Amey Ghadge_TSF Data Science intern

Task-2 Prediction using unsupervised machine learning

In [1]:

```
#Importing the libraraies
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
df=pd.read_csv("Iris.csv")
```

In [3]:

```
df
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [4]:

```
df.isnull().any()
```

Out[4]:

```
Id               False
SepalLengthCm    False
SepalWidthCm     False
PetalLengthCm    False
PetalWidthCm     False
Species          False
dtype: bool
```

In [5]:

```
df.shape
```

Out[5]:

```
(150, 6)
```

In [6]:

```
#getting information of dataset
df.info()
```

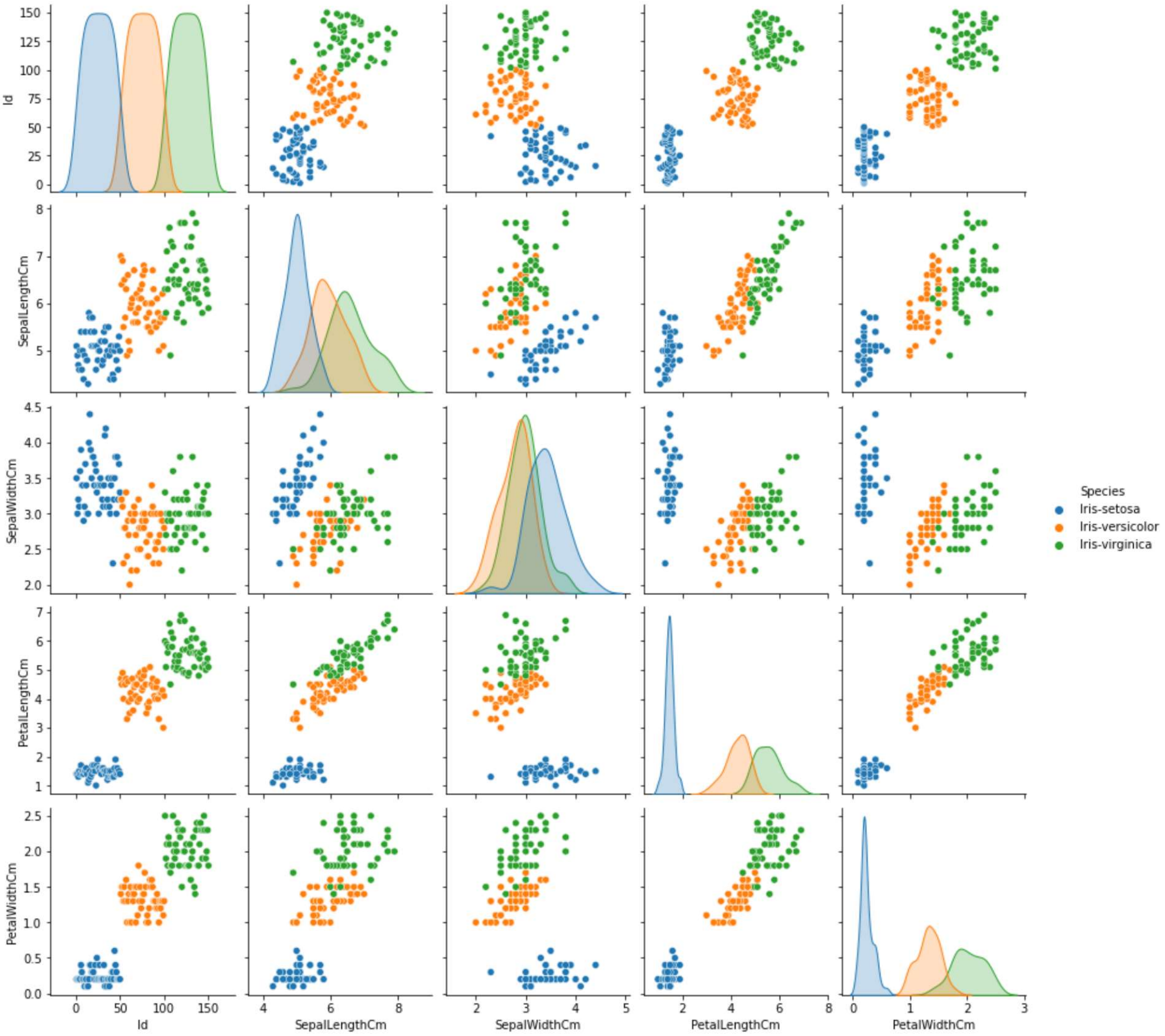
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [7]:

```
# let's plot pair plot to visualise the attributes all at once
sns.pairplot(data=df, hue = 'Species')
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x2b51ad53df0>
```

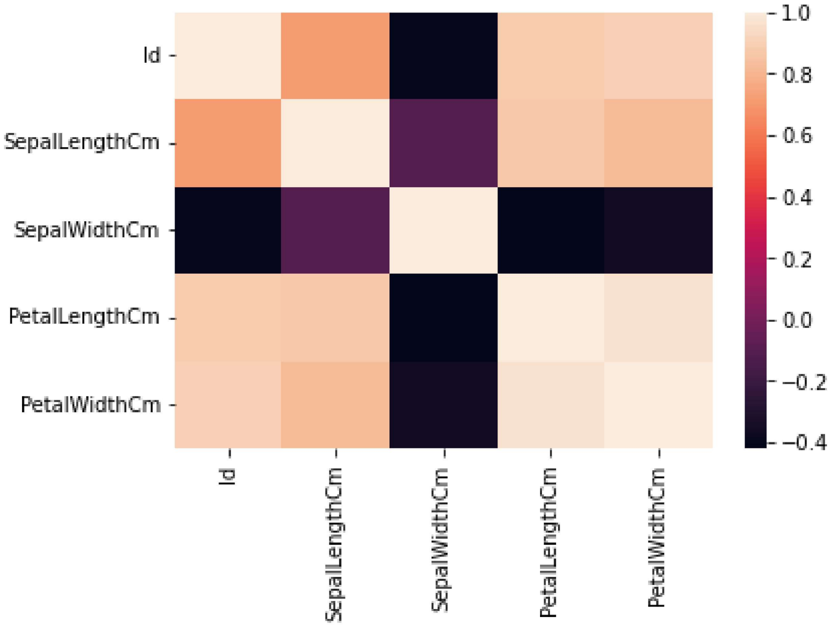


In [8]:

```
# correlation matrix
sns.heatmap(df.corr())
```

Out[8]:

```
<AxesSubplot:>
```



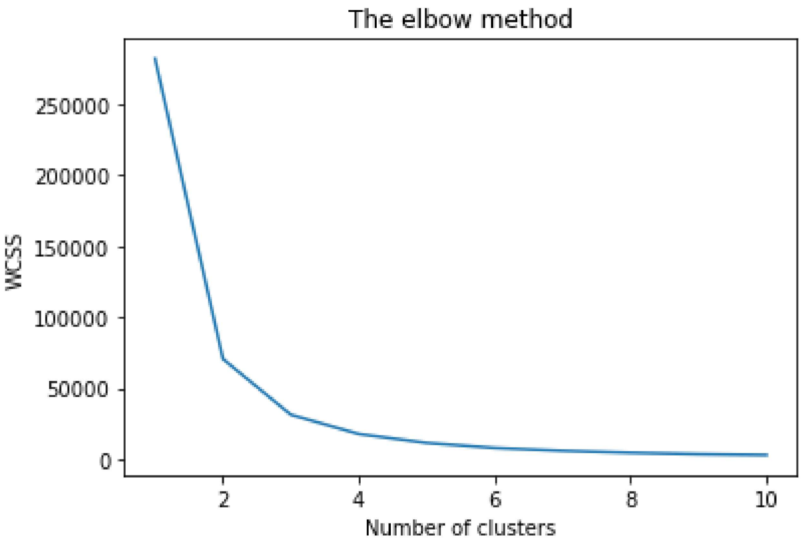
In [9]:

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

In [10]:

```
#Finding the optimum number of clusters for k-means classification

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,
    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()
```

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



In [11]:

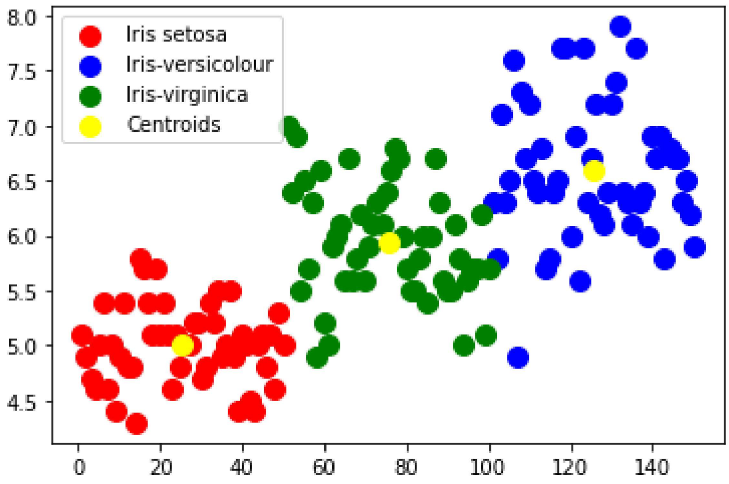
```
#Applying kmeans to the dataset/creating the means classifier
kmeans = KMeans (n_clusters = 3, init = "k-means++", max_iter = 300, n_init = 10, ra
y_kmeans = kmeans.fit_predict(x)
```

In [12]:

```
#visualising the clusters
plt.scatter(x[y_kmeans == 0, 0], x[y_kmeans == 0, 1], s = 100, c='red', label = "Iri
plt.scatter(x[y_kmeans == 1, 0], x[y_kmeans == 1, 1], s = 100, c= 'blue', label = "I
plt.scatter(x[y_kmeans == 2, 0], x[y_kmeans == 2, 1], s = 100, c='green', label="Iri

#plotting the centrolds of the clusters
plt.scatter (kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1], s= 100, c=
plt.legend()
```

Out[12]:

```
<matplotlib.legend.Legend at 0x2b51e86fbe0>
```



In [ ]: