Name - ARYAN BAJAJ

Task 2 - Prediction using Unsupervised ML (Level - Beginner) ¶

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

In [2]:
```
# Load the iris dataset
iris = datasets.load_iris()
df = pd.DataFrame(iris.data, columns = iris.feature_names)
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

Exploratory Data Analysis

In [3]:
```
df.shape
```

Out[3]:
```
(150, 4)
```

In [4]:
```
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 [5]:
```
df.isna().sum()
```

Out[5]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

In [6]:
```
df.nunique()
```

Out[6]:
```
sepal length (cm)    35
sepal width (cm)     23
petal length (cm)    43
petal width (cm)     22
dtype: int64
```

In [7]:
```
df['sepal length (cm)'].unique()
```

Out[7]:
```
array([5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.4, 4.8, 4.3, 5.8, 5.7, 5.2, 5.5,
       4.5, 5.3, 7. , 6.4, 6.9, 6.5, 6.3, 6.6, 5.9, 6. , 6.1, 5.6, 6.7,
       6.2, 6.8, 7.1, 7.6, 7.3, 7.2, 7.7, 7.4, 7.9])
```

In [8]:
```
df['sepal width (cm)'].unique()
```

Out[8]:
```
array([3.5, 3. , 3.2, 3.1, 3.6, 3.9, 3.4, 2.9, 3.7, 4. , 4.4, 3.8, 3.3,
       4.1, 4.2, 2.3, 2.8, 2.4, 2.7, 2. , 2.2, 2.5, 2.6])
```

In [9]:
```
df['petal length (cm)'].unique()
```

Out[9]:
```
array([1.4, 1.3, 1.5, 1.7, 1.6, 1.1, 1.2, 1. , 1.9, 4.7, 4.5, 4.9, 4. ,
       4.6, 3.3, 3.9, 3.5, 4.2, 3.6, 4.4, 4.1, 4.8, 4.3, 5. , 3.8, 3.7,
       5.1, 3. , 6. , 5.9, 5.6, 5.8, 6.6, 6.3, 6.1, 5.3, 5.5, 6.7, 6.9,
       5.7, 6.4, 5.4, 5.2])
```

In [10]:
```
df['petal width (cm)'].unique()
```

Out[10]:
```
array([0.2, 0.4, 0.3, 0.1, 0.5, 0.6, 1.4, 1.5, 1.3, 1.6, 1. , 1.1, 1.8,
       1.2, 1.7, 2.5, 1.9, 2.1, 2.2, 2. , 2.4, 2.3])
```

In [11]:
```
df.corr()
```

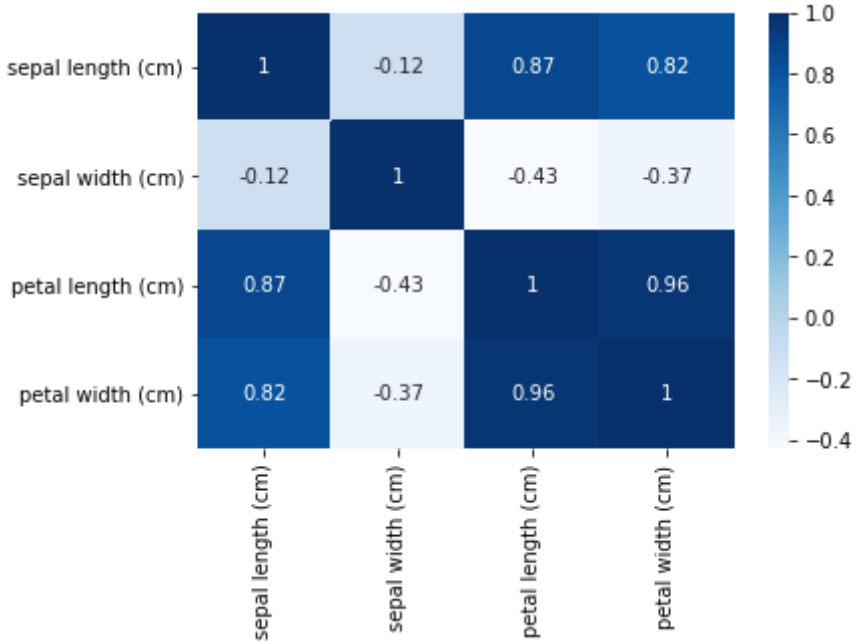
Out[11]:

	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

Visualizing the Data

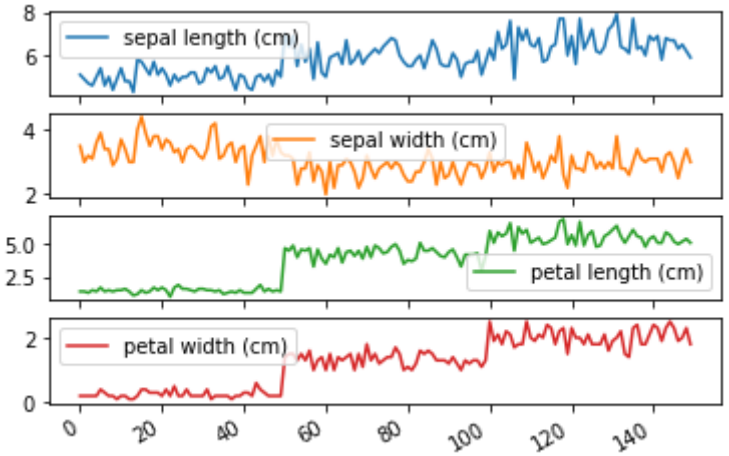
In [12]:
```
sns.heatmap(df.corr(),annot=True,cmap='Blues')
```

Out[12]: <AxesSubplot:>



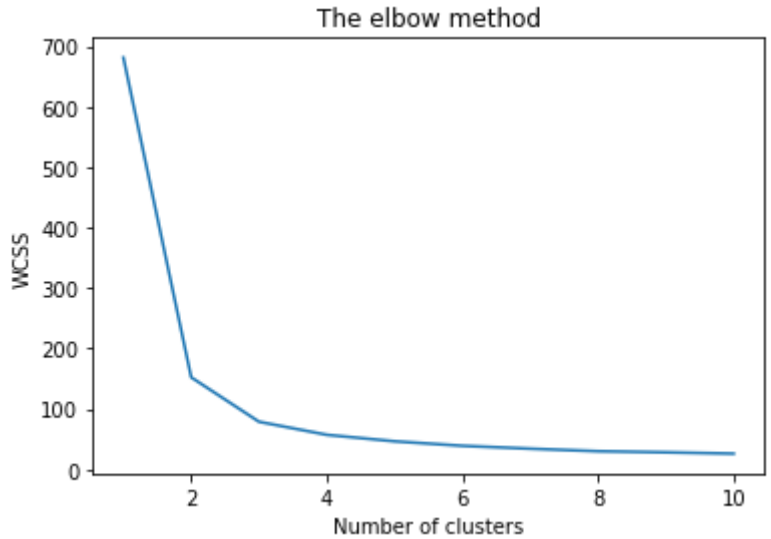
In [13]:
```
df.plot(subplots=True)
```

Out[13]:
```
array([<AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>],
      dtype=object)
```



In [14]:
```
# Finding the optimum number of clusters for k-means classification

x = df.iloc[:, [0, 1, 2, 3]].values

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()
```

```
E:\Anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is known to have a memory leak on Win
dows 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(
```

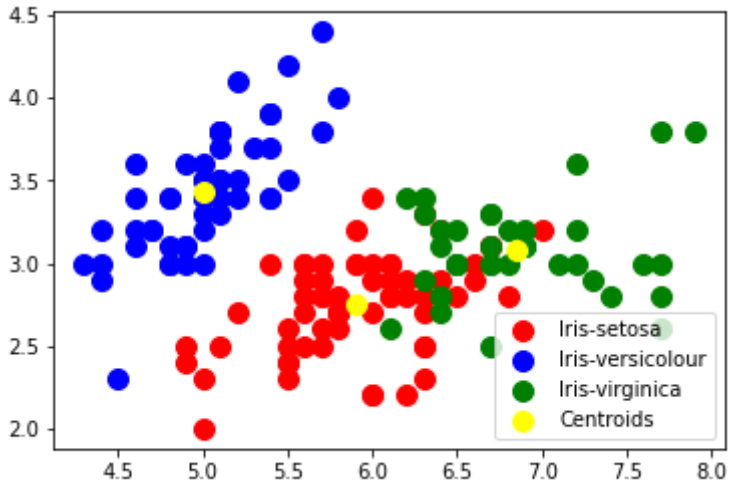


You can clearly see why it is called 'The elbow method' from the above graph, the optimum clusters is where the elbow occurs. This is when the within cluster sum of squares (WCSS) doesn't decrease significantly with every iteration.

In [15]:
```
# 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 [16]:
```
# 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[16]:
```
<matplotlib.legend.Legend at 0x2a4fa506970>
```



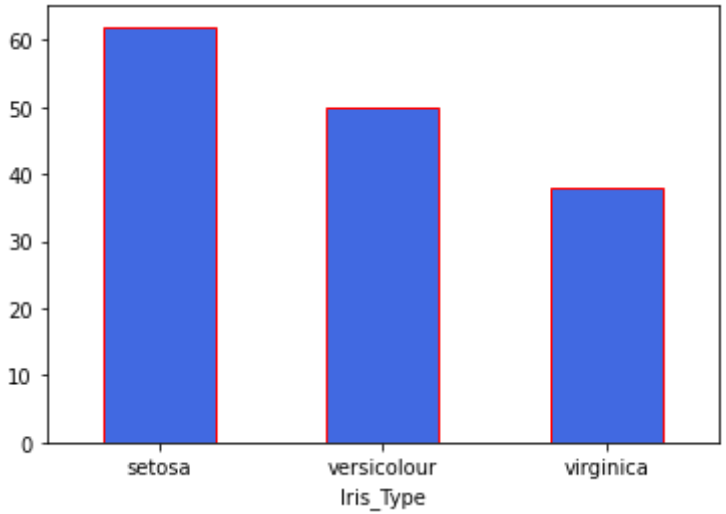
In [17]:
```
# Generate result using pandas
Iris_Type = []
for value in y_kmeans:
    if value > 1:
        Iris_Type.append("virginica")
    elif value < 1:
        Iris_Type.append("setosa")
    else:
        Iris_Type.append("versicolour")
df["Iris_Type"] = Iris_Type

df.head()
```

Out[17]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Iris_Type
0	5.1	3.5	1.4	0.2	versicolour
1	4.9	3.0	1.4	0.2	versicolour
2	4.7	3.2	1.3	0.2	versicolour
3	4.6	3.1	1.5	0.2	versicolour
4	5.0	3.6	1.4	0.2	versicolour

In [18]:
```
CP=df.value_counts('Iris_Type')
CP.plot(kind='bar', rot=0, color=['royalblue'],edgecolor='red')
print('-------------------------------------------')
print('  Number of type of Iris in Percentage (%)  ')
print('-------------------------------------------')
print(df.value_counts('Iris_Type')/df.value_counts('Iris_Type').sum()*100)
```

```
-------------------------------------------
  Number of type of Iris in Percentage (%)
-------------------------------------------
Iris_Type
setosa         41.333333
versicolour    33.333333
virginica      25.333333
dtype: float64
```



Hence, Setosa's are in greater number

THANK - YOU ^_^