The Spark Foundation - Internship

Data Science And Business Analytics

Author:Shubhashree M
TASK : 2
Performing Clustering Techniques on Iris Dataset

Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

Reading the Dataset

In [10]:

```
df = pd.read_csv('Downloads/iris.csv')
```

In [11]:

```
df.head(10)
```

Out[11]:

	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
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

Data cleaning

Checking for null values

In [13]:

```
df.isna().sum()
```

Out[13]:

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
```

```
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Displaying the shape of the data

In [15]:

```
df.shape
```

Out[15]:

```
(150, 6)
```

Displaying the information abt the dataset

In [16]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
Id               150 non-null int64
SepalLengthCm    150 non-null float64
SepalWidthCm     150 non-null float64
PetalLengthCm    150 non-null float64
PetalWidthCm     150 non-null float64
Species          150 non-null object
dtypes: float64(4), int64(1),object(1)
memory usage: 7.1+ KB
```

In [17]:

```
df.describe()
```

Out[17]:

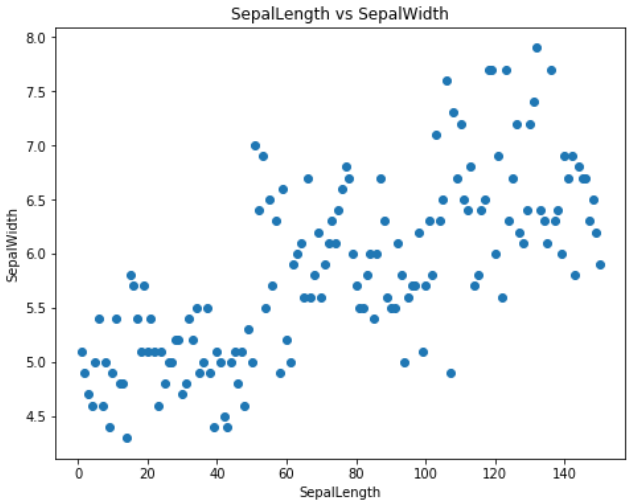
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	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

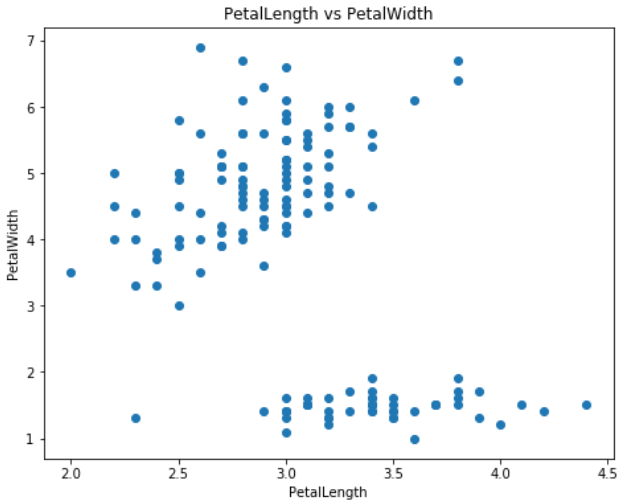
Relation between Sepal length, sepal width and petal length, petal width

In [18]:

```
fig=plt.figure(figsize=(17,6))
ax1=fig.add_subplot(121)
ax2=fig.add_subplot(122)

ax1.scatter(df.iloc[:,0],df.iloc[:,1])
ax1.set_title('SepalLength vs SepalWidth')
ax1.set_xlabel('SepalLength')
ax1.set_ylabel('SepalWidth')

ax2.scatter(df.iloc[:,2],df.iloc[:,3])
ax2.set_title('PetalLength vs PetalWidth')
ax2.set_xlabel('PetalLength')
```

```
ax2.set_ylabel('PetalWidth')

plt.show()
```

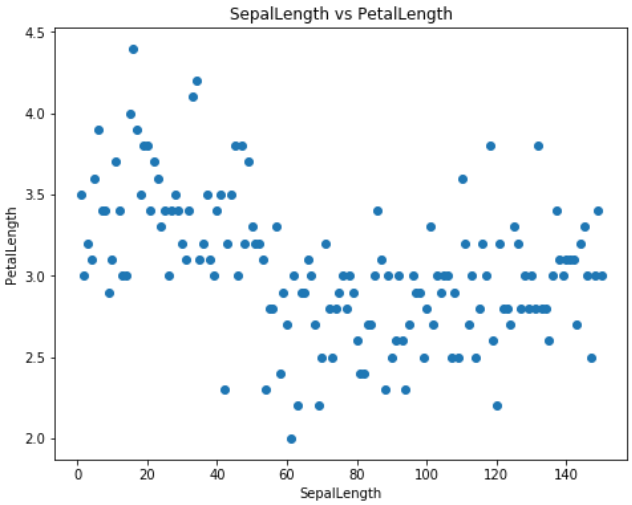


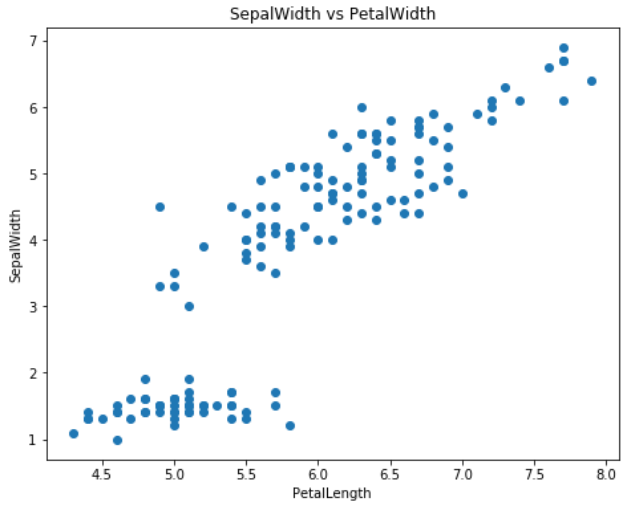


Relation between Sepal length, petal length and sepal width, petal width

In [19]:

```
fig=plt.figure(figsize=(17,6))
ax1=fig.add_subplot(121)
ax2=fig.add_subplot(122)

ax1.scatter(df.iloc[:,0],df.iloc[:,2])
ax1.set_title('SepalLength vs PetalLength')
ax1.set_xlabel('SepalLength')
ax1.set_ylabel('PetalLength')

ax2.scatter(df.iloc[:,1],df.iloc[:,3])
ax2.set_title('SepalWidth vs PetalWidth')
ax2.set_xlabel('PetalLength')
ax2.set_ylabel('SepalWidth')
plt.show()
```

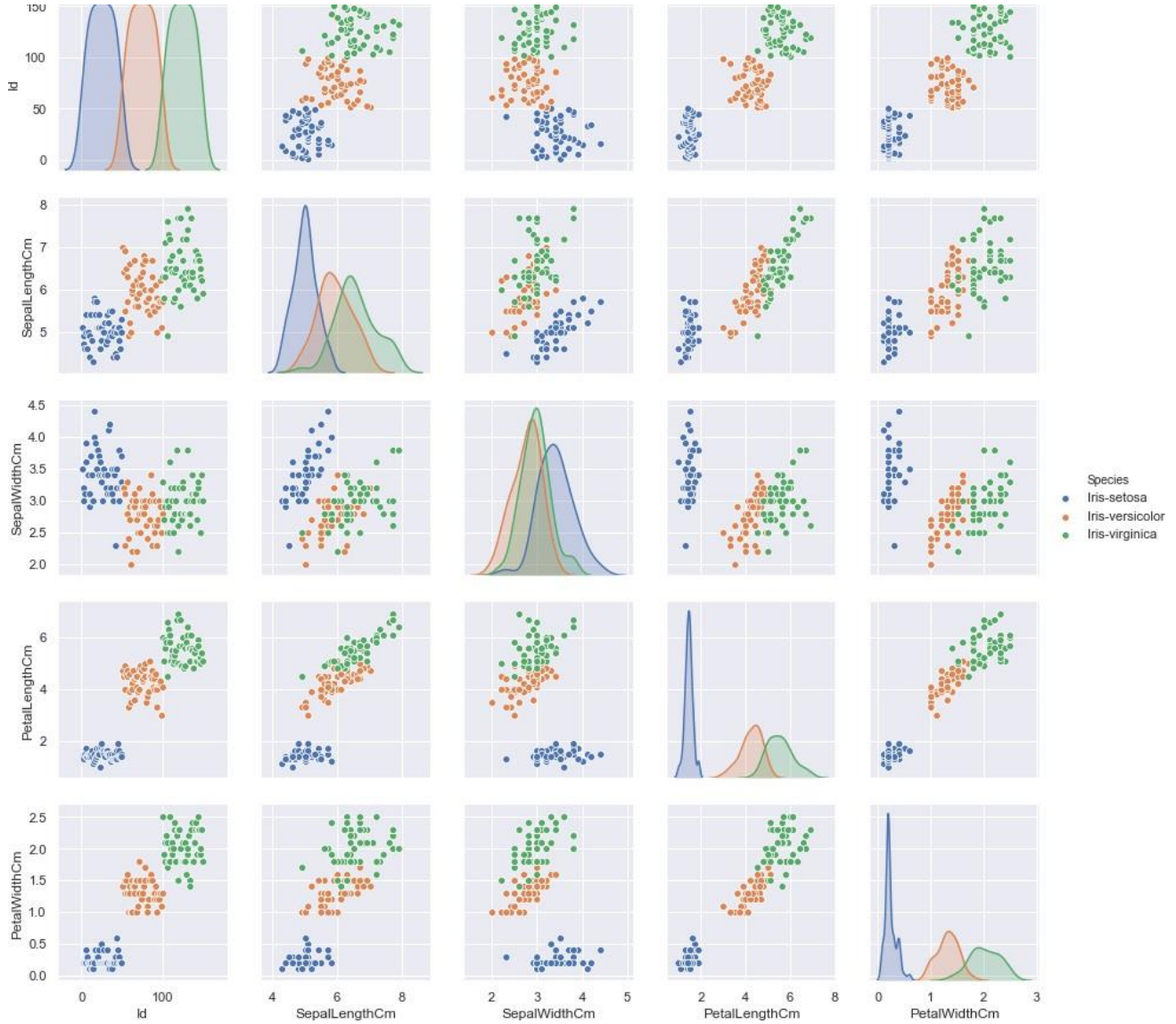




EDA

In [30]:

```
sns.pairplot(df,hue='Species')
plt.show()
```



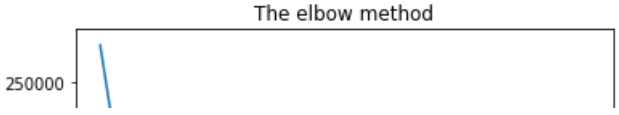
Elbow plot

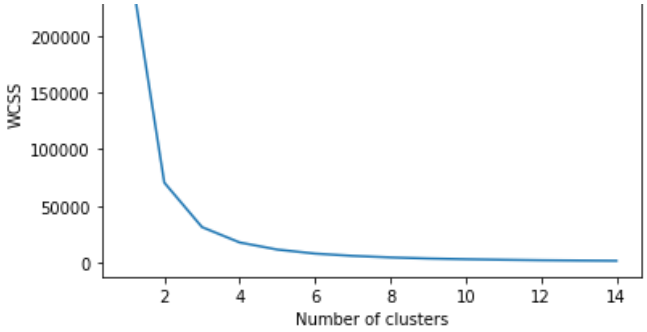
In [22]:

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

In [23]:

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 15):
    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, 15), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') # Within cluster sum of squares
plt.show()
sns.set(rc={'figure.figsize':(4,4)})
```





K-means Classification

In [24]:

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

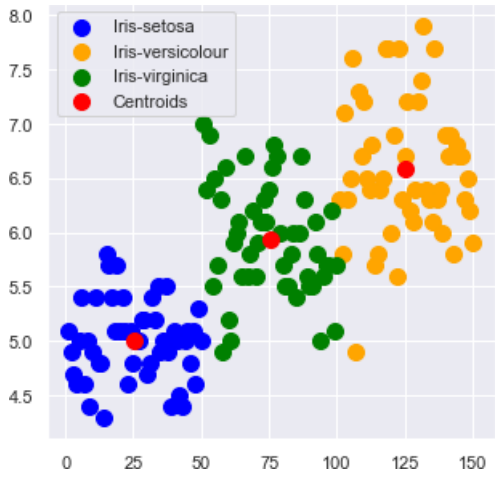
Out[24]:

```
array([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, 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, 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])
```

Visualizing the Clusters with Centroids

In [46]:

```
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1],
            s = 100, c = 'blue', label = 'Iris-setosa')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1],
            s = 100, c = 'orange', 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 = 'red', label = 'Centroids')
plt.legend()

sns.set(rc={'figure.figsize':(10,10)})
```



In [39]:

```
plt.figure(figsize=(6,5))
plt.scatter(X[y_kmeans==0,2],X[y_kmeans==0,3],s=50,c='blue',label='iris-setosa')
plt.scatter(X[y_kmeans==1,2],X[y_kmeans==1,3],s=50,c='green',label='iris-versicolour')
plt.scatter(X[y_kmeans==2,2],X[y_kmeans==2,3],s=50,c='orange',label='iris-virginica')

plt.scatter(kmeans.cluster_centers_[:,2],  kmeans.cluster_centers_[:,3],
            s = 50, c = 'red', label = 'Centroids')
plt.legend()
plt.show()
```

THANK YOU

In [ ]: