GRIP : THE SPARKS FOUNDATION

DATA SCIENCE AND BUSINESS ANALYTICS INTERN

AUTHOR : SWETA PATEL

TASK 2 : Prediction using Unsupervised ML

Aim : To predict the optimum number of clusters and represent it visually.

Importing the Data

In [30]:

```
#importing all the libraries
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
```

In [38]:

```
#Reading the data
df = datasets.load_iris()
df = pd.DataFrame(iris.data, columns = iris.feature_names)
df.head()
```

Out[38]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [43]:

```
df.info
```

Out[43]:
```
<bound method DataFrame.info of      sepal length (cm)  sepal width (cm)  p
etal length (cm)  petal width (cm)
0                  5.1               3.5               1.4               0
.2
1                  4.9               3.0               1.4               0
.2
2                  4.7               3.2               1.3               0
.2
3                  4.6               3.1               1.5               0
.2
4                  5.0               3.6               1.4               0
.2
..                 ...               ...               ...               .
..
145                6.7               3.0               5.2               2
.3
146                6.3               2.5               5.0               1
.9
147                6.5               3.0               5.2               2
.0
148                6.2               3.4               5.4               2
.3
149                5.9               3.0               5.1               1
.8

[150 rows x 4 columns]>
```

In [57]:
```
df.shape
```

Out[57]:
```
(149, 4)
```

In [58]:
```
df.columns
```

Out[58]:
```
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)'],
      dtype='object')
```

In [62]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 149 entries, 0 to 149
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  149 non-null    float64
 1   sepal width (cm)   149 non-null    float64
 2   petal length (cm)  149 non-null    float64
 3   petal width (cm)   149 non-null    float64
dtypes: float64(4)
memory usage: 5.8 KB
```

In [63]:
```
df.describe()
```

Out[63]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	149.000000	149.000000	149.000000	149.000000
mean	5.843624	3.059732	3.748993	1.194631
std	0.830851	0.436342	1.767791	0.762622
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.300000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [64]:

```
df.isnull().sum()
```

Out[64]:

```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

In [65]:

```
df.drop_duplicates(inplace=True)
```

Data Visualisation

In [52]:

```
X = df.iloc[:, [0,1,2, 3]].values
```

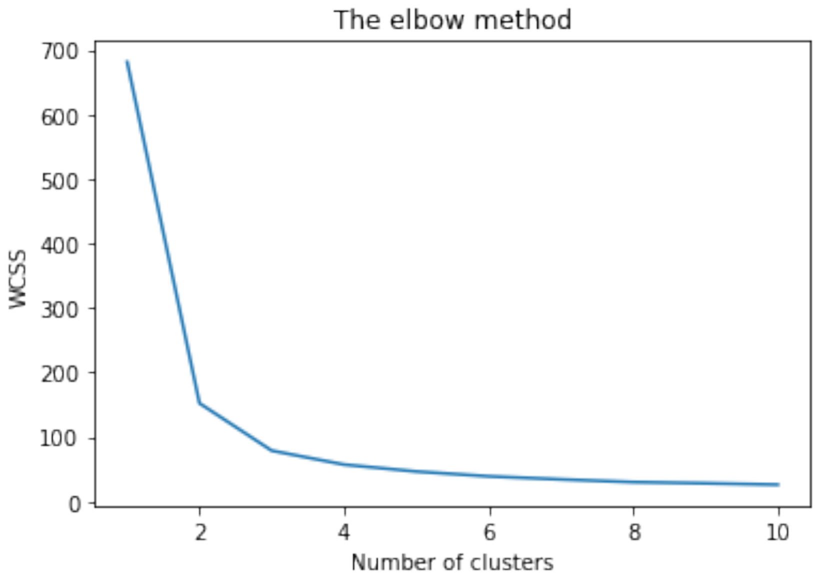
In [82]:

```
from sklearn.cluster import KMeans
wcss = []

for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
                    max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
```

In [80]:

```
plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') # Within cluster sum of squares
plt.show()
```



To determine the optimal number of clusters, the value of k at the elbow is selected i.e. the point after which the distortion start decreasing in a linear fashion. Thus form the above graph, it can be said that the optimum clusters the data is 3.

In [84]:

```
# K-Means to the dataset
kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 0)
```

The experimental results show the robustness of the Y-means algorithm as well as its good performance against a set of other well known unsupervised clustering techniques.

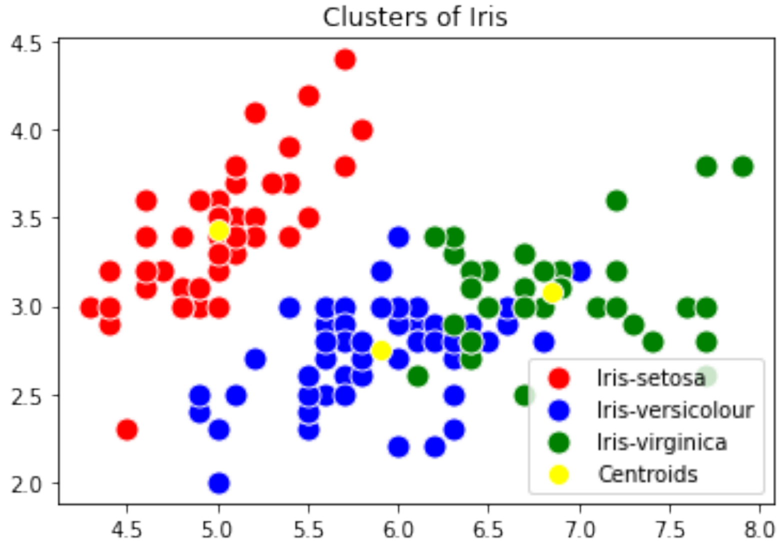
In [85]:

```
y_kmeans = kmeans.fit_predict(X)
```

In [88]:

```
#visualising the data
import seaborn as sns
#plt.figure(figsize=(9,5))
sns.scatterplot(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], color = 'red', la
sns.scatterplot(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], color = 'blue', l
sns.scatterplot(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], color = 'green',

## Plotting the centroids of the clusters
sns.scatterplot(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1
                label = 'Centroids',s=100)
plt.grid(False)
plt.title('Clusters of Iris')
plt.legend()
plt.show()
```


Clusters of Iris
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Iris-setosa
Iris-versicolour
Iris-virginica
Centroids