

Task: From the given 'Iris' dataset, predict the optimum number of clusters and represent it visually.

Libraries/Datasets Used: Scikit Learn, Pandas, Numpy, Iris Dataset

By Gayathri R

```
In [1]: # Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import datasets
```

```
In [2]: # Load the iris dataset
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)
iris_df.head() # See the first 5 rows
```

Out[2]:

	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 [3]: iris_df.shape
```

Out[3]: (150, 4)

```
In [4]: iris_df.describe()
```

Out[4]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
sepal length (cm)    150 non-null float64
sepal width (cm)     150 non-null float64
petal length (cm)    150 non-null float64
petal width (cm)     150 non-null float64
dtypes: float64(4)
memory usage: 4.8 KB
```

In [6]: `iris_df.isnull().sum()`

Out[6]:

sepal length (cm)	0
sepal width (cm)	0
petal length (cm)	0
petal width (cm)	0

dtype: int64

FINDING NUMBER OF CLUSTERS

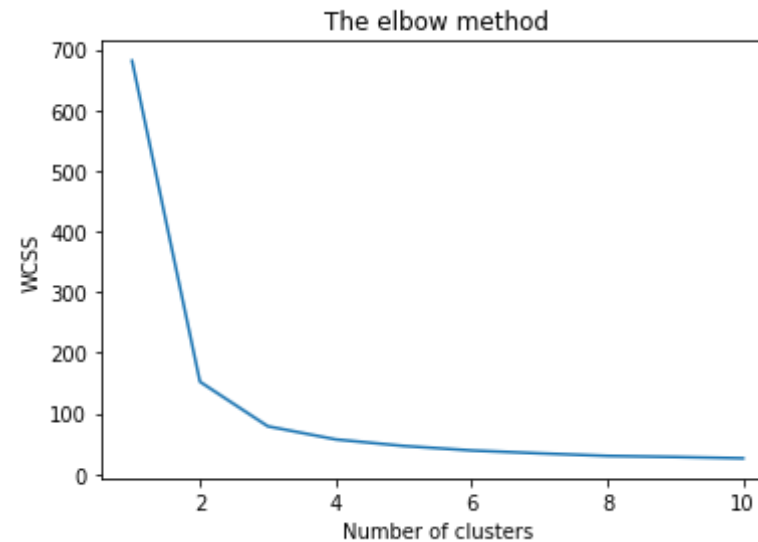
```
In [7]: # Finding the optimum number of clusters for k-means classification
x = iris_df.iloc[:, [0, 1, 2, 3]].values

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_)
```

Plotting the results onto a line graph,

`allowing us to observe 'The elbow'

```
In [8]: 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()
```



From the above graph, we can observed that before the value $k=3$, the SSE decreases and the curve is almost parallel to x-axis grater than $k=3$.

From this we choose the number of clusters as '3'.

MODEL BUILDING, TRAINING & PREDICTION

```
In [9]: # Applying kmeans to the dataset
# Creating the kmeans classifier

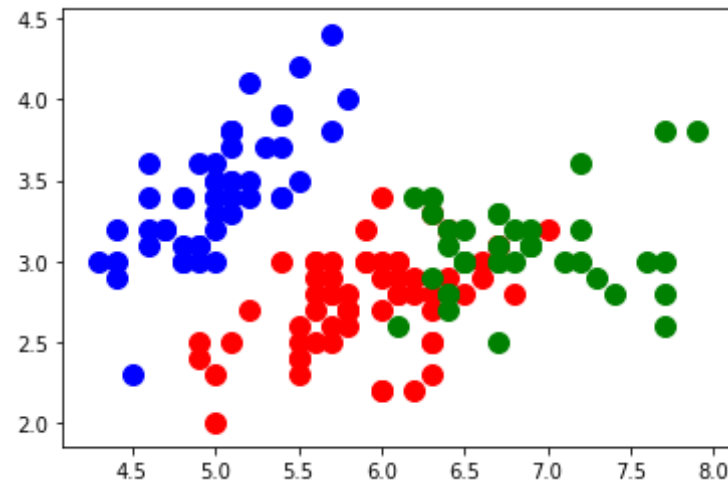
kmeans = KMeans(n_clusters = 3, init = 'k-means++',
                max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(x)
```

Visualizing the cluster data

```
In [10]: # Visualising the clusters
```

```
# Preferably on the first two columns
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1],
            s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1],
            s = 100, c = 'blue', label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1],
            s = 100, c = 'green', label = 'Iris-virginica')
```

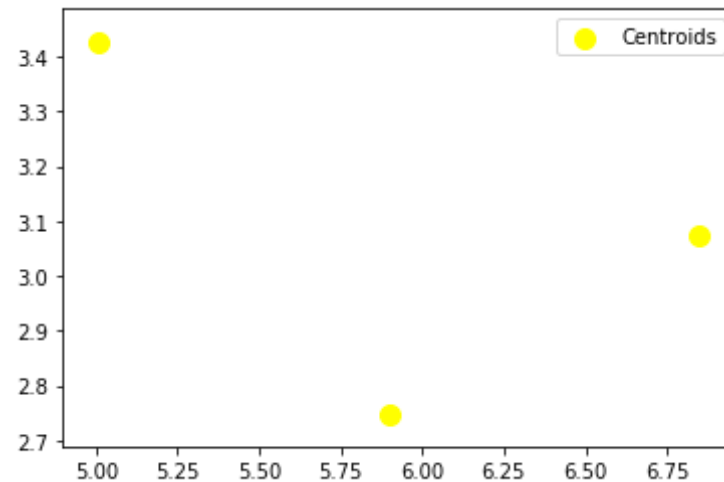
Out[10]: <matplotlib.collections.PathCollection at 0x164a8807808>



```
In [11]: # Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers[:, 0], kmeans.cluster_centers[:, 1],
            s = 100, c = 'yellow', label = 'Centroids')

plt.legend()
```

Out[11]: <matplotlib.legend.Legend at 0x164a887eb08>



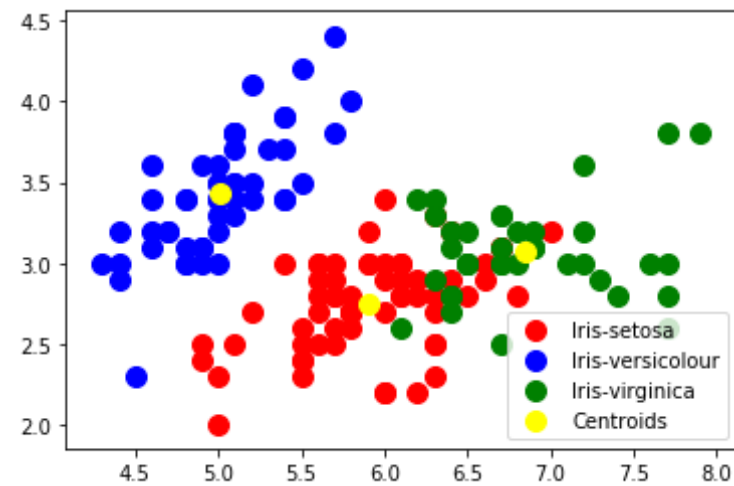
Now Combining both the above graphs together

```
In [12]: # Visualising the clusters
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1],
            s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1],
            s = 100, c = 'blue', label = 'Iris-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1],
            s = 100, c = 'green', label = 'Iris-virginica')

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

plt.legend()
```

Out[12]: <matplotlib.legend.Legend at 0x164a890be88>



Result: Optimum value of number of clusters is k=3.

In []: