**Workshop-1: K-Means Clustering**

This notebook will walk through some of the basics of K-Means Clustering.

**Author: Xaltius Pte.Ltd.**

Copyright Xaltius Pte.Ltd., 2018

Not for distribution.

In [1]:



*# Importing the libraries*

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** pandas **as** pd

**from** sklearn **import** datasets

​

*# 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[1]:

|  | **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 |

How do we find the optimum number of clusters for K-Means? How does one determine the value of K?

In [2]:



*# 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'*

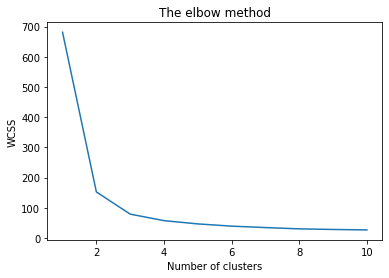
plt.plot(range(1, 11), wcss)

plt.title('The elbow method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS') *# Within cluster sum of squares*

plt.show()



You 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 cluster sum of squares (WCSS) doesn't decrease significantly with every iteration.

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

In [3]:



*# 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)

In [4]:



*# Visualising the clusters - 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')

​

*# Plotting the centroids of the clusters*

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:,1],

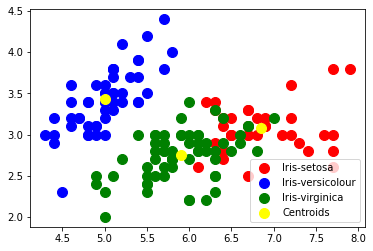
s **=** 100, c **=** 'yellow', label **=** 'Centroids')

​

plt.legend()

Out[4]:

<matplotlib.legend.Legend at 0x1e87edbbb20>



In [ ]:



​