DMG ASSIGNMENT 3
Harshal Dev, 2019306
Sneh Suman, 2019337

Procedure, steps, and outcomes

Assumption-

1) K-mean clustering is used out of all after testing for the large dataset provided to us.
2) The number of clusters = The number of targets we have
3) Since we get values from 0 to k-1 when we set the cluster as k. We will increment all our clusters by 1 so that both clusters and targets become from 0 to 1.

Procedure-

1) We will first preprocess our dataset.

```
df.head(10)
```

	Elevation	Aspect	Slope	Hillshade_9am	Hillshade_Noon	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Horizontal_Distance_To_Fire_Points
0	elevation_medium	aspect_medium	slope_low	hillshade_9am_max	hillnoon_max	0	1	low
1	elevation_high	aspect_medium	slope_low	hillshade_9am_max	hillnoon_max	1	1	mid
2	elevation_medium	aspect_low	slope_low	hillshade_9am_max	hillnoon_max	1	1	low
3	elevation_high	aspect_ultra	slope_medium	hillshade_9am_max	hillnoon_max	2	1	low
4	elevation_high	aspect_high	slope_low	hillshade_9am_max	hillnoon_max	2	1	mid
5	elevation_high	aspect_low	slope_low	hillshade_9am_max	hillnoon_max	1	1	low
6	elevation_medium	aspect_ultra	slope_high	hillshade_9am_min	hillnoon_max	1	2	low
7	elevation_high	aspect_low	slope_low	hillshade_9am_max	hillnoon_max	0	1	mid
8	elevation_medium	aspect_low	slope_low	hillshade_9am_max	hillnoon_max	0	1	high
9	elevation_medium	aspect_low	slope_low	hillshade_9am_max	hillnoon_max	0	1	mid

```
[6] df.shape
```

```
(406708, 11)
```

```
[29] df['Hillshade_Noon'] = df['Hillshade_Noon'].replace("hillnoon_min", 0)
     df['Hillshade_Noon'] = df['Hillshade_Noon'].replace("hillnoon_max", 1)
```

```
[30] df.Hillshade_Noon.unique()
```

```
array([1, 0])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 406708 entries, 0 to 406707
Data columns (total 11 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   Elevation                           406708 non-null  int64
 1   Aspect                              406708 non-null  int64
 2   Slope                               406708 non-null  int64
 3   Hillshade_9am                       406708 non-null  int64
 4   Hillshade_Noon                      406708 non-null  int64
 5   Horizontal_Distance_To_Hydrology    406708 non-null  int64
 6   Vertical_Distance_To_Hydrology      406708 non-null  int64
 7   Horizontal_Distance_To_Fire_Points  406708 non-null  object
 8   Soil_Type                           406708 non-null  int64
 9   Wilderness                          406708 non-null  int64
 10  target                              406708 non-null  int64
dtypes: int64(10), object(1)
memory usage: 34.1+ MB
```

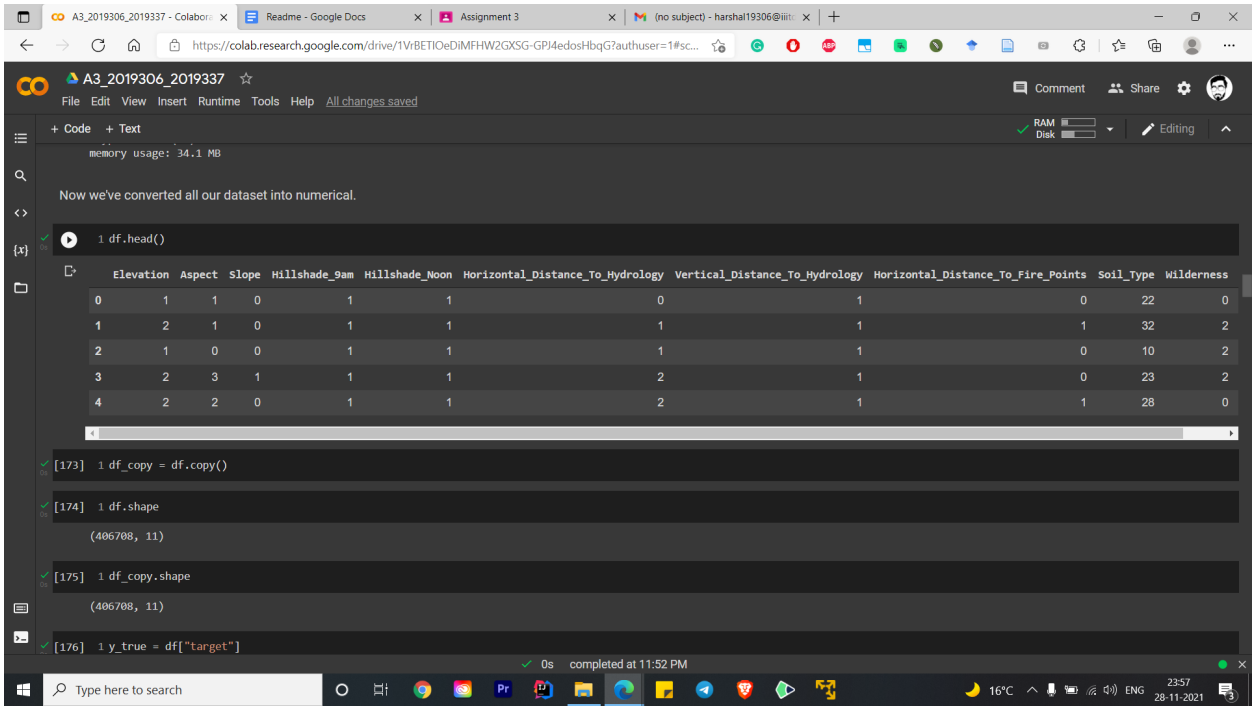
2)

```
df.describe()
```

	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Soil_Type	Wilderness	target
count	406708.000000	406708.000000	406708.000000	406708.000000	406708.000000
mean	0.901701	1.202819	23.358461	1.114679	2.051472
std	1.048836	0.478138	9.483622	1.061301	1.396507
min	0.000000	0.000000	0.000000	0.000000	1.000000
25%	0.000000	1.000000	19.000000	0.000000	1.000000
50%	1.000000	1.000000	28.000000	1.000000	2.000000
75%	1.000000	1.000000	30.000000	2.000000	2.000000
max	6.000000	5.000000	39.000000	3.000000	7.000000

```
[8] df.corr()
```

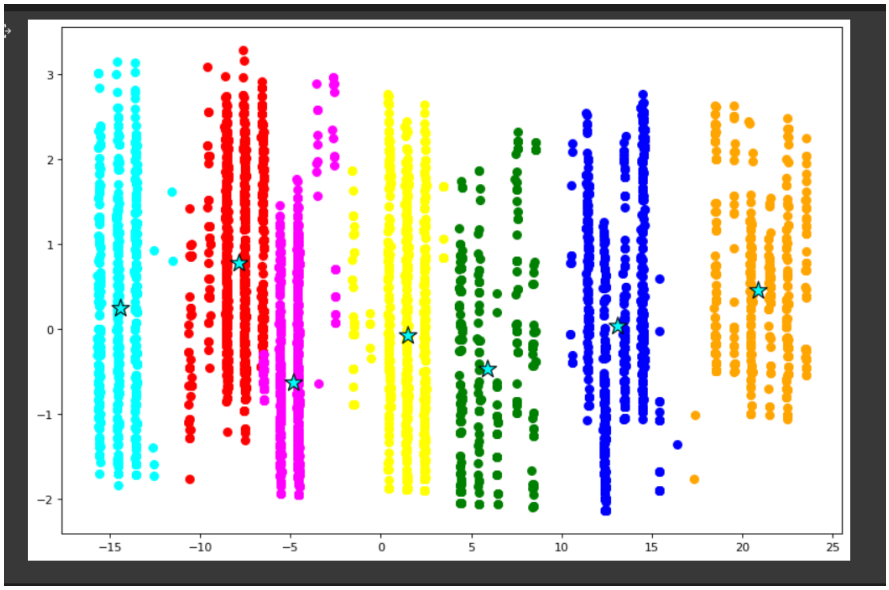
	Horizontal_Distance_To_Hydrology	Vertical_Distance_To_Hydrology	Soil_Type	Wilderness	target
Horizontal_Distance_To_Hydrology	1.000000	0.506584	0.187463	0.056313	-0.017247
Vertical_Distance_To_Hydrology	0.506584	1.000000	0.060589	0.165132	0.073802
Soil_Type	0.187463	0.060589	1.000000	-0.271669	-0.163359
Wilderness	0.056313	0.165132	-0.271669	1.000000	0.275715
target	-0.017247	0.073802	-0.163359	0.275715	1.000000



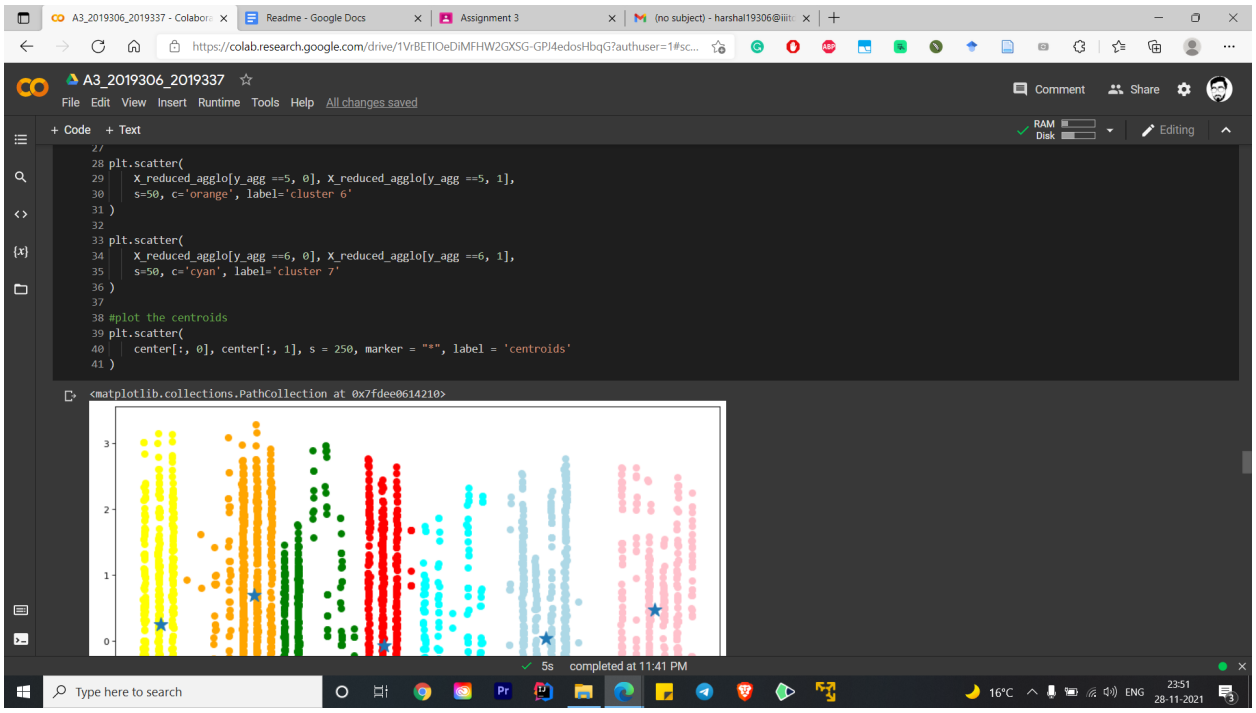
3) Since there are some columns in the dataset which have string values. We will change those strings with numbers.
4) Now since we have all of our columns having numerical values we can continue our work.
5) Now we will be doing three clusterings Algo namely- k means, agglo, spectral.
6) After that, we will do the visualization for each algorithm we have implemented in the previous steps.
7) Then we will compare the count of points with target and our predictions
8) Here we will obtain a gaussian-based model with the other three models.

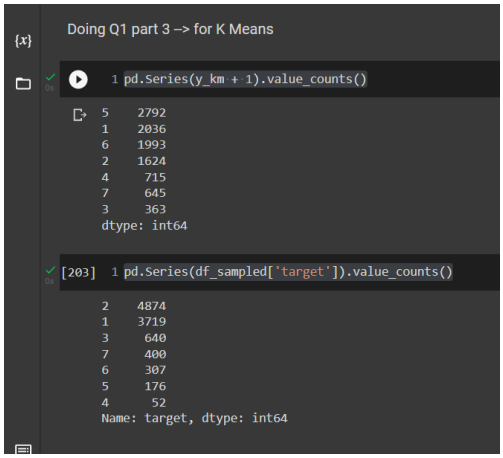
Outcomes-
Solution 1)

K Means



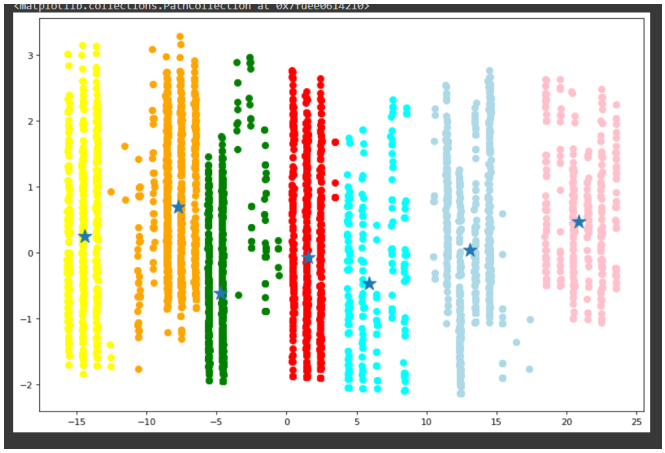
Q1) 1st and 2nd part - We've plotted the graph with clusters and centroids for k means .



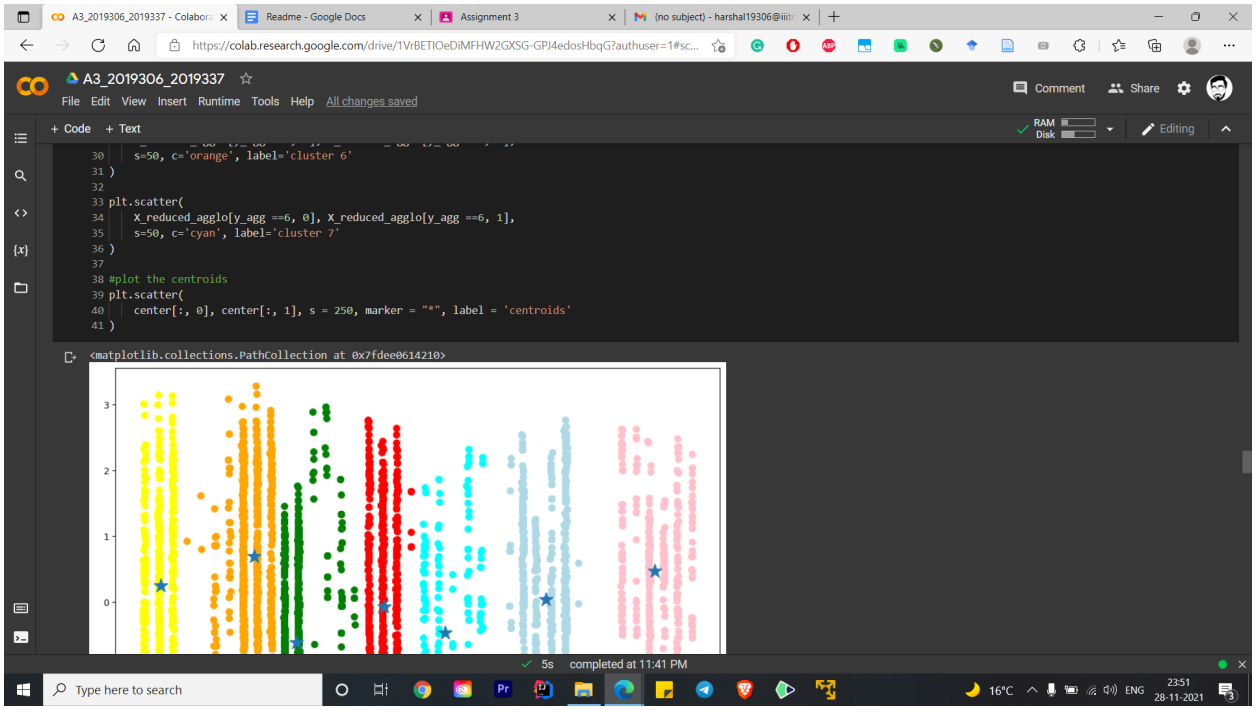


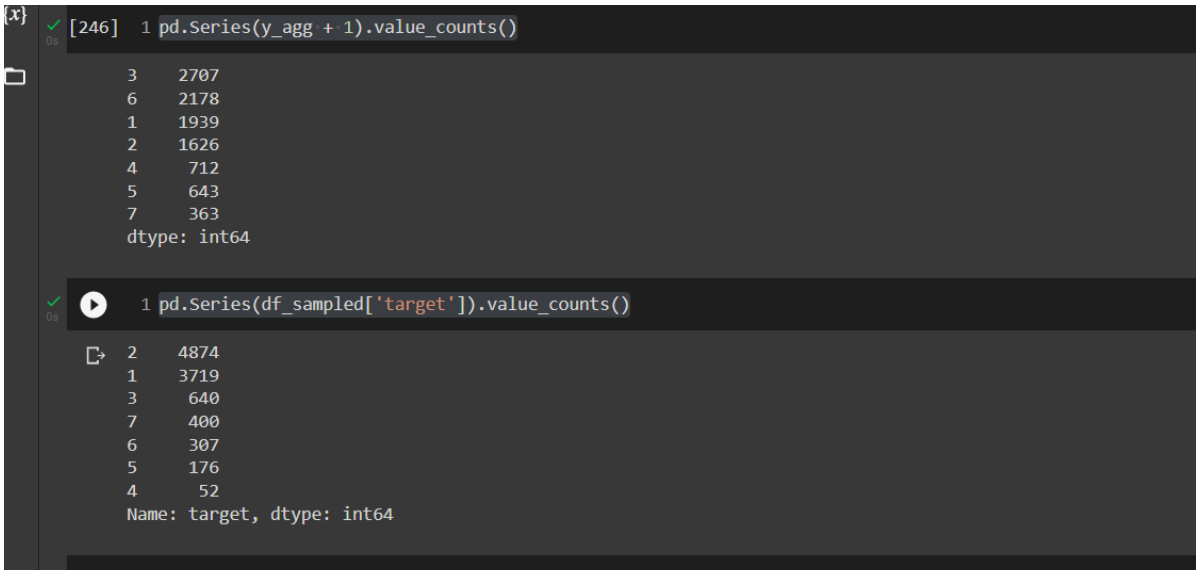
Q1 3rd part - Cluster with target = 5 is the most populated, whereas originally it is cluster with target = 2 that is the most populated

Agglomerative Clustering



Q1) 1st and 2nd part - We've plotted the graph with clusters and centroids for agglomerative clustering .

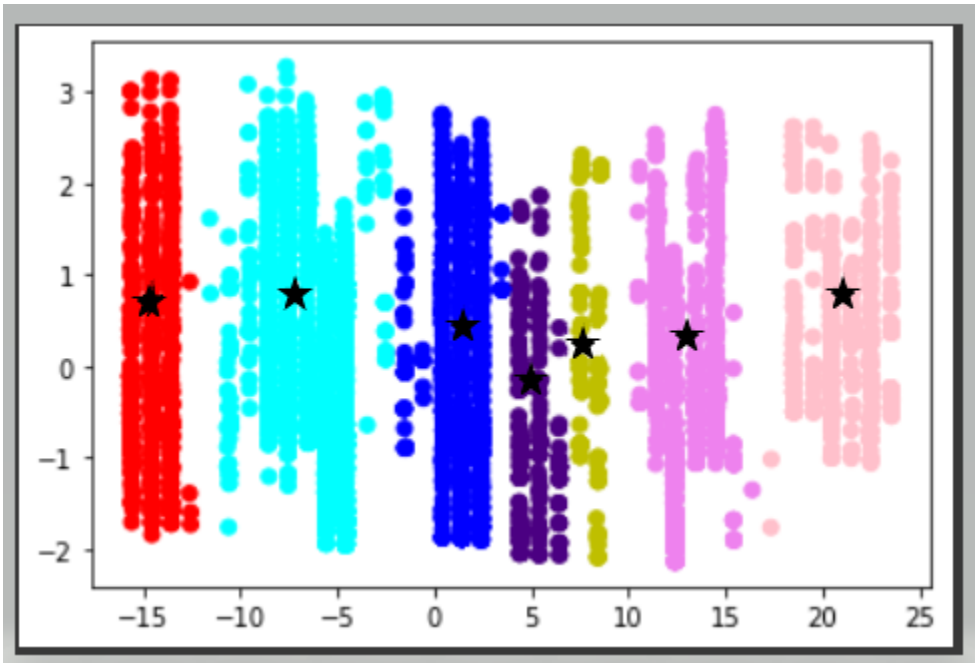




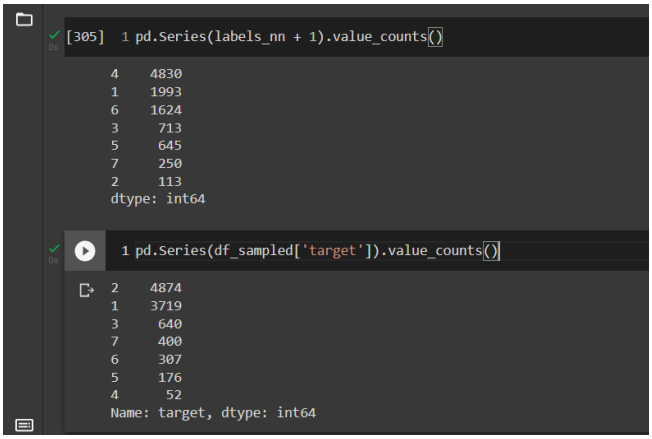
Q1 3rd part - Cluster with target = 3 is the most populated, whereas originally it is cluster with target = 2 that is the most populated

---

Spectral Clustering

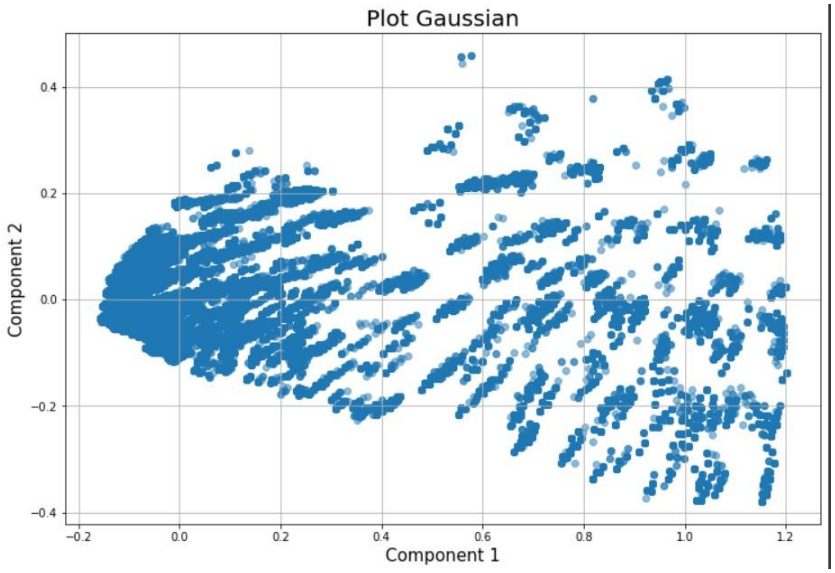


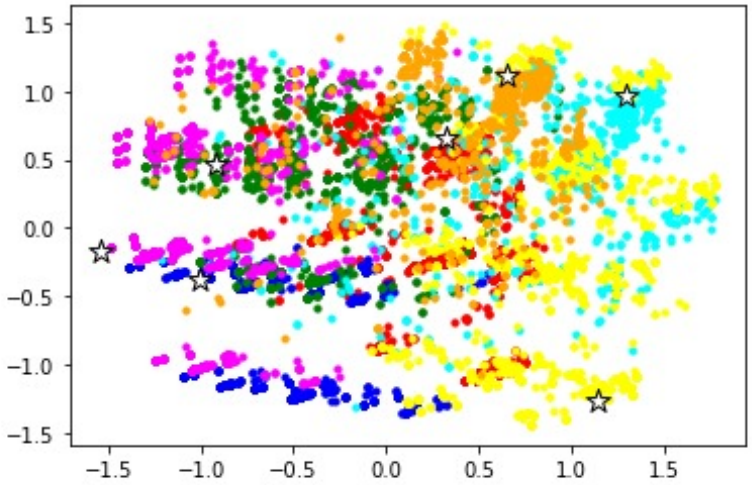
Q1) 1st and 2nd part - We've plotted the graph with clusters and centroids for spectral clustering



Q1 3rd part - Cluster with target = 4 is the most populated, whereas originally it is cluster with target = 2 that is the most populated

Gaussian based clustering model -





Q1) 1st and 2nd and 3rd part - We've plotted the graph with clusters and centroids for gaussian.

Q1 4th part
Gaussian seems to be more accurate compared to all the graphs
Kmeans and gaussian cluster are strikingly similar

Solution 2)

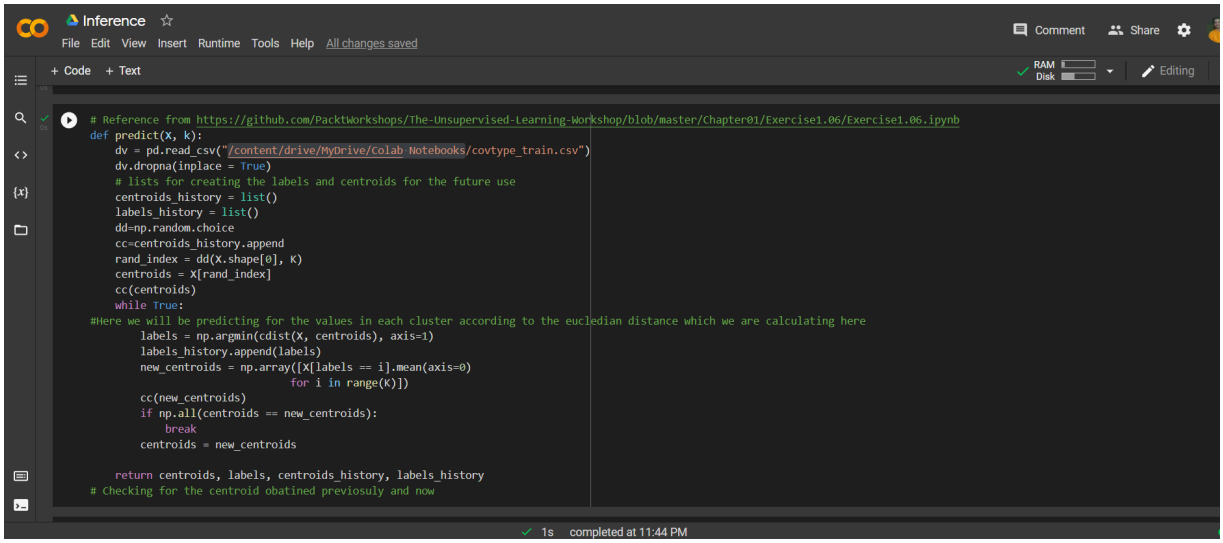
```
[49] X_mat = dx.values

[50] #np.random.seed(0)
     centroids, labels, centroids_history, labels_history = predict(X_mat, 7)

[51] foo = np.array(labels)
     m = np.median(foo[foo > 0])
     foo[foo == 0] = int(m)

     sdd=sm.accuracy_score(y_true.to_numpy(), foo) * 100
     sdd

     44.652182892886294
```



We are sending clusters =7. As we have 7 target values to prefdict. Then we are comparing the and calculating the accuracy.