CS5011: Machine Learning Programming Assignment 3

Rishiraj Surti, EE12B120

October 31, 2015

Note:

- For each of the clustering algorithms, class attribute has been ignored.
- For **each and every** implementation below, the corresponding models and ARFF files are attached.
- File description is provided in the respective sections.
- Cluster Purity is used for performance comparison of various clustering algorithms.
- Seperate code has been written to calculate Cluster Purity from ARFF file generated by Weka (which includes details about clusters as well). The Java project file is saved as *Cluster_Purity2*.

1 Clustering

1.1 Conversion to ARFF format

Full form of ARFF is Attribute Relation File Format. The data was converted to ARFF format by inserting @RELATION, @ATTRIBUTE and @DATA tags. For each dataset, the first column is attributed as *feature1*, second column as *feature2*, both with type *NUMERIC*, last column as *class* with class labels evident from the dataset.

1.2 Visualization

Each dataset is visualized by plotting *feature1* on *x-axis* vs *feature2* on *y-axis*. A sample screenshot:

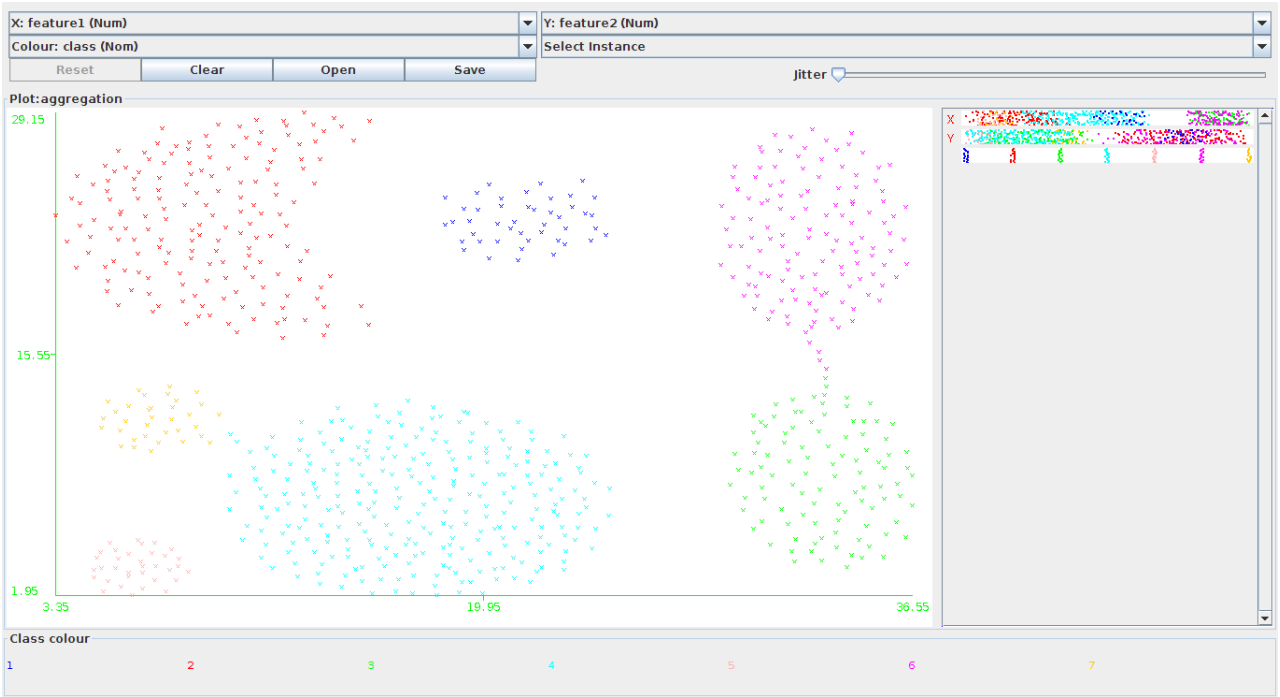

Figure 1:

A short introduction to Single and Complete Link Hierarchical Clustering (used to analyze datasets):
In Single Link Hierarchical clustering, distance measured is between the closest elements in clusters. It produces long chains. It groups the clusters in bottom-up fashion at each step combining two clusters that contain the closest pair of elements not yet belonging to the same cluster as each other. It tends to produce long thin clusters in which nearby elements of the same cluster have small distances, but elements at opposite ends of a cluster may be much farther from each other than to elements of other clusters.
In Complete Link Hierarchical clustering (also known as **farthest neighbour clustering**), distance measured is between the farthest elements in clusters. It forces spherical clusters with consistent diameter. At the beginning of the process, each element is in a cluster of its own. The clusters are then sequentially combined into larger clusters until all elements end up being in the same cluster. At each step, the two clusters separated by the shortest distance are combined. The distance between clusters equals the distance between those two elements (one in each cluster) that are farthest away from each other. The shortest of these links that remains at any step causes the fusion of the two clusters whose elements are involved.

Each of the datasets are visualized below and the algorithms are analyzed.

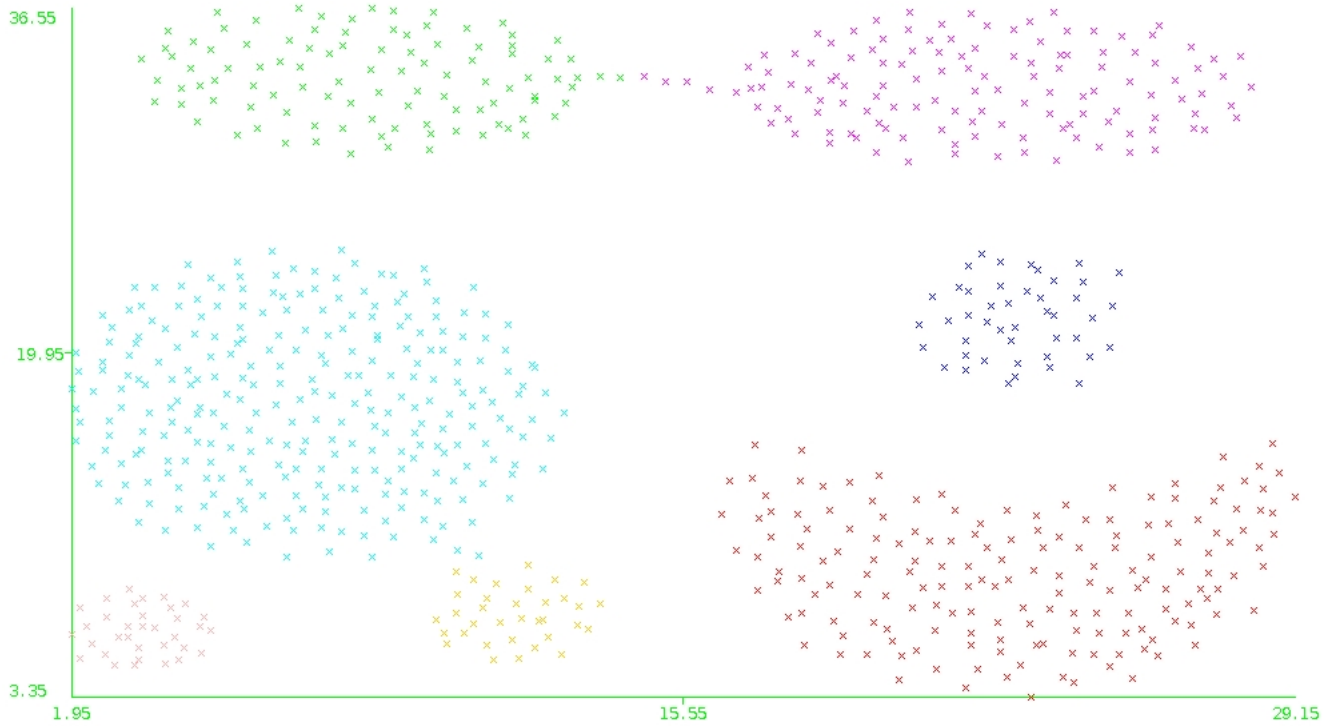

Figure 2: Aggregation

On this dataset, K-means will optimize the clusters and minimize the distortion. Performance of the K-Means algorithm is evaluated using Euclidean Distance measure. It is possible that the clusters that are elongated (as evident from figure) could form two seperated clusters in K-means algorithm. Intra-cluster similarity will be low and inter-cluster similarity will be high.
DBSCAN will allot data points to the same cluster if they are Density Connected. DBSCAN has been explained in detail in Section 1.4 of this report. In the above dataset, we observe a thin trail of datapoints between two clusters in the upper part of figure. DBSCAN will allot these two clusters as one cluster.
Single link will compute the distance between two closest points and fuse the elements into one cluster. On this dataset, the top two clusters joined by a thin chain of data points might be fused.
Complete link would choose pairs whose merge will have smallest diameter. Hence it is possible that large chunks of cluster points might get divided into several clusters.

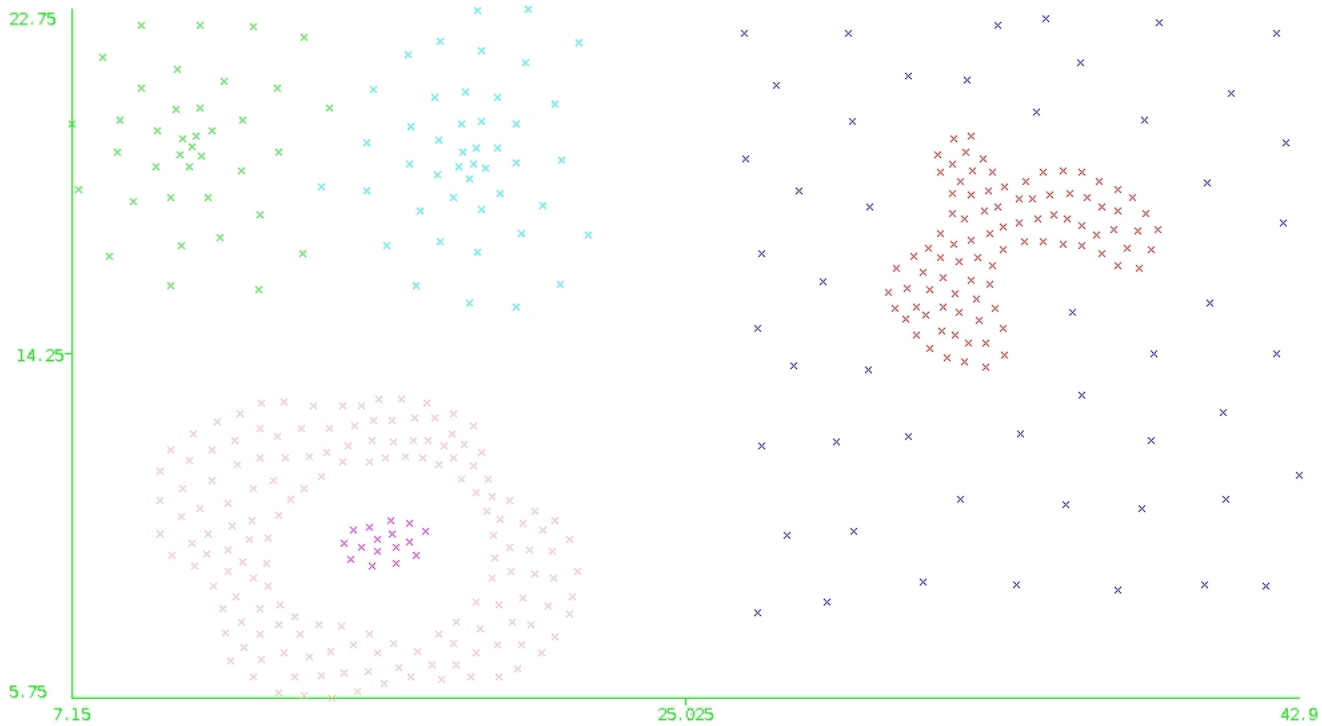

Figure 3: Compound

This is a very tricky dataset. For K-means, choosing the optimal threshold distance value becomes important. If a very small value is chosen, there will be no problem with the dense clusters, but for each of the sparse data points, there will be a new cluster. Number of clusters will not be optimal. If a large value is chosen, the two clusters which are nearly concentric will be considered as one whole cluster. Intra-cluster similarity and inter-cluster similarity both will be low.
DBSCAN will not be able to detect the cluster with sparse points. To detect the sparse points, epsilon and minpoints value should be high, which in turn would distort the structure of other clusters. Otherwise, the dense clusters would be easily identified.
Single link would perform poorly here. The sparse datapoints would easily be clustered wrongly along with the cluster concentric with them.
Complete link would choose pairs whose merge will have smallest diameter. Hence here it is not all possible for Complete Link to detect the concentric clusters properly.

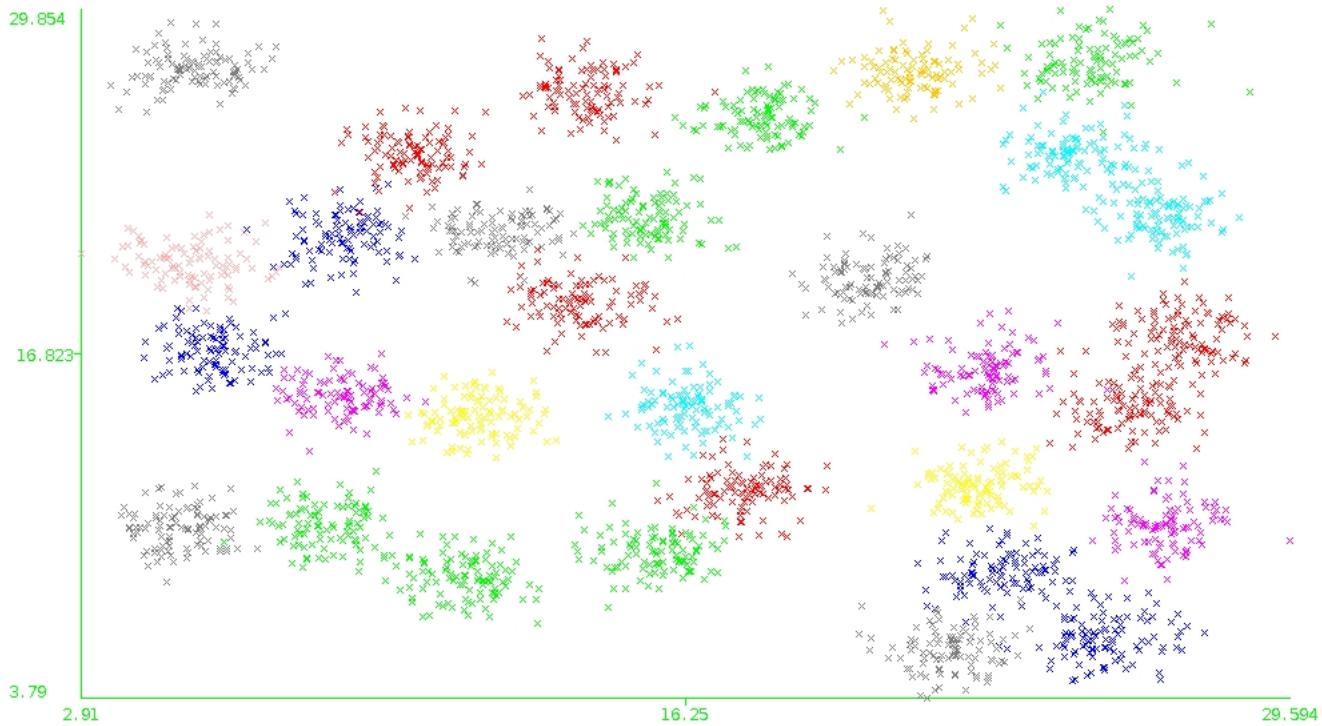

Figure 4: D31

Here, the clusters are very close to each other and very densely populated. K-means might not give correct result in one iteration. As number of clusters are 31 and centroids are initialized randomly, there is high probability for the cluster structure to get distorted. There might be improvements in K-means after certain number of iterations. Intra-cluster similarity and inter-cluster similarity both might be high.
It would be very difficult for DBSCAN to detect all the clusters. As the clusters are dense, most of the data points are density connected. Due to the sparse points in between the dense cluster points, the number of clusters detected by DBSCAN would be definitely less than 31.
Single link would compute distance between two nearest points. On this dataset, Single link would produce long chains of cluster.
Complete link would perform well. As smaller diameter is favourable for Complete Cluster, the clusters would almost be correctly identified.

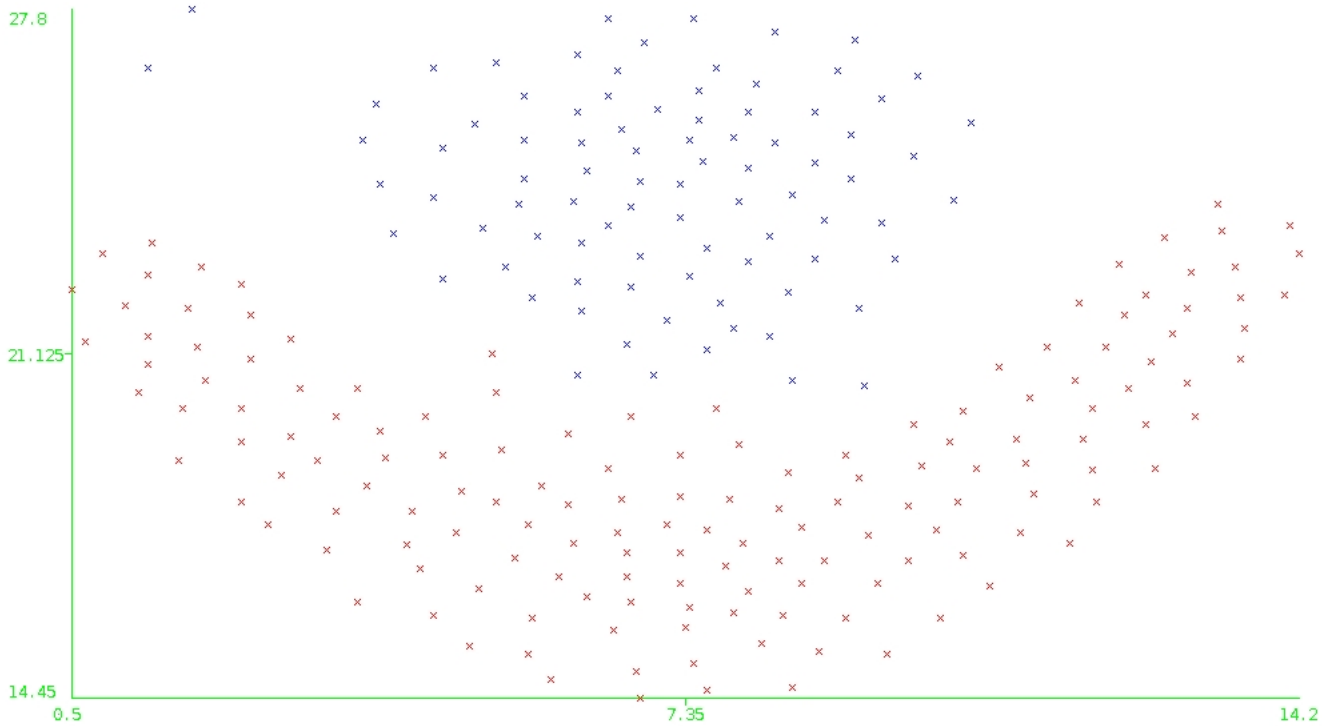

Figure 5: Flames

For K-means, the elongated cluster might get divided into more than one clusters. There seems to be two outliers here. The region around these outliers could be evaluated as one cluster. K-means will minimize the distortion but clusters will not be optimized. Intra-cluster similarity will be high and inter-cluster similarity will be low.
If observed carefuly, the Eucledian Distance between two neighbor data points is almost same in whole dataset. DBSCAN would not work for low values of epsilon and minpoints. It will allot more than two clusters. Even for higher values, I don't think DBSCAN would perform great. Leaving the two outliers, it would evaluate all the datapoints as one complete cluster as all the points are density reachable.
Single link on this dataset would produce very few layers. As distance between datapoints is not too much varying, the hierarchical cluster would have just a couple of layers.
Complete Link would distort the cluster structure. It is highly probable that Complete Link would divide the cluster such that half the data points are from one cluster and rest from another. In a sense it would try to divide the cluster from the middle (as an example) to reduce the diameter of cluster after merging the data points.

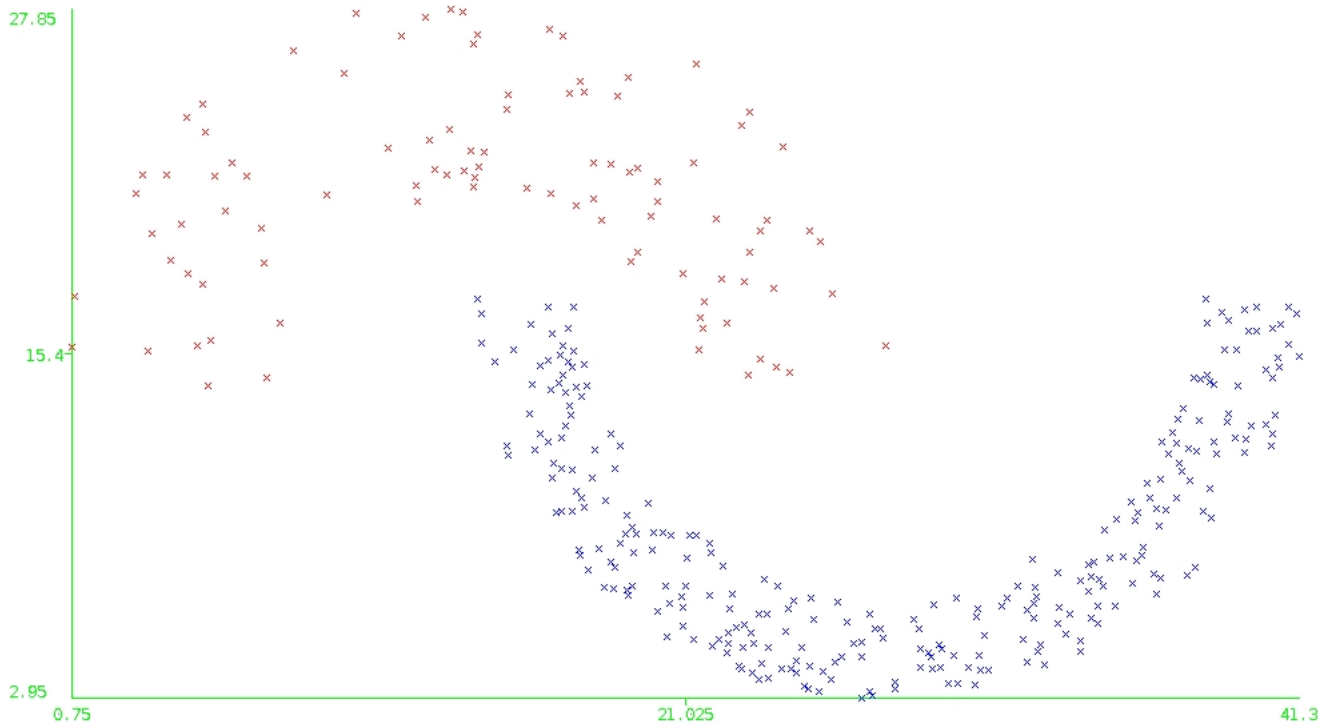

Figure 6: Jain

K-means have a bad relation with spiral-like datasets. Whatever you do, the clusters will never be optimized here. Due to the nearly overlapping region in the center, there is high probablity that these data points may fall into the distance threshold of the centroid of opposite cluster.
Spiral-type datasets are perfect examples to demonstrate how DBSCAN works. But here, one cluster has dense data points while other has sparse points. The one with dense points will get detected easily, while the other one will be divided into more than one clusters.
Single tends to focus on shortest distance. Due to this, it creates long chains of clusters. The dense cluster will be fused first. The cluster with sparse data points will be fused into several clusters, i.e. in hierarchical clustering, there will be more than two clusters.
Complete link would not be able to correctly identify clusters points where the spirals are trying to overlap(near the center).

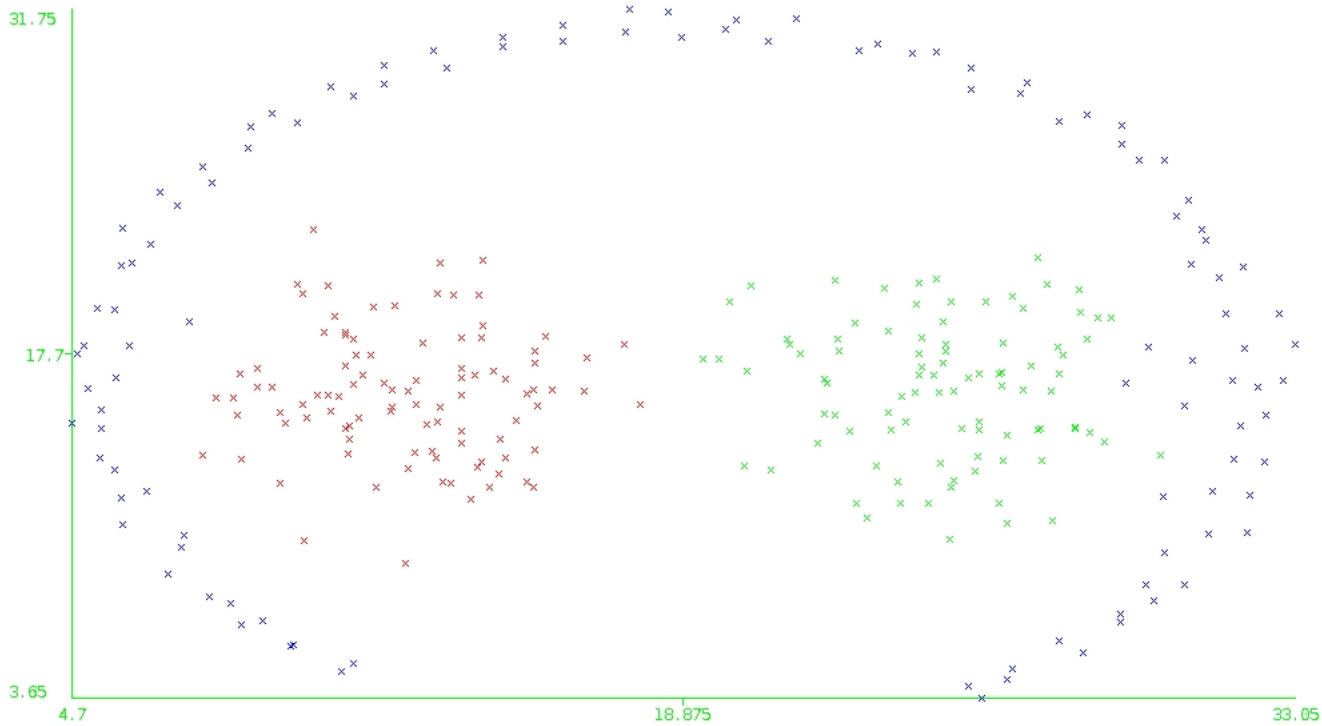

Figure 7: Path-based

Again, this dataset illustrates the concept of Density Connected and Density Reachable points. K-means would not be able to get the spiral type cluster correct.
DBSCAN would perform well here. There is a very high probabilty that all the clusters will be detected.
Single link will perform well and all the clusters will be identified correctly with a high probability.
This would give a tuff time to Complete Link. As complete link tries to minimize the diameter of cluster, the huge spiral-shaped cluster data points would be merged with the other nearest data points.

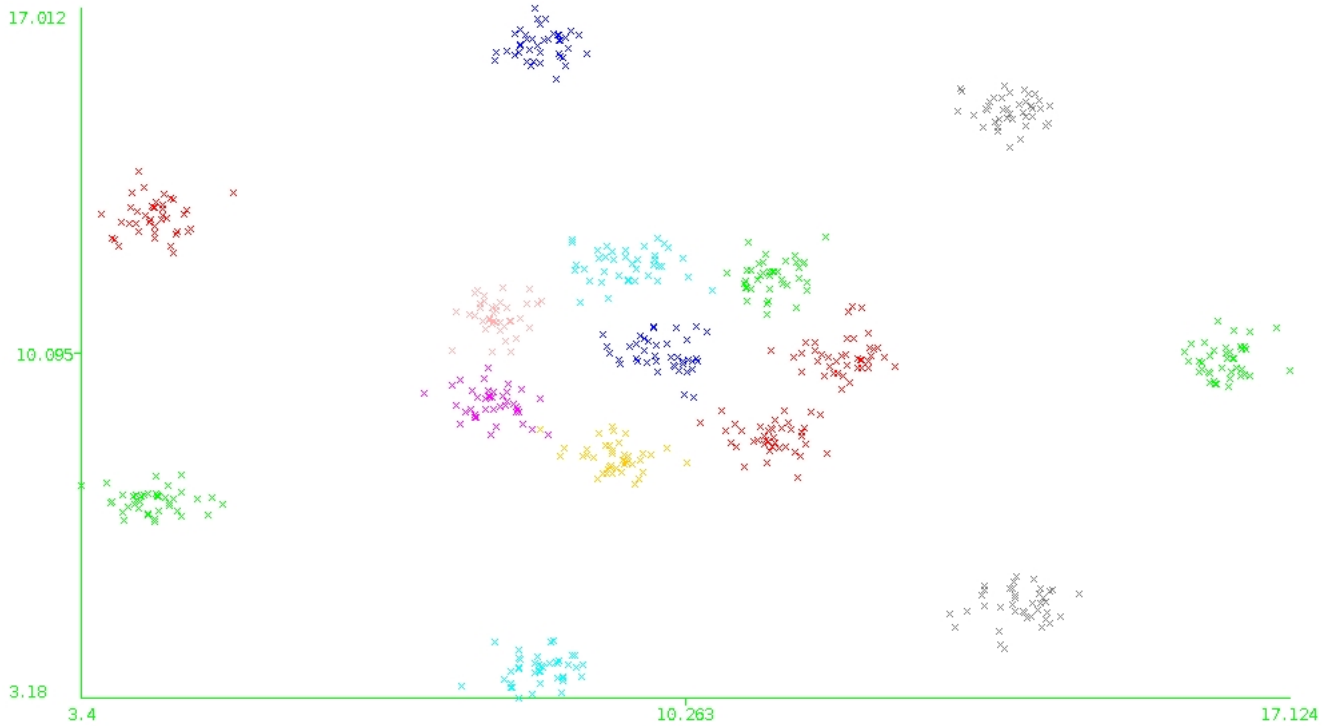

Figure 8: R15

This dataset would give a cakewalk to K-means, provided the distance threshold is kept low. There are few clusters spread very far apart, which would easilty get clustered. While the ones in center, if observed carefully, are dense at their respective centroids. Although the data points on outline of each clusters might get wrongly clustered, but K-means will optimized the number of clusters and minimize the distortion. Intra-cluster similarity will be high and inter-cluster similarity will be low.
DBSCAN would detect clusters situated along the outermost concentric ring. But for the centre cluster and the clusters surrounding it, it seems there are points which can make the cluster data points density reachable. Hence DBSCAN would make the clusters along the inner concentric ring as one complete cluster and maybe include the centre one as well.
Single link would perform well. Except for the data points on the inner concentric ring, which might get fused into one whole cluster at first iteration, the outer clusters will be identified correctly.
Complete link would perform well and all clusters will be detected correctly.

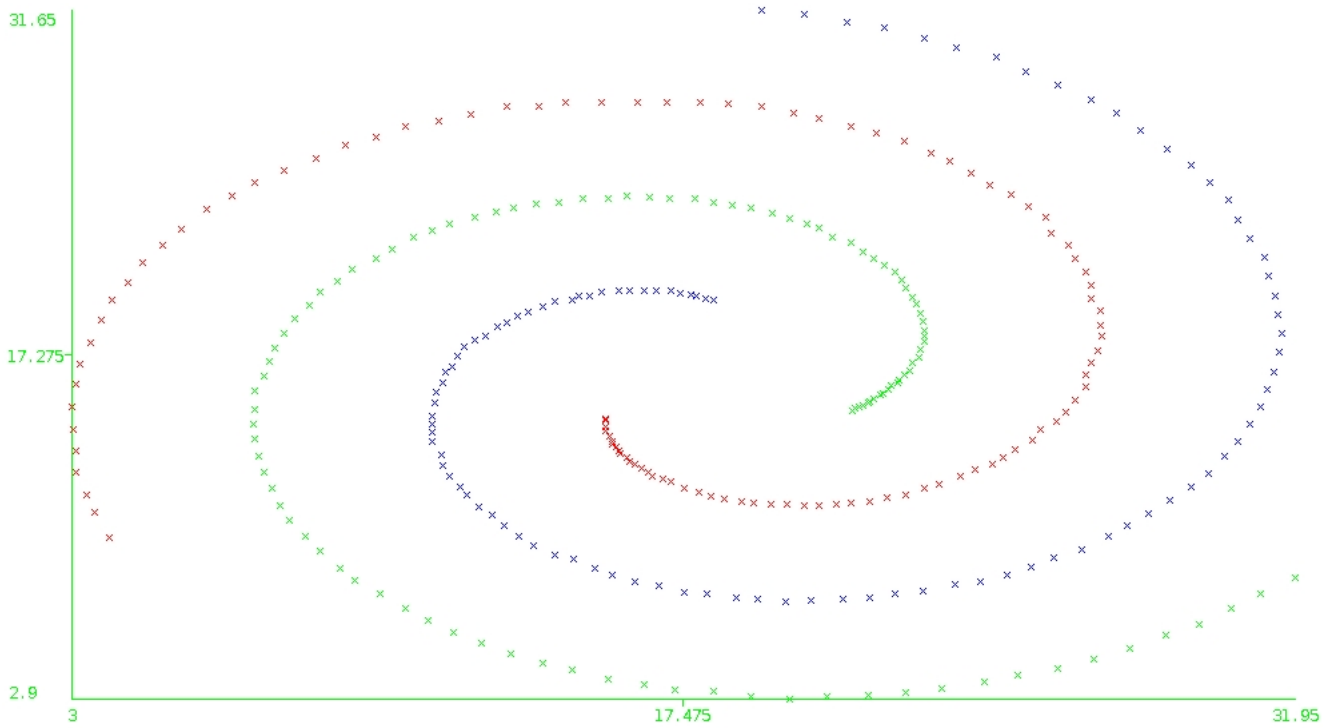

Figure 9: Spiral

Spiral again! K-means will definitely fail. Do any number of iterations, K-means would not be able to detect the clusters. Intra-cluster similarity and inter-cluster similarity both will be high.
Perfect dataset for DBSCAN. This will give a cakewalk to DBSCAN. All the clusters will be dectected accurately without any kind of error.
Single link would perform well and all clusters would be identified correctly.
Spiral-shaped clusters do not go well with Complete Link. Complete link would try to shape them in spherical clusters.

1.3 K-means

K-means was performed on R15 dataset. In 'Cluster' tab, Clusterer is chosen to be 'SimpleKMeans'. In 'Ignore attributes', 'class' attribute is ignored while performing K-means. Cluster Mode is set to 'Training Set'. The model is stored as r15_kmeans_k_8. The Cluster Purity for k=8 was found to be 0.53. Cluster Purity for k varying from 1 to 20 is stored in file named r15_kmeans_cpurity.txt.

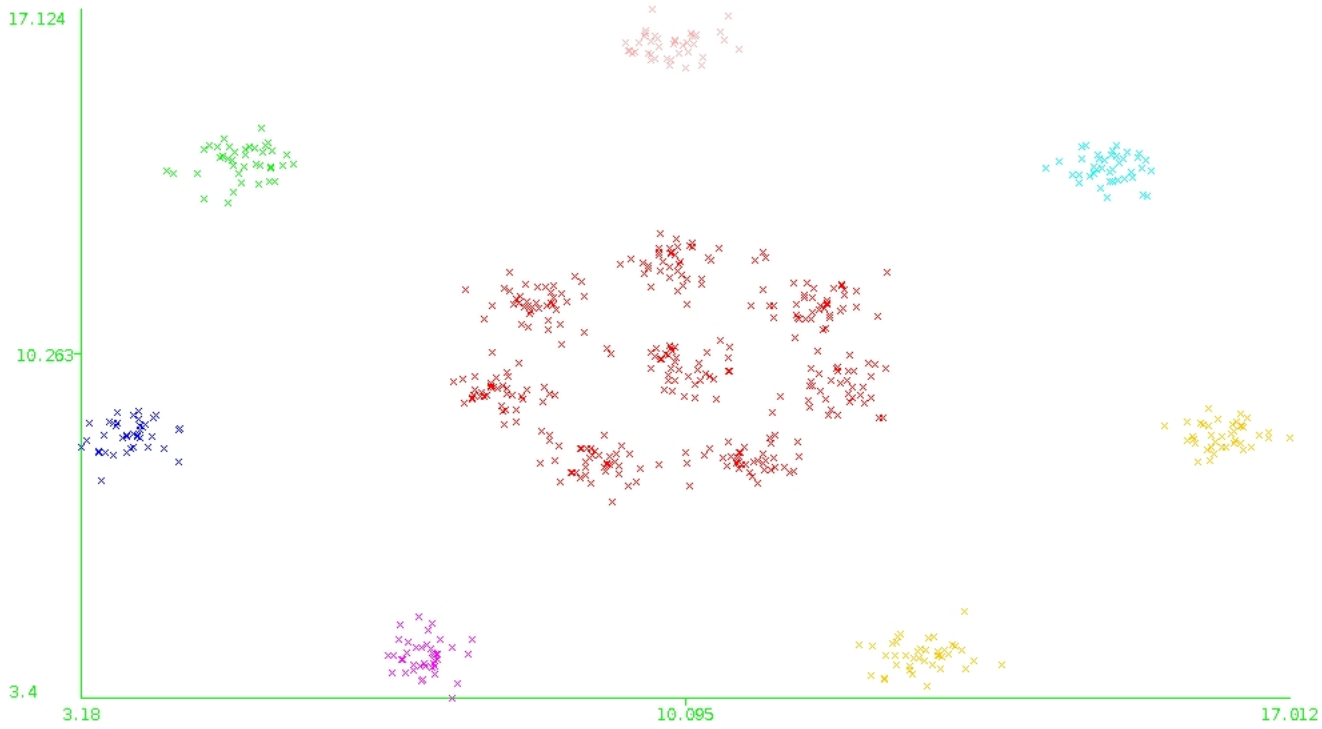

Figure 10: KMeans on R15 with k=8

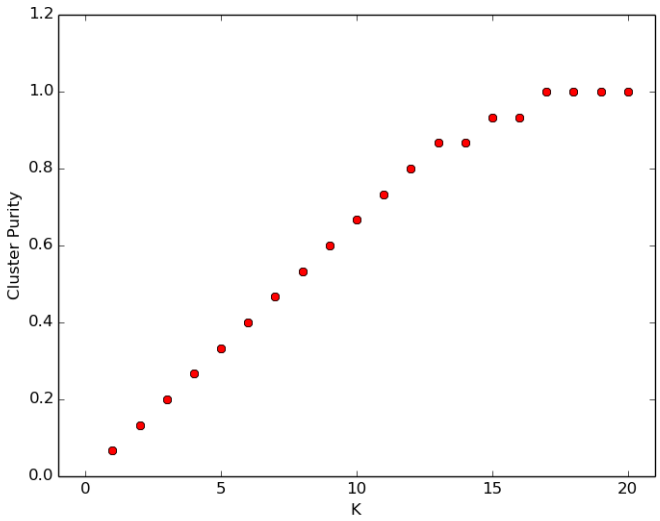

Figure 11: Cluster Purity for KMeans on R15, K varying from 1 to 20

1.4 DBSCAN on Jain dataset

DBSCAN is a clustering algorithm which performs clustering based on density. ϵ and *minpoints* are metrics which signify *minpoints* number of points in a cirle with ϵ radius around a point. If a point has equal or more number of data points than *minpoints* around its ϵ radius circle, then it is considered as core point. Two points are density reachable if the point is reachable from other by traversing only throught core points. Two points are density connected if they are density reachable by a single core point. Two points belong to the same cluster if they are density connected. After varying the parameters, it was found that keeping ϵ constant, the number of unclustered instances increase as we increase the *minpoints*.

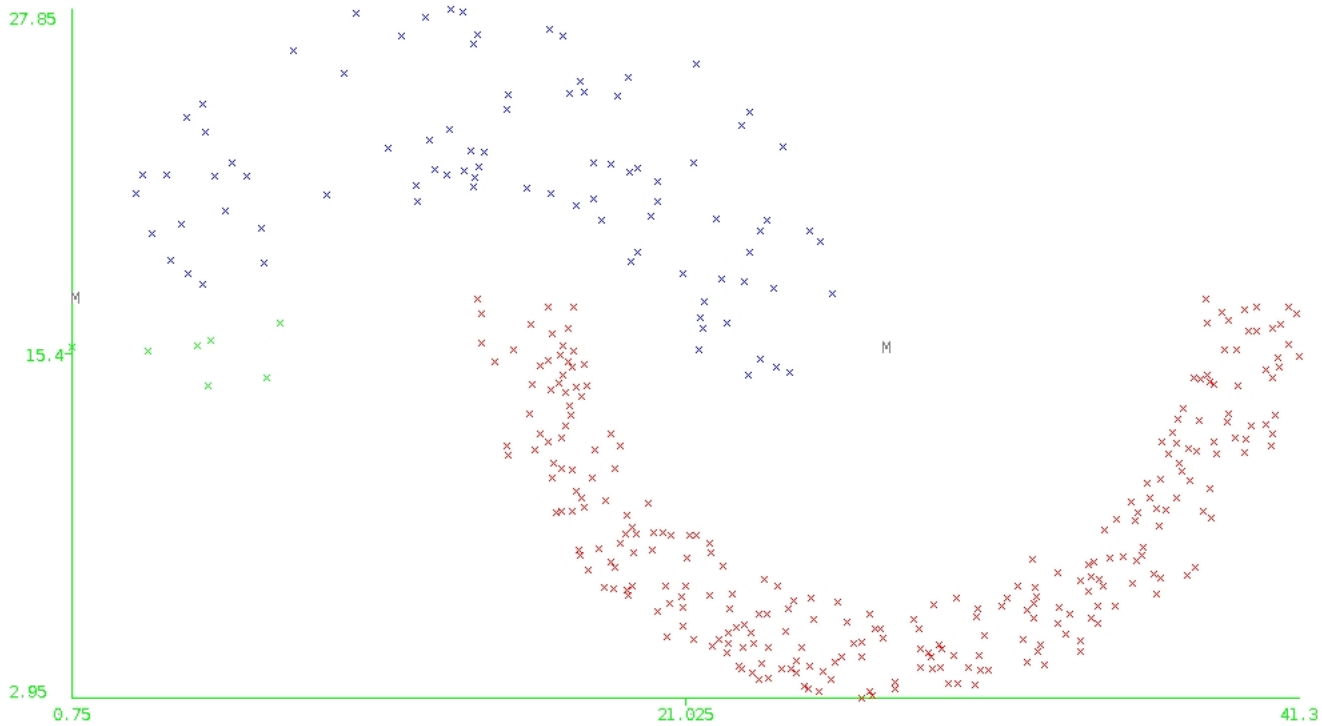

Figure 12: DBSCAN on Jain Dataset, epsilon=0.081, minpoints=4, Cluster Purity = 1.0

Minpoints and epsilon are varied with respect to the optimal points as mentioned in the above figure.

Keeping ϵ=0.081, varying minpoints from 1 to 10, it was found that the Cluster Purity remained constant to 1.0.
Keeping minpoints=4, varying ϵ from 0.05 to 0.09, in steps of 0.005 it was found that from 0.05 to 0.08, Cluster Purity remained constant at 1.0. Cluster Purity at ϵ=0.085 was 0.745 and at ϵ=0.09 was 0.742. The ARFF files for each of the runs are stored in folder /data/jain_dbscan.

1.5 Path-based, Spiral and Flames datasets

Cluster Purity is used as a performance measure for the datasets. For each of the datasets, metrics are provided along with the figures below.

1.5.1 DBSCAN

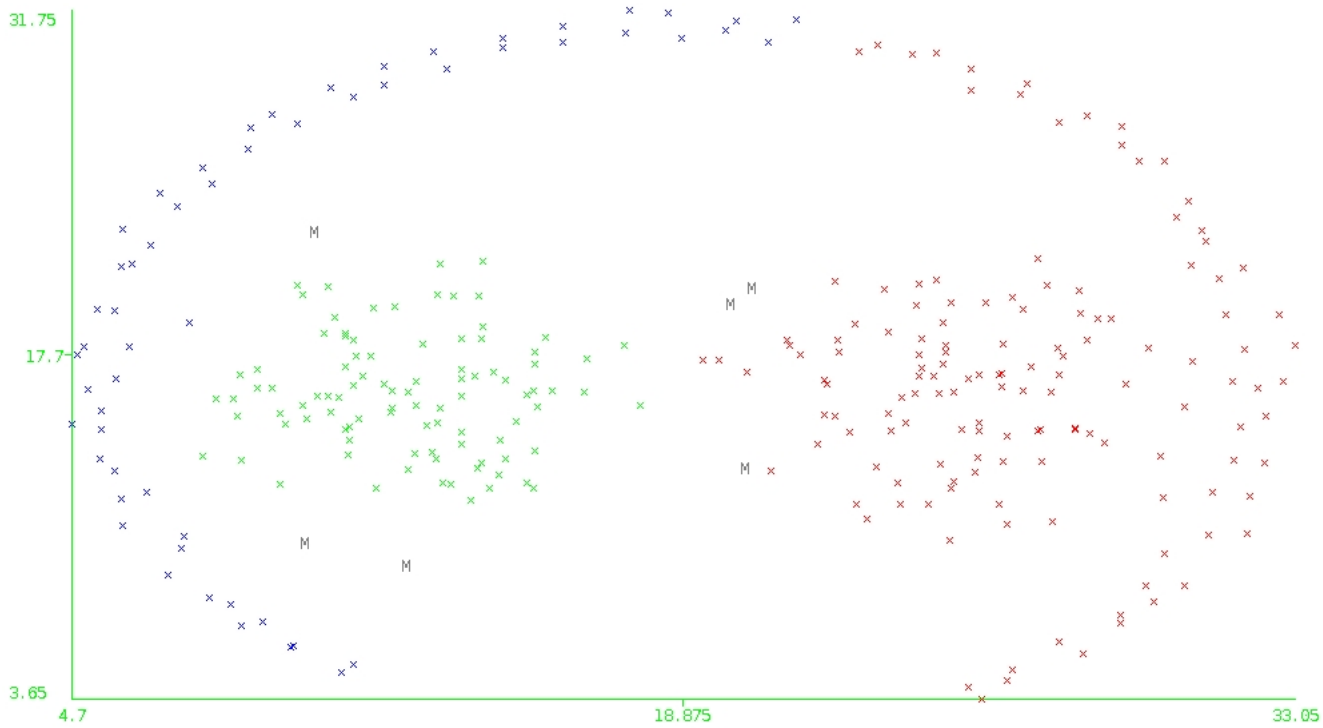

Figure 13: DBSCAN on Path-based Dataset, epsilon=0.065, minpoints=4, Cluster Purity = 0.813

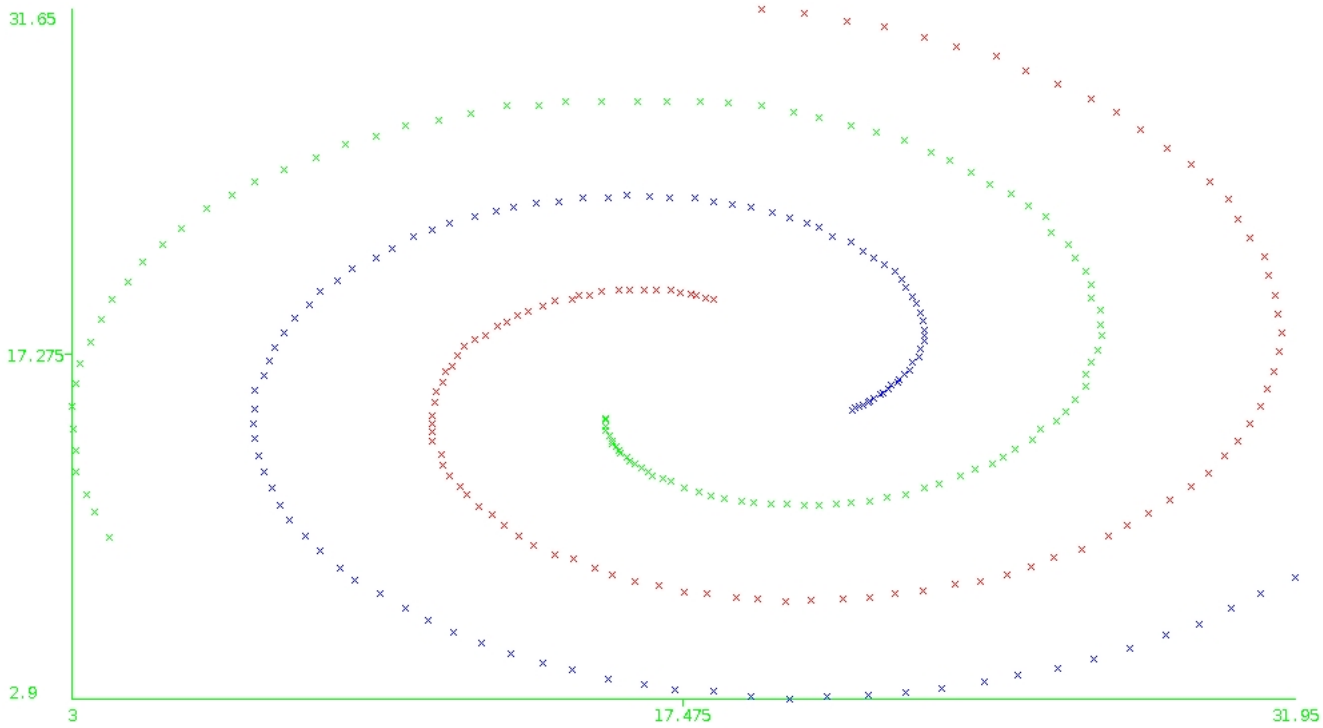

Figure 14: DBSCAN on Spiral Dataset, epsilon=0.065, minpoints=2, Cluster Purity = 1.0

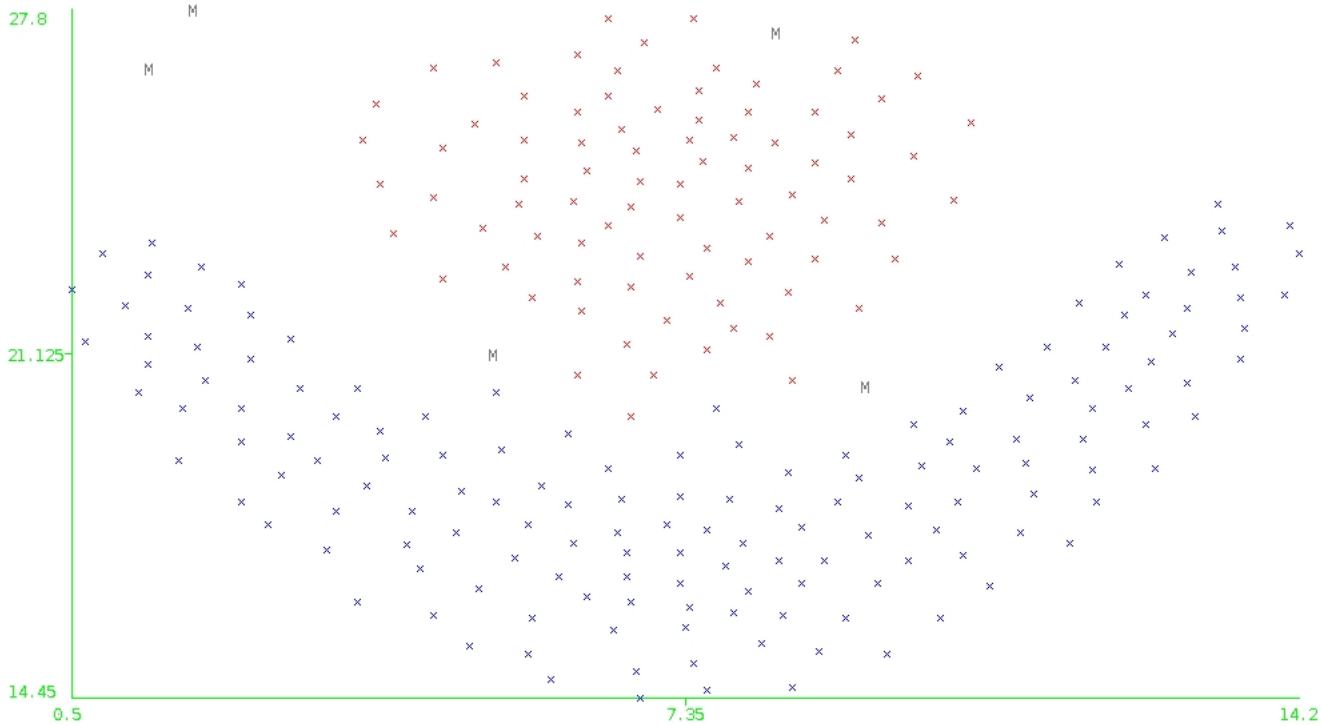

Figure 15: DBSCAN on Flames Dataset, epsilon=0.0685, minpoints=4, Cluster Purity = 0.991

1.5.2 Hierarchical

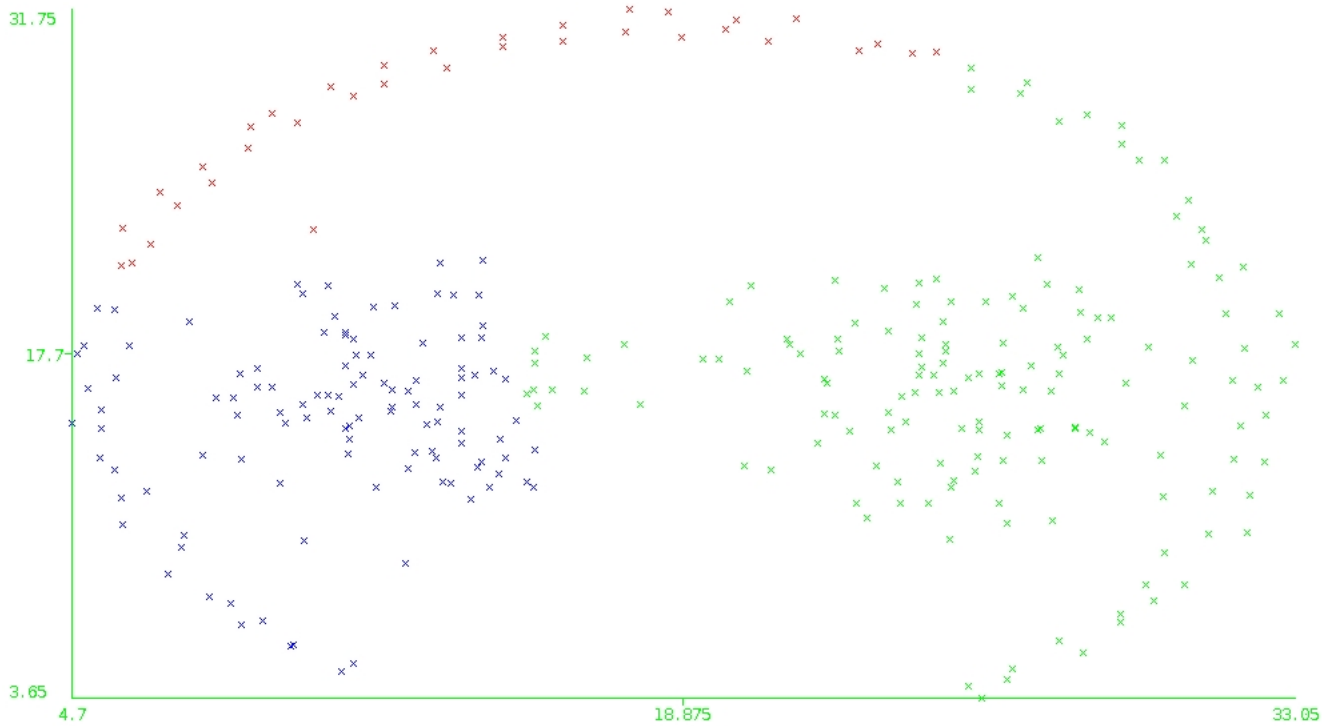

Figure 16: Hierarchical Clustering with COMPLETE LINK on Path-based Dataset, Cluster Purity = 0.706

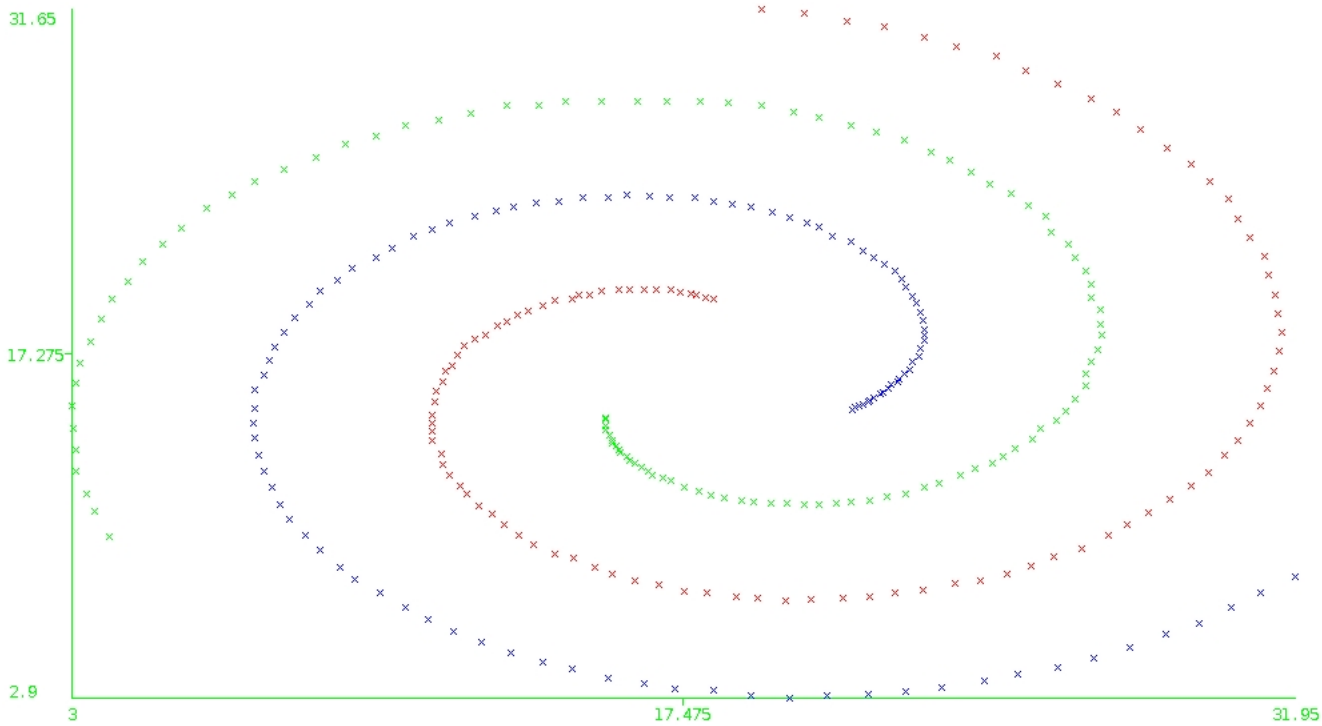

Figure 17: Hierarchical Clustering with SINGLE LINK on Spiral Dataset, Cluster Purity = 1.0

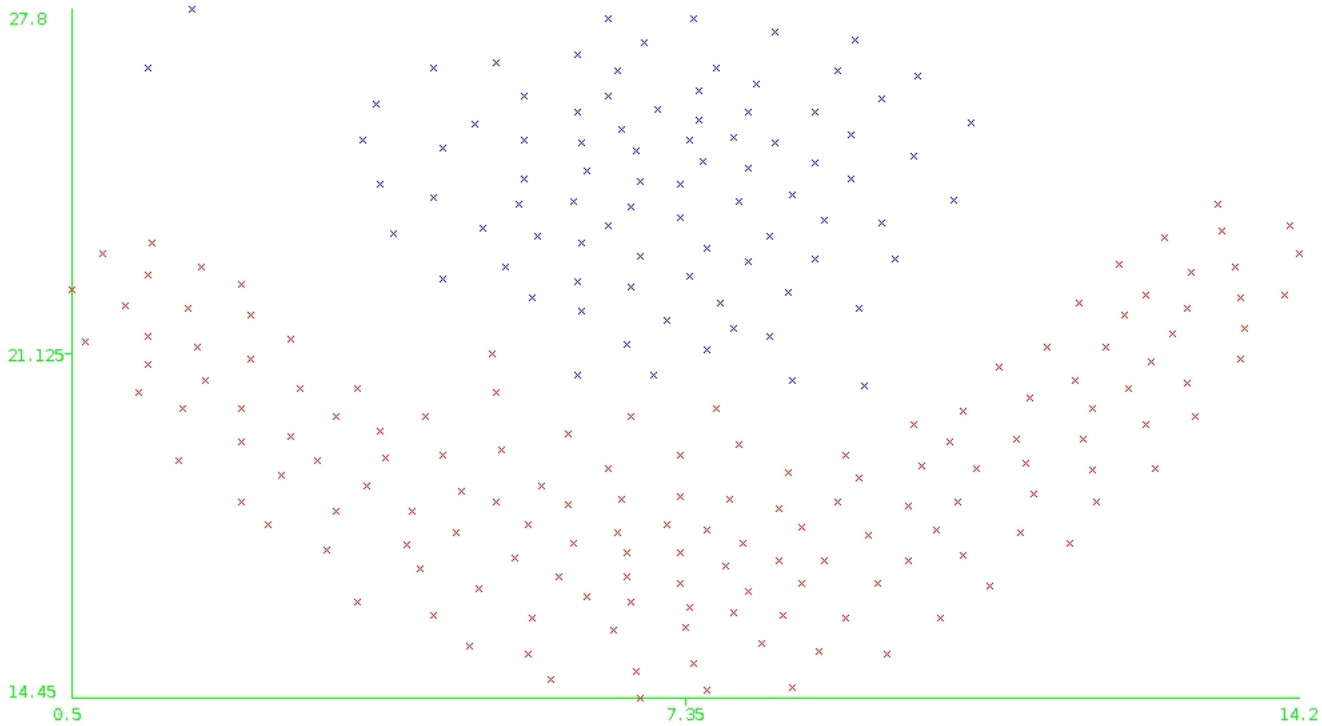

Figure 18: Hierarchical Clustering with WARD LINK on Flames Dataset, Cluster Purity = 1.0

1.6 K-means on D31

K-means was performed on D31 with k=31. Cluster Purity was found to be 0.849.

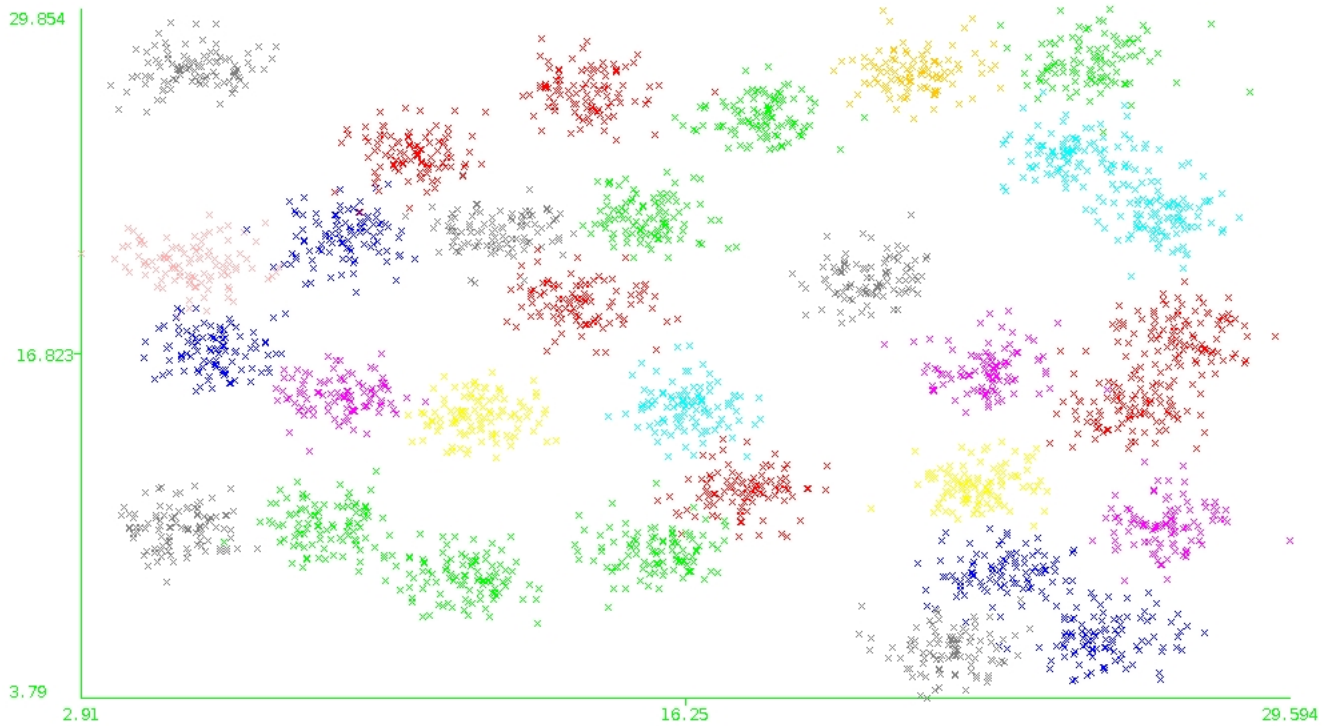

Figure 19: D31 dataset

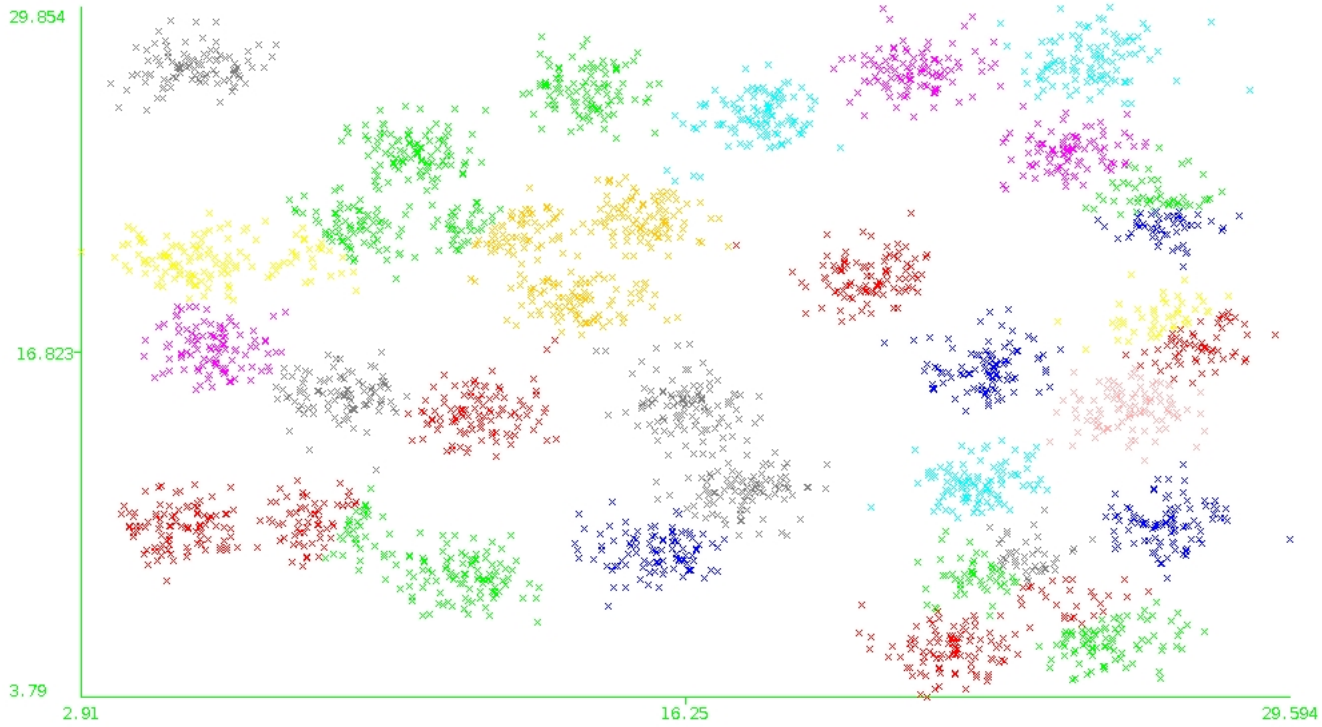

Figure 20: KMeans on D31, k=31

As evident from above, it is not possible to recover the clusters by using k=31. Increasing k alone would not help. This is a very densely populated data. Sampling several times and running K-means with increasing k, would give better results. Shown below, are plots of K-means on D31, for k=32, 35, 40, 62, without sampling. Cross-validation would optimize the number of clusters.

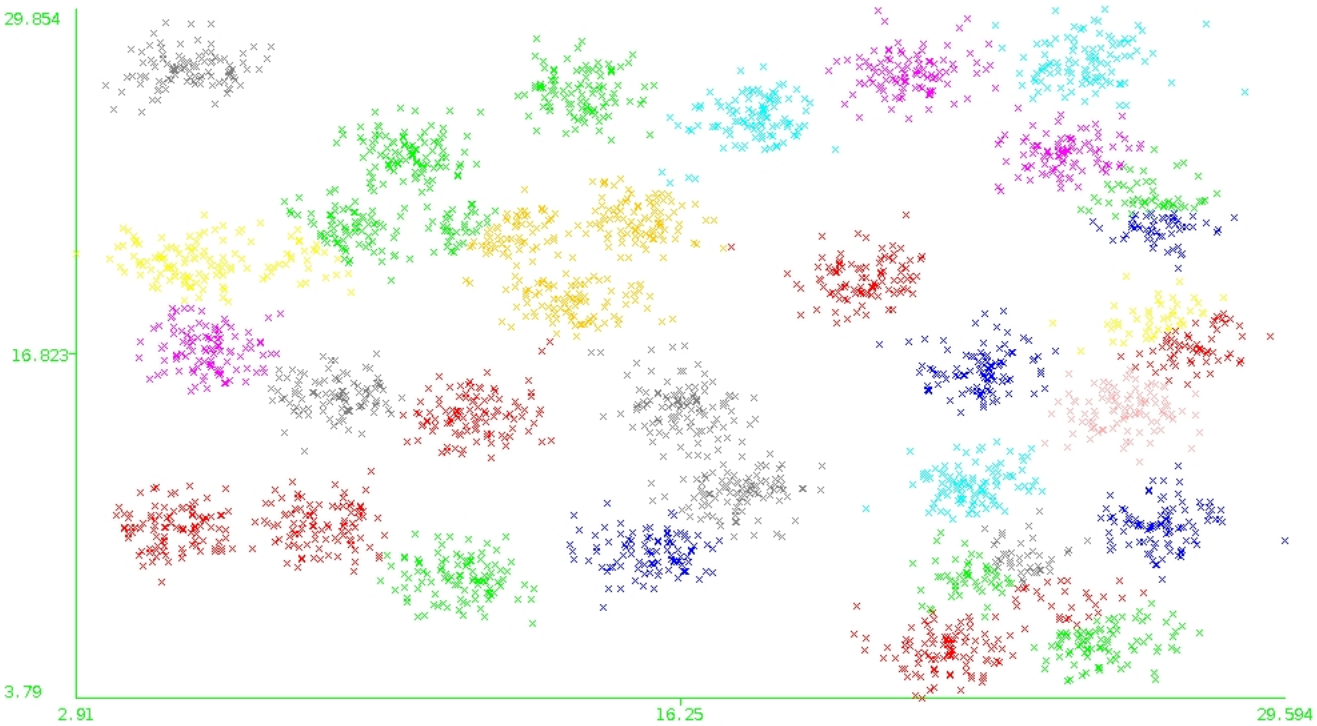

Figure 21: KMeans on D31, k=32

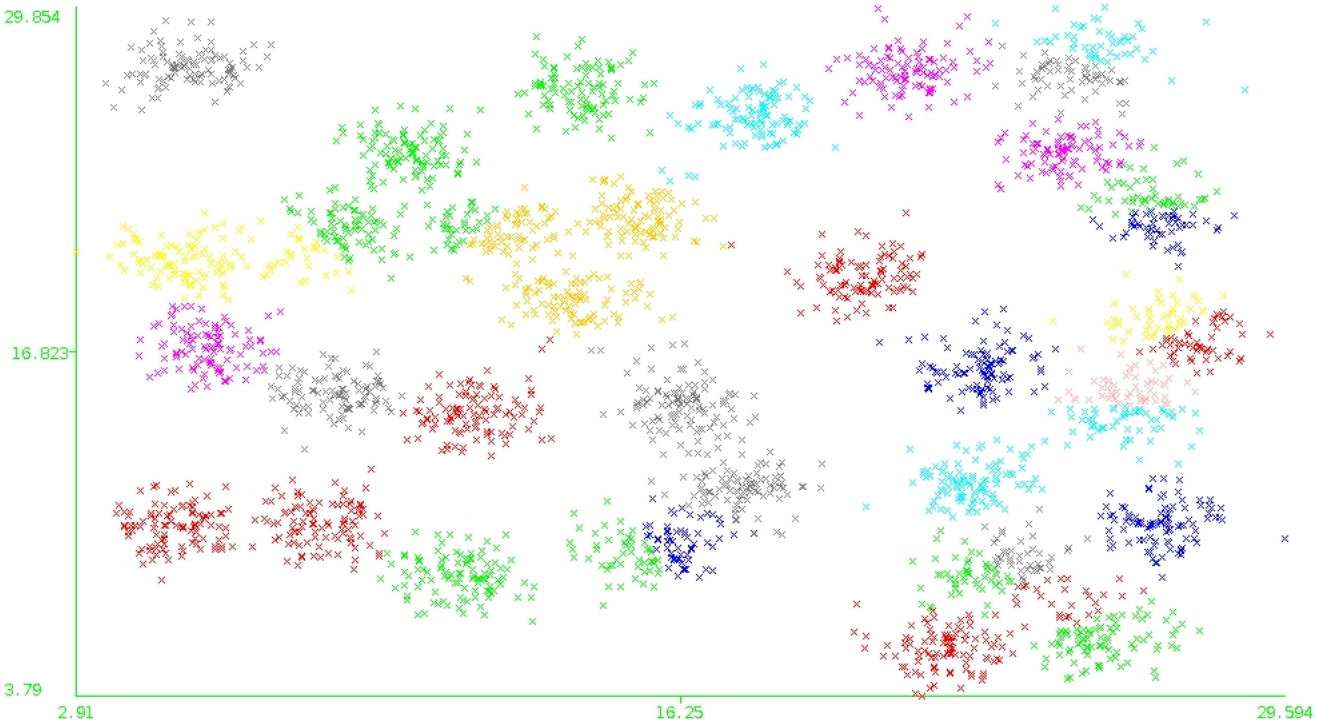

Figure 22: KMeans on D31, k=35

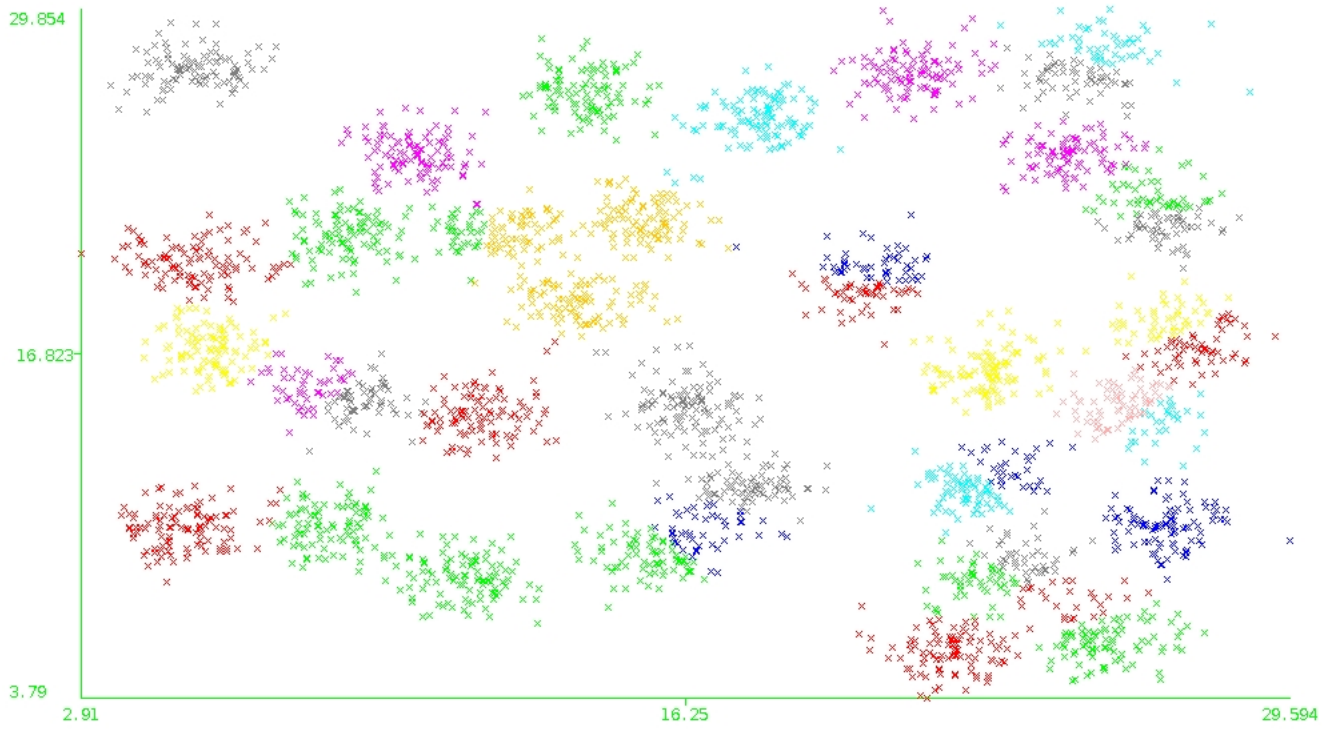

Figure 23: KMeans on D31, k=40

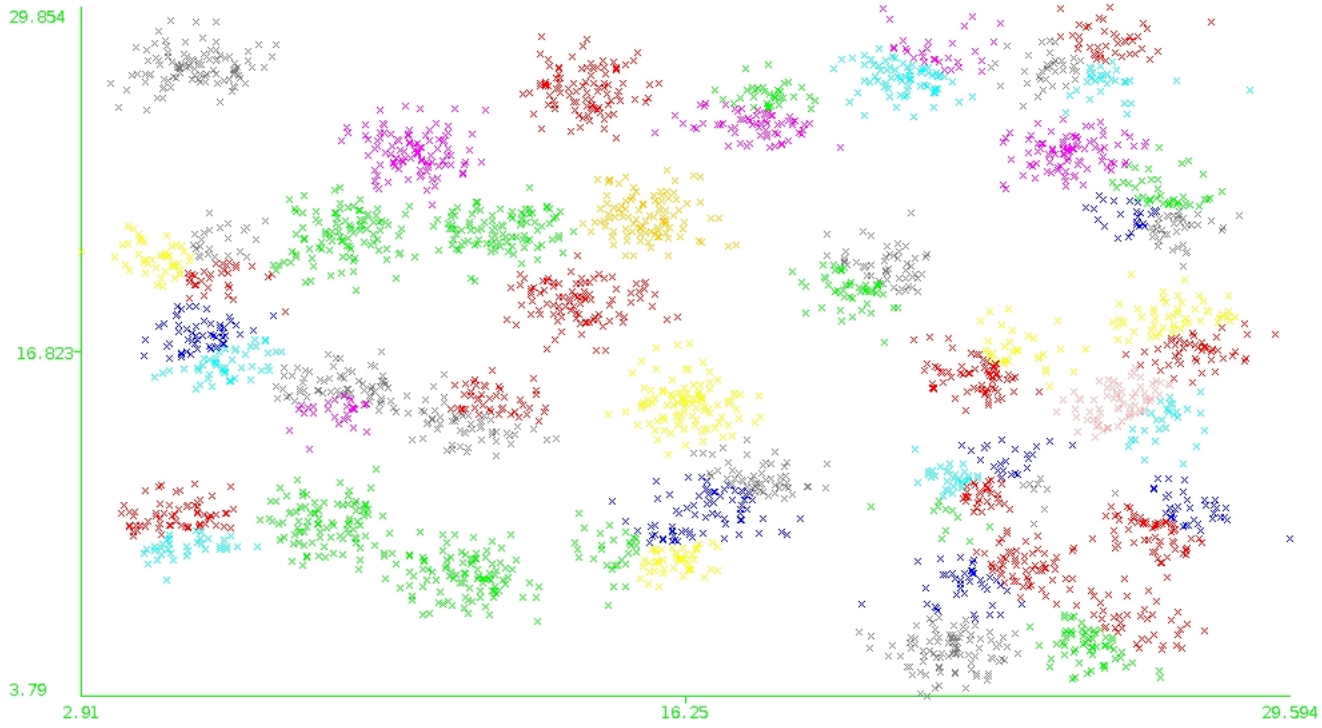

Figure 24: KMeans on D31, k=62

1.6.1 DBSCAN on D31

DBSCAN performs better than KMeans on D31 dataset. Setting ϵ=0.0173 and *minpoints*=3. Initially it looked like K-means would outperform DBSCAN, but with really low value of ϵ and corresponding *minpoints*, following plot is obtained.

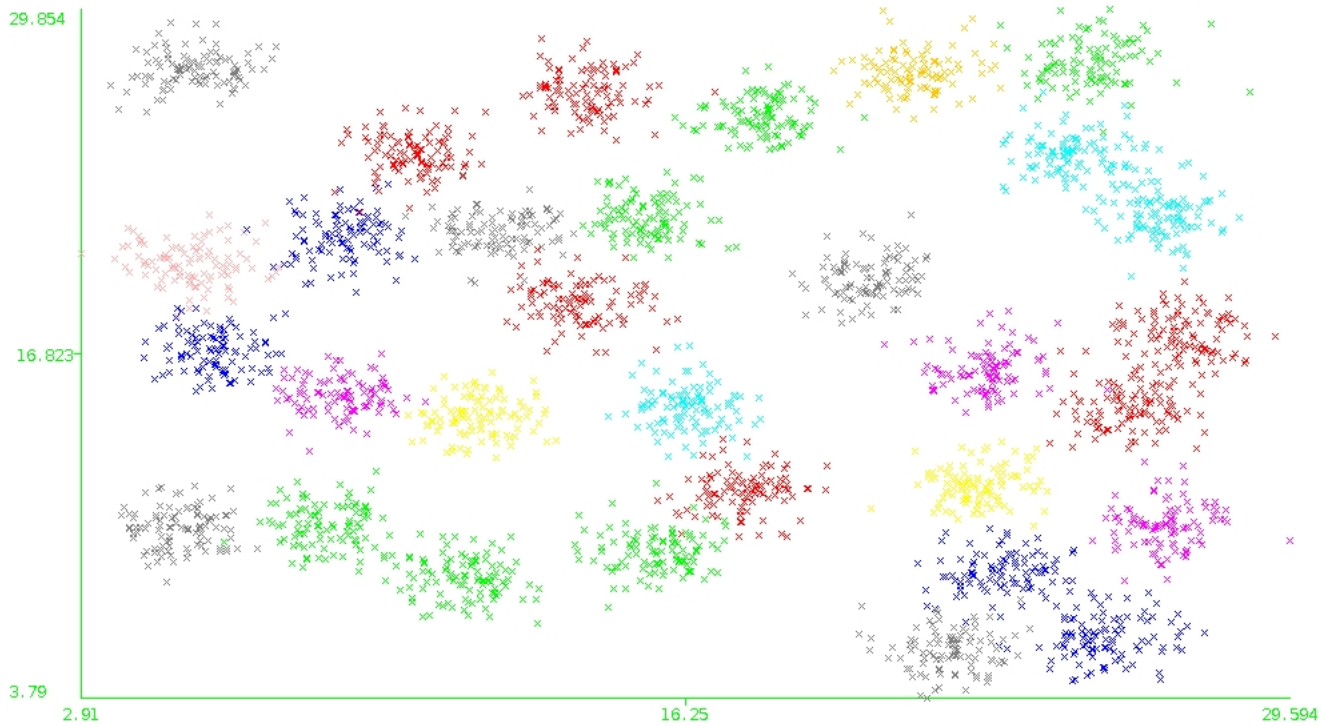

Figure 25: DBSCAN on D31, ϵ=0.0173, *minpoints*=3

1.6.2 Hierarchical Clustering with Wards Linkage

Hierarchical Clustering with Wards Linkage is performed on D31 dataset. Number of clusters was set to 31. It was observed that Hierarchical clustering took a longer time comparatively. Comparing with the original dataset, it is observed that the clusters are nearly optimized. It seems that some of the original clusters are seperated into two clusters. This prevented the clusters to get distorted, in the sense that all clusters are of almost the same size and density. The model is saved as *d31_hierarchical_wards.model*. Following visualization is obtained:

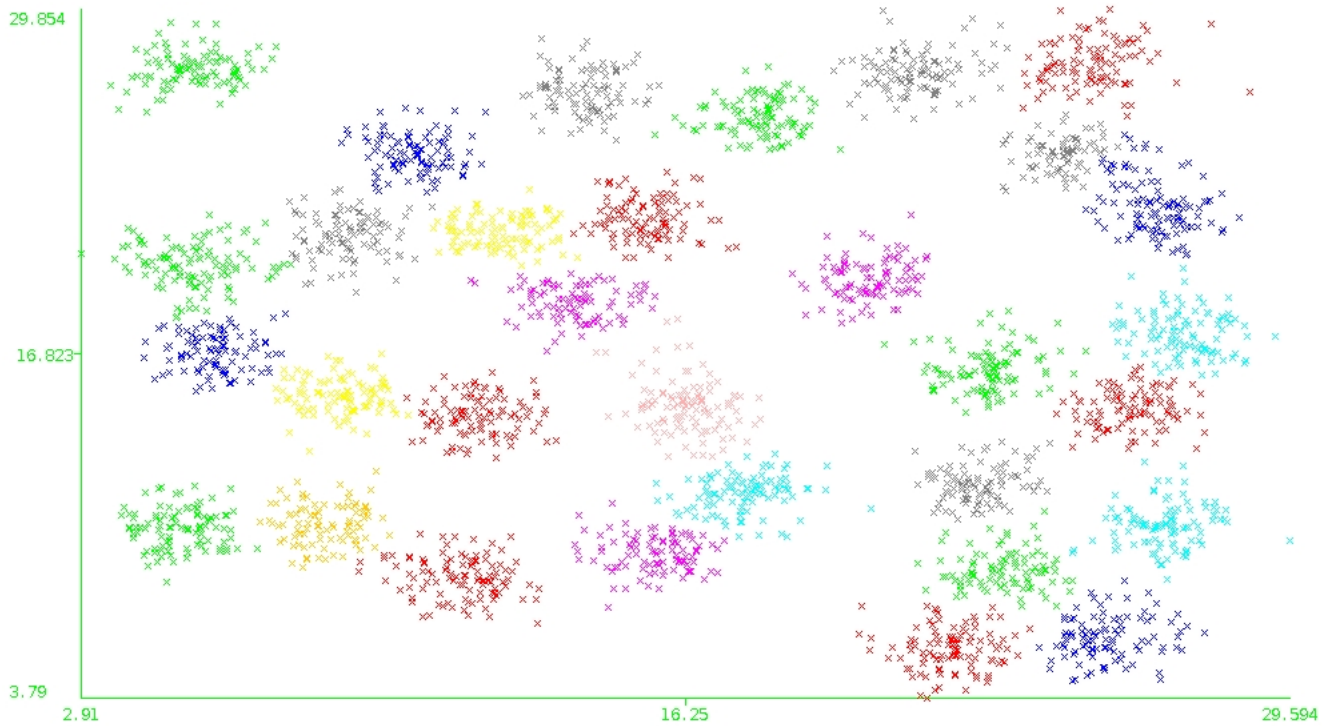

Figure 26: Hierarchical with Wards Linkage on D31, Cluster Purity=0.963

2 Decision Trees

2.1 ARFF format

The data, converted to ARFF (Attribute Relation File Format) is saved as mushroom_train.arff and mushroom_test.arff.

2.2 J48

After running J48 algorithm on train data, the precision, recall and f1-measure were all found to be 1.0 after tested on the test data. The model is saved as j48_model.model. Following tree is visualized:

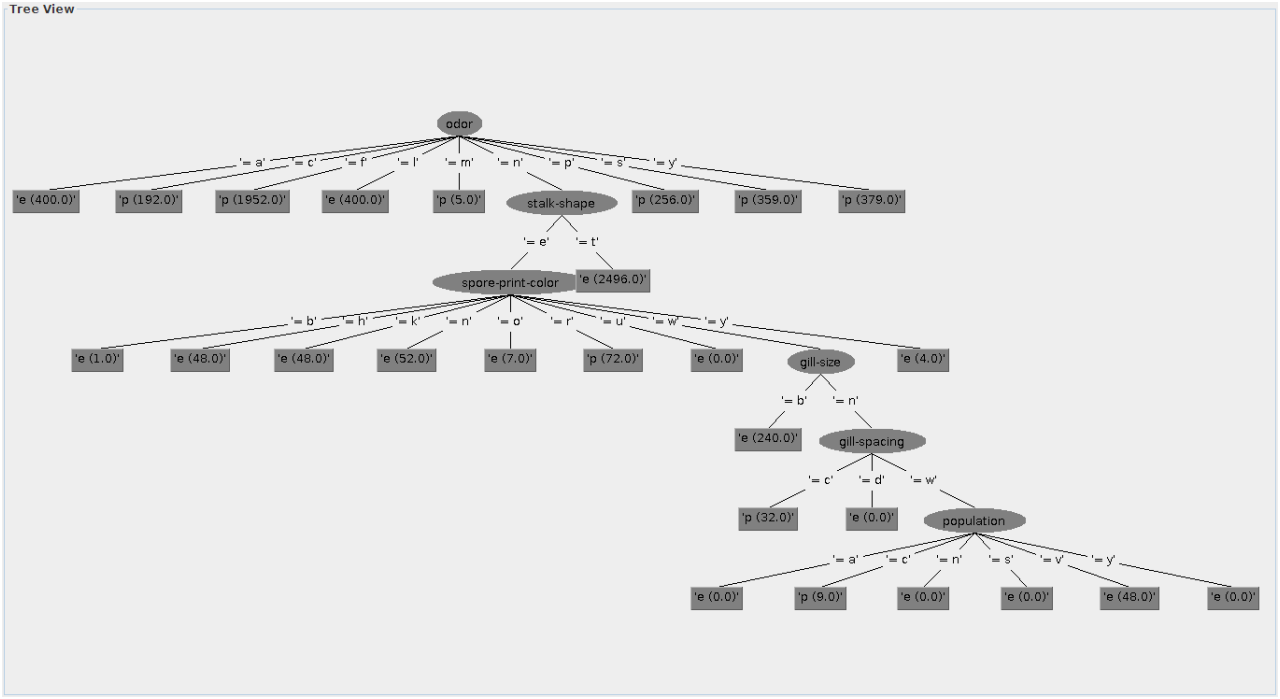

Figure 27: J48 Decision tree on Mushroom data

Initially, the *MinNumObj* was set to 2. While increasing the value gradually to about 25, it is observed that the error is increasing. *MinNumObj* basically sets the minimum number of instances per leaf. It guarantees that at each split at least 2 of the brances will have the *MinNumObj* number of instances. Branches are a way to seperate out data. Separating one instance from 100 instances doesn't give much information, so we set a minimum amount of separation. Minimum number of instances per leaf is better thought of as "the minimum amount of data separation per branching", in the case of multiway trees.

Reduced error pruning is a post-pruning method that uses a hold-out set (a fraction of the training data) for making pruning decisions. So less data is used to determine the structure of the tree than the other pruning methods. However, once the structure has been learned, the full training data can be back-fitted against the structure in order to determine the the node and leaf class distributions. The model with Reduced Error Pruning is saved as j48_model_rep.model

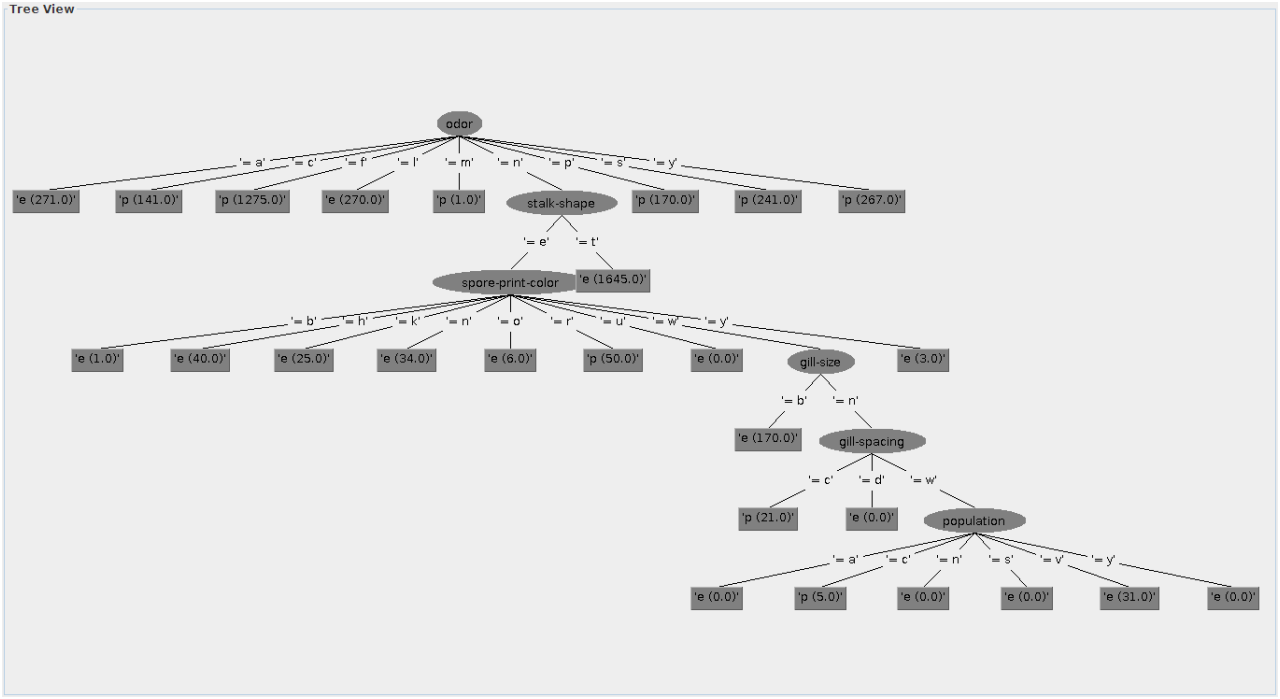

Figure 28: J48 Decision tree on Mushroom data with Reduced Error Pruning

2.3 Best Features

Using 'Attribute Evaluator' available in Weka, it is found that *odor, stalk-surface-above-ring* best describe the classes. The Evaluation Method was CfsSubsetEval and Search Method was BestFirst, with forward search direction.

2.4 Model

The Decision Tree model named as j48.model is attached alongwith this report.