# Benchmarking Clustering and Classification Tasks using K-Means, Fuzzy C-Means and Feedforward Neural Networks optimized by PSO

Adam Pickens

Adam Pickens

# Benchmarking Clustering and Classification Tasks using K-Means, Fuzzy C-Means and Feedforward Neural Networks optimized by PSO

Submitted in partial fulfillment
of the requirements
for the Murray State University Honors Diploma

Adam Pickens

May 2021

## I. Introduction

The concept of machine learning has been studied and expanded upon over the course of many years. There are a wide variety of applications that the various methods and algorithms belonging to the discipline can be applied to, and as technology grows and improves, more and more techniques have become more feasible to implement given a spurt in computational capacity. The ability to classify data into groups of interest has a wide range of applications in many fields from medical imaging to astronomy. In addition to that, it is particularly important to businesses that rely on field data to make decisions about resource allocation. It is thus important to verify their effectiveness to ensure the validity of the end results generated from metrics returned by the algorithms. This work attempts to explore a handful of classification and clustering algorithms, with a focus on the successes and limitations for each. To do so, a set of benchmarking data sets of varying sizes and complexities were taken and ran through implementations of the algorithms, and the performance indices were calculated and recorded for later analysis. The algorithms explored in this work are the clustering algorithms K-Means and Fuzzy C-Means, and a feed forward Neural Network with a Particle Swarm Optimization (PSO) driven weight update scheme. K-Means is a widely known algorithm and has been utilized in many different situations. While it is known for being flexible and simple to implement, it is extremely sensitive to its initialization parameters, and tends to become stuck in local optima [2]. The Fuzzy C-Means (FCM) algorithm takes a soft clustering approach, utilizing a membership function that allows a data point to belong to multiple clusters at once. Like K-Means however, models can often become stagnant and fall into local optima [1]. Branching out somewhat from clustering algorithms is a classifier utilizing a Feed-forward

Neural Network with a PSO optimizer instead of traditional gradient descent. Neural Networks can have very promising performance for pattern recognition and classification problems, as they are able to approximate a function that can represent the patterns found in the input data [3]. Unless adequate care is taken however, the network can end up over-fitting or under-fitting the data and its optimizer may be trapped in local minima. To remedy this. it is necessary to properly propagate the node weights at the various iterations of the training process. This is achieved by using a gradient-free swarm algorithm viz. the PSO which fine-tunes the network weights. While PSO can be used in conjunction with other algorithms, it is also able to act as a standalone optimizer. PSO works well for finding globally optimization solutions as it can sample a large region of the search space within a given computational budget, but it must be parametrized properly to avoid trapping in a local optimum. There are a number of parameters that influence the functionality of the algorithm and while this can allow it to be reworked to better fit an individual data set, incorrect tuning can also cause its performance to suffer. Thus, hybridizing PSO with another classifier algorithm can help eliminate the weaknesses of both [2].

There have been several studies that look at integrating swarm optimizers within feed-forward neural networks in order to maximize performance gains irrespective of the nature of the cost function across a variety of swarm intelligence algorithms [20] [21] [22]. In general swarm optimizers are a collection of guided random search techniques which draw inspiration from natural or physical processes and use multi-agent group dynamics to formulate laws of motion and convergence [23] [24] [25] [26] [27] [28]. These algorithms are distributed in nature and are well-known for their efficacy in approximately solving real-world engineering problems such as Job Shop Scheduling [29], Vehicle Routing [30] [31], Digital Filter Design [32] [33], Portfolio Optimization [34] [35] and more.

The goal of this work is to explore the performance of these algorithms and comment on the successes and limitations of each, and by doing so form a better understanding of how they can influence the results so that when they are applied to real world problems the meaning behind the results is better illuminated. A flowchart representing the experimental process can be found in Figure 1. The subsequent sections are structured as follows: Section II details the K-Means algorithm, Section III does the same for the Fuzzy C-Means algorithm, Section IV describes the Feed-forward Neural Network tuned by PSO, Section V details the experimental setup, Section VI covers the results and analysis of the experiment and finally Section VII contains the conclusions reached and details possible future work in the area.

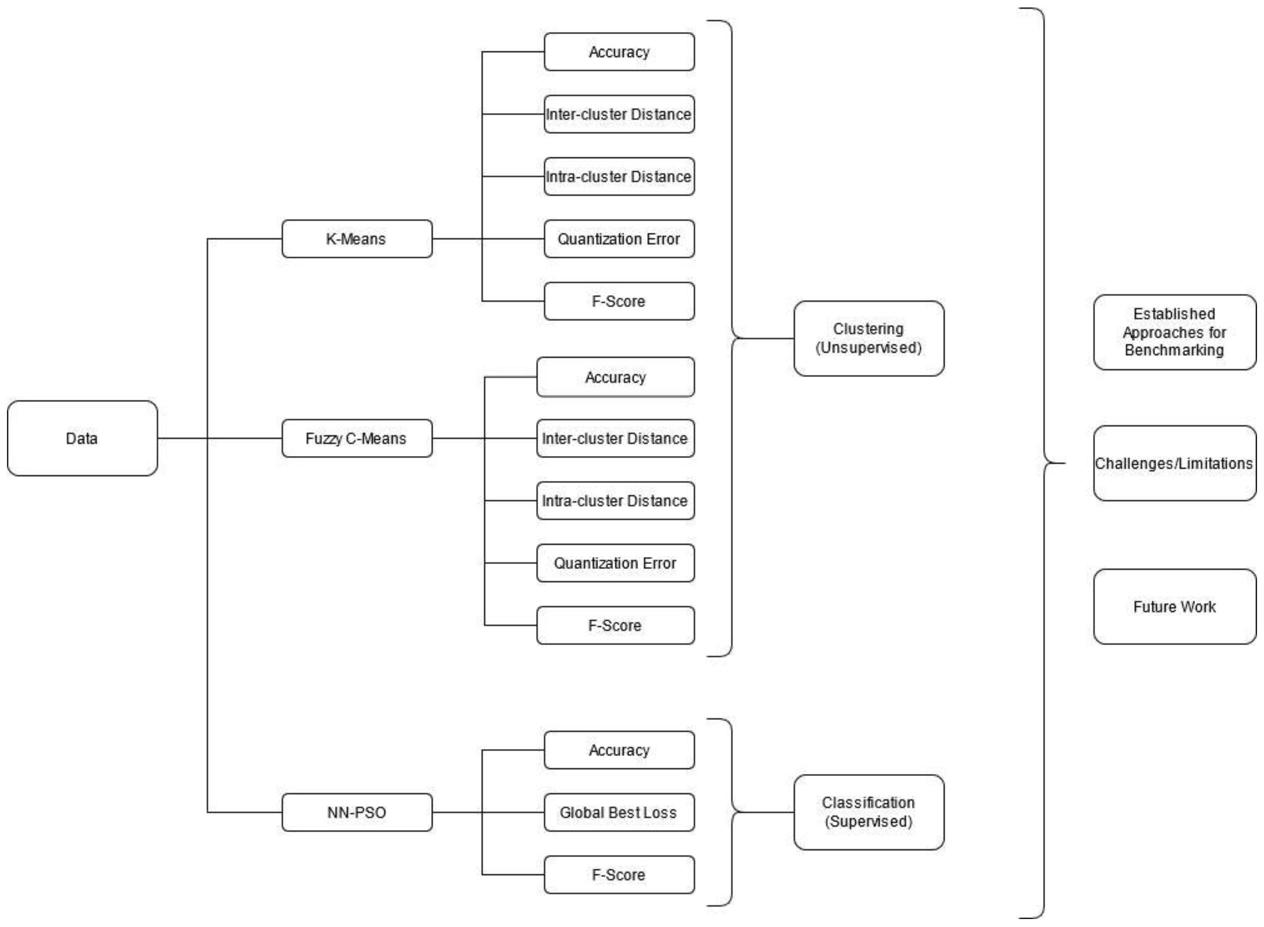

**Figure 1. Experimental Flow Chart**

To put the main objective of the experiment in other words, the goal is to take a more intimate look at the set of algorithms and attempt to understand not just how they function, but what that functionality means for the situations to which they are applied. Every algorithm has strengths and weaknesses that can be bolstered and worked around, but by understanding these strengths and weaknesses a new perspective can be gained on how they can apply to the results. While it is easy enough to apply the algorithm and take the results at face value, understanding how those results came about lets us make more sense of how the data set is interacting with the algorithm and if any biases or misrepresentations are influencing our ability to understand the model.

This is not an easy task, however. Not all algorithms can be broken apart easily and making sense of any information gleaned from doing so can be difficult given the complexity that goes into these algorithms. Furthermore, the data being analyzed plays a large part in this process, so a decent understanding of the data sets is also incredibly important.

## II. K-Means Algorithm

The K-Means algorithm is a well-known clustering algorithm and has been widely applied in many different computational problems [4] [36] [37]. It is an iterative algorithm, which means it runs continuously from a given starting point and generates new approximate solutions by adding improvements to the results from previous runs. As K-Means is a clustering algorithm, its goal is to assign all data points from a given data set into a predefined number of clusters. To do this, it attempts to minimize the Euclidean distance between each of the data points belonging to the distinct cluster groups. The Euclidean distance between two points (in Euclidean space) is simply the length of the line segment between the two chosen points. It can be computed from the Cartesian coordinates of the chosen points using the Pythagorean theorem and is also the Pythagorean distance.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{1}$$

$p, q$ are two points in Euclidean n-space, $q_i$ and $p_i$ are the Euclidean vectors initiated at a chosen origin. By minimizing the Euclidean distance between points in the group a higher similarity score can be determined from the groupings. Furthermore, the points in the cluster groups are represented by the cluster centroid, which is determined by calculating the average of all points in the cluster, much like how the center of mass is computed in an n-body problem. Given this context it can be said that the K-Means algorithm attempts to determine the positions of the centroids that give us the best results in terms of grouping data points that share similar signatures or are members of the same cluster.

There are many benefits to using the K-Means algorithm. First off, it is comparatively simple to implement and understand. Given that there are only a handful of calculations needed for the vanilla version of the algorithm, and that those calculations are simple to both understand and implement, the algorithm can be put together in a functional way much more easily than many other similar algorithms. Second, execution is fast. As one may expect from the simple design, the iterations of the algorithm can be run very quickly, leading to quick convergence. K-Means is also a very flexible algorithm. Its structure makes data sets of all sizes usable for the purpose of clustering, making it perfect for situations where many data sets of varying size and complexity need to be worked with.

K-Means does have its drawbacks, however. The algorithm does not determine the optimal set of clusters on its own, relying on the user to provide them. It is also very sensitive to the initialization of parameters. Changes in how the cluster centroids are initialized or how the data set is ordered or formatted can end up changing the results put forward by the algorithm. The algorithm also assumes that the clusters in the data will be spherical, which means that its effectiveness is much more limited when working with data sets with varied cluster shapes.

The standard implementation scheme of the K-Means algorithm is structured below [2]:

1. Randomly initialize the *k* cluster centroids in space

$$Z = \{z_1, z_2, \dots, z_k\} \tag{2}$$

2. Repeat until a termination condition is satisfied

   (a) Assign to each data point in space the cluster centroids which has the closest distance to the point. The Euclidean distance of data point $y_p$ to the centroid in d-dimensional space is given as:

$$D(y_p . z_j) = \sqrt{\sum_{i=1}^{d} (y_{pi} - z_{ji})^2} \tag{3}$$

   (b) Recalculate the cluster centroids based on the definition of centroid. So, the centroid for cluster $j$ is determined as follows:

$$z_j = \frac{1}{n_j} \sum_{\forall y_p \in C_j} y_p \tag{4}$$

   Where $C_j$ is the subset of data points belonging to the cluster $j$ and $n_j$ is the number of data points in this cluster.

The clustering process terminates when one of the following conditions is satisfied:

1. The number of iterations exceeds a predefined maximum.
2. When change in the cluster centroids is negligible.
3. When there is no cluster membership change.

An interested reader may take a look at J. MacQueen's seminal work on methods for classification and analysis of multivariate observations [4] for a deeper understanding of the foundations of the k-means algorithm.

## III. Fuzzy C-Means Algorithm

Similar to the K-Means algorithm, the Fuzzy C-Means (FCM) algorithm attempts to group a set of $N$ objects into $C$ clusters [19]. In other words, the objective is to divide the object set $D = \{D_1, D_2, D_3, \dots D_N\}$ into C clusters $(1 < C < N)$ with $\Omega = \{\Omega_1, \Omega_2, \Omega_3, \dots, \Omega_c\}$ representing the cluster centers. Every data point being investigated is assigned to a cluster, of which the centroids are randomly initialized. This assignment is done using a membership function $\mu_{ij}$ defined in [1]:

$$\mu_{ij} = \frac{1}{\sum_{r=1}^{C}(\frac{d_{ij}}{d_{rj}})^{\frac{2}{m-1}}} \tag{5}$$

$d_{ij} = \|x_i - y_j\|$ represents the distance between the $i_{th}$ center and the $j_{th}$ data point, $d_{rj} = \|x_r - y_j\|$ is the distance between the $r_{th}$ center and the $j_{th}$ data point, and $m\epsilon[1, \infty)$ is a fuzzifier. The FCM algorithm uses an iterative gradient descent, an optimization algorithm used to find the extremum of a function, to compute centroids. The centroids, in turn, are updated using the following equation:

$$x_i = \frac{\sum_{j=1}^{N} \mu_{ij}^{m} y_j}{\sum_{j=1}^{N} \mu_{ij}^{m}} \tag{6}$$

The objective function to be minimized by the algorithm can be formulated as the sum of membership weighted Euclidean distances:

$$\varphi = \sum_{i=1}^{C}\sum_{j=1}^{N} \mu_{ij}^{m}(\|x_i - y_j\|)^2 \tag{7}$$

Through the recursive calculation of equations (5) and (6) the algorithm can be stopped once a termination condition is met. Similar to other algorithms that utilize gradient descent, FCM can stagnate to local optima in a multidimensional search space. Therefore, the utilization of a stochastic optimization approach to compute cluster centroids can help mitigate this problem to a great extent.

## IV. Feed Forward Neural Network with a Particle Swarm Optimization (PSO) driven Weight Update Scheme

For this project, an implementation of a Neural Network using Particle Swarm Optimization as the weight optimization function was utilized, and while the two are different topics, as they are intrinsically linked in this project this section will contain descriptions of both.

### a. Feedforward Neural Network

A neural network consists of several connected functional units or neurons that come together as parallel information processing units in order to solve classification and regression tasks. These units working together can separate the input space into a discrete number of classes i.e., they can approximate the underlying function that maps inputs to outputs. For this work, a feed-forward neural network is being used. This is a multi-layered network of neurons consisting of an input and output layer of nodes, along with several hidden layers that link them. The hidden layers learn the intermediate mappings that eventually are propagated to form the final mapping in the form of the output. Within a hidden layer each neuron is connected to the outputs of a subset (or all) of the neurons in the previous layer each multiplied by a number called a weight. Neural networks learn by changing its weights to approximate a function representative of the input patterns. Instead of traditional gradient descent, the PSO algorithm is being used as the optimizer to fine tune the network weights as the neural network learns the optimal decision boundary on the benchmarking data sets. A simple model of the discussed network is shown in Figure 2:

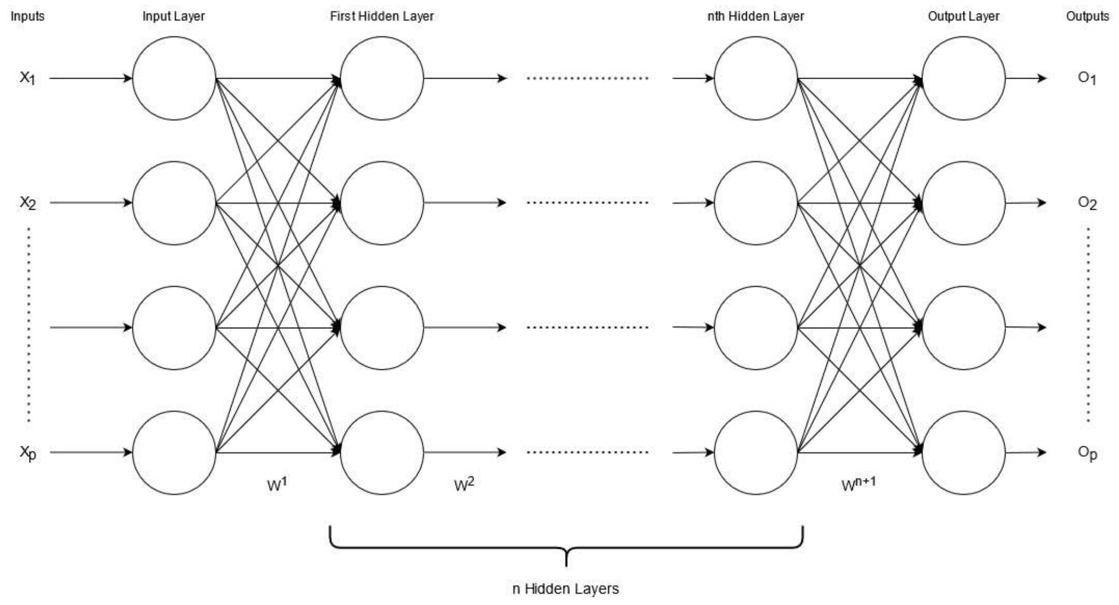

**Figure 2. A Feed-forward Neural Network with n hidden layers, p input variables, and q output variables, with weights W**

While there are various types of propagation in neural networks, the feed-forward network only passes the data between nodes in one direction, toward the output nodes. In this type of algorithm, the first step is to randomly initialize the weights before tuning the values to minimize the error. This is followed by passing the data forward between layers, applying the weights and evaluating the predictions. These steps will repeat for each layer. The output for the layers in the network can be represented by the following equation [3]:

$$X_{i+1} = f_i(W_i X_i + b_i) \tag{8}$$

In this equation $X_i$ is the input and $f_i$ is the activation function in layer $i$. The output of said layer, $X_{i+1}$, becomes the input for the subsequent layer. Once trained the network should be able to approximate a function that can fulfils the design goal for the algorithm [3].

There are several problems that can arise if the network is not properly trained. Things like overfitting, results being trapped by local minima, and the vanishing gradient problem can all affect the results depending on how the algorithm is structured. For this reason, many techniques such as altering the weight initialization technique have been used to help mitigate some of these issues [3].

### b. Particle Swarm Optimization (PSO)

The Particle Swarm Optimization (PSO) algorithm is a metaheuristic global optimization paradigm that has received prominence over the last two decades owing to its easy implementation with efficient performance over multidimensional problems that cannot be solved using traditional deterministic algorithms. Its design was inspired by the movement of a flock of birds [2]. For the scope of the algorithm, each “bird” would be represented by a particle, while the “flock” is represented by the swarm. To utilize the swarming mechanisms of the PSO, each particle is initialized by the algorithm and then placed in the search space. After this initialization, the program is run iteratively, with each particle utilizing a velocity calculation to determine how they “fly” through the search space. In this computation, three different parameters are used. First, the inertial weight of the particle, which represents how much the previous direction of movement influences the next.

Second, the social weight, which determines how much the global best position influences the particles' movement. Finally, the cognitive weight, which determines how much the location of an individual particle's personal best location influences its movement. The equations for calculating the velocity and position after each iteration is as follows [1]:

$$v_{ij}(t+1) = \omega v_{ij}(t) + C_1 r_1(t)\left(p_{ij}(t) - x_{ij}(t)\right) + C_2 r_2(t)\left(p_{gj}(t) - x_{ij}(t)\right) \tag{9}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{10}$$

$C_1$ and $C_2$ are constants representing the social and cognitive weights, $r_1$ and $r_2$ are independent and identically distributed random numbers between 0 and 1. $x_{ij}$ and $v_{ij}$ represent the position and velocity respectively of the $i_{th}$ particle in $j_{th}$ dimension whereas $p_{ij}(t)$ and $p_{gj}(t)$ are the personal best (*pbest*) and global best (*gbest*) positions. In the velocity update scheme, $\omega$ represents the inertial weight of the $i_{th}$ particle, and the following two terms add in guided vectors towards areas of attraction (also known as local attractors) in the particle's direction of movement. The personal best update uses a greedy update scheme with regards to a cost minimization goal, modeled using the following equation:

$$f\left(x_i(t+1)\right) < f\left(p_i(t)\right) \Rightarrow p_i(t+1) = x_i(t+1)$$
$$else\; p_i(t+1) = p_i(t) \tag{11}$$

In this equation set $f$ is the cost, and $p_i$ is the personal best of the particle. The global best $(p_g)$ is the element with the lowest cost result from the set of personal bests of a particular particle. While PSO can cover a large hyperplane of the search space, finding the right values of the parameter set which leads to the best outcome is a time-consuming task that requires careful analysis.

## V. Experimental Setup

### *A. Parameter Settings*

In our analysis of the clustering benchmarks using the K-Means implementation, the centroids were determined through a randomly selected group of points. To ensure adequate variability in the starting points, each run of the algorithm had its starting points determined independently and the mean and standard deviation of the performance metrics were reported. The K-Means algorithm expects the number of clusters to be set beforehand, so care was taken to do the same for each of the datasets considered.

The Fuzzy C-Means implementation is like K-Means in how starting centroid locations need to be determined before the model begins to run. Just as with K-Means the centroids were randomly chosen, however unlike K-Means the starting centroids were randomly chosen locations within the search space of the data set instead of from the existing points. In addition to this, the membership matrix needed to be initialized. Since each data point has 3 factors determining membership with all clusters, each point was given a corresponding list of factors, and was randomly assigned one cluster to have a strong correlation with.

The parameters for the Feed-forward Neural Network with Particle Swarm Optimization (FNN-PSO) for the experiments were the inertia weight ω and the cognitive and social parameters $C_0$ and $C_1$. For the inertia weight an upper bound and a lower bound were chosen to be the range determinants and during execution a randomly generated value inside this range was chosen. This was repeated for each particle in the swarm during the optimization phase of the neural network. To ensure optimal results for all benchmarking data sets the inertia weight range and the social and cognitive parameters were tailored to each individual data set. Table 1 contains the values used for each data set. The Neural Network parameters also needed to be initialized for each run. To ensure the data set could be properly processed the number of input nodes was set to the number of attributes for each individual data set, and to compliment this, the number of output nodes was set to one less than the number of input nodes. As for the number hidden layers, a static 20 was used for all data sets. This was done to add a bit more consistency to the repeated runs of each data set.

**Table 1. Neural Network with PSO Parameter Information**

| | Inertia Weight Range | Cognitive Factor | Social Factor |
|---|---|---|---|
| Iris | 0.7 - 0.9 | 0.5 | 0.7 |
| Breast Cancer | 0.9 - 0.9 | 0.4 | 0.1 |
| Seeds | 0.7 - 0.9 | 0.9 | 0.1 |
| Mammographic Mass | 0.6 - 0.9 | 0.1 | 0.9 |
| Sonar | 0.9 - 0.9 | 0.2 | 0.1 |

While there is overlap between many of the factors utilized for this experiment, given how sensitive the implementation was to each of them, adequate fine tuning was needed to ensure the algorithm generated the best results. The best example for this collection is the Seeds data set. Unlike K-Means, the Neural Network with PSO did not have the number of categories/clusters initialized at the start, so the algorithm needed to generate the categories for each run. The Seeds data set contains 3 categories, and while outside of this the data itself does not appear to be much more complex than the other sets, getting a properly usable set of models from the initialization was much harder for it than the other data sets. Seeds took longer than any other data set to find the proper parameters for, and still was the data set the model struggled the most with to get accurate results. The further the initialization parameters got from their optimal values the more difficult it was to get accurate results from the model. At times, the model would end up with a prediction of the incorrect number of categories, usually ending up stagnated on one prediction category.

## *A. Data sets*

The primary data sets utilized for the benchmarking are 5 well known ones. All but the Iris data set were taken from the UCI Machine Learning Repository [15], with the Iris data set being taken from the Sci-kit Learn library [16]. The data sets are described below:

1) R. A. Fisher's Iris data set covering three species of Iris flowers (Setosa, Versicolor, and Virginica) containing a total of 150 instances with 4 attributes each. The attributes are as follows: Sepal Length in cm, Sepal Width in cm, Petal Length in cm, and Petal Width in cm.

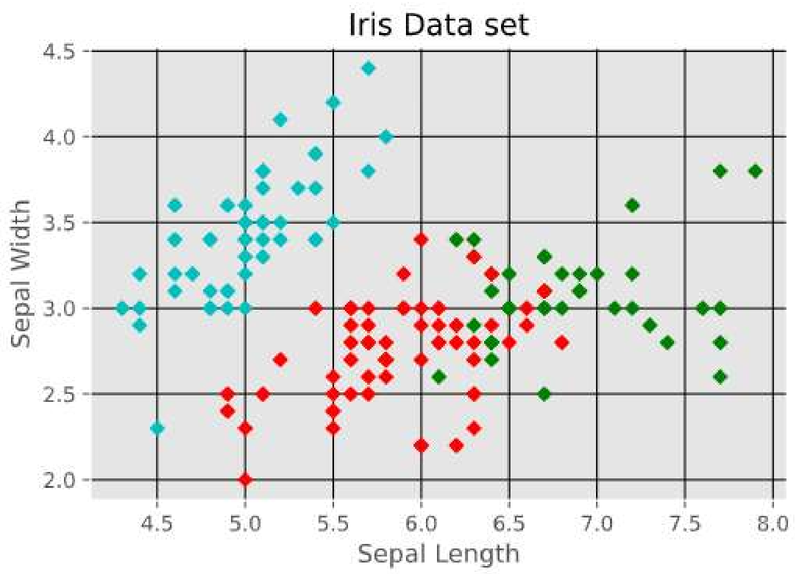

**Figure 3. Clustering using K-Means on the Iris data set**

2) Breast Cancer Wisconsin (Original) data set consisting of 699 instances and containing 10 attributes. Instances are classified into one of two categories, benign or malignant. The attributes are as follows: Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, and Mitoses.

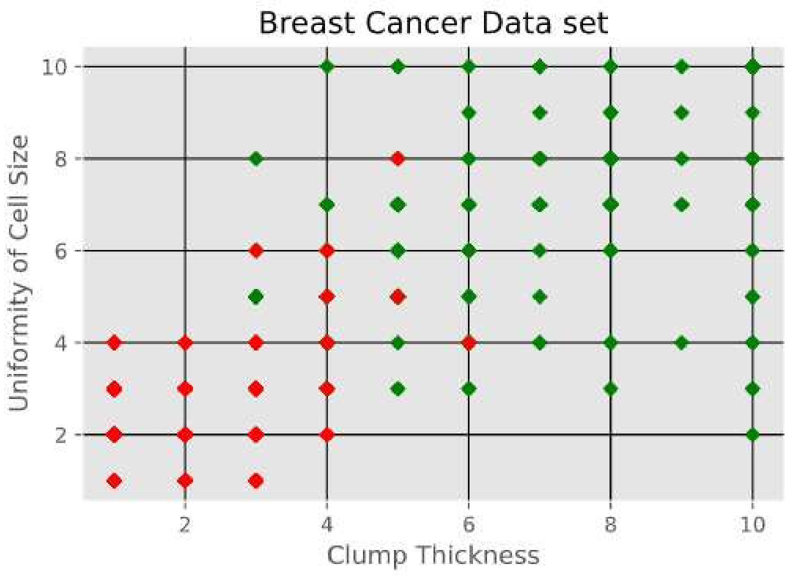

**Figure 4. Clustering using K-Means on the Breast Cancer data set**

3) Seeds data set, used to identify 3 different varieties of wheat: Kama, Rosa, and Canadian. The data set contains 210 instances with 7 attributes each. The attributes are as follows: Area, Perimeter, Compactness, Length of Kernel, Width of Kernel, Asymmetry Coefficient, and Length of Kernel Groove.

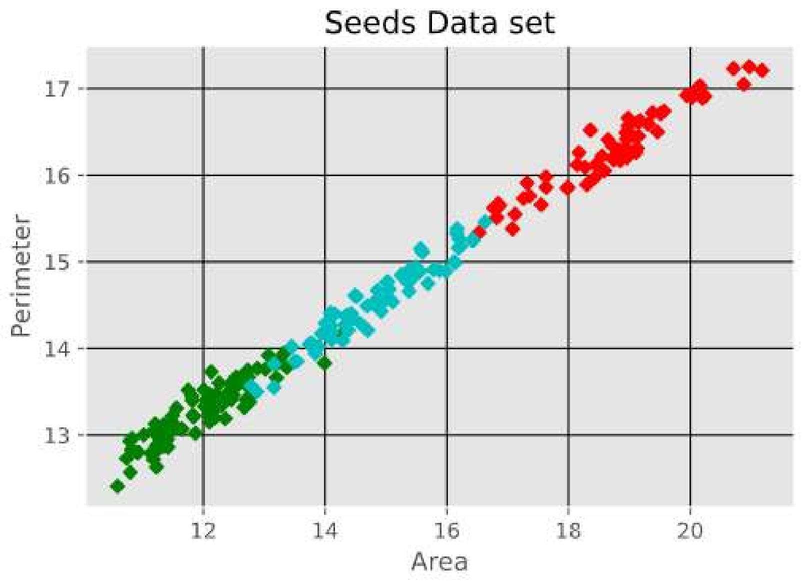

**Figure 5. Clustering using K-Means on the Seeds data set**

4) Mammographic Mass data set containing 961 instances and used to classify data into two categories, benign and malignant. Each instance contains 6 attributes based on BI-RADS data and the patient's age. The attributes are as follows: BI-RADS Assessment, Age, Shape, Margin, Density, and Severity.

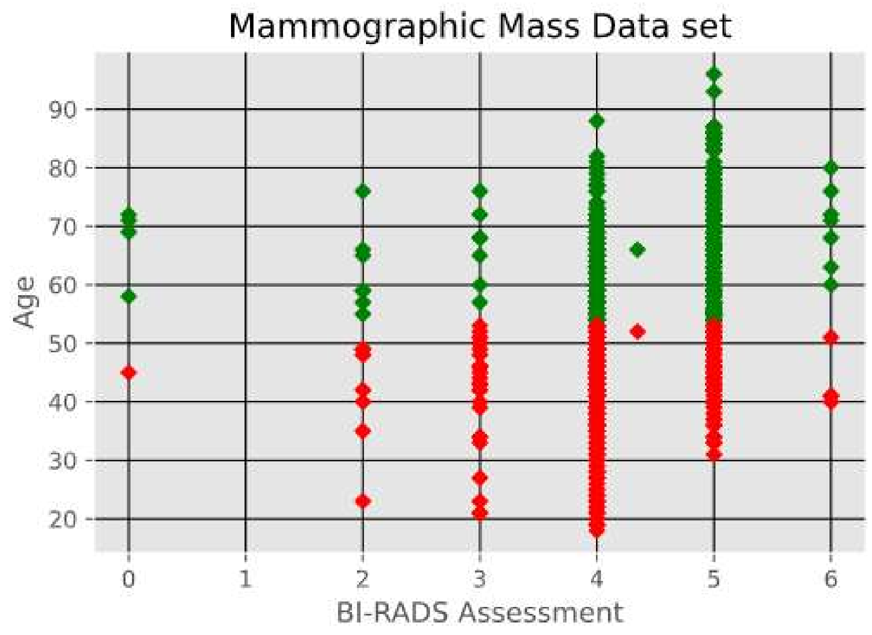

**Figure 6. Clustering using K-Means on the Mammographic Mass data set**

5) Sonar Data set, consisting of 208 instances with 60 attributes each. The instances are classified into either mines or rocks. The attributes represent the energy within a specific frequency band.

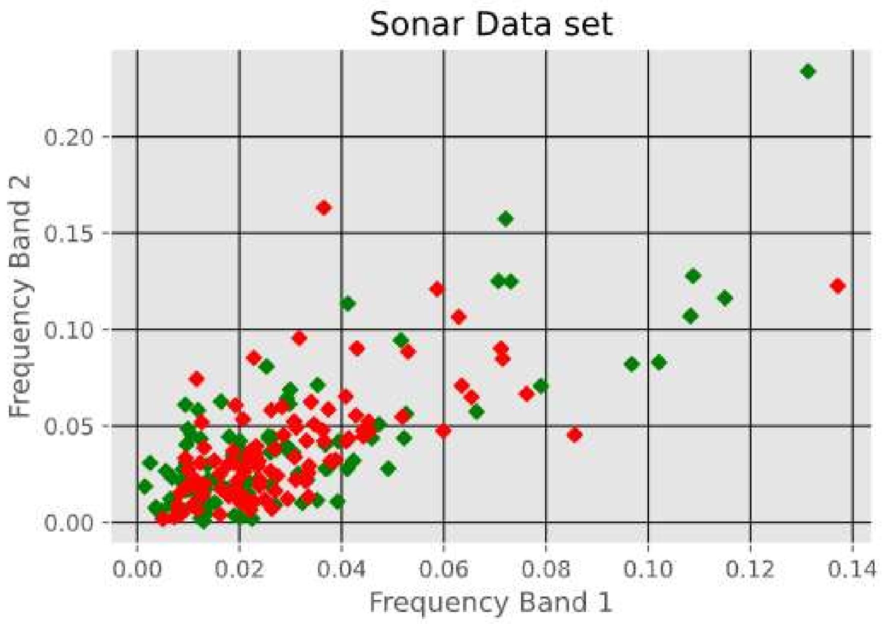

**Figure 7. Clustering using K-Means on the Sonar data set**

Missing Value Replacement: In the cases of the Breast Cancer data set and the Mammographic Mass data set, the data contained instances where attributes were missing, and thus needed to be dealt with for the algorithms to function. To this end the average value for each attribute was calculated, and any missing values were replaced with that average.

B. *Performance Indices*

Measuring the performance indices allows us to determine the effectiveness of a given model to correctly classify instances in each data set. The clustering indices are as follows:

(a) Accuracy: Also known as the Rand index. The accuracy is described as the percentage of correct decisions made by the algorithm. The formula for accuracy is as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{12}$$

In this equation $TP$ is the number of true positives, $TN$ is the number of true negatives, $FP$ is the number of false positives, and lastly $FN$ is the number of false negatives.

(b) F-Score: Another means of calculating accuracy. It is calculated using the precision and recall of a model the formula for F-Score is as follows:

$$F = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{13}$$

(c) Inter-cluster Distance: The sum of the distances between each cluster centroid. Larger values indicate a greater separation between clusters, meaning less overlap between clusters in the model. The formula for inter-cluster distance is as follows [18]:

$$Inter - clust\ dist = \min\,(||c_i - c_j||)^2 \tag{14}$$

Where $c_j$ represents the centroid of cluster $j$.

(d) Intra-cluster Distance: The sum of the distances between individual data points and their respective parent centroid. Smaller values indicate more compact clusters and are therefore desired. The formula for intra-cluster distance is as follows [18]:

$$Intra - cluster\ dist = \frac{1}{n}\sum_{j=1}^{k} ||x_{i-c_j}||^2 \tag{15}$$

$k$ is the number of clusters, $c_j$ represents the centroid $j$, and $n$ refers to the number of data instances. The section $||x_{i-c_j}||^2$ represents the distance between data instances and their respective centroid.

(e) Quantization Error: The sum of the distances between data points and their parent centroid, divided by the total number of data points belonging to the cluster, then summed over all clusters and averaged for all clusters in the model. The formula for quantization error is as follows [18]:

$$Quantization\ error = \frac{\sum_{j=1}^{k}\left\{\frac{\sum_{i=1}^{N_j}||x_{i-c_j}||^2}{N_j}\right\}}{k} \tag{16}$$

In this equation, $k$ is the number of clusters, $c_j$ is the centroid of cluster $j$, $N_j$ represents the number of data points in cluster $j$, and finally the section $||x_{i-c_j}||^2$ represents the distance between data instances and their respective centroid.

As the PSO Neural Network is not a clustering algorithm rather performs a classification task, it uses a different indicator to measure performance. It is as follows:

(a) Global Best Loss: The best value generated by the loss function for all iterations of the optimization algorithm.

(b) Indices such as F-Score and Accuracy for the data sets are also calculated and included in the results.

The algorithms are implemented in Python 3.9.1 and executed with an AMD Ryzen 5 3600 6-Core Processor at 3.6GHz. Experimental results for 50 trials are tabulated and then analyzed. Table 2 details the data sets used in this work.

## C. *K-Means*

For the K-Means implementation, the vanilla version of the algorithm was not altered. Before the start of the algorithm the data set needed to be split into the base data and classifications associated with each data point. To achieve this, two containers were set, with the first containing a list of the attribute values for each data point with any missing values replaced, and the second containing a list of the corresponding classifications. From this point the centroids were determined. Originally, the centroids were statically chosen from the data set, but this method was replaced with a randomly selected set of points. While the original method was able to return predictable results, the goal of the experiment was to study the algorithm’s overall performance across a range of choices, and by varying the initialization of the centroids a much larger range of the feature space was available to be investigated. The algorithm was set to terminate after a set number of iterations were completed if there were no significant performance gains.

After the algorithm finished each of its runs the final positions of the centroids were utilized to calculate the performance indices for the run. Once calculated, the results were stored in a separate table to later be collected and averaged. The mean value and the standard deviation over 50 runs were calculated and stored for comparison. The results for the K-Means runs can be found in Table 3.

### C.1 Trapping in Local Optima

For the K-Means algorithm, and many clustering algorithms, the starting points can have a strong influence on the performance of the model. Just as there are good starting points that allow the algorithm to find appropriate centroids to represent the clusters, there are also starting points that cause the algorithm to incorrectly model the data. The Iris data, as an example, is quite susceptible to poorly initialized centroids. Figure 8 shows a model with centroids that correctly represent the data, and Figure 9 shows a model where the starting centroids cause the model to incorrectly model the data.

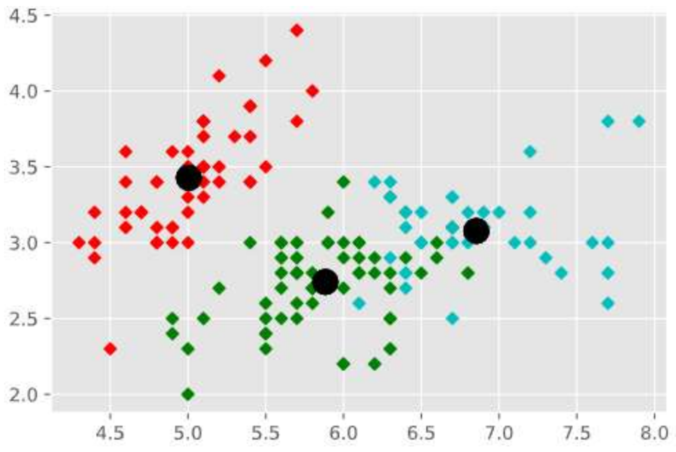

**Figure 8. Correctly modeled Iris clusters**

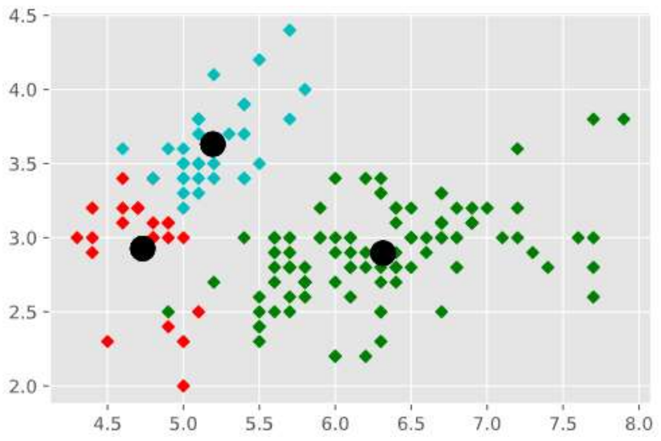

**Figure 9. Incorrectly modeled Iris clusters**

In the dimensions of the data the grouping on the top left of the graph represents a single cluster, while the larger mass is split into the remaining two. However, in the situation where the initialization places two of the centroids in the upper left group, there is a chance that the algorithm will assign two of the clusters to the upper left grouping and leave the remaining centroid to represent the larger grouping in the lower right. This can come as a result of the K-Means algorithm's weakness with regards to getting stuck on local minima [2]. The random initialization of the points is what was used to help mitigate some of the sensitivity to different initialization.

## *D.* Fuzzy C-Means

As with the K-Means algorithm the base functionality of the algorithm was not altered. Before processing the data was split into two parts. One containing the data points themselves, and one the contained their respective classifications. From there, the centroids were initialized. Unlike K-Means, for the Fuzzy C-Means implementation the points were not initialized from points in the data set. Instead, the data was used to set up the range of space where the values are located, and a random set of positions in that space were chosen to act as the centroids. Also unlike K-Means, the membership matrix needed to be initialized. To do so a membership array was created for each data point in the set, with a number of values equal to the number of classifications/clusters. After the matrix itself was created, each data point was randomly assigned one cluster to have a strong relationship with. As with the other algorithms, the randomness introduced in the initialization facilitated the study of the algorithm's performance over a larger range of the feature space. The algorithm was set to terminate after a set number of iterations were completed if there were no significant performance gains.

After each run of the algorithm was completed, the final positions of the centroids were used to calculate the various performance indices of the run. After being calculated, the results were stored in a separate table for later collection and averaging. The mean value and standard deviation over 50 runs were calculated and stored for comparison. The results from the FCM runs can be found in Table 4.

### *E. Feed-Forward Neural Network tuned by a Particle Swarm Optimization Algorithm*

While the results of the K-Means and FCM algorithms were promising, to better understand the grouping of the benchmarking data from a predictive analysis point of view, a Feed-Forward Neural Network using a PSO optimizer was implemented. Using a supervised learning algorithm such as this that functions quite differently than the clustering algorithms utilized in the experiment allowed us to look at the problem with a different perspective. The results were good, as was expected, but the range of values were notably different than the ones returned by the other algorithms, and as such different deductions could be made about the functionality of both types of algorithms utilized.

As for the implementation itself, like the K-Means implementation, the data needed to be split into groups of the data and the classifications. The training data and labels were set up with the raw data and the categories respectively, and any fixes/replacement of values to the data set were applied. Once finished, the values for all necessary nodes were initialized. For this implementation, the dimensionality of number of inputs, outputs, and hidden nodes were computed and care was taken to fit the network with the data. The number of input nodes was set to match the number of features for the data set being analyzed, and the number of output nodes was set to be one less than the number of input nodes. For the number of hidden nodes, the number was set to 20 for each data set. Furthermore, the data sets were split into training and testing groupings, with 70% of the data used to train the model and the remaining 30% used to evaluate the accuracy of the model.

After each of the runs the accuracy, global best loss, and F-score are calculated and stored for later use. The results from 50 runs were stored, and the mean and standard deviation were calculated and stored in Table 5.

## VI. Results and Analysis

Tables 3 through 5 contain the results from the three algorithms used in this experiment. They contain the mean and standard deviation of performance metrics calculated over 50 runs. Figures 10 through 14 plot the accuracy of the K-Means algorithm over its iterations, and figures 15 through 19 plot the clustering model using K-Means in three dimensions. Figures 20 through 24 plot the accuracy of the FCM algorithm over its iterations, and figures 25 through 29 plot the clustering model using FCM in three dimensions. Figure 30 plots the loss curve of the FNN-PSO over 1000 iterations, and figures 31 through 35 compare the Ground Truth mapping of the data verses the predicted mappings from the FNN-PSO. Finally, figures 36 through 40 model the mean and standard deviation of the accuracy for each distinct data set.

**Table 2. Data Set Information**

| | Number of Instances | Number of Attributes | Number of Categories |
|---|---|---|---|
| Iris | 150 | 4 | 3 |
| Breast Cancer | 699 | 10 | 2 |
| Seeds | 210 | 7 | 3 |
| Mammographic Mass | 961 | 6 | 2 |
| Sonar | 208 | 60 | 2 |

**Table 3. Results from K-Means Algorithm**

| | Accuracy | Inter Cluster Distance | Intra Cluster Distance | Quantization Error | F-Score |
|---|---|---|---|---|---|
| Iris | 84.3467%±11.5433% | 9.9448±0.5077 | 100.9036±9.1452 | 0.6525±0.0174 | 0.8435±0.1154 |
| Breast Cancer | 95.9484%±0.1393% | 27.5215±0.0287 | 3050.4339±1.2969 | 5.2425±0.0069 | 0.9595±0.0014 |
| Seeds | 89.2952%±0.2379% | 15.9245±0.1182 | 313.4652±0.2585 | 1.4967±0.0003 | 0.893±0.0024 |
| Mammographic Mass | 66.2042%±0.7268% | 47.8242±0.2174 | 6993.7479±10.2771 | 7.2945±0.0048 | 0.662±0.0073 |
| Sonar | 52.6442%±3.7956% | 2.5068±0.0124 | 235.1373±0.4677 | 1.127±0.0056 | 0.5264±0.038 |

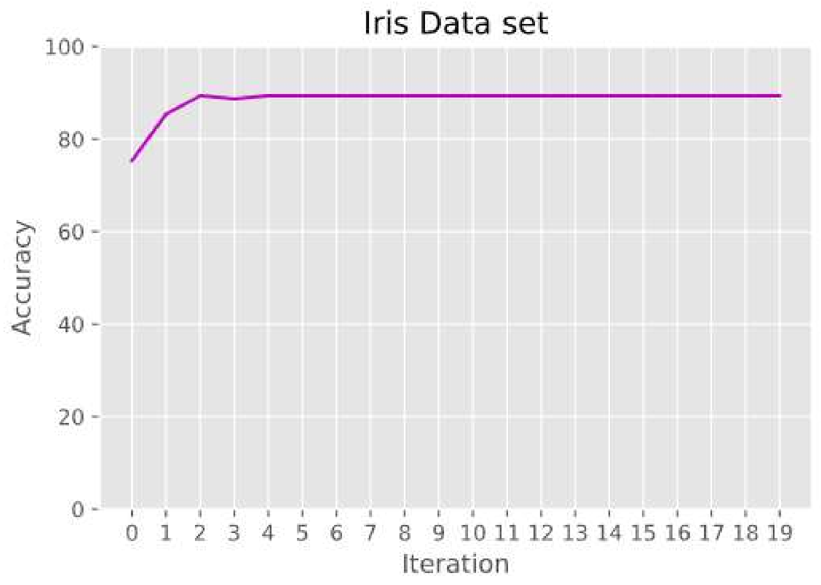

**Figure 10. Accuracy using K-Means on the Iris data set**

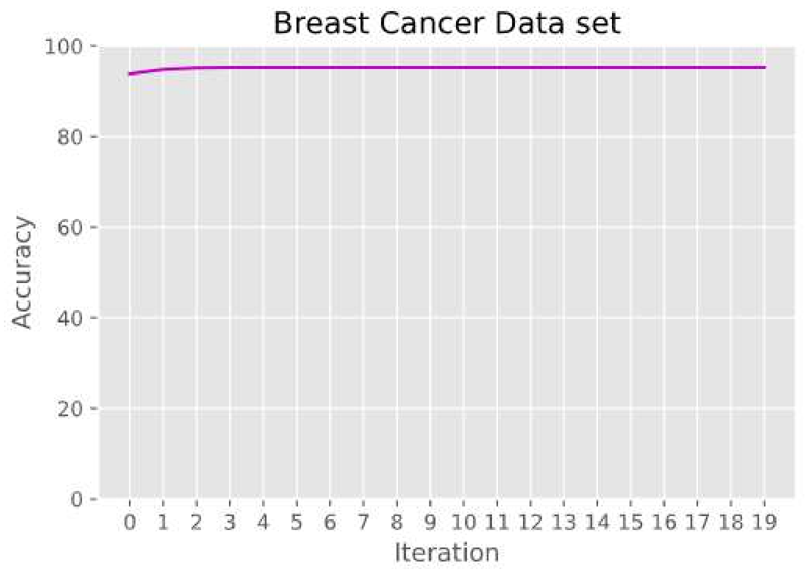

Figure 11. Accuracy using K-Means on the Breast Cancer data set

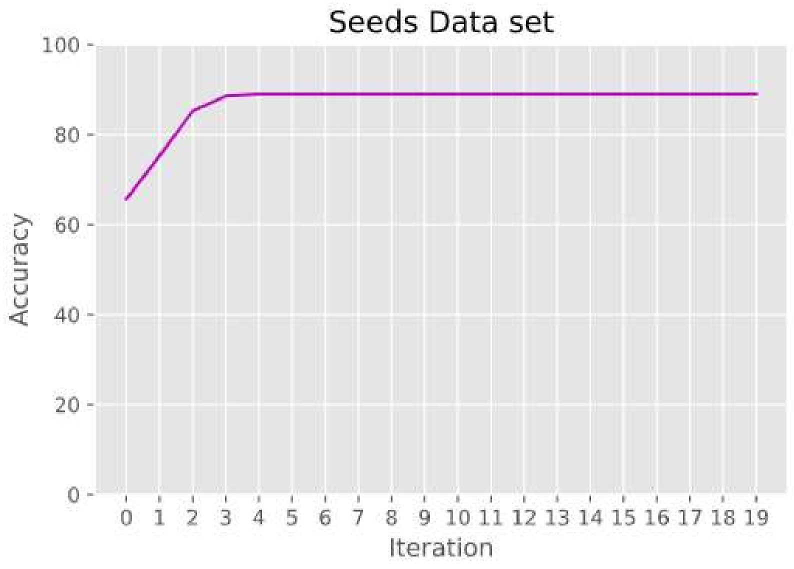

Figure 12. Accuracy using K-Means on the Seeds data set

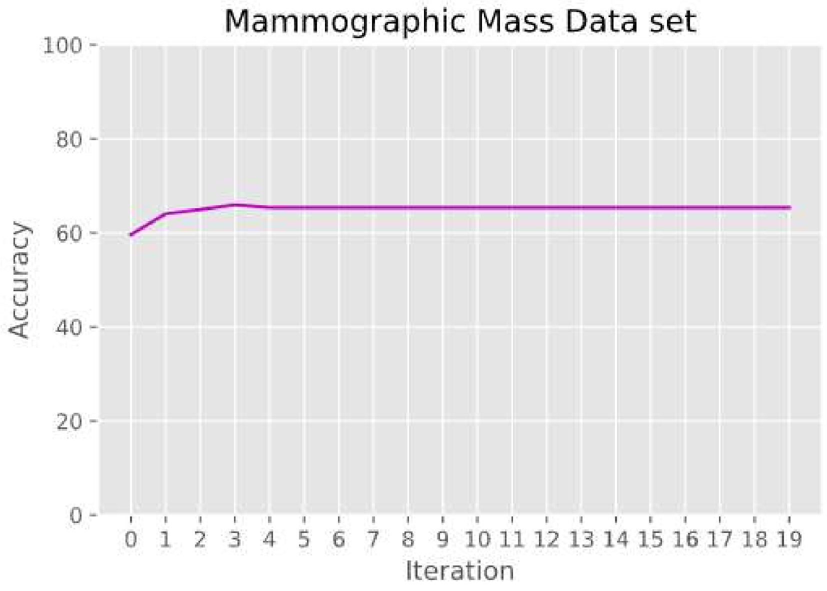

Figure 13. Accuracy using K-Means on the Mammographic Mass data set

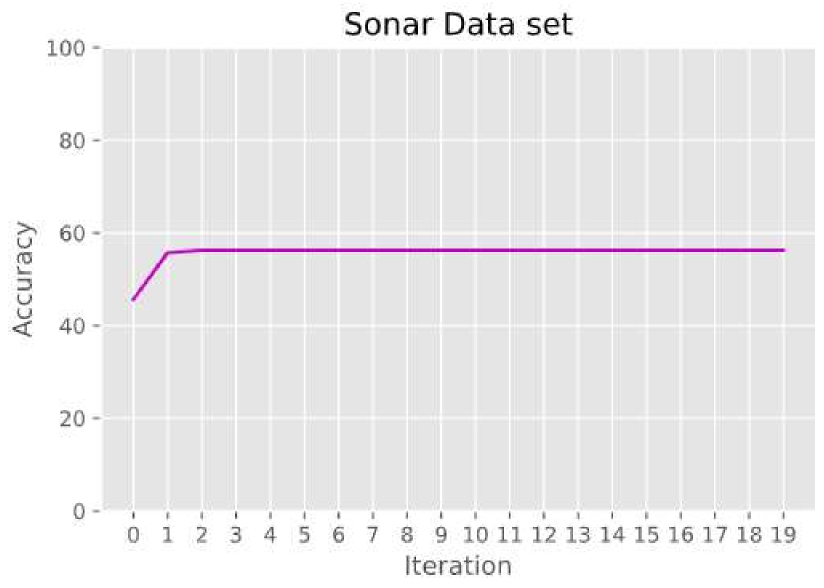

**Figure 14. Accuracy using K-Means on the Sonar data set**

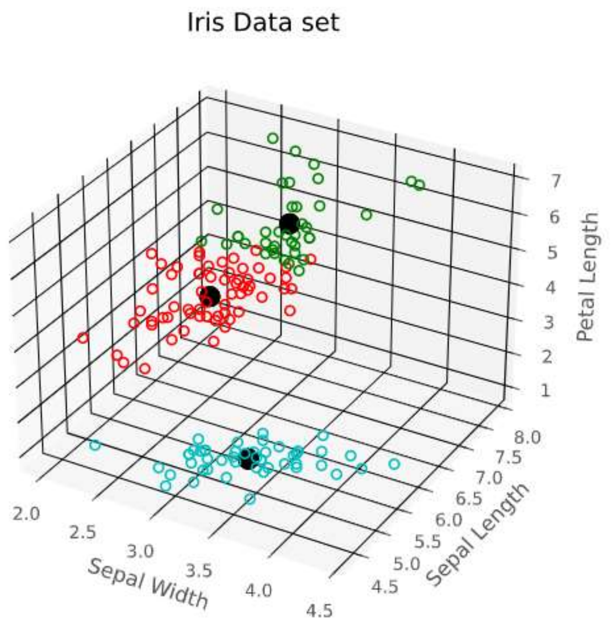

**Figure 15. Clustering using K-Means on the Iris data set**

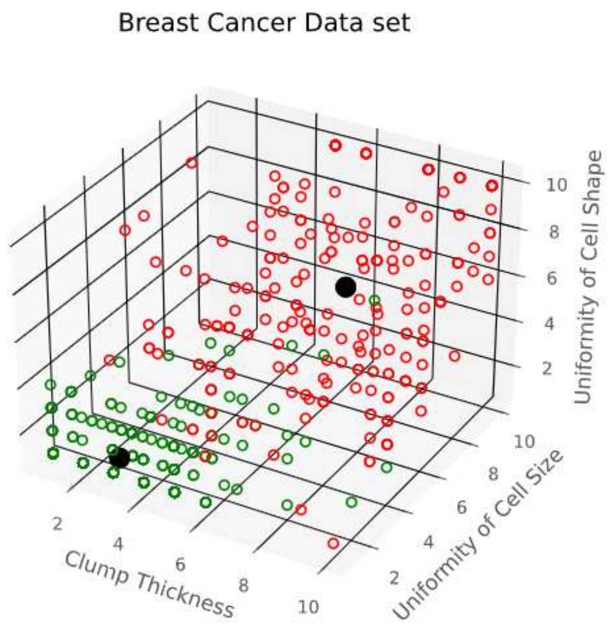

**Figure 16. Clustering using K-Means on the Breast cancer data set**

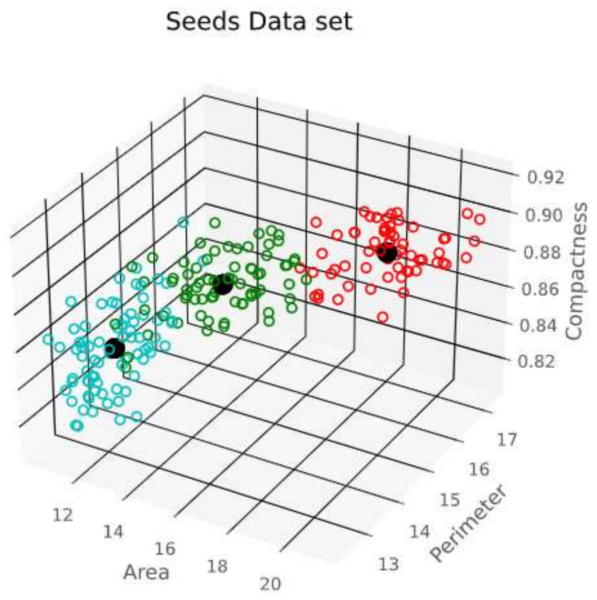

**Figure 17. Clustering using K-Means on the Seeds data set**

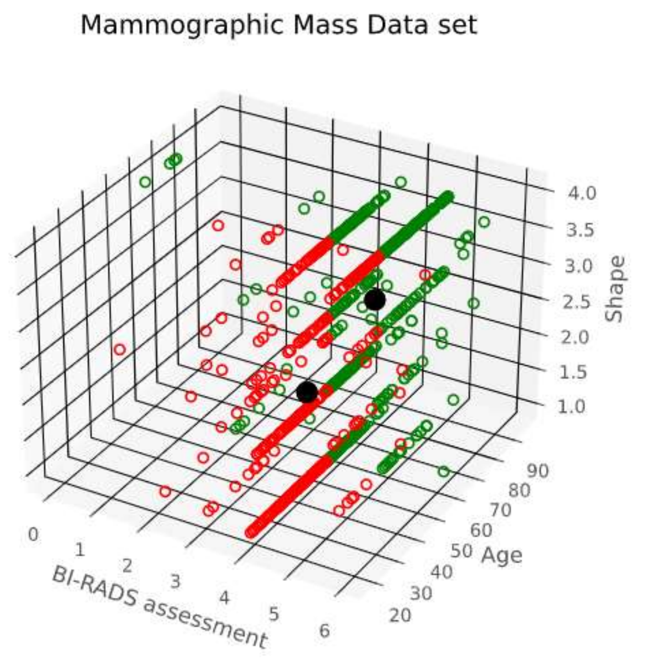

**Figure 18. Clustering using K-Means on the Mammographic Mass data set**

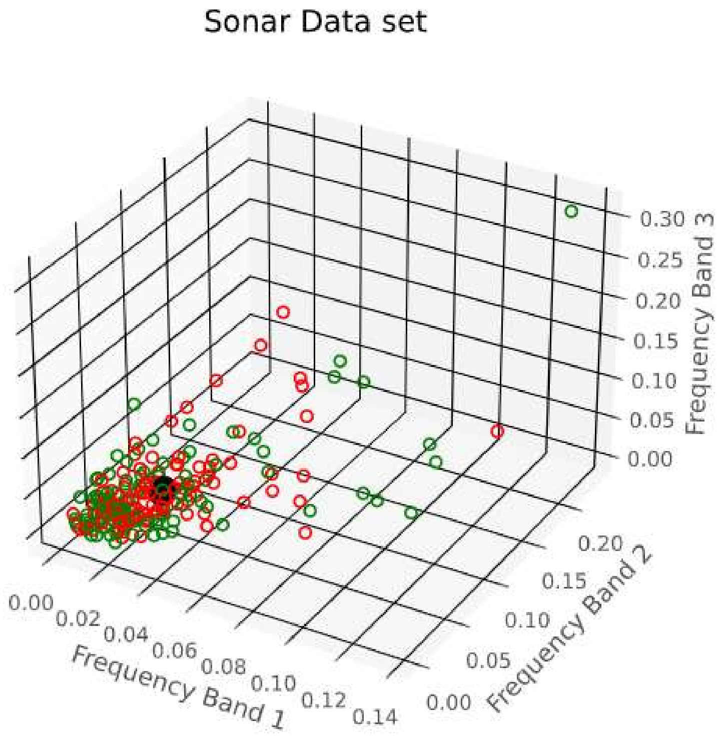

**Figure 19. Clustering using K-Means on the Sonar data set**

**Table 4. Results from Fuzzy C-Means Algorithm**

| | Accuracy | Inter Cluster Distance | Intra Cluster Distance | Quantization Error | F-Score |
|---|---|---|---|---|---|
| Iris | 88.6667%±1.1102e-14% | 9.994±3.7006e-15 | 96.9562±2.309e-14 | 0.646±9.5505e-17 | 0.8867±1.1102e-16 |
| Breast Cancer | 94.8424%±2.2204e-14% | 26.9631±7.4012e-15 | 2681.0984±4.5475e-13 | 4.7631±1.2243-15 | 0.9484±2.2204e-16 |
| Seeds | 89.5238%±1.1102e-14% | 16.1425±1.4921e-14 | 312.5735±4.9555e-14 | 1.4936±1.3688e-16 | 0.8952±1.1102e-16 |
| Mammographic Mass | 66.875%±2.2204e-14% | 48.7586±.572e-14 | 7023.1509±5.2389e-12 | 7.3333±4.432e-15 | 0.6687±2.2204e-16 |
| Sonar | 55.2885%±0.0000% | 1.8598±1.1508e-14 | 237.3494±4.5848e-13 | 1.139±1.9135e-15 | 0.5529±0.0000 |

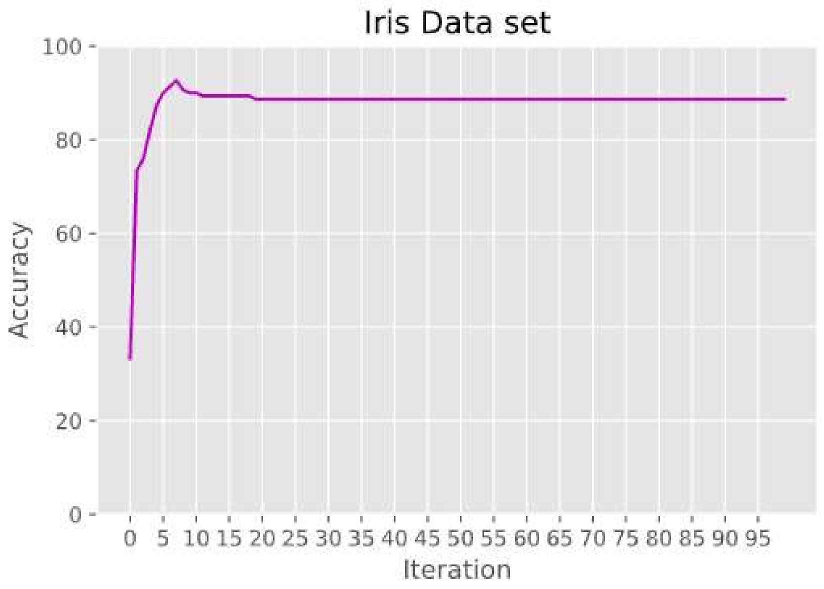

**Figure 20. Accuracy using FCM on the Iris data set**

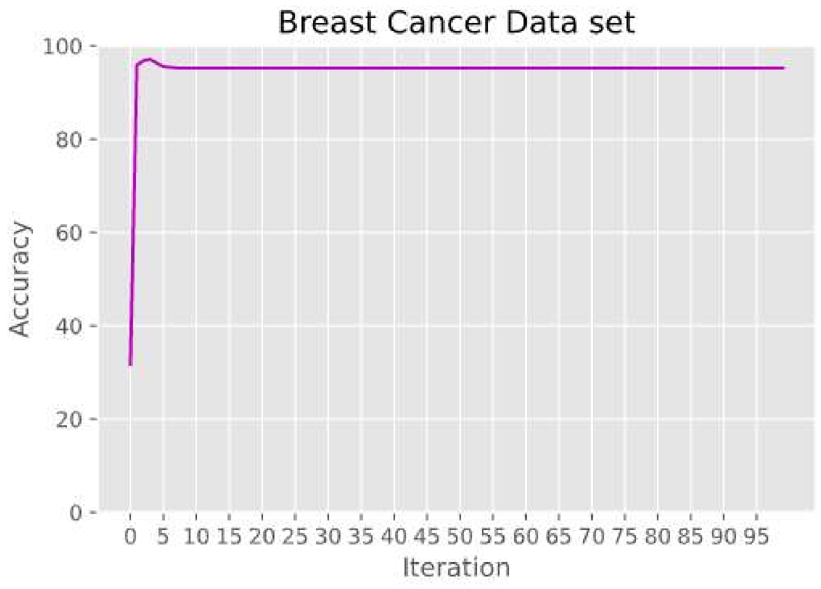

**Figure 21. Accuracy using FCM on the Breast Cancer data set**

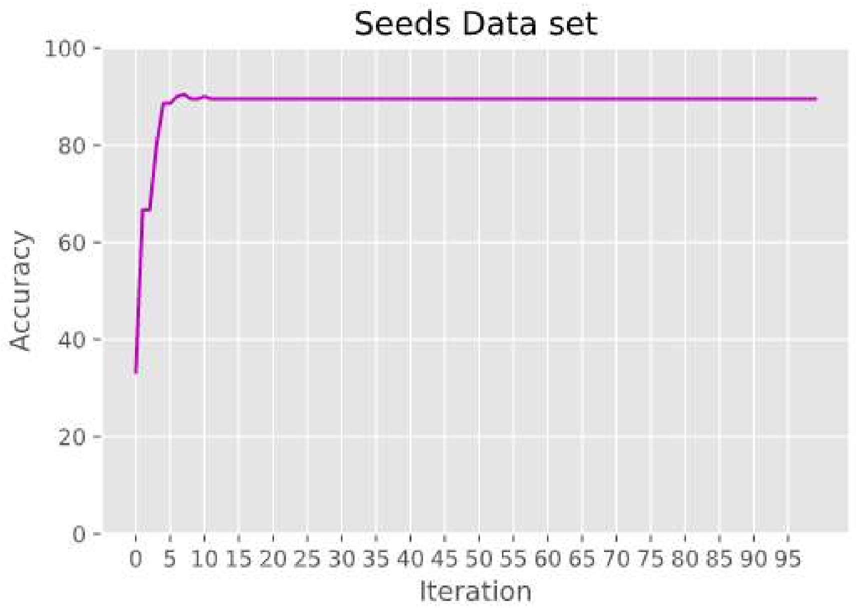

**Figure 22. Accuracy using FCM on the Seeds data set**

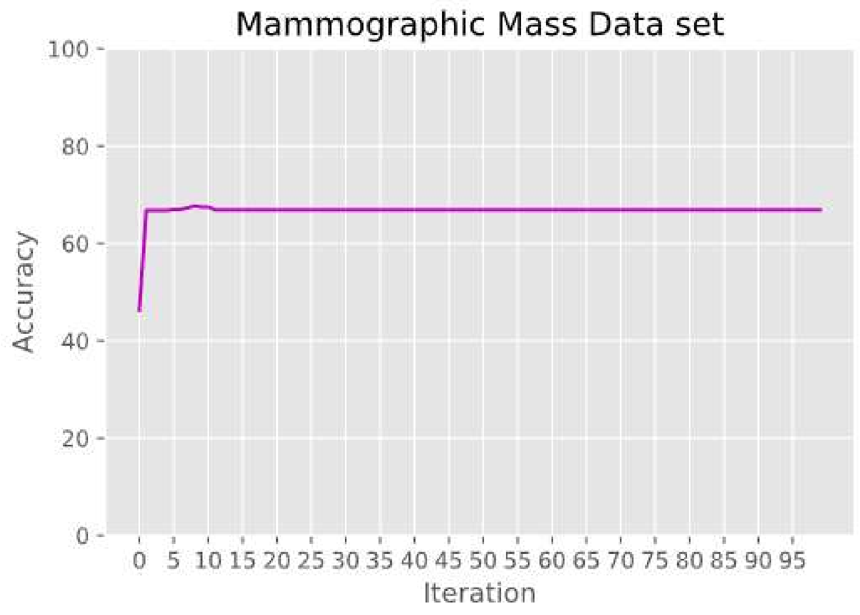

**Figure 23. Accuracy using FCM on the Mammographic Mass data set**

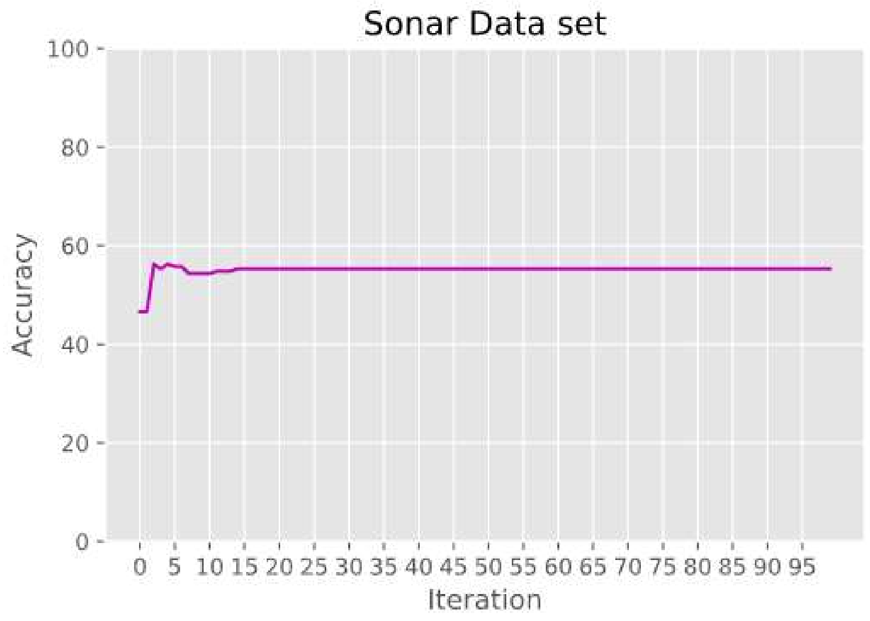

**Figure 24. Accuracy using FCM on the Sonar data set**

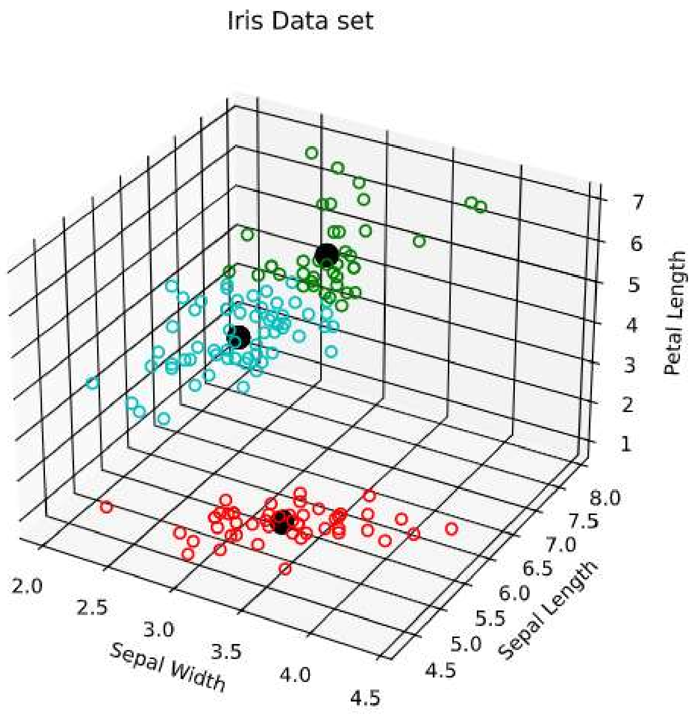

**Figure 25. Clustering using FCM on the Iris data set**

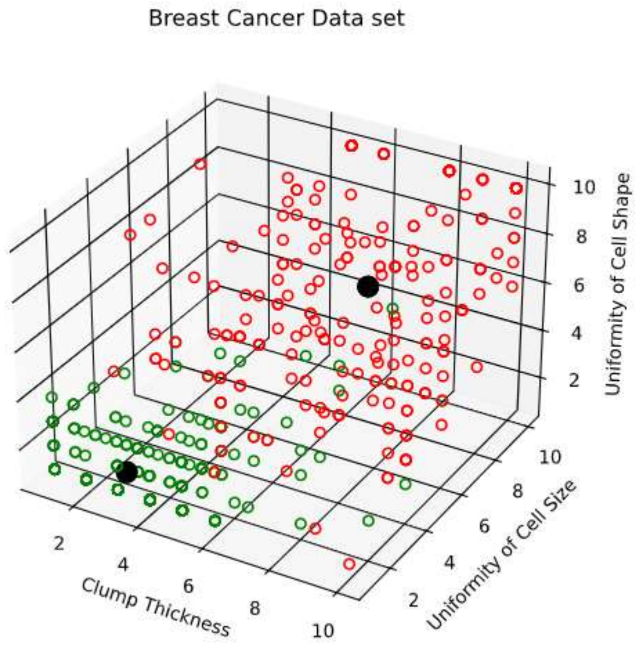

**Figure 26. Clustering using FCM on the Breast Cancer data set**

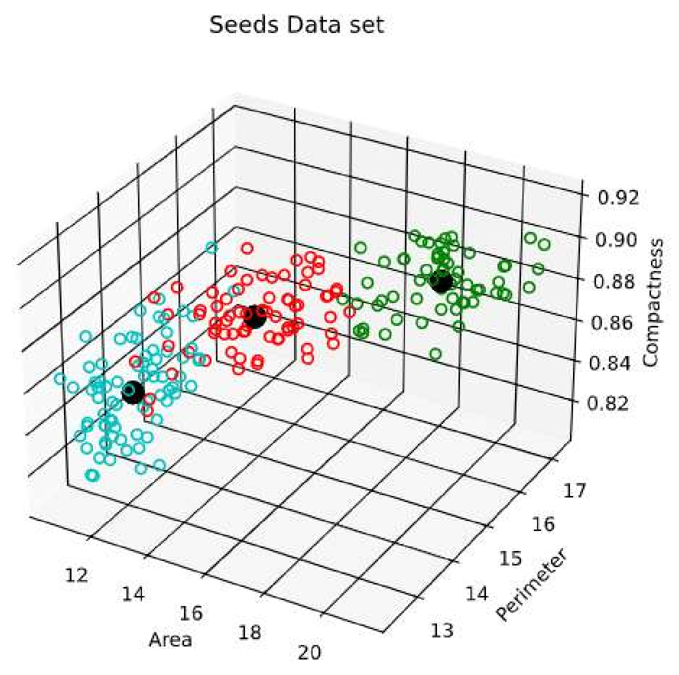

**Figure 27. Clustering using FCM on the Seeds data set**

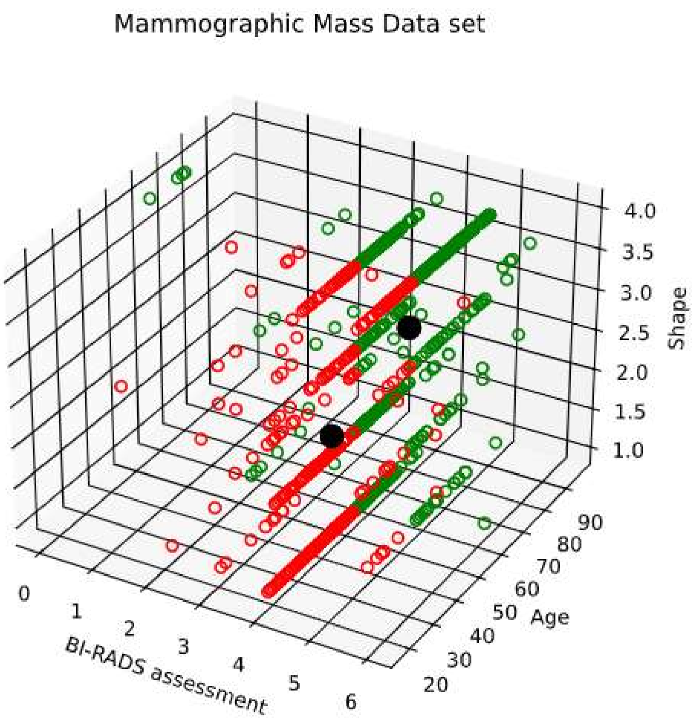

**Figure 28. Clustering using FCM on the Mammographic Mass data set**

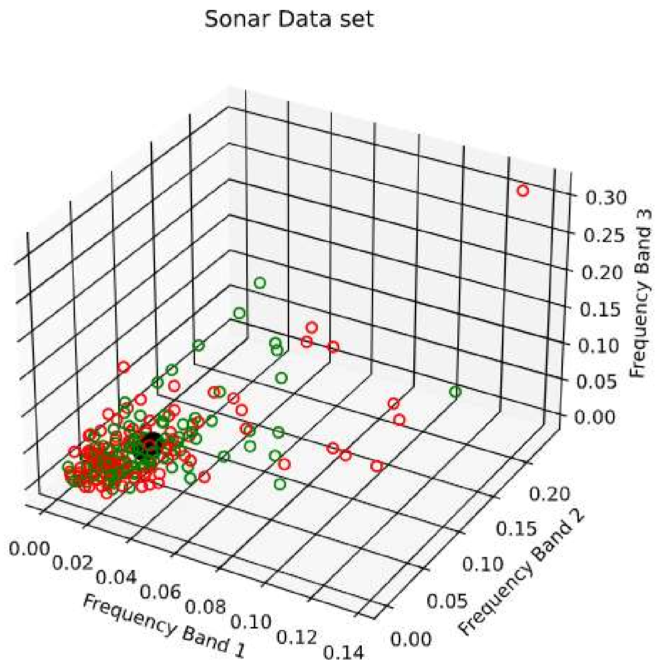

**Figure 29. Clustering using FCM on the Sonar data set**

**Table 5. Results from FNN-PSO Algorithm**

| | Accuracy | Global Best Loss | F-Score |
|---|---|---|---|
| Iris | 99.1556%±1.9317% | 0.0481±0.0353 | 0.9916±0.0193 |
| Breast Cancer | 94.0381%±5.9493% | 0.1541±0.109 | 0.9404±0.0595 |
| Seeds | 81.7143%±7.4872% | 0.4653±0.0995 | 0.8171±0.0749 |
| Mammographic Mass | 78.8264%±7.1728% | 0.4864±0.0638 | 0.7883±0.0717 |
| Sonar | 71.2666%±8.492% | 0.546±0.0809 | 0.7127± 0.0849 |

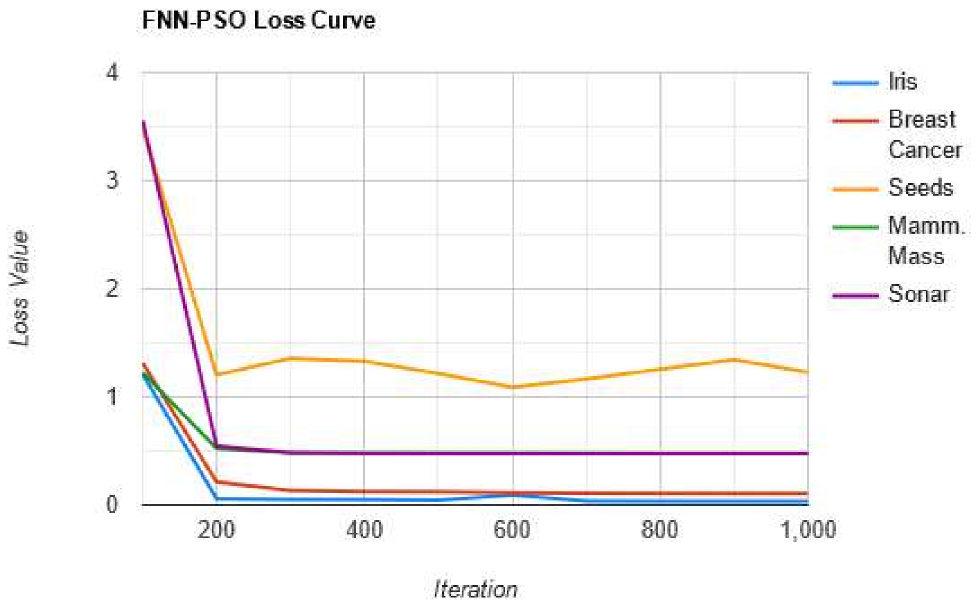

**Figure 30. Loss from FNN-PSO algorithm for each data set over 1000 iterations**

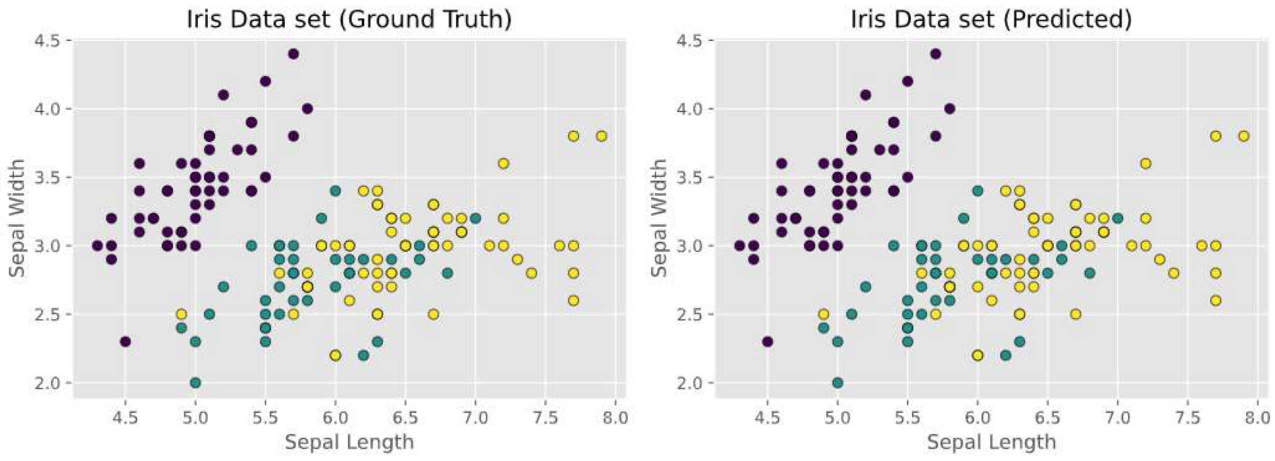

**Figure 31. Ground Truth vs. Predicted labels from FNN-PSO algorithm on the Iris data set**

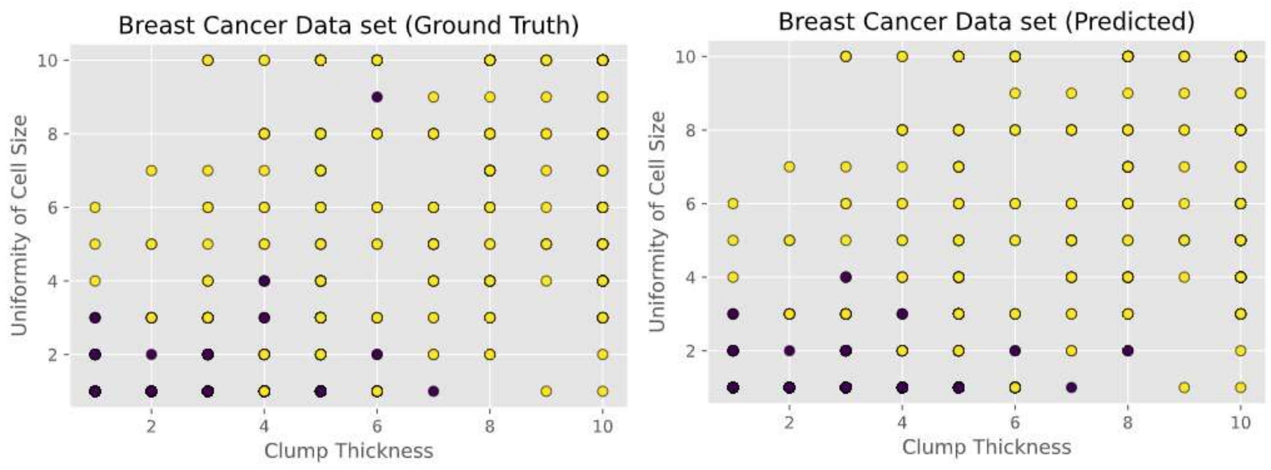

**Figure 32. Ground Truth vs. Predicted labels from FNN-PSO algorithm on the Breast Cancer data set**

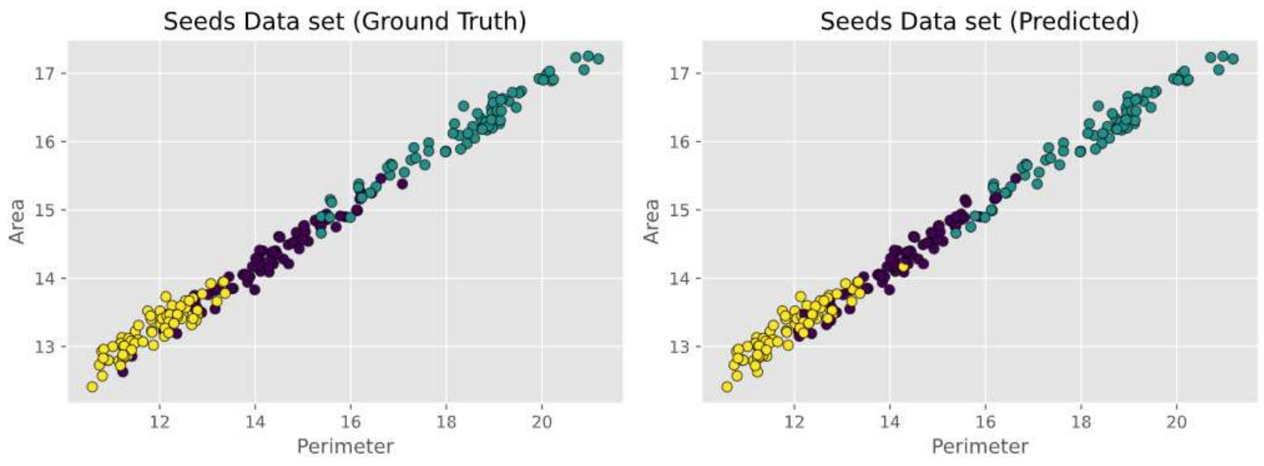

**Figure 33. Ground Truth vs. Predicted labels from FNN-PSO algorithm on the Seeds data set**

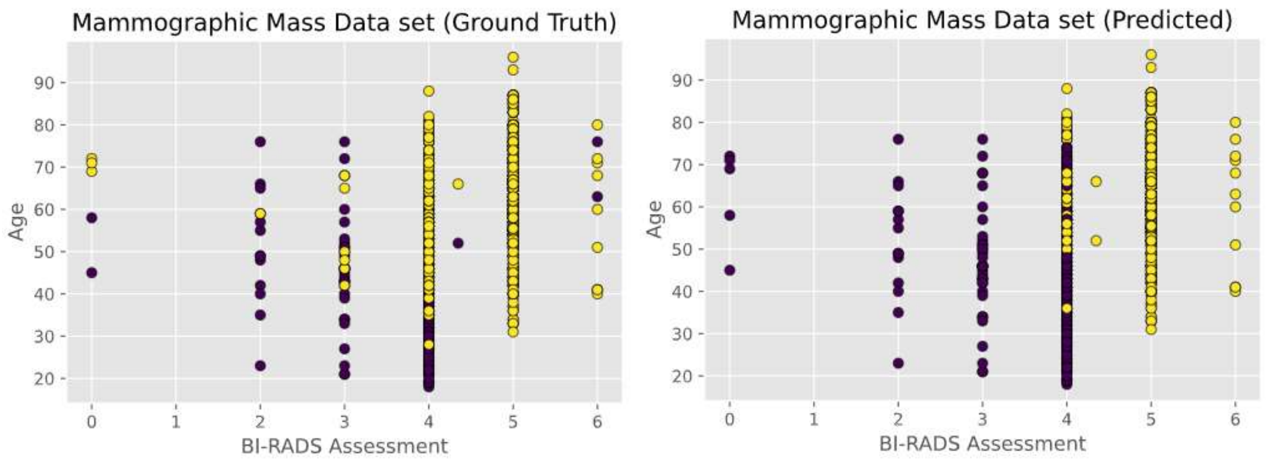

**Figure 34. Ground Truth vs. Predicted labels from FNN-PSO algorithm on the Mammographic Mass data set**

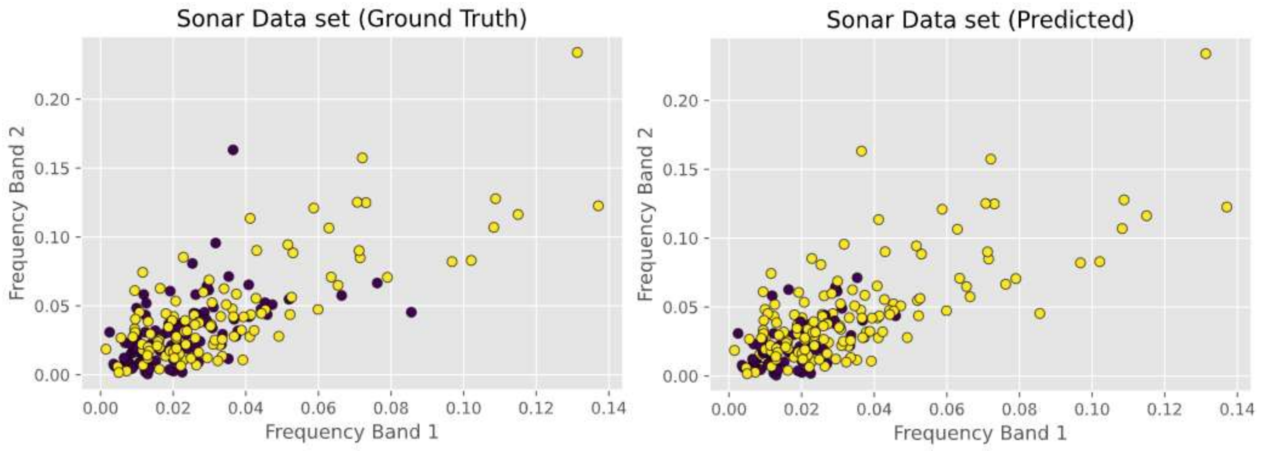

**Figure 35. Ground Truth vs. Predicted labels from FNN-PSO algorithm on the Sonar data set**

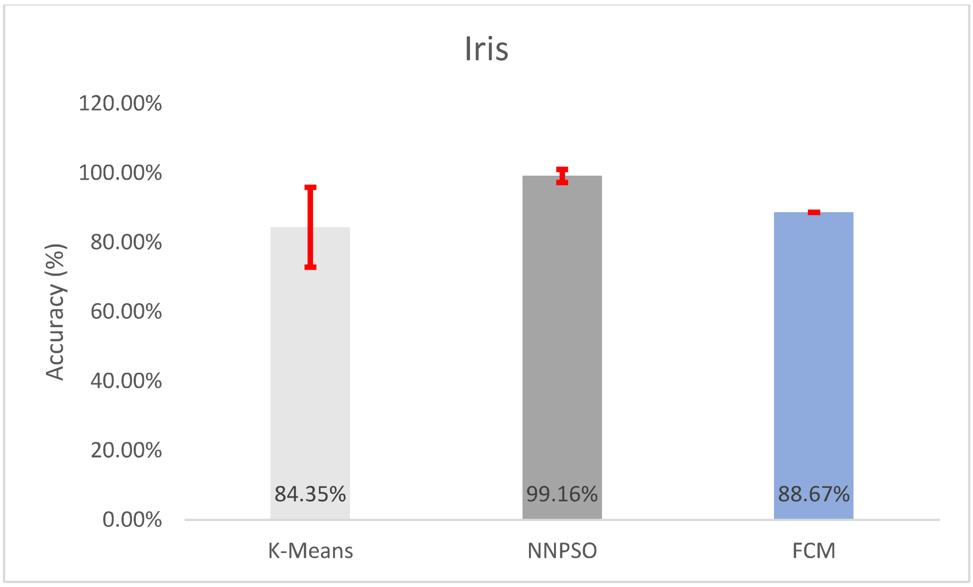

**Figure 36. Accuracy of the Algorithms on Iris**

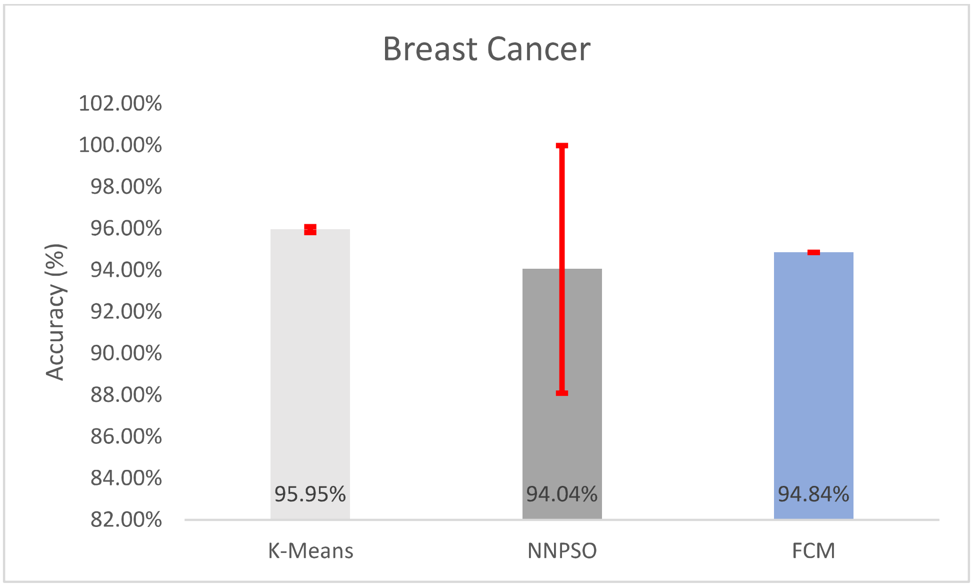

**Figure 37. Accuracy of the Algorithms on Breast Cancer**

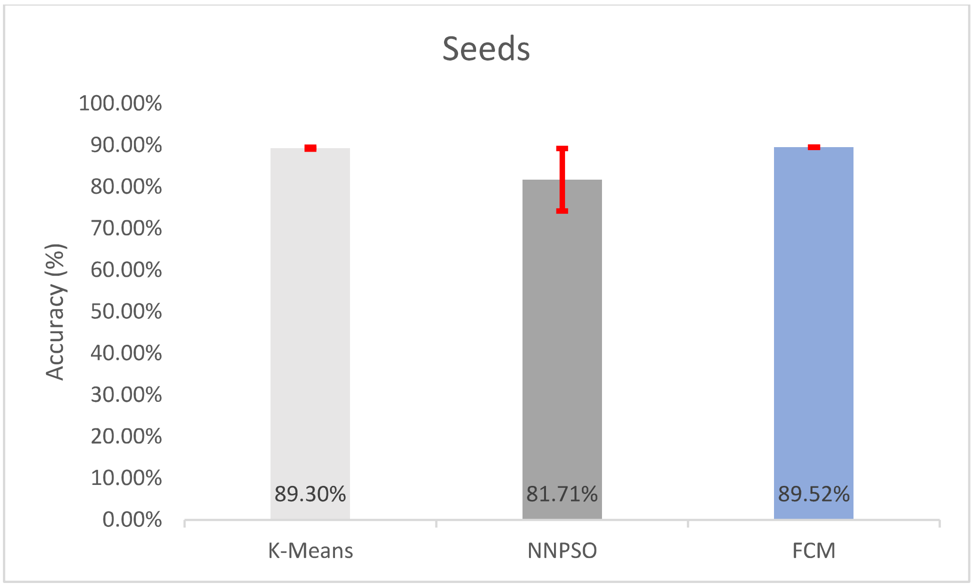

**Figure 38. Accuracy of the Algorithms on Seeds**

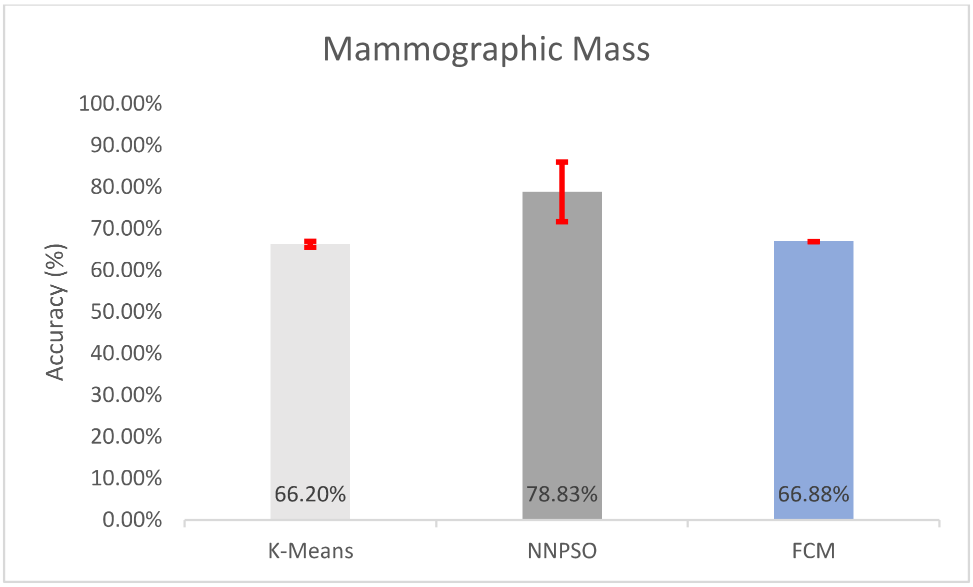

**Figure 39. Accuracy of the Algorithms on Mammographic Mass**

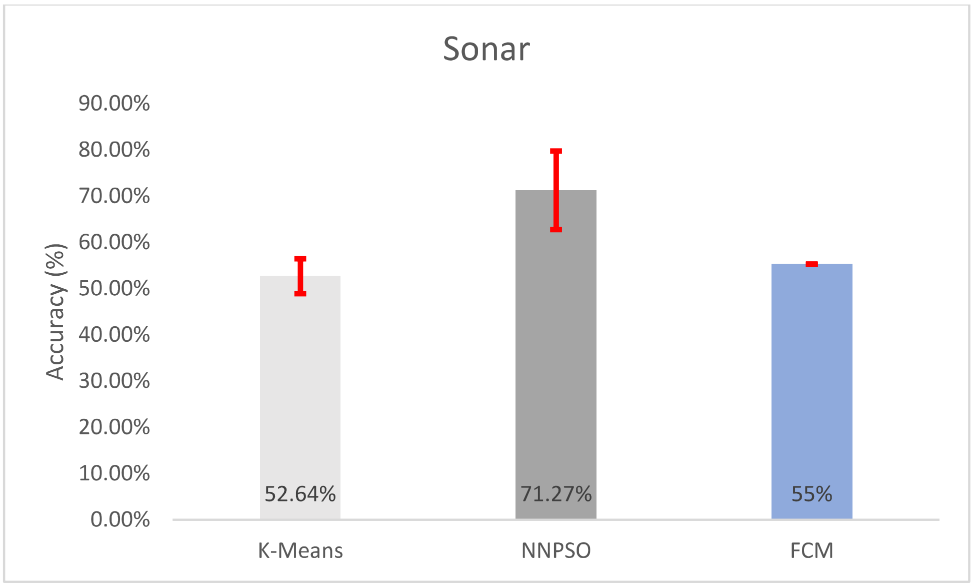

**Figure 40. Accuracy of the Algorithms on Sonar**

The K-Means algorithm was the first algorithm that was implemented for the experiment, and so it was often used as a baseline for comparing the other algorithms. While the performance did not excel in many categories, the results were all within the expected range based on results from other similar experiments and in the literature [1]. One potential problem to note is that the standard deviation for the performance metrics from the Iris results are noticeably higher then results from the other data sets. This was found to be due to the algorithm's tendency to get stuck at local minima [2], which is detailed more in section V. The results from the Fuzzy C-Means algorithm have averages that are more or less similar to the results from K-Means. However, the standard deviation for those values is much lower. Despite randomly initializing the starting centroids, the model seemed to always converge to positions that gave off effectively the same performance metrics. It can be inferred that the 'fuzzy' aspect of the model is contributing to this, allowing for any particular point to belong to each cluster centroid to a certain degree, but when it came time to classify, the highest weighted relationship with a centroid (no matter how small the difference between the various weights) was what determined which cluster the point belonged to. Fuzzy C-Means also did not seem to share the issue for the Iris data set with K-Means. This could imply that the membership matrix also allows better exploration of the data sets in certain conditions.

As for the Neural Network, while it does not share most of the performance metrics with the other two algorithms, accuracy and F-Score, which it does share, seem to have a significant improvement on most of the data sets. The Mammographic Mass and Sonar data sets in particular have a notable improvement over the two clustering algorithms. The most likely factor in this is the difference in how the algorithm determines the classification for the data points. As the Neural Network does not rely on the Euclidean distance to determine which points belong to which category and instead uses the approximation function generated by the network, the overlapping of the cluster areas which caused problems for the clustering algorithms is not apparent. It is also worth noting however that the performance of the Seeds data set was notably worse for the network. As shown in figure 30 the Seeds data set struggles to converge, unlike most of the data sets. Looking at the standard deviations for the accuracies may offer some insight into this discrepancy. The network appears to have more volatility in its predictions then both clustering algorithms, and so it is possible that the setup of the nodes and activation function is adding in some problematic handling of the data that was not apparent in K-Means or Fuzzy C-Means.

## VII. Conclusions and Future Work

The goal of this work was to investigate the strengths and weaknesses of several clustering and classification algorithms to help validate the performance of each of them and to gain a better understanding of where they might be applicable. The simpler and relatively computationally cheap K-Means is well suited for general tasks but tends to fall short on more complex data sets. Fuzzy C-Means is similarly suited for general tasks and has the bonus of being able to better handle ambiguity between clusters using a probabilistic membership matrix. Still, the limitations of utilizing a deterministic cluster update scheme can leave it falling short in data sets with overlapping clusters. Furthermore, both algorithms can end up trapped in local optima creating an inaccurate model. The same problem may be looked at from a classification standpoint where a Feed-Forward Neural Network optimized by PSO performs reliably on all the datasets considered. The sensitivity to initialization parameters implies care must be taken to ensure that the algorithm returns meaningful results for any individual data set. However, there is no guarantee that even the most optimal set of parameters will result in a perfect classification as that would depend on the complexity of the underlying mapping represented by the dataset.

While there is still work that can be done in this area, the experiments done in this paper have given some of the insights that were hoped for. While the clustering data sets performed admirably on some data sets, others were only slightly better than guesswork. This shows that while spatially grouping data can be adequate in some situations, it will only go so far. Furthermore, the improved exploration provided by the “Fuzzy” aspects of

FCM did allow it to perform more consistently and even better in some situations than K-Means. Still, this strength did not allow it to overcome the fundamental weakness in spatially grouped data with overlapping clusters. The FNN-PSO had its strengths as well, relying not on spatial clustering but approximating an equation to represent the data set, meaning overlapping data was not an issue. It did however fall short in the complexity of the implementation, especially with regards to the Seeds data set where even the most optimal parameters that were found did not allow it to properly approximate the data. Also, as with the Breast Cancer data set it did not perform much better than the simpler clustering algorithms despite this extra complexity. Going forward it will be important to keep these things in mind, as knowing how the data set works is just as important as finding an algorithm that works well for the data set.

Future work in this area would include moving beyond benchmarking data sets and into more complex data in order to apply some of the insights gained from these introductory experiments. Furthermore, exploring the functionality of the algorithms on a step-by-step basis could give further insights into the exact strengths and weaknesses of the algorithms and their applicability to a large variety of interesting problems.

Resources:

[1] Sengupta, S., Basak, S., & Peters, R. A. (2018, January). Data clustering using a hybrid of fuzzy c-means and quantum-behaved particle swarm optimization. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 137-142). IEEE.

[2] Ahmadyfard, A., & Modares, H. (2008, August). Combining PSO and k-means to enhance data clustering. In 2008 International Symposium on Telecommunications (pp. 688-691). IEEE.

[3] Sengupta, S., Basak, S., Saikia, P., Paul, S., Tsalavoutis, V., Atiah, F., Ravi, V., Peters, A. (2020). A review of deep learning with special emphasis on architectures, applications and recent trends. Knowledge-Based Systems, 194. doi:10.1016/j.knosys.2020.105596

[4] J. MacQueen (1967). Some methods for classification and analysis of multivariate observations. Proc. Fifth Berkeley Symp. on Math. Statist. and Prob., Vol. 1 (Univ. of Calif. Press, 1967), 281--297.

[5] Van der Merwe, D. W., & Engelbrecht, A. P. (2003, December). Data clustering using particle swarm optimization. In The 2003 Congress on Evolutionary Computation, 2003. CEC'03. (Vol. 1, pp. 215-220). IEEE.

[6] Nayak, J., Naik, B., & Behera, H. (2015). Fuzzy C-means (FCM) clustering algorithm: a decade review from 2000 to 2014. Computational intelligence in data mining-volume 2, 133-149.

[7] Torra, V. (2015, June). On Fuzzy c -Means and Membership Based Clustering. In International Work-Conference on Artificial Neural Networks (pp. 597-607). Springer, Cham.

[8] Zhang, J., & Ma, Z. (2020). Hybrid Fuzzy Clustering Method Based on FCM and Enhanced Logarithmical PSO (ELPSO). Computational intelligence and neuroscience, 2020.

[9] Feng, Y., Lu, H., Xie, W., Yin, H., & Bai, J. (2018). An improved fuzzy C-means clustering algorithm based on multi-chain quantum bee colony optimization. Wireless Personal Communications, 102(2), 1421-1441.

[10] Sengupta, S., Basak, S., & Peters, R. A. (2019). Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives. Machine Learning and Knowledge Extraction, 1(1), 157-191.

[11] Hanif, M., parallelized PSO clustering, (2020), GitHub repository, https://github.com/ms03831/parallelized-PSO-clustering

[12] ShritiK, Fuzzy C Means Clustering, (2020), GitHub repository, https://github.com/ShristiK/Fuzzy-C-Means-Clustering

[13] Ahmad, Z. (2020, May 02). Train Neural Network (Numpy)- Particle Swarm Optimization(PSO). Retrieved from https://medium.com/@zeeshanahmad10809/train-neural-network-numpy-particle-swarm-optimization-pso-93f289fc8a8e

[14] Prateekk94. (2020, January 30). Fuzzy C-Means Clustering on Iris Dataset. Retrieved from https://www.kaggle.com/prateekk94/fuzzy-c-means-clustering-on-iris-dataset

[15] UCI Machine Learning Repository, URL: http://archive.ics.uci.edu/ml/

[16] Scikit-learn Library, URL: https://scikit-learn.org/stable/

[17] K-Means Advantages and Disadvantages | Clustering in Machine Learning. (2021, January 13). Retrieved from https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages

[18] Patel, G. K., Dabhi, V. K., & Prajapati, H. B. (2017). Clustering Using a Combination of Particle Swarm Optimization and K-means. Journal of Intelligent Systems, 26(3), 457-469. doi:10.1515/jisys-2015-0099

[19] Bezdek, J. C. (1981). Modified Objective Function Algorithms. Pattern Recognition with Fuzzy Objective Function Algorithms, 155-201. doi:10.1007/978-1-4757-0450-1_5

[20] Omid Tarkhaneh, Haifeng Shen, Training of feedforward neural networks for data classification using hybrid particle swarm optimization, Mantegna Lévy flight and neighborhood search, Heliyon, Volume 5, Issue 4, 2019, e01275, ISSN 2405-8440, https://doi.org/10.1016/j.heliyon.2019.e01275.

[21] Zhang C., Shao H. (2000) Particle Swarm Optimisation in Feedforward Neural Network. In: Malmgren H., Borga M., Niklasson L. (eds) Artificial Neural Networks in Medicine and Biology.

Perspectives in Neural Computing. Springer, London. https://doi.org/10.1007/978-1-4471-0513-8_50

[22] Karaboga D., Akay B., Ozturk C. (2007) Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks. In: Torra V., Narukawa Y., Yoshida Y. (eds) Modeling Decisions for Artificial Intelligence. MDAI 2007. Lecture Notes in Computer Science, vol 4617. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-73729-2_30

[23] Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995.

[24] Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. J. Glob. Optim. 2007, 39, 459–471.

[25] S. Sengupta, S. Basak and R. A. Peters, "QDDS: A Novel Quantum Swarm Algorithm Inspired by a Double Dirac Delta Potential," 2018 IEEE Symposium Series on Computational Intelligence (SSCI), 2018, pp. 704-711, doi: 10.1109/SSCI.2018.8628792.

[26] Yang, X.S.; Deb, S. Cuckoo Search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.

[27] Colorni, A.; Dorigo, M.; Maniezzo, V. Distributed Optimization by Ant Colonies. In Actes de la Première Conférence Européenne sur la vie Artificielle, Paris, France; Elsevier Publishing: Amsterdam, The Netherlands, 1991; pp. 134–142.

[28] Sengupta, S.; Basak, S.; Peters, R.A., II. Chaotic Quantum Double Delta Swarm Algorithm Using Chebyshev Maps: Theoretical Foundations, Performance Analyses and Convergence Issues. J. Sens. Actuator Netw. 2019, 8, 9. https://doi.org/10.3390/jsan8010009

[29] D.Y. Sha, Hsing-Hung Lin, A multi-objective PSO for job-shop scheduling problems,
Expert Systems with Applications, Volume 37, Issue 2, 2010, Pages 1065-1070,
ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2009.06.041.

[30] C. Chen and K. Zhou, "Application of Artificial Bee Colony Algorithm in Vehicle Routing Problem with Time Windows," 2018 International Conference on Sensing,Diagnostics, Prognostics, and Control (SDPC), 2018, pp. 781-785, doi: 10.1109/SDPC.2018.8664999.

[31] Marinakis Y., Marinaki M., Migdalas A. (2018) Particle Swarm Optimization for the Vehicle Routing Problem: A Survey and a Comparative Analysis. In: Martí R., Pardalos P., Resende M. (eds) Handbook of Heuristics. Springer, Cham. https://doi.org/10.1007/978-3-319-07124-4_42

[32] S. Sengupta and S. Basak, "Computationally efficient low-pass FIR filter design using Cuckoo Search with adaptive Levy step size," 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), 2016, pp. 324-329, doi: 10.1109/ICGTSPICC.2016.7955321.

[33] S. Dhabal and S. Sengupta, "Efficient design of high pass FIR filter using quantum-behaved particle swarm optimization with weighted mean best position," Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT), 2015, pp. 1-6, doi: 10.1109/C3IT.2015.7060145.

[34] Hanhong Zhu, Yi Wang, Kesheng Wang, Yun Chen, Particle Swarm Optimization (PSO) for the constrained portfolio optimization problem, Expert Systems with Applications, Volume 38, Issue 8, 2011,

Pages 10161-10169, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2011.02.075.

[35] Divya Kumar, K.K. Mishra, Portfolio optimization using novel co-variance guided Artificial Bee Colony algorithm, Swarm and Evolutionary Computation, Volume 33, 2017, Pages 119-130, ISSN 2210-6502, https://doi.org/10.1016/j.swevo.2016.11.003.

[36] Basak S., Sun F., Sengupta S., Dubey A. (2019) Data-Driven Optimization of Public Transit Schedule. In: Madria S., Fournier-Viger P., Chaudhary S., Reddy P. (eds) Big Data Analytics. BDA 2019. Lecture Notes in Computer Science, vol 11932. Springer, Cham. https://doi.org/10.1007/978-3-030-37188-3_16

[37] Santosh Kumar Majhi, Shubhra Biswal, Optimal cluster analysis using hybrid K-Means and Ant Lion Optimizer, Karbala International Journal of Modern Science, Volume 4, Issue 4, 2018, Pages 347-360, ISSN 2405-609X, https://doi.org/10.1016/j.kijoms.2018.09.001.