Task2

April 14, 2021

1 The Sparks Foundation

1.1 Graduate Rotational Internship Program (GRIP)

1.2 Task 2 - Prediction using Unsupervised ML

1.3 Problem Statement - From the given 'Iris' dataset, predict the optimum number of clusters and represent it visually

1.4 Programming Language - Python

1.5 IDE - Jupyter Notebook

1.6 AUTHOR - Kalyankar Sreeja

1.7 STEP 1 - Import Libraries

[19]:
```
#Import all the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

1.8 STEP 2 - Import Dataset

[3]:
```
#Displaying the whole dataset
df = pd.read_csv('iris.csv')
df
```

[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	\
0	1	5.1	3.5	1.4	0.2	
1	2	4.9	3.0	1.4	0.2	
2	3	4.7	3.2	1.3	0.2	
3	4	4.6	3.1	1.5	0.2	
4	5	5.0	3.6	1.4	0.2	
..	…	…	…	…	…	
145	146	6.7	3.0	5.2	2.3	
146	147	6.3	2.5	5.0	1.9	
147	148	6.5	3.0	5.2	2.0	

```
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              …
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

[9]:
```
#Viewing a series of numeric values
df.describe()
```

[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

[10]:
```
#Printing a concise summary of a DataFrame
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
```

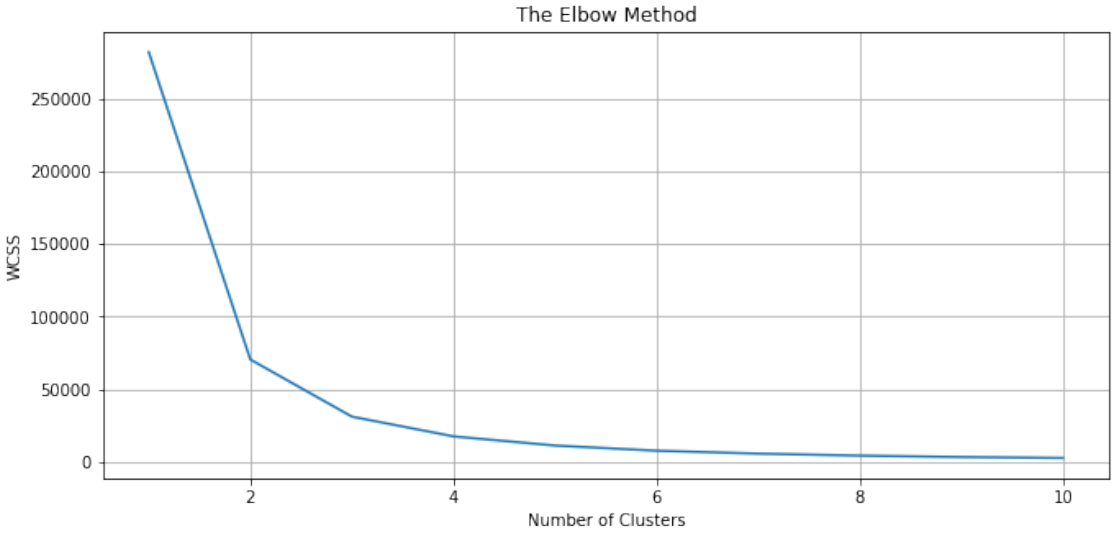
1.9 STEP 3 - Find the optimum number of clusters using "Elbow Method"

[18]:
```
#Finding the optimum number of clusters for k-means classification and also␣
 ↪showing how to determine the value of K
x = df.iloc[:, [0, 1, 2, 3]].values

from sklearn.cluster import KMeans
wcss = []   #wcss stands for within-cluster sum of squares

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_)

#Plotting the results onto a line graph,
#Allows us to observe "The Elbow"
plt.figure(figsize = [11, 5])
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()
```



1.9.1 The optimum clusters are the place the elbow happens. This is the point at which the WCSS doesn't reduce essentially with each iteration. From this, we choose the number of clusters as "3".

1.10 STEP 4 - Train KMeans Model

[20]:
```
#Creating the kmeans classifier
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init =␣
↪10, random_state = 0)
y_kmeans = kmeans.fit_predict(x)
print(y_kmeans)
```

```
[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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 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
 1 1]
```

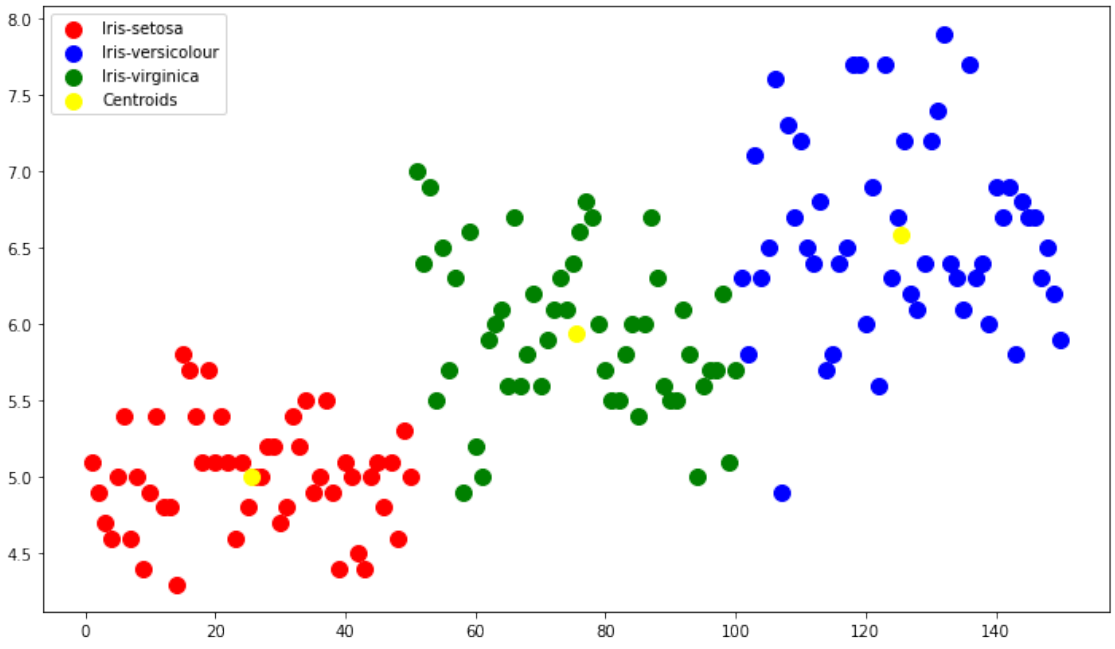
1.11 STEP 5 - Visualize Model

[11]:
```
plt.figure(figsize = [12, 7])

plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, color = 'red',␣
↪label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, color = 'blue',␣
↪label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, color = 'green',␣
↪label = 'Iris-virginica')

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

[11]:
```
<matplotlib.legend.Legend at 0x23eee81aeb8>
```



1.11.1 The plotted centroids of the cluster in the graph shows the predicted optimum number of clusters from the iris dataset

1.12 CONCLUSION - The optimum number of clusters in the iris dataset are "3"