GRIP:The Spark Foundation

Data Science and Bussiness Analytics Intern

Author: Shalaka Patil

Task 2 :Prediction Using Unsupervised ML

Importing the Data

In this step we will import the required libraries & data set with the help of pandas library.

In [1]:
```
# Importing the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
# To ignore the warning
import warnings as wg
wg.filterwarnings("ignore")
```

In [2]:
```
# Reading data Iris Dataset
df = pd.read_csv("C:/Users/DELL/OneDrive/Desktop/Iris Data.csv")
print("Impoprting Data Sucessfully")
```

```
Impoprting Data Sucessfully
```

In [3]:
```
df.head()
```

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

Visualization the Data In this step we will try to visualize our dataset.

In [4]:
```
df.tail()
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [5]:
```
df.shape
```

Out[5]:
```
(150, 6)
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [7]:
```
df['Species'].unique()
```

Out[7]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [8]:
```
df.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 [9]:
```
df.describe()
```

Out[9]:

	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

Now,we will drop the label column because it is an unsupervised learning problem.

In [10]:
```
iris = pd.DataFrame(df)
iris_df = iris.drop(columns= ['Species' ,'Id'] )
iris_df.head( )
```

Out[10]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
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

Finding the optimum no. of clusters

```
Before clustering the data using K-Means .We need to specify the number of clusters.In order to find the optimum number of clusters.There are various methods available like Elbow method.Here, the elbow method
is used.
```

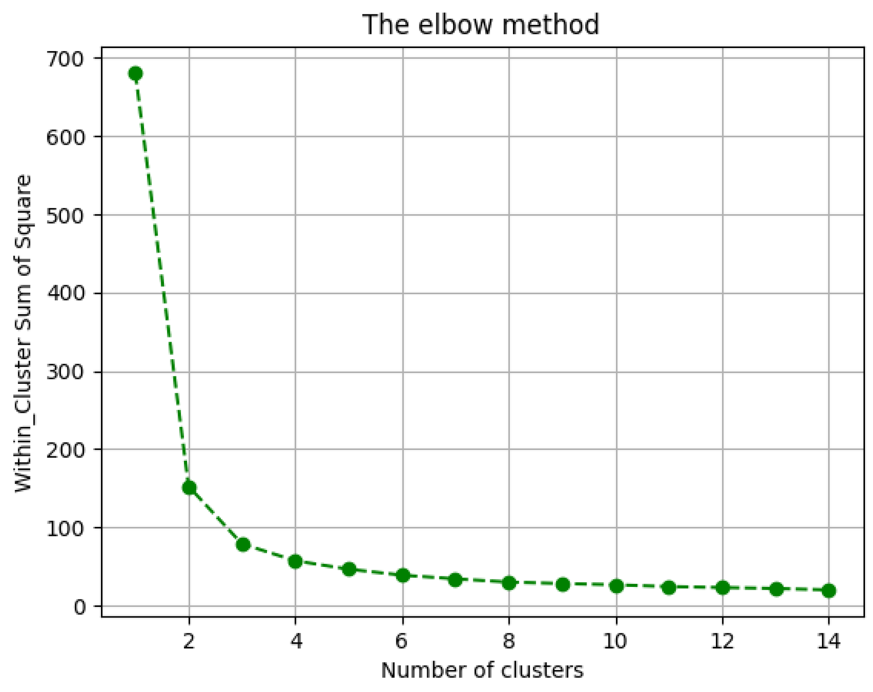
Brief about the Elbow Method

```
In this method,the number of clusters are various withine a certain range.For each number of within cluster sum of square value is calculated and stored in the list.These values are then plotted against the
the range of number of cluster used before the location of bend in the second plot indicates the appropriate number of clusters.
```

In [11]:
```
#Calculating the within cluster sum of square
Within_cluster_sum_of_square=[]
clusters_range=range(1,15)
for k in clusters_range:
    km=KMeans(n_clusters=k)
    km=km.fit(iris_df)
    Within_cluster_sum_of_square.append(km.inertia_)
```

Plotting the Within_cluster_sum of Square against clusters range

In [12]:
```
plt.plot(clusters_range,Within_cluster_sum_of_square,'go--',color='green')
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('Within_Cluster Sum of Square')
plt.grid()
plt.show()
```



We can clearly see why it is called "The elbow method" from the above graph, the optimum clusters is where the elbow occurs.This is when the within clusters sum of square dpoesn't decreases significantly with every iteration. From this we choose the number of clusters as **3**

Applying K Means clustering on the data

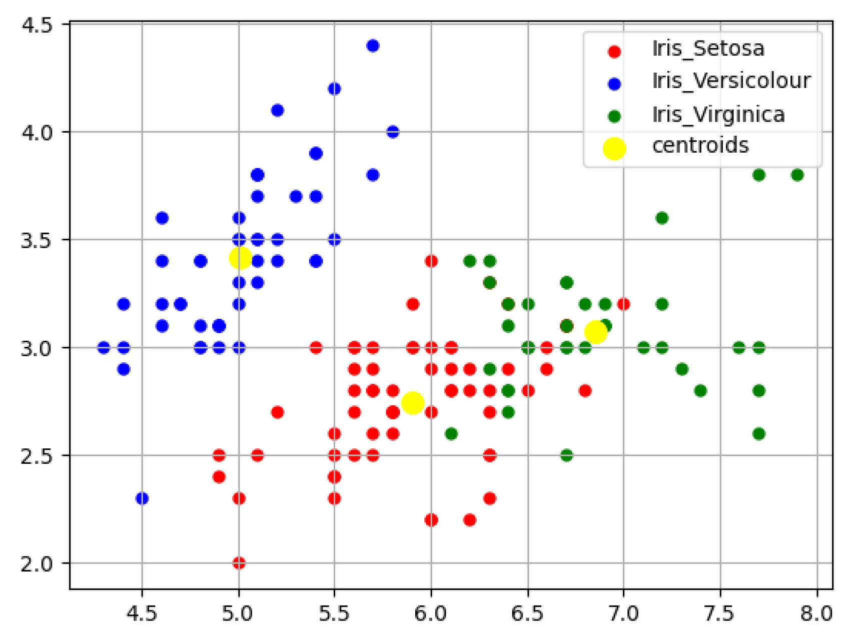
In [13]:
```
from sklearn.cluster import KMeans

model = KMeans(n_clusters=3, init = 'k-means++', max_iter=300, n_init = 10, random_state = 0)
predictions = model.fit_predict(iris_df)
```

Visualising the clusters

In [14]:
```
x=iris_df.iloc[:,[0,1,2,3]].values
plt.scatter(x[predictions == 0, 0], x[predictions == 0, 1],s=25,c='red',label='Iris_Setosa')
plt.scatter(x[predictions == 1, 0], x[predictions == 1, 1],s=25,c='Blue',label='Iris_Versicolour')
plt.scatter(x[predictions == 2, 0], x[predictions == 2, 1],s=25,c='green',label='Iris_Virginica')

# Plotting the cluster centers
plt.scatter(model.cluster_centers_[:,0],model.cluster_centers_[:,1],s=100,c='yellow',label='centroids')
plt.legend()
plt.grid()
plt.show()
```



Iris_Setosa
Iris_Versicolour
Iris_Virginica
centroids
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