University of Windsor

Master of Engineering (Semester-III)
Electrical and Computer Engineering

Data Mining Project: C10
Fuzzy C-Means Clustering Imputation Using Support Vector Regression and a Genetic Algorithm

Instructor: Dr. Roozbeh Razavi-Far

Submitted by:
Suraj Khanna Ramesh(110063550)
Email: ramesh12@uwindsor.ca
Varun Pillai (110070811)
Email: pillai4@uwindsor.ca

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Clustering techniques can be divided into two major categories, hard clustering and soft (fuzzy) clustering. In hard clustering techniques, data object is belonged to only one cluster which is the most similar cluster, however in fuzzy clustering a dataset object is belong to each one of clusters with a certain similarity given by membership function. Hard clustering imputation techniques has been employed by many researchers such as k-means in which incomplete data object missing values is imputed based on cluster information it is belong to. However, in case of missing dataset there is uncertainty of incomplete data object is belonging definitely to certain cluster, so the need for fuzzy clustering imputation methods have been introduced such as FCM. The intra-variance in clusters is decreases by FCM compared to k-means algorithm, moreover FCM is less sensitive to stuck on local minimum situation because of continuous membership function values [1]. Fuzzy imputation achieved higher performance compared to hard clustering imputation as denoted in experimental results. Fuzzy c-means clustering works similarly to k-means, the only exception is the membership function isn't binary. In c-means, we can conveniently use the membership functions to evaluate the quality of the partition. A good fuzzy partitioning would be a partitioning where every object has a high higher membership value for one cluster and low membership value for other clusters [2]. In other words, a good partitioning is a partitioning with low fuzziness.

2 OVERVIEW OF THE PROJECT

2.1 CLUSTERING ALGORITHMS

There are many types of clustering algorithm. Many algorithms use similarity or distance measures between examples in the feature space in an effort to discover dense regions of observations. As such, it is often good practice to scale our data prior to using clustering algorithms. A clustering method attempts to group the objects based on the definition of similarity supplied to it. Some clustering algorithms require us to specify or guess at the number of clusters to discover in the data, whereas others require the specification of some minimum distance between observations in which examples may be considered [3]. The data is clustered into separate groups and the missing values are found using Fuzzy C-Means (FCM) clustering in this report. FCM clustering is extended from another majorly used clustering technique called K-means clustering. So, the brief explanation of both the techniques is explained below.

2.1.1 K-Means Clustering Algorithm

K-means algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The following are some of the disadvantages of clustering with the K-means algorithm:

1. K-means algorithm requires to specify the number of clusters (k) in advance.
2. K-means algorithm can't handle noisy data and outliers are difficult to deal with.
3. In K-means Algorithm each data point belongs to a single cluster, with no values in between.

This led to a need to create more advanced clustering technique to overcome the disadvantages of K-Means clustering called FCM clustering.

2.1.2 Fuzzy C-Means (FCM) Clustering Algorithm

Clustering can be either hard or fuzzy type. The patterns are distinguished in a well-defined cluster boundary region. But due to the overlapping nature of the cluster boundaries, some class of patterns may be specified in a single cluster group or dissimilar group. This property limits the use of hard clustering in real life applications. To, reduce such limitations fuzzy type clustering came into the picture and helps to provide more information about the memberships of the patterns. After the fuzzy theory was invented, the researchers put the fuzzy theory into clustering [4]. The Fuzzy clustering problems have been expansively studied and the foundation of fuzzy clustering was implemented.

Fuzzy clustering problems can be grouped into three branches:
(a) Based on fuzzy relation
(b) Based on fuzzy rule learning
(c) Based on optimization of an objective function.

The fuzzy clustering based on the objective function is quite popularly known to be Fuzzy c-means clustering (FCM). In FCM method, the pattern may belong to all the cluster classes with a certain fuzzy membership degree.

The following graph represents the membership values of hard and soft clustering methods. Let us consider a mono-dimensional data that is represented on x-axis. We assume that the given data is divided into 2 clusters A and B. The first graph shows that each point belonging to a dataset will have a membership value of either 0 or 1 which is known as hard clustering technique. Whereas, in hard clustering, the membership values range from 0 to 1 which is represented by a smooth curve in the 2nd graph.

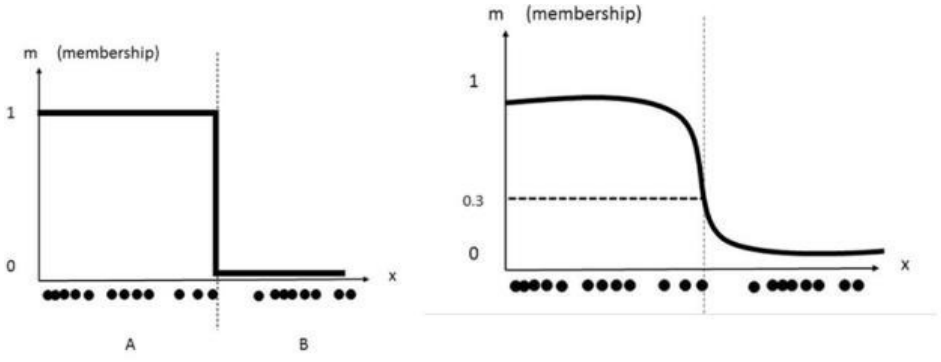

Figure 1: Comparison Between hard clustering and soft clustering using membership value [5]

2.2 MISSING DATA ANALYSIS

In general, there are three categories of missing data:

1) Completely missing at random (MCAR) - The missing value is independent of any other variable.
2) Missing at random (MAR) - The value of the missing variable is determined by other variables. Other variables can be used to estimate the missing value.
3) Missing not at random (MNAT) - The missing value is dependent on other missing values, making it impossible to predict missing data from existing variables.

In this report, we assume that the data are MAR, which means that the missing numbers can be deduced from the remaining data in a complicated way.

2.3 FUZZY C-MEANS IMPLEMENTATION

This algorithm is examined to analyze based on the distance between the various input data points. The clusters are formed according to the distance between data points and cluster centers are formed for each cluster. The basic structure of the FCM algorithm is discussed below. The Algorithm Fuzzy C-Means (FCM) is a method of clustering which allows one piece of data to belong to two or more clusters [6]. This method is frequently used in pattern recognition. It is based on minimization of the following objective function given below. C denotes the cluster number, which ranges between two and the record count (N). The weighting factor, m, is a parameter that ranges from one to infinity

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^{m} (X_i - C_j)^2$$

$$2 \leq c \leq N$$

The amount of fuzziness in the clustering process is controlled by this variable. There is no theoretically ideal c and m value. The characteristics can be adjusted depending on the dataset's properties and the relationships between the attributes. The goal of this algorithm is to discover the best c and m parameters for the data set currently in use. Each data object xi in fuzzy clustering has a membership function that determine the level to which it belongs to a particular cluster cj.

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{X_i - c_j}{X_i - c_k} \right)^{\frac{2}{m-1}}}$$

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} . X_i}{\sum_{i=1}^{N} u_{ij}^{m}}$$

2.3.1 Support Vector Regression Algorithm

Support Vector Regression leverages the same concept as Support Vector Machine, but for regression issues. SVMs are extensively used to group and sort problems. SVMs aren't always well-documented when it comes to regression**.** Finding a function that approximates the mapping from the input domain to the actual number in the training sample idea is challenging with regression. SVR is mostly concerned with considering points within a decision's bounds. SVR is a popular algorithm that allows you to pick between a satisfactory error margin and an acceptance adjustment that exceeds the acceptable error rate, giving you more freedom**.**

2.3.2 Genetic Algorithm

Optimization techniques are those that are used to find the optimal result out of all the ones that are feasible given the constraints. The genetic algorithm is a type of optimization algorithm that is based on nature's natural evolutionary process. Here, the concepts of Natural Selection and Genetic Inheritance are applied.

Unlike traditional algorithms, it employs guided random search, which involves starting with a random initial cost function and then searching exclusively in the space with the lowest cost to discover the best answer. When working with large and complex data sets, this technique is ideal.

2.4 BASIC MODEL OF SVR AND GA ALGORITHM IMPLEMENTATION

1) Choose the examples in which no attribute values are lacking.

2) Assign one of the condition attributes (the input attribute) as the decision attribute (the output attribute), with some of its values missing, and the decision attributes as the condition attributes.

3) Predict decision attribute values using SVM regression.

Before being utilized for data imputation, the support vector regression model must first be trained with complete records. When utilizing SVR, the inputs are recalled on output, and the culture GA approximates xu (the unknown attribute value). The model input is made up of xk (known data attributes) and xu, and the model output is f (function), which corresponds to the input. The model is supposed to remember the input data; hence the difference is referred to as an error [6]. The cultural GA aims to reduce the error between the model output and the input; for minimization, the error must be non-negative, resulting in a data variable that is most likely the missing value. However, all of the outputs are used to reduce the approximated value's inaccuracy for completeness. The formulae for SVR input, SVR output, error function and generic algorithm fitness function are shown below

$$\text{SVR Input} = \begin{pmatrix} Xk \\ Xu \end{pmatrix}$$

$$\text{SVR Output} = f\begin{pmatrix} Xk \\ Xu \end{pmatrix}$$

$$\text{Error} = \begin{pmatrix} Xk \\ Xu \end{pmatrix} - f\begin{pmatrix} Xk \\ Xu \end{pmatrix}$$

$$\text{GA fitness function} = \left(\begin{pmatrix} Xk \\ Xu \end{pmatrix} - f\begin{pmatrix} Xk \\ Xu \end{pmatrix}\right)^2$$

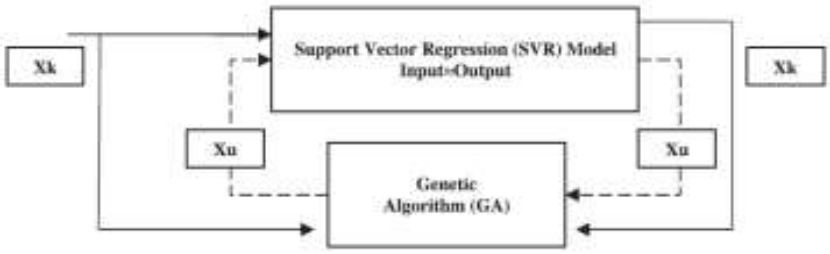

Figure 2: Basic model of SVR and GA to impute missing value.

2.5 SAMPLE IMPUTATION USING FCM

Let us now look at an example of how the imputation is performed for a given data. Let us assume that any given data is clustered into 3 different clusters as shown in the fig. 'x' in the graph represents the centroid of 3 clusters and the symbol '?' indicates one of the missing values in a dataset that has to be imputed using FCM

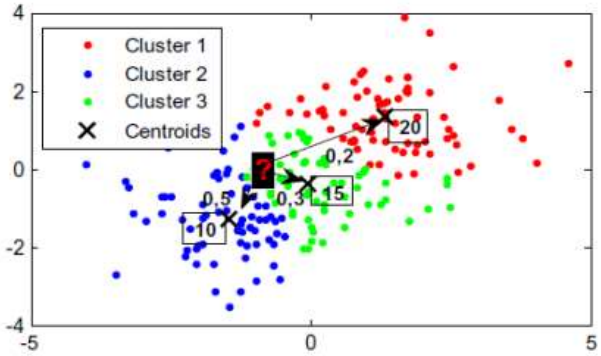

Figure 3: Fuzzy c-means Imputation [6]

- No. of clusters (c) = 3
- Weighting factor (m) = 2
- Membership values of '?' are estimated to be 0.5, 0.3, and 0.2
- Centroids of clusters = 10, 15, and 20

• Thus, missing value'?' = 0.5 * 10 + 0.3 * 15 + 0.2 * 20 = 13.5

3. COMPLEXITY ANALYSIS FOR FCM USING SVR-GA

Time complexity is the amount of time taken by an algorithm to run and provide the output. It depends on the length of input. It measures the time taken to run each statement of code in an algorithm.

Step-by-step analysis of time complexity of FCM imputation is as follows:

- Let us consider a dataset with 'm' records and 'n' attributes.
- Let us suppose that there exists n' incomplete attributes over the whole dataset, and m' records with one or more missing values.
- Complete data is then clustered based on the number of clusters, Euclidean distance, and other parameters of FCM. Therefore, the time complexity of FCM is given by**: $O(mnc^2i)$** where 'c' = number of clusters and 'i' = number of iterations [7].
- A for loop is initiated m' times in which various operations are performed:

    1. for loop is used to calculate the Euclidean distance between the cluster centers and missing values **O(c).**
    2. Another for loop is used to calculate the membership values for missing attributes **O(n')**
    3. Finally, a for loop is performed for c times to find the estimated values using FCM

- For SVR algorithm, we have the time complexity of **$O(n^2)$** [7]. Missing data is predicted by setting the labels as missing feature column.
- Genetic Algorithm is initiated n times, the optimized C and M parameters are calculated by fitness score, which is calculated by $(X-Y)^2$, where X is the missing value imputed from FCM with params from the current population.

  And Y is the missing value imputed from SVR. For Genetic Algorithm based on the number of population p, generally the complexity is **O(p)**.

Thus, the overall complexity of FCM using SVR-GA upon simplification is as follows **$O(mn + mnc^2ip + n^2p + m'n'c)$**. Assuming m' << m and the datasets used in these experiments are typical numerical datasets, in which the number of records is much larger than the number of attributes (m >> n).
Therefore, the complexity of FCM imputation is **$O(mnc^2ip)$**.

4. IMPLEMENTATION FCM USING SVR-GA

This project is implemented in Spyder (Python) version 3.9 64 bit. Basic requirements to run the code **is** a windows operation system (OS) with Spyder or google collab online tool in it and a suitable processor such as Intel core or AMD Ryzen which are compatible to perform the coding.

The first step involves importing all necessary libraries to get the desired output. These are as follows

- NumPy**: NumPy**, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.
- Pandas**:** Pandas is an open-source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named NumPy, which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages, Pandas works well with many other data science modules inside the Python ecosystem, and is typically included in every Python distribution**.**
- Sklearn: Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.
- Random: The random module is a built-in module to generate the pseudo-random variables. It can be used perform some action randomly such as to get a random number, selecting a random element from a list, shuffle elements randomly**.**
- Skfuzzy: This package implements many useful tools and functions for computation and projects involving fuzzy logic, also known as grey logic. Most of the functionality is actually located in sub packages, but like NumPy we bring most of the core functionality into the base namespace.
- Copy: Copy Module is a set of functions that are related to copying different elements of a list, objects, arrays, etc. It can be used to create shallow copies as well as deep copies.
- OS**:** The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The os and os.path modules include many functions to interact with the file system.
- Sys: The sys module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment. It allows operating on the interpreter as it provides access to the variables and functions that interact strongly with the interpreter.
- Time: As the name suggests Python time module allows to work with time in Python. It allows functionality like getting the current time, pausing the Program from executing, etc.

In the next step, the incomplete and the complete values are loaded into an array using NumPy module. Then, os.listdir() is used to locate the files in the local remote system. The class *SVREstimator*, is used to compute the missing values. A function is created to separate the complete and incomplete dataset and stored in array complete_rows and incomplete rows respectively. In class *SVREstimator*, a function called estimate_missing_value is created, where the complete_rows are used to train the algorithm and return estimated_data containing estimated data of the missing values.

In the class FCMeansEstimator, a function called getCenters() is defined which takes the data from complete_rows for all the columns and calculates the centers for all the entries using the defined

parameters. In the function estimate_missing_value, the missing value is calculated for a sample by taking sum of multiplying the membership value and centroid for a particular sample.

In the class Genetic_Algorithm, a random population will be initialized with C and M values, the optimized C and M values are calculated by the process of matting and mutation of all the population, the above process requires fitness score which is calculated using the formulae, $(X\text{-}Y)^2$, where X is the missing value imputed from FCM with params from the current population, and Y is the missing value imputed from SVR.

The final step, we obtain the optimized C and M parameters from the genetic algorithm, which is then applied to FCM estimator to find the missing values. The NRMS value is calculated between the imputated data and the complete data.

The above-mentioned procedure can also be followed for running the code in an online platform such as Google Colab where the syntax of Python can be used and edited according to the requirement

5. Experimental Results

5.1 Experimental Setting

As mentioned in the implementation section, necessary parameters are given and are served as the input to the code so that the calculations are performed, and the code runs successfully.

The parameters we defined are as follows:

- **nrms = [ ]**
  An empty list of nrms is defined so that code stores all the calculated values of NRMS for each dataset. The formula for calculating NRMS is given by

$$\text{NRMS} = \frac{||X^{estimate} - X^{original}||_F}{||X^{original}||_F}$$

  Where $X^{original}$ and $X^{estimate}$ are the originally complete dataset and the imputed dataset, respectively; and ||.|| stands for the Frobenius norm

5.2 Results Analysis

5.2.1 Distribution of NRMS based on the percentage of missing data

The various scenarios are performed on the code and values are imputed by changing the parameter settings to obtain the optimal values for the missing data. The analysis of these results is discussed below.

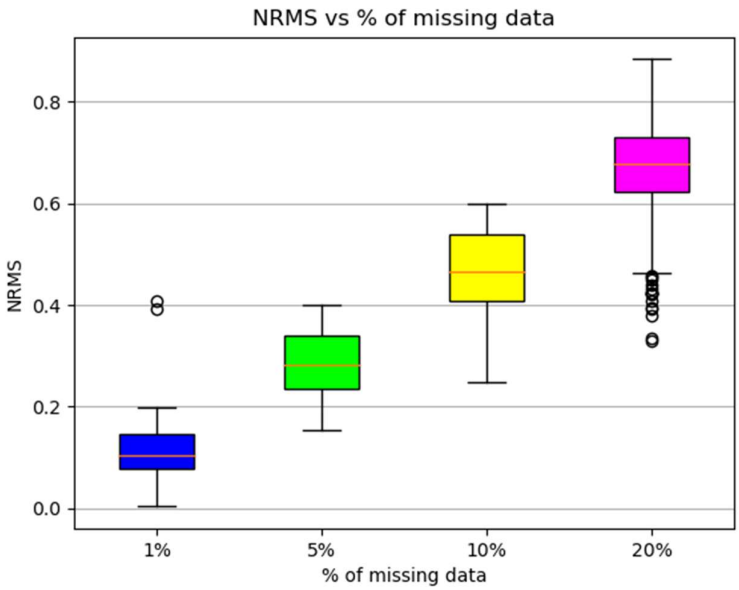

Figure 4: Distribution of imputation results in terms of NRMS

Looking at the above fig, one can notice that hybrid model of FCM performs better on less percentages of missing data. The inter-quartile range (IQR) increases in proportion to the percentage of missing data. IQR for 5% of missing data is minimum compared to others which signifies values imputed are almost approximate to the original values. 75% of the NRMS values are less than 20% as seen in the graph.

Circles outside the range represents the outliers. In this case, the number of outliers present are higher for 20% of the data. The cross marks present within the box plots represent the median for each class of data. The mean NRMS value for all the missing ratios is calculated to be approximately 10% which signifies that the imputed values are close enough to the real values.

5.2.2 Distribution of NRMS based on type of data

AE	AG	AL	AN	AW	C	NE	NG	NL	NN	NW
0.0942	0.0849	0.0871	0.0924	0.0998	0.0991	0.1194	0.1066	0.131	0.4081	0.3926
0.2291	0.222	0.2124	0.2073	0.2177	0.2051	0.268	0.2531	0.293	0.2681	0.2574
0.3206	0.3084	0.2983	0.3084	0.3019	0.3109	0.3743	0.3745	0.4103	0.4081	0.3926
0.4217	0.4226	0.4226	0.4564	0.4308	0.4491	0.5223	0.5223	0.5319	0.5225	0.5091
0.0907	0.0843	0.0879	0.0832	0.1025	0.1006	0.0984	0.0983	0.1229	0.1116	0.1121
0.2331	0.1985	0.2228	0.2135	0.211	0.2129	0.2574	0.234	0.2798	0.2798	0.2457
0.3251	0.2968	0.3257	0.2982	0.3083	0.3269	0.3622	0.3702	0.3691	0.3924	0.3722
0.4562	0.4246	0.4757	0.4891	0.4623	0.6429	0.8837	0.6513	0.7633	0.8227	0.8017
0.0033	0.1908	0.0089	0.0699	0.0582	0.0871	0.0502	0.0992	0.0608	0.1699	0.1461
0.2469	0.3066	0.315	0.2437	0.2159	0.2199	0.2939	0.3964	0.3551	0.2505	0.2477
0.5523	0.4489	0.5746	0.511	0.5058	0.4084	0.4069	0.4506	0.4088	0.4497	0.5793
0.6326	0.6535	0.6876	0.7499	0.7885	0.6513	0.6333	0.7245	0.6568	0.7055	0.7811
0.141	0.088	0.138	0.116	0.0997	0.0726	0.1673	0.051	0.0625	0.0816	0.0665
0.3462	0.3249	0.2025	0.2553	0.3353	0.2564	0.2786	0.247	0.3443	0.3924	0.2088
0.4718	0.4582	0.5137	0.4096	0.4617	0.5543	0.5744	0.5015	0.407	0.4452	0.555
0.6857	0.7689	0.73	0.7171	0.7217	0.7037	0.6294	0.6379	0.7981	0.6234	0.6044
0.0744	0.0732	0.0735	0.1561	0.1872	0.1589	0.1692	0.1861	0.0976	0.1873	0.0093
0.1667	0.1543	0.2905	0.2153	0.3233	0.3635	0.3147	0.3007	0.3189	0.314	0.2568
0.2484	0.2879	0.2486	0.5134	0.5717	0.4706	0.481	0.5593	0.5825	0.465	0.4039
0.3305	0.3351	0.7385	0.6872	0.6122	0.6746	0.6137	0.7225	0.624	0.6903	0.6897
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
0.7082	0.6669	0.718	0.7938	0.7232	0.7844	0.769	0.7466	0.601	0.7704	0.6214
0.1635	0.0274	0.0477	0.0594	0.1105	0.1782	0.1572	0.1462	0.0874	0.0778	0.1905
0.3087	0.334	0.383	0.2427	0.2515	0.3396	0.2881	0.2236	0.2879	0.3115	0.3843
0.4559	0.4882	0.4678	0.4438	0.5906	0.5689	0.5992	0.4369	0.5537	0.4716	0.5927
0.7398	0.7821	0.7828	0.6799	0.6008	0.6215	0.7853	0.6066	0.7094	0.6257	0.6244

Table 1: NRMS values for different types of data

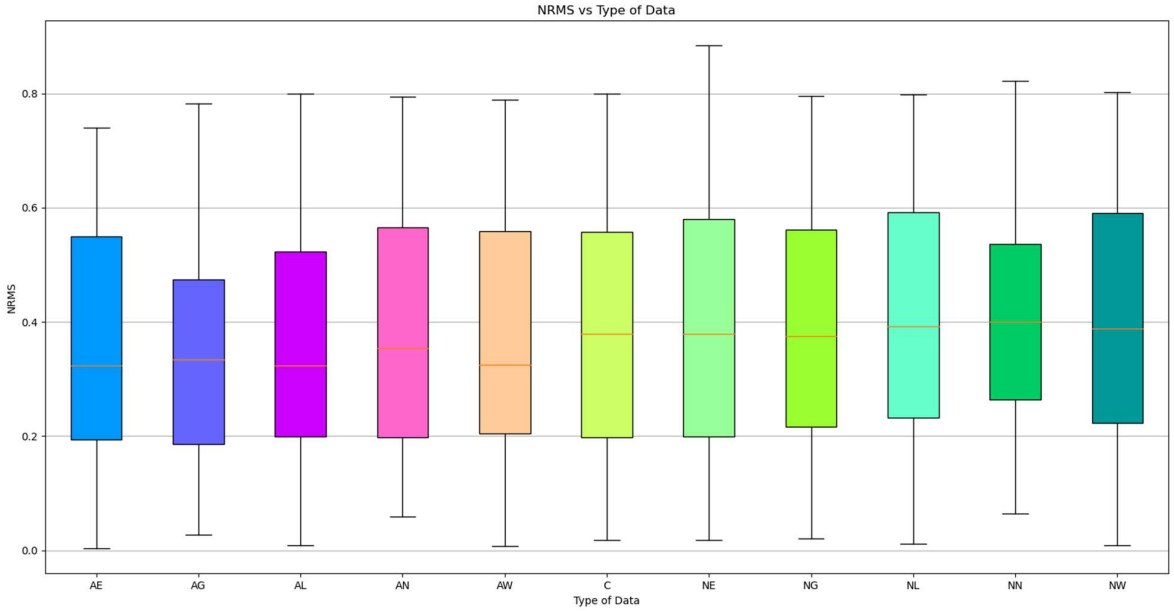

Figure 5: NRMS v/s Type of Data

The NRMS values of different types of data are tabulated as shown in table 1. The NRMS from all type of data are considered to plot a box plot and discuss the results. As seen in fig, NRMS values for all the types of data have similar IQR which represents that the FCM technique does not discriminate the data based on type it contains. Also, there are no outliers present in the plot which suggest that all the values of NRMS are within the range of IQR.

5.3.3 Average NRMS based on different values of 'm'

M	Average NRMS
2.0	0.4008
3.0	0.3722
4.0	0.3655
5.0	0.3448
6.0	0.3343

Table 2: Average NRMS for different values of n.

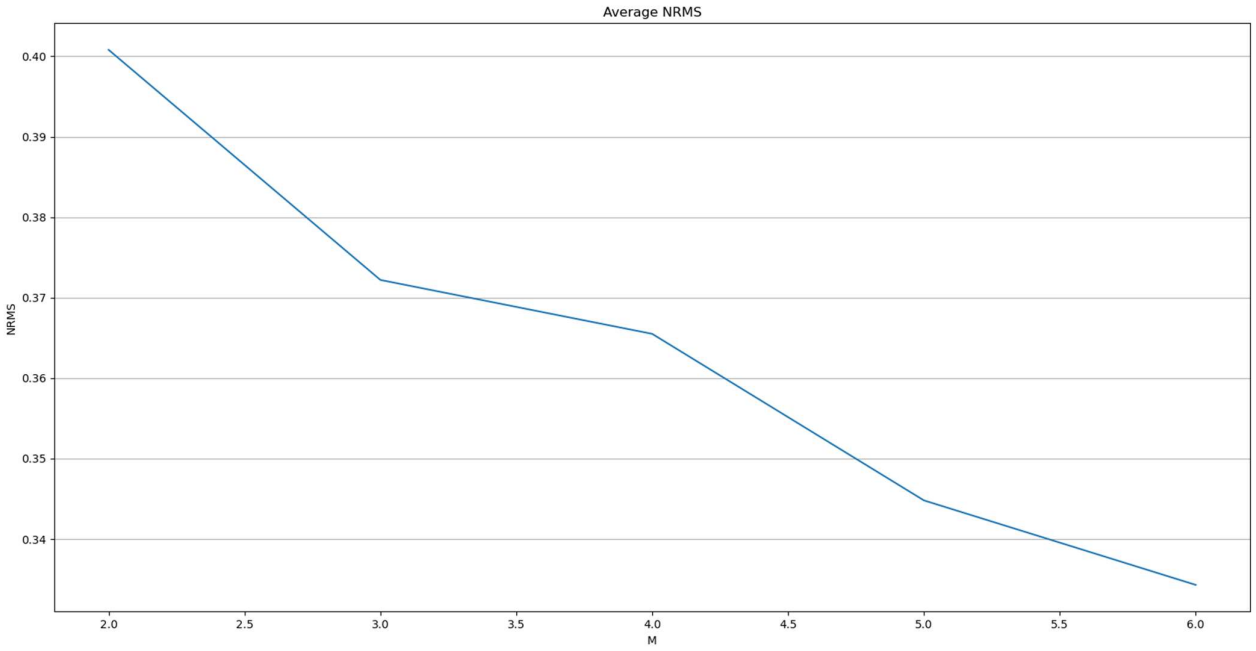

Figure 6: Fuzzifier m v/s NRMS

The weighting factor 'm' is called fuzzifier. It has an influence on the performance of Fuzzy c- means technique. We have calculated the NRMS of all the numerical datasets using different values of 'm' and then calculated the mean of it for each case. The average values of NRMS for every case are as shown in table 2. From the graph we can see that the Average value of NRMS decreases as the value of 'm' increases.

6. REFERENCES

[1] Sefidian, Amir Masoud, and Negin Daneshpour. "Missing Value Imputation Using a Novel Grey Based Fuzzy C-Means, Mutual Information Based Feature Selection, and Regression Model." Expert Systems with Applications, vol. 115, 2019, pp. 68–94., https://doi.org/10.1016/j.eswa.2018.07.057.

[2] "Chapter 448 fuzzy clustering - ncss-wpengine.netdna-ssl.com." [Online]. Available: https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Fuzzy_Clustering.pdf. [Accessed: 24-Jul-2022].

[3] J. Brownlee, "10 clustering algorithms with python," Machine Learning Mastery, 20-Aug-2020. [Online]. Available: https://machinelearningmastery.com/clustering-algorithms-with-python/. [Accessed: 24-Jul-2022].

[4] X. Gao and W. Xie, "Advances in theory and applications of fuzzy clustering," Chinese Science Bulletin, vol. 45, no. 11, pp. 961–970, 2000.

[5] "Fuzzy clustering," Wikipedia, 29-Jan-2022. [Online]. Available: https://en.wikipedia.org/wiki/Fuzzy_clustering. [Accessed: 24-Jul-2022].

[6] I. B. Aydilek and A. Arslan, "A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm," *Information Sciences*, vol. 233, no. 0, pp. 25–35, 2013.

[7] J. F. Kolen and T. Hutcheson, "Reducing the time complexity of the fuzzy C-means algorithm," IEEE Transactions on Fuzzy Systems, vol. 10, no. 2, pp. 263–267, 2002.