Data Mining I

Summer semester 2017

Data Mining Project 2- Unsupervised Learning

Group members with Matriculation numbers:
1. Shruthi Shetty (4850454)
2. Emetis Niazmand (4850519)
3. Suma Kori (4850522)

University: TU Braunschweig
Study/ Course : ITIS (Semester 1)
E-Mail: shruthi.shetty@tu-braunschweig.de , e.niazmand@tu-braunschweig.de , suma.kori93@gmail.com

3. Tasks

3.1 Dataset understanding

Attributes: ID, RI, Na, Mg, Al, Si, K, Ca, Ba, Fe, class

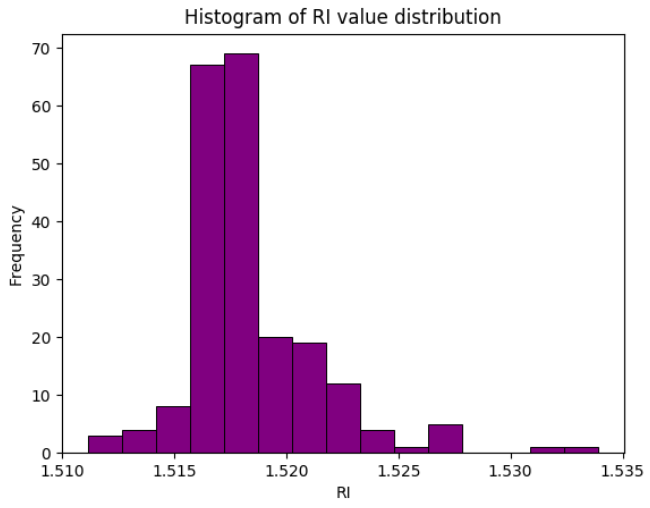
Class attribute: Class attribute have values 1 to 7.
1. building_windows_float_processed
2. building_windows_non_float_processed
3. vehicle_windows_float_processed
4. vehicle_windows_non_float_processed (none in this database)
5. containers
6. tableware
7. Headlamps

Number of instances: 412

- **Uni-variate analysis: Plot the value distribution of each feature**

We plot the value distribution of each feature in form of histogram using 15 bins.
The following are the observations obtained from the histogram distribution of each feature:

1. Histogram of feature RI



We observe that the distribution of RI in the bins with RI values between 1.516 and 1.518 have the highest frequencies. The frequency of RI above 1.518 goes on reducing constantly to the right. This frequency is 0 between RI values of 1.528 and 1.531.

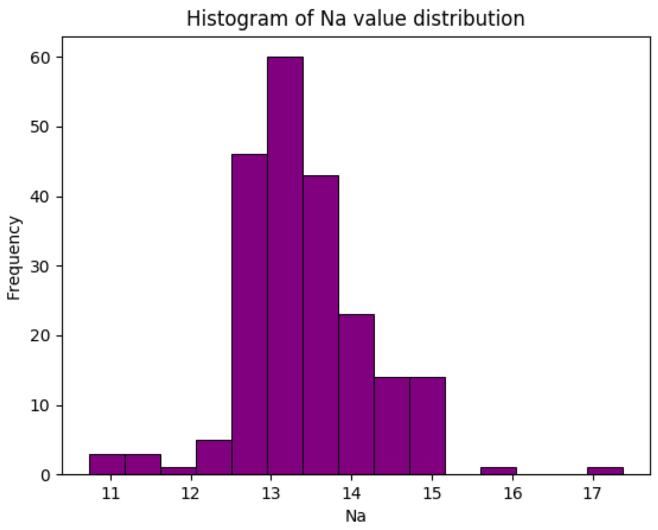
Range of RI :

Min value(RI): 1.511150

Max value(RI): 1.533930

The histogram is right skewed.

2. Histogram of feature Na



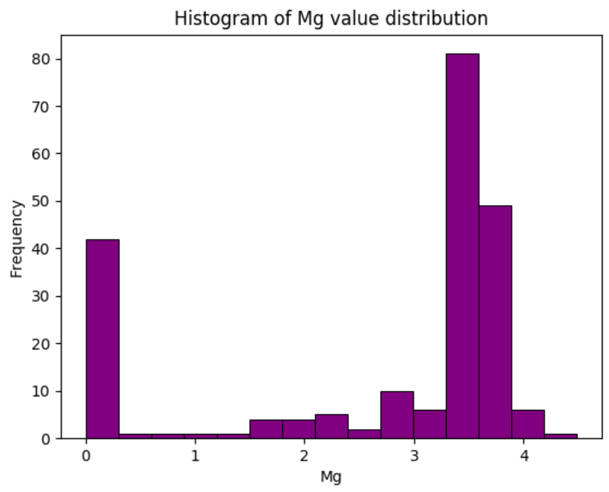
We observe that the distribution of Na in the bin with Na values between 13 to 13.5 has the highest frequency of 60. The frequency of Na goes on reducing to the right and to the left.

Range of Na :

Min value(Na): 10.73

Max value(Na): 17.38

3. Histogram of feature Mg



We observe that the distribution of Mg in the bin with Mg values between 3.25 and 3.5 has the highest frequency of 80. The frequency of Mg goes on reducing to the left and have very low frequencies until the bin with Mg value 0-0.25 with a frequency of 40.

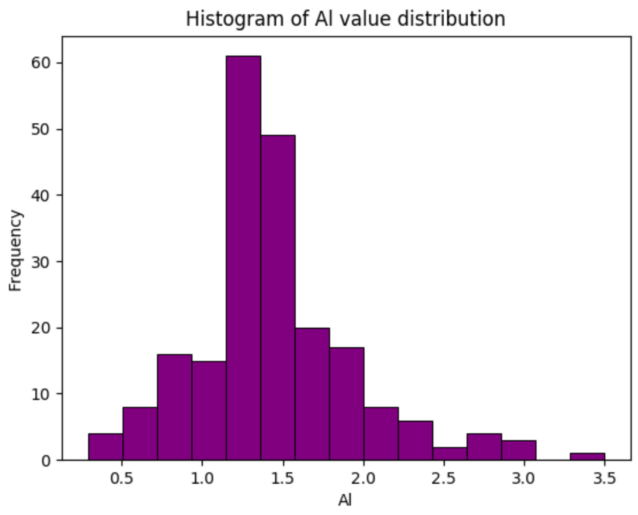
<u>Range of Mg :</u>

Min value(Mg): 0

Max value(Mg): 4.49

The histogram is left skewed.

4. Histogram of feature Al



We observe that in the distribution of Al the highest frequencies are in range of 1.25 and 1.5. The frequencies on either sides of these values goes on reducing.

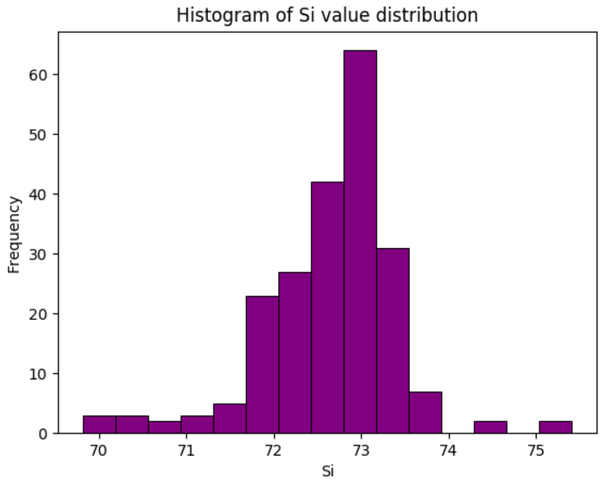
<u>Range of Al :</u>

Min value(Al): 0.29

Max value(Al): 3.5

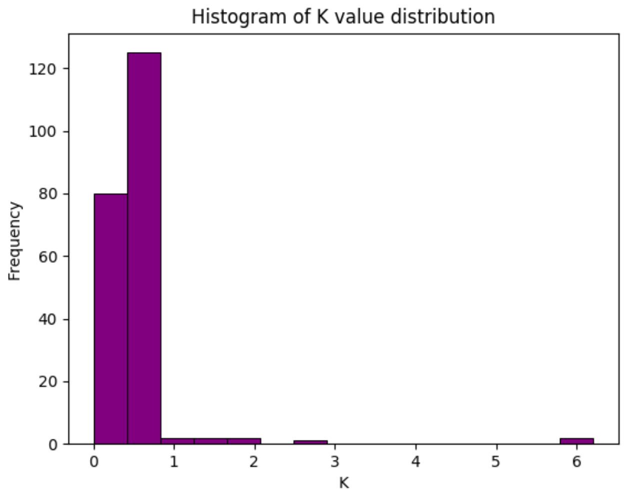
The histogram is right skewed.

5. Histogram of feature Si



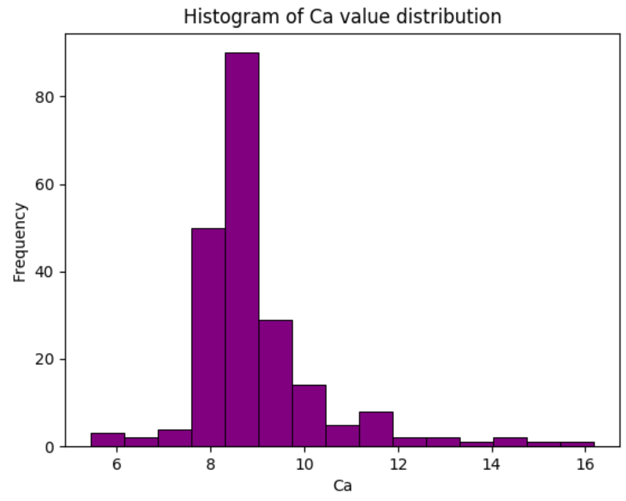
We observe that the frequency is highest in the bin with Si value of 73 with a frequency of more than 60. The frequency on either sides of this value goes on reducing.
<u>Range of Si:</u>
Min value(Si): 69.81
Max value(Si): 75.41

6. Histogram of feature K



We observe that the frequency distribution is the highest at the bin with a K value of 0.5 – 1. The frequencies at all bins to the right is very low and is approximately equal to 1 and have 0 frequencies as well for certain values of K.
<u>Range of K</u> :
Min value(K): 0
Max value(K): 6.21
This histogram is right skewed.

7. Histogram of feature Ca



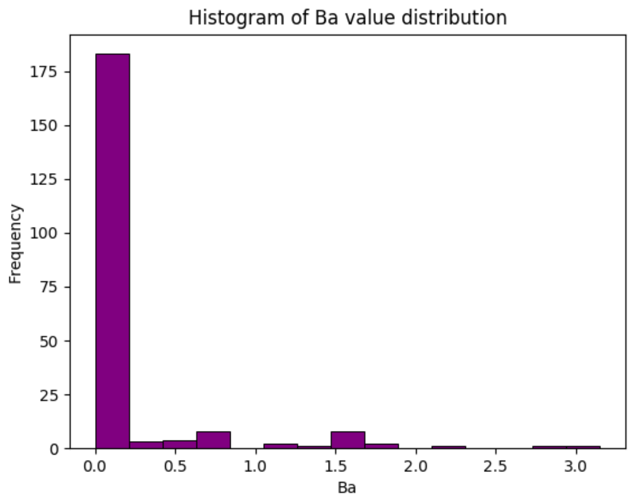
We observe that the frequency of Ca is maximum (with a frequency of more than 80) at the bin with Ca values of 8.4-9. The frequencies goes on reducing on either sides.

<u>Range of Ca</u>:

Min value(Ca): 5.43

Max value(Ca): 16.19

8. Histogram of feature Ba



From this histogram we observe that more than half of instances have Ba values in the range of 0 to 0.3 with a highest frequency of more than 175.

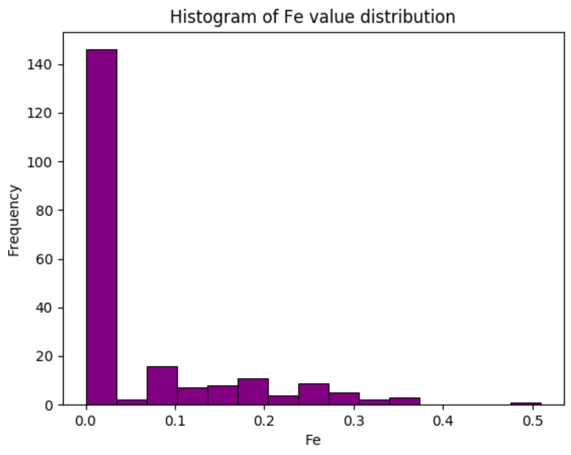
The frequencies for the remaining Ba values are very low and are as low as 0 for some values of Ba.

<u>Range of Ba :</u>

Min value(Ba): 0

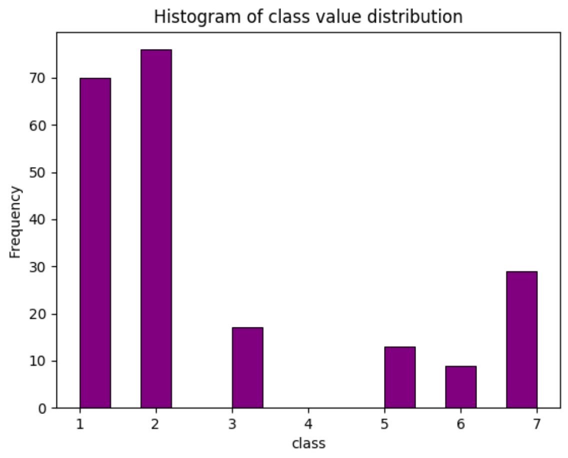
Max value(Ba): 3.15

9. Histogram of feature Fe



We observe that more than half of instances have Fe values in the range of 0 to 0.04 with a highest frequency of more than 140.
The frequencies for the remaining Fe values are very low and are as low as 0 Fe values between 0.38 – 0.48.
Range of Fe :
Min value(Fe): 0
Max value(Fe): 0.51

10. Histogram of ‘Class’



We observe that classes 1(building_windows_float_processed) and 2 (building_windows_non_float_processed) have highest frequencies.

The frequencies of other classes are relatively low.

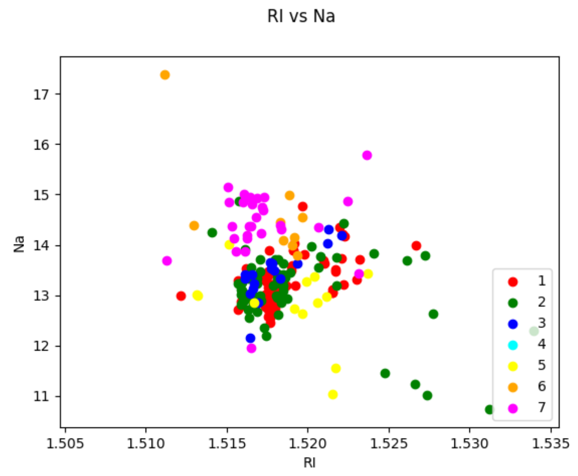
Another key observation made here is that **there are no instances** for **class 4** (vehicle_windows_non_float_processed)

- **<u>Bi-variate analysis: Checking for correlations</u>**

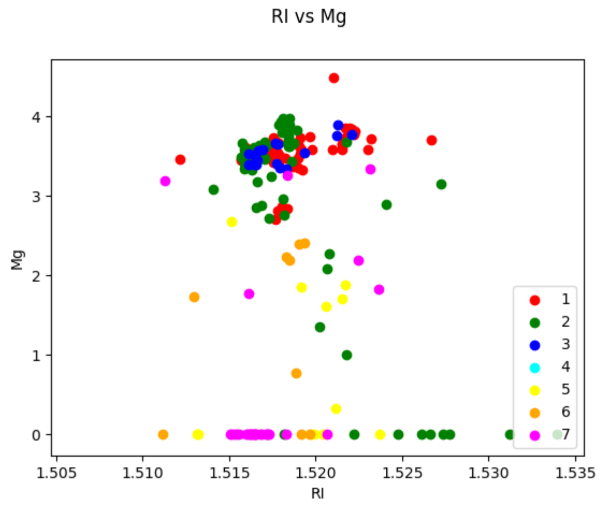
  If $r_{XY} > 0$ - positive correlation
  If $r_{XY} < 0$ - negative correlation
  If $r_{XY} \sim 0$ - no correlation/ independent

  Below are correlation plots of RI with other features :

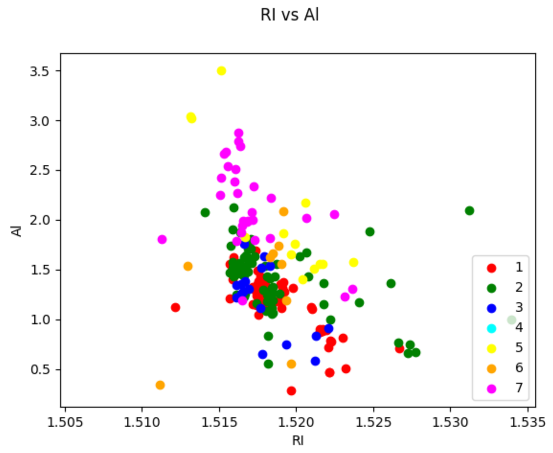
  1. RI vs Na :
     RI and Na have a negative correlation with each other.



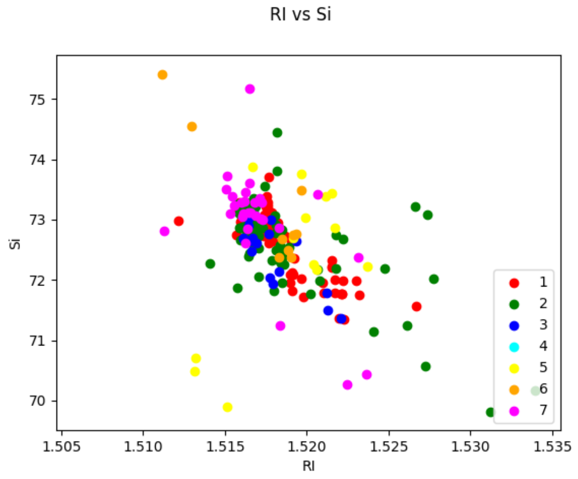
  2. RI vs Mg :
     RI and Mg have a negative correlation with each other.



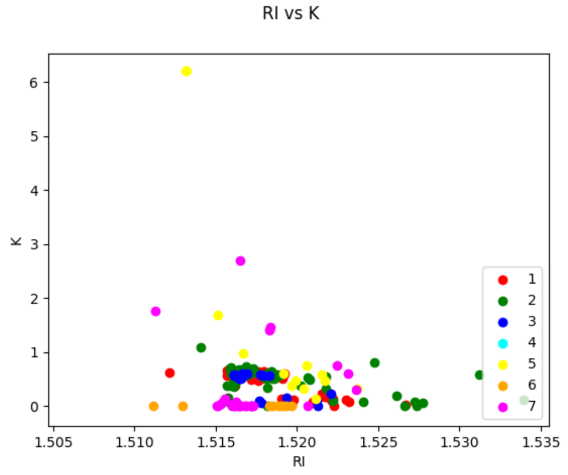
3. RI vs Al :
   RI and Al have a negative correlation with each other.



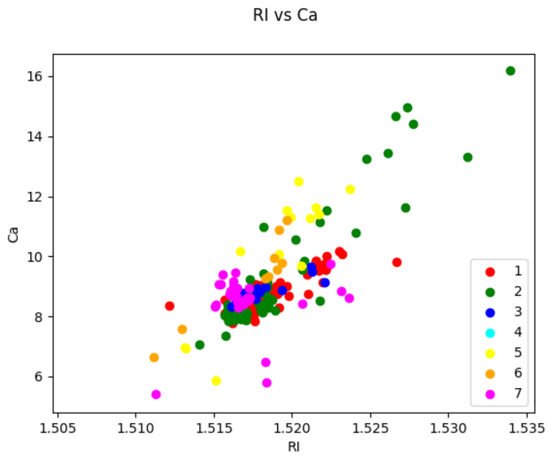
4. RI vs Si:
   RI and Si have a negative correlation with each other.



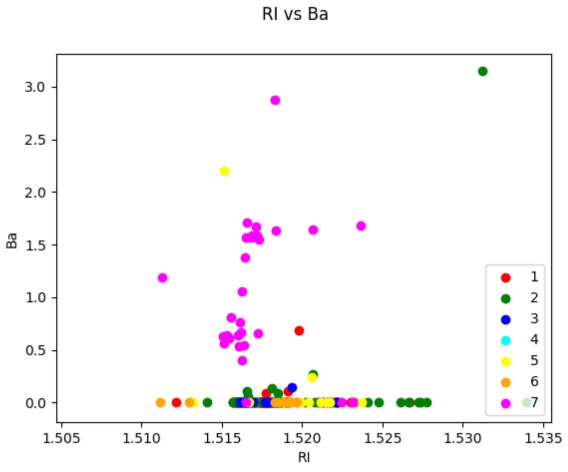
5. RI vs K:
   RI and K have a negative correlation with each other.



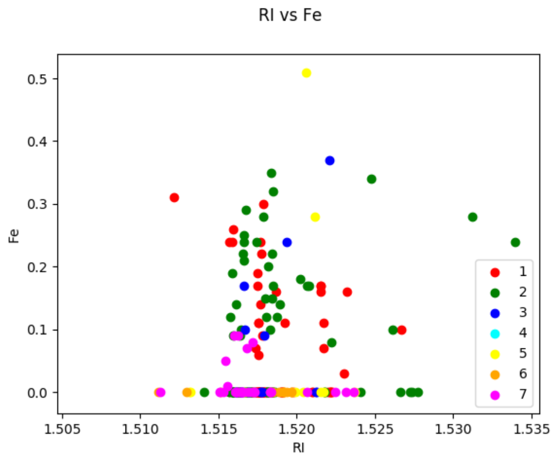
6. RI vs Ca:
   RI and Ca have a positive correlation with each other.



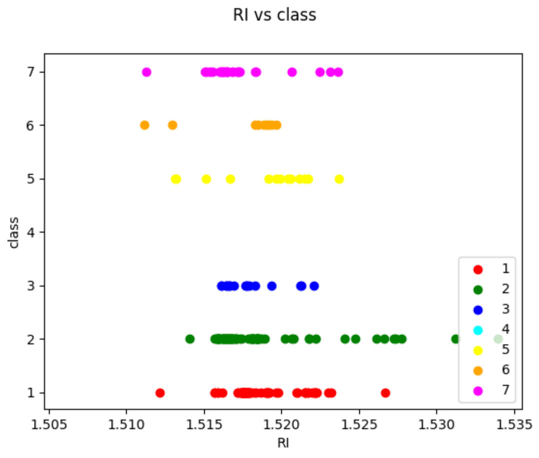
7. RI vs Ba:
   RI and Ba have no correlation with each other.



8. RI vs Fe:
   RI and Fe have no correlation with each other.



9. RI vs Class:
   RI and Class have no correlation with each other.



We continue this process for other attributes as well. From the scatter plots we find that there are correlated attributes.

1. Positive linear- correlation
The following pairs of attributes have a positive correlation with each other. Some of them have a strong correlation while some have weak correlation.

a) RI vs Ca
b) Mg vs Fe
c) Si vs K

2. Negative linear- correlation
The following pairs of attributes have a negative correlation with each other. Some of them have a strong correlation with each other while some have a weak correlation.

a) RI vs Na
b) RI vs Mg
c) RI vs Al
d) RI vs Si
e) RI vs K
f) Al vs Ca
g) K vs Ca
h) Mg vs Ca
i) Mg vs Ba
j) Mg vs Al
k) Na vs Mg
l) Na vs Ca
m) Na vs Fe
n) Na vs K
o) Na vs Si
p) Mg vs Si

3. No correlation

The following pairs of attributes have no correlation between them.

a) RI vs Ba
b) RI vs Fe
c) RI vs class
d) Al vs Ba
e) Al vs Class
f) Al vs Fe
g) Al vs K
h) Al vs Si
i) Ba vs Class
j) Ba vs Fe
k) Ca vs Ba
l) Ca vs Class
m) Ca vs Fe
n) Fe vs Class
o) K vs Ba
p) K vs Class
q) K vs Fe
r) Mg vs Class
s) Mg vs K
t) Na vs Al
u) Na vs Ba
v) Na vs Class
w) Si vs Ba
x) Si vs Ca
y) Si vs Class
z) Si vs Fe

- **<u>Normalization</u>**

From our univariate analysis by histograms, we observe that the range of values for different features is different. For example, 'RI' has values ranging between 1.51115-1.53393 while 'Si' has values ranging between 69.81-75.41.

Hence we need to normalize all values in the dataset to a specific range before we proceed with clustering. In this project, we have normalized all values to the range 0 – 1.

By doing so, we can avoid our clustering to be dominantly influenced by a certain attribute.

3.2 Preprocessing/ transformation

We do not require 'ID' attribute for clustering since ID does not have any significance / meaning with respect to the properties/ features of each of the glass instances in the given dataset. 'ID' is an attribute just used for numbering the instances.

Final Dataset for clustering **:**
RI, Na, Mg, Al, Si, K, Ca, Ba, Fe, class

3.3 Clustering

1. **K-Means clustering:**
   K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters.

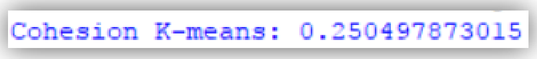
- **Parameters setting in k-Means**
- **n_clusters**: The number of clusters.
  We have set n_clusters = 7.
- **metric**: Distance between points.
  We have set it to Euclidean.
- **max_iter**: Maximum number of iteration of k-means algorithm.
  max_iter: 500
- **n_init**: Number of times the k-means algorithm run with different centroid seeds.
  n_init: 20
- **random_state**: The generator used to initialize the centers.
  Random_state: 0

K-means function:
KMeans(n_clusters=7, max_iter=500, n_init=20, random_state=0)

- **Performance Measures**

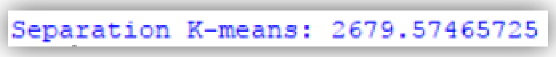
Internal Measures:

- **Cluster Cohesion**: Measures how closely related are objects in a cluster.



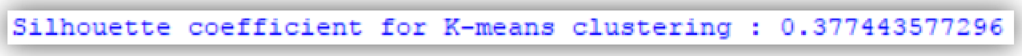
```
Cohesion K-means: 0.250497873015
```

- **Cluster Separation**: Measure how distinct or well separated a cluster is from other clusters.



```
Separation K-means: 2679.57465725
```

➢ **Silhouette Coefficient**: It combines ideas of both cohesion and separation, but for individual points.



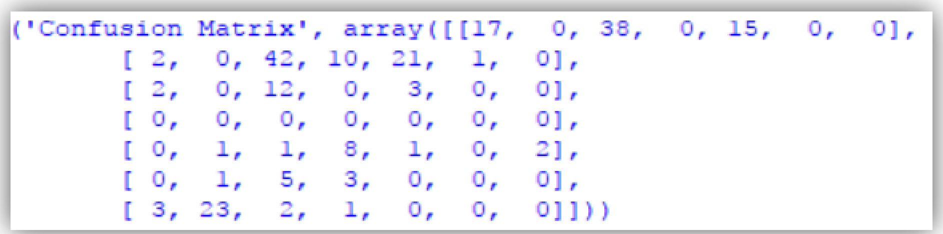
```
Silhouette coefficient for K-means clustering : 0.377443577296
```

Since the silhouette coefficient value for K-means is 0.377 which is close to 0, it indicates that the clusters overlap over each other.

External Measures:

➢ **Clustering Purity**: Purity is an external evaluation criterion of cluster quality.
To calculate 'Purity' first we compute the confusion matrix. This can be done by looping through each cluster and counting how many objects were classified as each class.

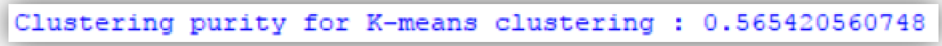
Confusion matrix for K-means:



```
('Confusion Matrix', array([[17,  0, 38,  0, 15,  0,  0],
       [ 2,  0, 42, 10, 21,  1,  0],
       [ 2,  0, 12,  0,  3,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0],
       [ 0,  1,  1,  8,  1,  0,  2],
       [ 0,  1,  5,  3,  0,  0,  0],
       [ 3, 23,  2,  1,  0,  0,  0]]))
```

Then for each cluster select the maximum value from each row, sum them together and finally divide the sum by the total number of data points.

Clustering Purity for K-means:



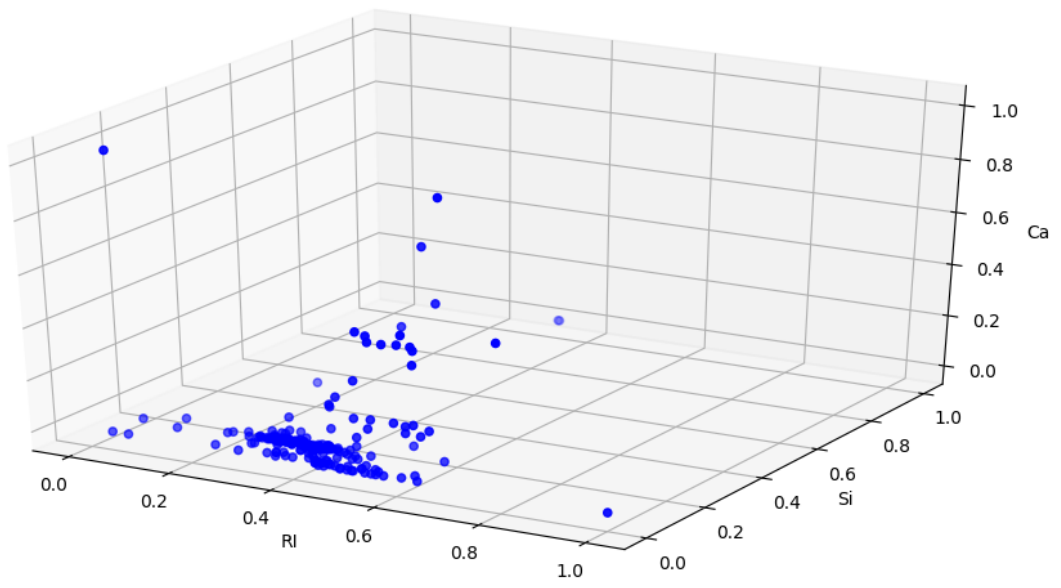
```
Clustering purity for K-means clustering : 0.565420560748
```

More than half of the data points were classified correctly.

- **Clustering Visualization**

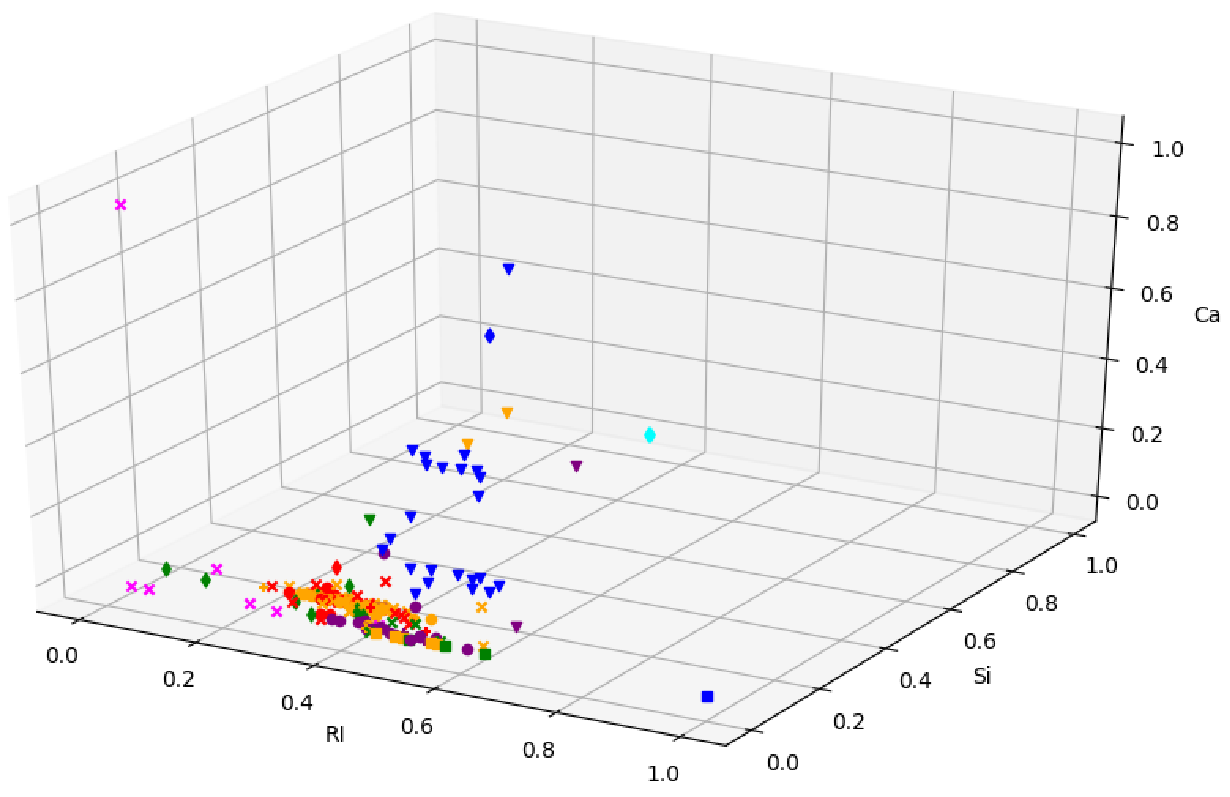
Since RI and Si have a strong negative correlation with each other and also, RI has a strong positive correlation with Ca, these 3 features are most representative in clustering and hence we have chosen RI, Si and Ca features to plot the clusters.

Scatter plot of data points before clustering



The above plot represents how the original data points of the given Glass dataset look like before clustering.

Scatter plot of data points after K-means clustering



Observations from visualization:

- 7 clusters have been formed (based on input for n_clusters) and is represented using 7 colours for each cluster.
- The clusters formed using K-means algorithm tends to get **highly influenced by outliers or noise**.
- The clusters formed are **not very well separated** and overlap over each other.
- Some of the clusters formed have mixed data points, i.e. data points belonging to other classes (obtained from the given dataset). For eg, cluster represented in blue have data points belonging to 3 different classes as visualized by ▼, ■ and ◆.

Interpretation based on visualization:
The K-means clustering algorithm works by taking initial centroids and assigning data points to the closest centroids. Hence the choice of initial centroids has a large influence on the results. They work well in assigning clusters when the data points are close to the centroids. But the centroids highly tend to get deviated by outliers or noise data due to which the centroids shift toward the outliers.

- **Performance measures vs. Visualization** :
  Low cohesion value of 0.2504 can be visualized from the plot that the data points in the clusters are not coherent.
  Silhouette coefficient of 0.377 (close to 0) can also be visualized that the clusters overlap.
  Cluster Purity of 0.565, meaning more than half of the data points were classified correctly. This can also be visualized from the plot. Some clusters have pure class labels and some have mixed labels.

2. **DBSCAN clustering:**
   Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a most widely used density based algorithm. It uses the concept of density reachability and density connectivity.

- **Parameter setting**

➢ **eps**: The maximum distance between two samples for them to be considered as in the same neighborhood.
We have set eps=0.4.

➢ **min_samples**: The number of samples in a neighborhood for a point to be considered as core point.
We have taken min_samples=4.

➢ **metric**: The metric to used when calculating distance between instances in a feature array.
We have used metric='manhattan' to calculate distance between instances**.**

DBSCAN function:

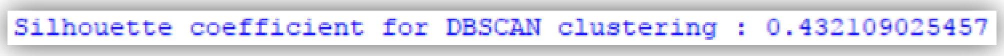
```
DBSCAN(eps=0.4, min_samples=4, metric='manhattan')
```

- **Performance Measures**

<u>Internal Measures</u>:

➢ **Silhouette Coefficient**

Silhouette Coefficient for DBSCAN:



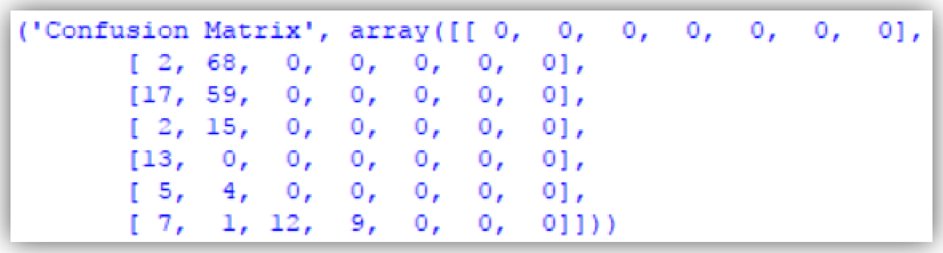
```
Silhouette coefficient for DBSCAN clustering : 0.432109025457
```

Since the silhouette coefficient value for DBSCAN which is 0.432 is not very close to 0 and is tending toward 1, it indicates that the clusters do not overlap much over each other. However, there may be a slight overlap.

<u>External Measures</u>

➢ **Clustering Purity**

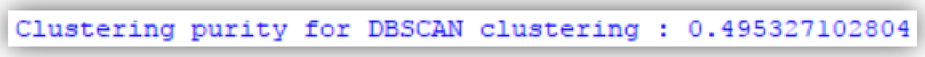
Confusion matrix for DBSCAN:



```
('Confusion Matrix', array([[ 0,  0,  0,  0,  0,  0,  0],
       [ 2, 68,  0,  0,  0,  0,  0],
       [17, 59,  0,  0,  0,  0,  0],
       [ 2, 15,  0,  0,  0,  0,  0],
       [13,  0,  0,  0,  0,  0,  0],
       [ 5,  4,  0,  0,  0,  0,  0],
       [ 7,  1, 12,  9,  0,  0,  0]]))
```

From the above confusion matrix select the maximum value from each row, sum them together and finally divide by the total number of data points.

Clustering Purity for DBSCAN:

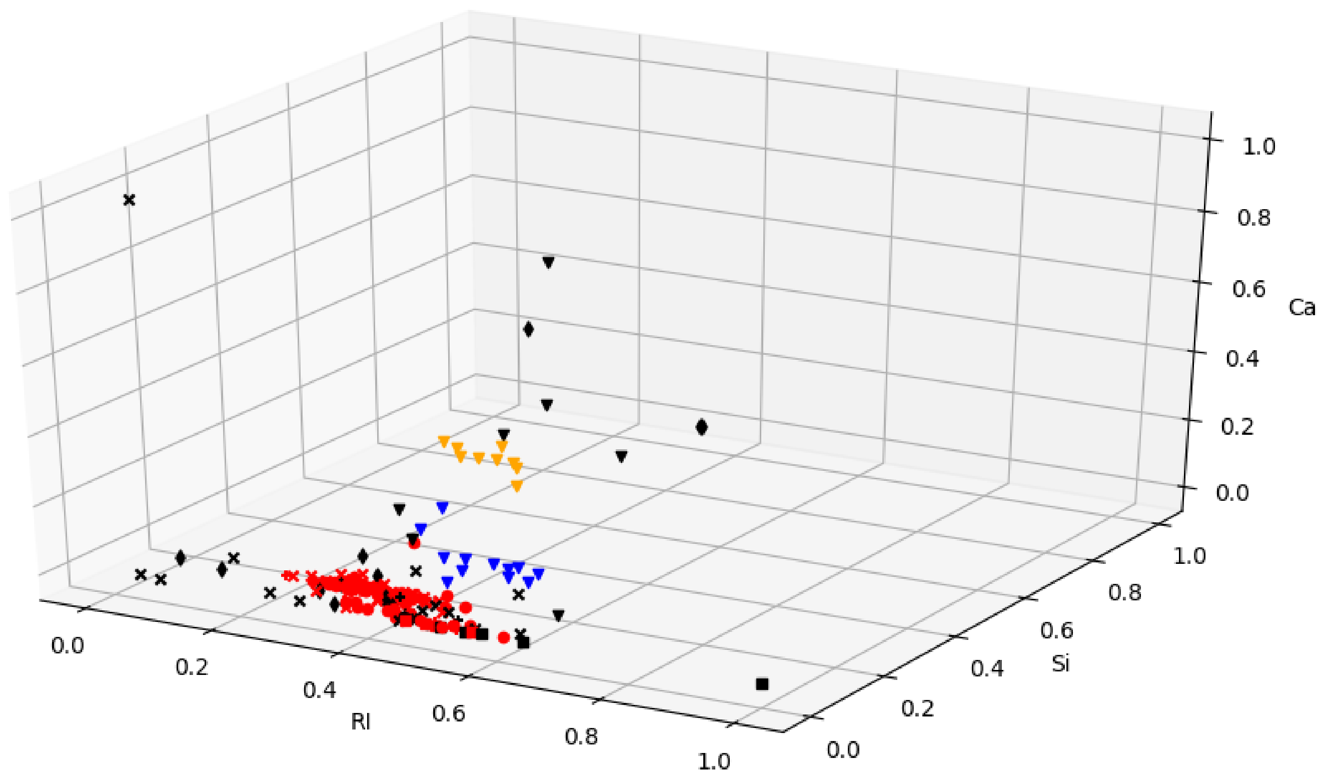


```
Clustering purity for DBSCAN clustering : 0.495327102804
```

This means that almost half of the data points were classified correctly.

- **Clustering Visualization**

Scatter plot of data points after DBSCAN clustering



Observations from visualization:

- 3 clusters have been formed and are represented using 3 colours for each cluster and noise is represented in black.
- All the outliers get classified as noise and do not form part of clusters.
- The clusters formed are **quite well separated** and do not overlap much over each other.
- Cluster represented in red has mixed data points ■, ● and ✕, i.e. data points belonging to other classes (obtained from the given dataset). However, the other 2 clusters represented in blue and yellow do not have mixed classes but both clusters have data points belonging to the same class '▼' from the given dataset.

Interpretation based on visualization:

The DBSCAN algorithm forms clusters by considering each point and finding data points which are in the vicinity of that point based on the eps value (which is a measure of radius). It also takes into account the min_samples value which is the number of data points required to form a cluster. Any number below min_samples will not be eligible to form a cluster. Since the eps value is mentioned as input, this algorithm is not prone to outliers/ noise data.

- **Performance measures vs. Visualization** :
  Silhouette coefficient of 0.432 can also be visualized that the clusters do not overlap over each other as in K-means and are not very well separated as well.
  Cluster Purity of 0.495, meaning almost half of the data points were classified correctly. This can also be visualized from the plot. Clusters represented in red have mixed labels, while other 2 clusters have pure labels and were clustered properly.

3. **Agglomerative hierarchical clustering**
   Agglomerative hierarchical clustering is a bottom-up clustering: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

- **Parameters setting**

➢ **n_clusters**: The number of clusters to find.
We have set n_clusters=6

➢ **affinity**: Metric used to compute the linkage. Can be 'euclidean' or 'manhattan'
We have used affinity='Euclidean' to compute the distance metric.

➢ **linkage:** The linkage criterion determines which distance to use between sets of observation. The algorithm will merge the pairs of cluster that minimize this criterion.
We have used linkage = 'complete' which uses the maximum distances between all observations of the two sets.

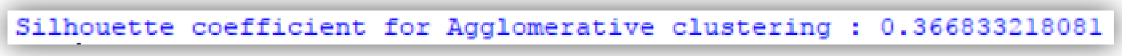
Agglomerative clustering function :

```
AgglomerativeClustering(n_clusters=6, affinity='euclidean' , linkage = "complete")
```

- **Performance Measures**

<u>**Internal Measure**</u>**:**

➢ **Silhouette Coefficient**
Silhouette Coefficient for Agglomerative hierarchical clustering**:**



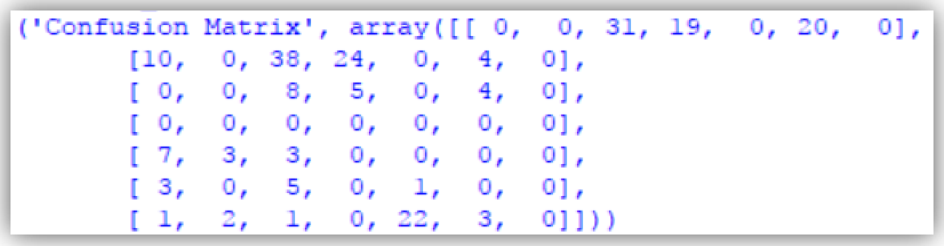
```
Silhouette coefficient for Agglomerative clustering : 0.366833218081
```

The silhouette coefficient value for Agglomerative hierarchical clustering is 0.366 which is very close to 0. It indicates that the clusters overlap over each other a lot.

<u>**External Measure:**</u>

➢ **Clustering Purity**

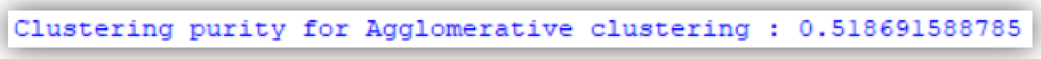
Confusion Matrix for Agglomerative hierarchical clustering**:**



```
('Confusion Matrix', array([[ 0,  0, 31, 19,  0, 20,  0],
       [10,  0, 38, 24,  0,  4,  0],
       [ 0,  0,  8,  5,  0,  4,  0],
       [ 0,  0,  0,  0,  0,  0,  0],
       [ 7,  3,  3,  0,  0,  0,  0],
       [ 3,  0,  5,  0,  1,  0,  0],
       [ 1,  2,  1,  0, 22,  3,  0]]))
```

From the above confusion matrix select the maximum value from each row, sum them together and finally divide by the total number of data points.

Cluster Purity for Agglomerative hierarchical clustering:

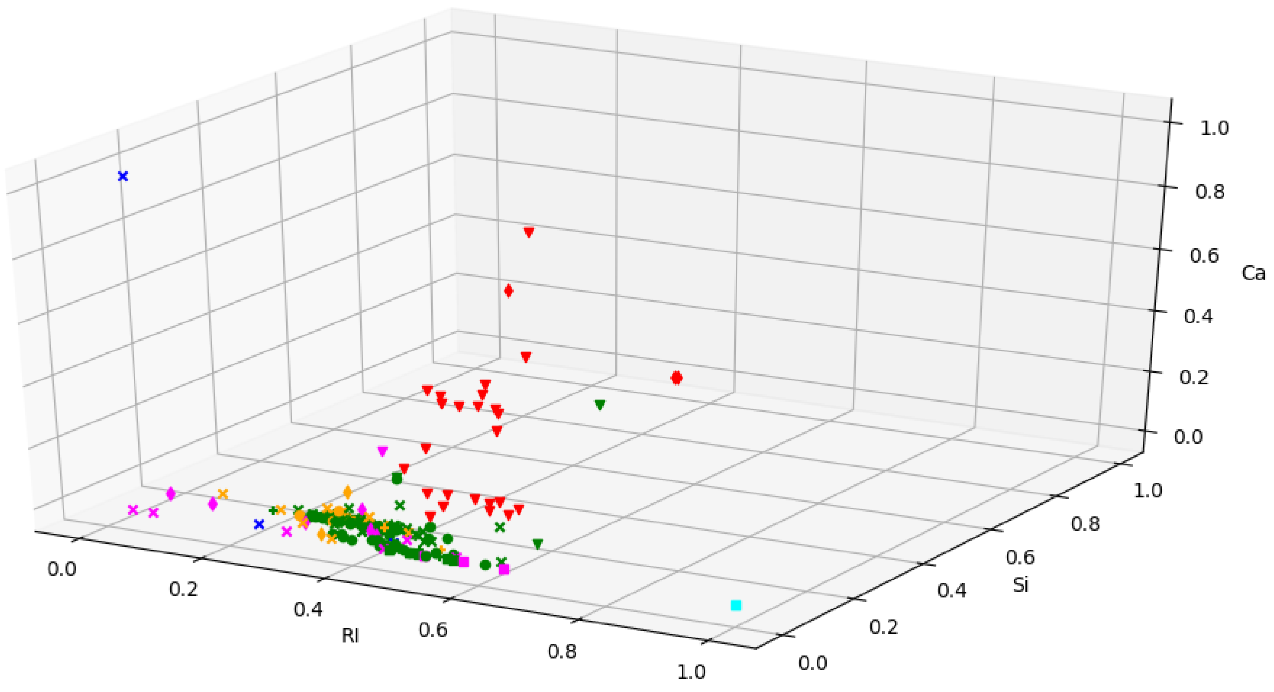


```
Clustering purity for Agglomerative clustering : 0.518691588785
```

This means that little more than half of the data points were classified correctly.

- **Cluster Visualization:**

Scatter plot of data points after Agglomerative hierarchical clustering



Observations from visualization:

- 6 clusters have been formed and are represented using 6 colours for each cluster.
- Outliers do not get merged with other clusters and form separate clusters.
- The clusters formed are **not well separated** and there is a high overlapping of clusters.
- Some of the clusters formed have mixed data points, i.e. data points belonging to other classes (obtained from the given dataset). For eg, cluster represented in green have data points belonging to 4 different classes as visualized by ▼, ■, ● and ✕.

Interpretation based on visualization:

The Agglomerative clustering algorithm first forms small clusters and then joins one cluster to other forming larger clusters. The final number of clusters is given as input. This algorithm is also less prone to outliers since the algorithm will try to merge clusters such that the 'complete' linkage between them is minimum.

- **Performance measures vs. Visualization** :

  Silhouette coefficient of 0.366 (close to 0) can also be visualized that the clusters are not well separated and overlap over each other a lot.

  Cluster Purity of 0.518, meaning little more than half of the data points were classified correctly. This can also be visualized from the plot. There are fewer clusters with mixed labels as compared to the other two algorithms.

Comparison of performance measures of the 3 clustering algorithms :

Performance measures	K-means clustering	DBSCAN clustering	Agglomerative clustering	Comparison
Silhouette coefficient (Internal measure)	0.377	0.432	0.366	The silhouette coefficient is minimum in the case of Agglomerative clustering and maximum for DBSCAN, which signifies there is less cluster overlap in DBSCAN.
Cluster purity (External measure)	0.565	0.495	0.518	Cluster purity is the highest in K-means clustering which means more data points were classified correctly as compared to the other two algorithms.