THE SPARKS FOUNDATION INTERNSHIP

SWATI SHAH

Task_2

PREDICATION USING UNSUPERVISED MACHINE LEARNING

GRIP2021

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

In [10]:
```
# Importing the required libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import datasets
import seaborn as sns
```

In [11]:
```
# Loading the iris dataset
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)
iris_df.head()
```

Out[11]:

	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 [12]:
```
iris_df.shape
```

Out[12]:
```
(150, 4)
```

In [13]:
```
iris_df.info()
```

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

In [14]:
```
iris_df.describe()
```

Out[14]:

	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

First we need to find the optimum number of clusters for K-Means. Here we will use The Elbow Method to determine the value of k in K-Means.

The Elbow Method

In Elbow method we calculate the Within-Cluster-Sum of Squared Errors (WCSS) for different values of k, and choose the k for which WCSS becomes first starts to diminish. In the plot of WCSS-versus-k, this is visible as an elbow

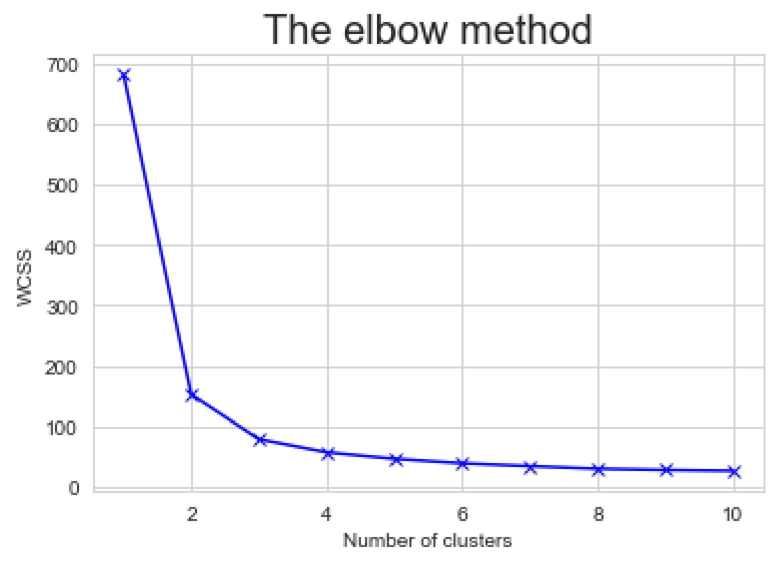
In [15]:
```
x = iris_df.iloc[:, :4].values
from sklearn.cluster import KMeans

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_)

pd.DataFrame({"Number of Clusters":range(1,11),"WCSS":wcss})
```

Out[15]:

	Number of Clusters	WCSS
0	1	681.370600
1	2	152.347952
2	3	78.851441
3	4	57.256009
4	5	46.446182
5	6	39.039987
6	7	34.299712
7	8	30.014398
8	9	28.036906
9	10	26.534529

Plotting the graph onto a line graph to observe the pattern

In [16]:
```
#elbow method
sns.set_style("whitegrid")
plt.plot(range(1, 11), wcss, 'bx-')
plt.title('The elbow method', size= 20)
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #within cluster sum of squares
plt.show()
```



"The elbow method" got its name from the elbow pattern forming something like above. The optimal clusters are formed where the elbow occurs. This is when the WCSS(Within Cluster Sum of Squares) doesn't decrease with every iteration significantly.

Here we choose the number of clusters as '3'.

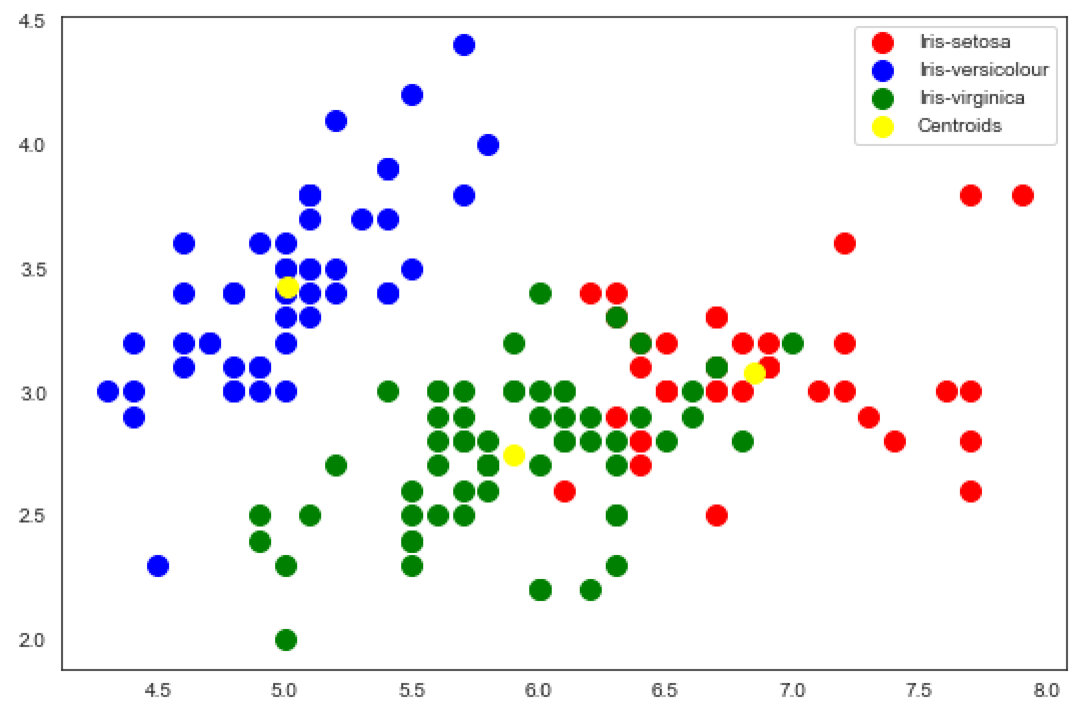
Creating K-Means Classifier

In [17]:
```
# 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 [18]:
```
# Visualising the clusters on the first two columns
sns.set_style('white')
plt.figure(figsize=[9,6])
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')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1],
            s = 100, c = 'yellow', label = 'Centroids')

plt.legend()
```

Out[18]:
```
<matplotlib.legend.Legend at 0x26be915dc40>
```



The yellow points are the centroids, we can identify the center points of the data by using following code:

In [19]:
```
centers = kmeans.cluster_centers_
print(centers)
```

```
[[6.85       3.07368421 5.74210526 2.07105263]
 [5.006      3.428      1.462      0.246     ]
 [5.9016129  2.7483871  4.39354839 1.43387097]]
```

Therefore, the optimum number of clusters is predicted and represented visually.