CSE1015 – Machine Learning Essentials

LAB – 8

Pinni Venkata Abhiram
20BAI1132

Experiment – 8 K-Means Clustering

Pinni Venkata Abhiram

20BAI1132

Introduction

Building a K Means Clustering model to classify the given data.

The fields in the data are - 'P1', 'P2', 'P3', 'P4', 'P5', 'P6', 'P7', 'P8', 'P9', 'P10', 'P11', 'P12', 'P13', 'P14', 'P15', 'P16', 'P17', 'P18', 'P19', 'P20', 'Target Label'

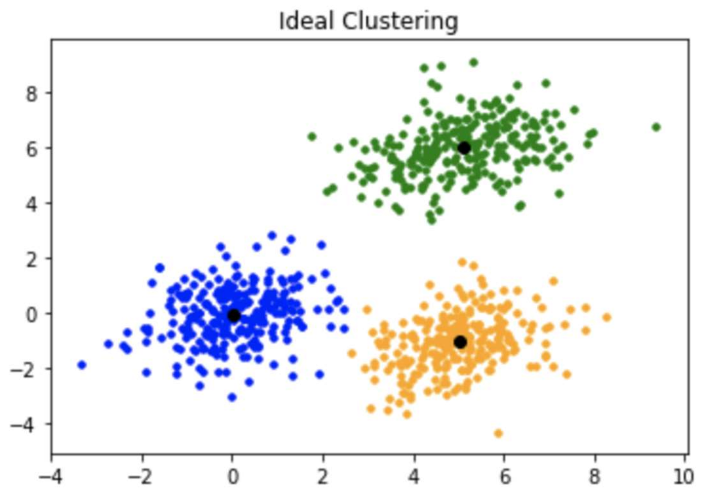
The model is built using sklearn cluster model which has the K Means Clustering Algorithm and various plots for visualising the clustering done by the K Means and box plot to view each column having the number quantity of elements per cluster provided by seaborn module and Classification report for statistical summary.

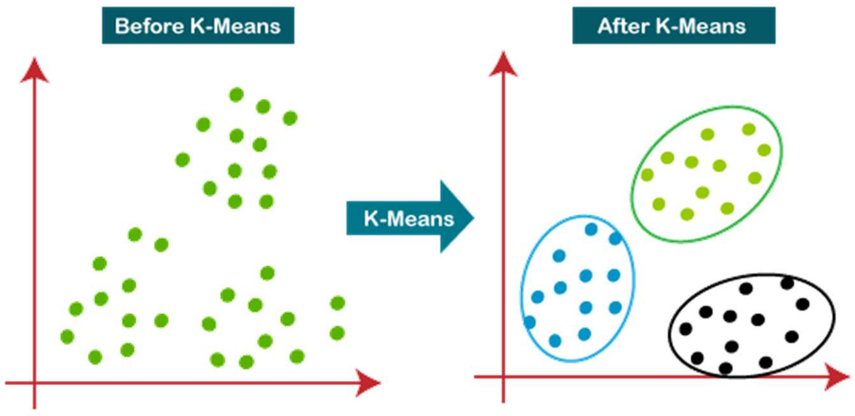
Methodology

The model is built using the concept of K Means Clustering.

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. k-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean distances. For instance, better Euclidean solutions can be found using k-medians and k-medoids.

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. Typically, unsupervised algorithms make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.


Ideal Clustering
8
6
4
2
0
−2
−4
−4
−2
0
2
4
6
8
10

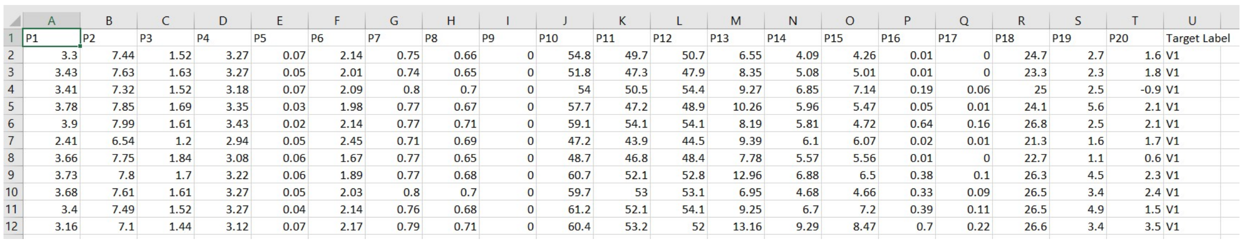

Before K-Means
K-Means
After K-Means

Dataset

The dataset used here is a .xlsx file which contains 2 .csv files which we can extract and get the values for the train set and the test set.

The columns in the test set and train set are 'P1', 'P2', 'P3', 'P4', 'P5', 'P6', 'P7', 'P8', 'P9', 'P10', 'P11', 'P12', 'P13', 'P14', 'P15', 'P16', 'P17', 'P18', 'P19', 'P20', 'Target Label'.

The test set has 100 rows and 21 columns , and the train set has 400 rows and 21 columns.

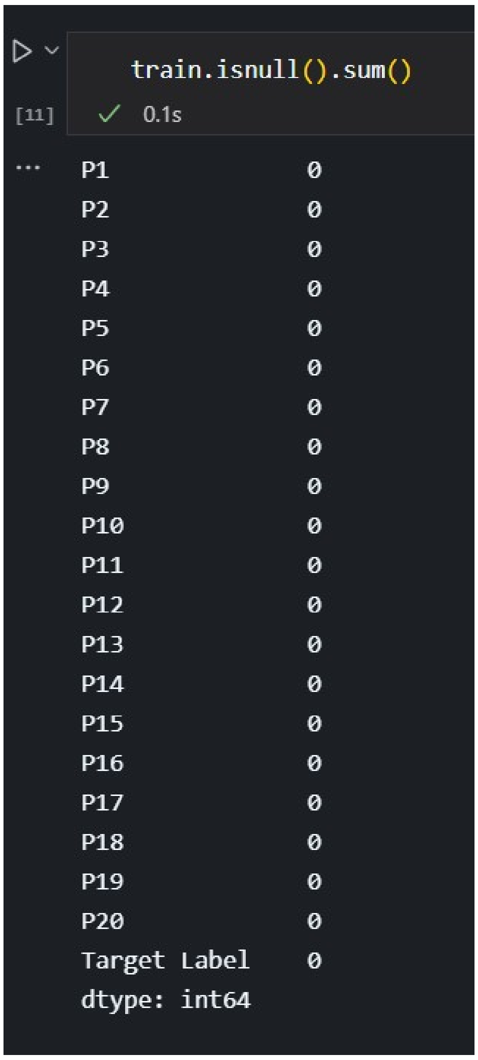


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	Target Label
2	3.3	7.44	1.52	3.27	0.07	2.14	0.75	0.66	0	54.8	49.7	50.7	6.55	4.09	4.26	0.01	0	24.7	2.7	1.6	V1
3	3.43	7.63	1.63	3.27	0.05	2.01	0.74	0.65	0	51.8	47.3	47.9	8.35	5.08	5.01	0.01	0	23.3	2.3	1.8	V1
4	3.41	7.32	1.52	3.18	0.07	2.09	0.8	0.7	0	54	50.5	54.4	9.27	6.85	7.14	0.19	0.06	25	2.5	-0.9	V1
5	3.78	7.85	1.69	3.35	0.03	1.98	0.77	0.67	0	57.7	47.2	48.9	10.26	5.96	5.47	0.05	0.01	24.1	5.6	2.1	V1
6	3.9	7.99	1.61	3.43	0.02	2.14	0.77	0.71	0	59.1	54.1	54.1	8.19	5.81	4.72	0.64	0.16	26.8	2.5	2.1	V1
7	2.41	6.54	1.2	2.94	0.05	2.45	0.71	0.69	0	47.2	43.9	44.5	9.39	6.1	6.07	0.02	0.01	21.3	1.6	1.7	V1
8	3.66	7.75	1.84	3.08	0.06	1.67	0.77	0.65	0	48.7	46.8	48.4	7.78	5.57	5.56	0.01	0	22.7	1.1	0.6	V1
9	3.73	7.8	1.7	3.22	0.06	1.89	0.77	0.68	0	60.7	52.1	52.8	12.96	6.88	6.5	0.38	0.1	26.3	4.5	2.3	V1
10	3.68	7.61	1.61	3.27	0.05	2.03	0.8	0.7	0	59.7	53	53.1	6.95	4.68	4.66	0.33	0.09	26.5	3.4	2.4	V1
11	3.4	7.49	1.52	3.27	0.04	2.14	0.76	0.68	0	61.2	52.1	54.1	9.25	6.7	7.2	0.39	0.11	26.5	4.9	1.5	V1
12	3.16	7.1	1.44	3.12	0.07	2.17	0.79	0.71	0	60.4	53.2	52	13.16	9.29	8.47	0.7	0.22	26.6	3.4	3.5	V1

Experiments and Results

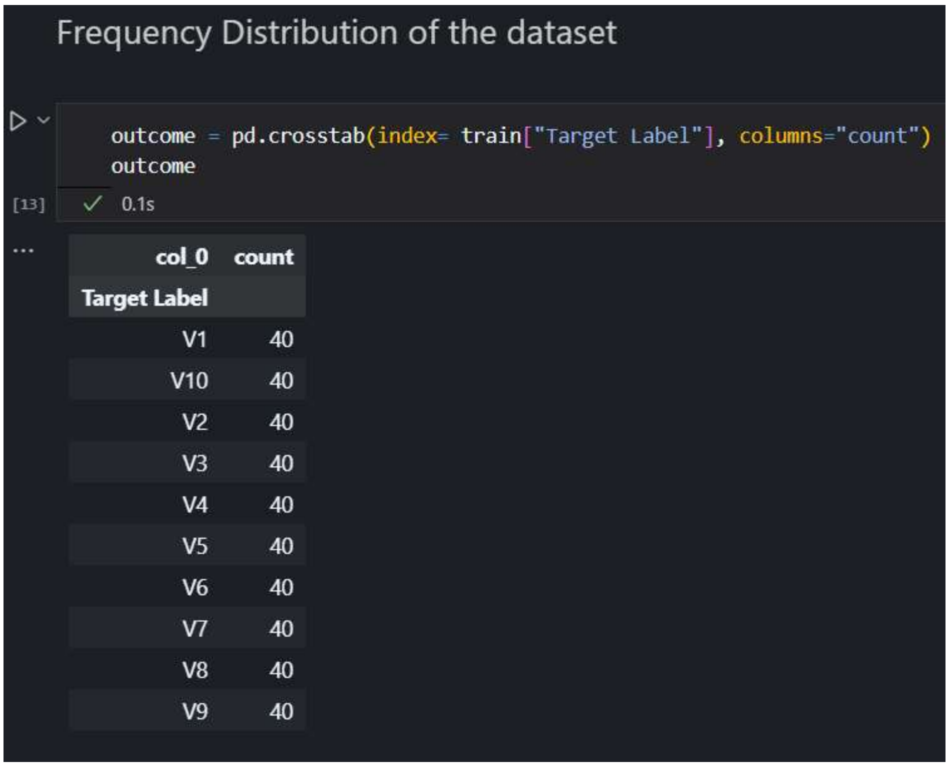
The experiment required numpy , pandas , matplotlib , seaborn and sklearn modules to implement

We do the preprocessing of the datasets i.e. train and the test dataset. Reading the dataset and dropping all the null rows if they exist. In this case there are no null rows or columns so we go ahead with describing and getting info of the dataset.



```
train.isnull().sum()
[11]  ✓ 0.1s
```

```
P1              0
P2              0
P3              0
P4              0
P5              0
P6              0
P7              0
P8              0
P9              0
P10             0
P11             0
P12             0
P13             0
P14             0
P15             0
P16             0
P17             0
P18             0
P19             0
P20             0
Target Label    0
dtype: int64
```



Frequency Distribution of the dataset

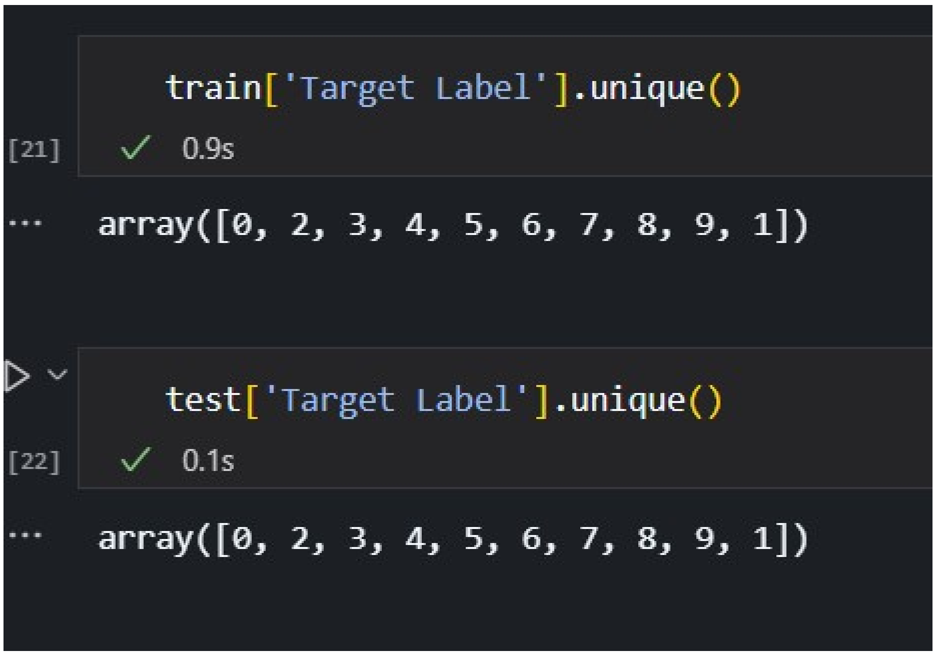
```
outcome = pd.crosstab(index= train["Target Label"], columns="count")
outcome
```

[13] ✓ 0.1s

col_0	count
Target Label	
V1	40
V10	40
V2	40
V3	40
V4	40
V5	40
V6	40
V7	40
V8	40
V9	40

Given the labels are strings and we can't work with strings so we encode the labels using the Sklearn's Module LabelEncoder and it will make all the target labels into numbers.

In the end we get the labels in this way.



```
train['Target Label'].unique()
```

[21] ✓ 0.9s

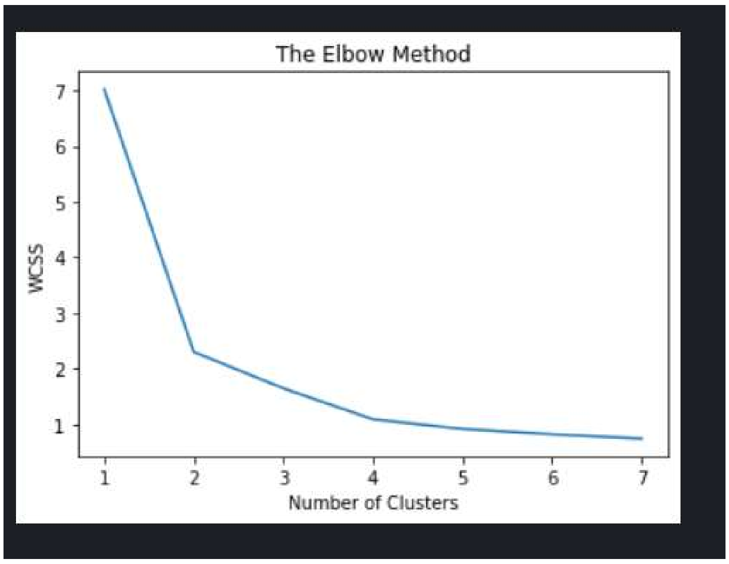
```
array([0, 2, 3, 4, 5, 6, 7, 8, 9, 1])
```

```
test['Target Label'].unique()
```

[22] ✓ 0.1s

```
array([0, 2, 3, 4, 5, 6, 7, 8, 9, 1])
```

For the train set we normalise it using the normalise function in sklearn and we plot the elbow graph to check the optimal number of clusters.



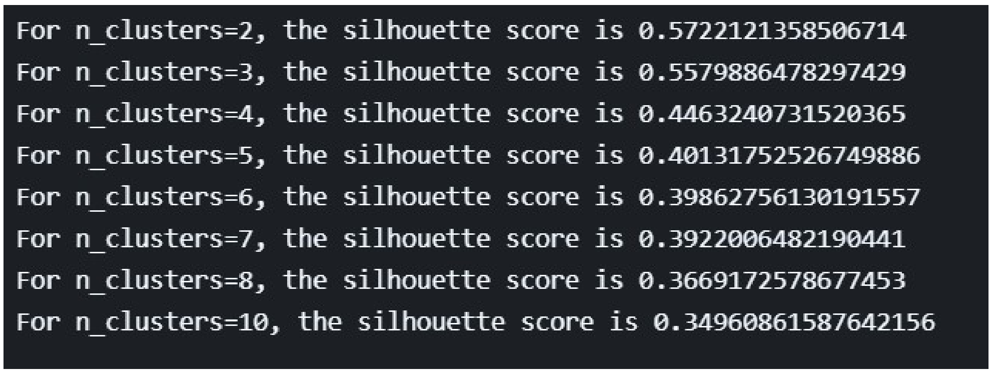
From the above plot, we observe, while the x axis value goes from 1 to 2 the WCSS decreases rapidly, and while the x axis value goes from 2 to 7, the WCSS decreases slowly.

This tells us the optimal number of clusters are 2.

For getting the optimal number of clusters we can use another method called silhouette score by sklearn module.

The value of the silhouette score range lies between -1 to 1. A score closer to 1 indicates that the data point is very similar to other data points in the cluster, A score closer to -1 indicates that the data point is not similar to the data points in its cluster.

The silhouette scores are as follows.


```
For n_clusters=2, the silhouette score is 0.5722121358506714
For n_clusters=3, the silhouette score is 0.5579886478297429
For n_clusters=4, the silhouette score is 0.4463240731520365
For n_clusters=5, the silhouette score is 0.40131752526749886
For n_clusters=6, the silhouette score is 0.39862756130191557
For n_clusters=7, the silhouette score is 0.3922006482190441
For n_clusters=8, the silhouette score is 0.3669172578677453
For n_clusters=10, the silhouette score is 0.34960861587642156
```

Applying the K Means Clustering algorithm with number of clusters as 2.

And fit the test set and predict the cluster which the row belongs to.

```
kmeans = KMeans(n_clusters = 2, init = 'k-means++', random_state = 42)
X = test.iloc[:, [3, 5]].values
y_kmeans = kmeans.fit_predict(X)
cluster_labels = kmeans.labels_
label = y_kmeans
```

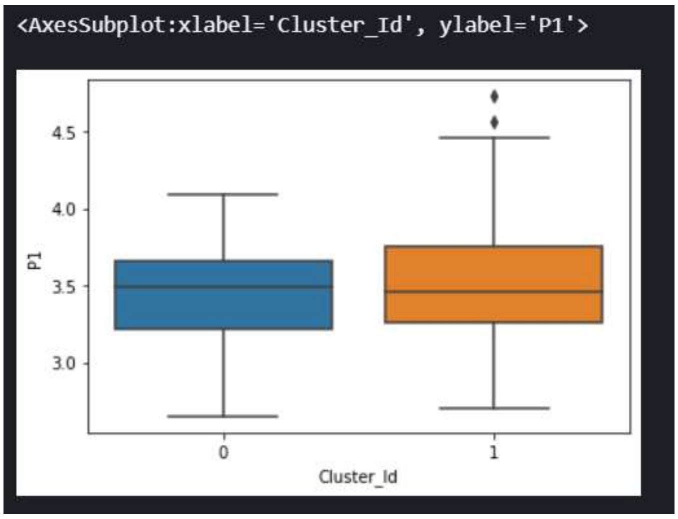
Predicted Values , 0 means Cluster – 1 and 1 means Cluster – 2

```
[1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 1 1 1 1 0 0 1 1 0 0 1
 1 1 1 1 1 0 1 1 0 0 1 0 0 1 0 1 1 1 0 0 0 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 1
 0 1 1 0 0 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0]
```

Plots and Visuals

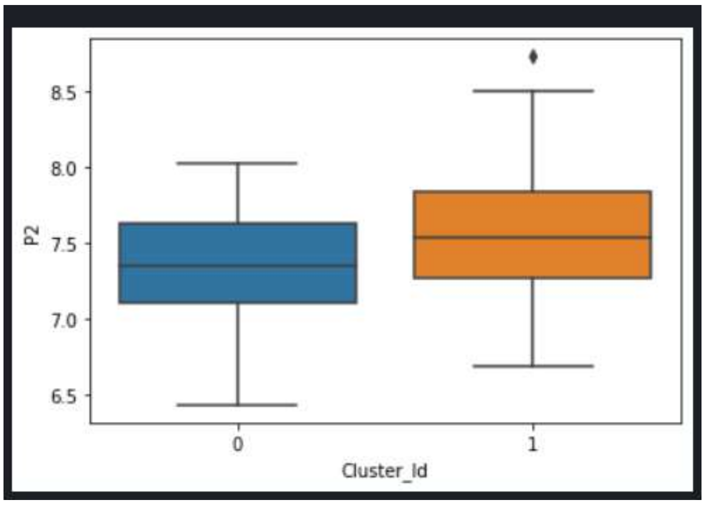
This boxplots will tell how many cluster ID and P1/P2/P3/P4/P5/P6 and so on are marked as 0 and 1 and so on.

For P1

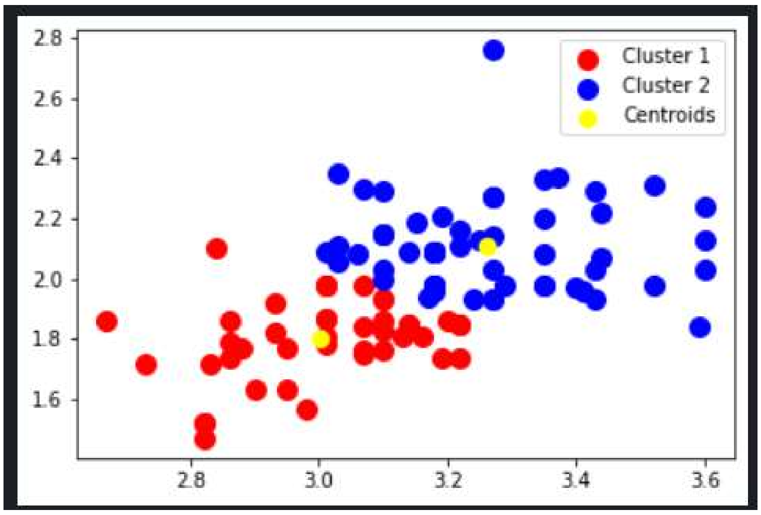


```
<AxesSubplot:xlabel='Cluster_Id', ylabel='P1'>
```

For P2

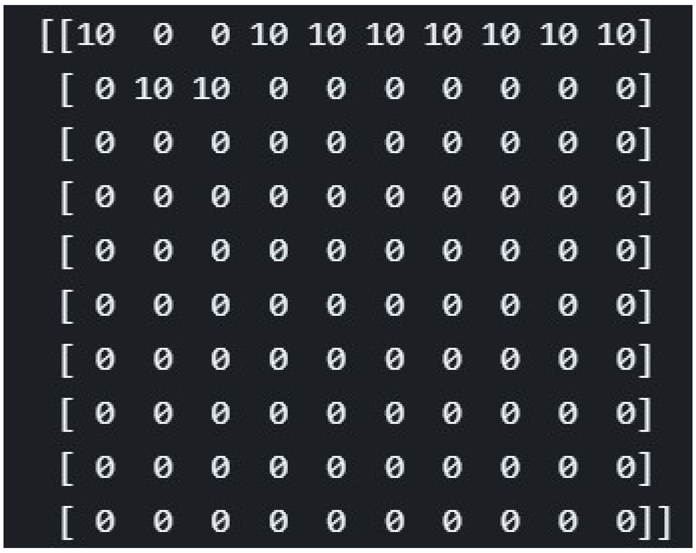


Visualising the Clusters with Centroids



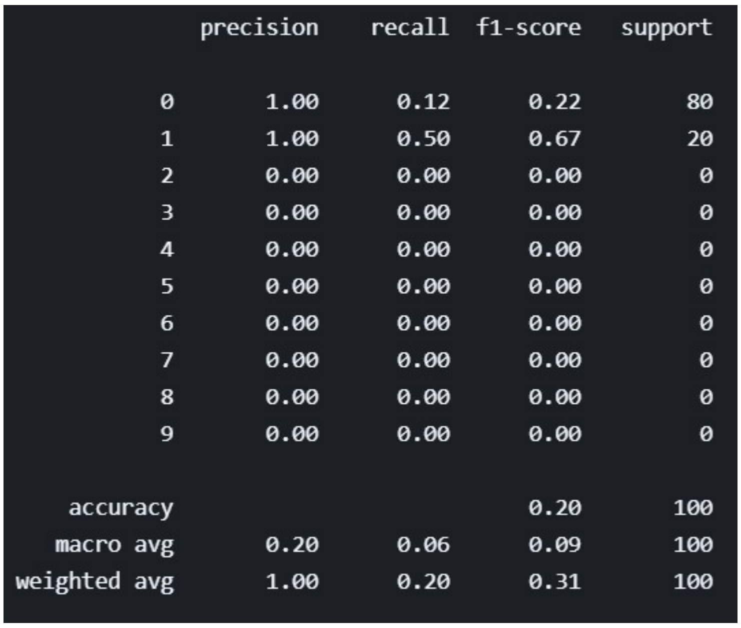
For the metrics we can draw the classification report and the confusion matrix

Confusion Matrix



```
[[10  0  0 10 10 10 10 10 10 10]
 [ 0 10 10  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0]]
```

Classification Report

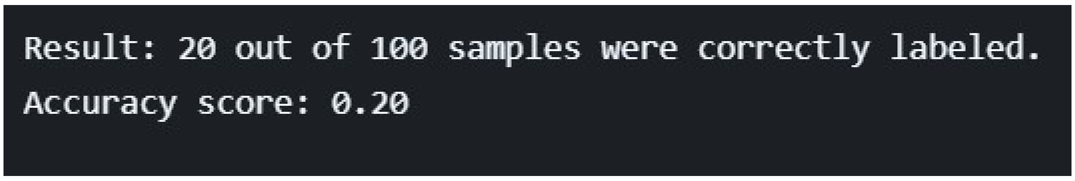


```
              precision    recall  f1-score   support

           0       1.00      0.12      0.22        80
           1       1.00      0.50      0.67        20
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       0.00      0.00      0.00         0
           5       0.00      0.00      0.00         0
           6       0.00      0.00      0.00         0
           7       0.00      0.00      0.00         0
           8       0.00      0.00      0.00         0
           9       0.00      0.00      0.00         0

    accuracy                           0.20       100
   macro avg       0.20      0.06      0.09       100
weighted avg       1.00      0.20      0.31       100
```

Final Accuracy



```
Result: 20 out of 100 samples were correctly labeled.
Accuracy score: 0.20
```

Conclusion

The K Means Clustering model of Machine Learning is successfully implemented and can predict the if the cluster with a given input and we can classify the given input / data.

References

https://en.wikipedia.org/wiki/K-means_clustering

https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning

https://www.w3schools.com/python/python_ml_scatterplot.asp

https://scikit-learn.org/stable/modules/clustering.html

https://developers.google.com/machine-learning/clustering/clustering-algorithms