Name: Aitik Gupta

Roll No: 2018IMT-010

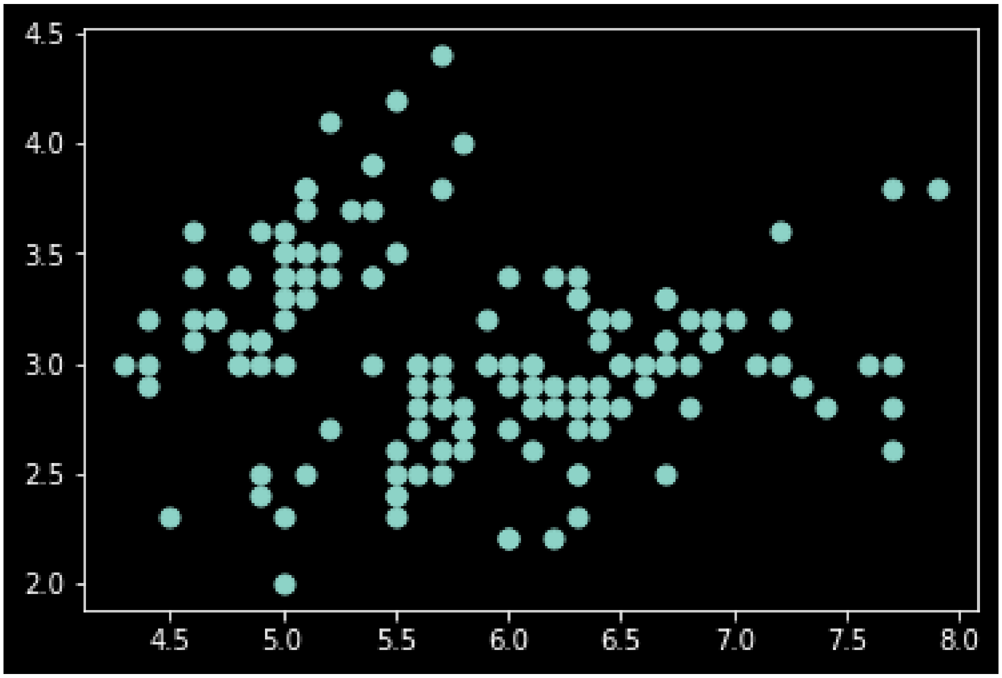
Course: ML-Lab

Course Code: ITIT - 4107

Code Link: https://github.com/aitikgupta/ITIT-4103-2021/tree/main/Assignment%206

```
# imports

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy import stats

from sklearn.datasets import load_iris
data = load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df.head()
X = data.data

Y = data.target
df = np.array(df)

plt.scatter(df[:, 0], df[:, 1], s=50)
```

<matplotlib.collections.PathCollection at 0x7f9177d5fd50>



```
pd.DataFrame(data.data, columns=data.feature_names).describe()
```

```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000   
mean            5.843333          3.057333           3.758000   
std             0.828066          0.435866           1.765298   
min             4.300000          2.000000           1.000000   
25%             5.100000          2.800000           1.600000   
50%             5.800000          3.000000           4.350000   
75%             6.400000          3.300000           5.100000   
max             7.900000          4.400000           6.900000   

       petal width (cm)  
count        150.000000  
mean           1.199333  
std            0.762238  
min            0.100000  
25%            0.300000  
50%            1.300000  
75%            1.800000  
max            2.500000  
```

K-Means Clustering

The Elbow approach was used to determine the number of clusters in our dataset.

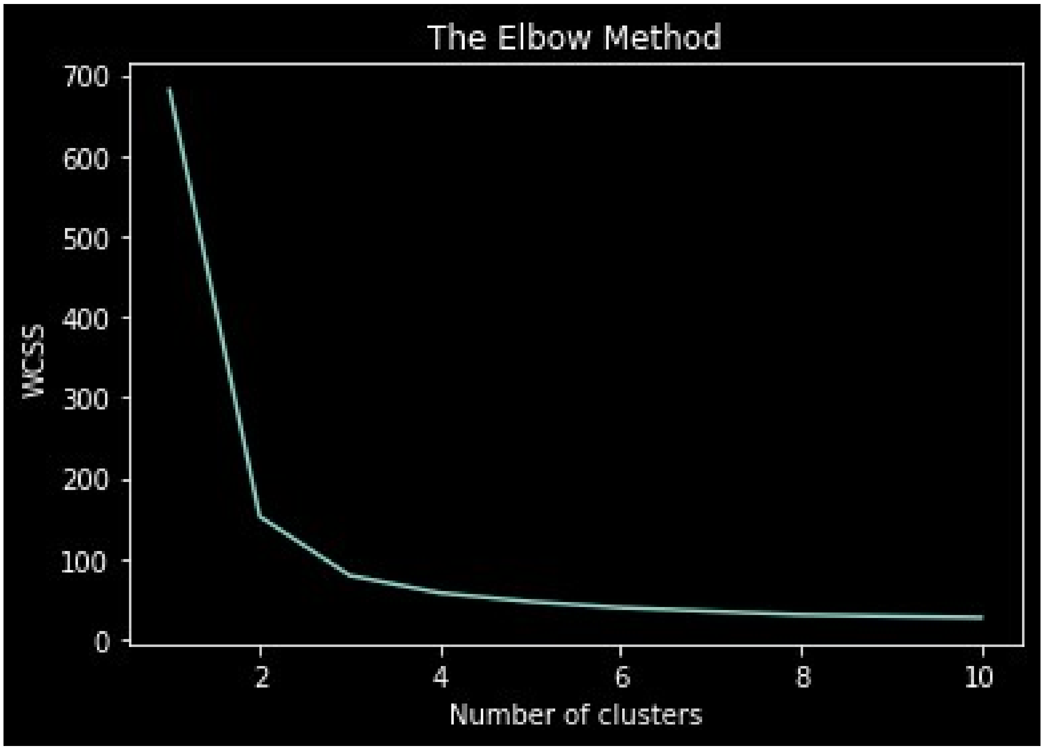
We are adjusting the number of clusters (K) in the Elbow approach from 1 to 10. We calculate WCSS for each value of K (Within-Cluster Sum of Square).

In a cluster, WCSS is the sum of squared distances between each point and the centroid. The plot appears like an Elbow when we plot the WCSS with the K value. The WCSS value will begin to fall as the number of clusters grows.

When K = 1, the WCSS value is the highest. When we examine the graph, we can see that it will shift rapidly at a point, forming an elbow shape. The graph begins to travel practically parallel to the X-axis at this point. The ideal K value, or the optimal number of clusters, corresponds to this point.

```
from sklearn.cluster import KMeans

wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(df)
    wcss.append(kmeans.inertia_)
```

```
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



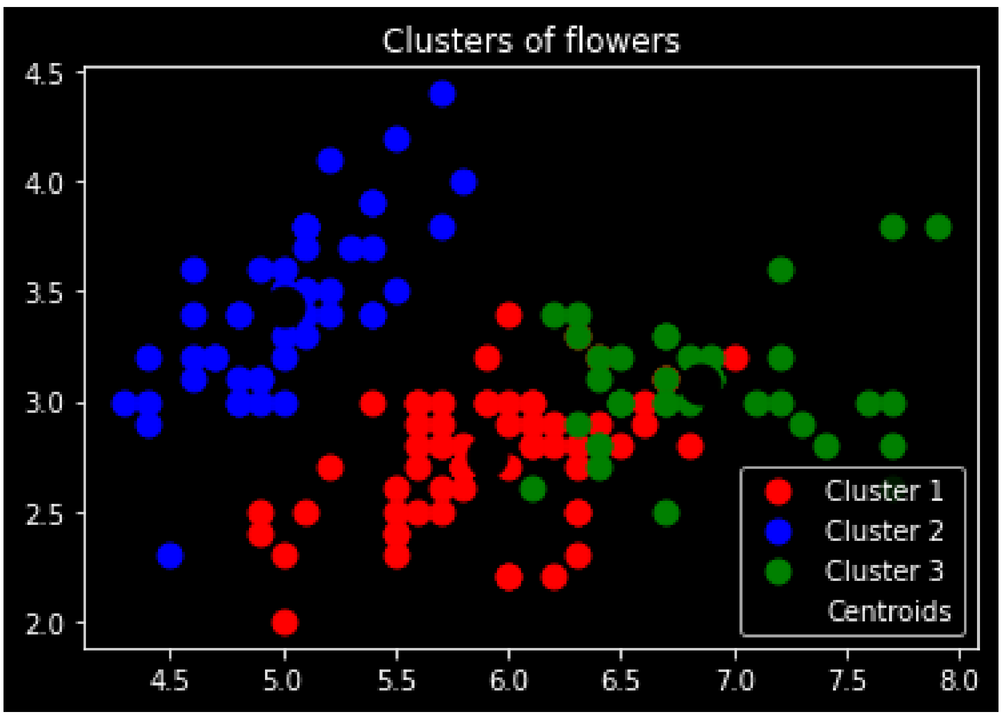
Since elbow lie at 3 on x-axis. We can conclude the no of clusters is 3.

```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
y_kmeans = kmeans.fit_predict(df)

y_kmeans

array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2,
2,
       2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2,
2,
       2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0],
dtype=int32)
```

```
from sklearn.metrics import accuracy_score

print('K-Mean model accuracy: {}'.format(accuracy_score(Y, y_kmeans)))
```

```
K-Mean model accuracy: 0.24
```

```
plt.scatter(df[y_kmeans == 0, 0], df[y_kmeans == 0, 1], s = 80, c =
'red', label = 'Cluster 1')
plt.scatter(df[y_kmeans == 1, 0], df[y_kmeans == 1, 1], s = 80, c =
'blue', label = 'Cluster 2')
plt.scatter(df[y_kmeans == 2, 0], df[y_kmeans == 2, 1], s = 80, c =
'green', label = 'Cluster 3')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], s = 220, c = 'black',alpha=1, label = 'Centroids')
plt.title('Clusters of flowers')

plt.legend()
plt.show()
```



An insight we can get from the scatterplot is the model's accuracy in determining Cluster 2 is comparatively more to Cluster 1 and Cluster 3.

PCA

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
dfs = pca.fit_transform(df)
```

```
explained_variance = pca.explained_variance_ratio_
```

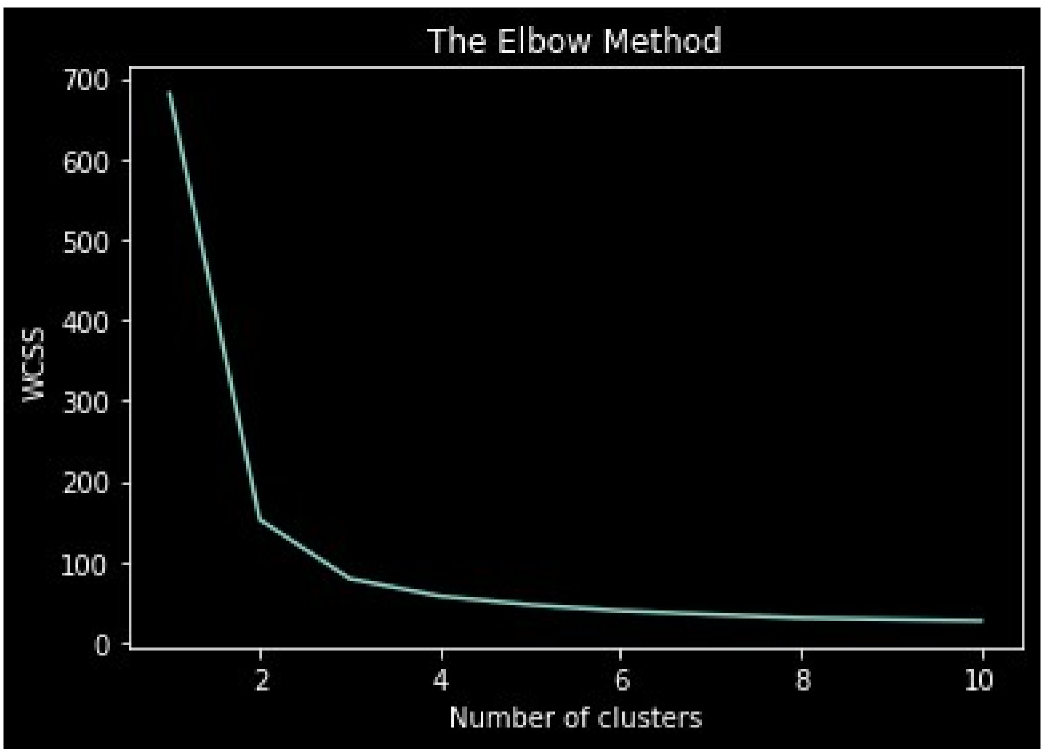
```
explained_variance
```

```
array([0.92461872, 0.05306648])
```

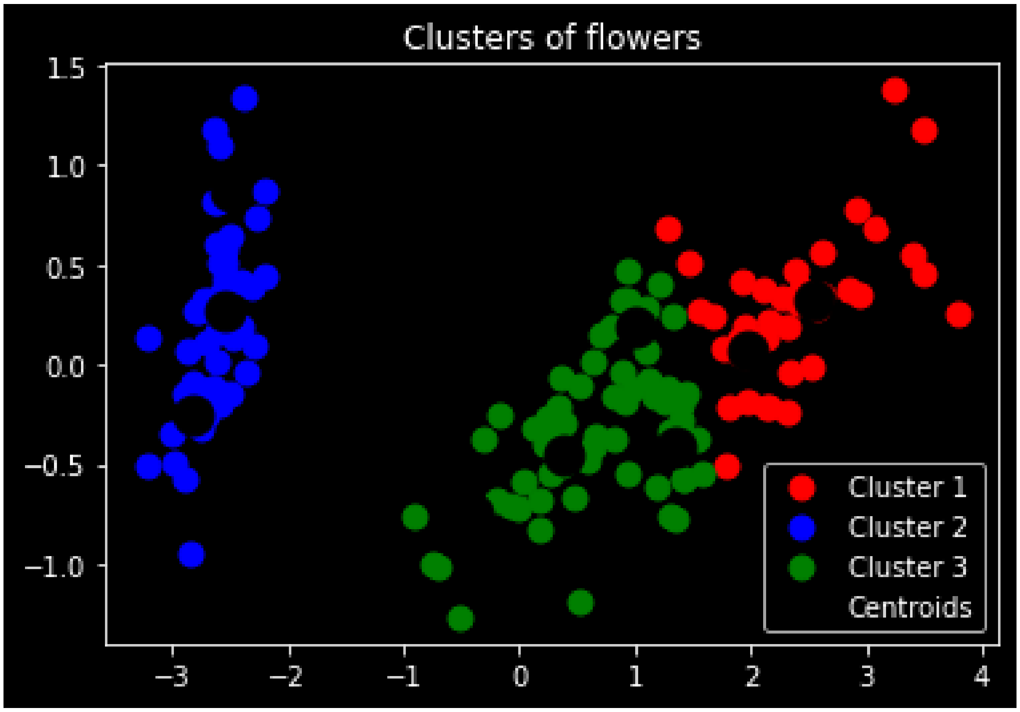
1st and 2nd elements represents variance in 1st and 2nd columns in transformed dataset respectively

K-Means with PCA

```
wcss_p = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state =
42)
    kmeans.fit(dfs)
    wcss_p.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



```
kmeans_p = KMeans(n_clusters = 3, init = 'k-means++', random_state =
42)
y_kmeans_p = kmeans_p.fit_predict(dfs)

y_kmeans_p

array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 0, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 0, 0, 0, 0, 2, 0, 0,
0,
       0, 0, 0, 2, 2, 0, 0, 0, 0, 2, 0, 2, 0, 2, 0, 0, 2, 2, 0, 0, 0,
0,
       0, 2, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 2],
dtype=int32)

# from sklearn.metrics import accuracy_score

print('K-Mean model accuracy with PCA is:
{}'.format(accuracy_score(Y,y_kmeans_p)))

K-Mean model accuracy with PCA is: 0.30666666666666664

plt.scatter(dfs[y_kmeans_p == 0, 0], dfs[y_kmeans_p == 0, 1], s = 80,
c = 'red', label = 'Cluster 1')
plt.scatter(dfs[y_kmeans_p == 1, 0], dfs[y_kmeans_p == 1, 1], s = 80,
c = 'blue', label = 'Cluster 2')
plt.scatter(dfs[y_kmeans_p == 2, 0], dfs[y_kmeans_p == 2, 1], s = 80,
c = 'green', label = 'Cluster 3')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], s = 220, c = 'black',alpha=1, label = 'Centroids')
plt.title('Clusters of flowers')

plt.legend()
plt.show()
```



An insight we can get from the scatterplot is the model's accuracy in determining Cluster 1 is comparatively more to Cluster 2 and Cluster 3.

EM algorithm

```
from sklearn.datasets import load_iris
iris = load_iris()
from sklearn.utils import shuffle
X = pd.DataFrame(iris.data)

Y = pd.DataFrame(iris.target)
X,Y = shuffle(X,Y)
from sklearn.mixture import GaussianMixture

model21 = GaussianMixture(n_components=3,random_state=3425)
model21.fit(X)

uu = model21.predict(X)

print('EM model accuracy is: {}'.format(accuracy_score(Y,uu)))
```

```
EM model accuracy is: 0.3333333333333333
```

EM algorithm with PCA

```
from sklearn.datasets import load_iris
iris = load_iris()
from sklearn.utils import shuffle
```

```
X = pd.DataFrame(iris.data)

Y = pd.DataFrame(iris.target)
X,Y = shuffle(X,Y)

from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_p = pca.fit_transform(X)

from sklearn.mixture import GaussianMixture

model2 = GaussianMixture(n_components=3,random_state=3425)
model2.fit(X_p)

res= model2.predict(X_p)

print('EM model with PCA accuracy is:
{}'.format(accuracy_score(Y,res)))
```

```
EM model with PCA accuracy is: 0.98
```

RESULTS

Accuracy of K-means and EM models

1. **The accuracy of K-Mean model is: 0.24**
2. **The accuracy of EM model is: 0.33**

Accuracy of K-means and EM models on applying PCA

1. **The accuracy of K-Mean model with PCA is: 0.30**
2. **The accuracy of EM model is: 0.98**

Conclusion

It can be seen that the EM method behaves and performs better than the K-means model in both raw data and PCA data (dimensionally reduced data). On semi-supervised learning, the EM Algorithm provides a viable alternative to classic k-means clustering. It finds multivariate Gaussian distributions for each cluster to offer stable solutions.