HOMEWORK 5

EXERCISES AND EXPERIMENTS ON CLUSTER ANALYSIS

Submitted by

Arun Rajagopalan

A20360689

For CS422 Data Mining

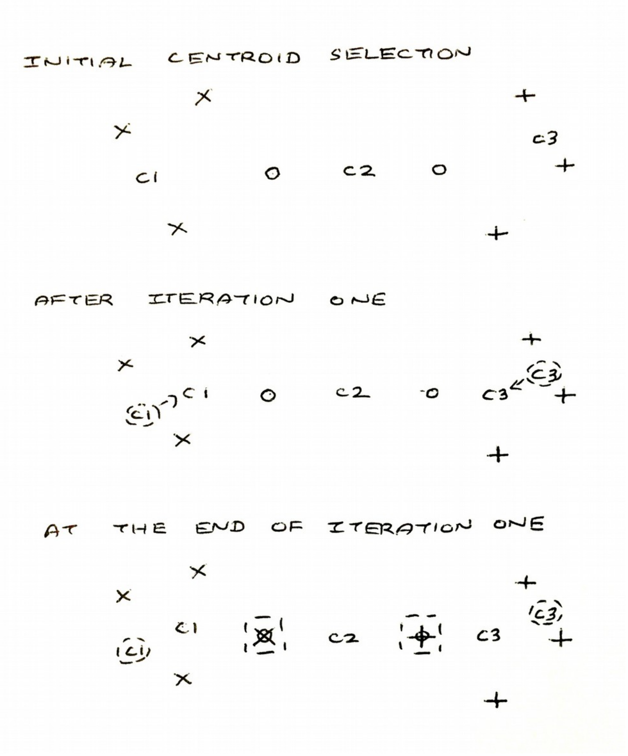
Illinois Institute of Technology

EXERCISES AND EXPERIMENTS ON ASSOCIATION ANALYSIS

Exercise 1:

1) Find an example of a small set of points and three initial centroids so that kMeans with k=3 converges to a clustering with an empty cluster. Note that the initial centroids do not have to be members of the set of points.

Consider the simple example presented below. In case of KMeans algorithm, the initial centroids can be chosen randomly and for the below example we assume so. C1, C2 and C3 are the initial centroids fixed randomly and based on the distances from these three centroids, the taken set of 8 points are clustered into 3 groups as x, o and + in iteration 1. After this assignment, the centroid of the groups just formed is calculated. Thus C1 shifts nearer to the one of the points of o cluster and C3 also moves nearer to the other point of o cluster. C2 remains unchanged.



Now, we re-group based on the newer centroids to get lower SSEs. Thus, one of the points in o cluster switches to the x cluster and the other point in the o cluster jumps to the + cluster. The centroid C2 is left without any points and thus an empty cluster is formed as a result of KMeans converging.

Exercise 2:

2) We use SSE(RSS) as the measure of cluster quality and kMeans minimizes it. If there is an empty cluster, can that clustering be the global minimum solution based on RSS? Show all details of you arguments. Use your own words.

No, the KMeans algorithm converges only to a local minimum, not a global minimum.

SSE or RSS is derived by computing and summing up the squared distances of all data points in a data set to their respective centroids and the main aim of KMeans algorithm to minimize this RSS at every step. Also it is known that KMeans, in its simplest form, randomly chooses initial centroids.

Considering the case of a dataset that has too much noise, where we most likely choose a noise point as a centroid, we will notice in the forthcoming iterations that no other point will be assigned to be grouped with this noisy centroid. Hence at the end of our maximum iterations, all we would have is an empty cluster or a cluster with only one data point (If the noisy centroid is a data point itself). RSS will be relatively large in this case.

But this is not the only solution. There are many more solutions available. If we chose a point that is not noise, to start clustering, we will get a cluster assignment that has lower RSS. Thus if we achieve an empty dataset, we cannot claim that this solution would be a globally minimum solution. In fact achieving a globally minimum solution using KMeans is computationally very expensive and is an np-hard problem.

Exercise 3:

3) Computes the fractional membership of a document in a cluster as a function of the distance D from its centroid. Write very detailed pseudo code of kMeans using this soft version.

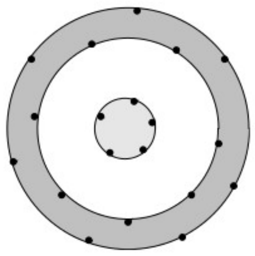
The Pseudo code for fuzzy or soft KMeans clustering is as follows.

```
1.  Let DS denote a set of all given vectors.
2.  Let K, S and Mi be the cluster size, initial seed and maximum iterations specified by user.
3.  Let D and M be collections to store distances and memberships for each vectors.
4.  stopping_condition_not_met = True
5.  Use S to select K random points as initial centroids.
6.  while(stopping_condition_not_met)
7.          Form K clusters by assigning all points to centroid where they have high membership
                                                                                        value
8.          for each point p in DS
9.                  for i in range (0 to K)
10.                         D[i] = || p – Ki  ||     //Find distances between all centroids
11.                 end for
12.                 for i in range (0 to K)
13.                         M[i] = 1 - ( D[i] / sum(D) )     //Find membership
14.                 end for
15.          end for
16.         Recompute the centroid of each cluster
17.         iteration = iteration + 1
18.         if (iteration == Mi) or (previous centroids == current centroids)
19.                  stopping_condition_not_met = False
20. Output cluster results
```

Exercise 5:

5) Consider two clusters that are a circle and a surrounding ring, as below. Suppose:
i. The radius of the circle is c.
ii. The inner and outer circles forming the ring have radii i and o, respectively.

iii. All representative points for the two clusters are on the boundaries of the clusters.
iv. Representative points are moved 20% of the distance from their initial position toward the centroid of their cluster.
v. Clusters are merged if, after repositioning, there are representative points from the two clusters at distance d or less.
In terms of d, c, i, and o, under what circumstances will the ring and circle be merged into a single cluster?



Cure algorithm works by hierarchical data clustering followed by moving the cluster representatives and merging them when are close.

For both circles to converge in this case, the i and c should be nearly equal so that the space between the circle and the ring is less. O should not be very much larger than i or else it will take too much time to converge.

Weka Experiments:

Cluster analysis is a means by which points or documents in a given dataset are grouped in such fashion that they are similar to points or documents in their own group than to the those in the other groups. This analysis is a fundamental task in data exploration and mining and is employed in a broad spectrum of fields and applications. To perform this analysis now, we use the machine learning tool Weka with the iris and the vote datasets that are available along with Weka.

Details about the datasets:

The iris dataset consists of 50 instances of each of the three categories namely setosa, virginica and versicolor. The vote dataset on the other hand, is the gathering of the votes cast by Republicans and Democrats, 435 in total, on key governmental issues. The number of attributes in the iris and the vote datasets are 4 and 16 respectively. Both the datasets have the class attribute available already which will come in handy when validating the clusters with the 'Classes to cluster evaluation' option present in Weka. But we will not use the class information during clustering, to fairly weigh the effectiveness and efficiency of the clustering algorithms.

Details about the algorithms:

Kmeans clustering algorithm is partitional method, which works by choosing a k number of initial representative points, grouping every point in the provided data set with its nearest representative point and then computing the new representative points based on the formed groups. This process of forming representative points (usually centroids of groups) and re-grouping is done iteratively till the centroids don't change or relatively few points switch clusters. Here k is the number of clusters needed. This is one of the least complex unsupervised learning algorithms that are used widely.

DBScan clustering algorithm, which is density based, has two main inputs that define its approach. Those are the radius (Ep) of the neighbourhood, in which density is to be checked and the minimum count of points that need to be present inside an area of radius Ep, for the algorithm to consider it dense. First, this algorithm categorizes every point in the given dataset as one of the core, boundary or noise points. This is done by checking the number of neighbours that each point has, in its vicinity of radius Ep. If the number is greater than the minimum count specified, the point is categorized as a core point. If a point is not core, but is near another core point, then it is categorized as a boundary point. If a point does not qualify to be a core point or a boundary point, it is labelled as a noise point.

Noise points are eliminated and with the rest of points, clustering is done by this process: assigning a cluster label to a core point initially, which labels all points in its Ep neighbourhood, its current cluster label. Then this process is repeated for every other core point that does not have a cluster label.

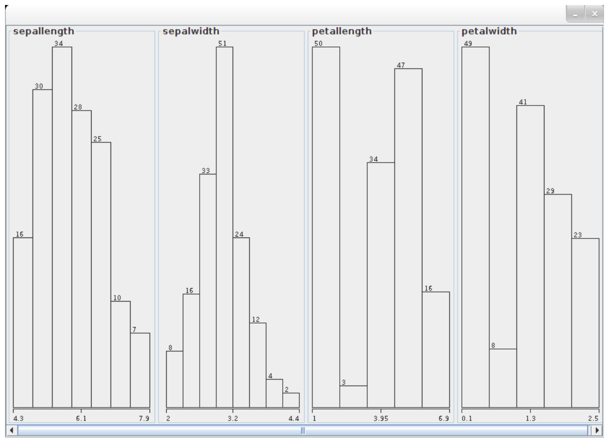
Visualization of data:

Generally, just by observing the spread of the datasets, we can make strong initial assessments. By using the Weka visualizer in the Preprocess tab, we observe the datasets to get the following estimations.

Iris:

Distribution of instances based on each attribute is visualized below where we can see that there are gradually increasing and decreasing distributions in case of sepal length and sepal width attribute values. With that kind of distribution, it is hard to make a call on number of clusters that will give be optimum and give less SSE. But in case of petal length and petal width, the instance distribution is not gradual. There are two disjoint peaks (50 & 47 in ~1-2 & ~4-5 range in case of petal length and 49 & 41 in ~0.1-0.5 & ~1-1.5 range in case of petal width) and several smaller peaks in the instance distribution plot based on petal length and width.

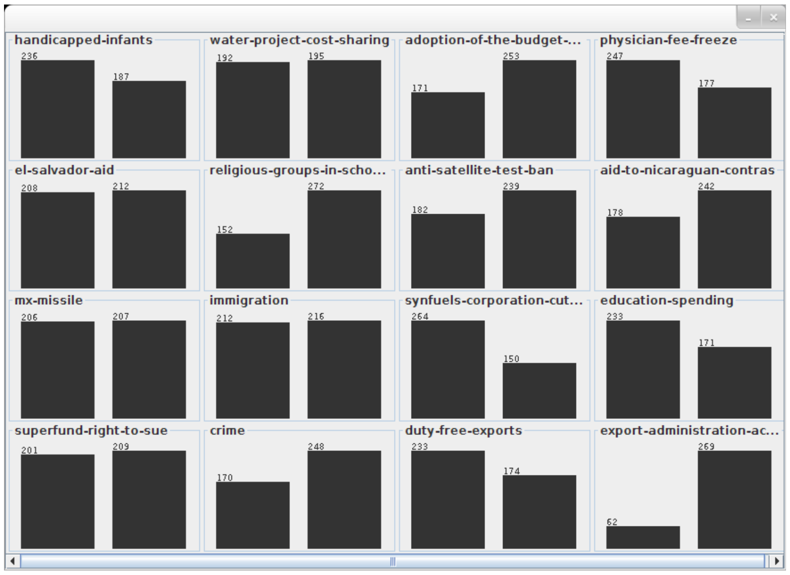
From here, we can assume that the number of clusters should be greater than 2, to accommodate the disjoint peaks and the rest of the smaller peaks. Also we guess that if only petal length or width is used to form clusters, then the clusters will be well demarcated, clear and will have less SSE due to the relatively dense distribution. In that case, incorrectly classified instances is also likely to be low. If sepal length or width is also used with petal length or width when creating clusters, then the resultant clusters and the metrics (SSE and incorrectly classified instances) might not be as effective as that achieved using only petal length or width, owing to the ambiguity that might arise in the estimation of cluster borders in this case.



Vote:

Visualization of distribution of records under vote dataset follows, where we can see that there are only two nominal outcomes each attribute value. Hence choosing the cluster size as 2 will most likely give the optimum clustering solution. The other cluster sizes can also be tried to make sure that our assumption is correct.

To predict the attributes that will give clusters with desirable SSE in this situation, we observe the magnitude of support for each outcome. Attributes where this support for outcome is comparable may not become good choices because we may not be able to divide the instances distinctly on the basis of values for these attributes. On the other hand, attributes which do not have commensurate support can naturally help in dividing the records into non-overlapping parts. But this assumption may not hold good for all cases. There might be attributes where the advantage gained by disparate outcome support is waned by unequal class distribution in them. So, attributes like physician-fee-freeze and education-spending should give clusters with desirable characteristics.



Clustering data using default parameters:

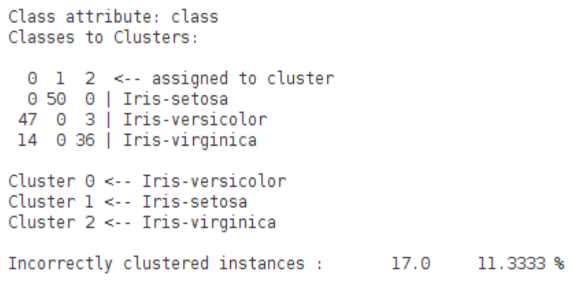
Weka has several clustering algorithms available out of which we are going to be using SimpleKMeans and DBScan for this assignment. Before starting with our actual experiments it would be great help to know how we can validate the clusters formed as result of our experiments and the algorithm parameters that influence the experiment results.

(i) Cluster evaluation in Weka:

It is very important to evaluate the outcome of any experiment for correctness and in our case now, Weka facilitates the verification of clusters created, with the following options.

Classes to clusters evaluation:

This option outputs the details of contents of the clusters by class values. Ideally we would want all distinct class values in disparate clusters. Looking at the confusion matrix achieved using this add-on tool, we can easily interpret the result of a clustering algorithm, on lines of its efficiency and mistakes.

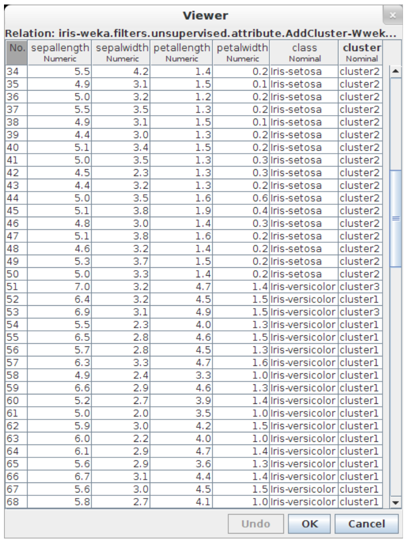


```
Class attribute: class
Classes to Clusters:

  0  1  2  <-- assigned to cluster
  0 50  0 | Iris-setosa
 47  0  3 | Iris-versicolor
 14  0 36 | Iris-virginica

Cluster 0 <-- Iris-versicolor
Cluster 1 <-- Iris-setosa
Cluster 2 <-- Iris-virginica

Incorrectly clustered instances :       17.0     11.3333 %
```

The above screensnip shows the output of this option in Weka for SimpleKMeans with default parameters. We can see that 3 instances of versicolor and 14 instances of virginica are incorrectly clustered. This evaluation method will be discussed more, in detail, in the upcoming pages.

AddCluster filter:

This is an unsupervised attribute filter which when specified a clustering algorithm adds a new attribute called cluster to the dataset which can compared with the class attribute to discover the clustering algorithm performance on each and every record. The effect of this filter on iris dataset using Kmeans with default values is as below.

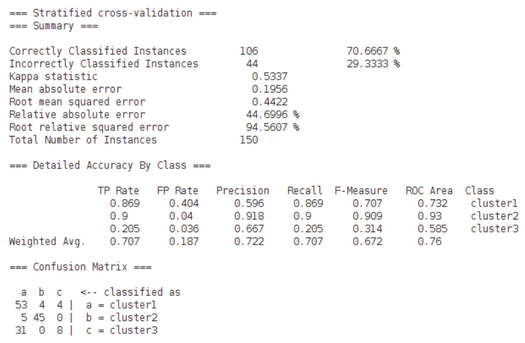


Relation: iris-weka.filters.unsupervised.attribute.AddCluster-Wwek...

No.	sepallength Numeric	sepalwidth Numeric	petallength Numeric	petalwidth Numeric	class Nominal	cluster Nominal
34	5.5	4.2	1.4	0.2	Iris-setosa	cluster2
35	4.9	3.1	1.5	0.1	Iris-setosa	cluster2
36	5.0	3.2	1.2	0.2	Iris-setosa	cluster2
37	5.5	3.5	1.3	0.2	Iris-setosa	cluster2
38	4.9	3.1	1.5	0.1	Iris-setosa	cluster2
39	4.4	3.0	1.3	0.2	Iris-setosa	cluster2
40	5.1	3.4	1.5	0.2	Iris-setosa	cluster2
41	5.0	3.5	1.3	0.3	Iris-setosa	cluster2
42	4.5	2.3	1.3	0.3	Iris-setosa	cluster2
43	4.4	3.2	1.3	0.2	Iris-setosa	cluster2
44	5.0	3.5	1.6	0.6	Iris-setosa	cluster2
45	5.1	3.8	1.9	0.4	Iris-setosa	cluster2
46	4.8	3.0	1.4	0.3	Iris-setosa	cluster2
47	5.1	3.8	1.6	0.2	Iris-setosa	cluster2
48	4.6	3.2	1.4	0.2	Iris-setosa	cluster2
49	5.3	3.7	1.5	0.2	Iris-setosa	cluster2
50	5.0	3.3	1.4	0.2	Iris-setosa	cluster2
51	7.0	3.2	4.7	1.4	Iris-versicolor	cluster3
52	6.4	3.2	4.5	1.5	Iris-versicolor	cluster1
53	6.9	3.1	4.9	1.5	Iris-versicolor	cluster3
54	5.5	2.3	4.0	1.3	Iris-versicolor	cluster1
55	6.5	2.8	4.6	1.5	Iris-versicolor	cluster1
56	5.7	2.8	4.5	1.3	Iris-versicolor	cluster1
57	6.3	3.3	4.7	1.6	Iris-versicolor	cluster1
58	4.9	2.4	3.3	1.0	Iris-versicolor	cluster1
59	6.6	2.9	4.6	1.3	Iris-versicolor	cluster1
60	5.2	2.7	3.9	1.4	Iris-versicolor	cluster1
61	5.0	2.0	3.5	1.0	Iris-versicolor	cluster1
62	5.9	3.0	4.2	1.5	Iris-versicolor	cluster1
63	6.0	2.2	4.0	1.0	Iris-versicolor	cluster1
64	6.1	2.9	4.7	1.4	Iris-versicolor	cluster1
65	5.6	2.9	3.6	1.3	Iris-versicolor	cluster1
66	6.7	3.1	4.4	1.4	Iris-versicolor	cluster1
67	5.6	3.0	4.5	1.5	Iris-versicolor	cluster1
68	5.8	2.7	4.1	1.0	Iris-versicolor	cluster1

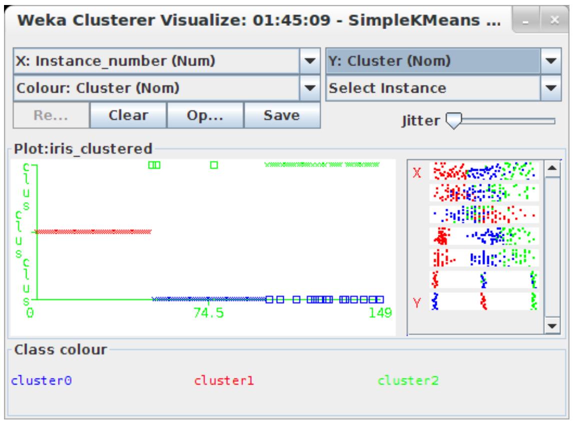
Classification via clustering:

This is a meta classifier which works by clustering the instances of a dataset and assigning all instances in a cluster, a class value that is found dominant in that particular cluster. Although this classifier is not as efficient as the other advanced classifiers such as J48, this option can be used to compare different types of clustering approaches. Output of this tool (on iris set with default Kmeans, is as below) is similar to that produced normally by other classifiers.



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         106               70.6667 %
Incorrectly Classified Instances        44               29.3333 %
Kappa statistic                          0.5337
Mean absolute error                      0.1956
Root mean squared error                  0.4422
Relative absolute error                 44.6996 %
Root relative squared error             94.5607 %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.869     0.404      0.596     0.869     0.707      0.732    cluster1
                 0.9       0.04       0.918     0.9       0.909      0.93     cluster2
                 0.205     0.036      0.667     0.205     0.314      0.585    cluster3
Weighted Avg.    0.707     0.187      0.722     0.707     0.672      0.76

=== Confusion Matrix ===

  a  b  c   <-- classified as
 53  4  4 |  a = cluster1
  5 45  0 |  b = cluster2
 31  0  8 |  c = cluster3
```

Visualization of result:
This is the traditional approach to cluster verification where we can right click on the clustering result in the result list section, select 'Visualize cluster assignments' and plot the instance number and any attribute. The cluster detail along with the info on whether it is correctly clustered or not (a square indicates a wrong assignment), can be gathered from this visualization. Example is below where the effect of default K-Means on iris set is visualized.



(ii) Parameters of algorithms:
The significant parameters that are supported by Weka, for the two algorithms that we are going to use, are as follows.

SimpleKMeans:
maxIterations: It is possible that our clustering algorithms can run excruciatingly long and hence it is good to set a maximum number of iterations that our algorithm will perform before stopping.
numClusters: This parameter lets the end users to specify the number of clusters that they need created from the given dataset.
distanceFunction: This parameter gives the freedom to choose the distance measure that we like. Default value used in Weka is the Euclidean distance.
displayStdDevs: As the name indicates, it displays the standard deviation values for numeric attributes which will be useful in understanding the result. In case of nominal attributes, counts are displayed.
dontReplaceMissingValues: When set to true, this algorithm replaces missing values in the dataset globally with mean or mode.
preserveInstancesOrder: This parameter is used to instruct the algorithm, whether or not to maintain the instance order.
seed: This is the random number seed that will be used by the algorithm for initial centroid generation and so on.

DBScan:
database_Type and **database_distanceType:** These parameters specify the type of database and the distance type that the algorithm uses.
epsilon: This parameter lets the users indicate the radius around every point, in which the algorithm will check whether the minPoints criteria is satisfied.
minPoints: This is the least number of data points that are required in an epsilon range for the algorithm to classify a point as a core point. If a point does not satisfy this criteria and is not near any

point that satisfies this criteria, then it is removed as a noise point.

(iii) Run with default parameters:

Now using both the datasets, we run the SimpleKMeans and DBScan algorithms with default values, document and analyze the results.

Using SimpleKMeans algorithm: Summary:

The important readings from the output of SimpleKMeans algorithm are tabulated below.

Testing option	Classes to cluster evaluation	
Dataset	Iris dataset	Vote dataset
With default parameters, clustering using all attributes		
Number of Iterations	7	3
Within cluster SSE	12.1436	1449.0
Incorrectly classified instances	33.3333%	14.023%

Confusion matrix	`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`
		`0   1  <-- assigned to cluster`
		`50 217 | democrat`
		`157  11 | republican`
		
		`Cluster 0 <-- republican`
		`Cluster 1 <-- democrat`

Inferences:

- Here, default parameters in SimpleKMeans has performed poorly because it has a value of 2 for number of clusters. As we observed from data visualization, Iris dataset would need more than 2 clusters to give less SSE and less incorrectly classified instances.

- In case of vote dataset, the default parameters seems to work well, since the default number of clusters 2, matches its true clusters. We will experiment with different values for parameters and use different attributes to cluster and see if this accuracy can be improved.

Using DBScan algorithm: Summary:

The crucial readings from the output of DBScan algorithm are recorded below.

Testing option	Classes to cluster evaluation	
Dataset	Iris dataset	Vote dataset
With default parameters, clustering using all attributes		
Incorrectly classified instances	66.6667%	21.8391%
Clustered Instances	150	122

Confusion matrix	` 0  <-- assigned to cluster`
	`50 | Iris-setosa`
	`50 | Iris-versicolor`
	`50 | Iris-virginica`
	
	`Cluster 0 <-- Iris-setosa`
		`  0  1  2  3  4  5  6  7  8  9 10 11 12 13  <-- assigned to cluster`
		`  0  0 14  0  0  0  0  0  0  8  9  8  6  7 | democrat`
		` 12 13  0 10  8  8  6  7  6  0  0  0  0  0 | republican`
		
		`Cluster  0 <-- No class`
		`Cluster  1 <-- republican`
		`Cluster  2 <-- democrat`
		`Cluster  3 <-- No class`
		`Cluster  4 <-- No class`
		`Cluster  5 <-- No class`
		`Cluster  6 <-- No class`
		`Cluster  7 <-- No class`
		`Cluster  8 <-- No class`
		`Cluster  9 <-- No class`
		`Cluster 10 <-- No class`
		`Cluster 11 <-- No class`
		`Cluster 12 <-- No class`
		`Cluster 13 <-- No class`

Inferences:

- Here, default parameters in DBScan has performed very poorly on Iris because with the values of 0.9 for epsilon and 6 for minimum number of points, most points would be rejected as noise. The Iris data is not that dense and hence this poor outcome has occurred. One way to improve the performance of DBScan now, is to reduce epsilon and change the minimum number of points for future experiments.

- In case of vote dataset, again the default parameters do not work well, with too many values clustered as no class separately. The high epsilon value would be the cause here too, we suppose. We can confirm our guesses with the upcoming experiments.

Clustering using selected attributes:

Using both the datasets, we run the SimpleKMeans and DBScan algorithms again, but this time we use only selected attributes for clustering, with mostly default algorithmic parameters. We note and reflect upon results here as well.

Using SimpleKMeans algorithm: Summary: (Default parameters with cluster size = 3)

Testing option	Classes to cluster evaluation	
Dataset	Iris dataset	Vote dataset
	Clustering using petal length alone	***Clustering using physician-fee-freeze alone***
Number of Iterations	5	2
Within cluster SSE	0.7083	0.0
Incorrectly classified instances	6.6667%	4.3678%

Confusion matrix	` 0  1  2  <-- assigned to cluster`
	` 0 50  0 | Iris-setosa`
	`49  0  1 | Iris-versicolor`
	` 9  0 41 | Iris-virginica`
	
	`Cluster 0 <-- Iris-versicolor`
	`Cluster 1 <-- Iris-setosa`
	`Cluster 2 <-- Iris-virginica`
		`Class attribute: Class`
		`Classes to Clusters:`
		
		`   0   1  <-- assigned to cluster`
		`  14 253 | democrat`
		` 163   5 | republican`
		
		`Cluster 0 <-- republican`
		`Cluster 1 <-- democrat`

Testing option	Classes to cluster evaluation

Dataset	Iris dataset	Vote dataset
	Clustering using petal width alone	***Clustering using physician-fee-freeze and adoption-of-budget-res alone***
Number of Iterations	5	3
Within cluster SSE	0.8572	56.0
Incorrectly classified instances	4%	9.1954%

Confusion matrix	`  0  1  2  <-- assigned to cluster`
	`  0 50  0 | Iris-setosa`
	` 49  0  1 | Iris-versicolor`
	`  5  0 45 | Iris-virginica`
	
	`Cluster 0 <-- Iris-versicolor`
	`Cluster 1 <-- Iris-setosa`
	`Cluster 2 <-- Iris-virginica`
		`   0   1  <-- assigned to cluster`
		`  37 230 | democrat`
		` 165   3 | republican`
		
		`Cluster 0 <-- republican`
		`Cluster 1 <-- democrat`

Testing option	Classes to cluster evaluation	
Dataset	Iris dataset	Vote dataset
	Clustering using sepal length and sepal width alone	***Clustering using phy-fee-freeze, el-s-aid, edu-spend and adop-of-bud-res alone***
Number of Iterations	7	3
Within cluster SSE	4.1438	166
Incorrectly classified instances	22.6667%	9.1954%

Confusion matrix	`  0  1  2  <-- assigned to cluster`
	` 49  1  0 | Iris-setosa`
	`  0 37 13 | Iris-versicolor`
	`  0 20 30 | Iris-virginica`
	
	`Cluster 0 <-- Iris-setosa`
	`Cluster 1 <-- Iris-versicolor`
	`Cluster 2 <-- Iris-virginica`
		`   0   1  <-- assigned to cluster`
		`  33 234 | democrat`
		` 161   7 | republican`
		
		`Cluster 0 <-- republican`
		`Cluster 1 <-- democrat`

Testing option	Classes to cluster evaluation	
Dataset	Iris dataset	Vote dataset
	Clustering using petal length, sepal length and sepal width alone	***Clustering using attributes other than phy-fee-freeze, el-s-aid, edu-spend and adop-of-bud-res***
Number of Iterations	6	3
Within cluster SSE	5.3235	1196.0
Incorrectly classified instances	16.6667%	22.7586%

Confusion matrix	
	```
	 0  1  2  <-- assigned to cluster
	 0 50  0 | Iris-setosa
	40  0 10 | Iris-versicolor
	15  0 35 | Iris-virginica

	Cluster 0 <-- Iris-versicolor
	Cluster 1 <-- Iris-setosa
	Cluster 2 <-- Iris-virginica
	```
		```
		  0   1  <-- assigned to cluster
		 85 182 | democrat
		154  14 | republican

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

Inferences:

- In the above set of experiments, we saw that our estimations based on visualization hold good. When we used only petal length for clustering Iris, we get very low SSE and less incorrectly classified instances. It is the same case for vote dataset, when we use only physician-fee-freeze, the attribute with the disparate outcome support magnitude, for clustering, SSE is zero and incorrectly classified instances are very less.

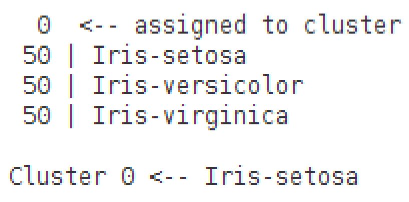
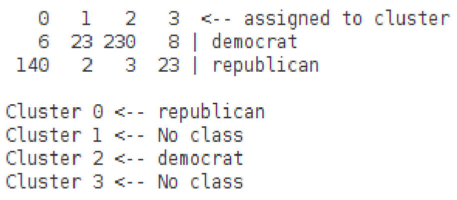
- When we mix up the best attributes with the attributes that don't have distinct instance distribution, we can see that the SSE and the incorrectly classified instances increase. Thus we can conclude that if we cluster based upon attributes that have well spread out instance distributions, we can get low SSE and the clustering will be very effective.

Using DBScan algorithm: Summary: (For Iris, 0.2 epsilon with 2 minimum points was used while default parameters were used for vote.)

Testing option	Classes to cluster evaluation	
Dataset	Iris dataset	Vote dataset
	Clustering using petal length alone	***Clustering using physician-fee-freeze alone***
Incorrectly classified instances	66.6667%	4.3678%
Clustered Instances	150	435

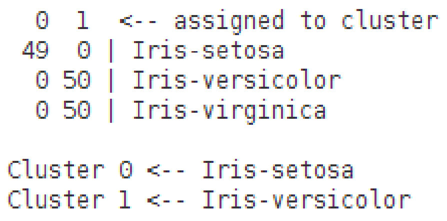
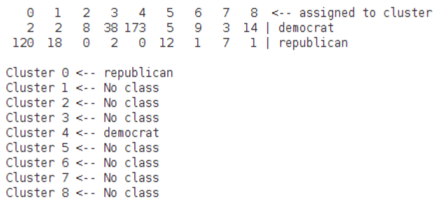
Confusion matrix	
	```
	 0  <-- assigned to cluster
	50 | Iris-setosa
	50 | Iris-versicolor
	50 | Iris-virginica

	Cluster 0 <-- Iris-setosa
	```
		```
		  0   1  <-- assigned to cluster
		 14 253 | democrat
		163   5 | republican

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

Testing option	Classes to cluster evaluation	
Dataset	Iris dataset	Vote dataset
	Clustering using petal width alone	***Clustering using physician-fee-freeze and adoption-of-budget-res alone***
Incorrectly classified instances	66.6667%	14.9425%
Clustered Instances	150	435

Confusion matrix		

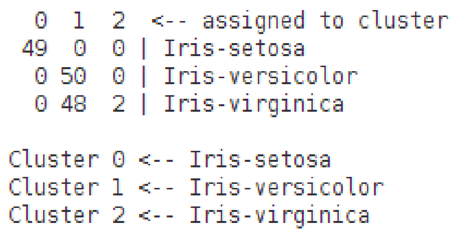
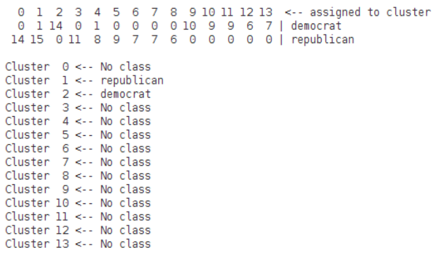
```
  0  <-- assigned to cluster
 50 | Iris-setosa
 50 | Iris-versicolor
 50 | Iris-virginica

Cluster 0 <-- Iris-setosa
```

```
   0   1   2   3  <-- assigned to cluster
   6  23 230   8 | democrat
 140   2   3  23 | republican

Cluster 0 <-- republican
Cluster 1 <-- No class
Cluster 2 <-- democrat
Cluster 3 <-- No class
```

Testing option	Classes to cluster evaluation	
Dataset	Iris dataset	Vote dataset
	Clustering using petal length, sepal length and sepal width alone	***Clustering using phy-fee-freeze, el-s-aid, edu-spend and adop-of-bud-res alone***
Incorrectly classified instances	33.3333%	28.046 %
Clustered Instances	149	415
Confusion matrix		

```
  0  1   <-- assigned to cluster
 49  0 | Iris-setosa
  0 50 | Iris-versicolor
  0 50 | Iris-virginica

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

```
   0   1   2   3   4   5   6   7   8  <-- assigned to cluster
   2   2   8  38 173   5   9   3  14 | democrat
 120  18   0   2   0  12   1   7   1 | republican

Cluster 0 <-- republican
Cluster 1 <-- No class
Cluster 2 <-- No class
Cluster 3 <-- No class
Cluster 4 <-- democrat
Cluster 5 <-- No class
Cluster 6 <-- No class
Cluster 7 <-- No class
Cluster 8 <-- No class
```

Testing option	Classes to cluster evaluation	
Dataset	Iris dataset	Vote dataset
	Clustering using all 4 attributes ignoring class	***Clustering using attributes other than phy-fee-freeze, el-s-aid, edu-spend and adop-of-bud-res***
Incorrectly classified instances	32.0%	24.1379%
Clustered Instances	149	134
Confusion matrix		

```
  0  1  2  <-- assigned to cluster
 49  0  0 | Iris-setosa
  0 50  0 | Iris-versicolor
  0 48  2 | Iris-virginica

Cluster 0 <-- Iris-setosa
Cluster 1 <-- Iris-versicolor
Cluster 2 <-- Iris-virginica
```

```
  0  1  2  3  4  5  6  7  8  9 10 11 12 13  <-- assigned to cluster
  0  1 14  0  1  0  0  0  0 10  9  9  6  7 | democrat
 14 15  0 11  8  9  7  7  6  0  0  0  0  0 | republican

Cluster  0 <-- No class
Cluster  1 <-- republican
Cluster  2 <-- democrat
Cluster  3 <-- No class
Cluster  4 <-- No class
Cluster  5 <-- No class
Cluster  6 <-- No class
Cluster  7 <-- No class
Cluster  8 <-- No class
Cluster  9 <-- No class
Cluster 10 <-- No class
Cluster 11 <-- No class
Cluster 12 <-- No class
Cluster 13 <-- No class
```

Inferences:

- One important inference made out of the above experiments is that when the density of data distribution is sparse or varying, DBScan cannot perform well. In case of Iris clustered with petal length alone, we can see that DBScan performs poorly, due to this reason. It has clustered 66% of data incorrectly. Also if the data is spread is highly dimensional, then also

DBScan does not perform well, which is also evident from the experiments above.

- In case of vote datasets though, DBScan does not have any conspicuous performance issues in case of best attributes. But on mix up, as we have seen before, performance is bad and is as expected. Example is the last table in the section above where the incorrectly classified instances are relatively more at 24%

Clustering using different sets of parameters:

Here, we run the SimpleKMeans and DBScan algorithms one more time, with different sets of parameters. Inferences follow the tables.

Using SimpleKMeans algorithm: Summary: (Using all attributes)

Changing number of clusters:

Testing option	Classes to cluster evaluation			
	Number of clusters			
	1	*2*	*3*	*4*
Dataset	Iris dataset			
Number of Iterations	*1*	*7*	*6*	*4*
Within cluster SSE	*47.39*	*12.14*	*6.99*	*5.53*
Incorrectly classified instances	*66.66%*	*33.33%*	*11.33%*	*29.33%*

Confusion matrix	`0 <-- assigned to cluster`
	`50 | Iris-setosa`
	`50 | Iris-versicolor`
	`50 | Iris-virginica`
	
	`Cluster 0 <-- Iris-setosa`
		`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`
			`0 1 2 <-- assigned to cluster`
			`0 50 0 | Iris-setosa`
			`47 0 3 | Iris-versicolor`
			`14 0 36 | Iris-virginica`
			
			`Cluster 0 <-- Iris-versicolor`
			`Cluster 1 <-- Iris-setosa`
			`Cluster 2 <-- Iris-virginica`
				`0 1 2 3 <-- assigned to cluster`
				`0 0 0 50 | Iris-setosa`
				`23 27 0 0 | Iris-versicolor`
				`19 2 29 0 | Iris-virginica`
				
				`Cluster 0 <-- No class`
				`Cluster 1 <-- Iris-versicolor`
				`Cluster 2 <-- Iris-virginica`
				`Cluster 3 <-- Iris-setosa`

Dataset	Vote dataset			
Number of Iterations	*1*	*3*	*5*	*3*
Within cluster SSE	*3173.0*	*1449.0*	*1296.0*	*1225.0*
Incorrectly classified instances	*38.62%*	*14.02%*	*23.90%*	*32.64%*

Confusion matrix	`0 <-- assigned to cluster`
	`267 | democrat`
	`168 | republican`
	
	`Cluster 0 <-- democrat`
		`0 1 <-- assigned to cluster`
		`50 217 | democrat`
		`157 11 | republican`
		
		`Cluster 0 <-- republican`
		`Cluster 1 <-- democrat`
			`0 1 2 <-- assigned to cluster`
			`43 176 48 | democrat`
			`155 10 3 | republican`
			
			`Cluster 0 <-- republican`
			`Cluster 1 <-- democrat`
			`Cluster 2 <-- No class`
				`0 1 2 3 <-- assigned to cluster`
				`22 45 52 148 | democrat`
				`145 7 9 7 | republican`
				
				`Cluster 0 <-- republican`
				`Cluster 1 <-- No class`
				`Cluster 2 <-- No class`
				`Cluster 3 <-- democrat`

Changing maximum number of iterations:

Testing option	Classes to cluster evaluation			
	Maximum number of iterations			
	1	*3*	*5*	*7*
Dataset	Iris dataset			
Number of clusters	*3*	*3*	*3*	*3*
Within cluster SSE	*36.93*	*7.44*	*7.00*	*6.998*
Incorrectly classified instances	*16.66%*	*10.66%*	*11.33%*	*11.33%*

Confusion matrix	` 0  1  2  <-- assigned to cluster`
	`  0 50  0 | Iris-setosa`
	` 45  5  0 | Iris-versicolor`
	` 20  0 30 | Iris-virginica`
	
	`Cluster 0 <-- Iris-versicolor`
	`Cluster 1 <-- Iris-setosa`
	`Cluster 2 <-- Iris-virginica`
		` 0  1  2  <-- assigned to cluster`
		`  0 50  0 | Iris-setosa`
		` 48  0  2 | Iris-versicolor`
		` 14  0 36 | Iris-virginica`
		
		`Cluster 0 <-- Iris-versicolor`
		`Cluster 1 <-- Iris-setosa`
		`Cluster 2 <-- Iris-virginica`
			` 0  1  2  <-- assigned to cluster`
			`  0 50  0 | Iris-setosa`
			` 47  0  3 | Iris-versicolor`
			` 14  0 36 | Iris-virginica`
			
			`Cluster 0 <-- Iris-versicolor`
			`Cluster 1 <-- Iris-setosa`
			`Cluster 2 <-- Iris-virginica`
				` 0  1  2  <-- assigned to cluster`
				`  0 50  0 | Iris-setosa`
				` 47  0  3 | Iris-versicolor`
				` 14  0 36 | Iris-virginica`
				
				`Cluster 0 <-- Iris-versicolor`
				`Cluster 1 <-- Iris-setosa`
				`Cluster 2 <-- Iris-virginica`

Dataset	Vote dataset			
Number of clusters	*2*	*2*	*2*	*2*
Within cluster SSE	*2419.0*	*1449.0*	*1449.0*	*1449.0*
Incorrectly classified instances	*14.02%*	*14.02%*	*14.02%*	*14.02%*

Confusion matrix	`  0   1  <-- assigned to cluster`
	`  50 217 | democrat`
	` 157  11 | republican`
	
	`Cluster 0 <-- republican`
	`Cluster 1 <-- democrat`
		`  0   1  <-- assigned to cluster`
		`  50 217 | democrat`
		` 157  11 | republican`
		
		`Cluster 0 <-- republican`
		`Cluster 1 <-- democrat`
			`  0   1  <-- assigned to cluster`
			`  50 217 | democrat`
			` 157  11 | republican`
			
			`Cluster 0 <-- republican`
			`Cluster 1 <-- democrat`
				`  0   1  <-- assigned to cluster`
				`  50 217 | democrat`
				` 157  11 | republican`
				
				`Cluster 0 <-- republican`
				`Cluster 1 <-- democrat`

Changing the random seed:

Testing option	Classes to cluster evaluation			
	Random seed			
	10	*11*	*20*	*35*
Dataset	Iris dataset			
Number of Iterations	*6*	*5*	*9*	*5*
Within cluster SSE	*6.998*	*6.998*	*7.13*	*6.998*
Incorrectly classified instances	*11.33%*	*11.33%*	*12%*	*11.33%*

Confusion matrix	` 0  1  2  <-- assigned to cluster`
	` 0 50  0 | Iris-setosa`
	`47  0  3 | Iris-versicolor`
	`14  0 36 | Iris-virginica`
	
	`Cluster 0 <-- Iris-versicolor`
	`Cluster 1 <-- Iris-setosa`
	`Cluster 2 <-- Iris-virginica`
		` 0  1  2  <-- assigned to cluster`
		` 0  0 50 | Iris-setosa`
		`47  3  0 | Iris-versicolor`
		`14 36  0 | Iris-virginica`
		
		`Cluster 0 <-- Iris-versicolor`
		`Cluster 1 <-- Iris-virginica`
		`Cluster 2 <-- Iris-setosa`
			` 0  1  2  <-- assigned to cluster`
			` 0  0 50 | Iris-setosa`
			`40 10  0 | Iris-versicolor`
			` 8 42  0 | Iris-virginica`
			
			`Cluster 0 <-- Iris-versicolor`
			`Cluster 1 <-- Iris-virginica`
			`Cluster 2 <-- Iris-setosa`
				` 0  1  2  <-- assigned to cluster`
				` 0 50  0 | Iris-setosa`
				` 3  0 47 | Iris-versicolor`
				`36  0 14 | Iris-virginica`
				
				`Cluster 0 <-- Iris-virginica`
				`Cluster 1 <-- Iris-setosa`
				`Cluster 2 <-- Iris-versicolor`

Dataset	Vote dataset			
Number of Iterations	*3*	*4*	*3*	*5*
Within cluster SSE	*1449.0*	*1448.0*	*1457.0*	*1457.0*
Incorrectly classified instances	*14.02%*	*13.56%*	*13.33%*	*13.33%*

Confusion matrix	`  0   1  <-- assigned to cluster`
	` 50 217 | democrat`
	`157  11 | republican`
	
	`Cluster 0 <-- republican`
	`Cluster 1 <-- democrat`
		`  0   1  <-- assigned to cluster`
		`220  47 | democrat`
		` 12 156 | republican`
		
		`Cluster 0 <-- democrat`
		`Cluster 1 <-- republican`
			`  0   1  <-- assigned to cluster`
			` 54 213 | democrat`
			`164   4 | republican`
			
			`Cluster 0 <-- republican`
			`Cluster 1 <-- democrat`
				`  0   1  <-- assigned to cluster`
				` 54 213 | democrat`
				`164   4 | republican`
				
				`Cluster 0 <-- republican`
				`Cluster 1 <-- democrat`

Inferences:

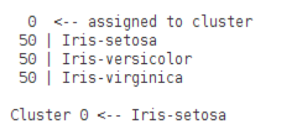
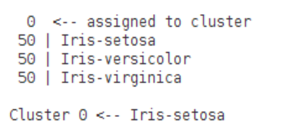
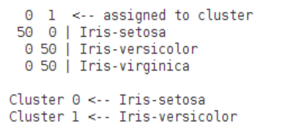
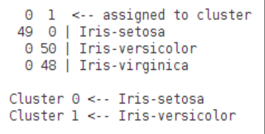
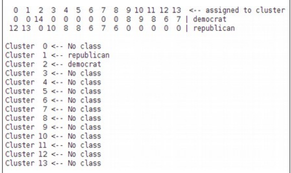
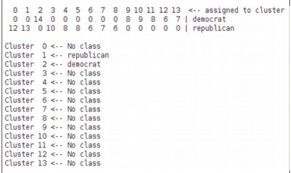
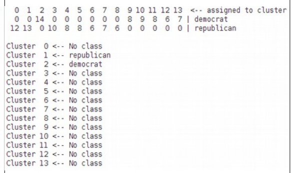
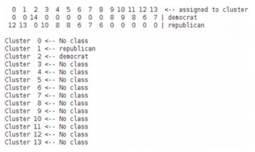
- Above set of experiments throw light on the important parameters and the desired values for the important parameters. As the number of clusters increases, the iterations increase which is as expected. The main outcome is that the performance is good (SSE=6.99) when the cluster size is chosen as 3 for Iris. This is in line with our visual observation. Although SSE lowers to 5.53 when the cluster size is 4, the incorrectly classifies instance number increases which is not desired. Similarly, performance on vote dataset is high when cluster size is two.

- In case of other parameters, if the number of iterations is set to one, clustering exits prematurely and is ineffective. If it is raised beyond a constant value, it has no impact. So each clustering algorithm has to run for a specific number of times for it to be effective. Changes in the random seed parameter changes results sometimes, since it influences the choice of initial centroids which is very crucial in KMeans. In case of vote dataset where number of records is more, the effect of random seed is felt more, since there are a lot of options for the algorithm to choose a random constant from.

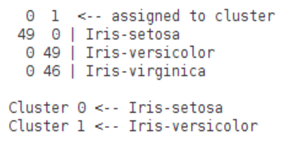
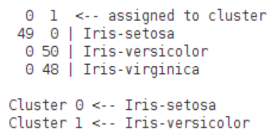
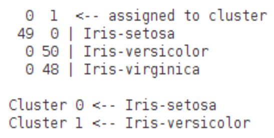
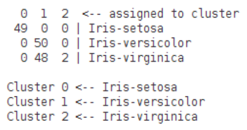
Using DBScan algorithm: Summary: (Using all attributes)

Changing the radius for neighbourhood check:

Testing option	Classes to cluster evaluation			
	Epsilon value with minPoints 6 for Iris and 10 for Vote			
	0.8	*0.6*	*0.4*	*0.2*
Dataset	Iris dataset			
Incorrectly classified instances	*66.66%*	*66.66%*	*33.33%*	*32%*

Clustered instances	*150*	*150*	*150*	*150*
Confusion matrix	 0 <-- assigned to cluster 50 | Iris-setosa 50 | Iris-versicolor 50 | Iris-virginica Cluster 0 <-- Iris-setosa	 0 <-- assigned to cluster 50 | Iris-setosa 50 | Iris-versicolor 50 | Iris-virginica Cluster 0 <-- Iris-setosa	 0 1 <-- assigned to cluster 50 0 | Iris-setosa 0 50 | Iris-versicolor 0 50 | Iris-virginica Cluster 0 <-- Iris-setosa Cluster 1 <-- Iris-versicolor	 0 1 <-- assigned to cluster 49 0 | Iris-setosa 0 50 | Iris-versicolor 0 48 | Iris-virginica Cluster 0 <-- Iris-setosa Cluster 1 <-- Iris-versicolor
Dataset	Vote dataset			
Incorrectly classified instances	*21.83%*	*21.83%*	*21.83%*	*21.83%*
Clustered instances	*122*	*122*	*122*	*122*
Confusion matrix	 0 1 2 3 4 5 6 7 8 9 10 11 12 13 <-- assigned to cluster 0 0 14 0 0 0 0 0 0 8 9 8 6 7 | democrat 12 13 0 10 8 8 6 7 6 0 0 0 0 0 | republican Cluster 0 <-- No class Cluster 1 <-- republican Cluster 2 <-- democrat Cluster 3 <-- No class Cluster 4 <-- No class Cluster 5 <-- No class Cluster 6 <-- No class Cluster 7 <-- No class Cluster 8 <-- No class Cluster 9 <-- No class Cluster 10 <-- No class Cluster 11 <-- No class Cluster 12 <-- No class Cluster 13 <-- No class	 0 1 2 3 4 5 6 7 8 9 10 11 12 13 <-- assigned to cluster 0 0 14 0 0 0 0 0 0 8 9 8 6 7 | democrat 12 13 0 10 8 8 6 7 6 0 0 0 0 0 | republican Cluster 0 <-- No class Cluster 1 <-- republican Cluster 2 <-- democrat Cluster 3 <-- No class Cluster 4 <-- No class Cluster 5 <-- No class Cluster 6 <-- No class Cluster 7 <-- No class Cluster 8 <-- No class Cluster 9 <-- No class Cluster 10 <-- No class Cluster 11 <-- No class Cluster 12 <-- No class Cluster 13 <-- No class	 0 1 2 3 4 5 6 7 8 9 10 11 12 13 <-- assigned to cluster 0 0 14 0 0 0 0 0 0 8 9 8 6 7 | democrat 12 13 0 10 8 8 6 7 6 0 0 0 0 0 | republican Cluster 0 <-- No class Cluster 1 <-- republican Cluster 2 <-- democrat Cluster 3 <-- No class Cluster 4 <-- No class Cluster 5 <-- No class Cluster 6 <-- No class Cluster 7 <-- No class Cluster 8 <-- No class Cluster 9 <-- No class Cluster 10 <-- No class Cluster 11 <-- No class Cluster 12 <-- No class Cluster 13 <-- No class	 0 1 2 3 4 5 6 7 8 9 10 11 12 13 <-- assigned to cluster 0 0 14 0 0 0 0 0 0 8 9 8 6 7 | democrat 12 13 0 10 8 8 6 7 6 0 0 0 0 0 | republican Cluster 0 <-- No class Cluster 1 <-- republican Cluster 2 <-- democrat Cluster 3 <-- No class Cluster 4 <-- No class Cluster 5 <-- No class Cluster 6 <-- No class Cluster 7 <-- No class Cluster 8 <-- No class Cluster 9 <-- No class Cluster 10 <-- No class Cluster 11 <-- No class Cluster 12 <-- No class Cluster 13 <-- No class

Changing the minimum number of points:

Testing option	Classes to cluster evaluation			
	minPoints with epsilon value 0.2			
	12	*10*	*8*	*6*
Dataset	Iris dataset			
Incorrectly classified instances	*30.66%*	*32%*	*32%*	*32%*
Clustered instances	*144*	*147*	*147*	*149*
Confusion matrix	 0 1 <-- assigned to cluster 49 0 | Iris-setosa 0 49 | Iris-versicolor 0 46 | Iris-virginica Cluster 0 <-- Iris-setosa Cluster 1 <-- Iris-versicolor	 0 1 <-- assigned to cluster 49 0 | Iris-setosa 0 50 | Iris-versicolor 0 48 | Iris-virginica Cluster 0 <-- Iris-setosa Cluster 1 <-- Iris-versicolor	 0 1 <-- assigned to cluster 49 0 | Iris-setosa 0 50 | Iris-versicolor 0 48 | Iris-virginica Cluster 0 <-- Iris-setosa Cluster 1 <-- Iris-versicolor	 0 1 2 <-- assigned to cluster 49 0 0 | Iris-setosa 0 50 0 | Iris-versicolor 0 48 2 | Iris-virginica Cluster 0 <-- Iris-setosa Cluster 1 <-- Iris-versicolor Cluster 2 <-- Iris-virginica
Dataset	Vote dataset			
Incorrectly classified instances	*2.75%*	*5.05%*	*14.48%*	*21.83%*
Clustered instances	*39*	*49*	*90*	*122*

Confusion matrix	` 0  1  2  <-- assigned to cluster`
	` 0  0 14 | democrat`
	`12 13  0 | republican`
	
	`Cluster 0 <-- No class`
	`Cluster 1 <-- republican`
	`Cluster 2 <-- democrat`
		` 0  1  2  3  <-- assigned to cluster`
		` 0  0 14  0 | democrat`
		`12 13  0 10 | republican`
		
		`Cluster 0 <-- No class`
		`Cluster 1 <-- republican`
		`Cluster 2 <-- democrat`
		`Cluster 3 <-- No class`
			` 0  1  2  3  4  5  6  7  8  <-- assigned to cluster`
			` 0  0 14  0  0  0  8  9  8 | democrat`
			`12 13  0 10  8  8  0  0  0 | republican`
			
			`Cluster 0 <-- No class`
			`Cluster 1 <-- republican`
			`Cluster 2 <-- democrat`
			`Cluster 3 <-- No class`
			`Cluster 4 <-- No class`
			`Cluster 5 <-- No class`
			`Cluster 6 <-- No class`
			`Cluster 7 <-- No class`
			`Cluster 8 <-- No class`
				` 0  1  2  3  4  5  6  7  8  9 10 11 12 13  <-- assigned to cluster`
				` 0  0 14  0  0  0  0  0  0  8  9  8  6  7 | democrat`
				`12 13  0 10  8  8  6  7  6  0  0  0  0  0 | republican`
				
				`Cluster  0 <-- No class`
				`Cluster  1 <-- republican`
				`Cluster  2 <-- democrat`
				`Cluster  3 <-- No class`
				`Cluster  4 <-- No class`
				`Cluster  5 <-- No class`
				`Cluster  6 <-- No class`
				`Cluster  7 <-- No class`
				`Cluster  8 <-- No class`
				`Cluster  9 <-- No class`
				`Cluster 10 <-- No class`
				`Cluster 11 <-- No class`
				`Cluster 12 <-- No class`
				`Cluster 13 <-- No class`

Inferences:

- It is very crucial to note that if the radius value is too big, we include all values and the clustering will not be efficient. If the value is too less, we will end up rejecting too many points as noise. What we need is an optimum amount for epsilon which is 0.2 for Iris and vote.

- The minimum number of points too should be optimal to get good results. High value for minimum points results in incorrectly labelling core points as boundary points which is not desired. For Iris, best results are obtained with minimum number of points as 6 and for vote, it is 10 with an epsilon value of 0.2.

Conclusion:

- SimpleKMeans performs relatively well in datasets with changing density of instance distributions and in case of dimensional data where DBScan performs poorly. It is not resistant to noise.

- DBScan scores over SimpleKMeans in case of non-globular shaped dataset distributions and in the case of noise. If the input dataset is noisy and is S shaped, DBScan can handle it while SimpleKMeans would fail. DBScan is noise resistant as well, noisy points are rejected even before the start of clustering in this approach.

- The major parameter for SimpleKMeans is number of clusters and for DBScan, it is epsilon and minimum number of points. The parameters have to be chosen carefully to get good clustering of data with optimal SSE.

- Visualizing the data before starting the experiments would help much in parameter selection and hence would give good clustering results.