GRIP@The Sparks Foundation

Name : Jagadish Rathod

Task 1 : Prediction using Unsupervised Machine Learning

Dataset : https://bit.ly/3kXTdox (https://bit.ly/3kXTdox)

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading tha data

In [2]:
```
data= pd.read_csv("Iris.csv",error_bad_lines=False,encoding='latin-1')
```

In [3]:
```
data.head(10)
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

In [4]:
```
data.shape
```

Out[4]:
```
(150, 6)
```

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [6]:
```
data.describe()
```

Out[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [7]:
```
X= data.iloc[:,1:5].values
```

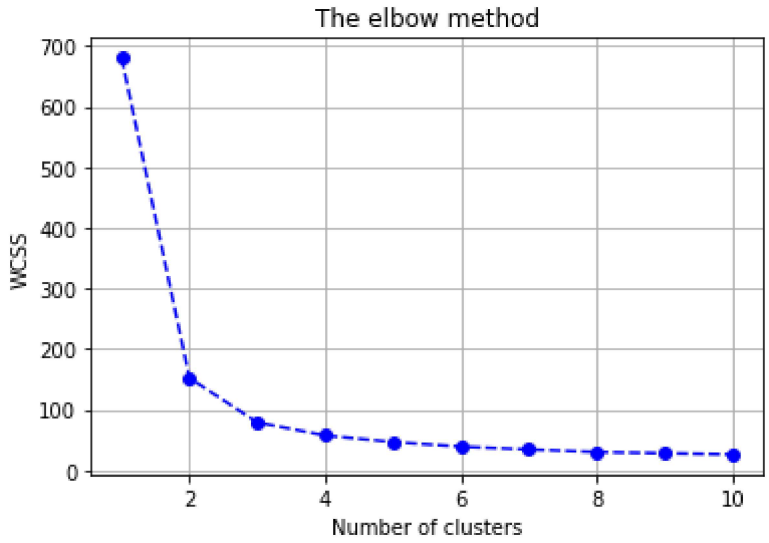
Finding the optimum value of clusters

In [8]:
```
from sklearn.cluster import KMeans
```

In [9]:
```
wcss= []
for i in range(1,11):
    km= KMeans(n_clusters= i)
    km.fit(X)
    wcss.append(km.inertia_)
```

Plotting the elbow method graph

In [10]:
```
plt.plot(range(1, 11), wcss, 'go--', color='blue')
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.grid()
plt.show()
```



The required value of the number of the clusters from the above graph is 3(because at from 3 onwards the graph becomes almost constant)

Applying KMeans Classifier

In [11]:
```
# Applying kmeans to the dataset / Creating the kmeans classifier
kmn = KMeans(n_clusters = 3, init = 'k-means++',
                 max_iter = 300, n_init = 10, random_state = 0)
y_pred = kmn.fit_predict(X)
```

Plotting the Clusters graph/ Visualizing the Clusters

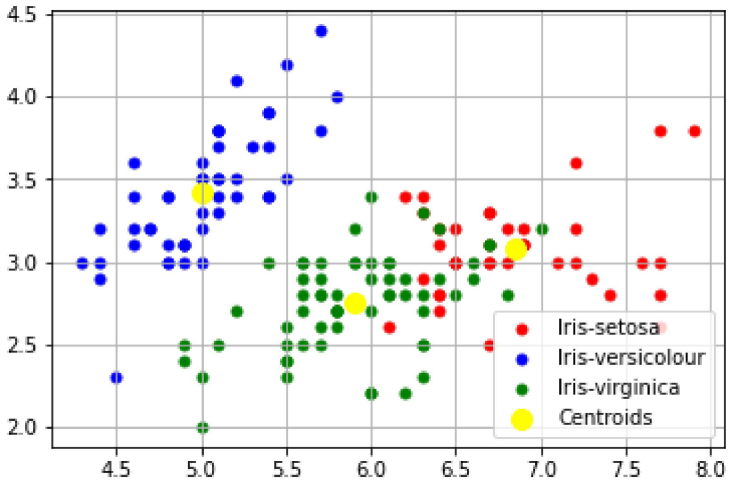
In [12]:
```
# Visualising the clusters - On the first two columns
plt.scatter(X[y_pred == 0, 0], X[y_pred == 0, 1], s = 25, c = 'red', label =
'Iris-setosa')

plt.scatter(X[y_pred == 1, 0], X[y_pred == 1, 1], s = 25, c = 'blue', label =
'Iris-versicolour')

plt.scatter(X[y_pred == 2, 0], X[y_pred == 2, 1], s = 25, c = 'green', label =
"Iris-virginica")

# Plotting the centroids of the clusters
plt.scatter(kmn.cluster_centers_[:, 0], kmn.cluster_centers_[:,1], s = 100, c
= 'yellow', label = 'Centroids')


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



Thank You!!

In [ ]:
```

```