2020

Project 3

GLASS IDENFICATION DATA SET

ARNAR LEÓ ÓLAFSSON OG ÞORSTEINN SIGURÐSSON

Contents

1.Description

We chose the Glass Identification Data Set . The objective of this project is to correctly identify types of glass.

The study of classification of types of glass was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence if it is correctly identified.

We can use clustering methods to help identify the type of glass and to measure the cluster quality we need to properly validate and compare different clusterings and for that we will need both internal and external measures, we will get to this later when we have described our clusters.

We have no missing values, 214 instances and 9 attributes and one class attribute.

1. RI: refractive index

2. Na: Sodium (unit measurement: weight percent in corresponding oxide, as are attributes 4-10)

3. Mg: Magnesium

4. Al: Aluminum

5. Si: Silicon

6. K: Potassium

7. Ca: Calcium

8. Ba: Barium

9. Fe: Iron

10. Type of glass: (class attribute)

-- 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

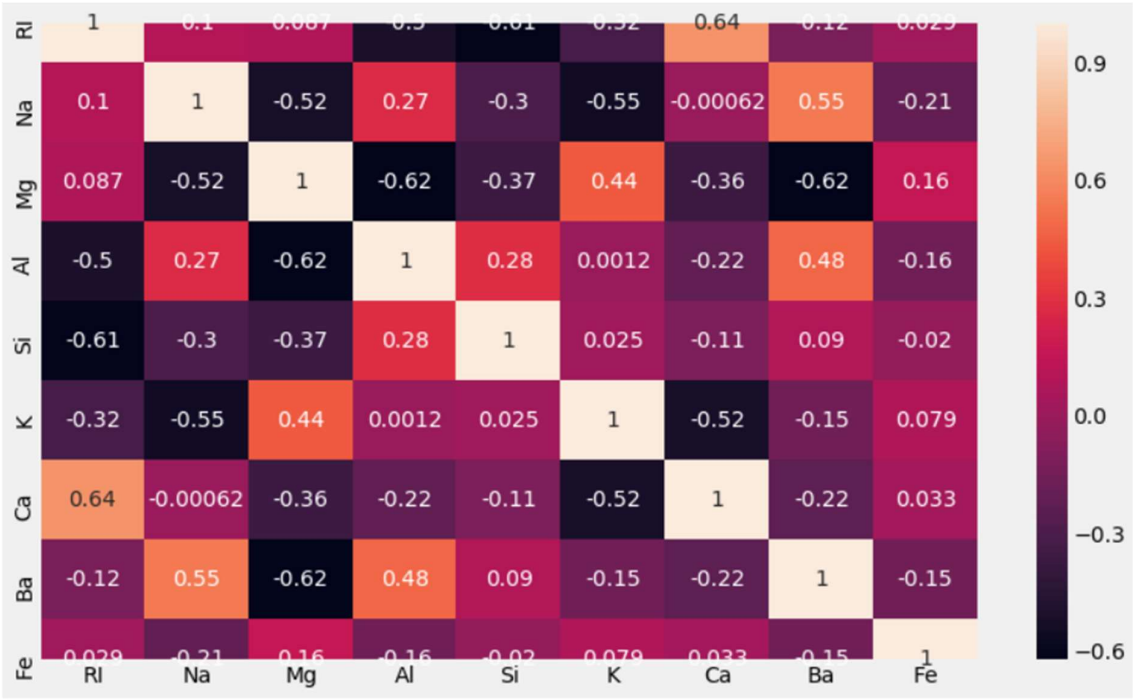
-- 7 headlamps

2.Discussion

2.1 Preprocessing

The preprocessing was done in Weka. We removed outliers by using the filters InterquartileRange and RemoveWithValues to remove outliers that were labeled as Yes. We also normalized the data using the filter Normalize. We did not feel that there was a need to use a subsample because after we removed the outliers we had less than 200 instances in our dataset.
There were no redundant attributes in our dataset so we didn't need to remove any.

Correlation heatmap for our attributes:



After preprocessing we are left with 198 instances

2.2 K-means clustering

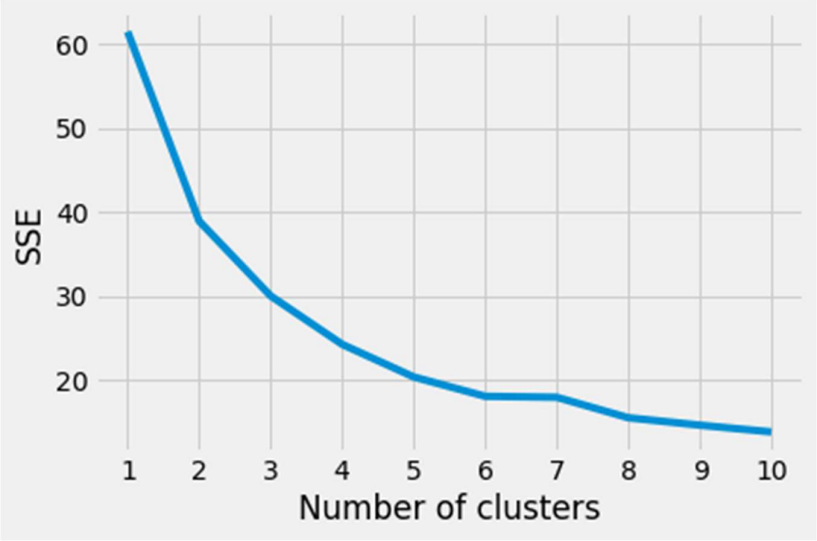
We used the KMeans class in the python package sklearn.cluster to find the best optimal cluster for our K means clustering algorithm.
We started by creating an empty array that holds all our SSE values for clusters between 1-10 and for each cluster the seeds are chosen randomly. After we found the SSE value for each cluster we plotted the correlating SSE array for numbers between 1-10 which indicate our clusters.
Kmeans.inertia is the SSE value for the resulting cluster.

```
#Elbow method to find the best k for our k means clustering algorithm
kmeans_kwargs = {
        "init": "random",
        "n_init":10,
        "max_iter":500,
        "random_state": 42,
        }

SSE = []
for k in range(1,11):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(df)
    SSE.append(kmeans.inertia_)

plt.style.use("fivethirtyeight")
plt.plot(range(1,11), SSE)
plt.xticks(range(1,11))
plt.xlabel("Number of clusters")
plt.ylabel("SSE")
plt.show()
```

Then we got the following graph.



Here we can use the elbow method to find the best optimal k for our k-mean clustering. We see from the graph that there is a tipping point in cluster 4 where the SSE value stops changing dramatically.

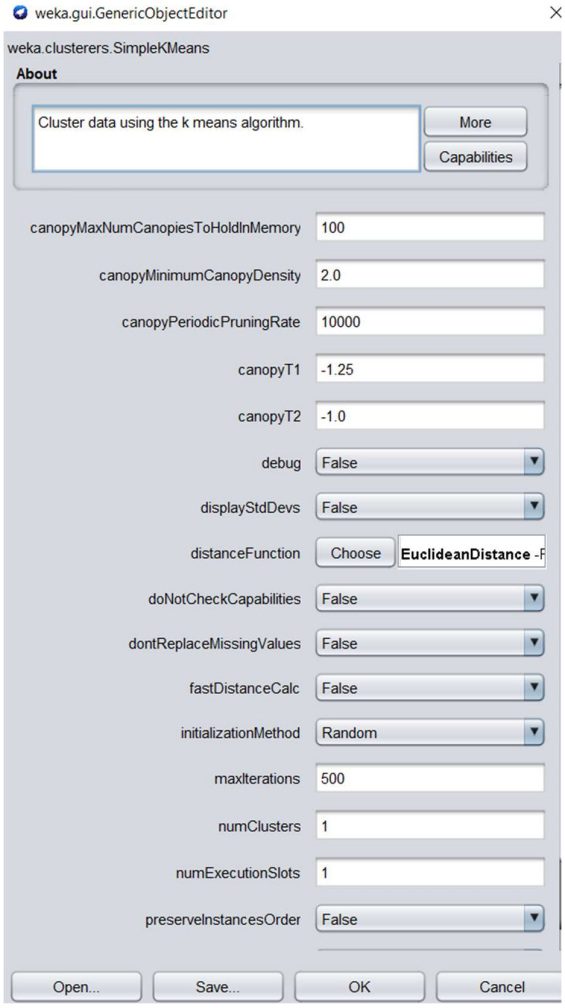
The tipping point is however not very clear so we used the Class KneeLocator in python for our SSE values and it gave the following result.

```
kl = KneeLocator(
        range(1,11), SSE, curve ="convex", direction="decreasing"
        )
kl.elbow
```

```
4
```

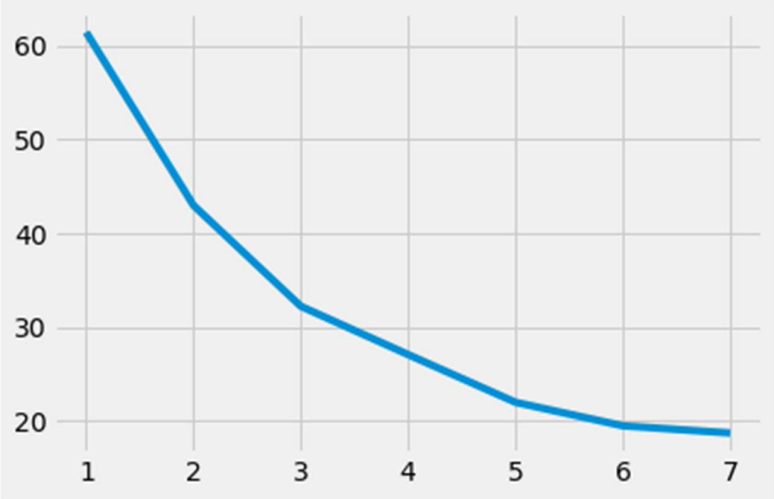
This shows us that the optimal K to use for the k-means method is four.

We also tried using Weka to see if we would get different results. We calculated the average SSE values for using clusters 1-7 and for each cluster we tried the seeds 5,10,15,20,25 and 30.
Parameters for our algorithm in Weka:



Cluster/Seeds	5	10	15	20	25	30	average
1	61,43	61,43	61,43	61,43	61,43	61,43	61,43
2	39,93	50,85	38,92	50,29	38,93	38,93	42,975
3	35,05	30,01	32,04	34,78	32,01	34,46	33,05833333
4	26,1	24,27	27,98	27,99	24,37	31,75	27,07666667
5	22,86	23,23	20,38	20,38	21,7	23,4	21,99166667
6	20,33	20,4	17,93	19,3	20,99	17,99	19,49
7	19,59	19,5	18,83	17,22	16,87	20,33	18,72333333

Then we plotted for the average SSE for each cluster.



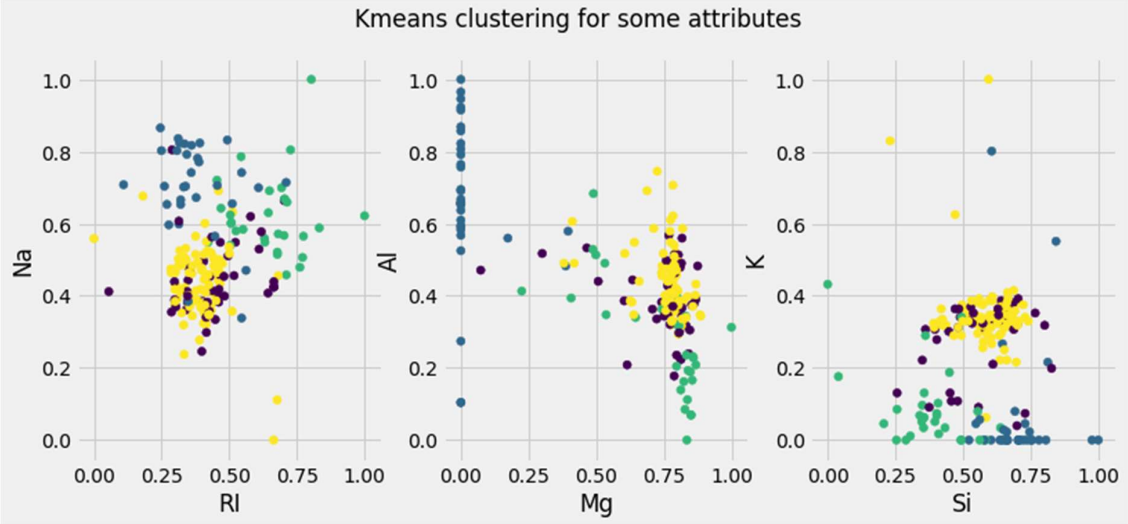
This led to the same result, that using 4 clusters would be best for the k-means method.

Now we can apply the K-means algorithm on our dataset for four clusters and we can predict the clusters for our dataset and show outcomes for some attributes in our dataset.

In [78]:
```
km4 = KMeans(n_clusters=4, n_init=10)
cluster4 = km4.fit_predict(df)
print(cluster4)
```

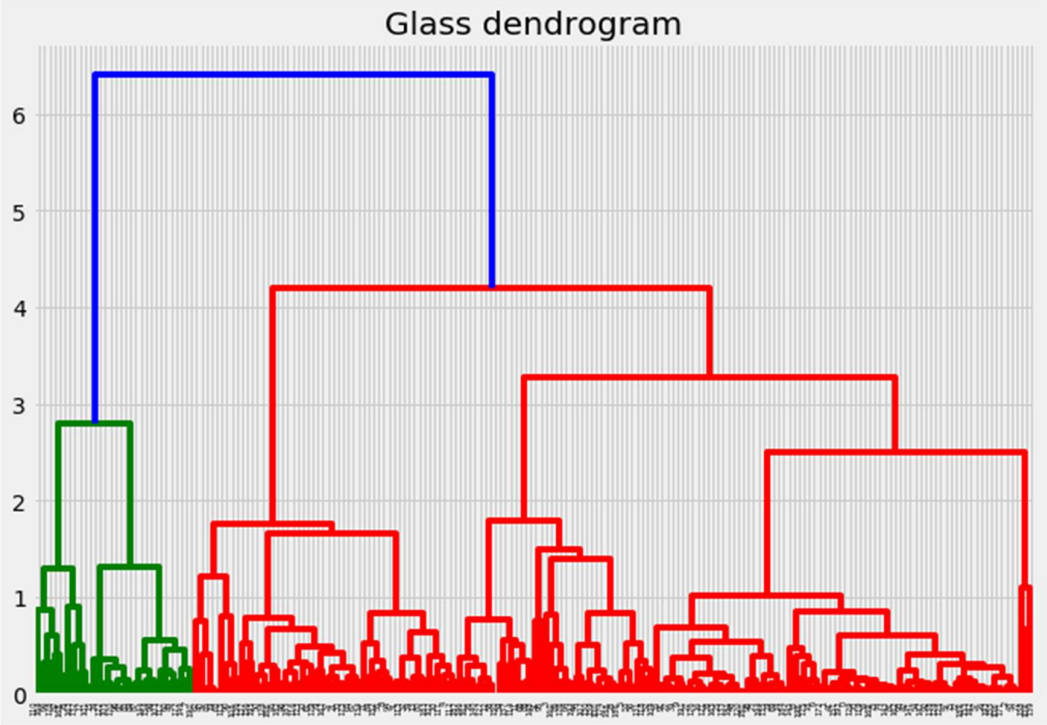
```
[2 2 2 0 1 3 2 0 1 2 1 3 3 1 2 0 2 0 2 1 1 1 2 2 0 2 2 2 2 2 2 1 1 2 1 2 2
 1 2 2 1 2 2 2 2 2 1 2 2 0 2 3 3 2 2 1 3 2 3 3 0 1 1 1 3 1 0 2 3 2 2 1 2 2
 2 2 3 2 2 1 1 2 3 0 1 1 3 0 3 0 1 2 2 2 1 3 0 0 2 0 2 0 1 0 3 0 2 3 1 2 0
 1 1 1 3 1 1 0 3 2 2 2 0 1 2 2 2 3 2 2 3 3 0 2 1 2 1 1 2 2 3 2 2 2 0 2 3 2
 3 0 3 2 1 2 0 2 1 3 0 2 0 3 2 2 0 2 2 2 0 2 0 2 2 2 2 2 3 0 0 1 2 2 2 1 2
 1 2 1 2 2 0 2 2 2 2 1 2 3]
```

Example of K-Mean clustering for four clusters:

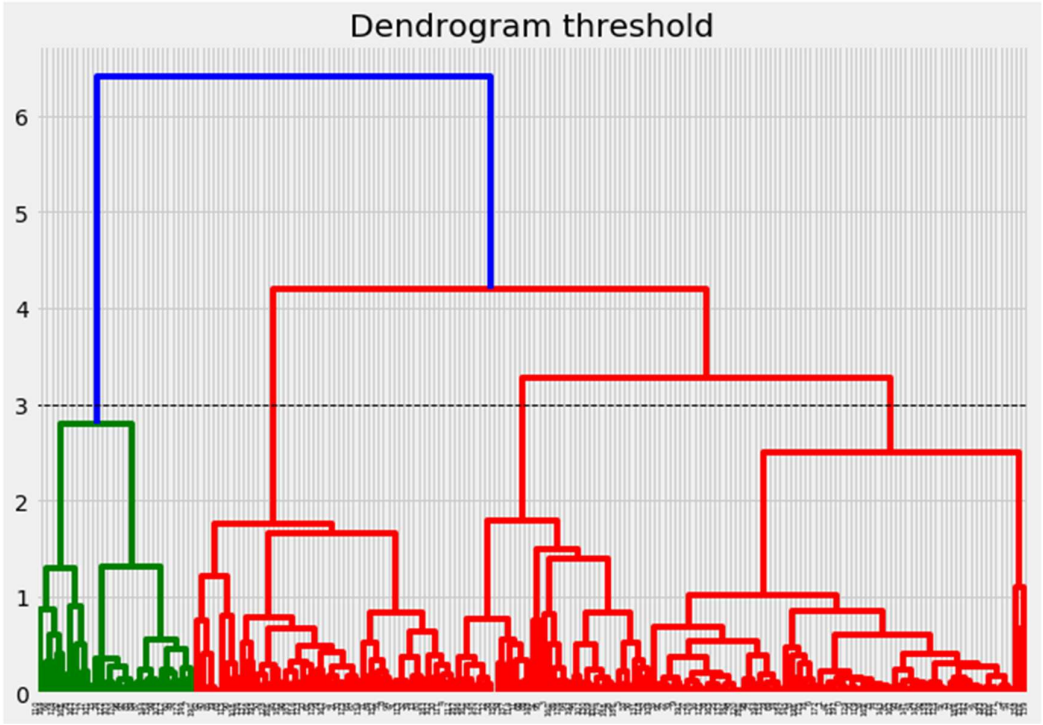


2.3 Hierarchical clustering:

For our dendrogram we used the ward method for measuring distance between two clusters. We plotted the dendrogram by using the python package scipy.cluster.hierarchy.



Then we chose the threshold distance that cuts through the largest vertical line in our dendrogram. It seems that our threshold is at the y-axis=3.



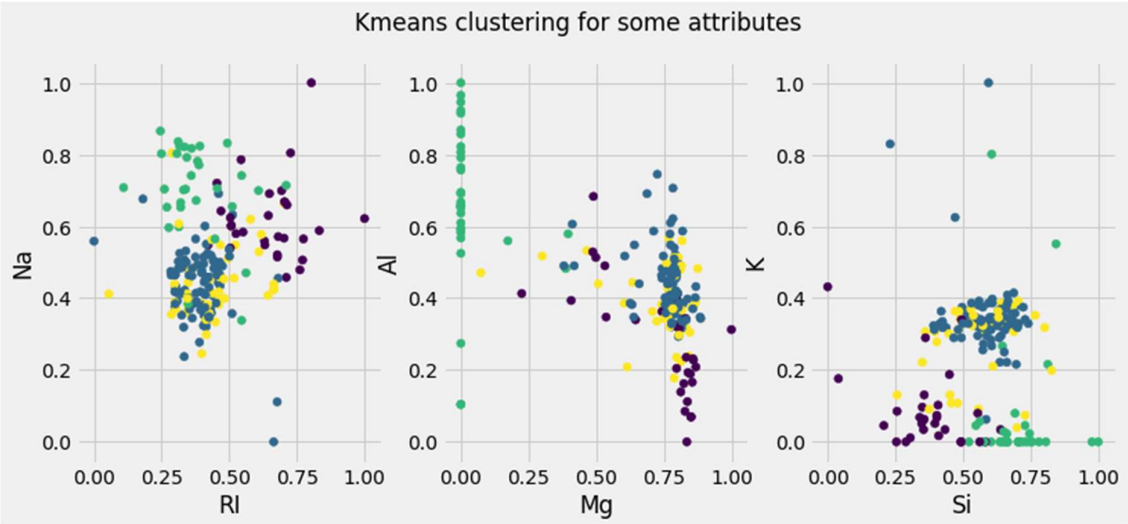
The threshold that we chose cuts through 4 lines so we choose 4 clusters for our hierarchical clustering and show some outcomes.

Now we know the number of clusters then we can apply the algorithm to predict the clusters for our data set and show some examples.

```
In [27]: Hier = AgglomerativeClustering(n_clusters=4, affinity='euclidean', linkage='ward')
         Hier.fit_predict(df)

Out[27]: array([1, 1, 1, 3, 2, 3, 1, 0, 2, 1, 2, 3, 3, 2, 1, 0, 1, 0, 1, 2, 2, 2,
                1, 1, 0, 1, 1, 1, 3, 1, 2, 2, 2, 1, 2, 1, 1, 2, 1, 1, 2, 1, 1, 1,
                3, 2, 2, 1, 1, 0, 2, 3, 3, 3, 1, 2, 3, 1, 3, 3, 0, 2, 2, 2, 3, 2,
                0, 2, 3, 1, 1, 2, 1, 1, 1, 1, 3, 1, 0, 2, 2, 1, 0, 0, 2, 2, 3, 0,
                3, 0, 2, 1, 1, 1, 2, 3, 0, 0, 1, 0, 3, 0, 2, 0, 3, 0, 1, 3, 2, 1,
                0, 2, 2, 2, 3, 2, 2, 0, 3, 1, 1, 1, 0, 2, 1, 1, 1, 3, 2, 1, 3, 3,
                0, 1, 2, 1, 2, 2, 1, 1, 3, 1, 1, 1, 0, 1, 3, 1, 3, 0, 3, 1, 2, 1,
                0, 1, 2, 3, 1, 3, 3, 3, 1, 0, 0, 1, 2, 2, 0, 1, 0, 1, 1, 2, 3, 1,
                3, 0, 0, 2, 1, 1, 2, 2, 1, 2, 3, 2, 3, 1, 0, 1, 1, 1, 1, 2, 1, 3],
               dtype=int64)
```

Example of Hierarchical clustering for four clusters:



3.Comparisons

3.1 Calculating the external and internal measures for clusters.

For internal measure we need to calculate the cohesion and seperation.

Cohesion:
Cohesion is measured with SSE(the sum of squared errors) with the formula

$$SSE = \sum_{i} \sum_{x \in C_i} (x - m_i)^2$$

Where C_i is cluster i and m_i is the centroid of that cluster.

Seperation:
Seperation is measured by the between cluster sum of squared with the following formula.

$$SSB = \sum_{i} |C_i|(m - m_i)^2$$

Where $|C_i|$ is the size of cluster i and m is the centroid of all data points.

For External measures we need to calculate the entropy and purity.

Entropy:
The degree to which each cluster consists of object of a single class with the formula

$$e_i = -\sum_{j=1}^{L} p_{ij} log_2(p_{ij})$$

Where L is the number of classes and p_{ij} is the probability that a member of cluster i belongs the a class j. Then we calculate the weighted sum of all entropies to find the total entropy of a clustering

Purity:
The extent to which a cluster contains object of a single class with the following formula

$$max p_{ij}$$

Then we calculate the wighted sum of purities to form the total purity of a clustering.

3.2 Internal and External measure for K-means clustering:

Internal measures:

	Full Data	0	1	2	3	Mean	SSB
Total	198	32	41	31	94		
RI	0.4379	0.3792	0.4446	0.6501	0.3850	0.4647	1.9134
Na	0.5087	0.7014	0.4354	0.6020	0.4443	0.5458	2.3404
Mg	0.6378	0.0299	0.7386	0.7292	0.7707	0.5671	15.1517
Al	0.4414	0.6659	0.3853	0.2886	0.4399	0.4449	2.4683
Si	0.5712	0.7024	0.5717	0.3823	0.5887	0.5613	1.7250
K	0.2477	0.0668	0.2876	0.0924	0.3431	0.1975	3.2151
Ca	0.5385	0.6074	0.5472	0.6639	0.4699	0.5721	1.3084
Ba	0.0559	0.2784	0.0065	0.0278	0.0109	0.0809	2.0232
Fe	0.1465	0.0397	0.5709	0.0410	0.0325	0.1710	9.4357
							Total SSB
							39.5811

And the total SSE was given from the SimpleK-means algorithm in Weka

Total SSE = 24.270591594432833

Total SSB = 39.5811

External Measures:

	Build wind float	Build wind non-float	vehic wind float	vehic wind non-float	containers	tableware	headlamps	total	entropy	purity
0	18	2	3	0	1	4	3	31	1,90	0,58
1	16	23	3	0	1	0	0	43	1,41	0,53
2	0	2	0	0	3	4	23	32	1,29	0,72
3	36	41	11	0	2	0	2	92	1,66	0,45
Total	70	68	17	0	7	8	28	198	1,58	0,53

Total entropy = 1.58

Total purity = 0.53

3.3 Internal and External measure for Hierarchical clustering

Internal measures:

Cluster 0		Cluster 1		Cluster 2		Cluster 3	
SSE_RI	1,060655721977	SSE_RI	0,83051965832220300	SSE_RI	0,60716632423375600	SSE_RI	0,15661744073483000
SSE_NA	1,196684654719	SSE_NA	0,60672530641783900	SSE_NA	1,30933497832274000	SSE_NA	0,10143788057120400
SSE_MG	1,545113486708	SSE_MG	1,08901822725672000	SSE_MG	0,35627798248716100	SSE_MG	0,06910928632154320
SSE_Al	0,855878644456	SSE_Al	0,59739708788147700	SSE_Al	1,53001757337784000	SSE_Al	0,14323469841056100
SSE_SI	1,257074333266	SSE_SI	1,06785509612741000	SSE_SI	0,38148582936334600	SSE_SI	0,05377694968320420
SSE_K	2,167636411518	SSE_K	0,46535128893808000	SSE_K	0,51223232790155500	SSE_K	0,02092091946900350
SSE_Ca	1,031017260247	SSE_Ca	0,69060747301216500	SSE_Ca	1,15755158146499000	SSE_Ca	0,08801623465694460
SSE_Ba	1,858648574203	SSE_Ba	0,01401019062138720	SSE_Ba	1,54821107894647000	SSE_Ba	0,00000000000000000
SSE_FE	0,026022592221	SSE_FE	2,31540345121272000	SSE_FE	0,22177629027868800	SSE_FE	0,29029542007556100
Total SSE	10,998731679316	Total SSE	7,67688777979000000	Total SSE	7,62405396637654000	Total SSE	0,92340882992285100
		Total_All SSE	27,22308225540520000				

	Full Data	0	1	2	3	Mean	SSB
Total	198	96	53	32	17		
RI	0.4379	0.3990	0.4354	0.4149	0.7089	0.4895	1.9387
Na	0.5087	0.4733	0.4518	0.6598	0.6015	0.5466	1.4529
Mg	0.6378	0.7447	0.7428	0.0374	0.8372	0.5905	14.3342
Al	0.4414	0.4328	0.4088	0.6578	0.1840	0.4209	2.7722
Si	0.5712	0.5807	0.5575	0.6877	0.3413	0.5418	1.6947
K	0.2477	0.3160	0.2950	0.0662	0.0561	0.1833	3.0640
Ca	0.5385	0.4743	0.5326	0.6651	0.6815	0.5884	1.7507
Ba	0.0559	0.0328	0.0050	0.2389	0.0000	0.0692	1.3484
Fe	0.1465	0.0017	0.4962	0.0397	0.0747	0.1531	8.9549
							Total SSB
							37.3106

Total SSE = 27.22

Total SSB = 37.3106

External measures:

	Build wind float	Build wind non-float	vehic wind float	vehic wind non-float	containers	tableware	headlamps	total	entropy	purity
0	37	37	10	0	1	5	6	96	1,94	0,39
1	19	27	5	0	1	0	1	53	1,46	0,51
2	0	3	0	0	5	3	21	32	1,46	0,66
3	14	1	2	0	0	0	0	17	0,83	0,82
Total	70	68	17	0	7	8	28	198	1,64	0,50

Total entropy = 1.64

Total purity = 0.50

4.Conclusions

From our results we can see that K-means clustering with four clusters gave us the best results. It gave us the lowest error and entropy and the higher purity value of the two clusters.