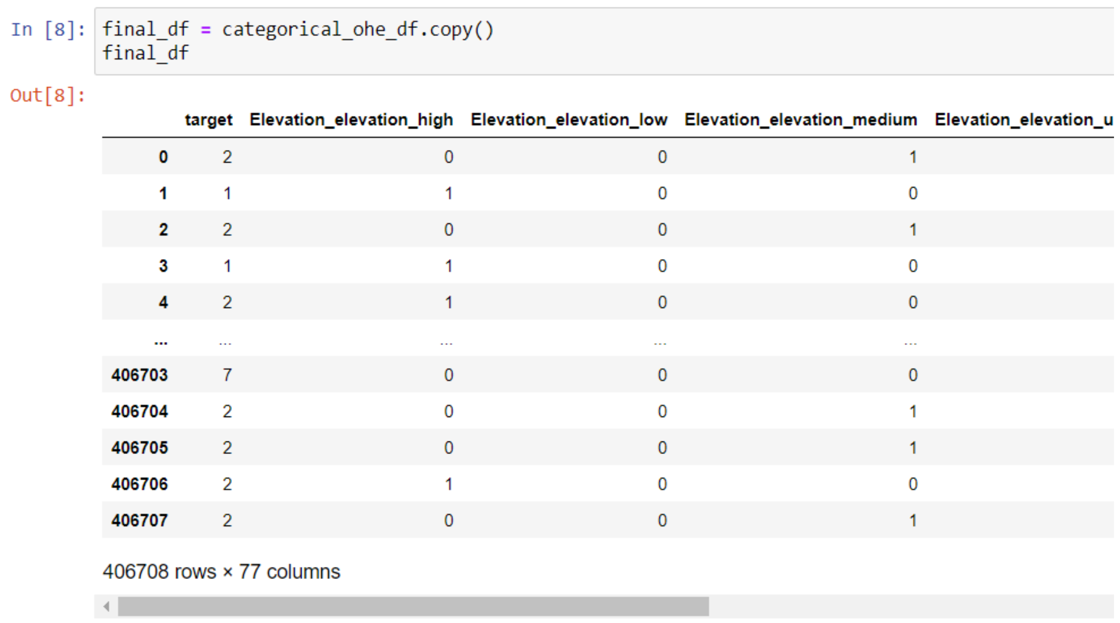
DMG Assignment 3 Report

Shashank Chaurasia 2019105
Vaibhav Girish 2019121

Q1. After importing the dataset, we found no null values and saw that all values in all columns were categorical in nature. So we one-hot encoded the data except for the 'target' column which denotes our true label value.



```
In [8]: final_df = categorical_ohe_df.copy()
        final_df
```

Out[8]:

	target	Elevation_elevation_high	Elevation_elevation_low	Elevation_elevation_medium	Elevation_elevation_u
0	2	0	0	1	
1	1	1	0	0	
2	2	0	0	1	
3	1	1	0	0	
4	2	1	0	0	
...	...	...	...	...	
406703	7	0	0	0	
406704	2	0	0	1	
406705	2	0	0	1	
406706	2	1	0	0	
406707	2	0	0	1	

406708 rows × 77 columns

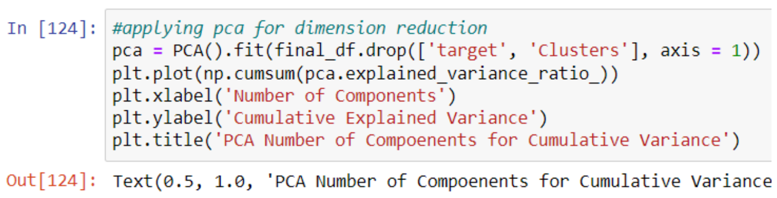
We are left with 77 columns of data. Out of which 76 are features. This is highly dimensional data so we will be using PCA for dimensionality reduction in all of our models.

All algorithms used were imported from the sklearn library in Q1.

We used 4 different types of clustering algorithms: Hierarchical (Agglomerative Clustering), Centroid based clustering (K-means), Density based clustering (DBSCAN) and Distribution based clustering (Gaussian Mixture).

K-means clustering: (Centroid based)
For this step, we use random initialization in Kmeans implementation in sklearn library. Default is Kmeans++ which is not what we are aiming for. We defined the number of clusters as 7 and max_iter as 1000 to get a nice convergence.

Clusters and centroids were obtained from the model and PCA was used to reduce the high dimensional clusters and centroids to 2 Dimensions so it can be visualised.



```
In [124]: #applying pca for dimension reduction
          pca = PCA().fit(final_df.drop(['target', 'Clusters'], axis = 1))
          plt.plot(np.cumsum(pca.explained_variance_ratio_))
          plt.xlabel('Number of Components')
          plt.ylabel('Cumulative Explained Variance')
          plt.title('PCA Number of Compoenents for Cumulative Variance')

Out[124]: Text(0.5, 1.0, 'PCA Number of Compoenents for Cumulative Variance
```

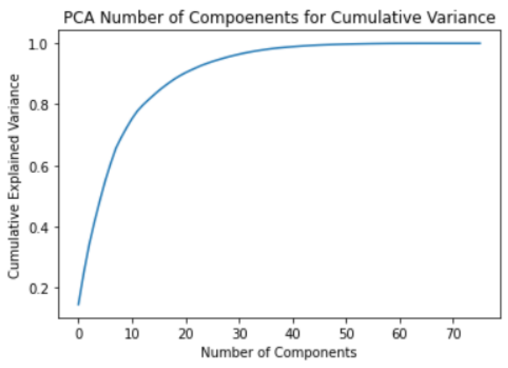
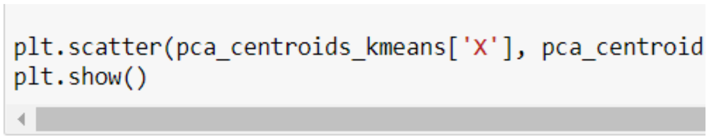


Figure 1: PCA done for all models separately

Visualisations were plotted as followed:



```
plt.scatter(pca_centroids_kmeans['X'], pca_centroid
plt.show()
```

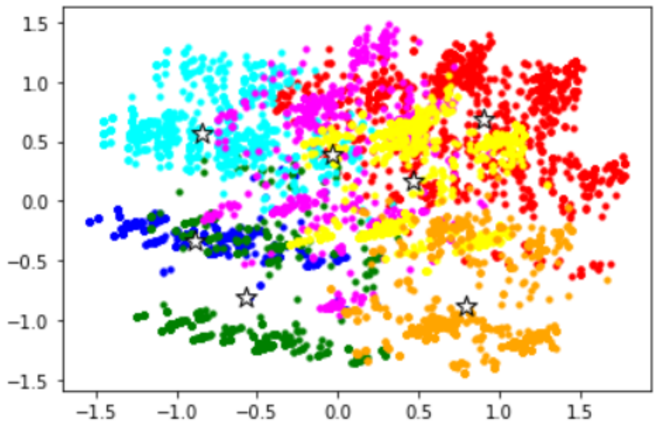


Figure 2: K-means clusters with centroids as white stars with black borders

Gaussian Mixture Modelling (Distribution based):

Model was made with PCA reduced data and plotted as follows:

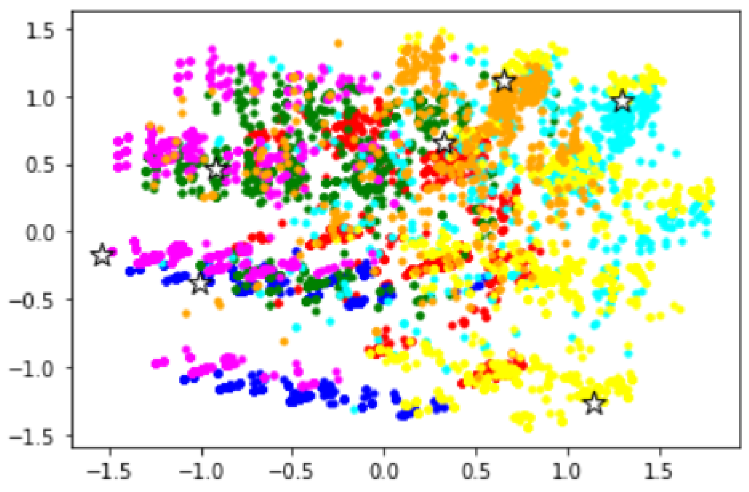
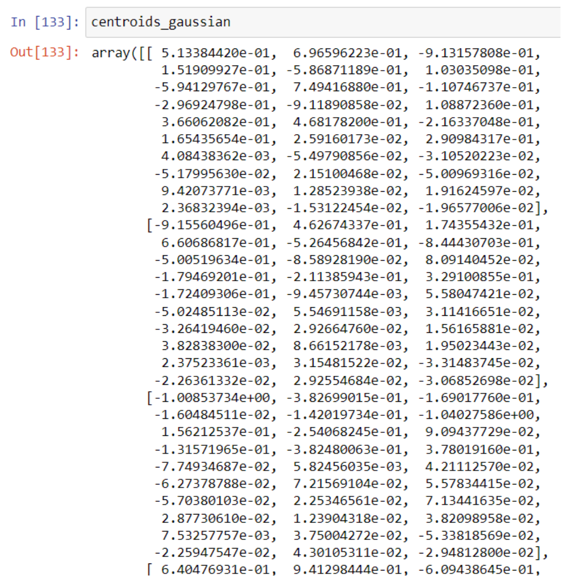


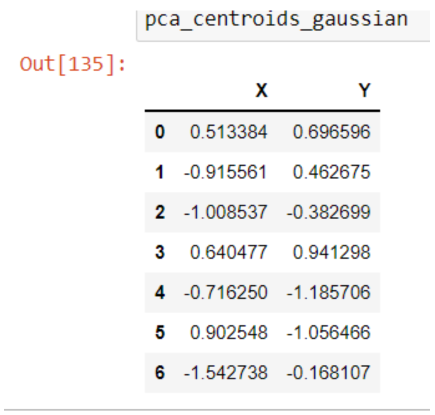
Figure 3: Gaussian Mixture clusters with its representatives as white stars with black borders

For the representative in Gaussian Mixture, we calculated the highest density point in every cluster using the mean and covariance of the cluster which is provided by the model. We did so because gaussian mixture is a density based clustering model. We got the following as the representative of the Gaussian model:



```
In [133]: centroids_gaussian

Out[133]: array([[ 5.13384420e-01,  6.96596223e-01, -9.13157808e-01,
                  1.51909927e-01, -5.86871189e-01,  1.03035098e-01,
                 -5.94129767e-01,  7.49416880e-01, -1.10746737e-01,
                 -2.96924798e-01, -9.11890858e-02,  1.08872360e-01,
                  3.66062082e-01,  4.68178200e-01, -2.16337048e-01,
                  1.65435654e-01,  2.59160173e-02,  2.90984317e-01,
                  4.08438362e-03, -5.49790856e-02, -3.10520223e-02,
                 -5.17995630e-02,  2.15100468e-02, -5.00969316e-02,
                  9.42073771e-03,  1.28523938e-02,  1.91624597e-02,
                  2.36832394e-03, -1.53122454e-02, -1.96577006e-02],
                [-9.15560496e-01,  4.62674337e-01,  1.74355432e-01,
                  6.60686817e-01, -5.26456842e-01, -8.44430703e-01,
                 -5.00519634e-01, -8.58928190e-02,  8.09140452e-02,
                 -1.79469201e-01, -2.11385943e-01,  3.29100855e-01,
                 -1.72409306e-01, -9.45730744e-03,  5.58047421e-02,
                 -5.02485113e-02,  5.54691158e-03,  3.11416651e-02,
                 -3.26419460e-02,  2.92664760e-02,  1.56165881e-02,
                  3.82838300e-02,  8.66152178e-03,  1.95023443e-02,
                  2.37523361e-03,  3.15481522e-02, -3.31483745e-02,
                 -2.26361332e-02,  2.92554684e-02, -3.06852698e-02],
                [-1.00853734e+00, -3.82699015e-01, -1.69017760e-01,
                 -1.60484511e-02, -1.42019734e-01, -1.04027586e+00,
                  1.56212537e-01, -2.54068245e-01,  9.09437729e-02,
                 -1.31571965e-01, -3.82480063e-01,  3.78019160e-01,
                 -7.74934687e-02,  5.82456035e-03,  4.21112570e-02,
                 -6.27378788e-02,  7.21569104e-02,  5.57834415e-02,
                 -5.70380103e-02,  2.25346561e-02,  7.13441635e-02,
                  2.87730610e-02,  1.23904318e-02,  3.82098958e-02,
                  7.53257757e-03,  3.75004272e-02, -5.33818569e-02,
                 -2.25947547e-02,  4.30105311e-02, -2.94812800e-02],
                [ 6.40476931e-01,  9.41298444e-01, -6.09438645e-01,
```

After PCA,



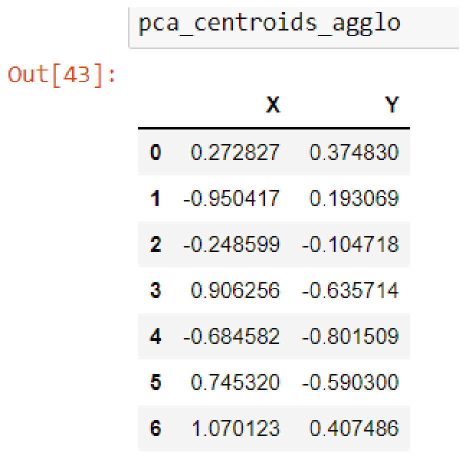
```
pca_centroids_gaussian

Out[135]:
```

	X	Y
0	0.513384	0.696596
1	-0.915561	0.462675
2	-1.008537	-0.382699
3	0.640477	0.941298
4	-0.716250	-1.185706
5	0.902548	-1.056466
6	-1.542738	-0.168107

Agglomerative Clustering: (Hierarchical)

For agglomerative clustering, we needed to reduce our data size as memory error occurred when we used our large dataset on this algorithm. So we sampled 40000 data points for this model and applied PCA for dimensionality reduction.

For representative, hierarchical clustering doesn't have a representative of its own. However we have calculated the centroid of each cluster using the NearestCentroid algorithm in the neighbours library of sklearn. The following are the centroids after PCA:



```
pca_centroids_agglo
```

Out[43]:

	X	Y
0	0.272827	0.374830
1	-0.950417	0.193069
2	-0.248599	-0.104718
3	0.906256	-0.635714
4	-0.684582	-0.801509
5	0.745320	-0.590300
6	1.070123	0.407486

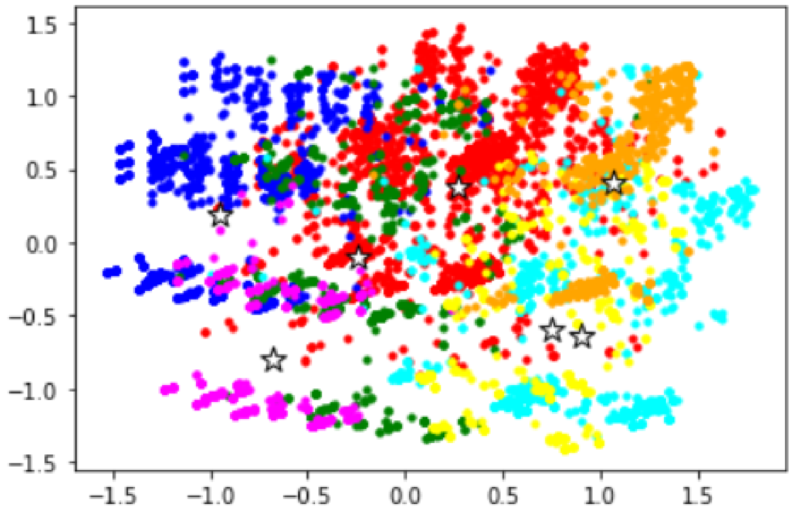


Figure 4: Agglomerative Clusters with its centroids as white stars with black borders

DBSCAN (Density based):

For this model, we had to do some hyperparameter tuning so that we get exactly 7 clusters. We end up getting 8 because the cluster labels marked as -1 are not classified into any clusters. The following is the plot of the clusters found by density based clustering algorithm, DBSCAN:

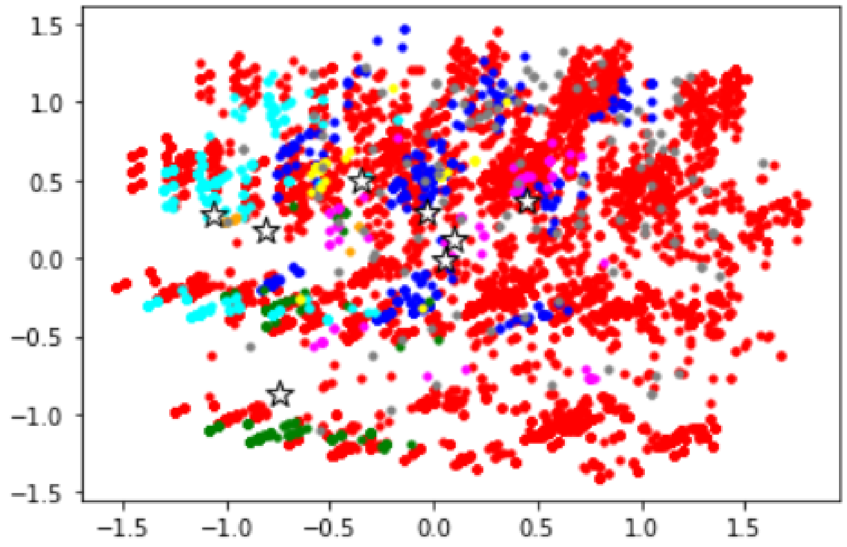
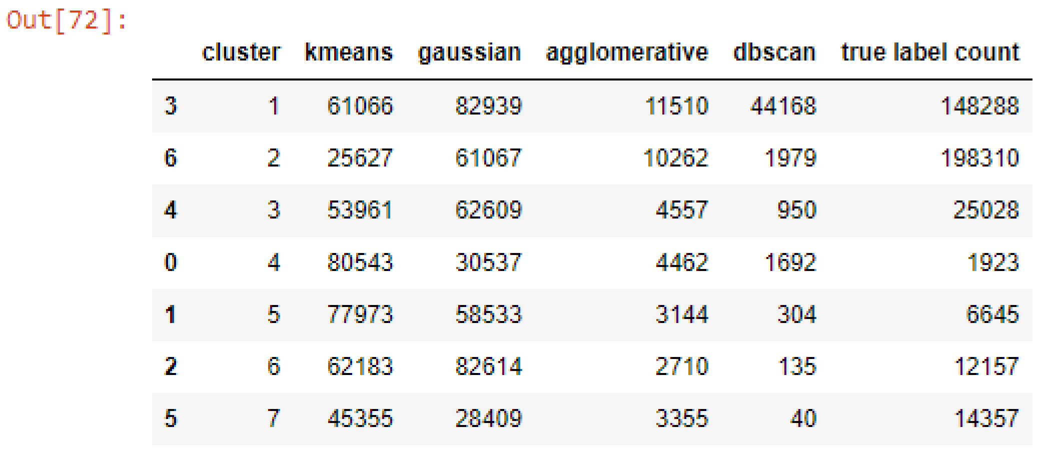


Figure 5: DBSCAN clusters

Since DBSCAN is a density based clustering model, it doesn't have a representative. Density based models have arbitrarily shaped clusters. Hence computing centroid is a waste of resources. However we have calculated the centroid of each cluster similar to the way we calculated for agglomerative clustering.

Comparison of cluster distribution with true label count:



Out[72]:

	cluster	kmeans	gaussian	agglomerative	dbscan	true label count
3	1	61066	82939	11510	44168	148288
6	2	25627	61067	10262	1979	198310
4	3	53961	62609	4557	950	25028
0	4	80543	30537	4462	1692	1923
1	5	77973	58533	3144	304	6645
2	6	62183	82614	2710	135	12157
5	7	45355	28409	3355	40	14357

Since the clusters were output in random order, there is no way to accurately compare the size of clusters with the true label count. Also the original data was sampled down to 50000, 40000 data points respectively for DBSCAN and agglomerative clustering because of memory error so a proper comparison cannot be made.

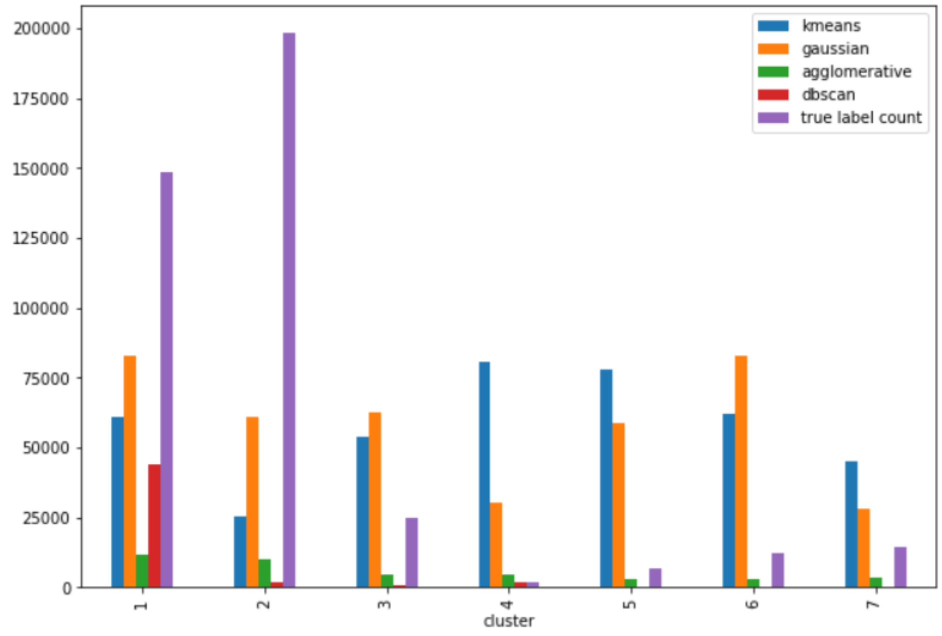
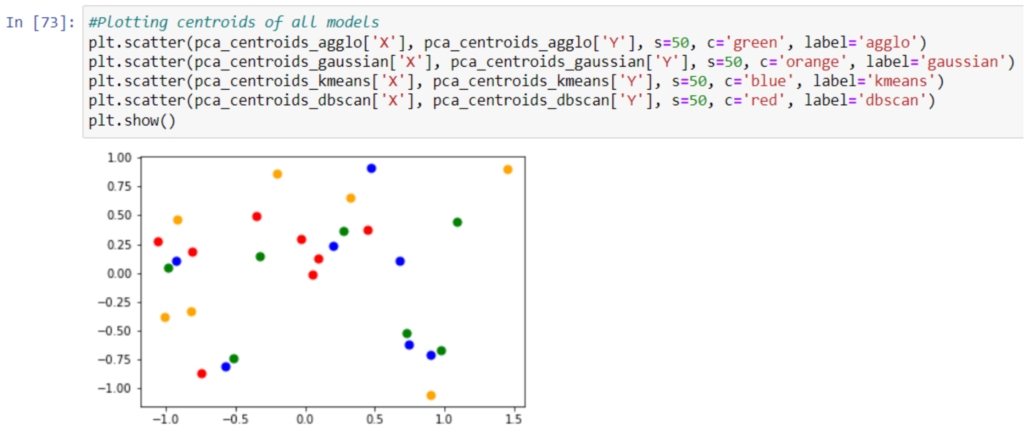


Figure 6: Multibar graph showing cluster counts vs true label counts



```
In [73]: #Plotting centroids of all models
plt.scatter(pca_centroids_agglo['X'], pca_centroids_agglo['Y'], s=50, c='green', label='agglo')
plt.scatter(pca_centroids_gaussian['X'], pca_centroids_gaussian['Y'], s=50, c='orange', label='gaussian')
plt.scatter(pca_centroids_kmeans['X'], pca_centroids_kmeans['Y'], s=50, c='blue', label='kmeans')
plt.scatter(pca_centroids_dbscan['X'], pca_centroids_dbscan['Y'], s=50, c='red', label='dbscan')
plt.show()
```

Figure 7: Centroids comparisons

Question 1 Part 4.

Gaussian vs other clustering algorithms

We used Kmeans, DBSCAN and agglomerative clustering. These are all different types of algorithms with different or no representatives of clusters. When looking at Figure 6, we can infer that Gaussian clustering and Kmeans clustering gives clusters of roughly the same size.

However when we compare the true label counts with gaussian and k means we find that gaussian holds truer label counts than Kmeans. So we can say that gaussian is a better clustering algorithm for the given dataset than the others.

Since the clusters are PCA transformed from very high dimensions to 2 dimensions it is hard to say which clusters are better. However due to the reasoning above, we conclude that Gaussian is the best.

DBSCAN gave the worst outcome as it is a density based clustering algorithm.

Question 2:

For this question we used the model we fit through the gaussian mix algorithm. We saved the model that fit on the previous dataset. In the function predict, we parsed the dataset, one hot encoded the dataset and then pca fit it. After pca fit, we loaded the saved model from the previous question. We then predicted using the loaded model and returned the answer.