
DERM_AN_20
DERM_AW_1
DERM_AW_5
DERM_AW_10
DERM_AW_20
DERM_NE_1
DERM_NE_5
DERM_NE_10
DERM_NG_1
DERM_NG_5
DERM_NG_10
DERM_NL_1
DERM_NL_5
DERM_NL_10
DERM_NN_1
DERM_NN_5
DERM_NN_10
DERM_NW_1
DERM_NW_5
DERM_NW_10
NRMS (ICKNNI)
NRMS (CCKNNI)


Difdoug
1.2
1
0.8
0.6
0.4
0.2
0
Difdoug_C_5
Difdoug_C_20
Difdoug_AE_5
Difdoug_AE_20
Difdoug_AG_5
Difdoug_AG_20
Difdoug_AL_5
Difdoug_AL_20
Difdoug_AN_5
Difdoug_AN_20
Difdoug_AW_5
Difdoug_AW_20
Difdoug_NE_5
Difdoug_NE_20
Difdoug_NG_5
Difdoug_NG_20
Difdoug_NL_5
Difdoug_NL_20
Difdoug_NN_5
Difdoug_NN_20
Difdoug_NW_5
Difdoug_NW_20
NRMS (ICKNNI)
NRMS (CCKNNI)



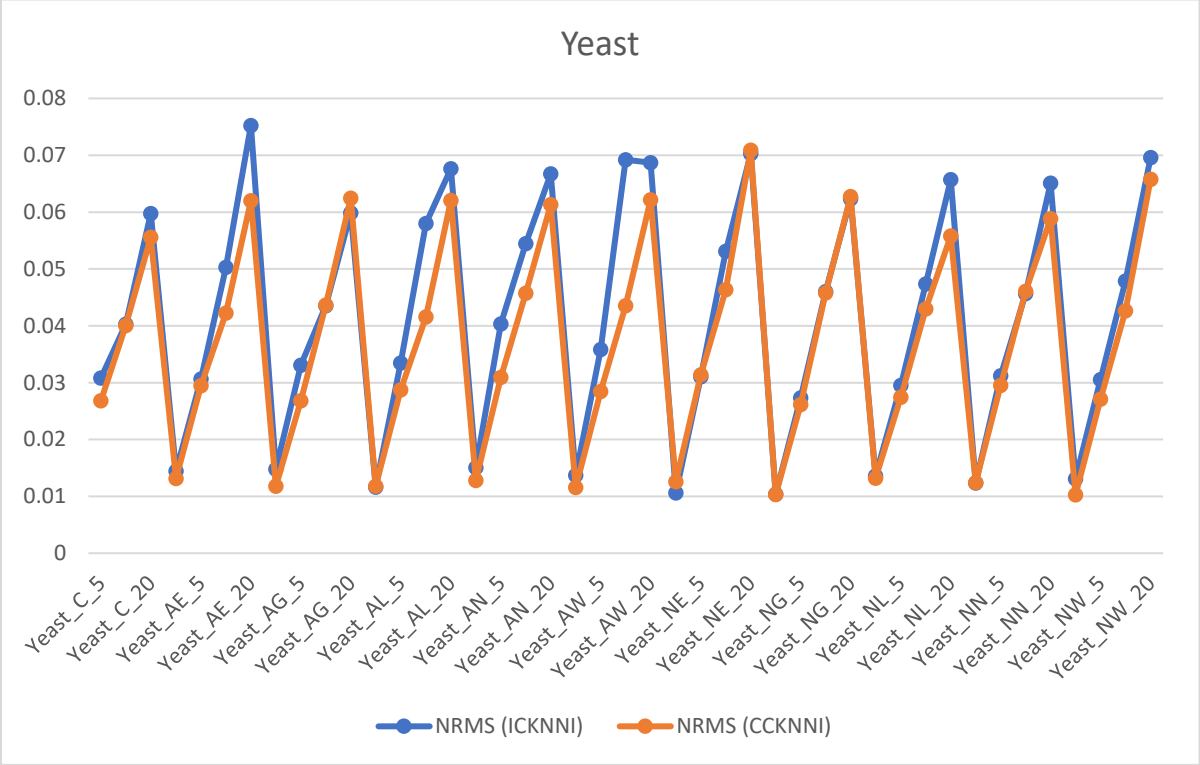




Fig. 2. Imputation comparison of CCKNNI and ICKNNI.

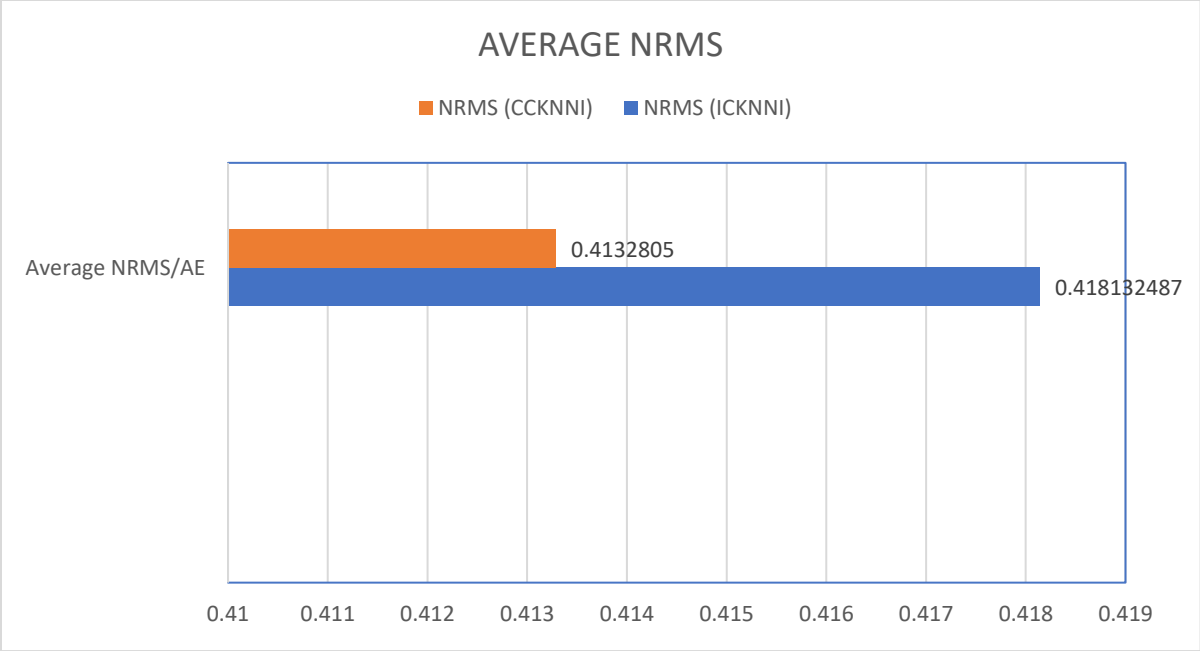



Fig. 3. Average NRMS comparison of CCKNNI and ICKNNI.

## 9. CONCLUSION

This paper completely demonstrates the whole methodologies of the ICKNNI and CCKNNI algorithms. CCKNNI is a great technique to impute missing values and also a very popular one. It is widely used in today's world. Moreover, when the volume of missingness is very high ICKNNI is an excellent alternative to the CCKKNI. It deals with all the instances and attributes to impute the missing values. The datasets used had less volume of missingness, resulting in more handsome NRMS value for CCKNNI. Although they both have shown very similar performance level. However, this paper extensively presents the study of ICKNNI which deserves more attention in the world of software engineering as the volume is usually high in real-world datasets. Also, as the data keeps increasing in the field, missingness keeps increasing as well, which leans the heads towards the method ICKKNI over CCKNNI for better performance and unbiased imputation.

## REFERENCES

[1] J. V. Hulse, T. M. Khoshgoftaar, "Incomplete-Case Nearest Neighbor Imputation In Software Measurement Data," An International Journal, Computer, and Electrical Engineering, Florida Atlantic University, Florida, USA, 2011.

[2] Z. G. Liu, Q. Pan, J. Dezert, A. Martin, "Adaptive Imputation Of Missing Values For Incomplete Pattern Classification," Pattern Recognition, Elsevier, 2016, 52, ff10.1016/j.patcog.2015.10.001ff. ffhal-01259203f.

[3] M. Sanjeevi. "Chapter 5: K-nearest neighbors algorithm with code from scratch." Medium. https://medium.com/deep-math-machine-learning-ai/chapter-5-k-nearest-neighbors-algorithm-with-code-from-scratch-7f93f653c860 (accessed August 08, 2020).