DATA MINING

Analyzing datasets using Weka tool

By:

SHEETAL MORE

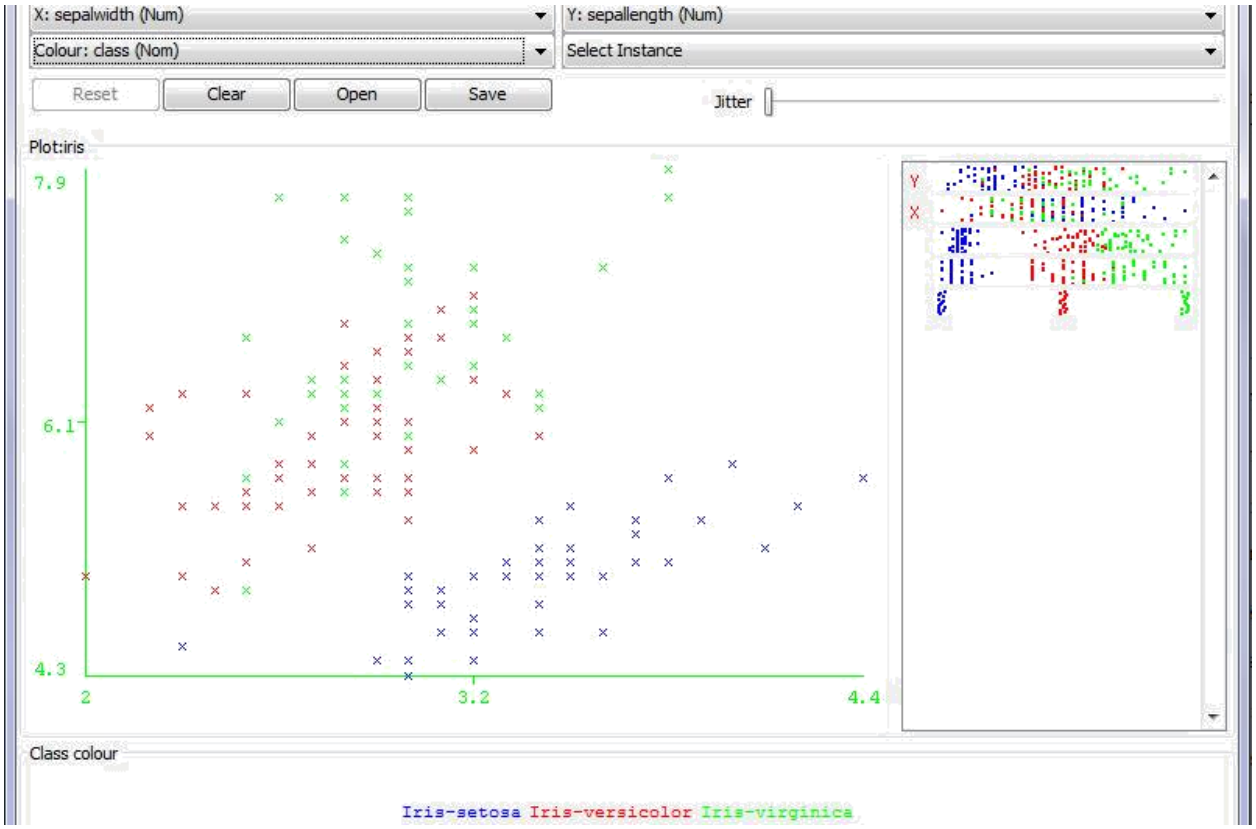
Smore2@hawk.iit.edu

PART II: WEKA EXPERIMENTS

IRIS DATASET

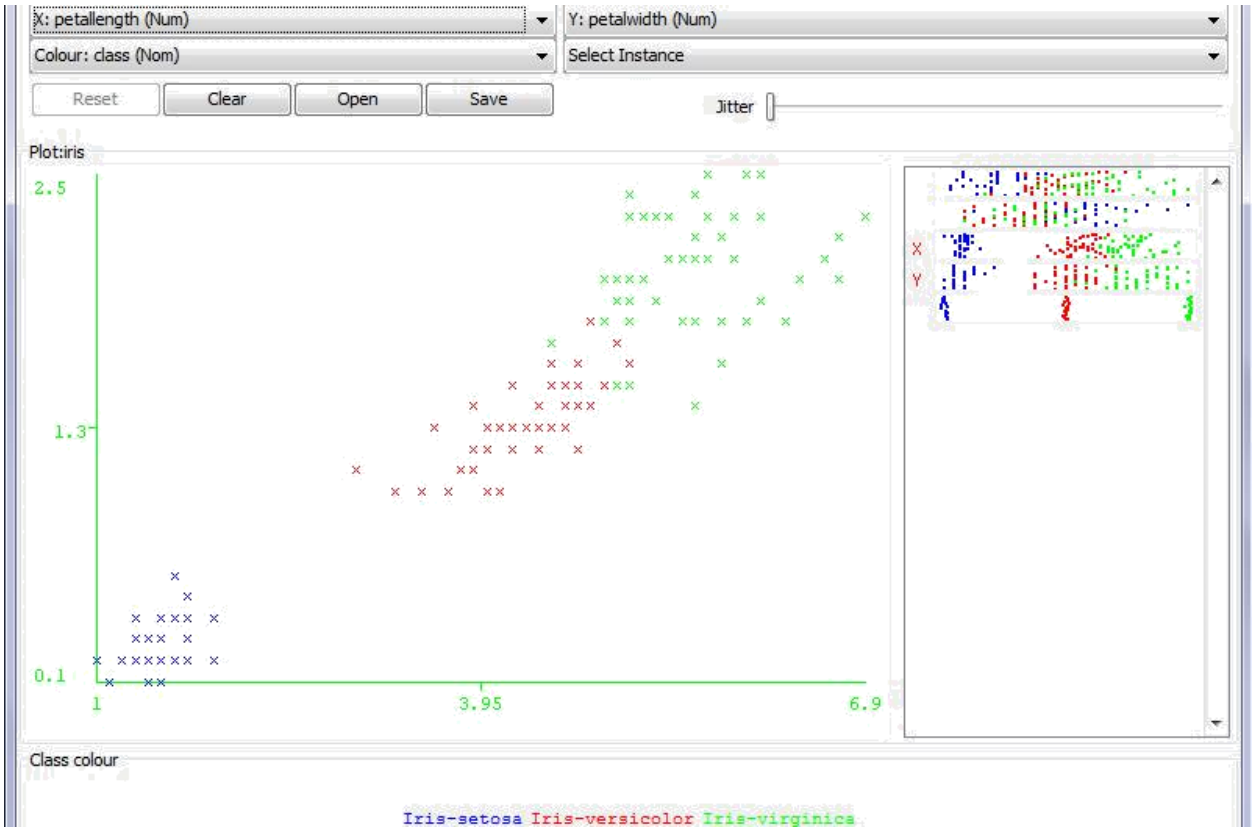
Q.1. Visualize the attributes:

Below is the plot between sepal length and sepal width.

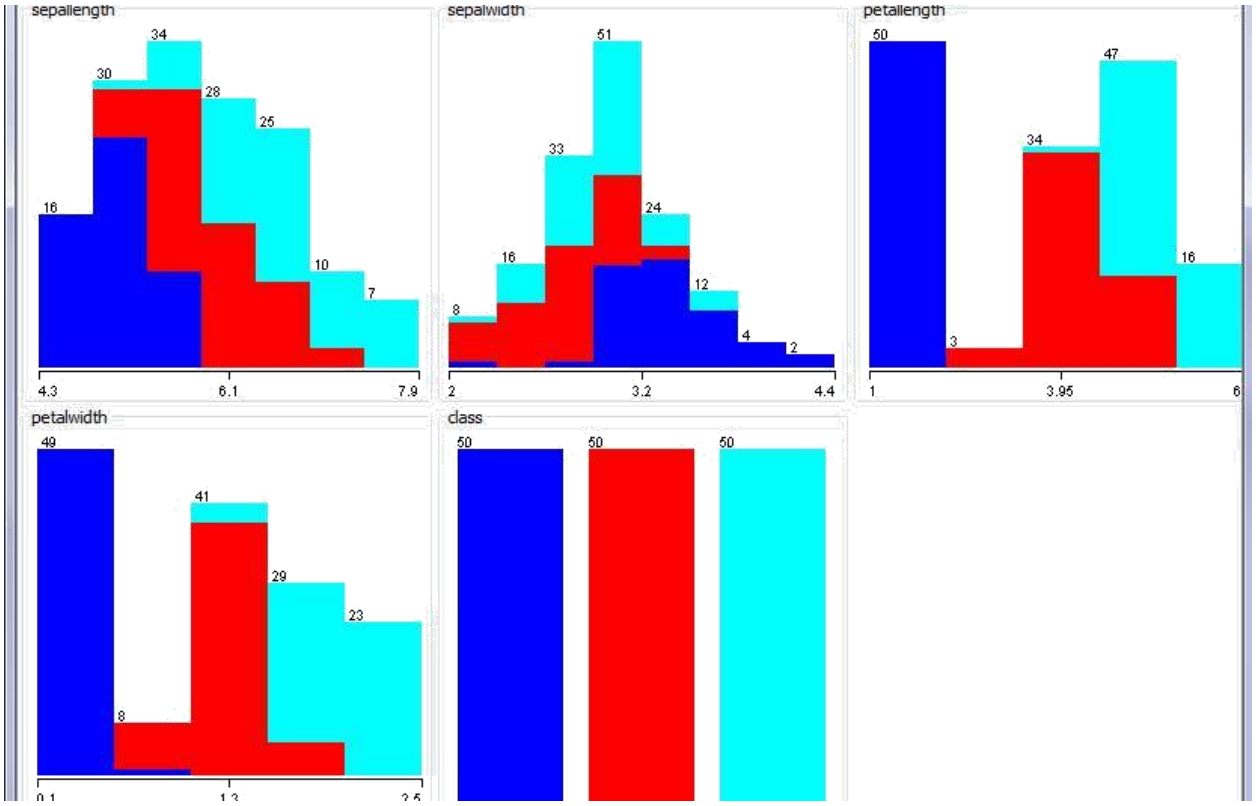


As seen in the above graph, the points seen are scattered all over i.e. the clusters are not formed. As a result, such attributes are not really useful in clustering algorithms.

Below is the graph of petal length and petal width.



As seen from above graph, the three types of flowers are seen as 3 separate clusters. Thus it is useful for clustering.



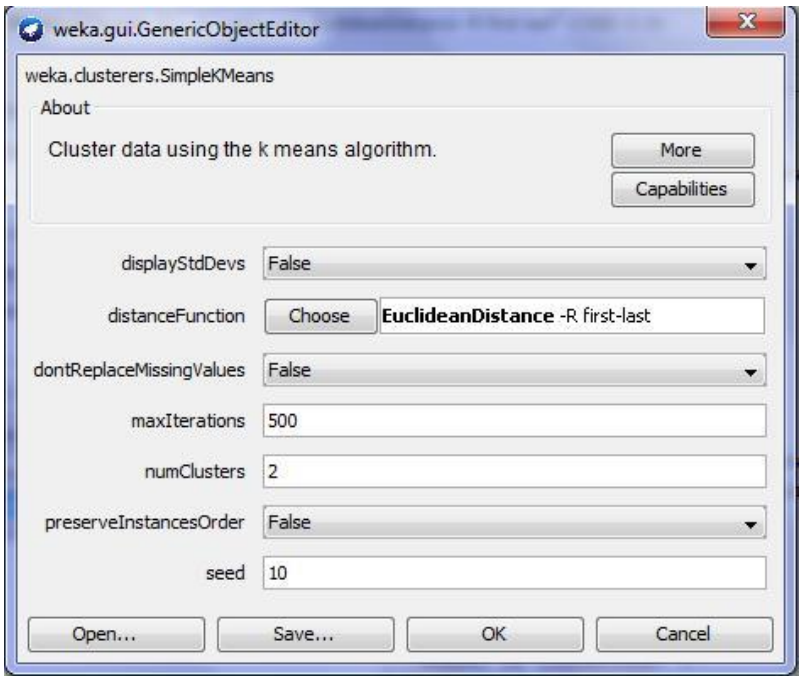
Analysis:

- On visualizing the data as shown in above graphs, I can analyse that petal length and petal width can be considered as good attributes to be used for clustering.
- Also, as seen from the bar graphs above, there is overlapping of attributes between the two different types of flowers.
- Since there is overlapping, it might become difficult in assigning the points to a specific cluster.
- sepal length and sepal width on the other hand do not form clusters and have points which would make it difficult to form a specific cluster in lesser iterations.

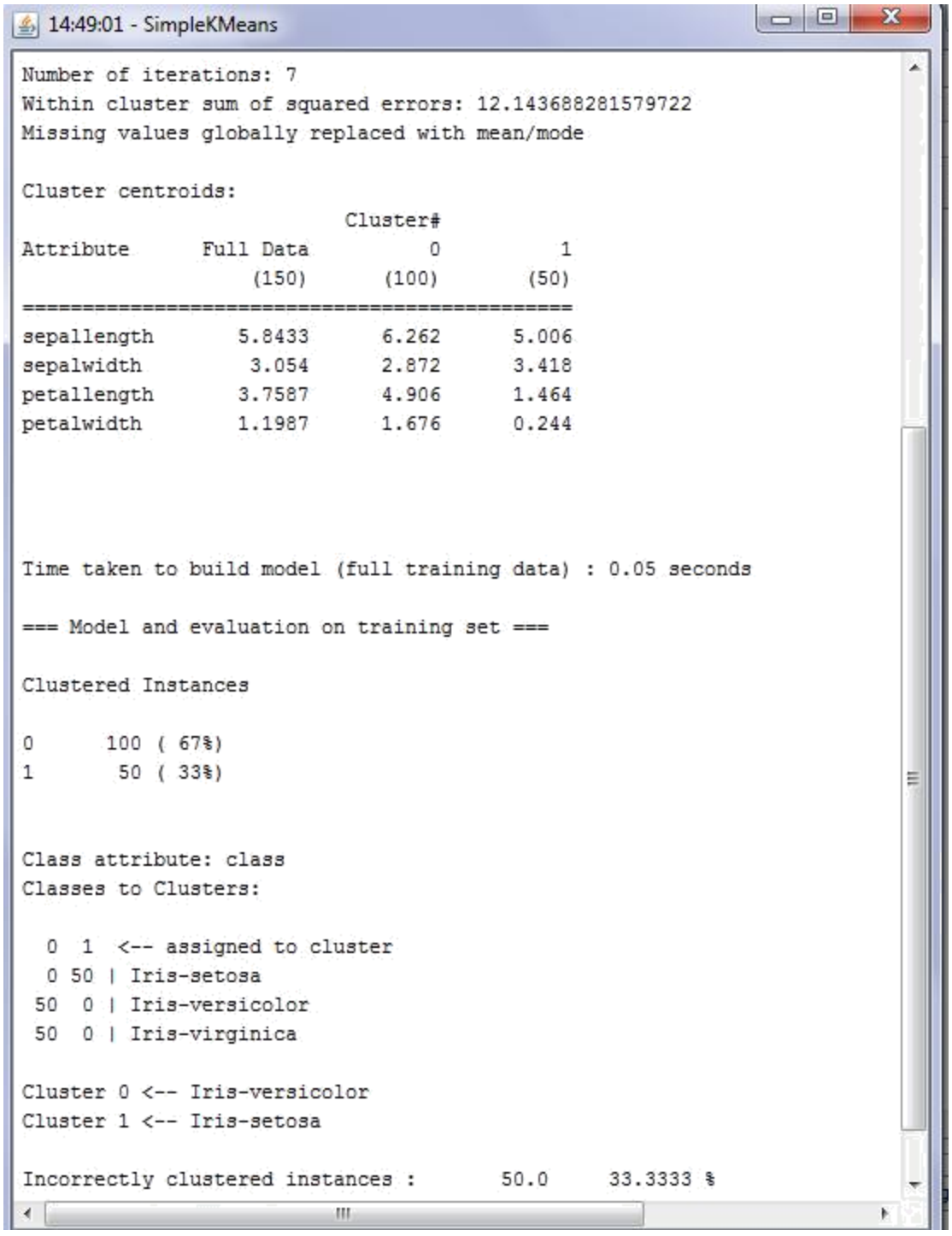
Q.2. Clustering using default values:

(1) SimpleKMeans Algorithm:

The default parameters for this algorithm are as follows:



- **distanceFunction :** This shows the distance measure used to calculate the distance between the two instances. The default value is Euclidean distance. This parameter is not changed for clustering purposes.
- **maxIterations:** This is the maximum number of iterations that K means algorithm will run to find an optimal cluster. If the algorithm finds an optimal clusters before reaching maxIterations then it stops else it continues till the maxIterations are reached. Default value is 500. This value is changed for seeing the effect on clustering.
- **numClusters:** This specifies the number of clusters into which the dataset has to be divided. In K means algorithm K indicates number of clusters that the user wants. This parameter is also changed to observe the effect on clustering.
- **preserveInstanceOrder**: The algorithm for random centroid selection changes the order of instances in the training data. Setting to parameter to true we can preserve the order of instance of the training data. The default value is false. This parameter is not changed for clustering.
- **Seed:** It is a value that is used to initialize a random number generator.
- **displayStdDevs:** This option when set to true, displays std deviations of numeric attributes and counts of nominal attributes.



Above are the results for clustering using default values for SimpleKMeans algorithm.

The performance evaluation of clustering algorithm can be done using two parameters as follows:

- sum of squared errors
- number of incorrectly classified instances

1. The result states that there are 50 incorrectly classified instances i.e. 33.33% of instances. There are 2 clusters formed, thus 2 of the classes are correctly classified and the third class is incorrectly classified for all of its instances.
2. Also, the SSE within the cluster is 12.14%.

Analysis of measures in output tab:

- **Sum of squared error:** This measure is the sum of squares of Euclidean distance of each data point to its closest centroid. Thus if the value of SSE is low then we can say that good clustering is done.
- **Incorrectly classified values:** This measure gives the instances those were not classified for its respective class. As seen in the above results, all the instances of class Iris-Virginica are clustered in the clusters of Iris-Setosa and Iris-Veronica. This incorrectly instances have occurred since there are 3 classes and we have given only 2 clusters as default value.
- **Classes assigned to clusters:** Using the default values, 2 clusters are formed as follows:

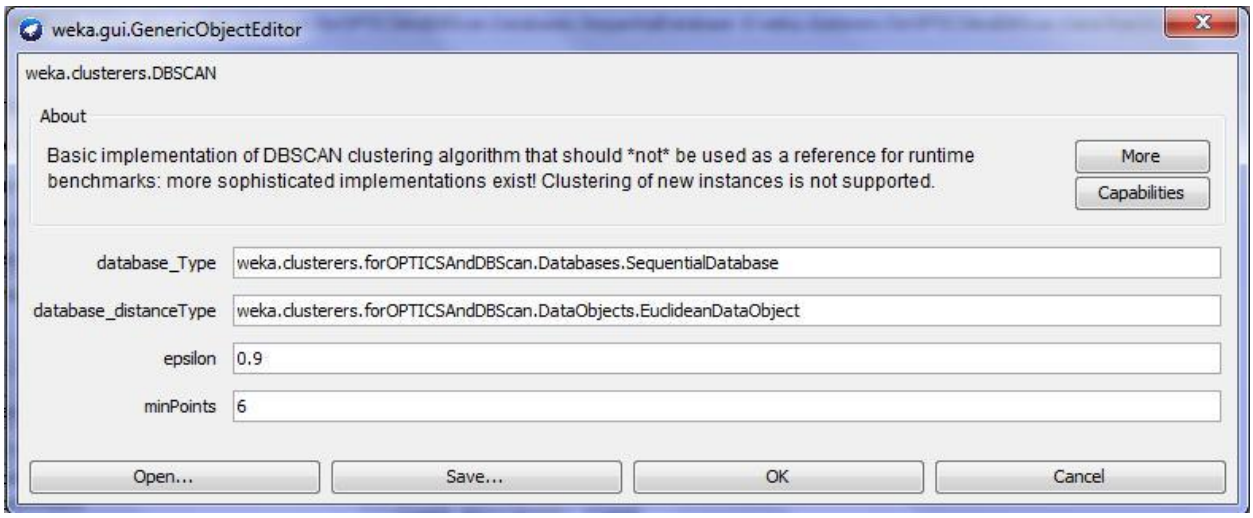
  ```
   0   1  <-- assigned to cluster
   0 50 | Iris-setosa
  50   0 | Iris-versicolor
  50   0 | Iris-virginica
  ```

  Cluster 0 <-- Iris-versicolor
  Cluster 1 <-- Iris-setosa

- **Classes to cluster evaluation:** This parameter defines the class which is to be assigned to a given cluster. Here the 2 cluster 0 and 1 are given the class labels Iris-versicolor and Iris-Setosa. In class based evaluation, initially the clusters are formed and then the class labels are assigned based on the majority of instances belonging to that class.

(2) <u>DBSCAN:</u>

The default parameters for DBSCAN algorithm are as follows:



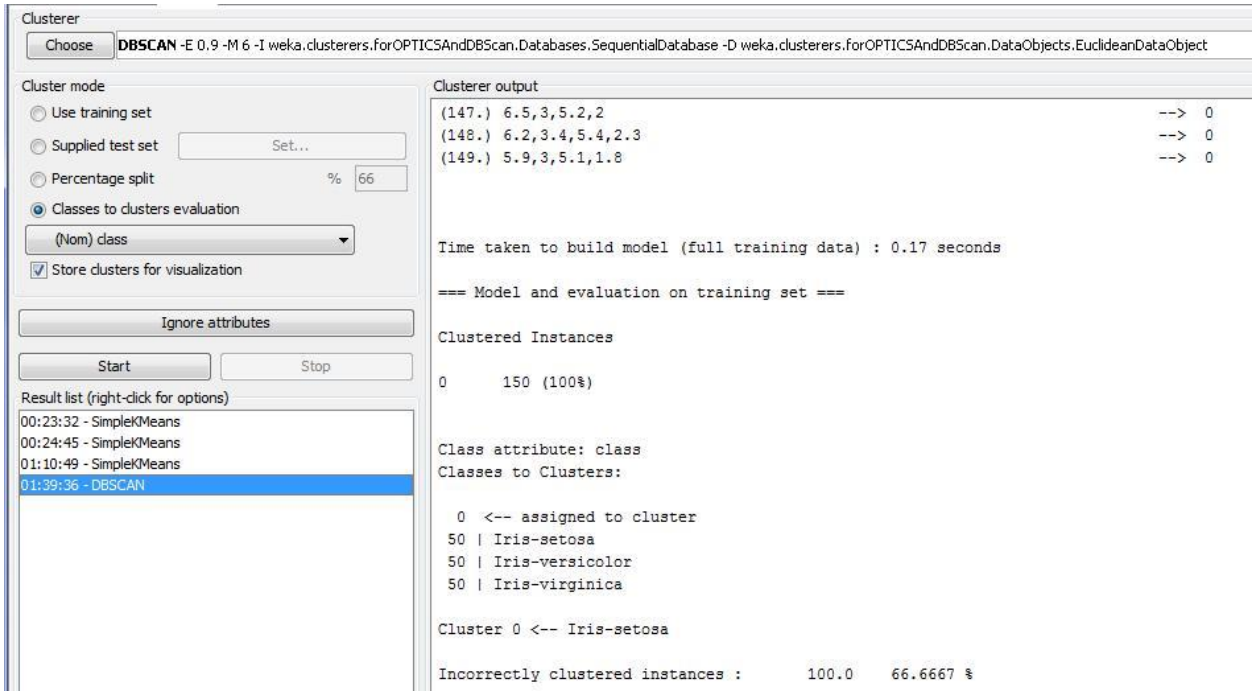
database_Type -- *used database*. This parameter is kept as defaulted since it is not needed for clustering purpose.

database_distanceType -- *used distance-type.* This parameter is kept as given and not changed as it has no role in clustering

epsilon -- *radius of the epsilon-range-queries.* This parameter gives the radius of the cluster formed which can thus be changed in order to see the effect on clustering. The effect of changing this parameter would result in the increase or decrease in the number of points included in the cluster. For a given value of epsilon, the number of points inside the radius are calculated and used for forming clusters

minPoints -- *minimum number of DataObjects required in an epsilon-range-query.* This parameter gives the minimum points needed to be present in a particular cluster. This can significantly have effect on the number of clusters that can be formed.

OUTPUT:



Observations and analysis of measures in output tab:

- The number of clusters formed is 1. Depending on the number of minimum points and the epsilon value, the number of clusters generated varies.
- Incorrectly classified instances are 100 which is 66.67% of the total number of instances.
- As seen from the confusion matrix, all the three classes are assigned to single cluster which is labelled as Cluster 0 for Iris-Setosa. Thus only the instances belonging to Iris-Setosa are correctly classified and remaining instances belonging to Iris-virginica and Iris-versicolor are not assigned a separate cluster.
- Clustered instances: Here the clustered instances are 100% i.e. all the instances are clustered into one cluster labelled as Iris-Setosa. All the instances are clustered and there are zero noise points.

Q.3. Removing attributes:

(a) Removing sepal length and sepal width

As analysed in part 1 where I visualized the graphs by plotting them using the instances given in the dataset, I observed that the graph of petal length and petal width looked useful for clustering since the three classes formed three well separated clusters.

Thus only these attributes (petal length and petal width) were used for clustering and other attributes were removed. Following is the output achieved:

```
=== Run information ===

Scheme:weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Relation:     iris-weka.filters.unsupervised.attribute.Remove-R1-2
Instances:    150
Attributes:   3
              petallength
              petalwidth
Ignored:
              class
Test mode:Classes to clusters evaluation on training data
=== Model and evaluation on training set ===


kMeans
======

Number of iterations: 6
Within cluster sum of squared errors: 5.179687509974783
Missing values globally replaced with mean/mode

Cluster centroids:
                         Cluster#
Attribute      Full Data         0          1
                   (150)      (100)       (50)
================================================
petallength       3.7587      4.906      1.464
petalwidth        1.1987      1.676      0.244
```

```
=== Run information ===

Scheme:weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Relation:     iris-weka.filters.unsupervised.attribute.Remove-R1-2
Instances:    150
Attributes:   3
              petallength
              petalwidth
Ignored:
              class
Test mode:Classes to clusters evaluation on training data
=== Model and evaluation on training set ===


kMeans
======

Number of iterations: 6
Within cluster sum of squared errors: 5.179687509974783
Missing values globally replaced with mean/mode

Cluster centroids:
                         Cluster#
Attribute      Full Data          0          1
                   (150)      (100)       (50)
=============================================
petallength       3.7587      4.906      1.464
petalwidth        1.1987      1.676      0.244
```

Observations and analysis:

- The SSE (sum of squared error) is observed to be 5.179 which is decreased
- the number of incorrectly classified instances are 50 i.e. 33.33% of total instances.
- number of iterations are reduced i.e. 6 instead of 7.
- Thus petal length and petal width are a good attributes which improve the quality of clustering

(b) Removing the petal length and petal width

After removing the petal length and petal width we can observe that the error has been increased as compared to previous one. The reason is the attributes that were removed had significant impact on clustering.

```
======

Number of iterations: 4
Within cluster sum of squared errors: 7.081076555902943
Missing values globally replaced with mean/mode

Cluster centroids:
                           Cluster#
Attribute      Full Data          0          1
                   (150)       (84)       (66)
=============================================
sepallength       5.8433     5.2333     6.6197
sepalwidth         3.054     3.1286     2.9591



Time taken to build model (full training data) : 0.03 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       84 ( 56%)
1       66 ( 44%)


Class attribute: class
Classes to Clusters:

  0  1  <-- assigned to cluster
 50  0 | Iris-setosa
 27 23 | Iris-versicolor
  7 43 | Iris-virginica

Cluster 0 <-- Iris-setosa
Cluster 1 <-- Iris-virginica

Incorrectly clustered instances :       57.0     38      %
```

As observed above, the number of incorrectly clustered instances increased a lot and also the error increased. Thus the attributes petal length and petal width should not be removed.

Q.4. EXPERIMENTATION BY CHANGING PARAMETERS

(1) SimpleKMeans Algorithm

	Number of clusters	Number of iterations	Maximum iterations	SSE	Incorrectly classified instances	Observation
1	5	5	500	0.857	51	In this case, the number of incorrectly classified instances was 51 which is not desirable. However the error is minimal.
2	1	1	1	27.35	50	Here, the decrease in the number od clusters and number of iterations caused the error rate to go high and the number of incorrectly clustered instances remained nearly the same.
3	2	2	2	7.5	50	A slight increase in the number of cluster has decreased the sum of squared error
4	3	2	2	4.74	8	Here the number of clusters were increased to 3 and thus the error was reduced and also the incorrectly classified instances were significantly reduced to 8
5	5	2	2	0.98	50	Here the number of clustered were increased which reduced the error however incorrectly clustered were high.
6	3	6	500	1.7	6	Increasing the number of iterations increased
7	3	3	3	1.97	5	Increasing the number of time will decrease the number of clusters as well as number of incorrectly classified instances.

Conclusion:

The results are best when the number of clusters are k=3 and maximum number of instances are 3. The SSE then are 1.97 and incorrectly classified instances are 5.

(2) DBSCAN ALGORITHM

	epsilon	minPoints	No of clusters	Noise	Incorrectly clustered	Observations
1	0.1	6	3	88	6	Default values gives less number of incorrectly classified however the noise is more
2	0.2	6	2	3	48	Increasing the epsilon value increases the incorrectly clustered
3	0.9	20	1	0	100	Increasing the number of minPoints gave zero noise points however the incorrectly classified instances increased.

4	0.4	6	2	0	50	Decreasing the epsilon value to 0.6 gave the zero noise values however the number of incorrectly classified instances are large

Analysis:

- The best result is achieved when the epsilon value is 0.4 and minPoints = 6 where the incorrectly clustered instances are 50.
- For epsilon value of 0.1, we get less number of incorrectly classified instances, however the noise value is 88 i.e. these many number of points are not at all clustered.
- for epsilon value of 0.9 and 0.4, there are zero noise points i.e. all the instances are considered. However here the number of incorrectly clustered instances are more.

Comparison of SimpleKMeans and DBSCAN algorithm:

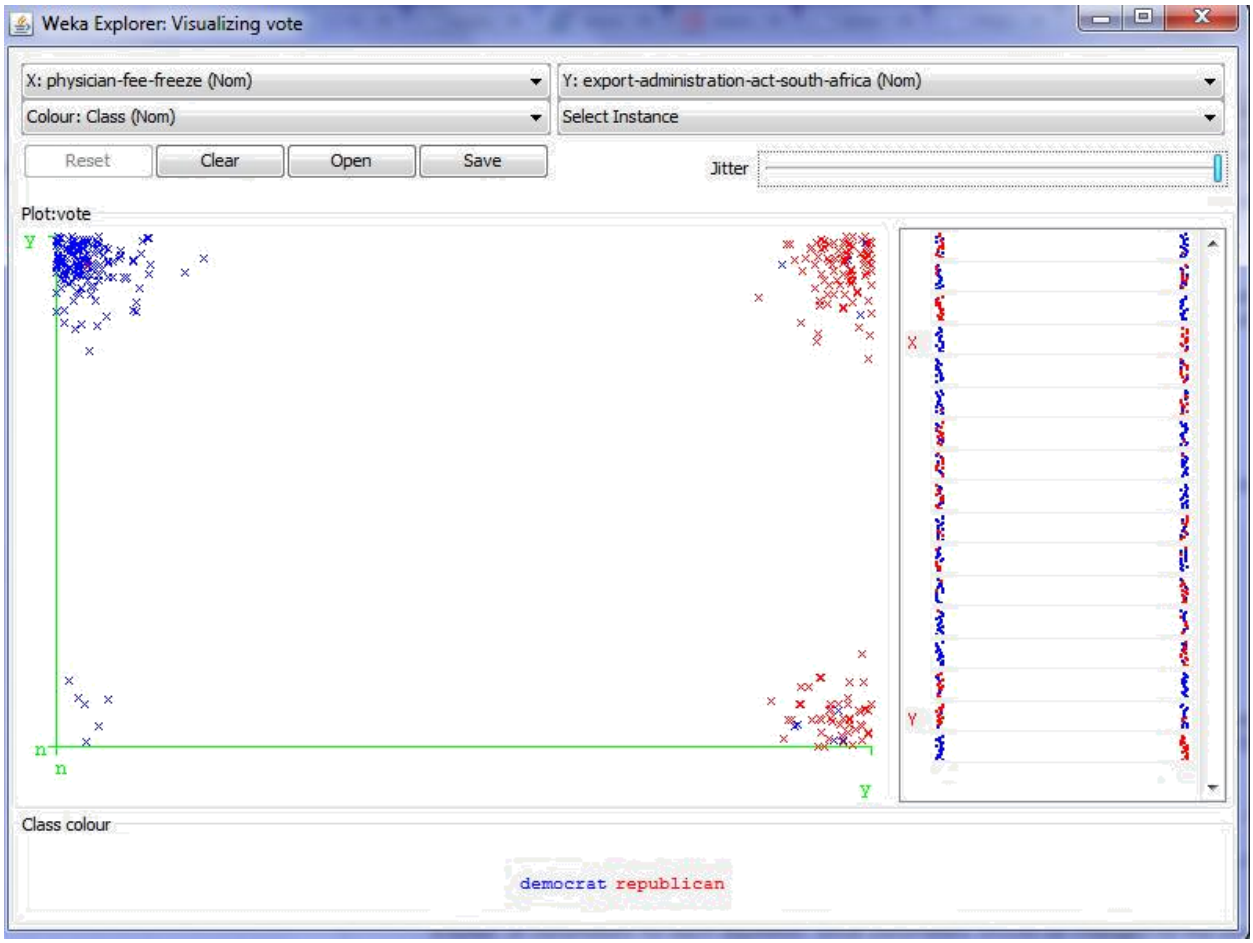
In Iris dataset, the SimpleKMeans algorithm works well when the number of clusters is equal to the number of classes.

The DBSCAN algorithm however does not give desirable results when clustered.

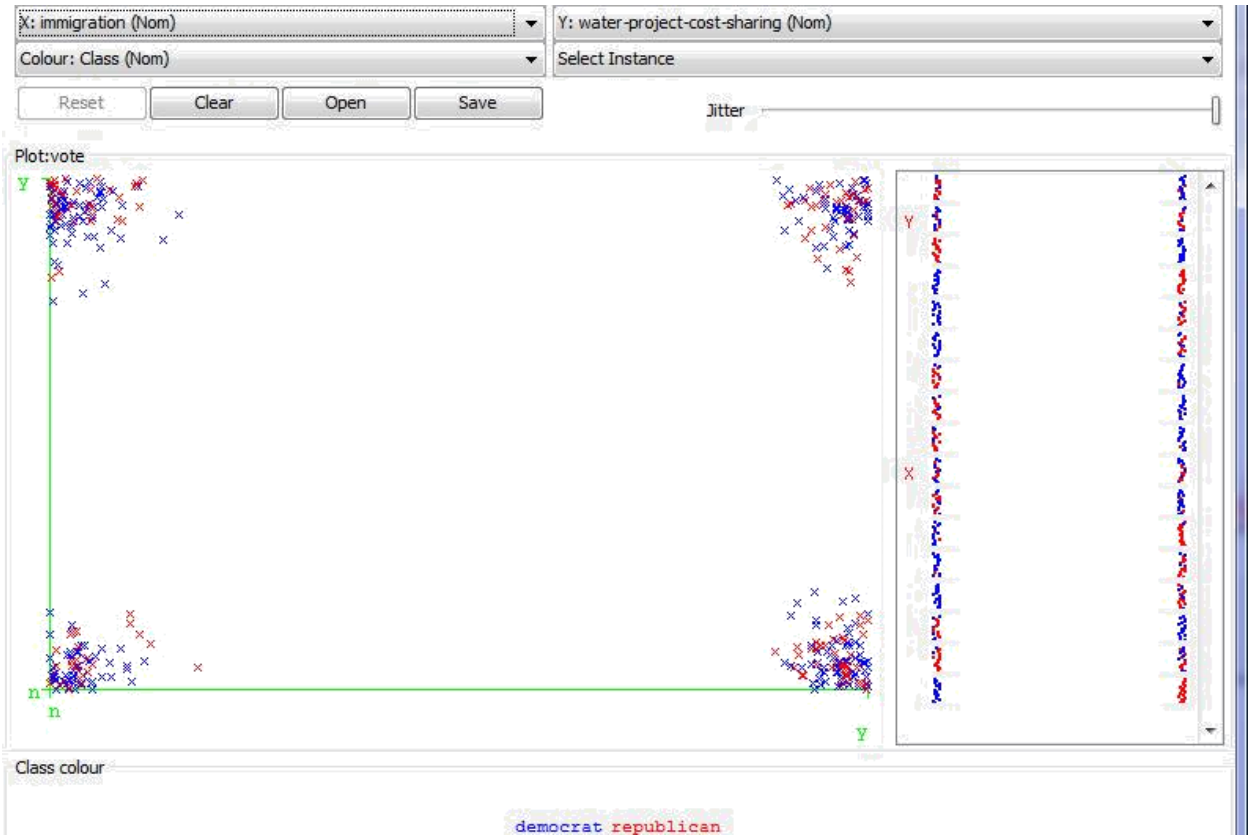
VOTE DATASET

Q.1.Visualizing data:

Below is the graph of visualizing the data points when plotted attributes on X and Y axes.



As seen in above graph of physician-fee-freeze versus export-administration-act-south-africa, the points are distributed in well separated clusters. Thus this attribute is good for clustering.



Also as seen in above graph, the points are mixed in clusters, thus we can say that these attributes are not ideal for clustering.

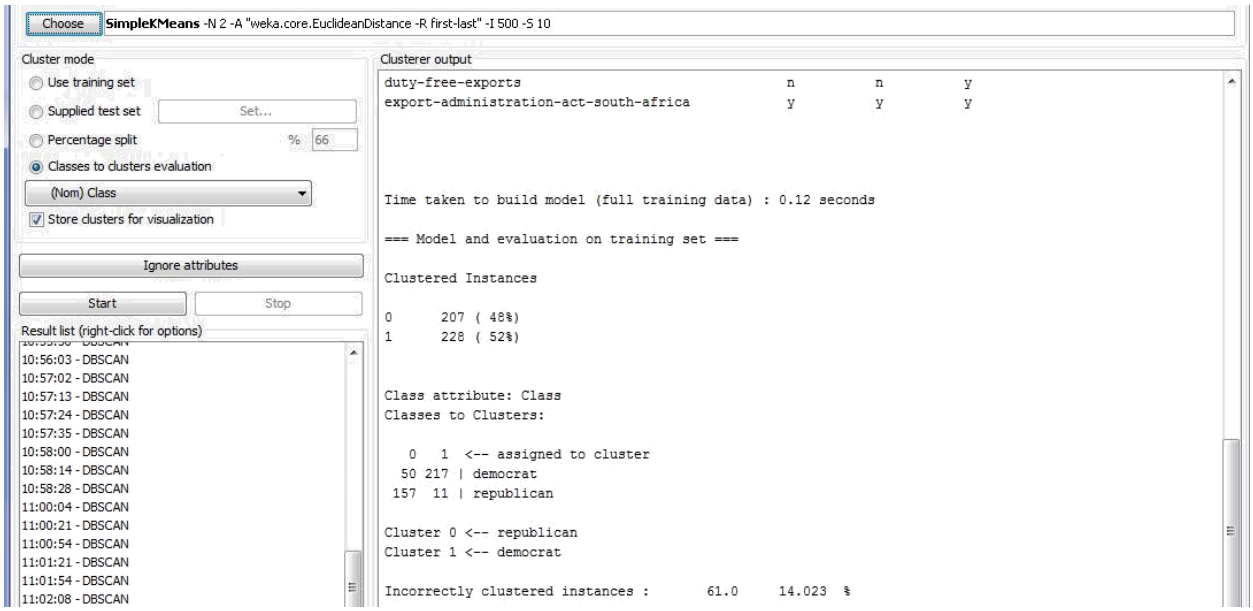
Analysis of visualization:

The attribute physician-fee-freeze is a good attribute for clustering.

Q.2. Clustering with default Values :

(1) SimpleKMeans Algorithm:

The result output on using the default parameters.

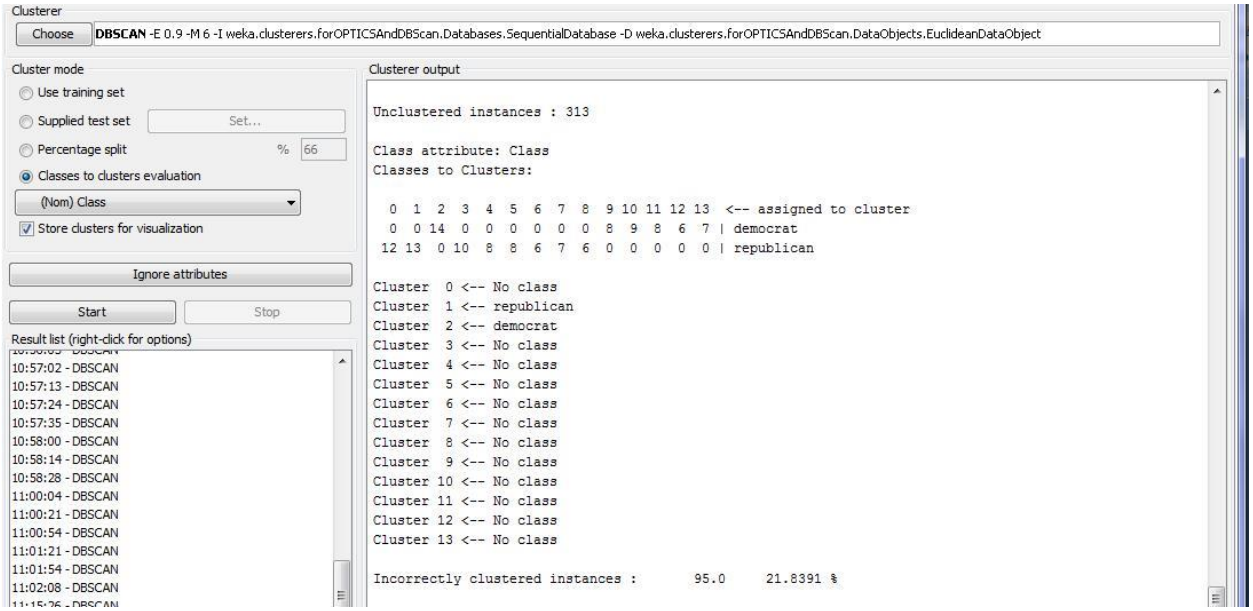


Observation and analysis:

- The number of incorrectly classified instances are 61
- sum of squared error is 1449
- Number of iterations taken to form the clusters are 3 i.e. after 3 iterations the centroids do not change
- The 2 clusters formed are labelled as republic and democratic.
- 50 instances of republic and 11 instances of democrat are incorrectly classified.

(2) DBSCAN Algorithm:

The default parameters run gives the following output:



Analysis:

- The number of incorrectly classified are 95
- The significant observation here is the number of unclustered instances (noise points) which are 313 out of total 435 instances.
- Number of clusters generated are 14 whereas we have only 2 classes. These many clusters were not needed.
- Since the number of unclustered data is too large. Thus this algorithm does not work well on vote dataset.

Q.3. Removing attributes:

Here, all other attributes are removed and the attribute physician-fee-freeze is only used for clustering. Below is the output observed:


```
kMeans
======

Number of iterations: 2
Within cluster sum of squared errors: 0.0
Missing values globally replaced with mean/mode

Cluster centroids:
                                 Cluster#
Attribute               Full Data          0          1
                            (435)      (177)      (258)
=========================================================
physician-fee-freeze            n          y          n



Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      177 ( 41%)
1      258 ( 59%)


Class attribute: Class
Classes to Clusters:

   0   1  <-- assigned to cluster
  14 253 | democrat
 163   5 | republican

Cluster 0 <-- republican
Cluster 1 <-- democrat

Incorrectly clustered instances :       19.0      4.3678 %
```

Analysis:

- The number of incorrectly classified instances are **significantly reduced** since the only attribute used is well distributed in the graph as visualized in question 1.
- **sum of squared errors is completely removed and is zero.**
- number of clusters given are 2 which is ideal since there are two class labels yes and no
- number of iterations are 2

Q.4. EXPERIMENTATION BY CHANGING PARAMETERS

(1) SimpleKMeans Algorithm:

	Number of clusters	Number of iterations	Maximum iterations	SSE	Incorrectly classified instances	Observation
1	2	3	500	1449	61	The sum of squared error here is very large i.e. 1449. However incorrectly classified instances i.e 61 out of 435.
2	2	2	2	1449	61	here decreasing the number of iterations would not cause any difference in the results
3	5	2	2	1177	147	increasing the cluster value decreases the sum of squared error; however the incorrectly classified instances increases.
4	10	2	2	971	227	further increasing the cluster values decreases the sum of squared error and also increases the incorrectly classified instances
5	50	2	2	541	358	increasing the cluster value to 50 drastically decreases the error value but the incorrect instances are large and it is not desirable since we are more interested in having maximum correct instances
6	4	500	3	1225	142	here inspite of increasing the number of iterations to 500 value, it does not change the results

Conclusions:

- The second analysis gives the best result where the number of clusters are 2 and number of iterations are 2. The number of incorrectly classified instances are 61 which are least as compared to others.
- As seen in first and second analysis, there are same results observed. So out of these 2, the second one is better since it has less number of iterations thus decreasing the computations.
- Observing the results in analysis 4 and 5, we can see that the sum of squared error is decreasing, however the number of incorrectly classified instances are increasing and it is not a good clustering. The reason for decreasing the sum of squared error is because we are restricting the k means algorithm to stop when the maximum number of iterations are achieved.

(2) DBSCAN Algorithm:

	epsilon	minPoints	No of clusters	Noise	Incorrectly clustered	Observations
1	0.9	6	14	313	95	The default parameters gives a lot of unclustered (noise) instances and the incorrectly clustered are also high 95. The noise is large as compared to total number of instances (435)
2	1.5	4	1	30	159	here the epsilon value is increased and minPoints value is decreased, this would result in only one cluster being formed and noise is also less. However the incorrectly clustered instances are 159 which is not a good clustering
3	2	90	2	66	32	significantly increasing the minPoints would increase the noise however decreasing the incorrectly clustered instances
4	2.1	90	2	3	51	Here the slight increase in epsilon value drastically decreases the unclustered instances and the number of points are also increased very slightly. The unclustered instances are 3 and the number of incorrectly instances are 51 which are very good as compared with previous values.

Conclusions:

- In above analysis, the best result in observed in 4th case where the epsilon value of 2.1 and minPoints are 90 which would give 3 unclustered instances and 51 incorrectly clustered instances.
- For a smaller values of epsilon and minPoints, lots of instances are considered noise.

Comparison between SimplekMeans and DBSCAN:

- SimpleKMeans algorithm gives a bad result since on vote dataset since the number of incorrectly clustered instances are 61 which form the 14.023 % of total number of instances.
- DBSCAN algorithm on the other hand gives 52 incorrectly classified instances which form 11.7% of the total number of instances.
- Thus we can say that DBSCAN algorithm performed well on the vote dataset.

SUMMARY AND REPORT

Conclusions:

- SimpleKMeans algorithm gives a better clustering results when the number of clusters is equal to the number of classes.
- SimpleKMeans algorithm works good on numerical values since it makes use of the mean values.
- DBSCAN algorithm on the other hand uses the density values when clustering
- DBSCAN also considers few of the instances as noise and thus do not use them in clustering. Having said this, this algorithm might work well when there are many noise instances in the dataset.

- I learnt from the above results and analysis that SSE cannot be considered as a good measure for calculating the performance evaluation since in spite of having low SSE value we observed that there are large incorrectly clustered instances which is a good clustering.

Comparison of the SimpleKMeans and DBSCAN algorithm on different datasets:

SimpleKMeans	DBSCAN Algorithm
In this algorithm, the initial centroids are selected and then the points nearer to this centroid are given to their cluster	In this algorithm, initially the points are divided as core points, border points and noise points; then the epsilon and minPoints are used to calculate the clusters
This algorithm uses the centroid values for clustering	It uses epsilon and minimum number of points to be present inside a radius from the core point.
Gives best results when the number of clusters are same as the number of classes	This gives uses density thus does not depend only on the numerical values
Here, we need to explicitly give the number of clusters to be formed using the given number of iterations	In this, we do not need to mention the number of clusters. Instead the number of clusters generated varies depending on the other parameters.
This algorithm tries to cluster all the instances in the dataset which is not idea since if there are noise points then they are wrongly clustered in one of the clusters	This algorithm on the other hand makes sure that the noise points are not used for clustering thus avoiding incorrect clustering

- In terms of data analysis, I learnt that while performing data analysis we need to consider the importance of each parameter and its effect on the analysis.
- I learnt that the error and accuracy might not be the only measures for performance evaluation. There can be multiple ways of finding a good performance depending on the requirement of what we are trying to analyze.
- Before reaching to a conclusion, it is necessary to perform multiple experiments and have good proof of our analysis.
- The data used for analysis should also be considered before analyzing and if pre processing is needed then it must be done.