DMG Assignment 4 Report

Navneet Agarwal
2018348

Aditya Singh
2018378

Data Analysis

We analyzed the different attributes and their values in order to figure out whether an attribute contains ordinal values, nominal values, or quantitative attributes(discrete). The three list shows the distribution of attributes into three categories.

```
Ordinal_Attr=["Elevation","Aspect","Slope","Hillshade_9am","Hillshade_noon","Horizontal_Distance_To_Fire_Points"]
Nominal_Attr=['Wilderness','Soil_Type']
Quantitative_Attr=["Horizontal_Distance_To_Hydrology","Vertical_Distance_To_Hydrology"]
```

Methodology

- Since the clustering algorithms require integer DataFrames instead of strings so we had to encode every attribute of the dataset.

- For the ordinal attributes, we used a label encoder. We created a separate mapping list for each of the attributes so that the order of the attributes is maintained even after the label encoding.

- For the nominal attributes, we used One hot encoder. The one-hot encoder basically creates separate features for all the distinct values of nominal attributes.

- The quantitative attributes were left untouched.

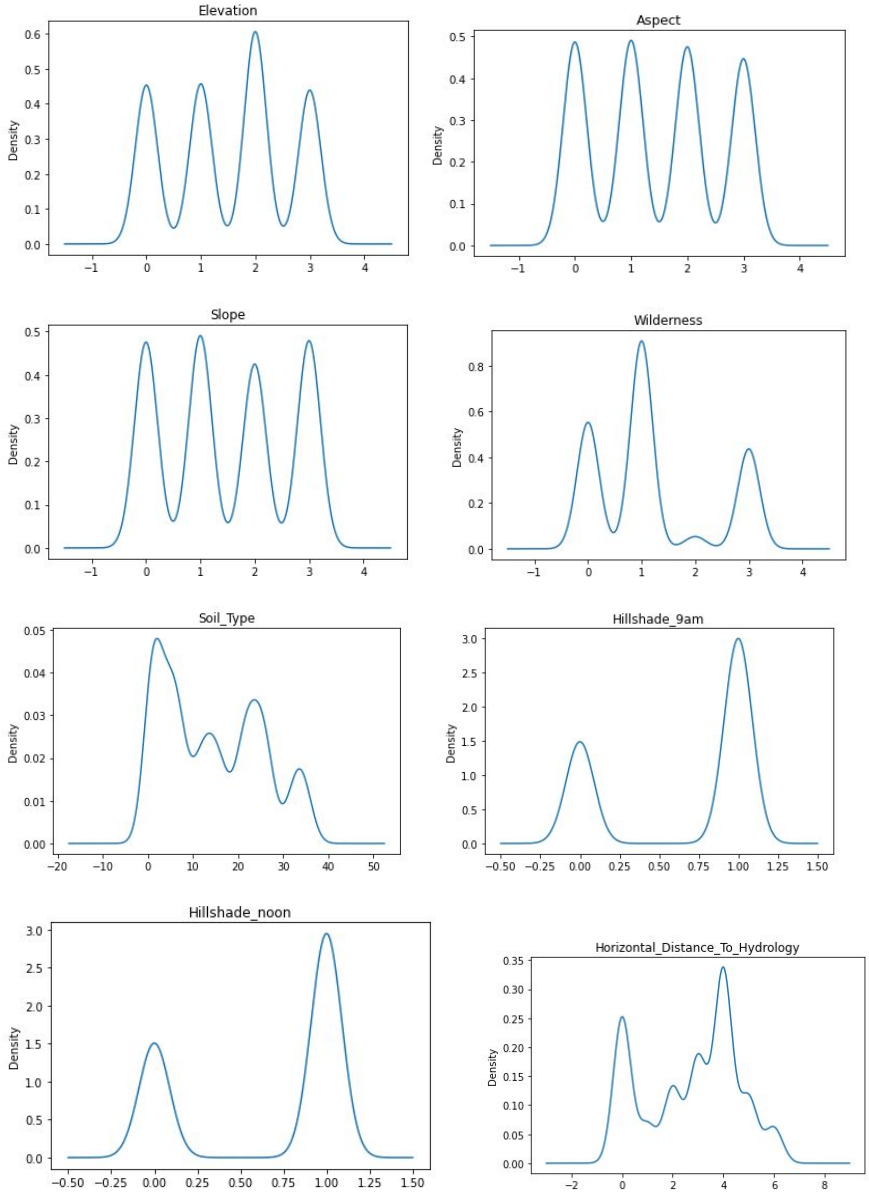
- After encoding the three types of attributes, they were combined into one single data frame to be used by the models further.

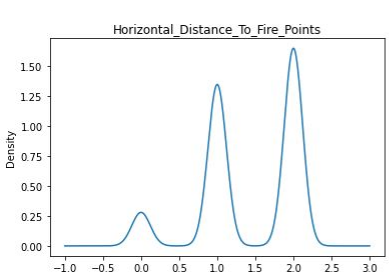
- Since the dimensions of the data frame increased from 10 to 48 after encoding, we had to reduce the dimensions in order to avoid the curse of dimensionality. We used PCA to reduce the dimensions to 20 (retained 97.5% variance).

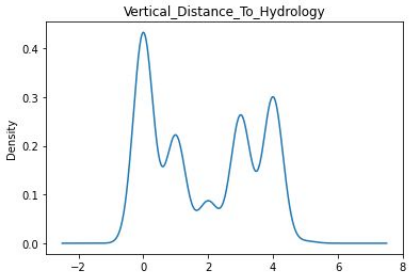
- This reduced dimension data frame was then used by different clustering methods to predict the cluster label.

Skewness Visualization

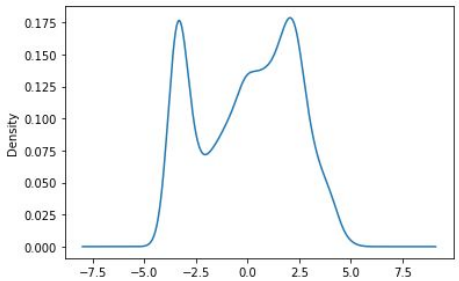
Before preprocessing

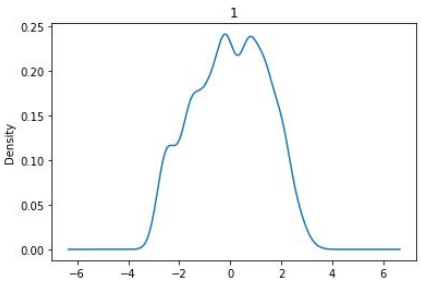


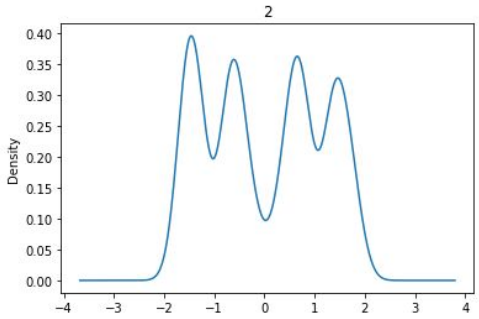


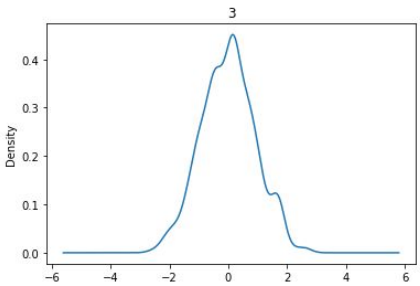


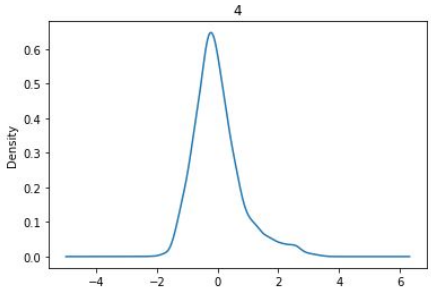
After preprocessing (Encoding is done and PCA applied)

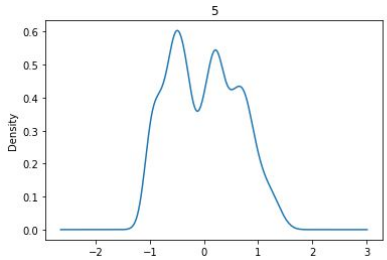


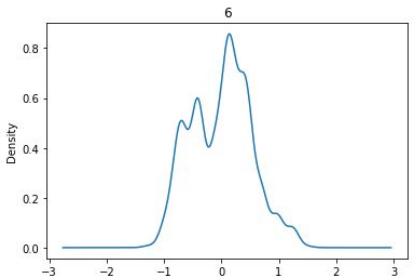


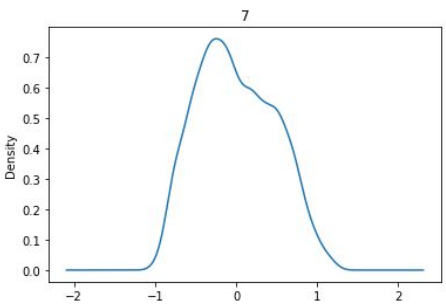


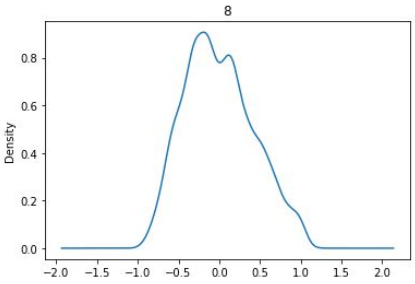


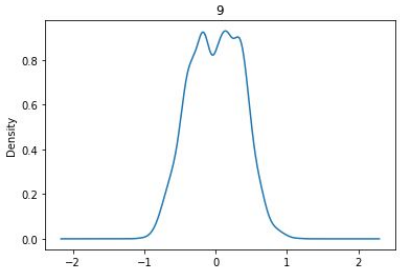


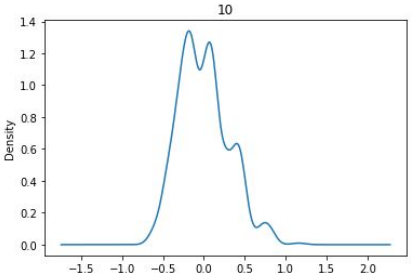


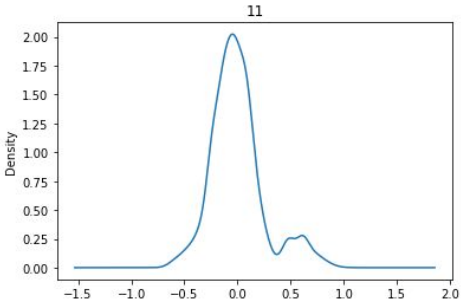


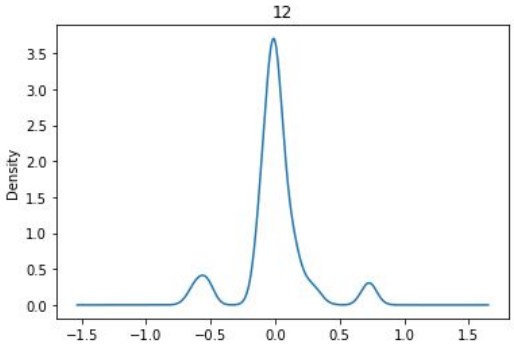


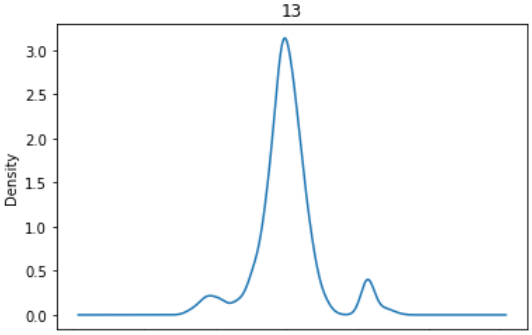

8
Density

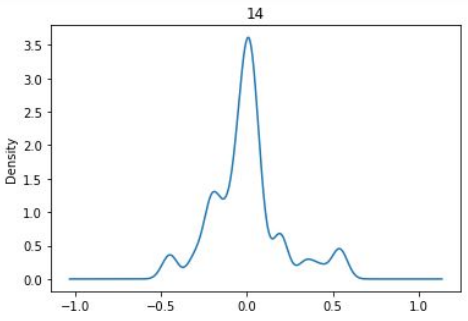

9
Density

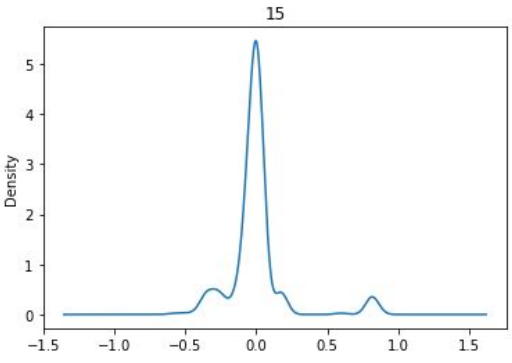

10
Density

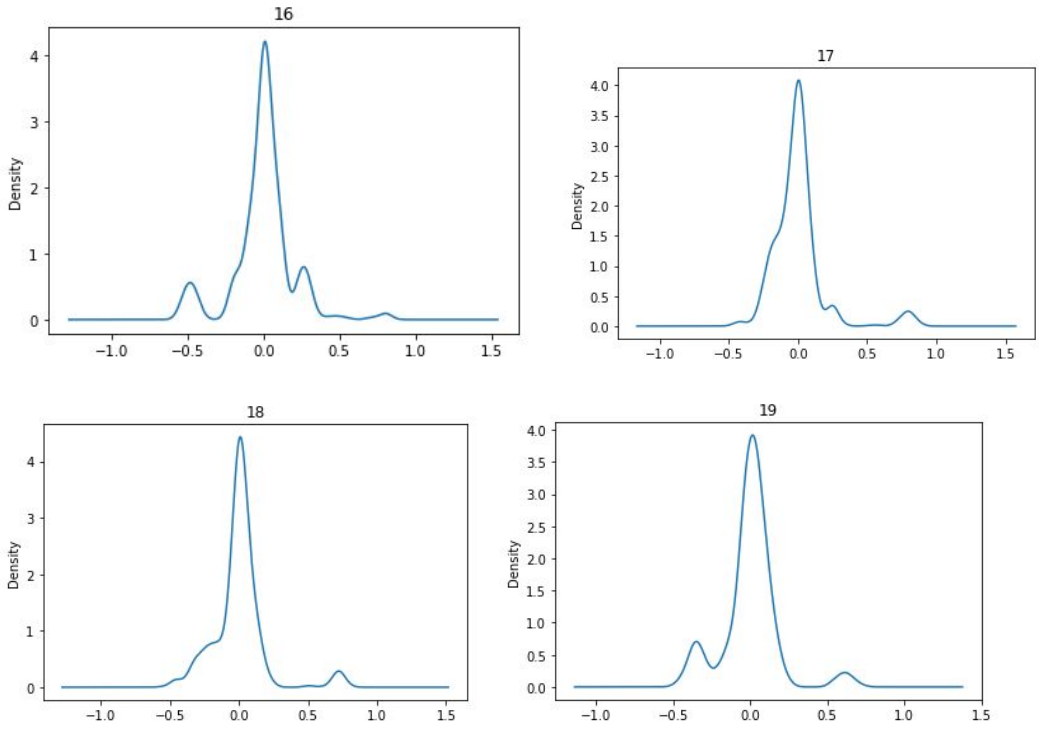

11
Density


12
Density


13
Density


14
Density


15
Density



Learning

- We learned about the implementation of different clustering algorithms like Kmeans, Kmeans++, and Agglomerative clustering.
- We learned about how different encoding techniques are used with different types of parameters.
- We also learned about the dimensionality reduction using PCA.
- We learned about how to plot these clusters.

Centroids of each cluster

[The centroids are presented after reducing the dimension using PCA, otherwise, there were 20 dimensions.]

Kmeans

```
kmeans = KMeans( init="random",n_clusters=7, max_iter=1000,
n_init=10,random_state=0 )
```

Cluster 1: [-2.89357531, 1.10484798]
Cluster 2: [-0.10978013, 1.69273064]
Cluster 3: [-3.21175589, -1.00526116]
Cluster 4: [1.5969437 , -1.92634231]
Cluster 5: [3.19927665, 1.14469332]
Cluster 6: [1.83160781, 0.02981607]
Cluster 7: [-0.67490637, -0.72867015]

```
kmeans = KMeans( init="random",n_clusters=7, max_iter=100,
n_init=100,random_state=0 )
```

Cluster 1: [3.04642143, 1.16270431]
Cluster 2: [-0.00508517, 1.66371003]
Cluster 3: [1.92872887, -0.28605471]
Cluster 4: [-2.86297546, 1.14050222]
Cluster 5: [-3.20736901, -1.00682593]
Cluster 6: [1.49980353, -2.05968167]
Cluster 7: [-0.67675757, -0.62966489]

```
kmeans = KMeans( init="random",n_clusters=5, max_iter=1000,
n_init=10,random_state=0 )
```

Cluster 1: [-2.86437104, -0.99708347]
Cluster 2: [-2.79839737, 1.12811798]
Cluster 3: [1.37225997, -1.5893385]
Cluster 4: [2.71826057, 0.74815938]
Cluster 5: [-0.02753129, 1.2677926]

```
kmeans = KMeans( init="random",n_clusters=9, max_iter=1000,
n_init=10,random_state=0 )
y_kmeans = kmeans.fit_predict(X_pca)
```

Cluster 1: [-3.02723908, 1.08544859]
Cluster 2: [2.06172543, -2.0756097]
Cluster 3: [-0.35908765, 1.89129186]
Cluster 4: [1.6168669 , 1.25645359]
Cluster 5: [3.5231272 , 0.99989309]

Cluster 6: [0.26984001, -1.82073371]
Cluster 7: [-3.15493884, -1.02295365]
Cluster 8: [-0.7147362 , 0.03375999]
Cluster 9: [2.04509799, -0.45552386]

Kmeans++

```
kmeans = KMeans( init="k-means++",n_clusters=5, max_iter=1000,
n_init=10,random_state=0 )
```

Cluster 1: [1.37007986, -1.59722156]
Cluster 2: [2.70847803, 0.74084837]
Cluster 3: [-0.03930668, 1.26496876]
Cluster 4: [-2.86749393, -0.9974643]
Cluster 5: [-2.81015737, 1.13058746]

```
kmeans = KMeans( init="k-means++",n_clusters=7, max_iter=1000,
n_init=10,random_state=0 )
```

Cluster 1: [1.52606708e+00, -2.06659616e+00]
Cluster 2: [-1.62130911e-04, 1.65387530e+00]
Cluster 3: [3.04900017e+00, 1.16280031e+00]
Cluster 4: [-3.21175589e+00, -1.00526116e+00]
Cluster 5: [1.92677248e+00, -2.85027873e-01]
Cluster 6: [-2.84618971e+00, 1.13947972e+00]
Cluster 7: [-6.59656297e-01, -6.80775784e-01]

```
kmeans = KMeans( init="k-means++",n_clusters=9, max_iter=1000,
n_init=10,random_state=0 )
```

Cluster 1: [-0.93010765, -0.06532259]
Cluster 2: [3.33143074, 1.16111684]
Cluster 3: [-3.06291654, 1.09252898]
Cluster 4: [2.04086232, -2.07879415]
Cluster 5: [-0.43756143, 1.89478737]
Cluster 6: [-3.19248374, -1.02780201]
Cluster 7: [1.24332031, 1.10062937]
Cluster 8: [2.19927331, -0.4375761]
Cluster 9: [0.33321322, -1.74896048]

```
model = AgglomerativeClustering(n_clusters=5, affinity='euclidean',
linkage=average)
```

Cluster 1: [-0.22760379, 1.54951384]

Cluster 2: [2.94112467, 1.22316416]
Cluster 3: [2.03685714, -0.85042994]
Cluster 4: [-3.24996932, 0.42177069]
Cluster 5: [-0.81573999, -0.38311047]
Cluster 6: [-2.91769541, -1.70816593]
Cluster 7: [0.56113948, -2.14868233]

AgglomerativeClustering

```
model = AgglomerativeClustering(n_clusters=7, affinity='euclidean',
linkage='ward')
```

Cluster 1: [1.85787065, -1.31191209]
Cluster 2: [2.70406404, 0.94289625]
Cluster 3: [-2.98729913, 0.85174042]
Cluster 4: [-1.03652234, -0.21716781]
Cluster 5: [0.03238413, 1.72727573]
Cluster 6: [-3.20620226, -1.39919985]
Cluster 7: [0.08403087, -1.76890915]

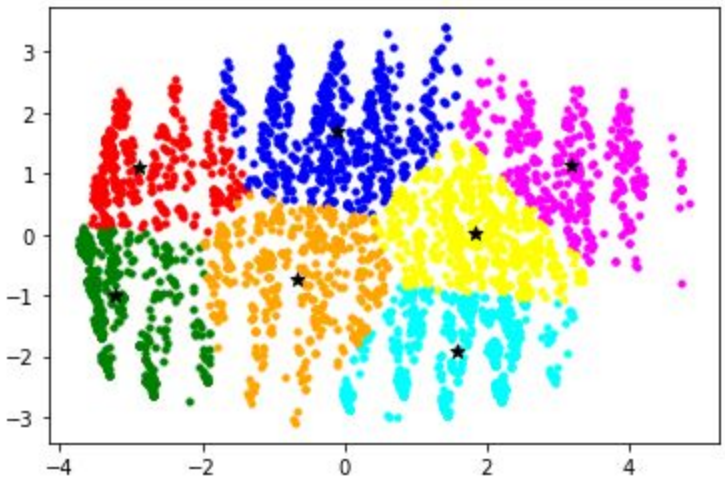
```
model = AgglomerativeClustering(n_clusters=9, affinity='euclidean',
linkage='complete')
```

Cluster 1: [-2.49356217, -0.87325027]
Cluster 2: [3.04013029, 0.60818765]
Cluster 3: [1.60112399, -1.79833478]
Cluster 4: [-0.44595446, 1.34835976]
Cluster 5: [-3.13135629, 1.24280917]
Cluster 6: [1.21374005, 2.15483143]
Cluster 7: [1.16180897, 0.01731472]

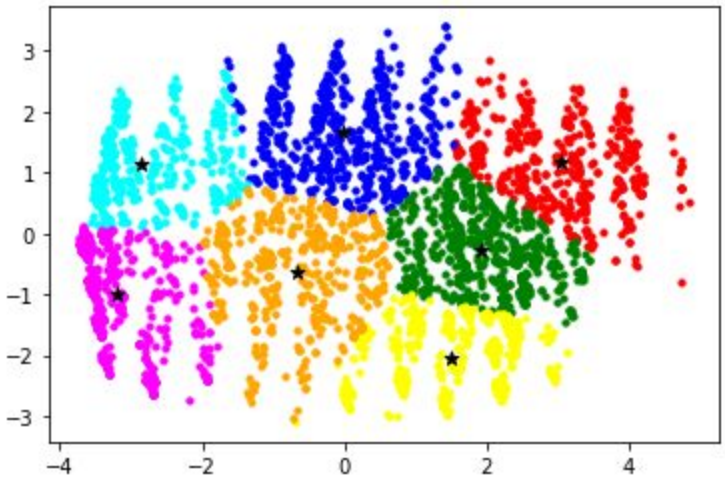
Cluster Visualization

Kmeans

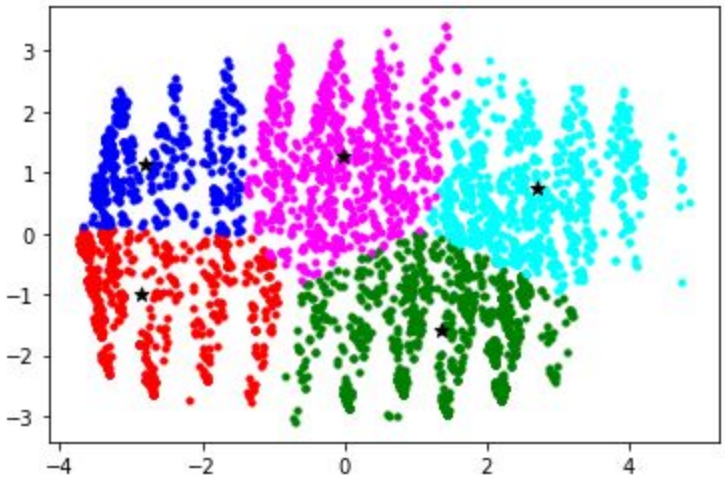
```
kmeans = KMeans( init="random",n_cluster=7, max_iter=1000,
n_init=10,random_state=0 )
```



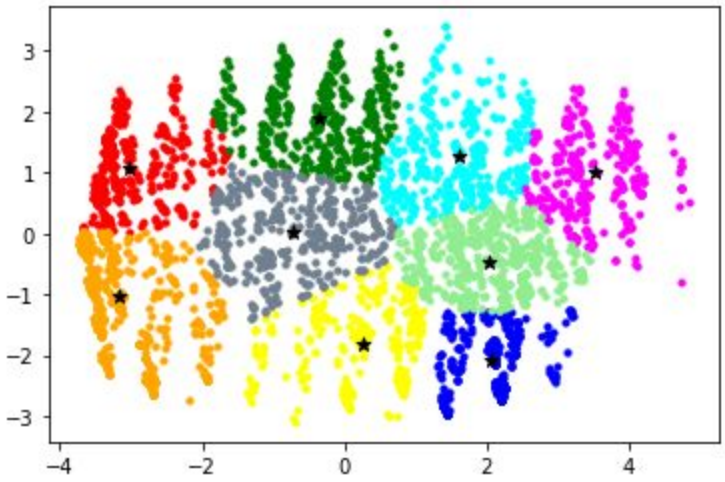

```
    kmeans = KMeans( init="random",n_clusters=7, max_iter=100,
n_init=100,random_state=0 )
```




```
kmeans = KMeans( init="random",n_clusters=5, max_iter=1000,
n_init=10,random_state=0 )
```

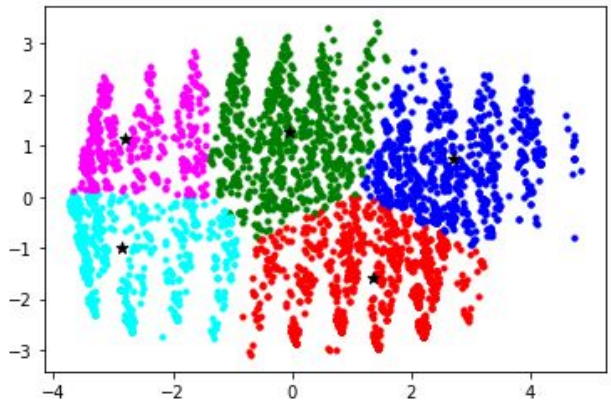



```
kmeans = KMeans( init="random",n_clusters=9, max_iter=1000,
n_init=10,random_state=0 )
y_kmeans = kmeans.fit_predict(X_pca)
```

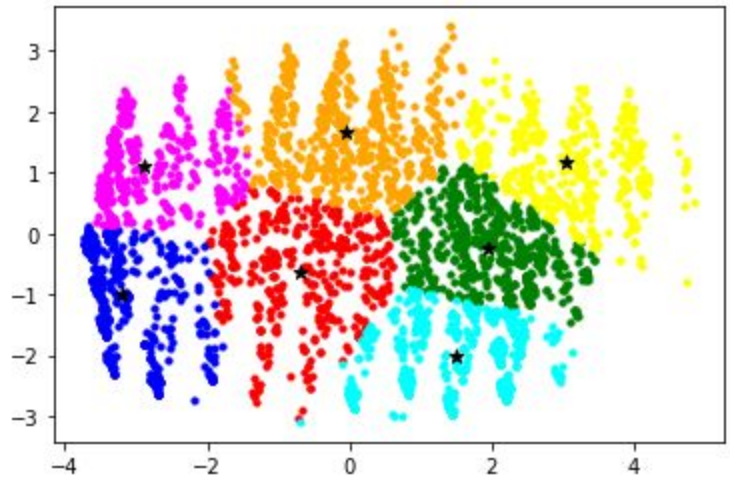


Kmeans++

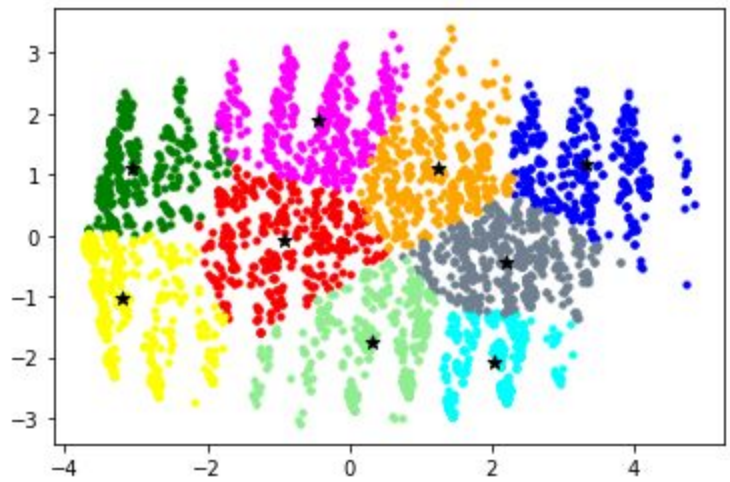
```
kmeans = KMeans( init="k-means++",n_clusters=5, max_iter=1000,
n_init=10,random_state=0 )
```




```
kmeans = KMeans( init="k-means++",n_clusters=7, max_iter=1000,
n_init=10,random_state=0 )
```

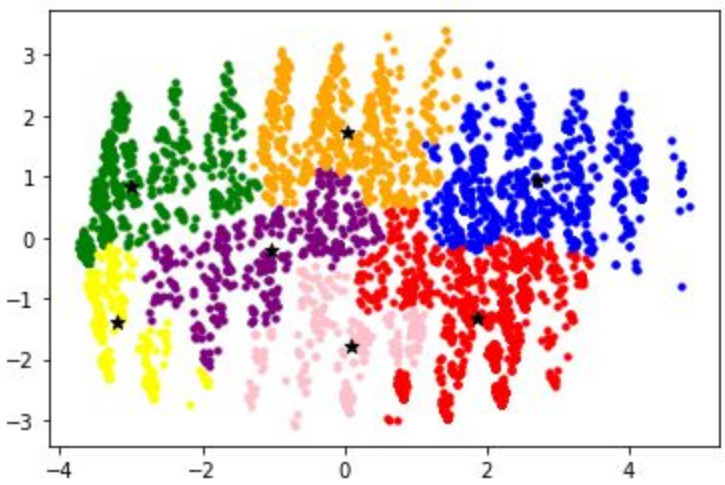



```
kmeans = KMeans( init="k-means++",n_clusters=9, max_iter=1000,
n_init=10,random_state=0 )
```

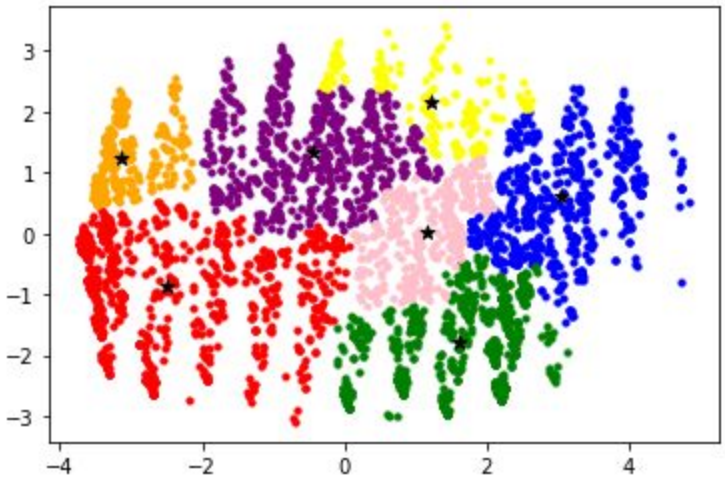


Agglomerative

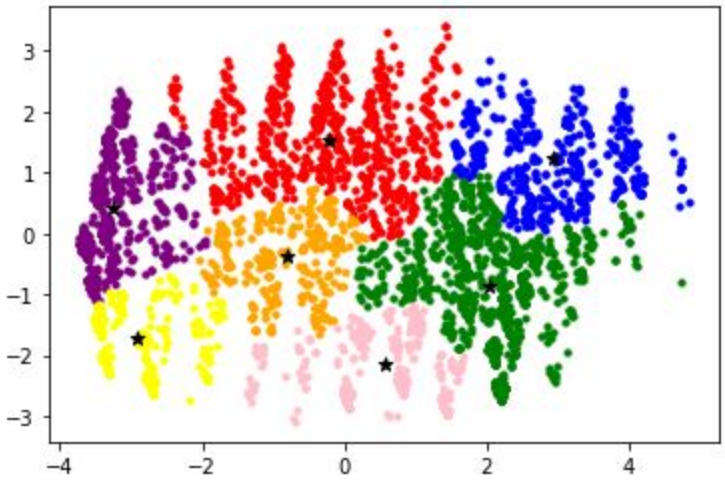
```
model = AgglomerativeClustering(n_clusters=7, affinity='euclidean',
linkage='ward')
```




```
model = AgglomerativeClustering(n_clusters=7, affinity='euclidean',
linkage='complete')
y_agg=model.fit_predict(X_pca)
labels = model.labels_
```




```
# clustering = AgglomerativeClustering()
model = AgglomerativeClustering(n_clusters=7, affinity='euclidean',
linkage='average')
```



Number of instances in each cluster and their comparison

Cluster	True count
1	540
4	540
5	540
7	540
2	542
6	675
3	743

Kmeans

```
kmeans = KMeans( init="random",n_clusters=7, max_iter=1000,
n_init=10,random_state=0 )
```

Counts for clusters = [507, 509, 530, 595, 598, 627, 754]
The number of instances in each cluster are almost identical to the original dataset.This means that the given algorithm performs really well in identifying the clusters.

```
kmeans = KMeans( init="random",n_clusters=7, max_iter=100,
n_init=100,random_state=0 )
```

Counts for clusters = [482, 516, 551, 554, 568, 635, 814]
When we increased the number of times k-means is run through for different seeds, we observe that for some of the clusters the values are still comparable while for the min and max value it is highly deviated.

```
kmeans = KMeans( init="random",n_clusters=5, max_iter=1000,
n_init=10,random_state=0 )
```

Counts for clusters = [616, 727, 892, 919, 966]
The comparison does not make sense here, as the number of clusters are less.

```
kmeans = KMeans( init="random",n_clusters=9, max_iter=1000,
n_init=10,random_state=0 )
```

Counts for clusters = [405, 426, 429, 456, 457, 460, 470, 507, 510]
Comparison does not make sense here, as the number of clusters are more. But we see the values are equally distributed in the clusters.

Kmeans++

```
kmeans = KMeans( init="k-means++",n_clusters=5, max_iter=1000,
n_init=10,random_state=0 )
```

Counts for clusters = [612, 669, 901, 928, 1010]
The comparison does not make sense here, as the number of clusters are less.

```
kmeans = KMeans( init="k-means++",n_clusters=7, max_iter=1000,
n_init=10,random_state=0 )
```

Counts for clusters = [482, 516, 539, 571, 572, 637, 803]
The values here are in correspondence with the original cluster distribution.

```
kmeans = KMeans( init="k-means++",n_clusters=9, max_iter=1000,
n_init=10,random_state=0 )
```

Counts for clusters = [354, 407, 436, 444, 468, 478, 484, 513, 536]
The comparison does not make sense here, as the number of clusters are more.

Agglomerative

```
model = AgglomerativeClustering(n_clusters=5, affinity='euclidean',
linkage=average)
```

Counts for clusters = [300, 332, 352, 543, 773, 814, 1006]
We can clearly see that these numbers do not align with the original cluster numbers thus average linkage should not be used.

```
model = AgglomerativeClustering(n_clusters=7, affinity='euclidean',
linkage='ward')
```

Counts for clusters = [488, 507, 511, 591, 648, 655, 720]
Some of the values are in correspondence to the original cluster distribution, thus we can say that ward linkage performs better than complete linkage for the given dataset.

```
model = AgglomerativeClustering(n_clusters=9, affinity='euclidean',
linkage='complete')
```

Counts for clusters = [354, 437, 456, 489, 494, 796, 1094]
These results are very poor as compared to the original results, this suggests complete linkage is not present between the instances.

Contribution:

- Equal Contribution was done by both the team members.
- Different algorithms were run by different members
- Variations of one algorithm were tried by one person
- Preprocessing and data analysis was done together.