TASK : 2 Prediction Using Unsupervised ML

Model - K - means clustering

Data preprocessing

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [6]:
```
df = pd.read_csv('C:\\Users\\Administrator\\Desktop\\GRIP spark foundation\\task 2\\Iris.csv')
```

In [7]:
```
df.head()
```

Out[7]:

	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

In [9]:
```
df.shape
```

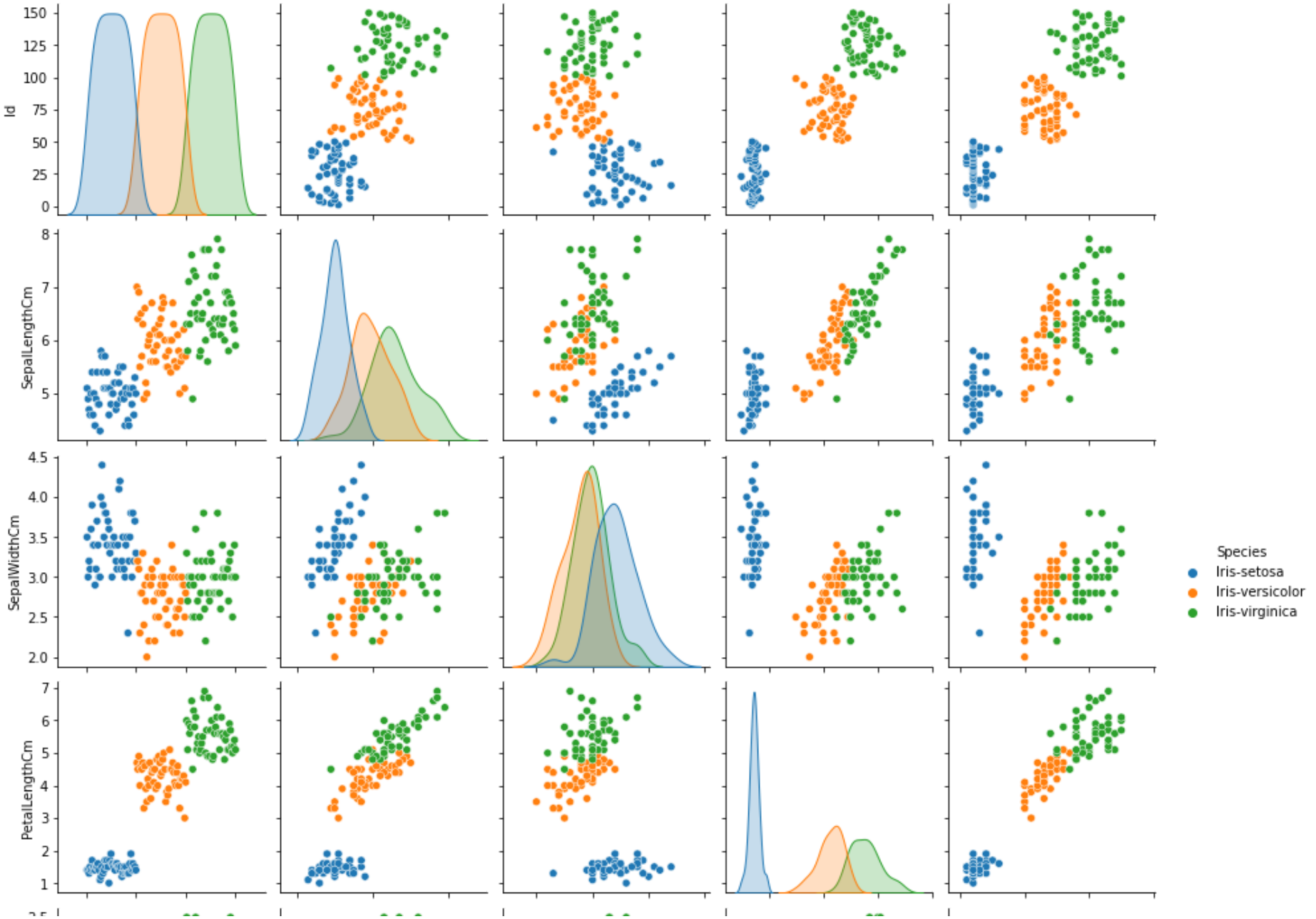
Out[9]:
```
(150, 6)
```

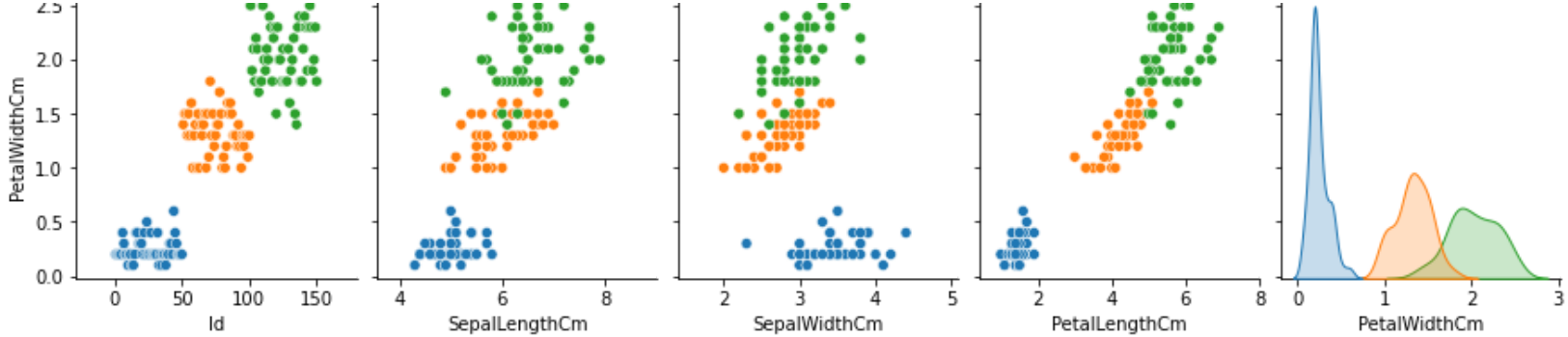
In [13]:
```
df.isnull().values.any()
```

Out[13]:
```
False
```

In [16]:
```
import seaborn as sns
sns.pairplot(df,hue = 'Species')
```

Out[16]: <seaborn.axisgrid.PairGrid at 0x2226e2e4460>





In [18]:
```
# we can easily observe that 'iris-setosa' makes a distnctive cluster in every parameter,
#while the other two pieces are overlapping a bit on each other
```

In [19]:
```
# we can determine the optimum number of cluster using elbow method
```

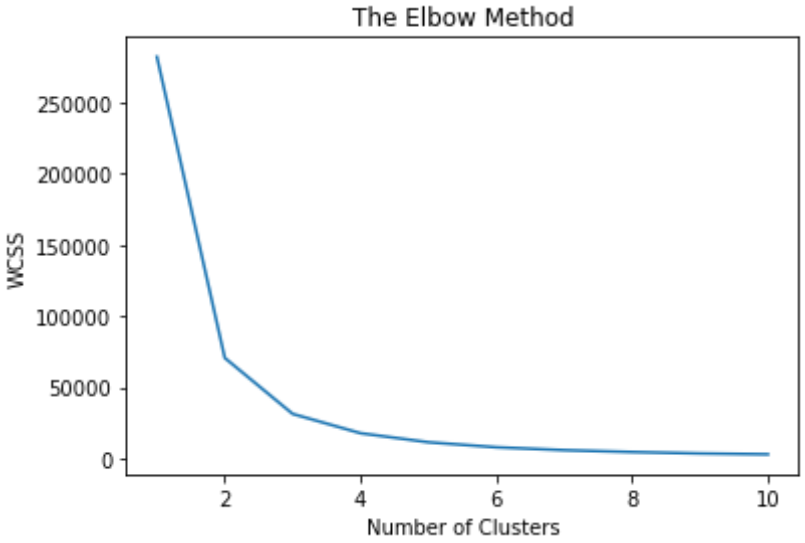
In [20]:
```
p = df.iloc[:,[0,1,2,3,4]].values
```

optimum number of clusters for K-means classification

In [23]:
```
from sklearn.cluster import KMeans
```

In [29]:
```
w = []
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(p)
    w.append(kmeans.inertia_)
```

In [31]:
```
plt.plot(range(1,11),w)
plt.title('The Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```



In [32]:
```
# we can clearly see why it is called 'the elbow method' from the above graph, the optimum clusters is where the elbo
#within cluster sum of squares (WCSS) doesn't decrese significantly with every iteration. in the above grapg, it is t
```

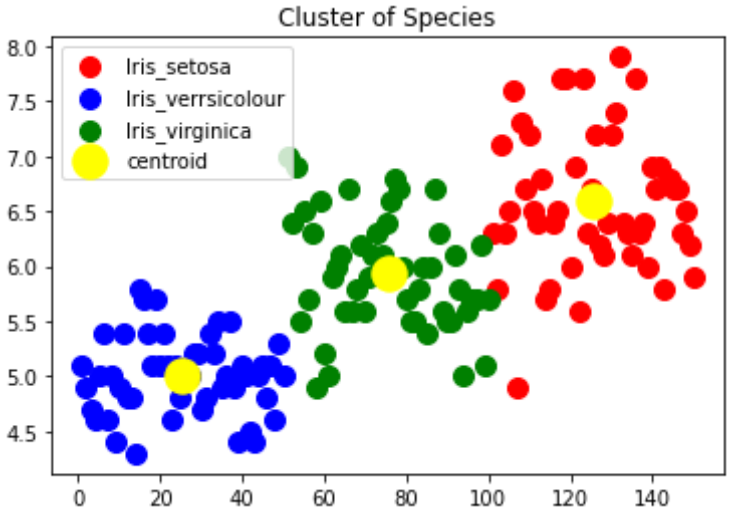
In [36]:
```
kmeans = KMeans(n_clusters = 3,init = 'k-means++', max_iter = 300, n_init=10,random_state=0)
y_kmeans = kmeans.fit_predict(p)
y_kmeans
```

Out[36]:
```
array([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, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 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, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Visualising the clusters - On the first two columns and plotting the cetroids of the clusters

In [37]:
```
plt.scatter(p[y_kmeans==0,0], p[y_kmeans==0,1],s = 100,c = 'red', label = 'Iris_setosa')
plt.scatter(p[y_kmeans==1,0], p[y_kmeans==1,1],s = 100,c = 'blue', label = 'Iris_verrsicolour')
plt.scatter(p[y_kmeans==2,0], p[y_kmeans==2,1],s = 100,c = 'green', label = 'Iris_virginica')
```

```
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s = 300, c='yellow', label= 'centroid')
plt.title('Cluster of Species')
plt.legend()
plt.show()
```



In [38]:
```
KModel = kmeans.fit(p)
KModel
```

Out[38]:
```
KMeans(n_clusters=3, random_state=0)
```

In [39]:
```
KModel.labels_
```

Out[39]:
```
array([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, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 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, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [41]:
```
KModel.cluster_centers_
```

Out[41]:
```
array([[125.5  ,   6.588,   2.974,   5.552,   2.026],
       [ 25.5  ,   5.006,   3.418,   1.464,   0.244],
```

```
       [ 75.5  ,   5.936,   2.77 ,   4.26 ,   1.326]])
```

In [ ]: