The Sparks Foundation GRIP

Author - AYUSH CHHOKER

DATA SCIENCE AND BUSINESS ANALYTICS INTERN¶

TASK -2 - Prediction using UnSupervised ML

From the given ‘Iris’ dataset, predict the optimum number of clusters and represent it visually

importing

In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import Normalizer
from sklearn.pipeline import make_pipeline
```

In [11]:

```
df =pd.read_csv('D:\\Iris.csv')
```

In [12]:

```
df.head()
```

Out[12]:

	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

EDA

In [14]:

```
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 [15]:

```
df.describe()
```

Out[15]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [16]:

```
df.shape
```

Out[16]:

```
(150, 6)
```

In [17]:

```
df.corr()
```

Out[17]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.716676	-0.397729	0.882747	0.899759
SepalLengthCm	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.882747	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.899759	0.817954	-0.356544	0.962757	1.000000

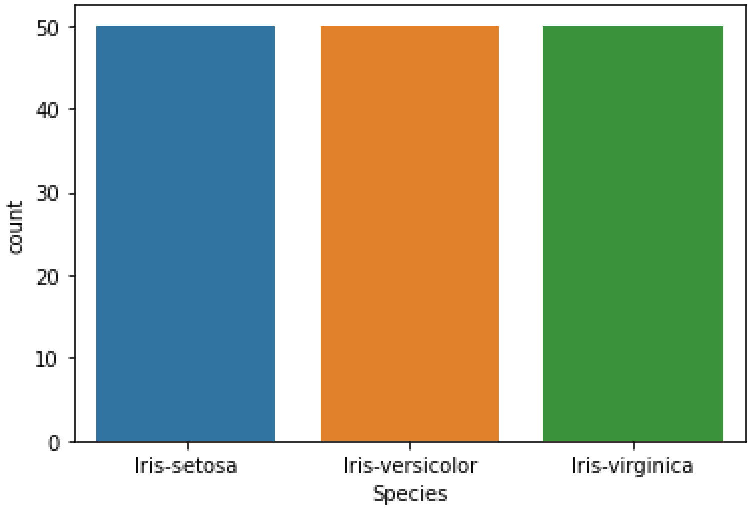
Data Visualizations

In [18]:

```
sns.countplot(df['Species'])
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x15f057d39a0>
```

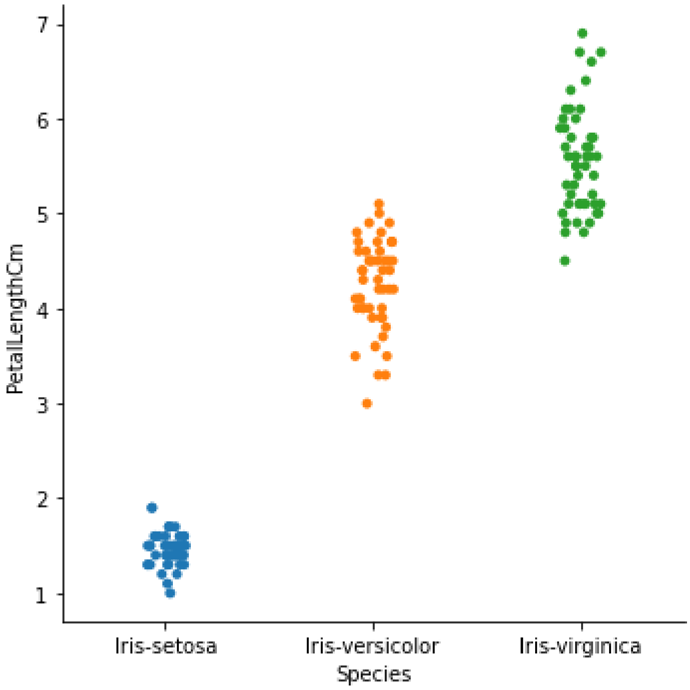


In [19]:

```
sns.catplot("Species", "PetalLengthCm", data = df)
```

Out[19]:

<seaborn.axisgrid.FacetGrid at 0x15f0588e250>

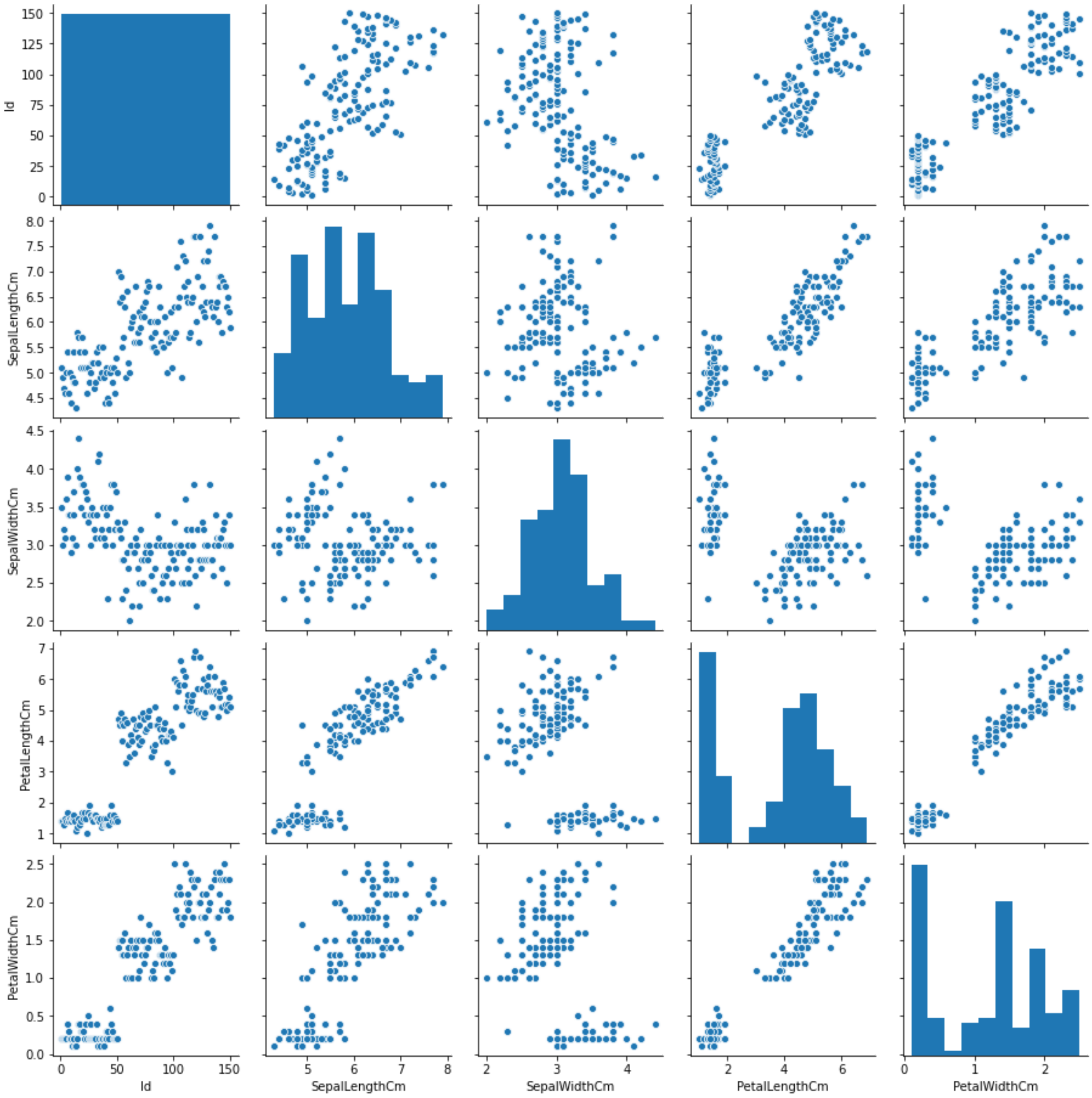


In [20]:

```
sns.pairplot(df)
```

Out[20]:

<seaborn.axisgrid.PairGrid at 0x15f058e3700>



Let's look at the Elbow graph to find k

In [23]:

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

In [24]:

```
type(x)
```

Out[24]:

```
numpy.ndarray
```

In [25]:

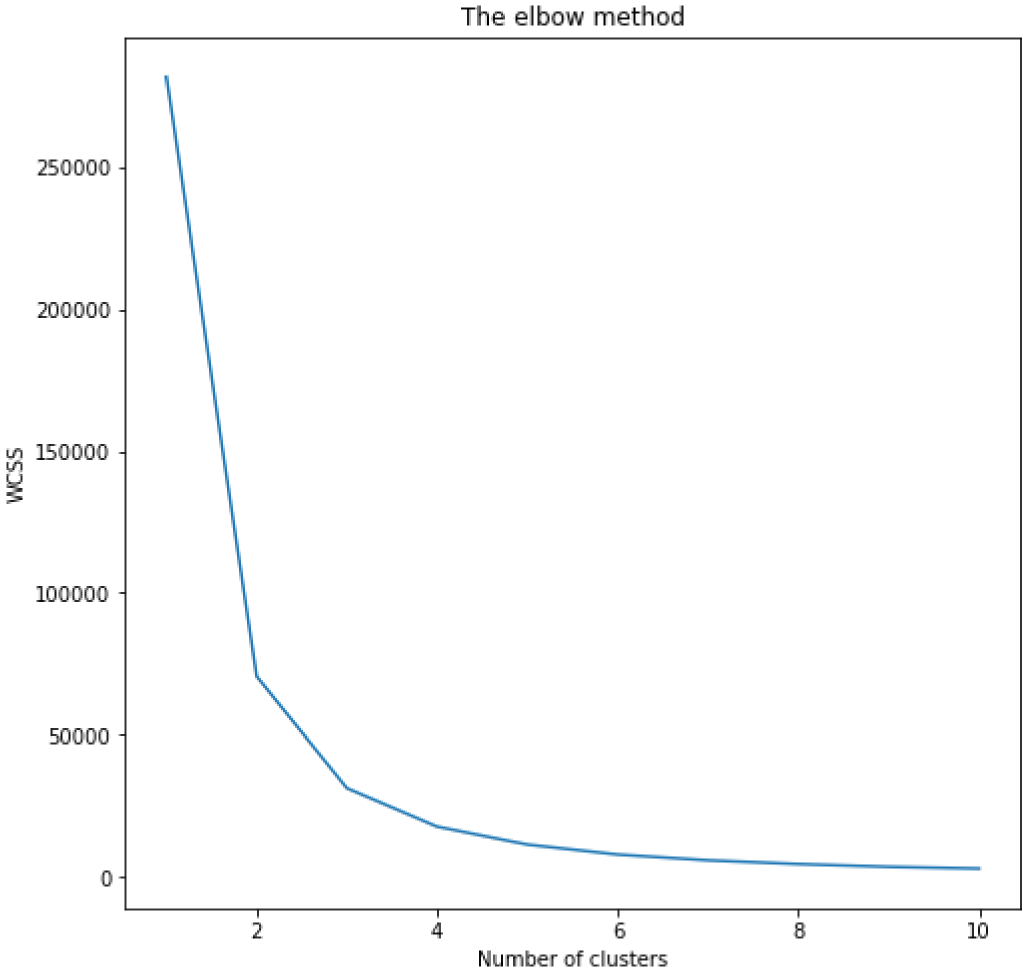
```
from sklearn.cluster import KMeans
```

In [26]:

```
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 [27]:

```
plt.figure(figsize=(8,8))
plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```


The elbow method
WCSS
250000
200000
150000
100000
50000
0
2
4
6
8
10
Number of clusters

In [28]:

```
kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300, n_init = 10, random
_state = 0)
y_kmeans = kmeans.fit_predict(x)
```

In [31]:

```
# Predict the cluster labels: labels
labels = kmeans.predict(x)
```

In [32]:

```
labels
```

Out[32]:

```
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, 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, 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, 0])
```

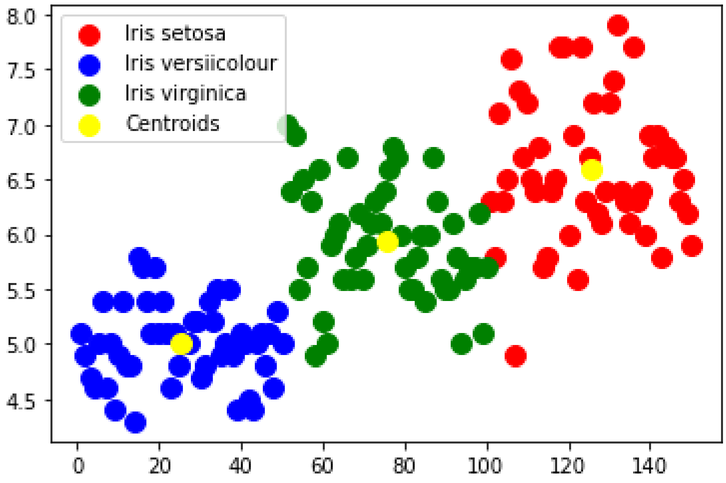
In [38]:

```
plt.scatter(x[labels == 0, 0], x[labels == 0, 1],
            s = 100, c = 'red', label = 'Iris setosa')
plt.scatter(x[labels == 1, 0], x[labels == 1, 1],
            s = 100, c = 'blue', label = 'Iris versiicolour')
plt.scatter(x[labels == 2, 0], x[labels == 2, 1],
            s = 100, c = 'green', label = 'Iris virginica')


plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,1],
           s = 100, c = 'yellow', label = 'Centroids')
plt.legend()
```

Out[38]:

```
<matplotlib.legend.Legend at 0x15f065321f0>
```



In [42]:

```
Species = ['Iris-setosa', 'Iris-versicolour','Iris-virginica']
Species_ = []
for i in labels:
 Species_.append(Species[i])
```

In [45]:

```
Species_
```

Out[45]:

```
['Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-versicolour',
 'Iris-virginica',
 'Iris-virginica',
 'Iris-virginica',
 'Iris-virginica',
 'Iris-virginica',
 'Iris-virginica',
 'Iris-virginica',
 'Iris-virginica',
 'Iris-virginica',
```

```
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-virginica',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
    'Iris-setosa',
```

```
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa',
 'Iris-setosa']
```

In [48]:

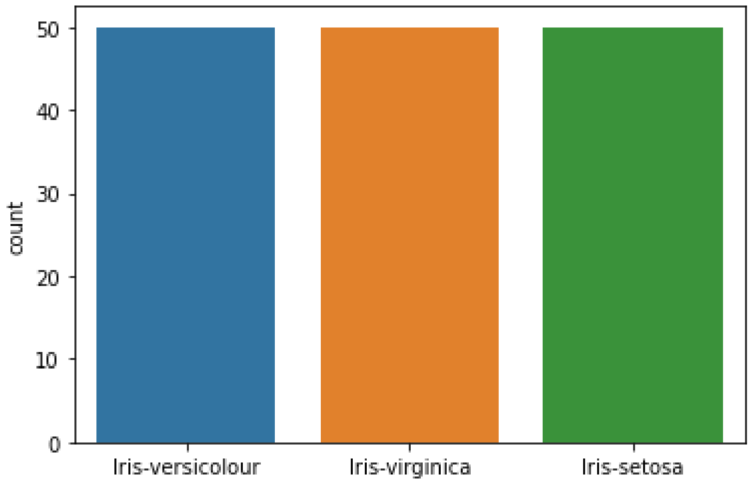
```
df['predicted_species'] = Species_
```

In [51]:

```
sns.countplot(Species_)
```

Out[51]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x15f08256a00>
```



In [ ]:

```

```