The Sparks Foundation

Task 2: Prediction Using Unsupervised ML

- Predict the optimal Number of Clusters and represent it visually.

Done By: Harini G

Importing Required Libraries

In [3]:

```
#importing libraries

import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder

#importing visualization libraries

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as pgo
```

Importing Dataset

In [6]:

```
df=pd.read_csv("D:/Harini(christ unniversity)/Internship/Iris.csv")
df.head(5)
```

Out[6]:

	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 [7]:

```
##dropping the column Id

df=df.drop('Id',axis=1)
```

In [8]:

```
X=df.drop("Species",axis=1)
y=df['Species']
```

Exploratory Data Analysis

In [9]:

```
##Finding the shape of the dataset

X.shape
```

Out[9]: (150, 4)

The dataset contain 150 instances with 4 features.

In [10]:

```
##finding the information of the dataset

X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
```

In [11]:

```
##finding the NaN values of the dataset

X.isna().sum()
```

Out[11]:

```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
dtype: int64
```

There is no Null values or NaN values in the dataset.

In [12]:

```
##Taking the description of the dataset

X.describe()
```

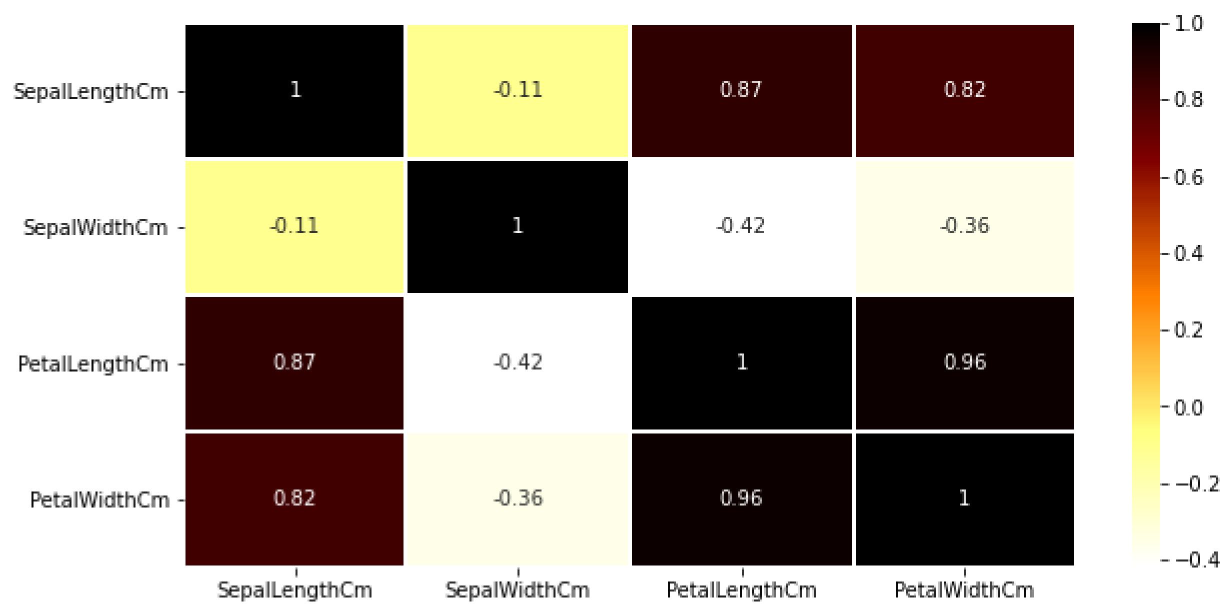
Out[12]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
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

Correlation

In [13]:

```
plt.figure(figsize = (10, 5))
sns.heatmap(X.corr(), linecolor = 'white', linewidths = 1, cmap ='afmhot_r', annot = True)
plt.show()
```

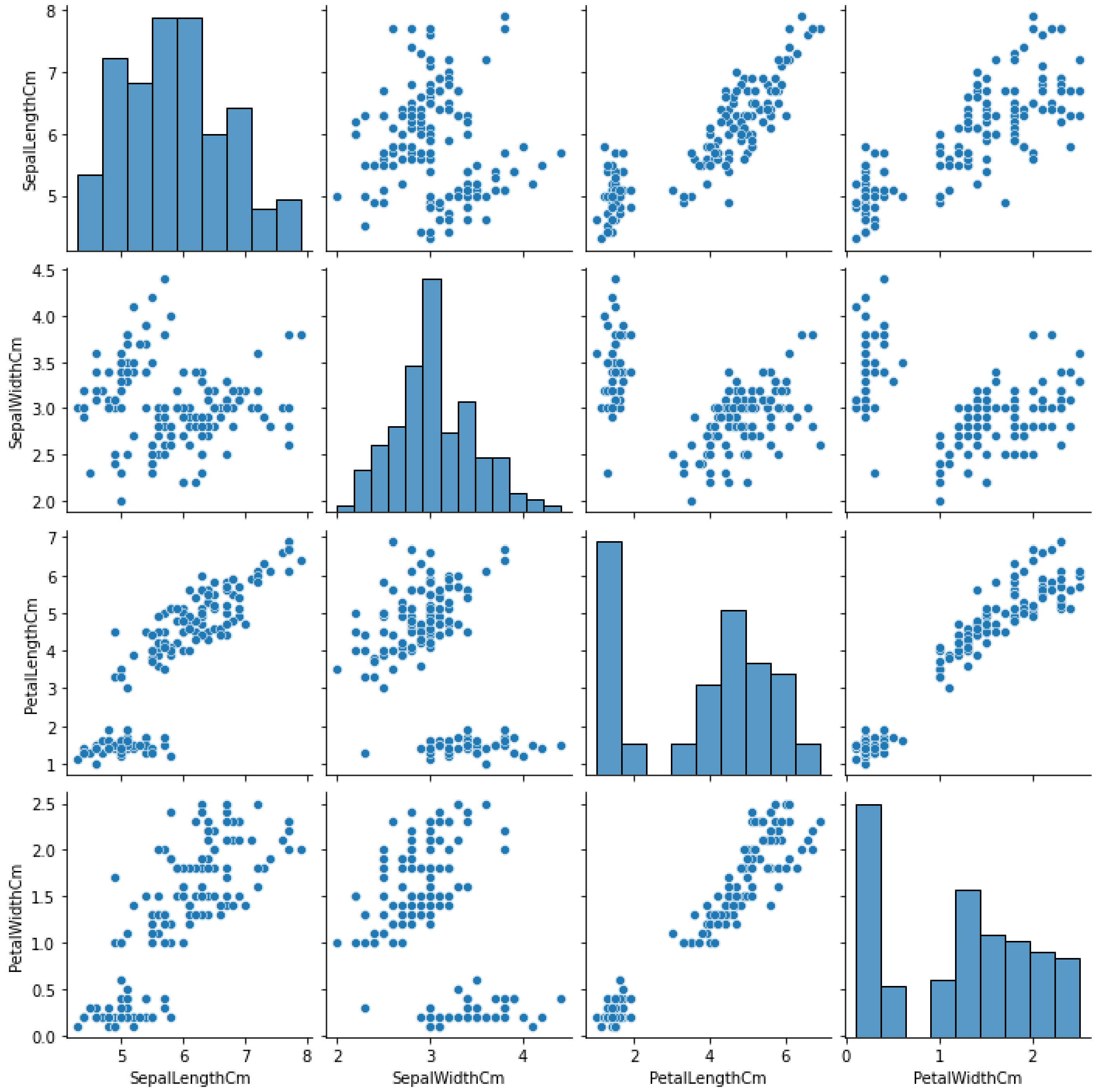


Here, Petal Length and Petal Width are highly correlated.

Pairplot

In [14]:
```
sns.pairplot(X)
```

Out[14]: <seaborn.axisgrid.PairGrid at 0x20233583f98>



In [15]:
```
x = X.iloc[:, [0, 1, 2, 3]].values
```

In [16]:
```
import warnings
warnings.filterwarnings("ignore")
```

Finding the optimum number of clusters for k-means clustering

In [17]:
```
#Finding the optimum number of clusters for k-means clustering
from sklearn.cluster import KMeans
Error =[]
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i).fit(x)
    kmeans.fit(X)
    Error.append(kmeans.inertia_)
```

In [18]:

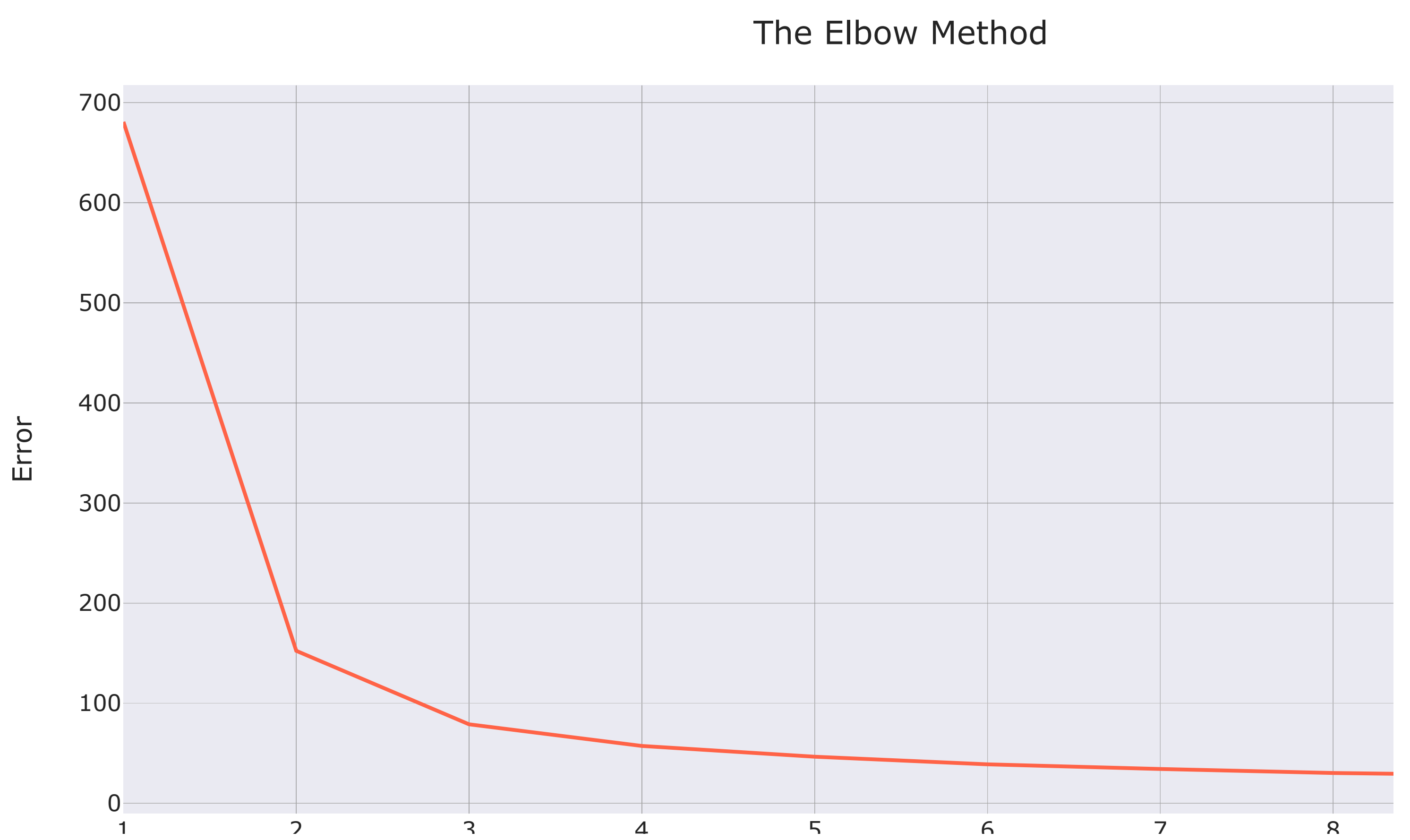
```
uu=pd.DataFrame(Error)
uu.columns=['Error']
```

In [19]:

```
fig =px.line(x=range(1, 11),y=uu['Error'],color_discrete_sequence=[ "tomato"])

fig.update_layout(title = {'text':'The Elbow Method',
                           'y':0.95,
                           'x':0.5},
                  xaxis_title='No.of Clusters',
                  yaxis_title='Error',
                  template='seaborn'
                  )

fig.show()
```



The Optimal Number of Cluter is 3.

K-Means Clustering

In [20]:

```
kmeans = KMeans(n_clusters=3)
y_kmeans = kmeans.fit_predict(x)
print(y_kmeans)

kmeans.cluster_centers_
```

```
[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 0 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 0 0 0 0 2 0 0 0 0
 0 0 2 2 0 0 0 0 2 0 2 0 2 0 0 2 2 0 0 0 0 0 2 0 0 0 0 2 0 0 0 2 0 0 0 2 0
 0 2]
```

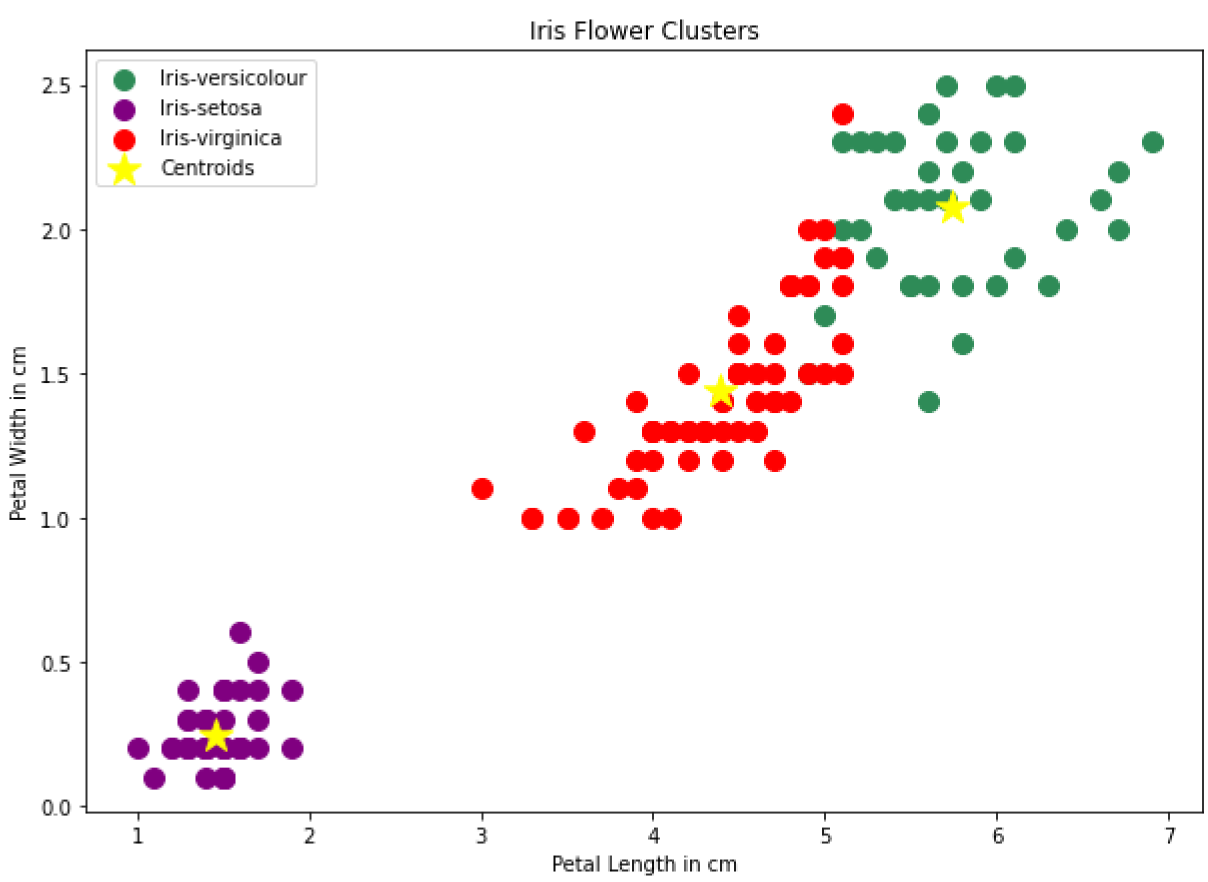
Out[20]:

```
array([[6.85      , 3.07368421, 5.74210526, 2.07105263],
       [5.006     , 3.418     , 1.464     , 0.244     ],
       [5.9016129 , 2.7483871 , 4.39354839, 1.43387097]])
```

In [21]:

```
fig = plt.figure(figsize=(10, 7))
plt.title('Clusters with Centroids',fontweight ='bold', fontsize=20)
plt.scatter(x[y_kmeans == 0, 2], x[y_kmeans == 0, 3], s = 100, c = 'seagreen', label = 'Iris-v
plt.scatter(x[y_kmeans == 1, 2], x[y_kmeans == 1, 3], s = 100, c = 'purple', label = 'Iris-set
plt.scatter(x[y_kmeans == 2, 2], x[y_kmeans == 2, 3],s = 100, c = 'red', label = 'Iris-virgini
plt.scatter(kmeans.cluster_centers_[:, 2], kmeans.cluster_centers_[:,3], s = 300, c = 'yellow'
            label = 'Centroids')
plt.title('Iris Flower Clusters')
plt.ylabel('Petal Width in cm')
plt.xlabel('Petal Length in cm')
plt.legend()
```

Out[21]: <matplotlib.legend.Legend at 0x20236c7ef60>



In [ ]:

```
 
```