Programming Assignment 3

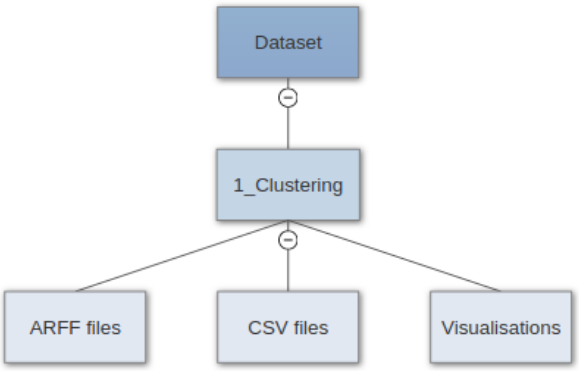
EE15B025 : Ganga Meghanath
November 11, 2017

CLUSTERING

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups.Eight 2-dimensional datasets have been provided for clustering: Aggregation, Compound, Path-based, Spiral, D31, R15, Jain, Flames. First two columns are the features and the third column is the class label. Various analysis and visualisations have been conducted on the same and the observations and results recorded.

1.CONVERSION OF DATA INTO ARFF FORMAT

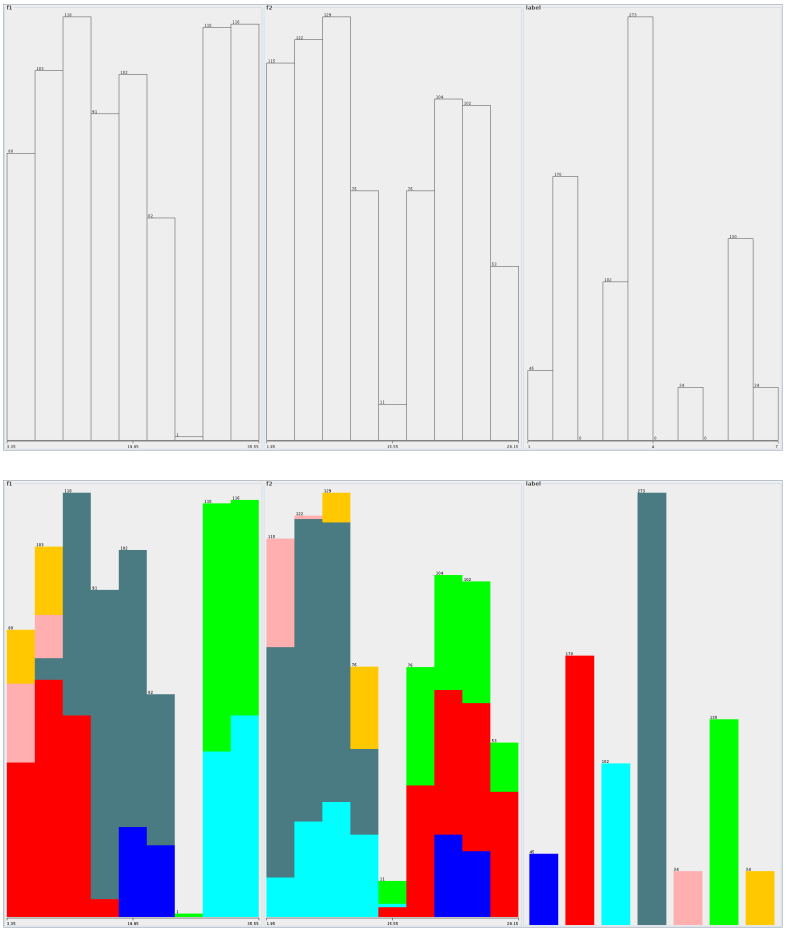
The conversion has been done using Weka. The python code for the manual conversion is present in Code->q1->arff.py. The folder structure is depicted as follows :



2. VISUALISATION

All the visualisations have been done using Weka. The histograms have been color-coded according to the class-labels (last column). The first two columns are features f1 and f2 respectively.

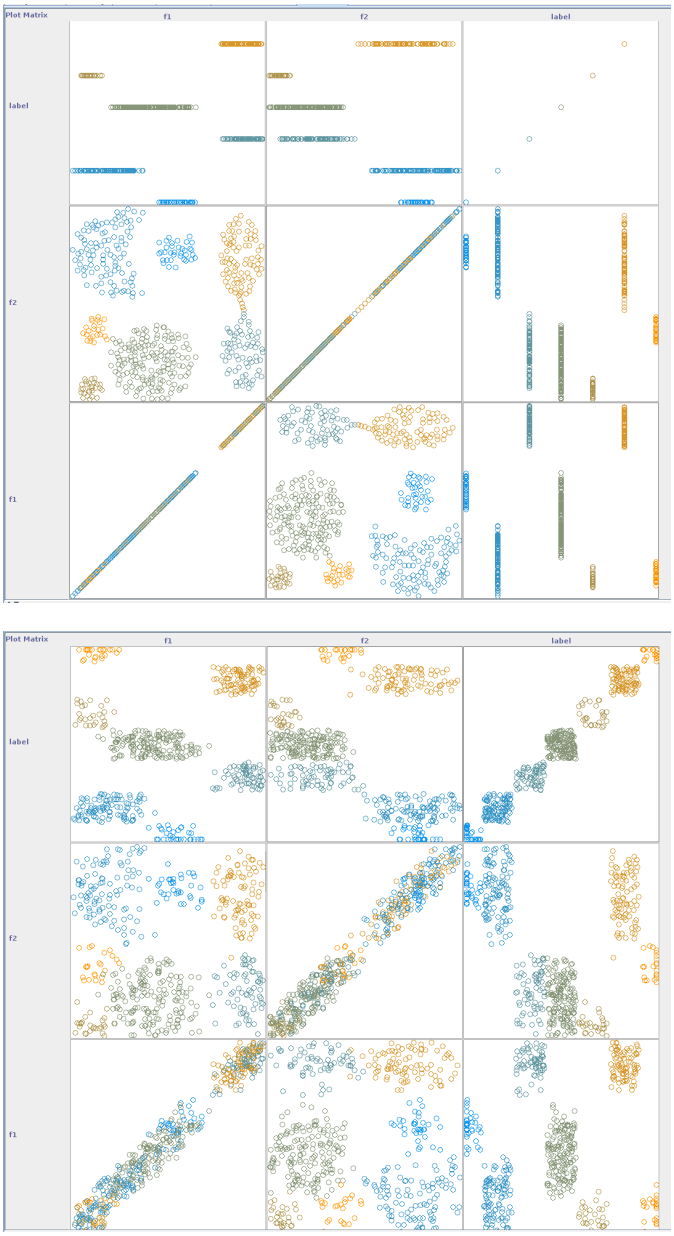
(I) AGGREGATION



Classwise distribution of datapoints along features

As we can see below, the clusters are well enough space seperated and hence K-means can converge to give optimum clusters that can be used for classification. The value of k has to chosen appropriately.

Clusters belonging to different classes have dense links between them. As a result, for very small ϵ and large MinPts, we probably would get bad results. But due to the interclass dense connection, it might cluster them together even for optimum ϵ and MinPts.

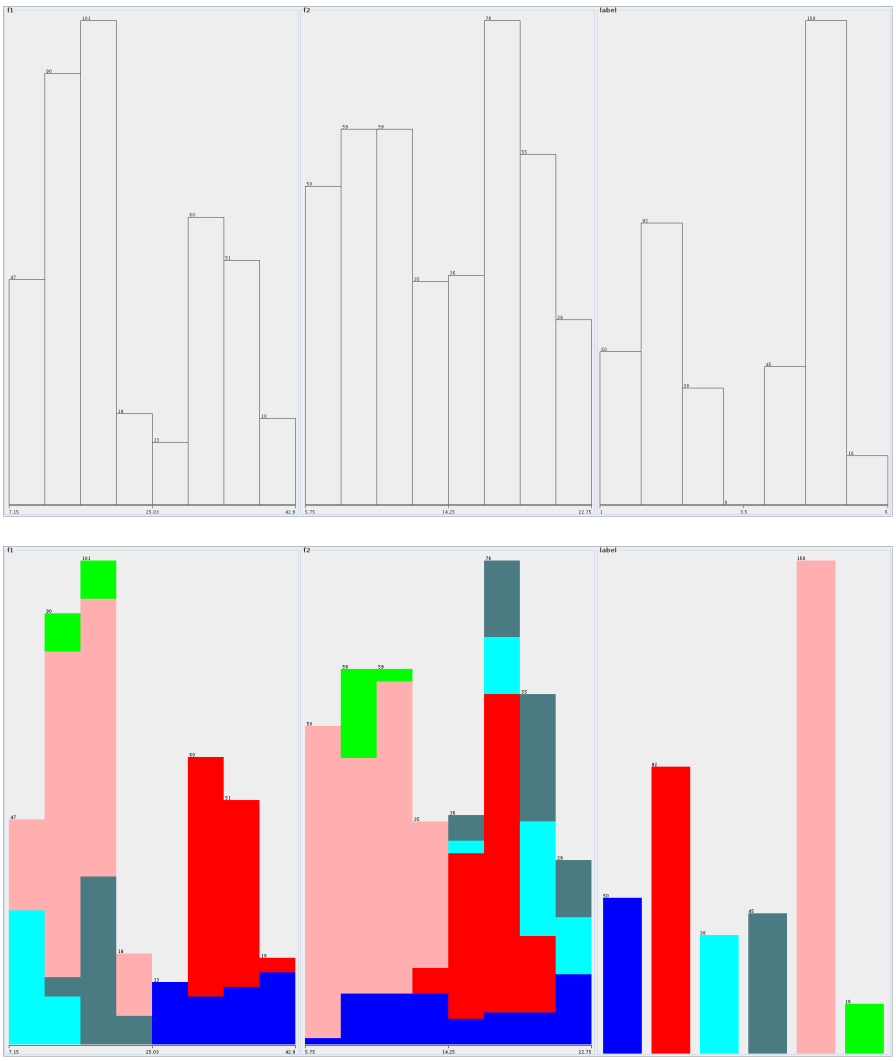


Correlation Analysis

Single link has possibilities of merging the interconnected clusters. It might treat the interconnections as cluster and hence end up with non-optimum clustering.

Complete link might give us a very good clustering output in this case since the points in the same cluster are clustered together and complete link checks for the minimum distance between the farthest points in two clusters before merging them.

(II) COMPOUND



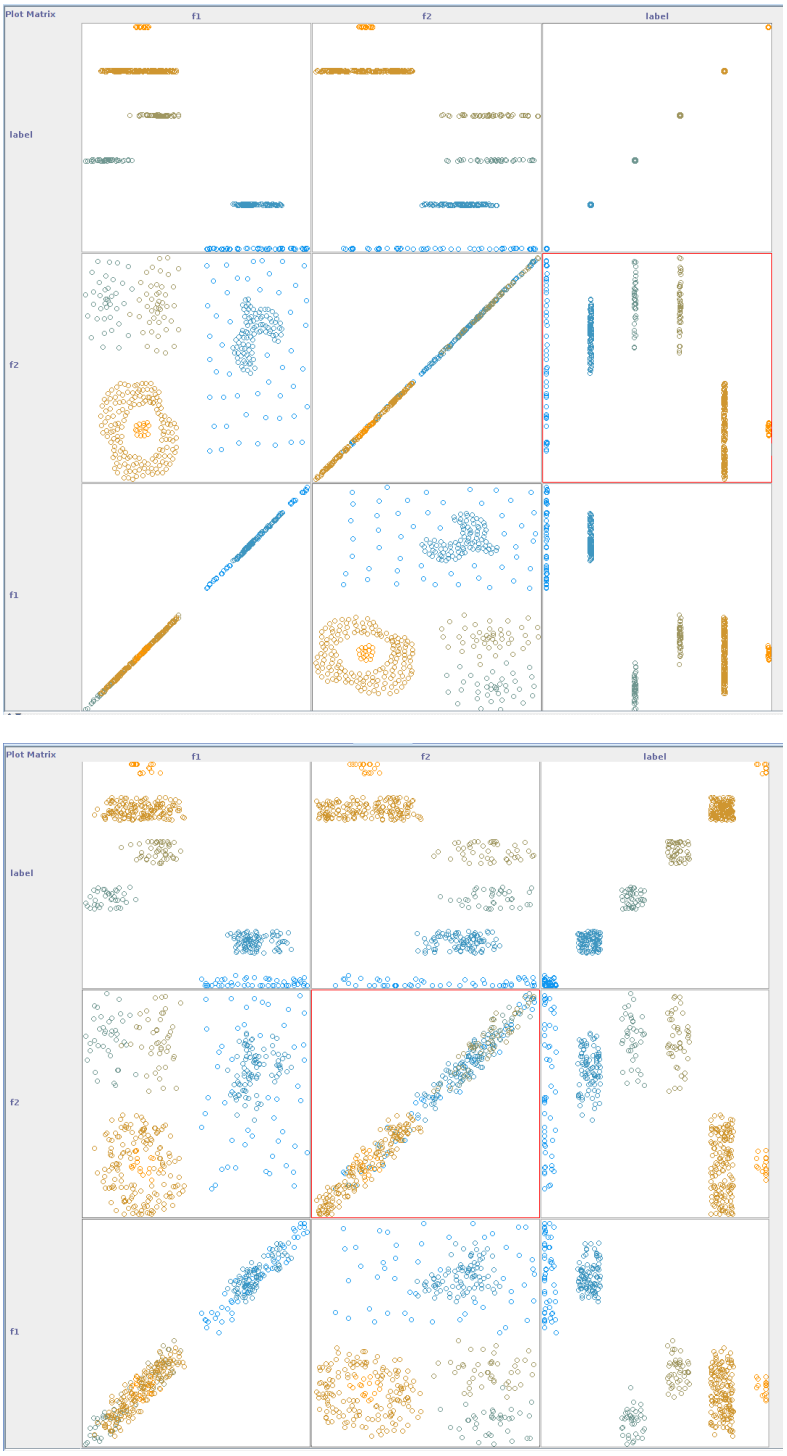
Classwise distribution of datapoints along features

Since one class encircles another, K-means won't give us the best results since it is based on distance from centroids of points.

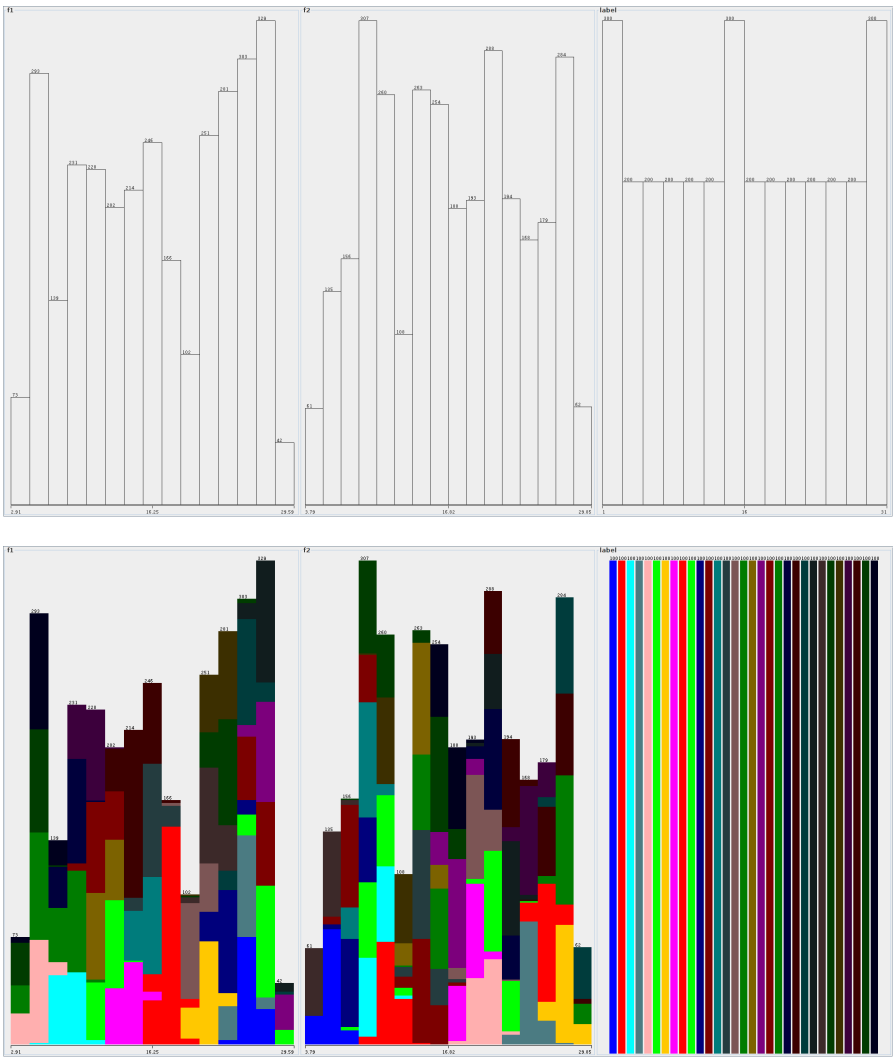
DBSCAN can identify the the classes having dense points such as with label 2,6,5,etc. But then it would fail in case of the datapoints in less dense classes. For higher value of ϵ, it would end up merging classes.

Single link might converge to a similar situation as DBSCAN. This is because of the encirclement of one class by another and due to the difference in class densities.

Complete Link might give us a good solution since it merges the clusters having minimum distance between the farthest datapoint in each cluster. Hence, we might reach a solution where in, by merging a few clusters, we can find a very good class seperation.



Correlation Analysis



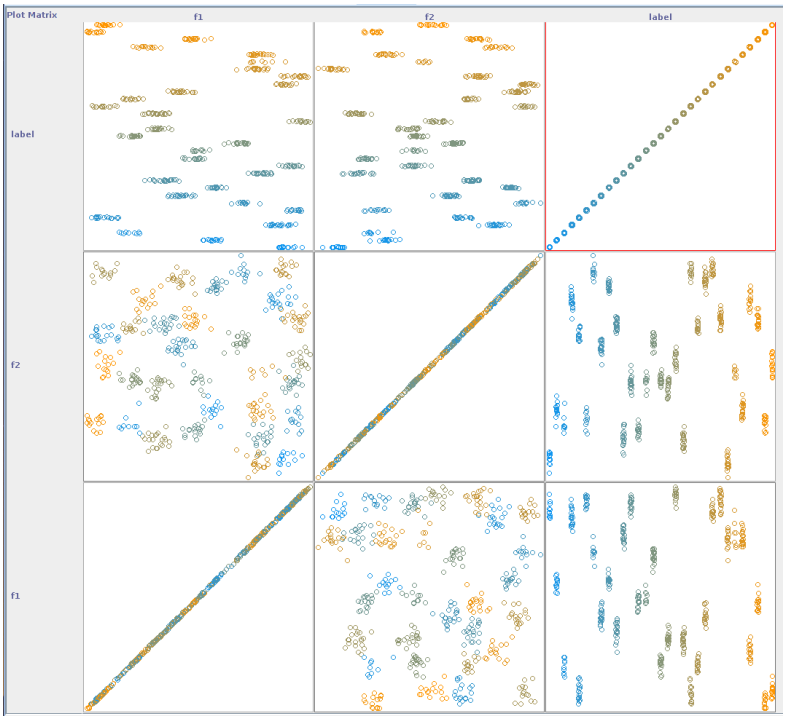
Classwise distribution of datapoints along features

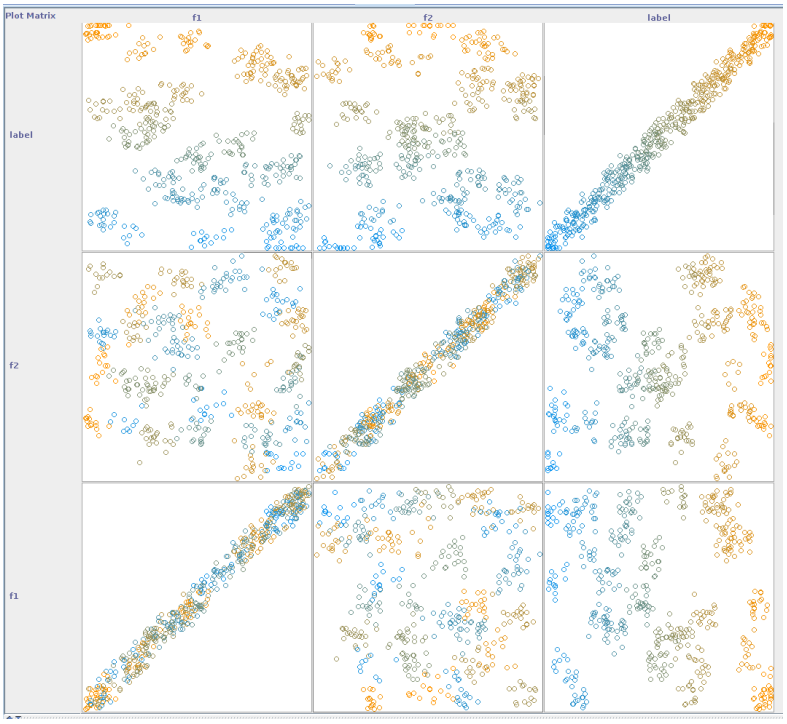
Since the datapoints are well seperated with sufficient between class variance and small in-class variance, K-means can give us very good clustering results.

Since the points belonging to the same class are densely clusterd together, DBSCAN can give us very good results by appropriately choosing ϵ and MinPts.

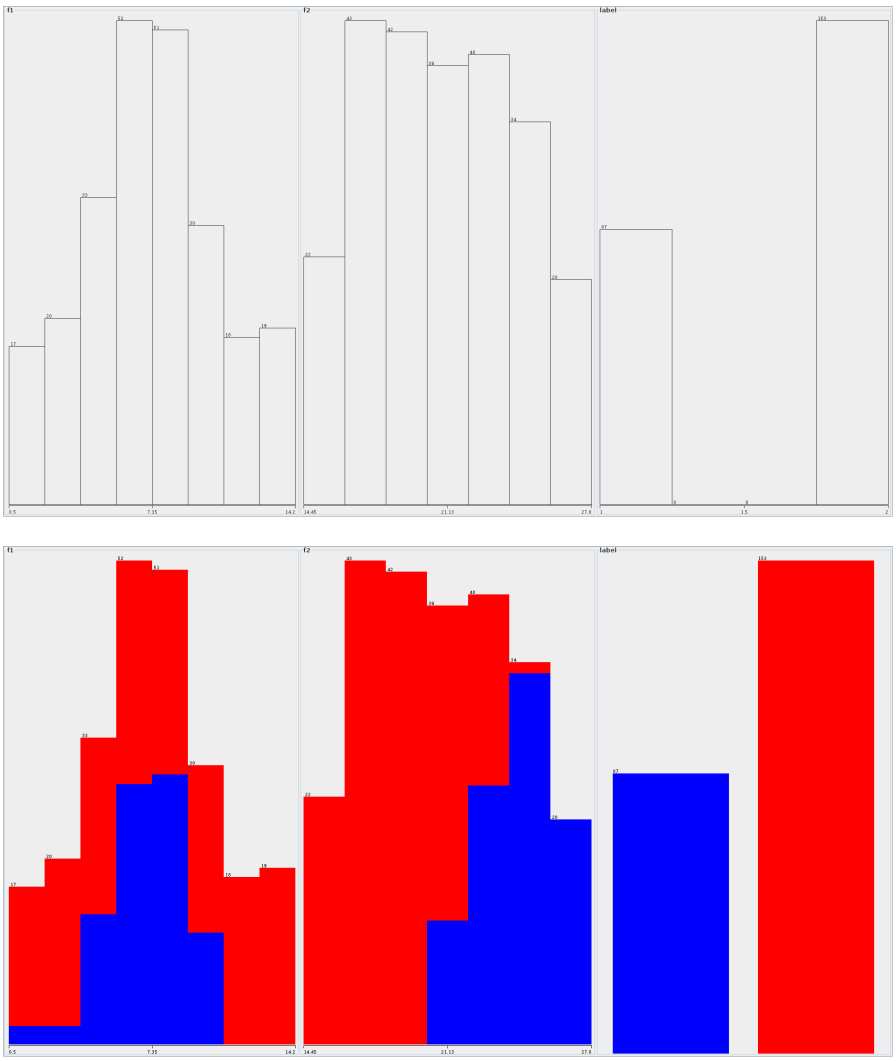
Since the points belonging to the same class are densely clusterd together(less distance between points), Single link hierarchial clustering can give us good results, as it checks for the distance between the nearest datapints in two clusters before merging them.

Complete Link might not give us a good solution since it merges the clusters having minimum distance between the farthest datapoint in each cluster. Hence, we might reach a solution where in, set of datapoints belonging to one class gets merged with another.





Correlation Analysis



Classwise distribution of datapoints along features

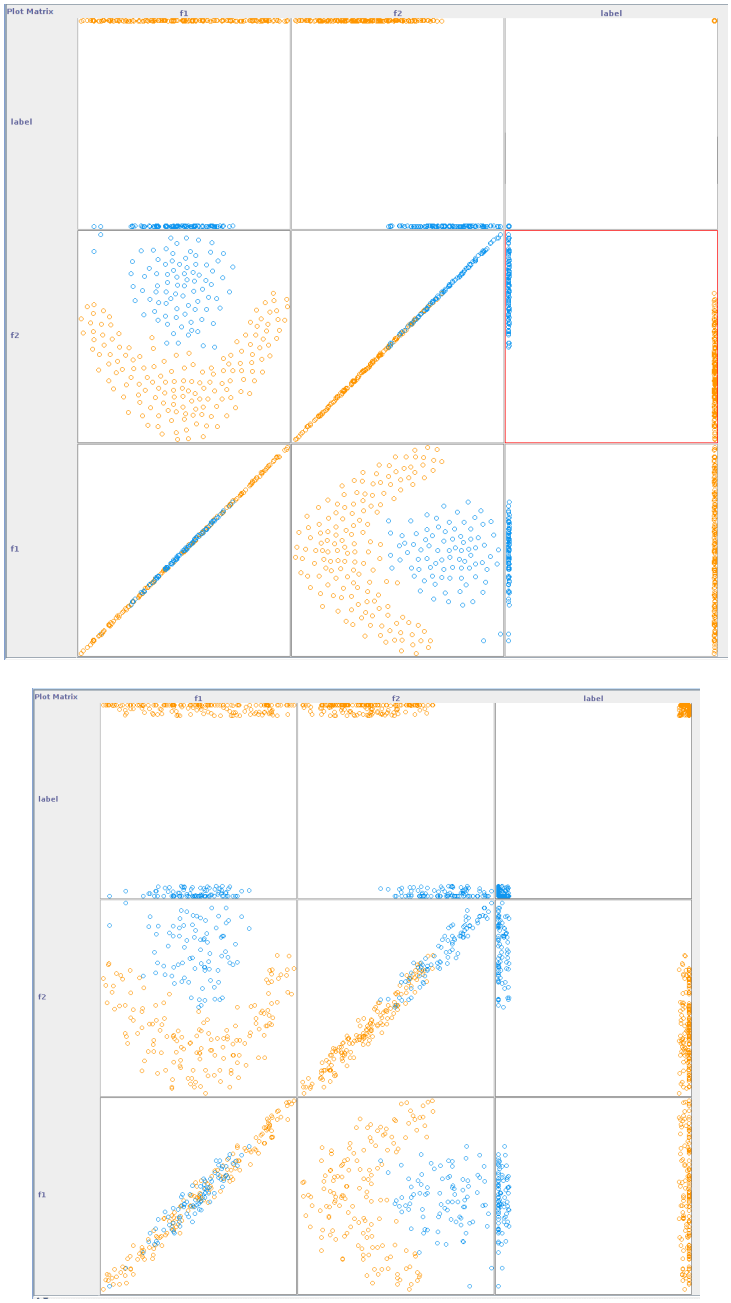
K-Means wont give us the optimum solution since one class partially encircles the other. Hence, K-means ends up misclassifying points.

DBSCAN will work really well for the given dataset since the points belonging to the same cluster are uniformly distributed(almost same density) and the density of the two classes are also nearly the same. Hence we get optimum clustering.

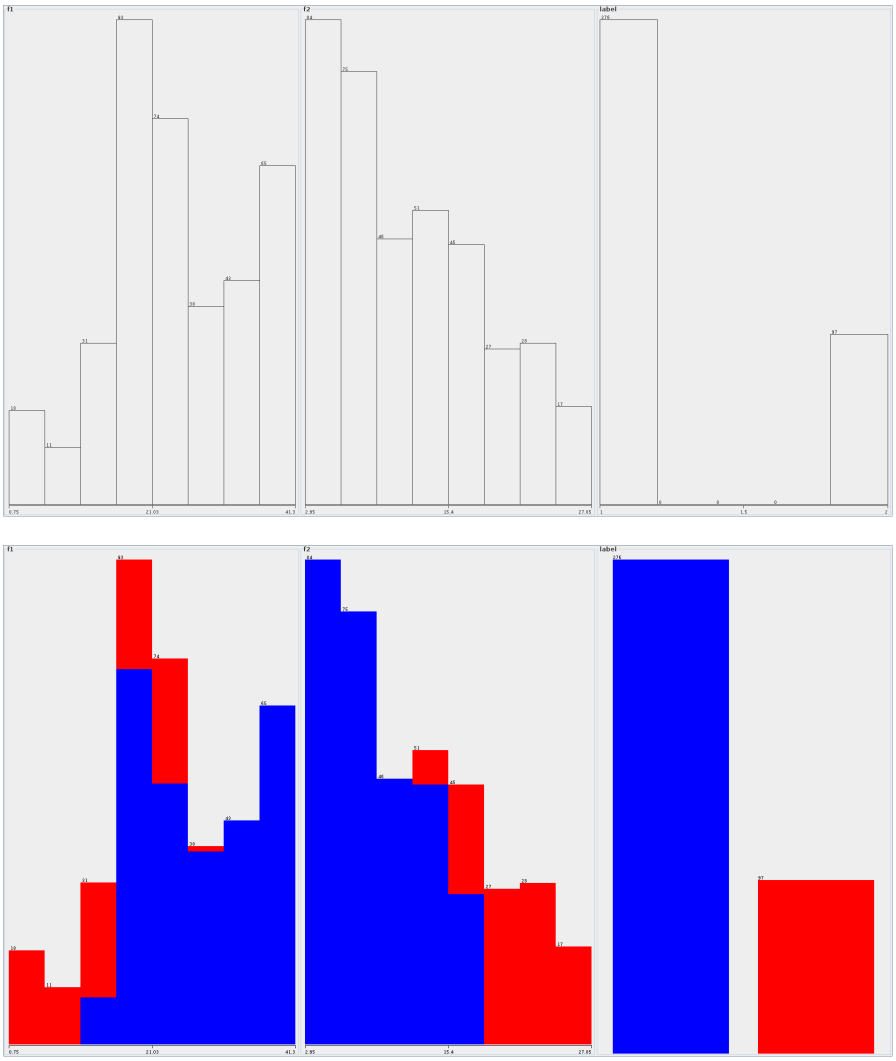
Since the points belonging to the same class are densely clusterd together(less distance between points), Single link hierarchial clustering can give us good results, as it checks for the distance between the nearest datapints in two clusters before merging them.

Complete Link might not give us a good solution since it merges the clusters having minimum distance between the farthest datapoint in each cluster. Hence, we might reach a

solution where in, set of datapoints belonging to one class gets merged with another. This is because the outer class is elongated and hence, Complete link might start merging a part of the other class(like a disk) with the existing clusters found using complete link while forming merges.



Correlation Analysis



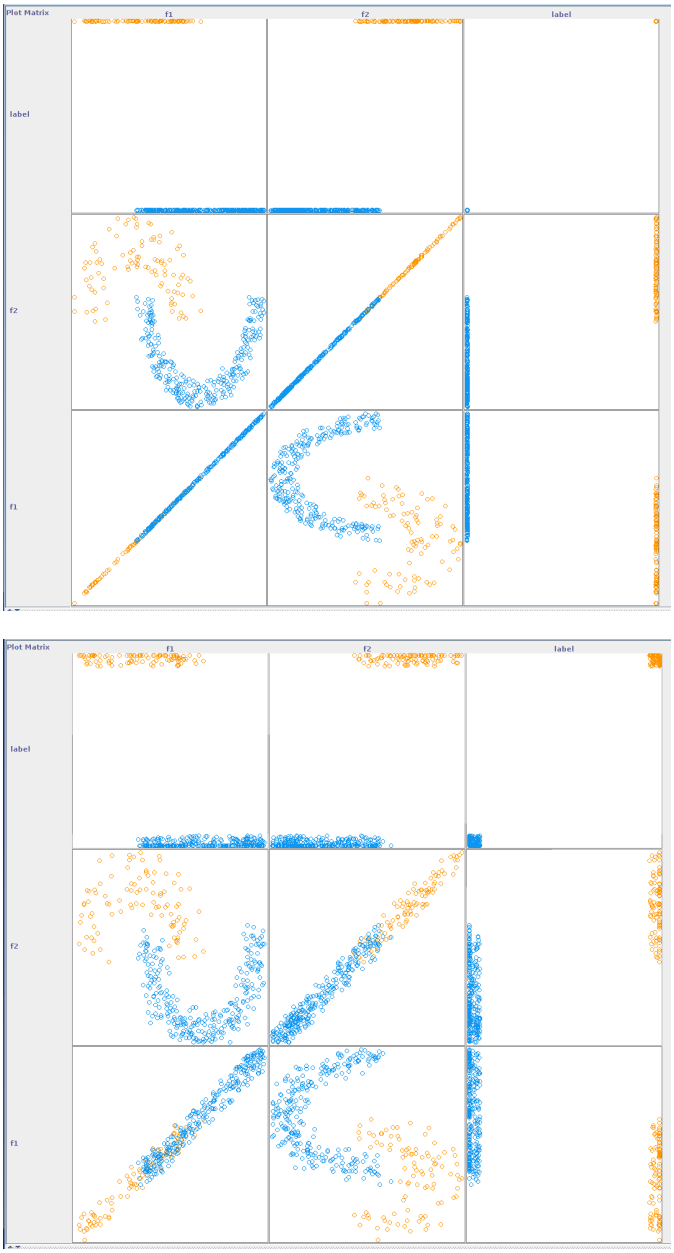
Classwise distribution of datapoints along features

Due to partial encirclement, K-means would misclassify some datapoints adn it would find a cluster seperation that is nearly a straight line, since it works on the principle of minimising the cluster centroids.

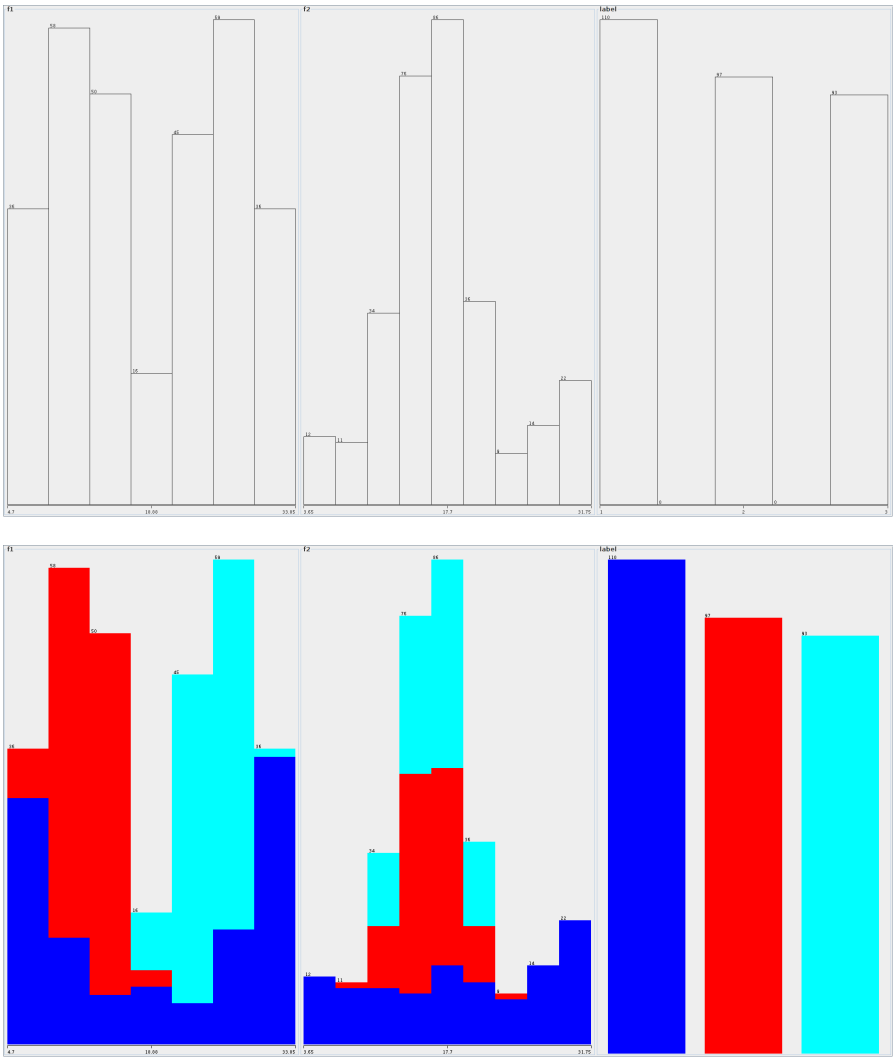
As the density within a class is almost uniform, DBSCAN may find optimum clusters, but it would depend on the relative densities between the classes as well. It may happen that a fraction of class yellow is treated as noise.

Since the points belonging to the same class are densely clusterd together(less distance between points), Single link hierarchial clustering can give us good results, as it checks for the distance between the nearest datapints in two clusters before merging them.

Complete Link might not give us a good solution since it merges the clusters having minimum distance between the farthest datapoint in each cluster. Hence, we might reach a solution where in, set of datapoints belonging to one class gets merged with another. This is because the classes are elongated and partially encircling and hence, Complete link might start merging a part of the other class with the existing clusters found using complete link while forming merges. If there are cluster points in yellow closer to blue, then they will get misclassified.



Correlation Analysis



Classwise distribution of datapoints along features

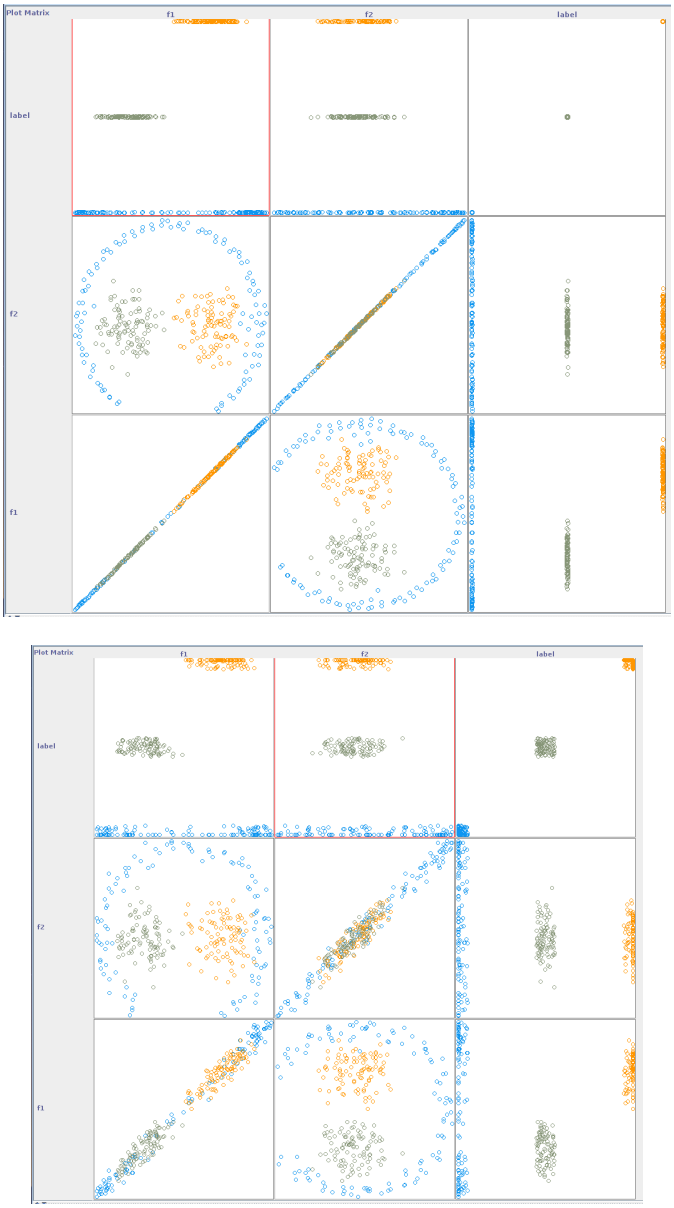
K-means would fail in the identification of 3 clusters, as one cluster is encircling the other two. Hence two classes can be correctly identified but a large fraction of the blus class gets misclassified.

As the density within a class is almost uniform, DBSCAN can find optimum clusters, but it would depend on the relative densities between the classes as well. It may happen that a fraction of class yellow or green are treated as noise. But by right choice of ϵ and MinPts, we may be able to find the optimum clustering.

Since the points belonging to the same class are densely clusterd together(less distance between points), Single link hierarchial clustering may give us good results, as it checks for

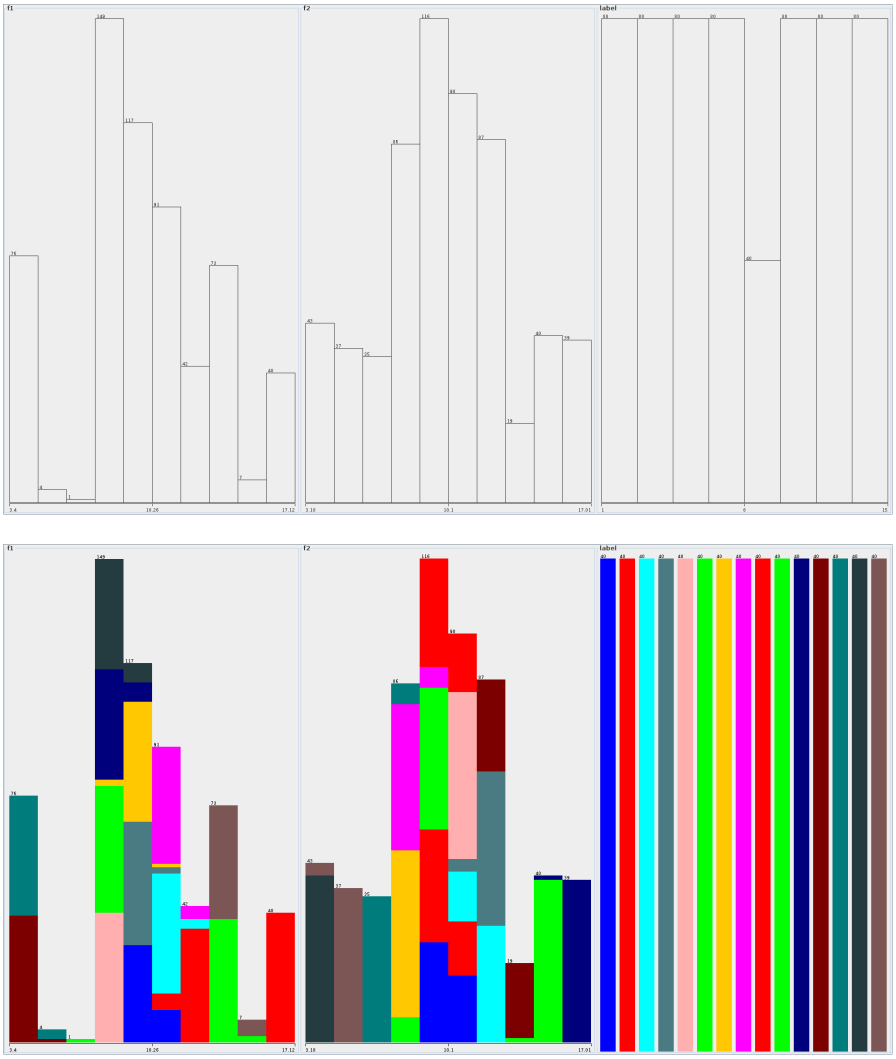
the distance between the nearest datapints in two clusters before merging them. But it might misclassify the datapoints belonging to different classes that are close to one another.

Complete Link won't give us a good solution since it merges the clusters having minimum distance between the farthest datapoint in each cluster. Hence, we might reach a solution where in, set of datapoints belonging to one class gets merged with another. This is because the blue classe is elongated and partially encircling the other two classes and hence, Complete link might start merging a part of the other class with the existing clusters found using complete link while forming merges.



Correlation Analysis

(VII)R15



Classwise distribution of datapoints along features

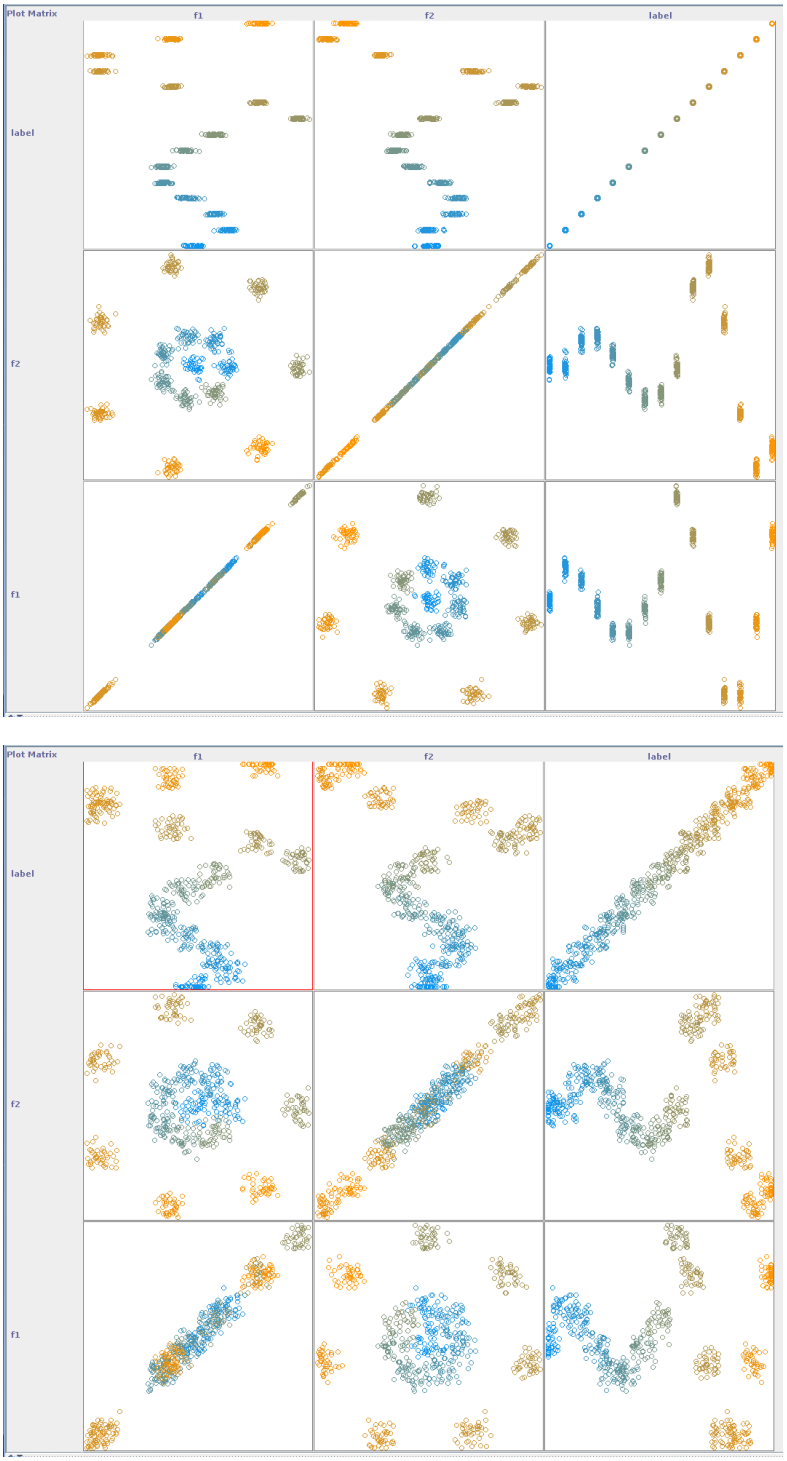
Since the datapoints are well seperated with sufficient between class variance and small in-class variance, K-means can give us very good clustering results, as it is based on the distance from centroids.

Since the points belonging to the same class are densely clusterd together, DBSCAN can give us very good results by appropriately choosing ϵ and MinPts.

Since the points belonging to the same class are densely clusterd together(less distance between points), Single link hierarchial clustering can give us good results, as it checks for the distance between the nearest datapoints in two clusters before merging them.

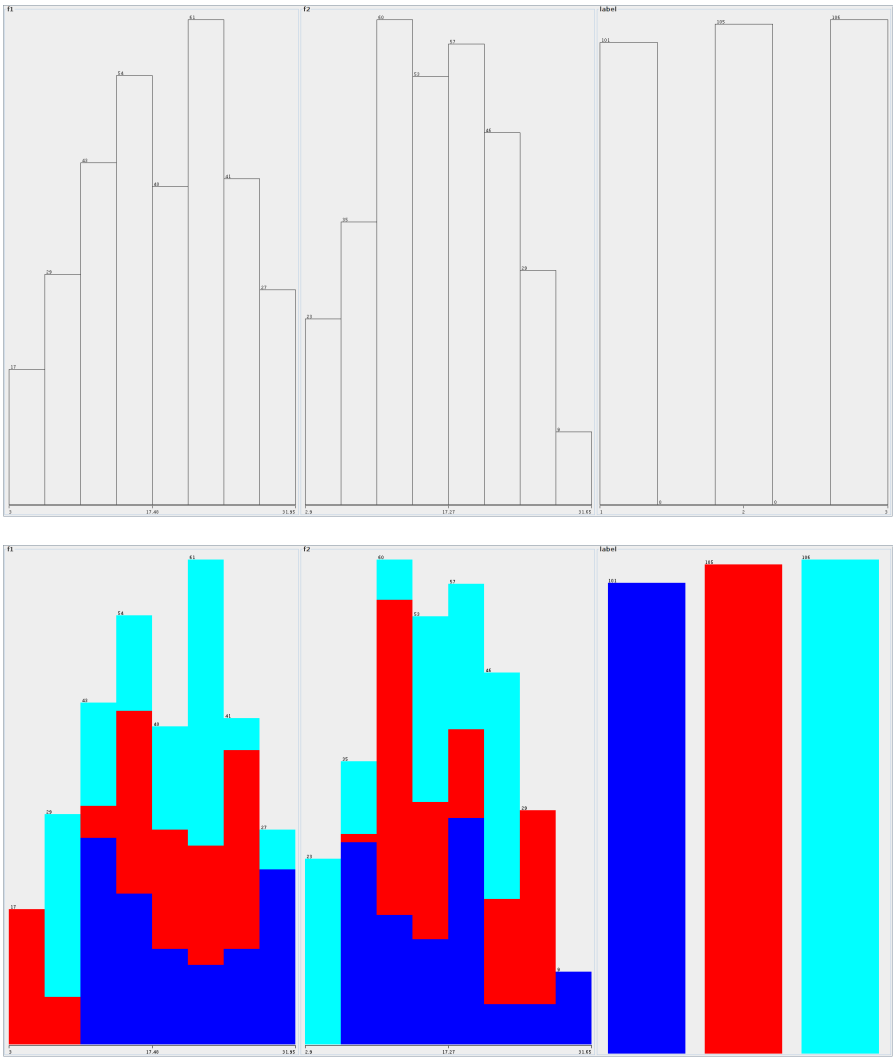
Complete Link might not give us a good solution since it merges the clusters having minimum distance between the farthest datapoint in each cluster. Hence, we might reach a

solution where in, set of datapoints belonging to one class gets merged with another.



Correlation Analysis

(VIII) SPIRAL



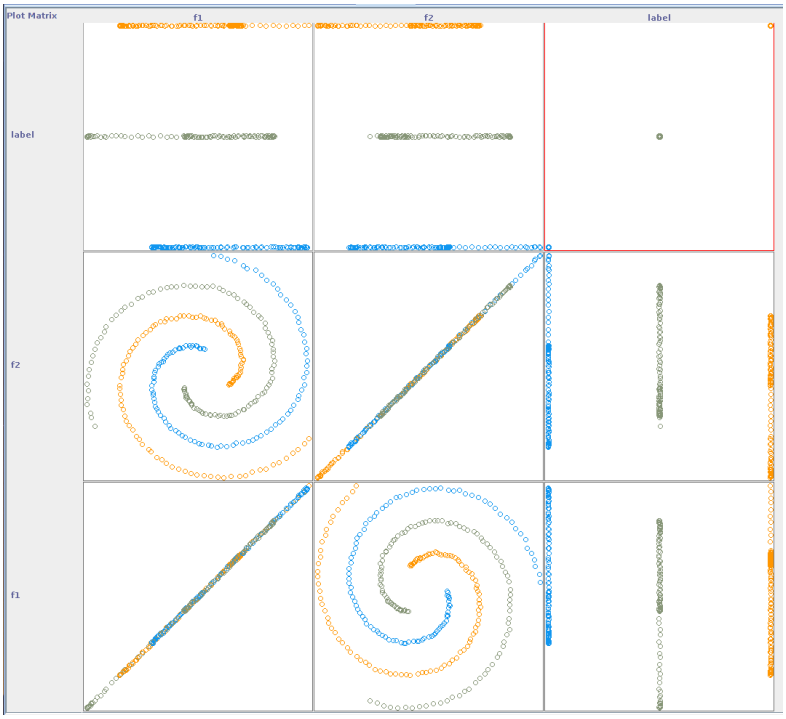
Classwise distribution of datapoints along features

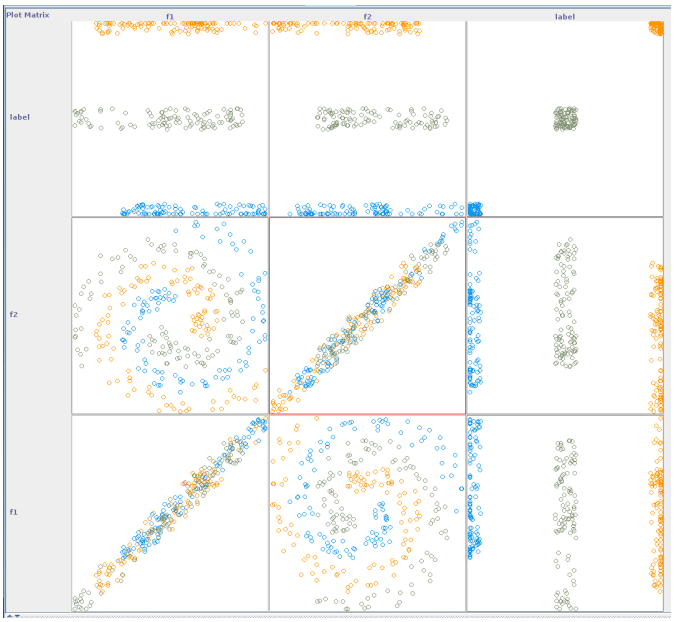
K-means would converge to a very bad clustering results as the classes are spiralling one another. Hence K-means cannot be used to identify the classes, as the optimum centroid is in the center of the spiral.

Since the points belonging to the same class are densely clusterd together, DBSCAN can give us very good results by appropriately choosing ϵ and MinPts. (Small ϵ and appropriate MinPts)

Since the classes are in the form of spirals, Single link can give us good clusters for classification.Since the points belonging to the same class are densely clusterd together(less distance between points), Single link hierarchial clustering can give us good results, as it checks for the distance between the nearest datapoints in two clusters before merging them.

Complete Link will give us a very bad solution since it merges the clusters having minimum distance between the farthest datapoint in each cluster. Hence, we will reach a solution where in, set of datapoints belonging to one class gets merged with another.





Correlation Analysis

3.R15 : K-MEANS AND CLUSTER PURITY

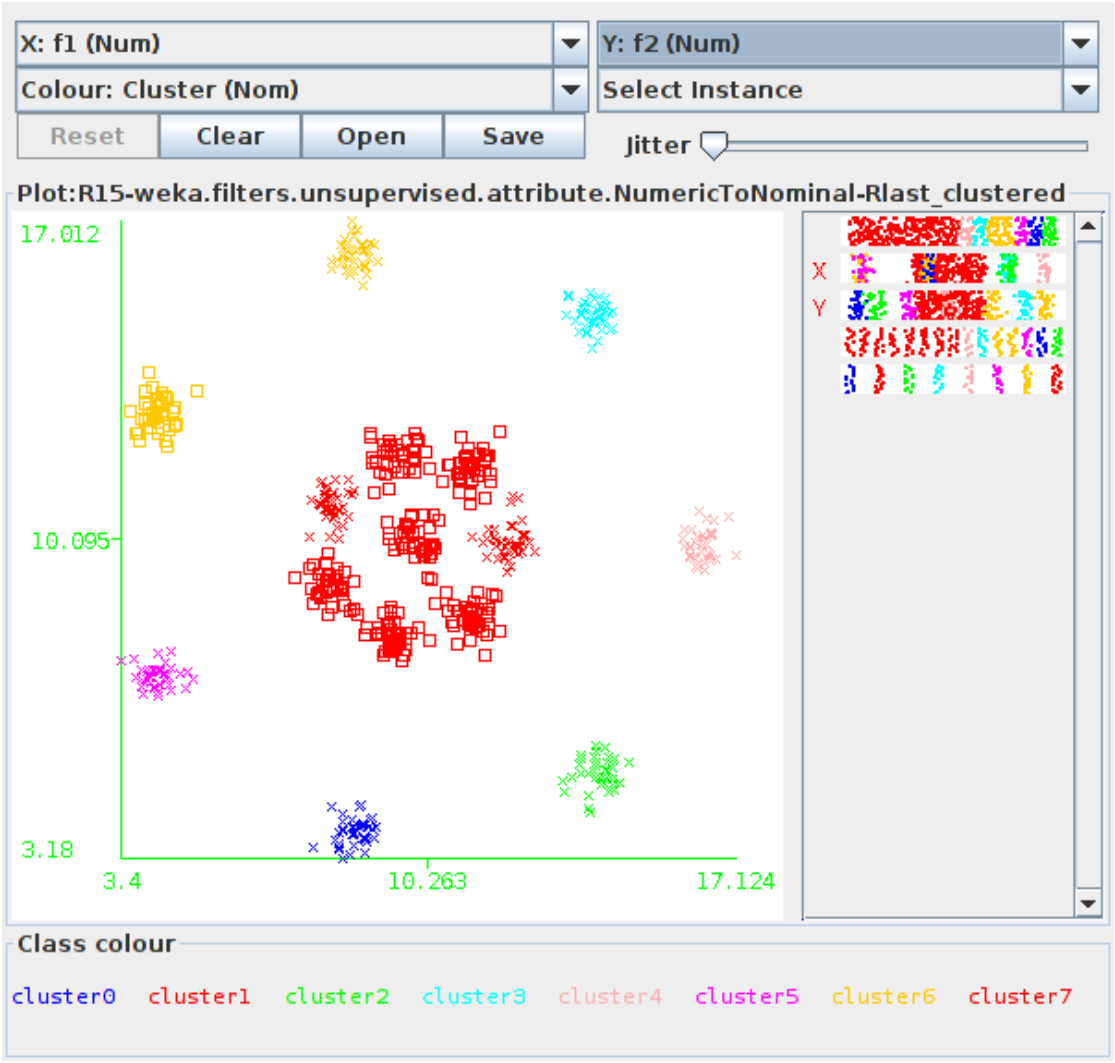
K-means clustering is a type of unsupervised learning, which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

Purity is a measure of the extent to which clusters contain a single class. It is an evaluation measure for the quality of clustering. It is calculated by counting the number of data points from the most common class in a cluster. and then take the sum over all clusters and divide by the total number of data points. Given some set of clusters M and some set of classes D, both partitioning N data points, purity can be defined as:

$$Purity = \frac{1}{N} \sum_{m \epsilon M} \max_{d \epsilon D} |m \cap d|$$

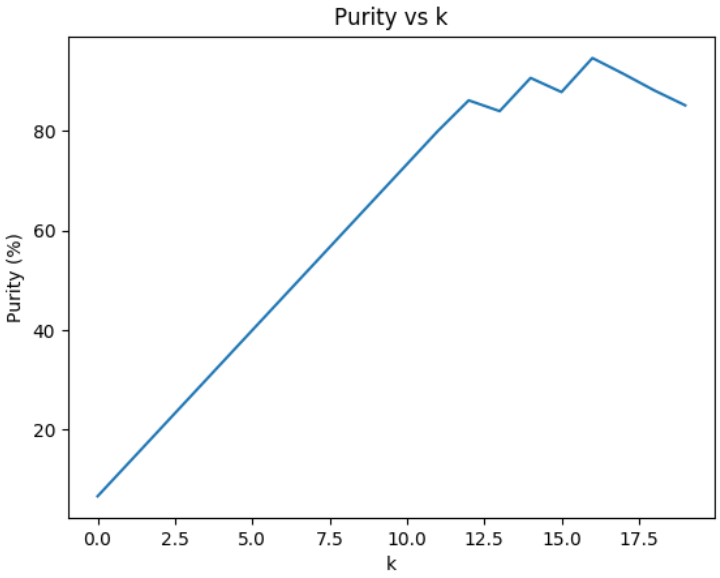
The results of clustering with k=8 is as follows :

```
Classes to Clusters:

  0  1  2  3  4  5  6  7  <-- assigned to cluster
  0 31  0  0  0  0  0  9 | 1
  0  0  0  0  0  0  0 40 | 2
  0  0  0  0  0  0  0 40 | 3
  0  0  0  0  0  0  0 40 | 4
  0 40  0  0  0  0  0  0 | 5
  0 40  0  0  0  0  0  0 | 6
  0 40  0  0  0  0  0  0 | 7
  0 38  0  0  0  0  0  2 | 8
  0  0  0  0 40  0  0  0 | 9
  0  0  0 40  0  0  0  0 | 10
  0  0  0  0  0  0 40  0 | 11
  0  0  0  0  0  0 40  0 | 12
  0  0  0  0  0 40  0  0 | 13
 40  0  0  0  0  0  0  0 | 14
  0  0 40  0  0  0  0  0 | 15

Cluster 0 <-- 14
Cluster 1 <-- 5
Cluster 2 <-- 15
Cluster 3 <-- 10
Cluster 4 <-- 9
Cluster 5 <-- 13
Cluster 6 <-- 11
Cluster 7 <-- 2

Incorrectly clustered instances :       280.0    46.6667 %
```



The cluster purity for different values of K have been listed below :

k	Purity (%)
1	6.6667
2	13.3333
3	20
4	26.6667
5	33.3333
6	40
7	46.6667
8	53.3333
9	60
10	66.6667
11	73.3333
12	80
13	86.1667
14	84
15	90.6667
16	87.8333
17	94.6667
18	91.5
19	88.1667
20	85.1667

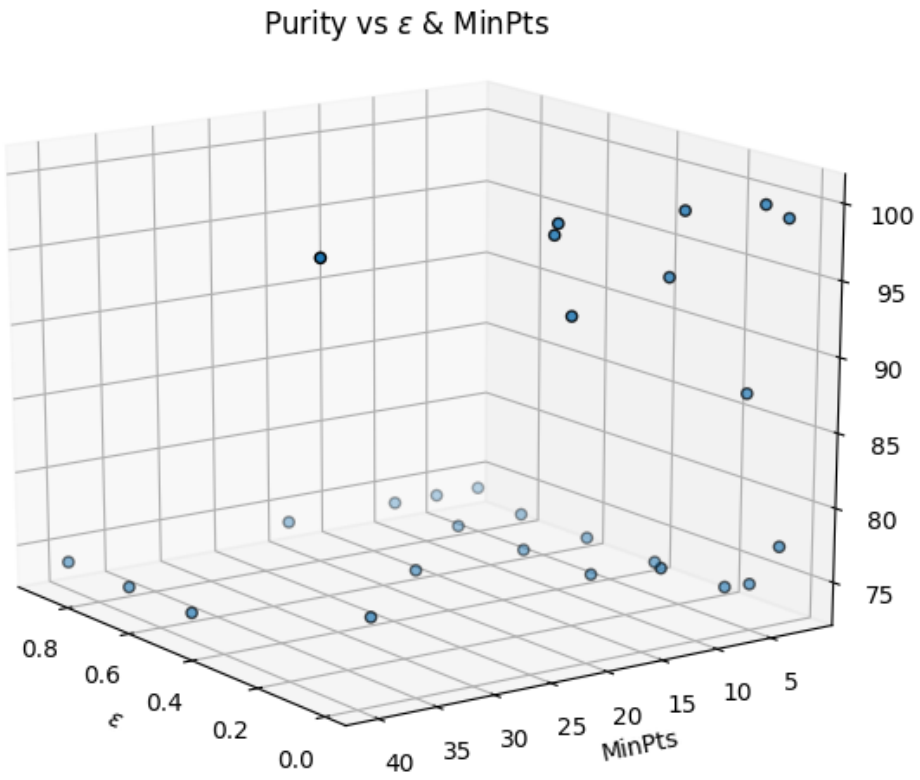


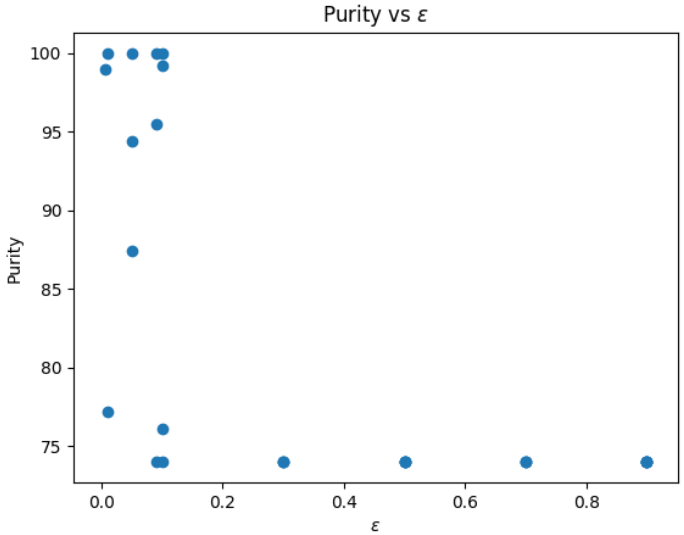
Purity vs No. of Clusters

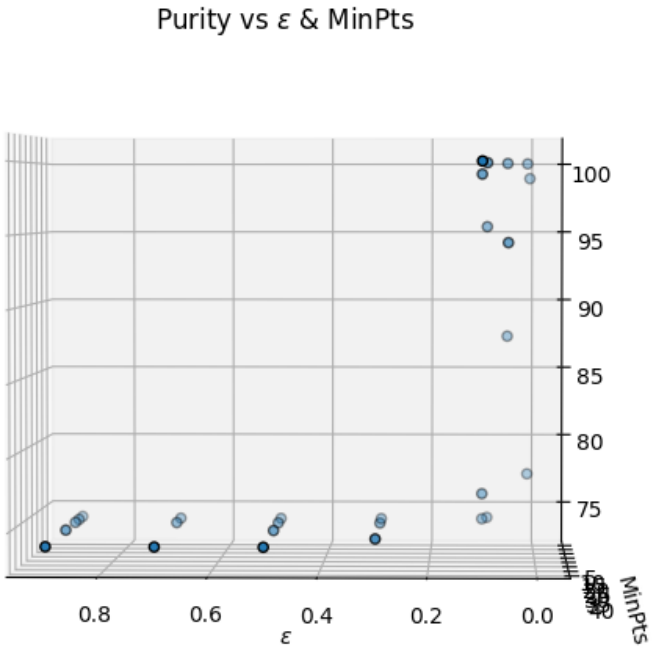
4. JAIN : DBSCAN AND CLUSTER PURITY

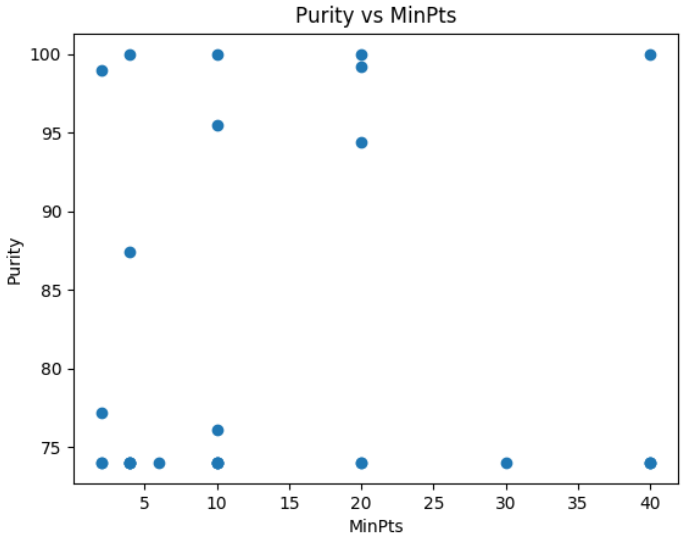
Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm which is density based: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low density regions (whose nearest neighbors are too far away).

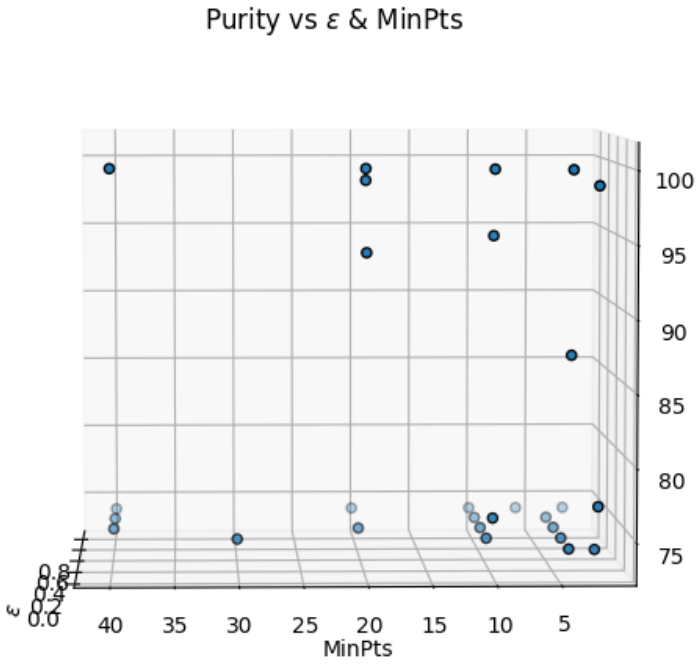
The variation of Purity with ϵ and MinPts have been evaluated for some changes of values and the plotted as shown below :


Purity vs ε & MinPts
100
95
90
85
80
75
0.8
0.6
0.4
0.2
0.0
ε
40
35
30
25
20
15
10
5
MinPts


Purity vs ε
Purity
100
95
90
85
80
75
0.0
0.2
0.4
0.6
0.8
ε


Purity vs ε & MinPts
100
95
90
85
80
75
0.8
0.6
0.4
0.2
0.0
ε
MinPts


Purity vs MinPts
Purity
100
95
90
85
80
75
5
10
15
20
25
30
35
40
MinPts



5. DBSCAN VS HIERARCHIAL CLUSTERING

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. The following are the strategies used for hierarchical clustering :

- Agglomerative: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy
- Divisive: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy

Usually, the merges and splits are determined in a greedy manner and the results of hierarchical clustering are presented in a dendrogram.

(I) PATH-BASED

DBSCAN			
ϵ	**MinPts**	**Purity**(%)	**Number of clusters**
0.1	4	36.6667	1
0.1	6	37	1
0.1	10	54.3333	2
0.1	15	94.3333	2
0.1	20	98	2
0.1	25	99	2
0.1	30	99.3333	2
0.1	35	99.3333	2
0.1	40	100	1

But we notice from the output that as we increase the number of MinPts, the number of identified clusters first increases and then reduces and it classifies the low density regions as noise and the number of points classified as noise increases as MinPts increases. Here we haven't been able to find 3 clusters corresponding to the number of class labels due to the choice of the value of ϵ. For $\epsilon = 0.1$, no choice of MinPts give us more than 2 clusters.

But say $\epsilon = 0.05$, then we see that for Minpts = 10, we get a purity of 94.3333% with 3 clusters.

Heirarchial	
TYPE	**Purity**(%)
Single	27
Complete	70.6667
Average	63
Mean	70
Centroid	73.3333
Ward	75.3333
AdjComplete	64
NeighborJoining	36.6667

This has been done for Number of Clusters = 3 = Number of Classes
From the above observation, WARD gives us the highest purity score for the Path-based dataset.

(II) SPIRAL

DBSCAN			
ϵ	**MinPts**	**Purity(%)**	**Number of clusters**
0.1	4	100	3
0.1	6	100	3
0.1	10	100	3
0.1	15	100	3
0.1	20	100	3
0.1	25	100	3

But we notice from the output that as we increase the number of MinPts, it classifies the low density regions as noise and the number of points classified as noise increases as MinPts increases.

```
Class attribute: label
Classes to Clusters:

   0   1   2  <-- assigned to cluster
   0 101   0 | 1
   0   0 105 | 2
 106   0   0 | 3

Cluster 0 <-- 3
Cluster 1 <-- 1
Cluster 2 <-- 2
```

(ai) $MinPts = 4$

```
Class attribute: label
Classes to Clusters:

  0  1  2  <-- assigned to cluster
  0 55  0 | 1
  0  0 62 | 2
 58  0  0 | 3

Cluster 0 <-- 3
Cluster 1 <-- 1
Cluster 2 <-- 2
```

(aj) $MinPts = 10$

```
Class attribute: label
Classes to Clusters:

  0  1  2  <-- assigned to cluster
  0 32  0 | 1
  0  0 33 | 2
 36  0  0 | 3

Cluster 0 <-- 3
Cluster 1 <-- 1
Cluster 2 <-- 2
```

(ak) $MinPts = 15$

```
Class attribute: label
Classes to Clusters:

  0  <-- assigned to cluster
  0 | 1
  0 | 2
 25 | 3

Cluster 0 <-- 3
```

(al) $MinPts = 25$

Hence for $\epsilon = 0.1$ and Minpts = 4, we get a purity of 100% with 3 clusters.

Heirarchial	
TYPE	**Purity**(%)
Single	100
Complete	38.141
Average	36.2179
Mean	39.1026
Centroid	40.3846
Ward	40.0641
AdjComplete	35.5769
NeighborJoining	33.9744

This has been done for Number of Clusters = 3 = Number of Classes.
From the above observation, SINGLE gives us the highest purity score for the Spiral dataset.

(III) FLAMES

DBSCAN			
ϵ	**MinPts**	**Purity**(%)	**Number of clusters**
0.1	4	62.5833	1
0.1	6	64.5833	1
0.1	10	99.1667	2
0.1	15	91.25	3
0.1	20	100	1

But we notice from the output that as we increase the number of MinPts, it classifies the low density regions as noise and the number of points classified as noise increases as MinPts increases.

```
Class attribute: label
Classes to Clusters:

  0  1  2  <-- assigned to cluster
  0  0 53 | 1
 58 21  0 | 2

Cluster 0 <-- 2
Cluster 1 <-- No class
Cluster 2 <-- 1
```

(am) $MinPts = 15$

```
Class attribute: label
Classes to Clusters:

  0  <-- assigned to cluster
  0 | 1
 24 | 2

Cluster 0 <-- 2
```

(an) $MinPts = 20$

```
Class attribute: label
Classes to Clusters:

   0  <-- assigned to cluster
  85 | 1
 153 | 2

Cluster 0 <-- 2
```

(ao) $MinPts = 6$

```
Class attribute: label
Classes to Clusters:

   0   1  <-- assigned to cluster
   1  82 | 1
 152   1 | 2

Cluster 0 <-- 2
Cluster 1 <-- 1
```

(ap) $MinPts = 10$

Hence for $\epsilon = 0.1$ and Minpts = 10, we get a purity of 99.1667% with 2 clusters.

Heirarchial	
TYPE	**Purity**(%)
Single	64.5883
Complete	51.6667
Average	83.3333
Mean	92.0833
Centroid	64.5833
Ward	100
AdjComplete	64.1667
NeighborJoining	63.75

This has been done for Number of Clusters = 2 = Number of Classes.
From the above observation, WARD gives us the highest purity score for the Flames dataset.

INFERENCE

Class	**Clustering Algorithm**	**Reason**
Path-based	DBSCAN ($\epsilon = 0.05$&$MinPts = 10$))	We get better purity value with 3 clusters
Spiral	DBSCAN & Hierarchial SINGLE	We get 100% purity value using both
Flames	Hierarchial SINGLE	We get 100% purity value with 2 clusters

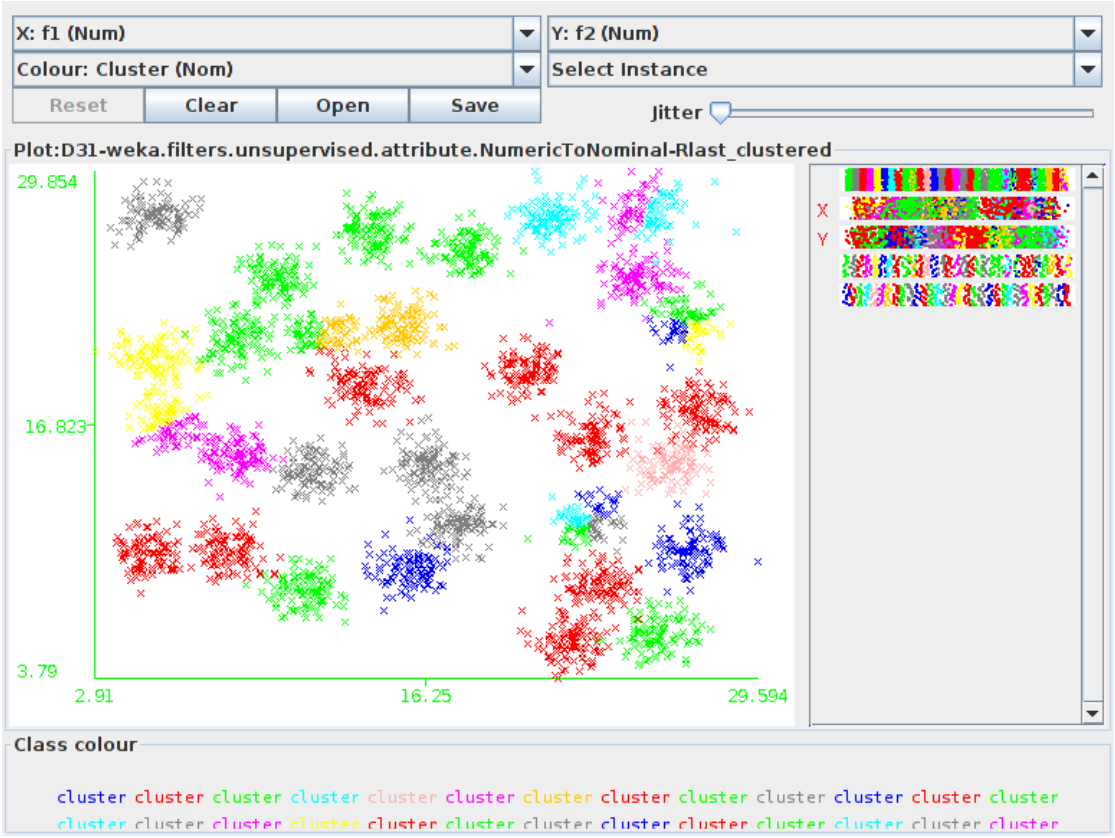
6. D31 DATASET

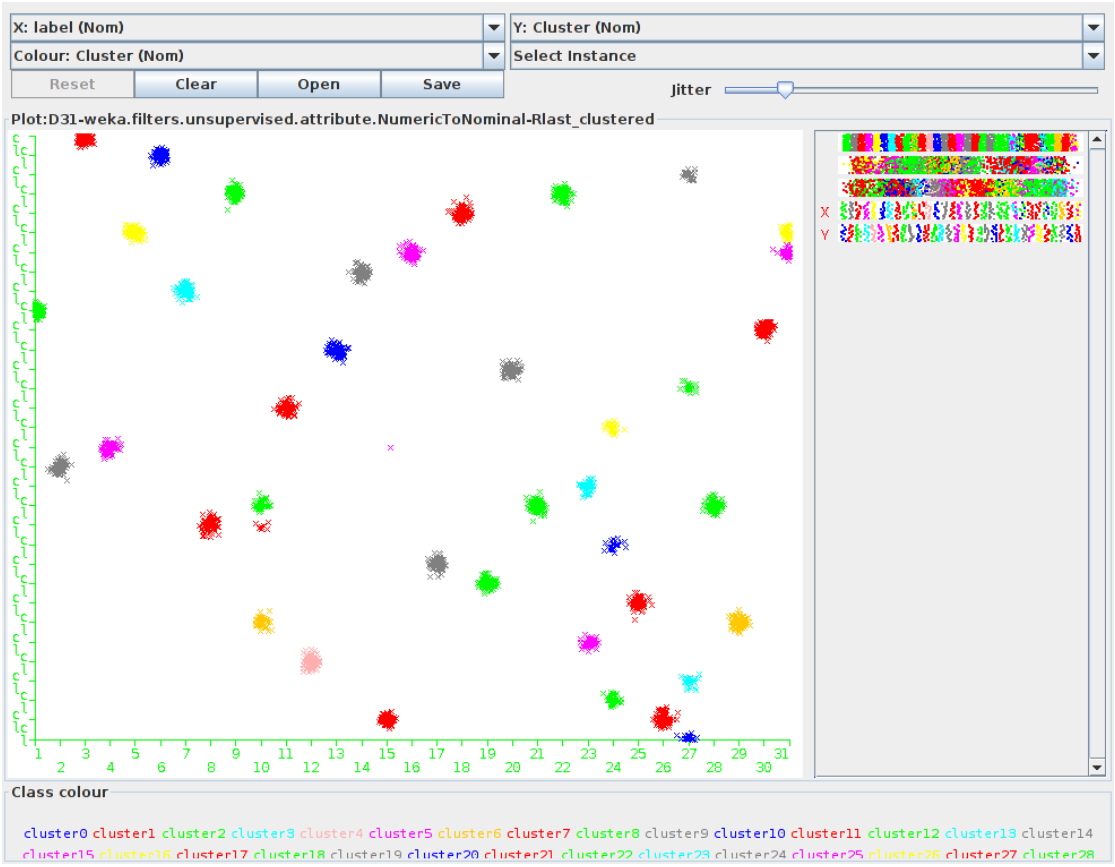
K-MEANS

On using sklearn kmeans package, the following results were obtained :

k	Rand Index
31	0.952873445973
32	0.945115106759
40	0.858723966641
50	0.774456828936
70	0.604866171665

On using Weka :o The data has 31 classes and 3100 datapoints. On directly running K-means with k=32 and supplying the class labels for comparison, convergence wasn't obtained even after leaving the code running for more than a day on Weka, even if we specify the number of iterations as 1 (which is the lowest). k=32 :

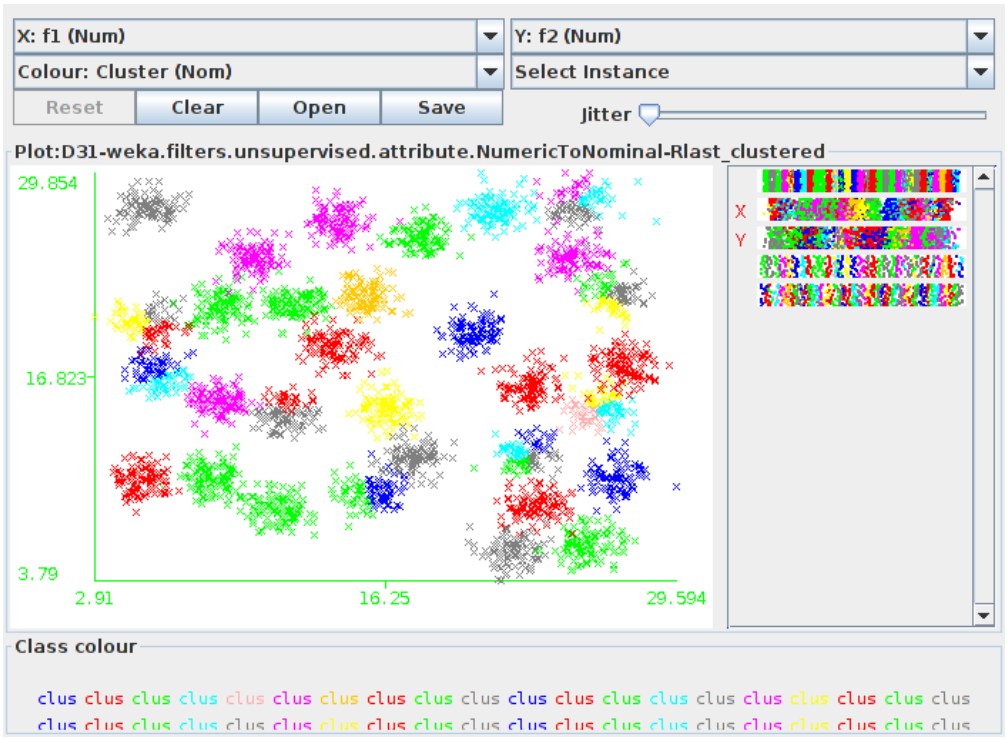


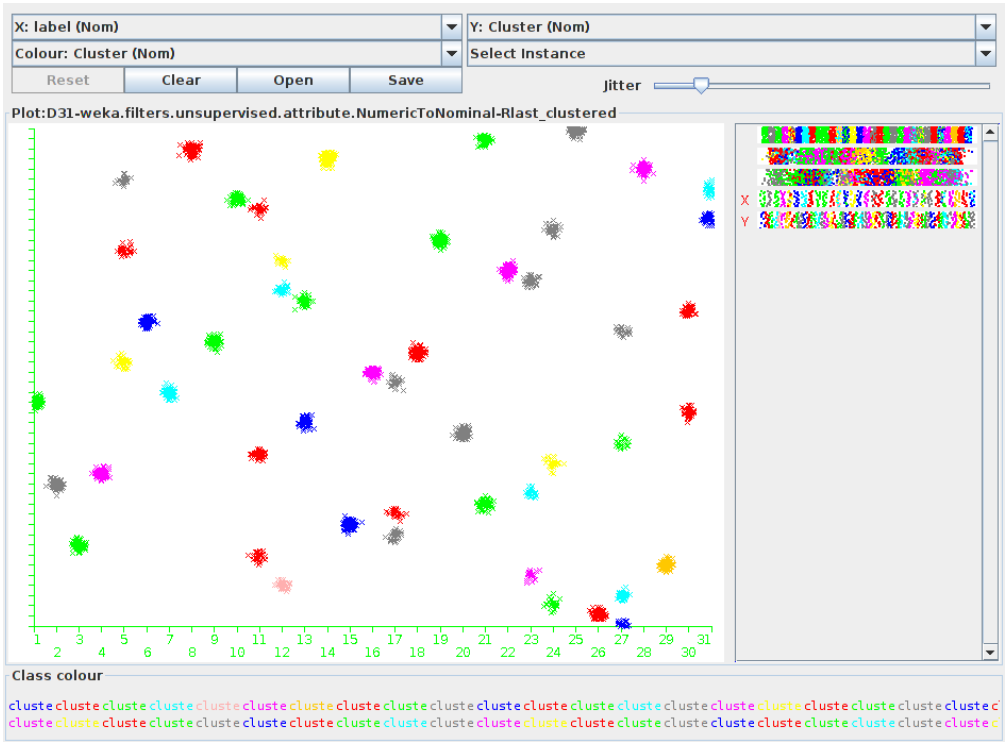


Hence, just the clustering has been implemented and the outcome of k=32 has been shown. On clicking on the clustered points, we get to know the cluster they belong to and for k<31 we see that there are datapoints close to one another that have been classified in different clusters. There are points farther away that has been classified as belonging to the same cluster as well. Hence for k=32, we'll mostly not be able to capture a good fraction of data belonging to different classes, but the amount of miss-classifications would reduce by clustering using a larger value of k because, as k increases, points that are more densly close to one another would be classified as belonging to the same cluster and the the points that are relatively farther would be classified in different clusters. Hence, by merging the appropriate clusters, we can get almost all the 31 real clusters by choosing an appropriate k value.

We also see from the cluster vs label figure that the points have been clustered randomly and not necessarily linearly.

For k=50 :


X: f1 (Num)
Y: f2 (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot:D31-weka.filters.unsupervised.attribute.NumericToNominal-Rlast_clustered
29.854
16.823
3.79
2.91
16.25
29.594
X
Y
Class colour


X: label (Nom)
Y: Cluster (Nom)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot:D31-weka.filters.unsupervised.attribute.NumericToNominal-Rlast_clustered
X
Y
Class colour

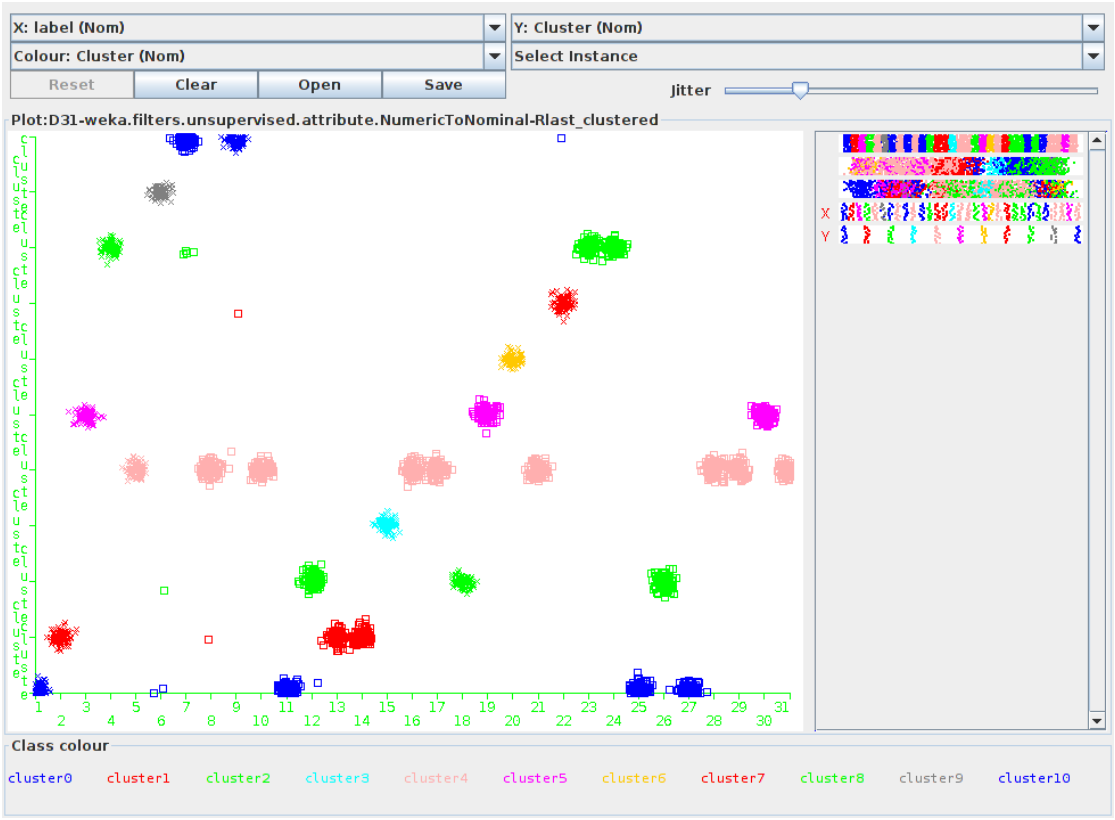
For k=10 :

```
Class attribute: label
Classes to Clusters:

   0   1   2   3   4   5   6   7   8   9  <-- assigned to cluster
   0   0   0   0   0   0   0 100   0   0 | 1
   0   0   0 100   0   0   0   0   0   0 | 2
   0   0   0   0   0   0   0   0 100   0 | 3
   0   0 100   0   0   0   0   0   0   0 | 4
   0   0   0   0   0   0   3   0   0  97 | 5
  98   0   0   0   0   0   0   2   0   0 | 6
   0   0  67   0   0  33   0   0   0   0 | 7
   0 100   0   0   0   0   0   0   0   0 | 8
   0   0   0   0   0 100   0   0   0   0 | 9
   0  98   0   0   0   0   2   0   0   0 | 10
  12   0   0   0   0   0   0  88   0   0 | 11
  61   0   0   0  39   0   0   0   0   0 | 12
   0   0   0 100   0   0   0   0   0   0 | 13
   0   2   0  98   0   0   0   0   0   0 | 14
   0   4   1   0  95   0   0   0   0   0 | 15
   0   0   0   0   0   0   0   0   2  98 | 16
   0   1   0  75   0   0   0   0   6  18 | 17
   0   0   0   0 100   0   0   0   0   0 | 18
   0   0   0   1   0   0   0   0  99   0 | 19
   0   0   0   0   0   0 100   0   0   0 | 20
   0   5   0   0   0   0  49   0   0  46 | 21
   0   0   0   0   0 100   0   0   0   0 | 22
   0   0 100   0   0   0   0   0   0   0 | 23
   0   0  88   0  12   0   0   0   0   0 | 24
   0   0   0   0   0   0   0 100   0   0 | 25
   0   0   0   0 100   0   0   0   0   0 | 26
 100   0   0   0   0   0   0   0   0   0 | 27
   0   0   0   0   0   0 100   0   0   0 | 28
   0  96   0   0   0   4   0   0   0   0 | 29
   0   0   0   0   0   0   0   0 100   0 | 30
   0   0   0   0   0   0   0   0   0 100 | 31

Cluster  0 <-- 27
Cluster  1 <-- 8
Cluster  2 <-- 4
Cluster  3 <-- 2
Cluster  4 <-- 18
Cluster  5 <-- 9
Cluster  6 <-- 20
Cluster  7 <-- 1
Cluster  8 <-- 3
Cluster  9 <-- 31

Incorrectly clustered instances :       2100.0   67.7419 %
```

Even DBSCAN wasn't converging for small values of ϵ and larger values of MinPts. DBSCAN is found to perform poorly in most of the cases where clustering was possible. Example is as follows :

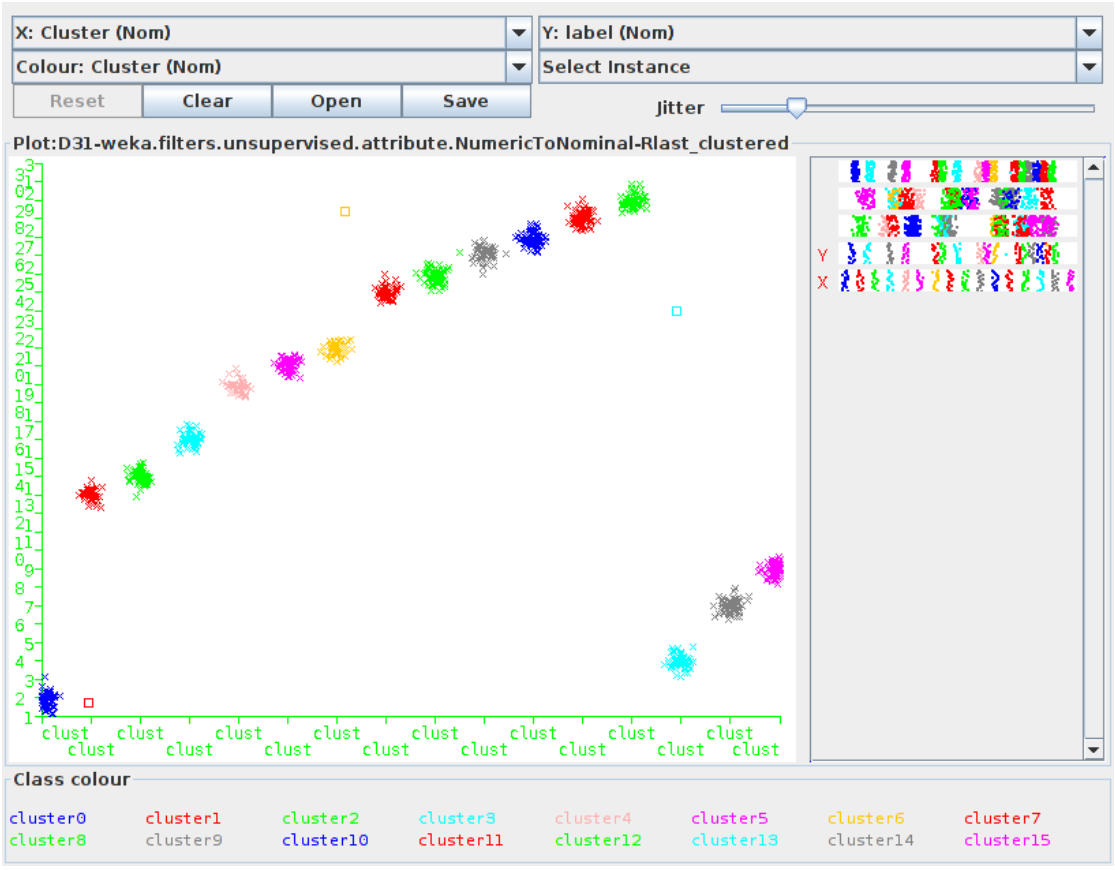
For $\epsilon = 0.05$ and MinPts = 45(maximum possible for the given ϵ) :

```
Class attribute: label
Classes to Clusters:

   0   1   2   3   4   5   6   7   8   9  10  <-- assigned to cluster
 100   0   0   0   0   0   0   0   0   0   0 | 1
   0 100   0   0   0   0   0   0   0   0   0 | 2
   0   0   0   0   0 100   0   0   0   0   0 | 3
   0   0   0   0   0   0   0   0 100   0   0 | 4
   0   0   0   0 100   0   0   0   0   0   0 | 5
   2   0   1   0   0   0   0   0   0  96   0 | 6
   0   0   0   0   0   0   0   0   4   0  94 | 7
   0   1   0   0  99   0   0   0   0   0   0 | 8
   0   0   0   0   0   0   0   1   0   0  99 | 9
   0   0   0   0 100   0   0   0   0   0   0 | 10
 100   0   0   0   0   0   0   0   0   0   0 | 11
   1   0  99   0   0   0   0   0   0   0   0 | 12
   0  99   0   0   0   0   0   0   0   0   0 | 13
   0  99   0   0   0   0   0   0   0   0   0 | 14
   0   0   0  99   0   0   0   0   0   0   0 | 15
   0   0   0   0 100   0   0   0   0   0   0 | 16
   0   0   0   0 100   0   0   0   0   0   0 | 17
   0   0 100   0   0   0   0   0   0   0   0 | 18
   0   0   0   0   0 100   0   0   0   0   0 | 19
   0   0   0   0   0   0 100   0   0   0   0 | 20
   0   0   0   0 100   0   0   0   0   0   0 | 21
   0   0   0   0   0   0   0  99   0   0   1 | 22
   0   0   0   0   0   0   0   0  97   0   0 | 23
   0   0   0   0   0   0   0   0  99   0   0 | 24
  99   0   0   0   0   0   0   0   0   0   0 | 25
   0   0  99   0   0   0   0   0   0   0   0 | 26
  99   0   0   0   0   0   0   0   0   0   0 | 27
   0   0   0   0 100   0   0   0   0   0   0 | 28
   0   0   0   0 100   0   0   0   0   0   0 | 29
   0   0   0   0   0 100   0   0   0   0   0 | 30
   0   0   0   0 100   0   0   0   0   0   0 | 31

Cluster  0 <-- 1
Cluster  1 <-- 2
Cluster  2 <-- 18
Cluster  3 <-- 15
Cluster  4 <-- 5
Cluster  5 <-- 3
Cluster  6 <-- 20
Cluster  7 <-- 22
Cluster  8 <-- 4
Cluster  9 <-- 6
Cluster 10 <-- 9

Incorrectly clustered instances :       1994.0   64.3226 %
```



DBSCAN worked very well for $\epsilon = 0.03$ and MinPts = 45 as shown below :

```
Class attribute: label
Classes to Clusters:

  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15  <-- assigned to cluster
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 1
 66  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 2
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 3
  0  0  0  0  0  0  0  0  0  0  0  0  0 65  0  0 | 4
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 5
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 6
  0  0  0  0  0  0  0  0  0  0  0  0  0  0 70  0 | 7
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 8
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 69 | 9
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 10
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 11
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 12
  0 44  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 13
  0  0 54  0  0  0  0  0  0  0  0  0  0  0  0  0 | 14
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 15
  0  0  0 58  0  0  0  0  0  0  0  0  0  0  0  0 | 16
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 17
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 18
  0  0  0  0 45  0  0  0  0  0  0  0  0  0  0  0 | 19
  0  0  0  0  0 69  0  0  0  0  0  0  0  0  0  0 | 20
  0  0  0  0  0  0 49  0  0  0  0  0  0  0  0  0 | 21
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 22
  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0 | 23
  0  0  0  0  0  0  0 63  0  0  0  0  0  0  0  0 | 24
  0  0  0  0  0  0  0  0 60  0  0  0  0  0  0  0 | 25
  0  0  0  0  0  0  0  0  0 58  0  0  0  0  0  0 | 26
  0  0  0  0  0  0  0  0  0  0 74  0  0  0  0  0 | 27
  0  0  0  0  0  0  1  0  0  0  0 71  0  0  0  0 | 28
  0  0  0  0  0  0  0  0  0  0  0  0 61  0  0  0 | 29
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 30
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | 31

Cluster  0 <-- 2
Cluster  1 <-- 13
Cluster  2 <-- 14
Cluster  3 <-- 16
Cluster  4 <-- 19
Cluster  5 <-- 20
Cluster  6 <-- 21
Cluster  7 <-- 24
Cluster  8 <-- 25
Cluster  9 <-- 26
Cluster 10 <-- 27
Cluster 11 <-- 28
Cluster 12 <-- 29
Cluster 13 <-- 4
Cluster 14 <-- 7
Cluster 15 <-- 9

Incorrectly clustered instances :       3.0       0.0968 %
```



But actually, it considered several points as Noise points and hence we get a very good putiry score.

But $\epsilon = 0.03$ and MinPts < 45, it didn't converge even after running for a long time.

HIERARCHIAL CLUSTERING : WARD LINKAGE

The results are as follows :
Number of Clusters = 10 :

```
Class attribute: label
Classes to Clusters:

  0   1   2   3   4   5   6   7   8   9  <-- assigned to cluster
100   0   0   0   0   0   0   0   0   0 | 1
  0 100   0   0   0   0   0   0   0   0 | 2
  0   0 100   0   0   0   0   0   0   0 | 3
  0   0   0 100   0   0   0   0   0   0 | 4
  0   0   0   0 100   0   0   0   0   0 | 5
100   0   0   0   0   0   0   0   0   0 | 6
  0   0   0 100   0   0   0   0   0   0 | 7
  0   5   0   0   0  95   0   0   0   0 | 8
  0   0   0   1   0   0  99   0   0   0 | 9
  0   0   0   0   0 100   0   0   0   0 | 10
100   0   0   0   0   0   0   0   0   0 | 11
  1   0   0   0   0   0   0  99   0   0 | 12
  0 100   0   0   0   0   0   0   0   0 | 13
  0 100   0   0   0   0   0   0   0   0 | 14
  0   0   0   0   0   0   0 100   0   0 | 15
  0   0   0   0   0   0   0   0 100   0 | 16
  0   0   0   0   0   0   0   0 100   0 | 17
  0   0   0   0   0   0   0 100   0   0 | 18
  0   0 100   0   0   0   0   0   0   0 | 19
  0   0   0   0   0   0   0   0   0 100 | 20
  0   0   0   0   4  96   0   0   0   0 | 21
  0   0   0   0   0   0 100   0   0   0 | 22
  0   0   0 100   0   0   0   0   0   0 | 23
  0   0   0  98   0   0   0   2   0   0 | 24
100   0   0   0   0   0   0   0   0   0 | 25
  0   0   0   0   0   0   0 100   0   0 | 26
 99   1   0   0   0   0   0   0   0   0 | 27
  0   0   0   0   0 100   0   0   0   0 | 28
  0   0   0   0   0 100   0   0   0   0 | 29
  0   0 100   0   0   0   0   0   0   0 | 30
  0   0   0   0 100   0   0   0   0   0 | 31

Cluster  0 <-- 1
Cluster  1 <-- 2
Cluster  2 <-- 3
Cluster  3 <-- 4
Cluster  4 <-- 5
Cluster  5 <-- 10
Cluster  6 <-- 22
Cluster  7 <-- 15
Cluster  8 <-- 16
Cluster  9 <-- 20

Incorrectly clustered instances :       2100.0   67.7419 %
```

Number of Clusters = 20 :

```
Classes to Clusters:

  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  <-- assigned to cluster
100   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 | 1
  0  95   5   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 | 2
  0   0   0  94   6   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 | 3
  0   0   0   0   0  96   4   0   0   0   0   0   0   0   0   0   0   0   0   0 | 4
  0   0   0   0   0   0   0 100   0   0   0   0   0   0   0   0   0   0   0   0 | 5
  1   0   0   0   0   0   0   0  99   0   0   0   0   0   0   0   0   0   0   0 | 6
  0   0   0   0   0   0   0   0   0 100   0   0   0   0   0   0   0   0   0   0 | 7
  0   0   5   0   0   0   0   0   0   0  95   0   0   0   0   0   0   0   0   0 | 8
  0   0   0   0   0   0   0   0   0   1   0  99   0   0   0   0   0   0   0   0 | 9
  0   0   0   0   0   0   0   0   0   0  97   0   3   0   0   0   0   0   0   0 | 10
 98   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0 | 11
  0   0   0   0   0   0   0   0   0   0   0   0   0  96   3   1   0   0   0   0 | 12
  0 100   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 | 13
  0   1  99   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 | 14
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 100   0   0   0 | 15
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 100   0   0 | 16
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 100   0   0 | 17
  0   0   0   0   0   0   0   0   0   0   0   0   0 100   0   0   0   0   0   0 | 18
  0   0   0   0 100   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 | 19
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 100   0 | 20
  0   0   0   0   0   0   0   4   0   0   0   0  96   0   0   0   0   0   0   0 | 21
  0   0   0   0   0   0   0   0   0   0   0   2   0   0   0   0   0   0   0  98 | 22
  0   0   0   0   0   2  96   0   0   2   0   0   0   0   0   0   0   0   0   0 | 23
  0   0   0   0   0  98   0   0   0   0   0   0   0   2   0   0   0   0   0   0 | 24
100   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 | 25
  0   0   0   0   0   0   0   0   0   0   0   0   0   1  99   0   0   0   0   0 | 26
  2   1   0   0   0   0   0   0   0   0   0   0   0   0   0  97   0   0   0   0 | 27
  0   0   0   0   0   0   0   0   0   0   0   0 100   0   0   0   0   0   0   0 | 28
  0   0   0   0   0   0   0   0   0   0 100   0   0   0   0   0   0   0   0   0 | 29
  0   0   0 100   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 | 30
  0   0   0   0   0   0   0 100   0   0   0   0   0   0   0   0   0   0   0   0 | 31

Cluster  0 <-- 1
Cluster  1 <-- 13
Cluster  2 <-- 14
Cluster  3 <-- 30
Cluster  4 <-- 19
Cluster  5 <-- 24
Cluster  6 <-- 23
Cluster  7 <-- 5
Cluster  8 <-- 6
Cluster  9 <-- 7
Cluster 10 <-- 29
Cluster 11 <-- 9
Cluster 12 <-- 28
Cluster 13 <-- 18
Cluster 14 <-- 26
Cluster 15 <-- 27
Cluster 16 <-- 15
Cluster 17 <-- 16
Cluster 18 <-- 20
Cluster 19 <-- 22

Incorrectly clustered instances :       1115.0   35.9677 %
```

NAIVE BAYES

Design and implementation of a Bayesian Spam Filter that classifies email messages as either spam(unwanted) or ham (useful), that is, $y_i \epsilon\ spam, ham$ using a Naive Bayes Classifier for the following four scenarios:

1. MAXIMUM LIKELIHOOD ESTIMATION ASSUMING LIKELIHOOD: MULTINOMIAL

This has been implemented using sklearn as well as from scratch in python.

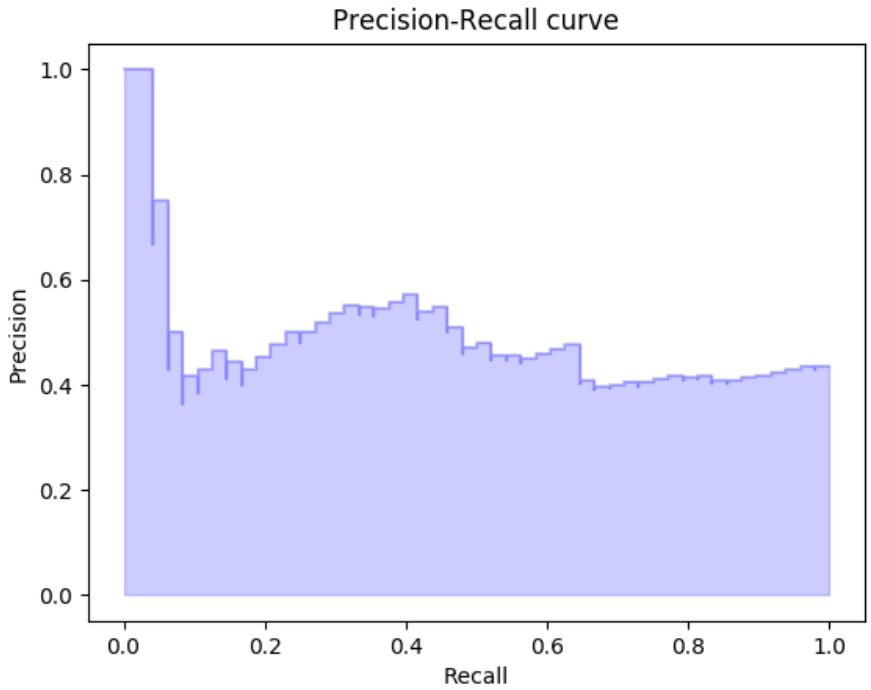
FROM SCRATCH

The results obtained are as follows :

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Legit	98	95	97
Spam 2	94	98	96
Avg	96	96	96

The PR curve has been plotted by giving the difference of the calculated scores with the minimum of all scores and dividing by modulus of the minimum.

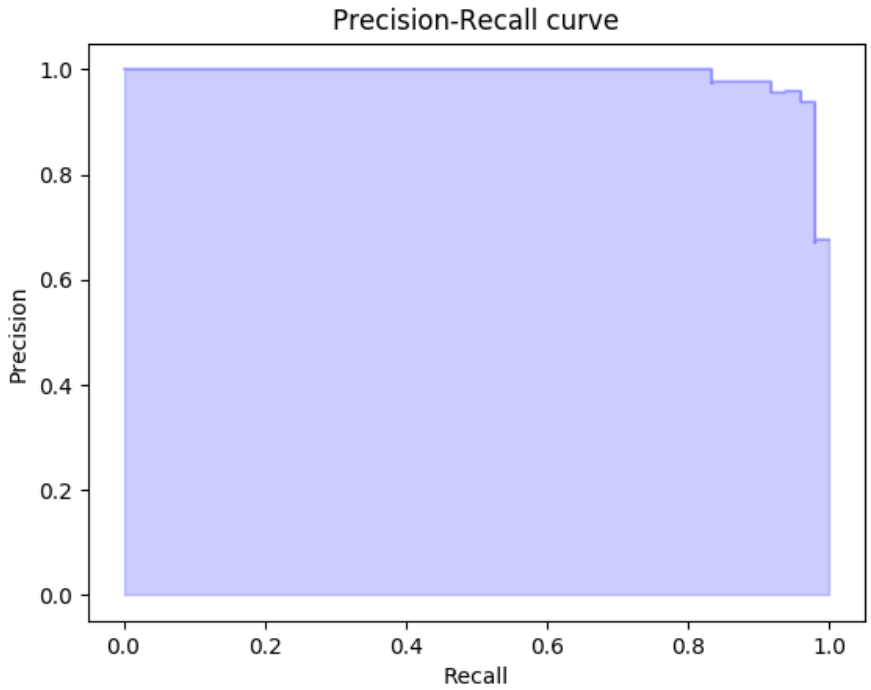
PR Curve for spam :



On calculating the probability as the score of legit divided by the score of legit and spam (since score is negative, we take the class with the less negative score and hence the relation), the result obtained is as follows :

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Legit	97	97	97
Spam 2	96	96	96
Avg	96	96	96

PR Curve for spam :

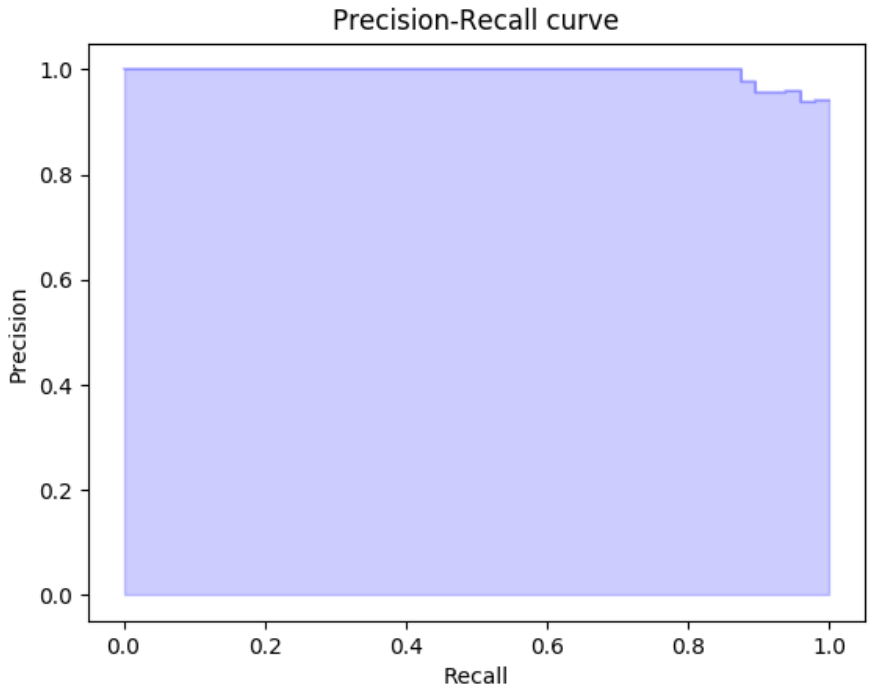


USING SKLEARN

The results obtained are as follows :

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Spam	94	95	94
Legit 2	94	92	93
Avg	94	94	94

PR Curve for spam :

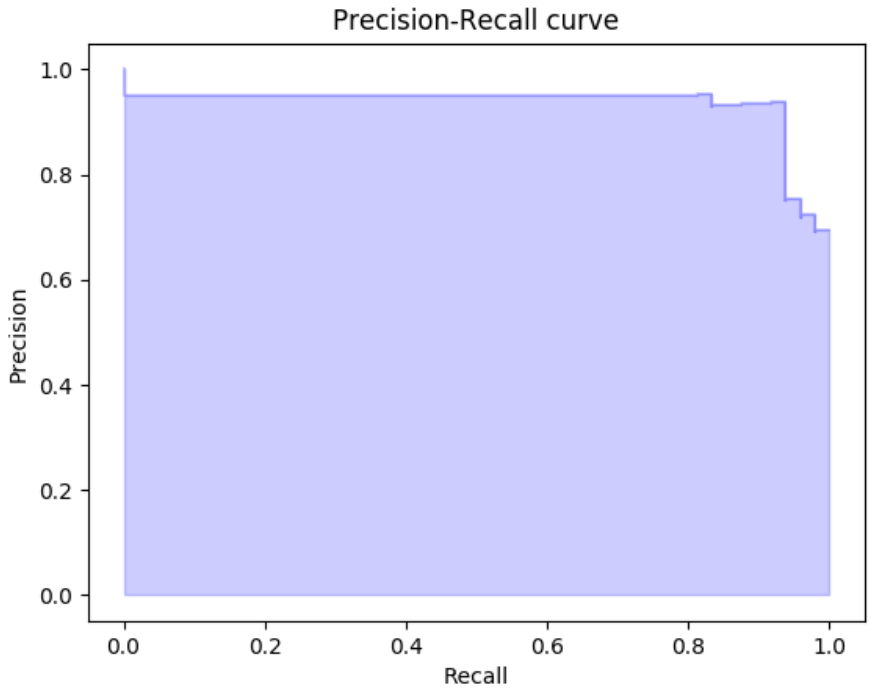


2. MAXIMUM LIKELIHOOD ESTIMATION ASSUMING LIKELIHOOD: BERNOULLI

The results obtained are as follows :

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Legit	91	98	94
Spam 2	98	88	92
Avg	94	94	94

PR Curve for spam :



3. Bayesian Parameter Estimation assuming the prior: Dirichlet

The Dirichlet distribution is given by,

$$p(P = \{p_i\} \mid \alpha_i) = \frac{\prod_i \Gamma(\alpha_i)}{\Gamma(\sum_i \alpha_i)} \prod_i p_i^{\alpha_i - 1}$$

The conditioning on each word is given by,

$$\theta_{\text{the}}^{MAP} = \frac{N_{\text{the}} + \alpha_{\text{the}} - 1}{\sum_{k=1}^{K} N_k + \sum_{k=1}^{K} \alpha_k - K}$$

when the number of words in the vocabulary is K. Here, I have defined twi different Dirichlet distribution for the two classes (Spam and Legit) and hence estimated the the class of each document. The code has been written from scratch and is a variation of the Multinomial Naive Bayes algorithm.

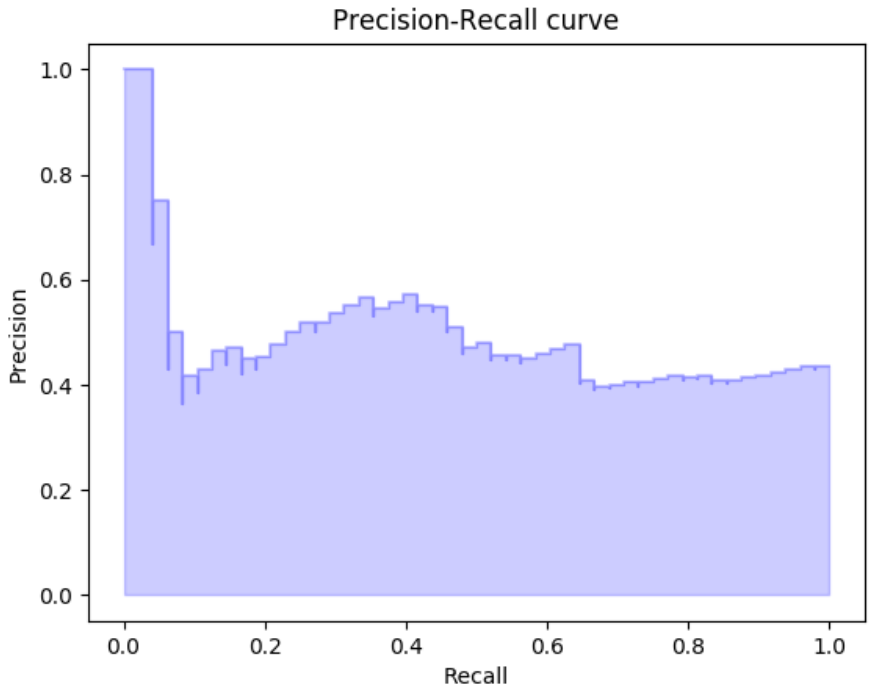
The PR curve has been plotted by giving the difference of the calculated scores with the minimum of all scores and dividing by modulus of the minimum.

The results obtained are as follows :

First run :

Labels	Per Class Estimates		
	Precision	**Recall**	**F-measure**
Spam	95	94	94
Legit	94	92	93
Avg	94	94	94

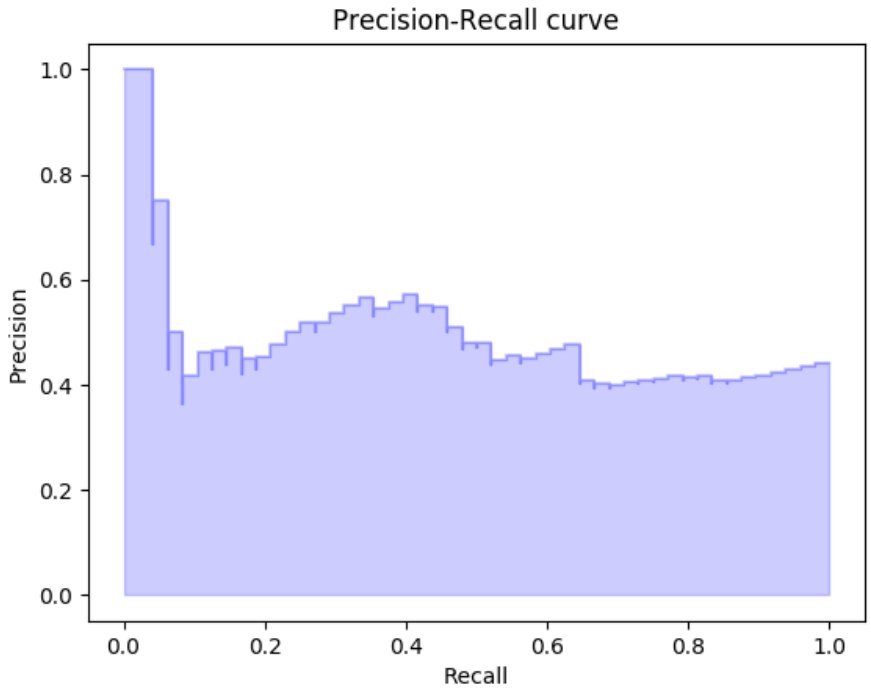
PR Curve for spam :



Second run :

Labels	Per Class Estimates		
	Precision	**Recall**	**F-measure**
Spam	97	97	97
Legit	96	96	96
Avg	96	96	96

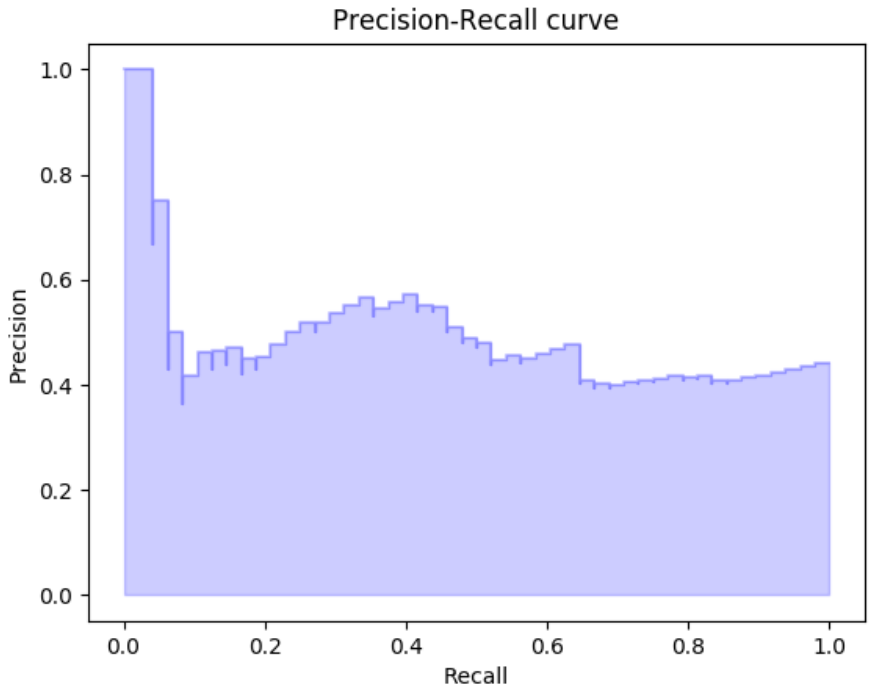
PR Curve for spam :



Third run :

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Spam	92	97	94
Legit	96	90	92
Avg	94	94	94

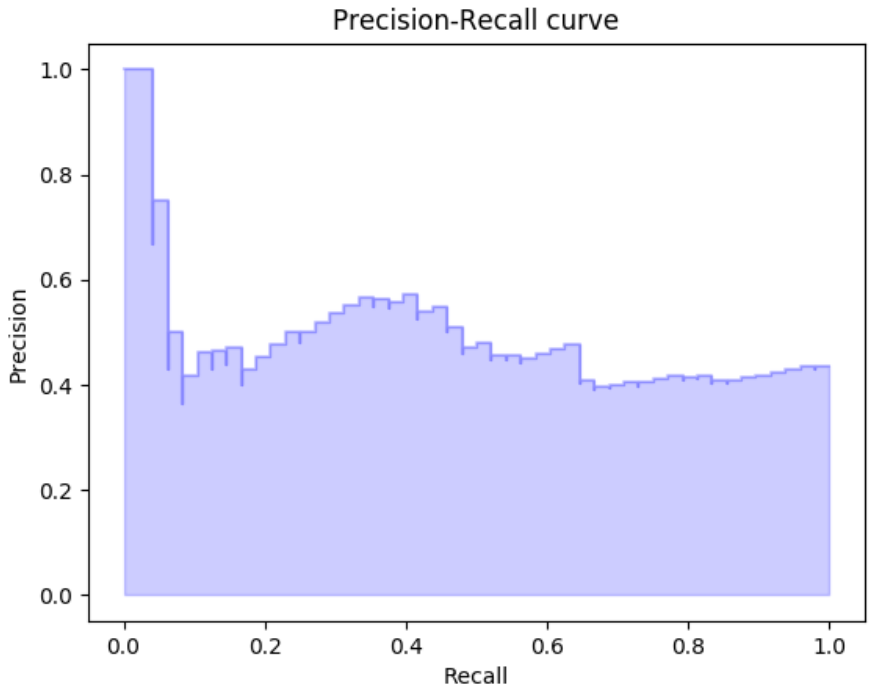
PR Curve for spam :



Fourth run :

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Spam	94	97	95
Legit	96	92	94
Avg	95	95	95

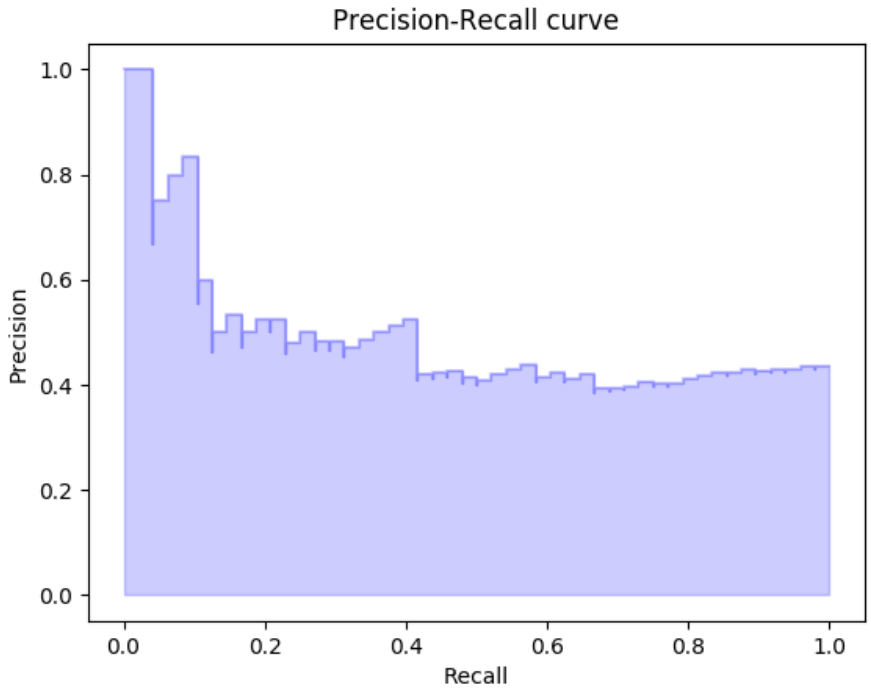
PR Curve for spam :



Fifth run :

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Legit	95	95	95
Spam	94	94	94
Avg	95	95	95

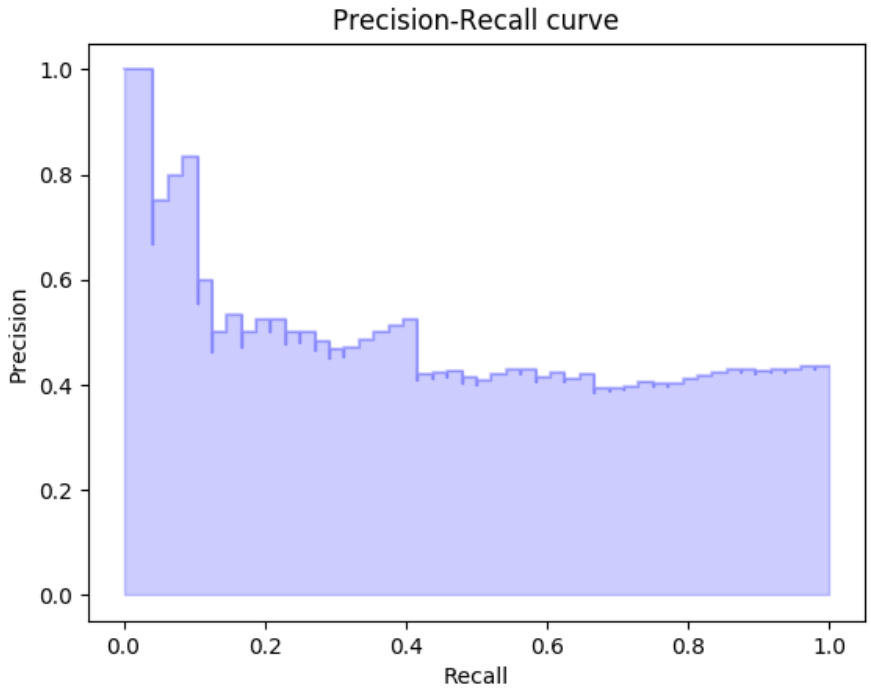
PR Curve for spam :



Sixth run :

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Legit	98	97	98
Spam	96	98	97
Avg	97	97	97

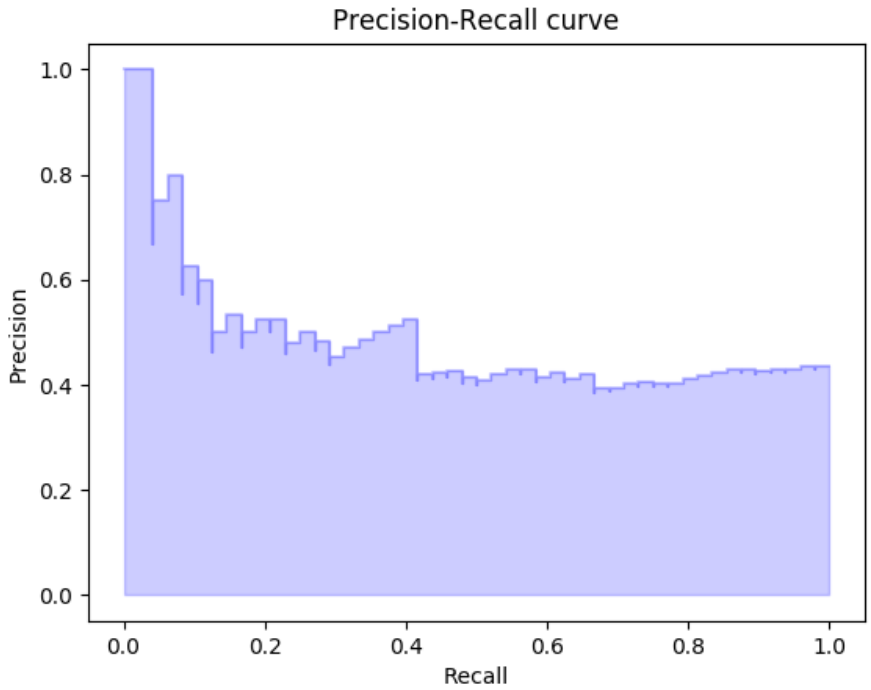
PR Curve for spam :



Seventh run :

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Spam	98	95	97
Legit	94	98	96
Avg	96	96	96

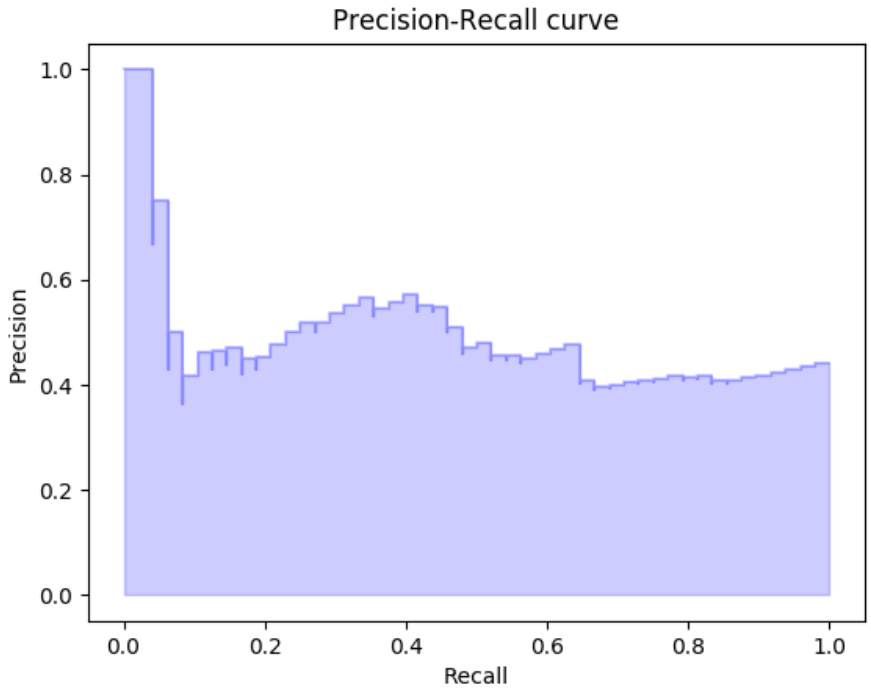
PR Curve for spam :



Eighth run :

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Legit	94	98	96
Spam	98	92	95
Avg	96	95	95

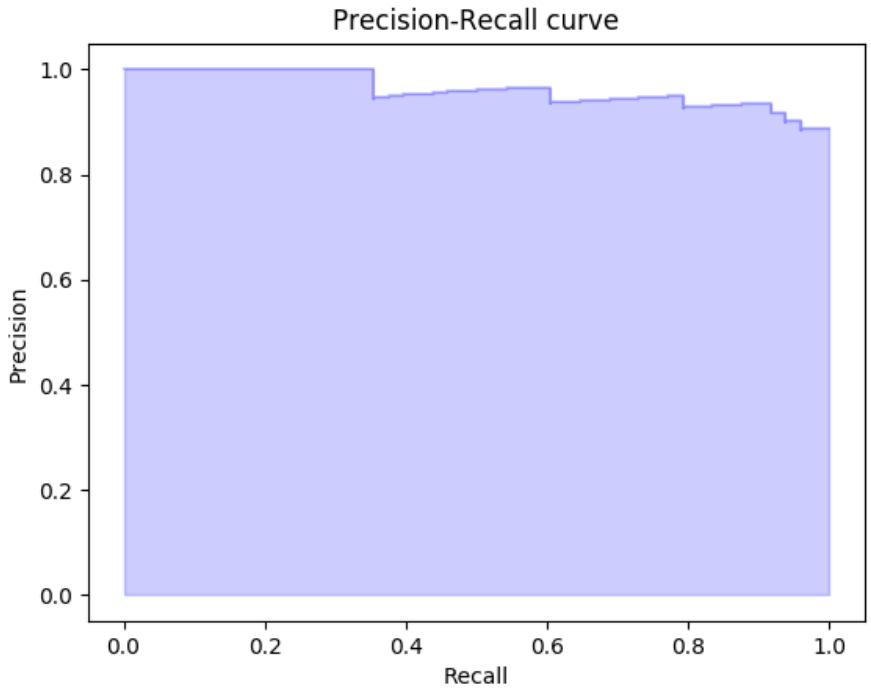
PR Curve for spam :



On calculating the probability as the score of legit divided by the score of legit and spam (since score is negative, we take the class with the less negative score and hence the relation), the result obtained is as follows :

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Legit	97	94	95
Spam	92	96	94
Avg	95	95	95

PR Curve for spam :



4. Bayesian Parameter Estimation, assuming the class prior: Beta

The beta distribution is given by :

$$p(p \mid \alpha, \beta) = \frac{1}{B(\alpha, \beta)} p^{\alpha-1}(1-p)^{\beta-1}$$

Here, the condition has been placed on the classes and not the words. The output has been determined by supplying the class priors to the Bernoulli Naive Bayes function in sklearn. The class priors are calculated as :

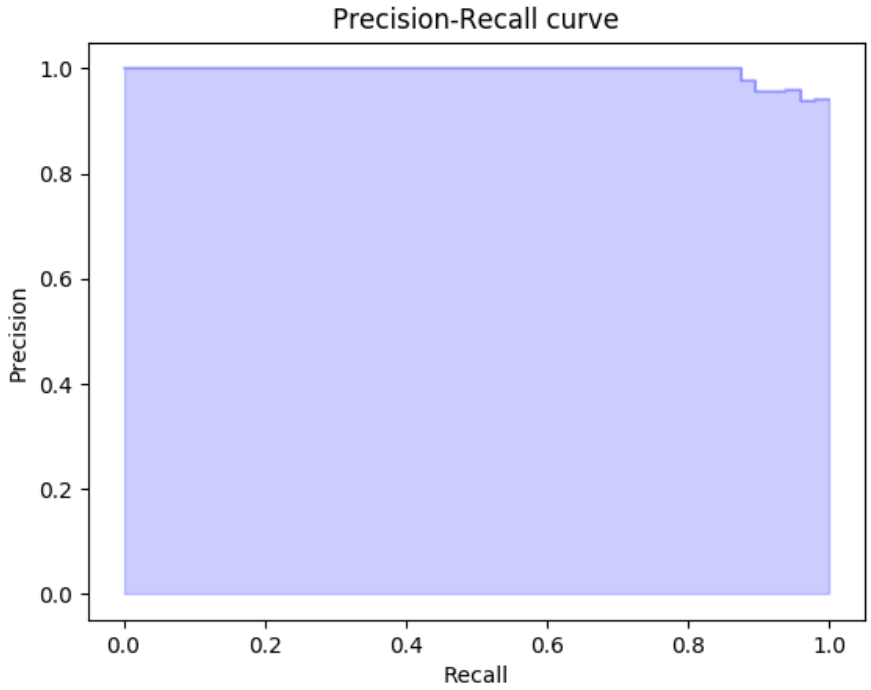
$$\theta^{MAP} = \frac{\#H + \alpha - 1}{\#T + \#H + \alpha + \beta - 2}$$

Note : The condition can also be placed on each word, wherein we'll have to code Bernouilli Naive Bayes from scratch.

The results obtained are as follows :
$\alpha = 7 \& \beta = 2$

Labels	Per Class Estimates		
	Precision	**Recall**	**F-measure**
Spam	91	98	94
Legit 2	98	88	92
Avg	94	94	94

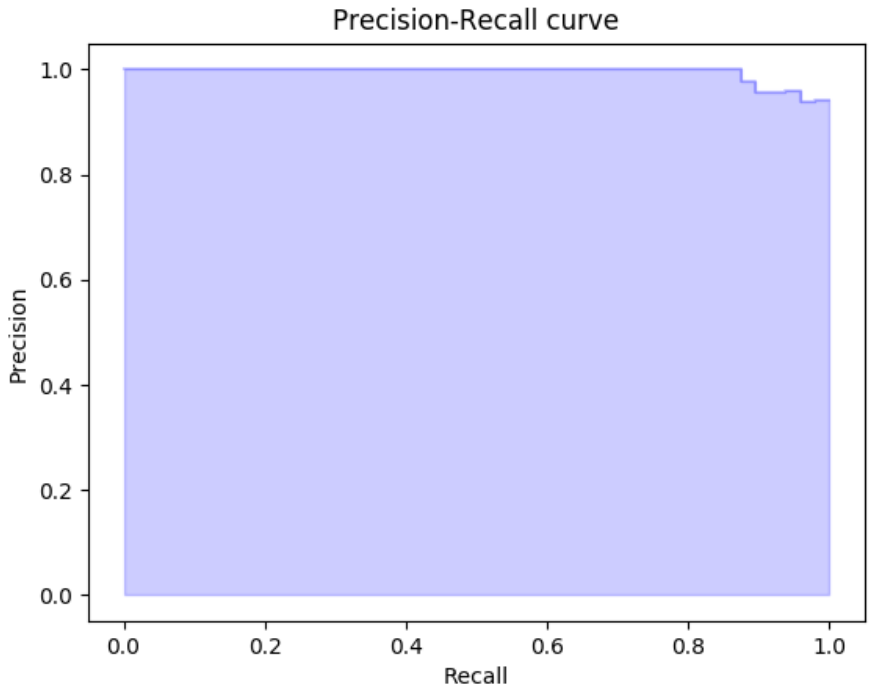
PR Curve for spam :



$\alpha = 10000 \& \beta = 1$

Labels	Per Class Estimates		
	Precision	**Recall**	**F-measure**
Spam	86	100	92
Legit 2	100	79	88
Avg	92	91	91

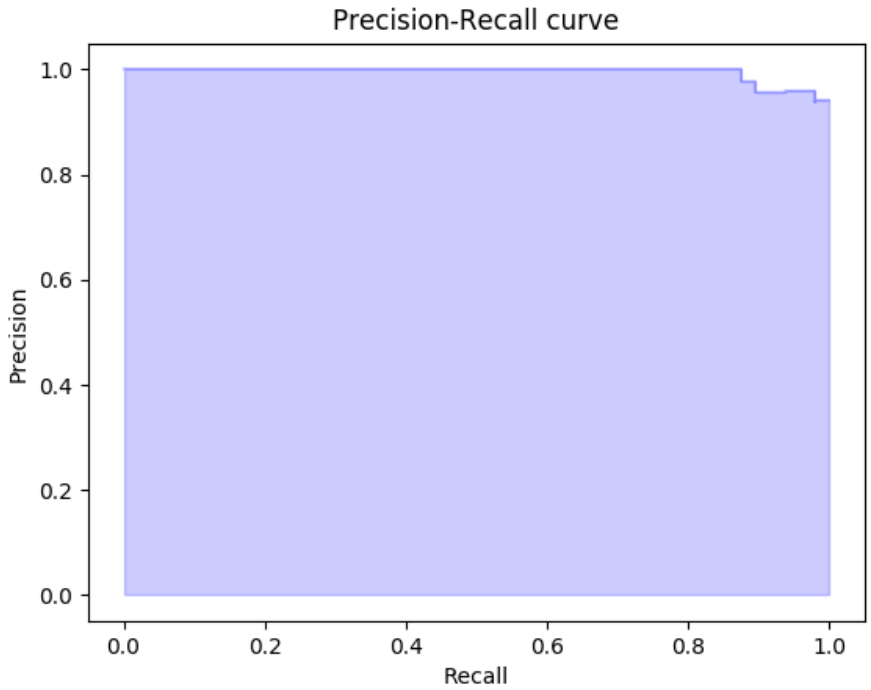
PR Curve for spam :



$\alpha = 1 \& \beta = 10000$

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Legit	94	97	95
Spam 2	96	92	94
Avg	95	95	95

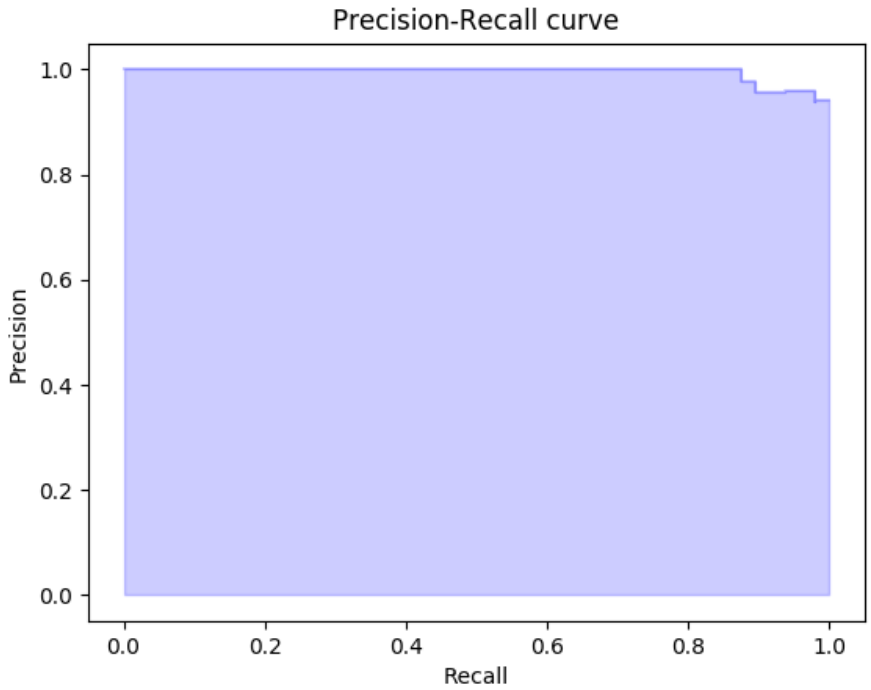
PR Curve for spam :



$\alpha = 0.001 \& \beta = 150$

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Spam	90	100	95
Legit	100	85	92
Avg	94	94	94

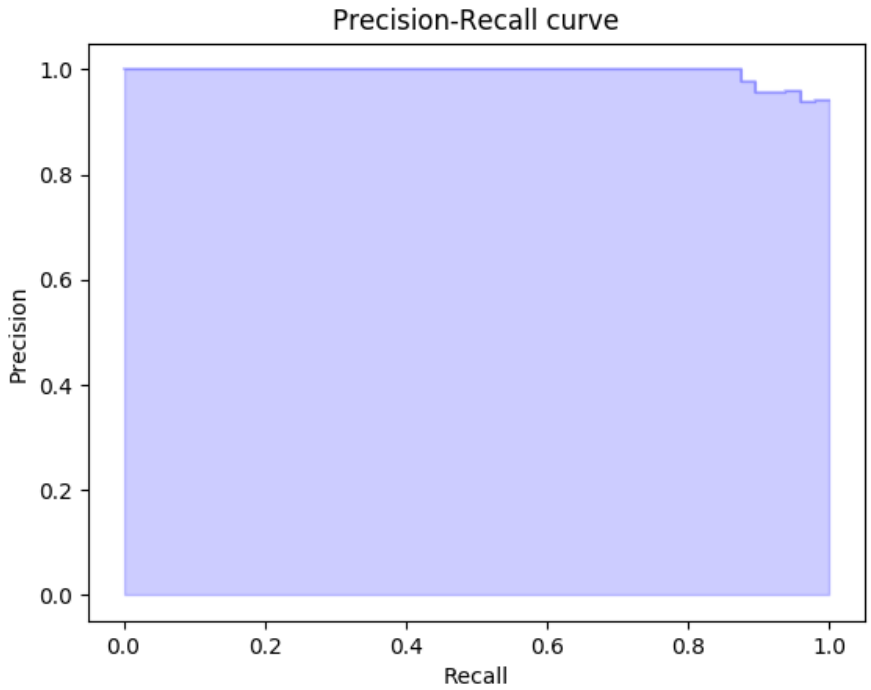
PR Curve for spam :



$\alpha = 150 \& \beta = 0.001$

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Spam	90	100	95
Legit	100	85	92
Avg	94	94	93

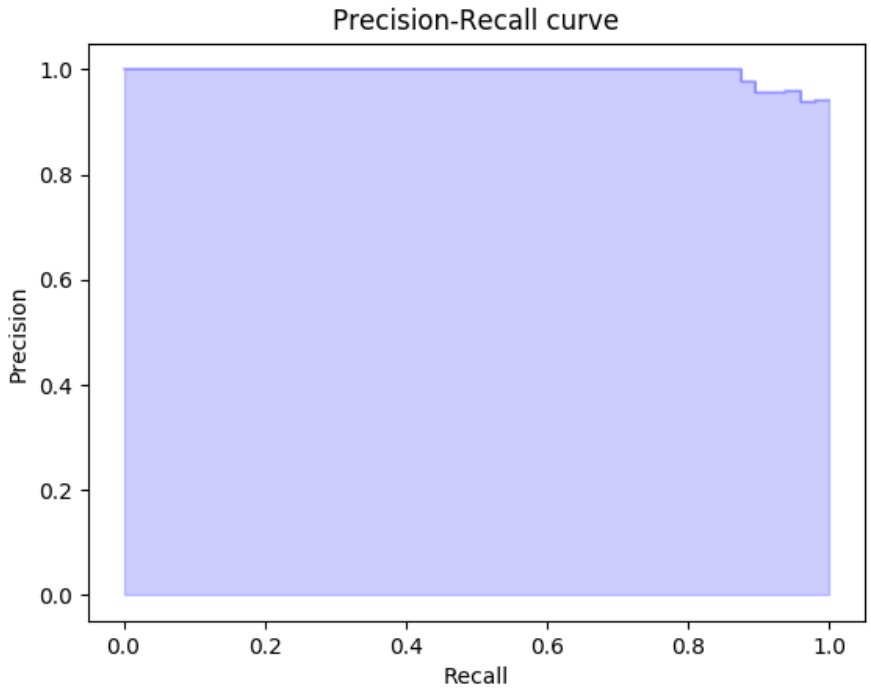
PR Curve for spam :



$\alpha = 1000000 \& \beta = 0.001$

Labels	**Per Class Estimates**		
	Precision	**Recall**	**F-measure**
Legit	82	100	90
Spam 2	100	73	84
Avg	90	88	88

PR Curve for spam :



Note : The class priors for Beta Distribution were also supplied to the Multinomial Naive Bayes algorithm and the results can be found in the folder containing Code->q2

REFERENCES

- https://www.cs.cmu.edu/~epxing/Class/10701-08s/recitation/dirichlet.pdf
- http://aritter.github.io/courses/5523_slides/dirichlet_nb.pdf