THE SPARK FOUNDATION

Data Science and Business Analytics Intern.

Task 2

Prediction using Unsupervised ML

- Objective : Predict the optimum number of clusters and represent visually.
- Method Used : K-Means Clustering
- Dataset : https://bit.ly/3kXTdox
- By : MAYURI ARUN PATHAK

Steps that we follow:

- 1) Reading the dataset
- 2) Checking and cleaning the data
- 3) EDA
- 4) Preparing Data for Modelling
- 5) Clustering Using K-means
- 6) Visualize the cluster.

- Setting Working directory

In [2]:
```
import os
os.chdir("H:\\Data Science\\Internship\\Spark")
```

Importing Libraries:

In [3]:
```
from sklearn.cluster import KMeans
from sklearn import datasets
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# Supress warnings
import warnings
warnings.filterwarnings("ignore")

#For K-Mean Clustering
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

Understanding Dataset

In [4]:
```
# Import Data
data_iris = pd.read_csv('iris.csv')
print("data load successfully")
```

```
data load successfully
```

In [5]:
```
# View some sample records
data_iris.head()
```

Out[5]:

	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 [6]:
```
#shape of the Dataframe
data_iris.shape
```

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

In [7]:
```
#Checking data type of variables
data_iris.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 [8]:
```
#Column wise Null value
data_iris.isnull().sum()
```

Out[8]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [9]:
```
#Row wise Null value
data_iris.isna().sum(axis=1)
```

Out[9]:
```
0      0
1      0
2      0
3      0
4      0
      ..
145    0
146    0
147    0
148    0
149    0
Length: 150, dtype: int64
```

- There is no null value present.

In [10]:
```
#Duplicate value checking
data_iris.duplicated()
```

Out[10]:
```
0      False
1      False
2      False
3      False
4      False
       ...
145    False
146    False
147    False
148    False
149    False
Length: 150, dtype: bool
```

- No duplicate value present.

In [11]:
```
#Counting Species
data_iris['Species'].value_counts()
```

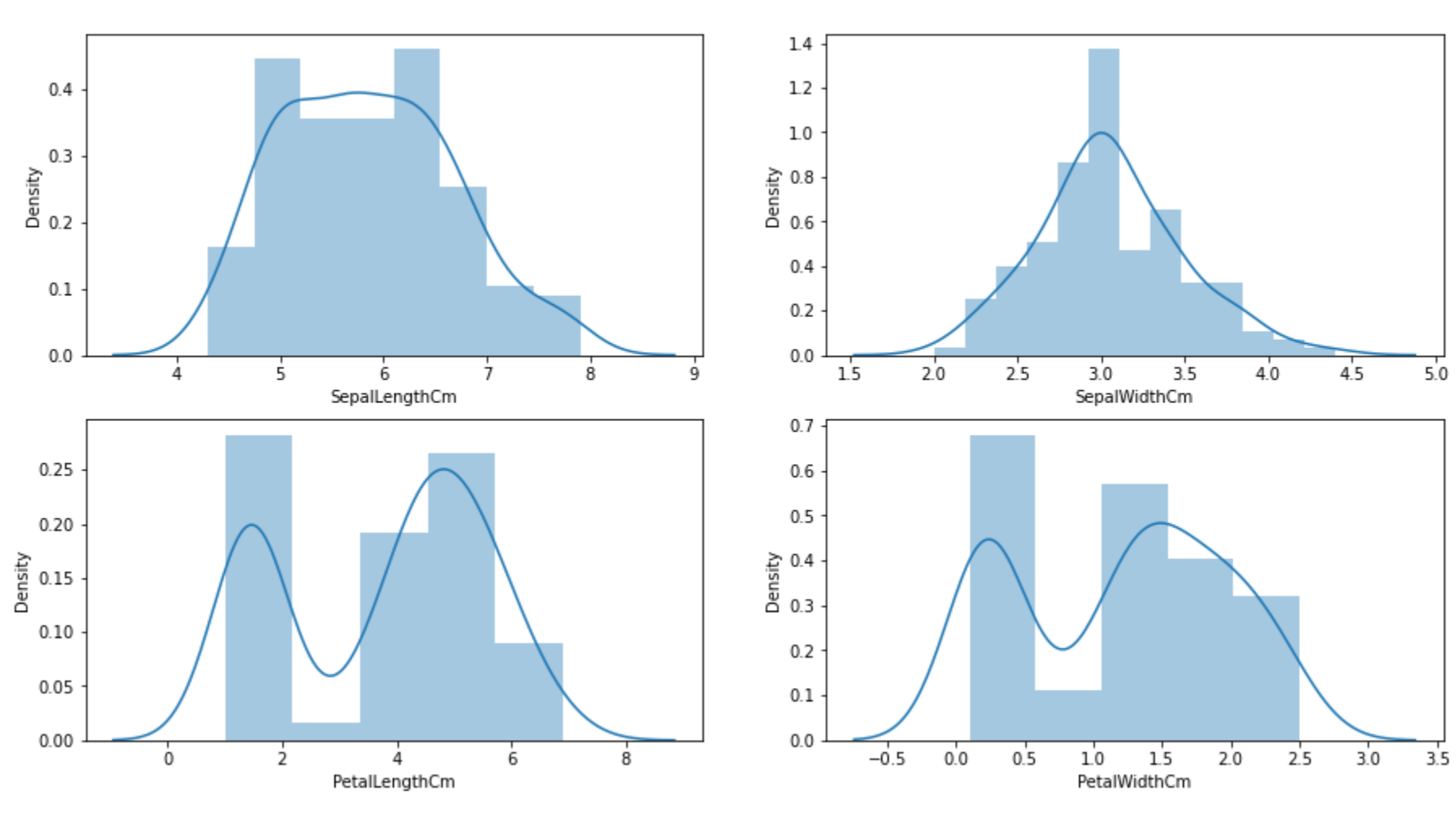
Out[11]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

- All species are in same quantity

EDA

In [12]:
```
# Distribution plot of all the continuous variables in the dataset
plt.figure(figsize = (15,8))
plt.subplot(2,2,1)
sns.distplot(data_iris["SepalLengthCm"])

plt.subplot(2,2,2)
sns.distplot(data_iris["SepalWidthCm"])

plt.subplot(2,2,3)
sns.distplot(data_iris["PetalLengthCm"])

plt.subplot(2,2,4)
sns.distplot(data_iris["PetalWidthCm"])
```

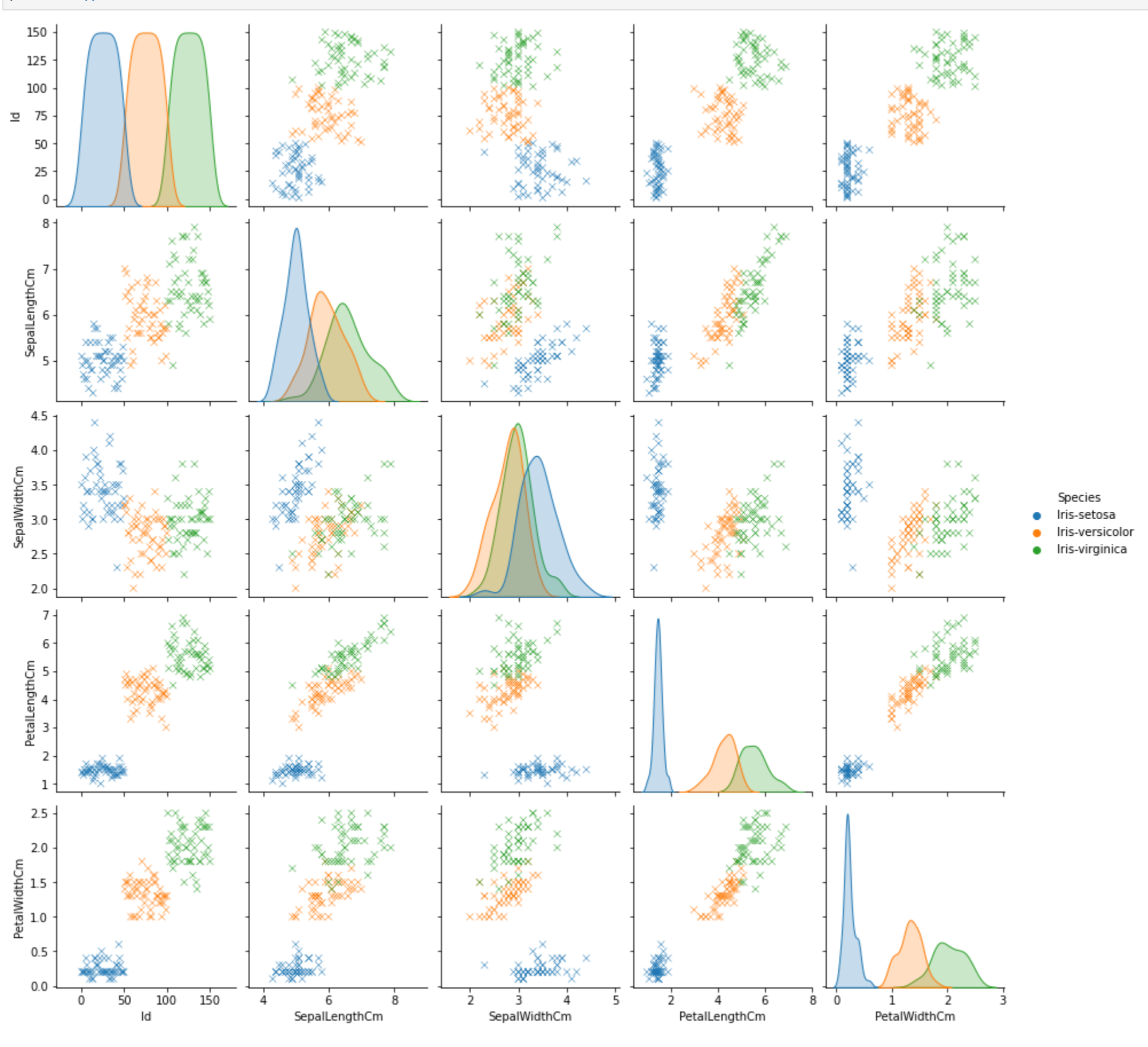
Out[12]:
```
<AxesSubplot:xlabel='PetalWidthCm', ylabel='Density'>
```



Observation

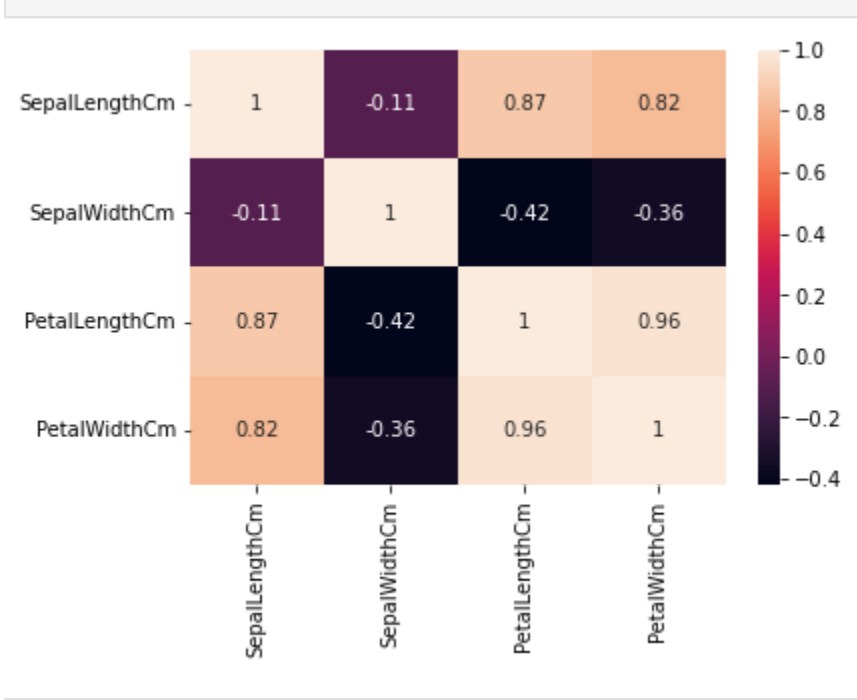
- Variables PetalLengthCm and PetalWidthCm have chance of internal grouping.Hence, For these variables we can do our further analysis.

In [13]:
```
#Pairplot of iris dataset
sns.pairplot(data_iris , hue = "Species" , markers = "x")
plt.show()
```

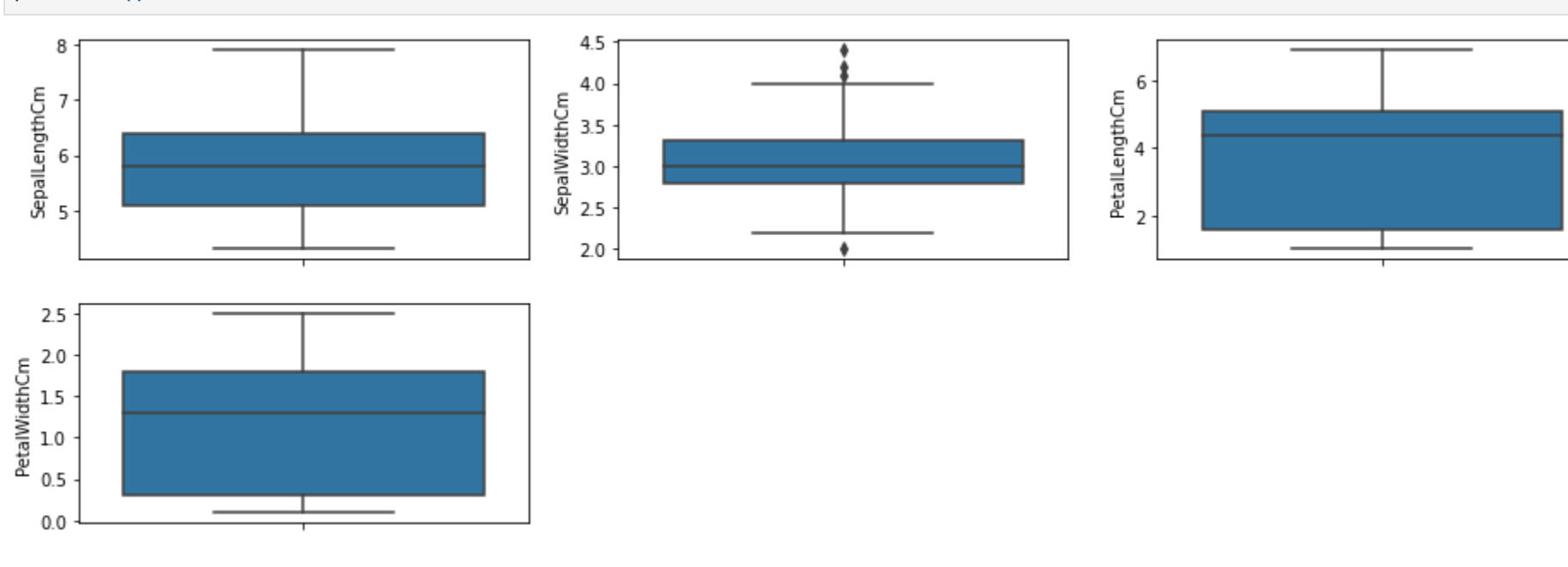


- After graphing the features in a pair plot, it is clear that the relationship beatween pairs of features of iris-setosa (in blue) is distinctly different from those of othe two species. There is some overlap in the pairwise relationships of the othe two species, iris-versicolor(orange) and iris-virginica(green).

In [14]:
```
#Correlation Betweeen Characteristic
sns.heatmap(data_iris[["SepalLengthCm","SepalWidthCm","PetalLengthCm","PetalWidthCm"]].corr() , annot = True)
plt.show()
```

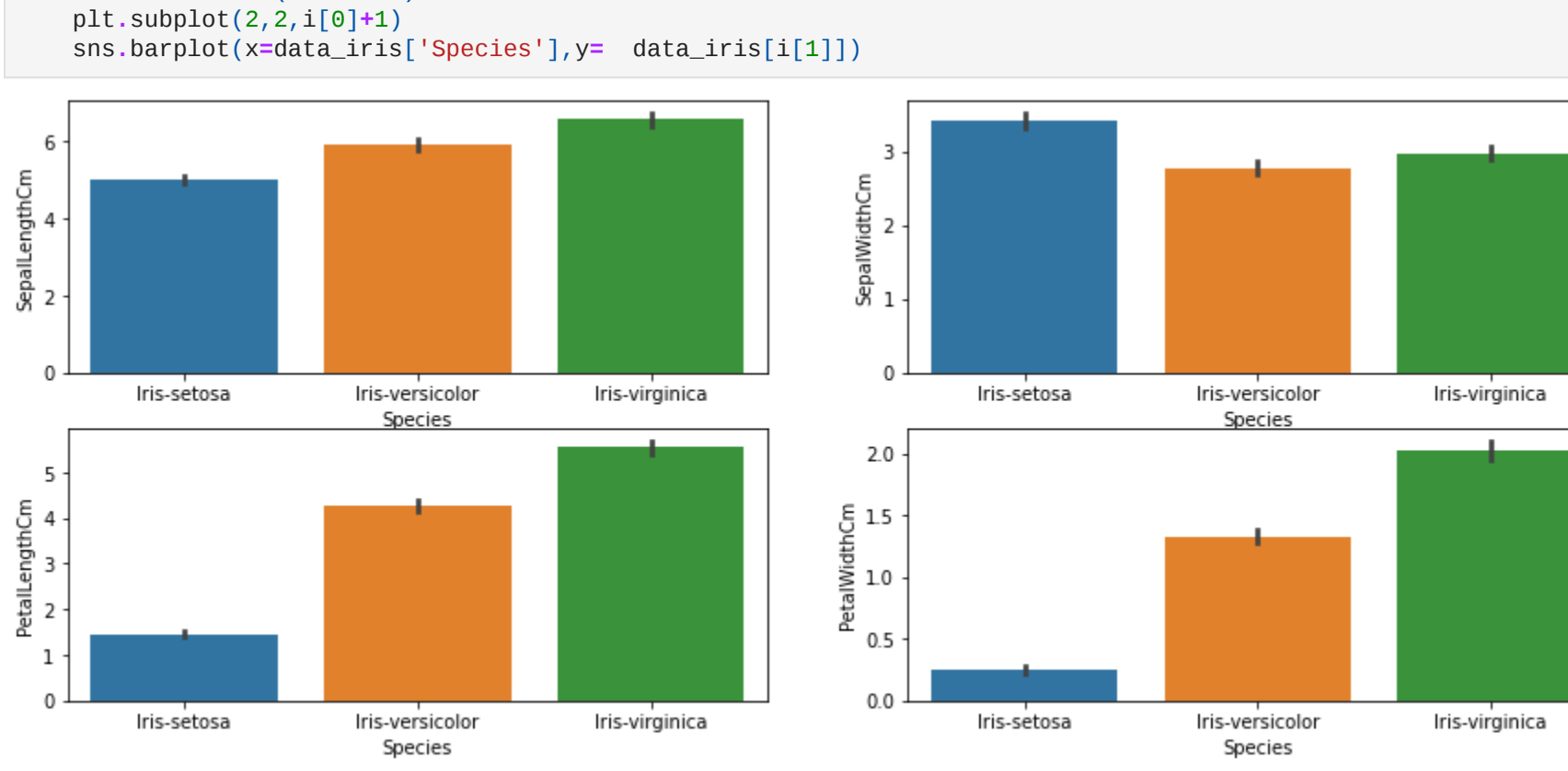


In [15]:
```
# Checking Outliers
plt.figure(figsize =(16,8))

plt.subplot(3,3,1)
sns.boxplot( y = data_iris["SepalLengthCm"])

plt.subplot(3,3,2)
sns.boxplot( y = data_iris["SepalWidthCm"])

plt.subplot(3,3,3)
sns.boxplot( y = data_iris["PetalLengthCm"])

plt.subplot(3,3,4)
sns.boxplot( y = data_iris["PetalWidthCm"])

plt.show()
```



- we can see that sepal width has some outliers.

In [16]:
```
# Barplot
plt.figure(figsize = (15,6))
features = data_iris.columns[1:5]
for i in enumerate(features):
    plt.subplot(2,2,i[0]+1)
    sns.barplot(x=data_iris['Species'],y=  data_iris[i[1]])
```



Preparing data for Modelling

In [17]:
```
data_iris.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

Modelling

In [18]:
```
import sklearn
from sklearn.preprocessing import StandardScaler
```

In [19]:
```
# Rescalling data for numeric columns
scaler = StandardScaler()
column_scaled = scaler.fit_transform(data_iris.drop(['Id','Species'],axis = 1))
column_scaled.shape
```

Out[19]:
```
(150, 4)
```

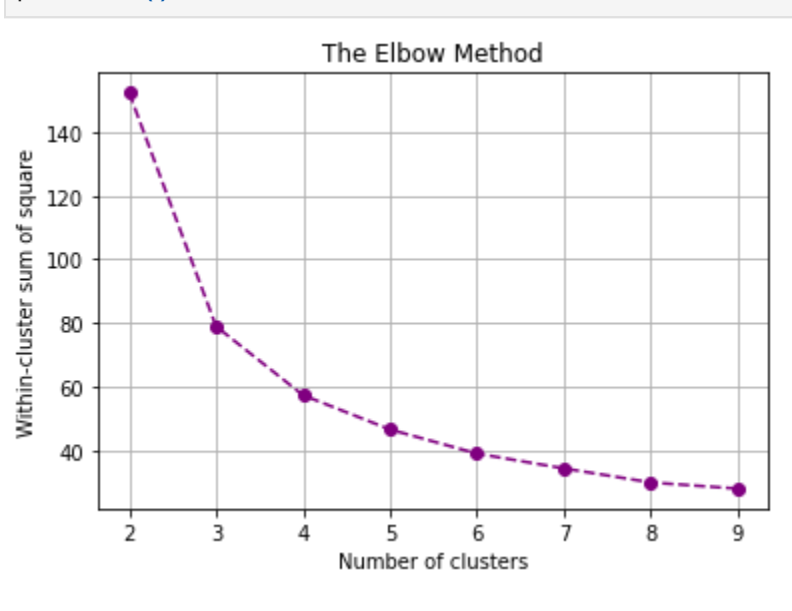
Clustering - [Using K-mean Clustering]

In [20]:
```
#checking whether clustering is possible or not using Hopkins

from sklearn.neighbors import NearestNeighbors
from random import sample
from numpy.random import uniform
import numpy as np
from math import isnan

def hopkins(X):
    d = X.shape[1]
    n = len(X)   #rows
    m = int(0.1 * n)
    nbrs = NearestNeighbors(n_neighbors=1).fit(X.values)

    rand_X = sample(range(0 ,n , 1), m)

    ujd = []
    wjd = []
    for j in range (0 ,m):
        u_dist, _ = nbrs.kneighbors(uniform(np.amin(X,axis=0),np.amax(X,axis=0),d).reshape(1,-1) , 2 , return_distance=True)
        ujd.append(u_dist[0][1])
        w_dist, _ = nbrs.kneighbors(X.iloc[rand_X[j]].values.reshape(1,-1),2,return_distance=True)
        wjd.append(w_dist[0][1])


    H = sum(ujd) / (sum(ujd) + sum(wjd))
    if isnan(H):
        print(ujd,wjd)
        H = 0

    return H
data_check_clustering = data_iris.drop(["Id","Species"], axis = 1)
hopkins(data_check_clustering)
```

Out[20]:
```
0.8506584915489421
```

- we knew it Hopkins Statistics is a way of measuring the clustering tendency of a

In [19]:
```
# Finding optimal number of clusters using elbow
ssd = []
range_n = [2,3,4,5,6,7,8,9]
for i in range_n:
    kmeans = KMeans(n_clusters=i , max_iter = 500 , random_state=50)
    kmeans.fit(data)
    ssd.append([i,kmeans.inertia_])
ssd_dataframe = pd.DataFrame(ssd)
plt.plot(ssd_dataframe[0] , ssd_dataframe[1],'go--' , color = 'purple')
plt.title("The Elbow Method")
plt.xlabel("Number of clusters")
plt.ylabel("Within-cluster sum of square")
plt.grid()
plt.show()
```



In [20]:
```
#Finding optimal clusters using sheloitte score
#Silhouette score measures that how a data point is similar to it's own cluster as compared to other cluster
range_n = [2,3,4,5,6,7]
ss = []
for i in range_n:
    kmeans = KMeans(n_clusters=i, max_iter = 500)
    kmeans.fit(data)

    cluster_label = kmeans.labels_
    silhouette_average = silhouette_score(data,cluster_label)
    ss.append([i , silhouette_average])
    print("for n cluster {0} Silhouette score is {1}".format(i,silhouette_average))
```

```
for n cluster 2 Silhouette score is 0.6808136202936816
for n cluster 3 Silhouette score is 0.5525919445499757
for n cluster 4 Silhouette score is 0.4972279726640147
for n cluster 5 Silhouette score is 0.4885175508886279
for n cluster 6 Silhouette score is 0.3695162748599971
for n cluster 7 Silhouette score is 0.3564328815074843
```

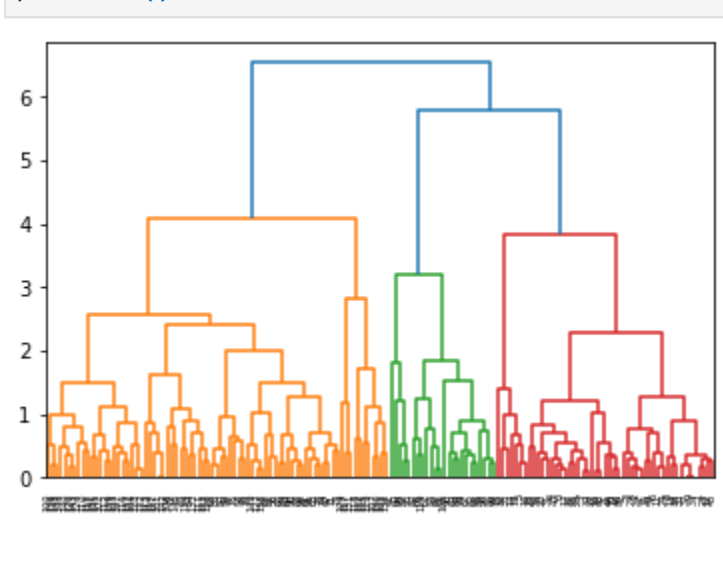
- So after checking the Elbow Curve & Silhouette Score we can choose our final k as 3.Because we have seen that in as sum of squared distance curve the 1st knee like bend has come at the level3,and in silhouette score, the score = 3 is also considerable for clustering. Though in the silhouette score for 2 is quite ipressive from score of 3,but intuitively we can say from the point of our dataset that chossing the point of cluster at 2 may not give us a optimal result that we can expect for. So for better clustering and better output and better output of our analysis. We will choose number of cluster as 3.

Hierarchical Clustering

In [32]:
```
# Reconfirming the number of cluster
```

In [33]:
```
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree
```

In [35]:
```
mergin = linkage(column_scaled,method="complete",metric="euclidean")
dendrogram(mergin)
plt.show()
```



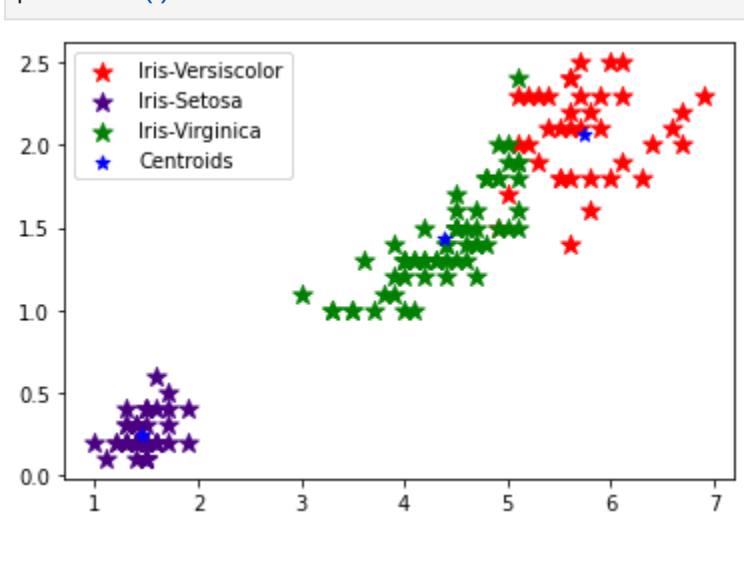
- We can clearly see, This dendrogram clearly indicate k=3. If we cut the tree between 4-5 and draw a straight line we can get 3 distinct set of clusters. We will move ahead for k-means cluster

In [23]:
```
#Assuming k = 3
kmeans = KMeans(n_clusters = 3,max_iter= 500 , random_state = 50)
Cluster_Id= kmeans.fit_predict(data)
```

In [24]:
```
Cluster_Id
```

Out[24]:
```
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, 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])
```

In [39]:
```
#Visualizing Cluster using petal length and petal width column
plt.scatter(data[Cluster_Id == 0 , 2 ], data[Cluster_Id == 0 , 3] ,s = 100 ,marker = '*',c = "red", label = "Iris-Versiscolor" )
plt.scatter(data[Cluster_Id == 1 , 2 ], data[Cluster_Id == 1 , 3] ,s = 100 ,marker = '*',c = "indigo", label = "Iris-Setosa" )
plt.scatter(data[Cluster_Id == 2 , 2 ], data[Cluster_Id == 2 , 3] ,s = 100 ,marker = '*',c = "green", label = "Iris-Virginica" )

#Centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:,2], kmeans.cluster_centers_[:,3], s = 50 ,marker = '*' ,c = "blue", label = "Centroids")

plt.legend()
plt.show()
```



Conclusion:

- Therefore, the optimum number of distinct clusters are 3.

Suggestions are always Welcome!

Thank You !