THE SPARKS FOUNDATION

(DATA SCIENCE AND BUSSINESS ANALYTICS)

GRIPSEPT-2021

TASK-1

AUTHOR:SUBALAKSHMI P

OBJECTIVE:PREDICT THE OPTIMUM NUMBER OF CLUSTERS & REPRESENT IT VISUALLY

IMPORTING THE LIBRARIES

In [1]:
```
#To import and analyze data
import pandas as pd
#To perform a ndimensional operation
import numpy as np
#To perform graphical plot into context
import matplotlib.pyplot as plt
#To visualize the data
import seaborn as sns
#To load the iris dataset
from sklearn import datasets
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data,columns = iris.feature_names)
print('Dataset imported effectively')
```

```
Dataset imported effectively
```

To See the first five rows:

In [2]:
```
iris_df.head()
```

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

To see how many rows and columns does the data have:

In [3]:
```
iris_df.describe
```

Out[3]:
```
<bound method NDFrame.describe of     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
..                ...               ...                ...               ...
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

[150 rows x 4 columns]>
```

Missing values

In [4]:
```
iris_df.isnull().sum()
```

Out[4]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

Hence there are no missing values are present in the data set

Statistical data

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

Out[5]:

	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

Summary of DataFrame

In [6]:
```
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
```

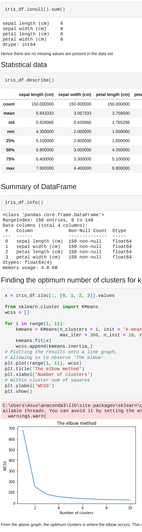
Finding the optimum number of clusters for k-means classification

In [7]:
```
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.xlabel('WCSS')
plt.show()
```

```
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



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. THUS THE OPTIMUM NUMBER OF CLUSTERS IS 3.

Training the model

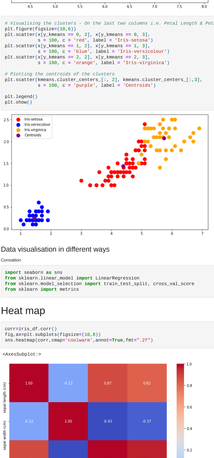
In [8]:
```
# Applying kmeans to the dataset / Creating the kmeans classifier
kmeans = KMeans(n_clusters = 3, init = 'k-means++',
                max_iter = 300, n_init = 10, random_state = 0)
y_kmeans = kmeans.fit_predict(x)
```

In [9]:
```
# Visualising the clusters - On the first two columns
plt.figure(figsize=(10,6))
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-versiculour')
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1],
            s = 100, c = 'orange', label = 'Iris-virginica')

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

plt.legend()
```

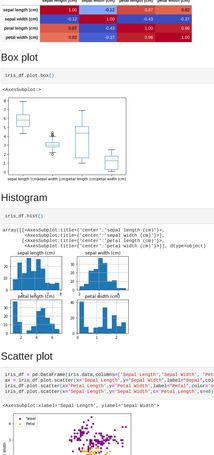
Out[9]:
```
<matplotlib.legend.Legend at 0x278a53bdd00>
```



In [10]:
```
# Visualising the clusters - On the last two columns i.e. Petal Length & Petal Width
plt.figure(figsize=(10,6))
plt.scatter(x[y_kmeans == 0, 2], x[y_kmeans == 0, 3],
            s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(x[y_kmeans == 1, 2], x[y_kmeans == 1, 3],
            s = 100, c = 'blue', label = 'Iris-versiculour')
plt.scatter(x[y_kmeans == 2, 2], x[y_kmeans == 2, 3],
            s = 100, c = 'orange', label = 'Iris-virginica')

# Plotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:, 2], kmeans.cluster_centers_[:,3],
            s = 100, c = 'purple', label = 'Centroids')

plt.legend()
plt.show()
```



Data visualisation in different ways

Correaltion

In [11]:
```
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn import metrics
```

Heat map

In [12]:
```
corr=iris_df.corr()
fig,ax=plt.subplots(figsize=(10,8))
sns.heatmap(corr,cmap='coolwarm',annot=True,fmt=".2f")
```

Out[12]:
```
<AxesSubplot:>
```


In [13]:
```
corr=iris_df.corr()
corr.style.background_gradient(cmap='coolwarm').set_precision(2)
```

Out[13]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
sepal length (cm)	1.00	-0.12	0.87	0.82
sepal width (cm)	-0.12	1.00	-0.43	-0.37
petal length (cm)	0.87	-0.43	1.00	0.96
petal width (cm)	0.82	-0.37	0.96	1.00

Box plot

In [14]:
```
iris_df.plot.box()
```

Out[14]:
```
<AxesSubplot:>
```


Histogram

In [15]:
```
iris_df.hist()
```

Out[15]:
```
array([[<AxesSubplot:title={'center':'sepal length (cm)'}>,
        <AxesSubplot:title={'center':'sepal width (cm)'}>],
       [<AxesSubplot:title={'center':'petal length (cm)'}>,
        <AxesSubplot:title={'center':'petal width (cm)'}>]], dtype=object)
```


Scatter plot

In [16]:
```
iris_df = pd.DataFrame(iris.data,columns=['Sepal Length','Sepal Width','Petal Length', 'Petal Width'])
ax = iris_df.plot.scatter(x='Sepal Length',y='Sepal Width',label='Sepal',color='purple')
iris_df.plot.scatter(x='Petal Length',y='Petal Width',label='Petal',color='orange',ax=ax)
iris_df.plot.scatter(x='Sepal Length',y='Sepal Width',c='Petal Length',s=40)
```

Out[16]:
```
<AxesSubplot:xlabel='Sepal Length', ylabel='Sepal Width'>
```


Conclusion

From the data set given the optimum number of clusters of iris is found to be 3.