# THE SPARKS FOUNDATION - GRIP(OCTOBER'21)

## Data Science and Business Analytics Internship

### Name - Shrutika Pramanik

#### Task 2 - Prediction Using Unsupervised ML

***Objective - In this task, we need to predict the optimum number of clusters from the given 'Iris' dataset and represent it visually.***

#### K-Means Clustering

**In this task, we will predict the optimum number of clusters from the given 'Iris' dataset. This notebook will walk through some of the basics of K-Means Clustering.**

## IMPORT THE REQUIRED LIBRARIES

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import datasets
import warnings
warnings.filterwarnings('ignore') #Ignore warnings
```

## LOAD THE 'IRIS' DATASET

In [2]:

```
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)
```

### The first 5 rows, last 5 rows, shape, description of the data are displayed:

In [3]:

```
iris_df.head() #gives the first 5 rows
```

Out[3]:

| | 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 [4]:

```
iris_df.tail() #gives the last 5 rows
```

Out[4]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| **145** | 6.7 | 3.0 | 5.2 | 2.3 |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 |

In [6]:

```
iris_df.shape #gives the shape of the data
```

Out[6]:

```
(150, 4)
```

In [7]:

```
iris_df.describe() #gives description of the data
```

Out[7]:

| | 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 [8]:

```
iris_df.info() #gives information about the data
```

```
<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
```

## Checking if there is null element in the dataset

In [9]:

```
print(iris_df.isnull().sum())
```

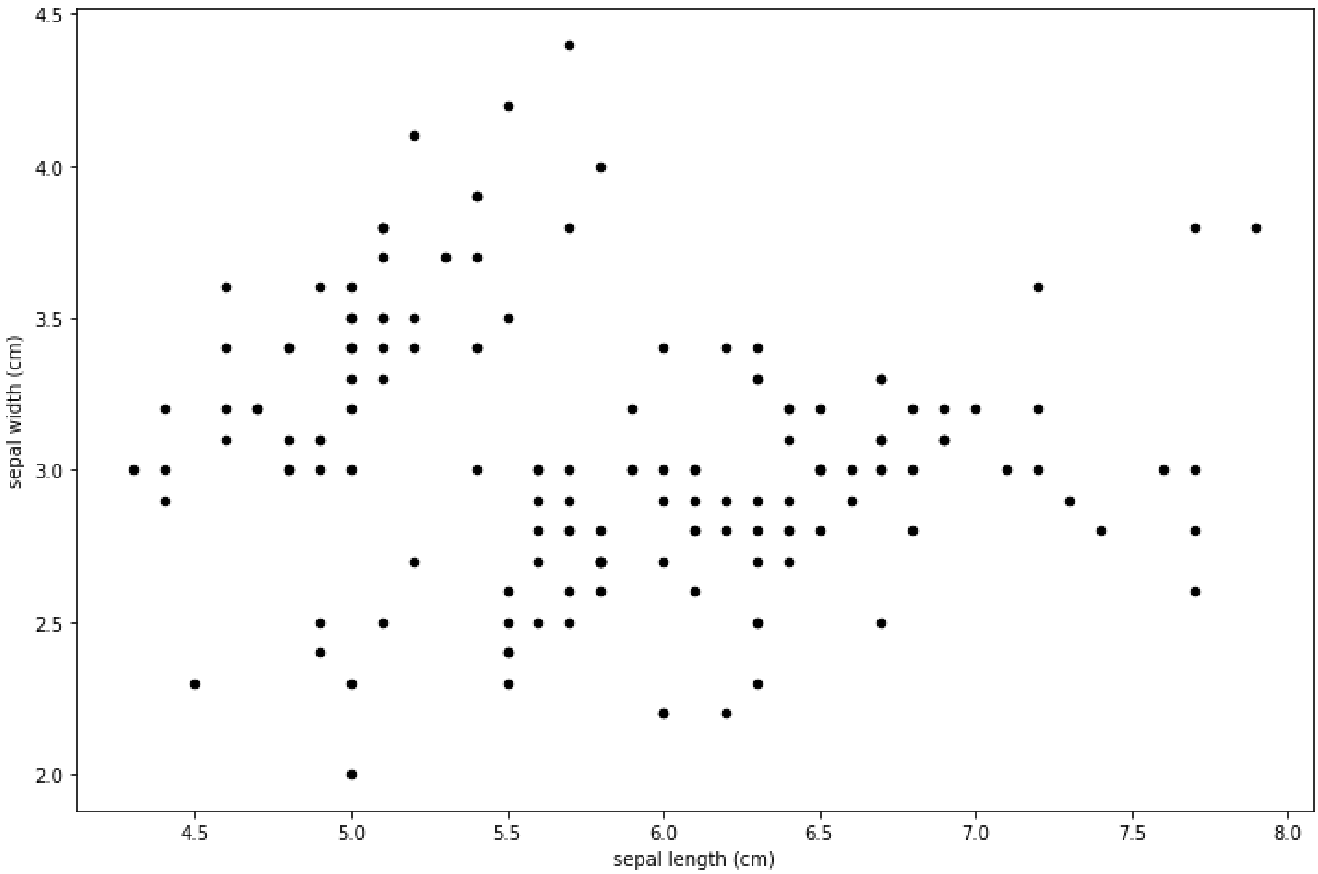
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

***Hence, there is no null element in the given 'Iris' dataset.***

# DATA VISUALIZATION

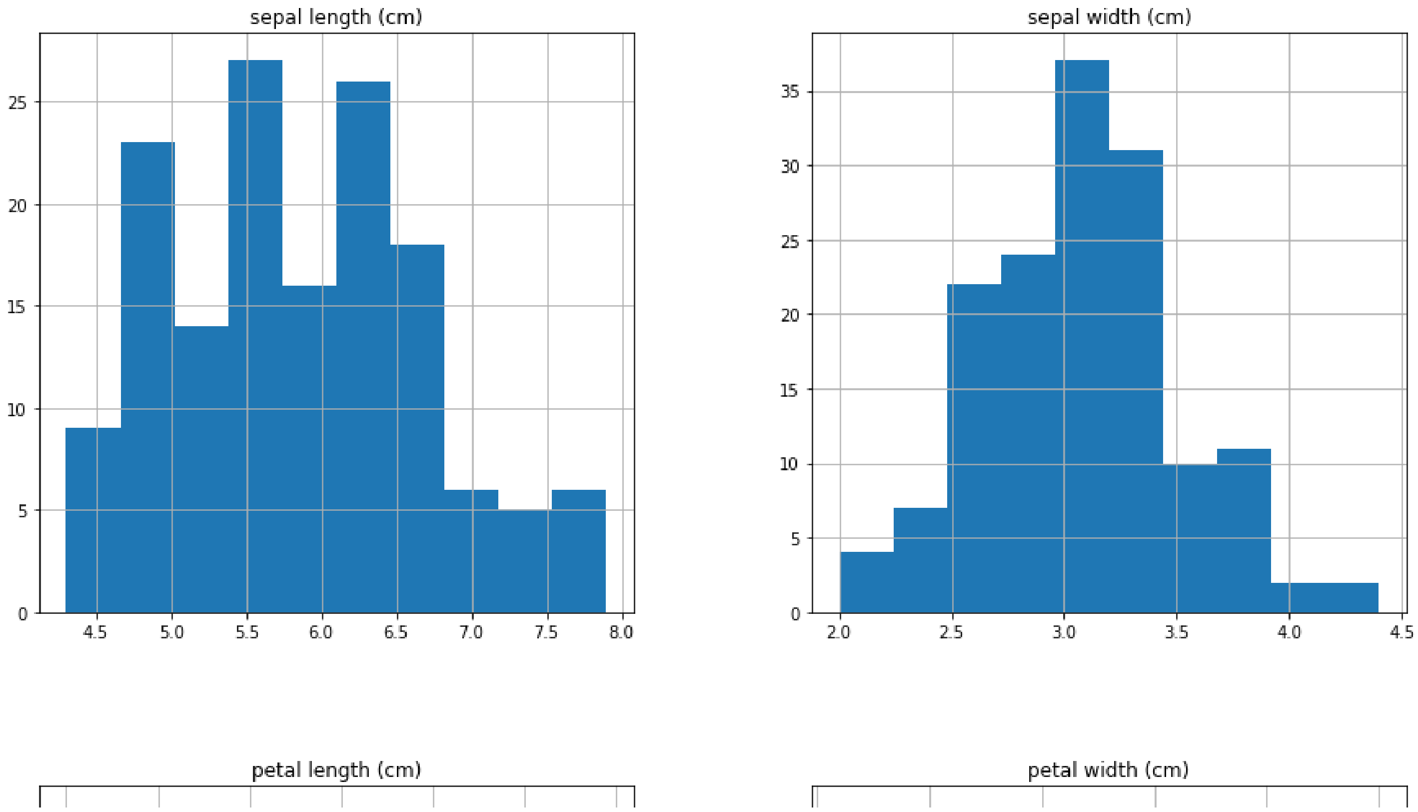
In [10]:

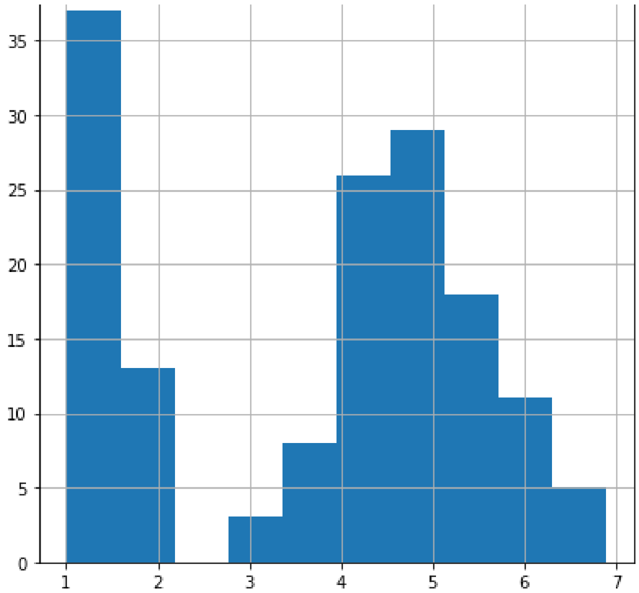
```
# scatter plot

iris_df.plot(kind = "scatter", x = "sepal length (cm)", y = "sepal width (cm)", figsize
= (12,8), color = 'black')
plt.show()
```

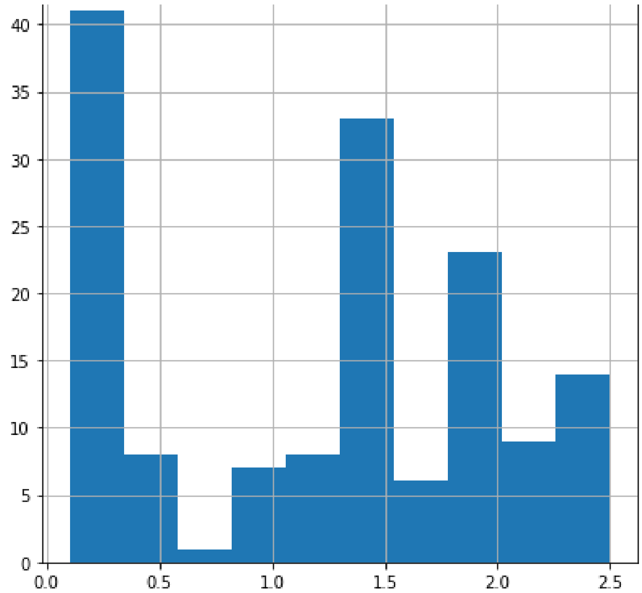



In [11]:

```
p = iris_df.hist(figsize = (15,15)) #histograms
```







***The above scatter plot and histograms give a rough representation of the given 'Iris' dataset.***

## CORRELATION OF THE DATA

In [12]:

```
iris_df.corr()
```

Out[12]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) |
|---|---|---|---|---|
| **sepal length (cm)** | 1.000000 | -0.117570 | 0.871754 | 0.817941 |
| **sepal width (cm)** | -0.117570 | 1.000000 | -0.428440 | -0.366126 |
| **petal length (cm)** | 0.871754 | -0.428440 | 1.000000 | 0.962865 |
| **petal width (cm)** | 0.817941 | -0.366126 | 0.962865 | 1.000000 |

## K- MEANS

**K-Means is a centroid-based algorithm, or a distance-based algorithm, where we calculate the distances to assign a point to a cluster. In K-Means, each cluster is associated with a centroid.**

#### How do you find the optimum number of clusters for K- Means? How does one determines the value of K?

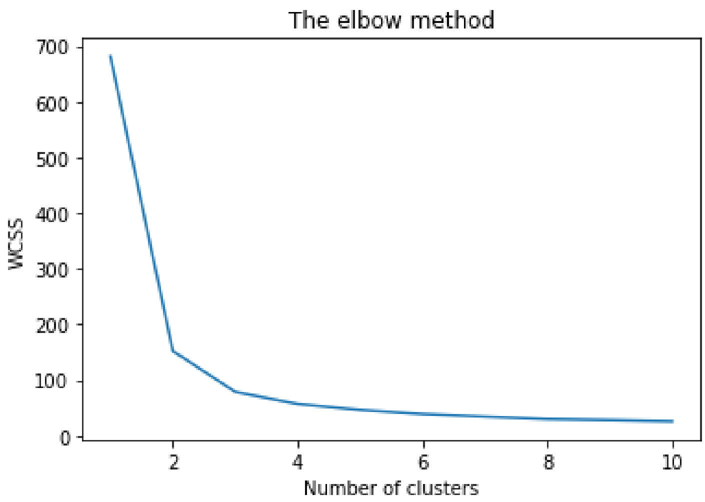
In [13]:

```
# 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, ran
dom_state = 0)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
```

In [14]:

```
# 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()
```




**We can clearly see why it is called 'The elbow method' from the above graph, the optimum clusters are 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'.**

## IMPLEMENTING THE K-MEANS CLUSTERING

In [15]:

```
# Applying k-means to the dataset/ Creating the k-means 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 [17]:

```
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)
```

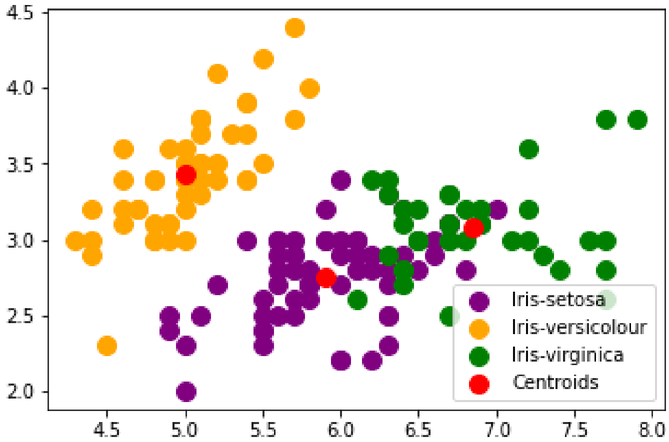
```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
```

In [18]:

```
# Visualising the clusters

plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c = 'purple', label = 'Ir
is-setosa')
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c = 'orange', label = 'Ir
is-versicolour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c = 'green', label = 'Iri
s-virginica')

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

plt.legend()
```

Out[18]:

```
<matplotlib.legend.Legend at 0x1fd39d89af0>
```




***Thus, the K-Means Workshop is concluded.***

# THANKING YOU.