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

- Setting Working directory

In [1]:
```
import os
os.chdir("H:\\Data Science\\Internship\\Spark")
```

Importing Libraries:

In [2]:
```
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 [3]:
```
# Import Data
data_iris = pd.read_csv('iris.csv')
print(data_iris)
```

```
     Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

In [4]:
```
# View some sample records
data_iris.head()
```

Out[4]:

	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 [5]:
```
#shape of the Dataframe
data_iris.shape
```

Out[5]:
```
(150, 6)
```

In [6]:
```
#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
```

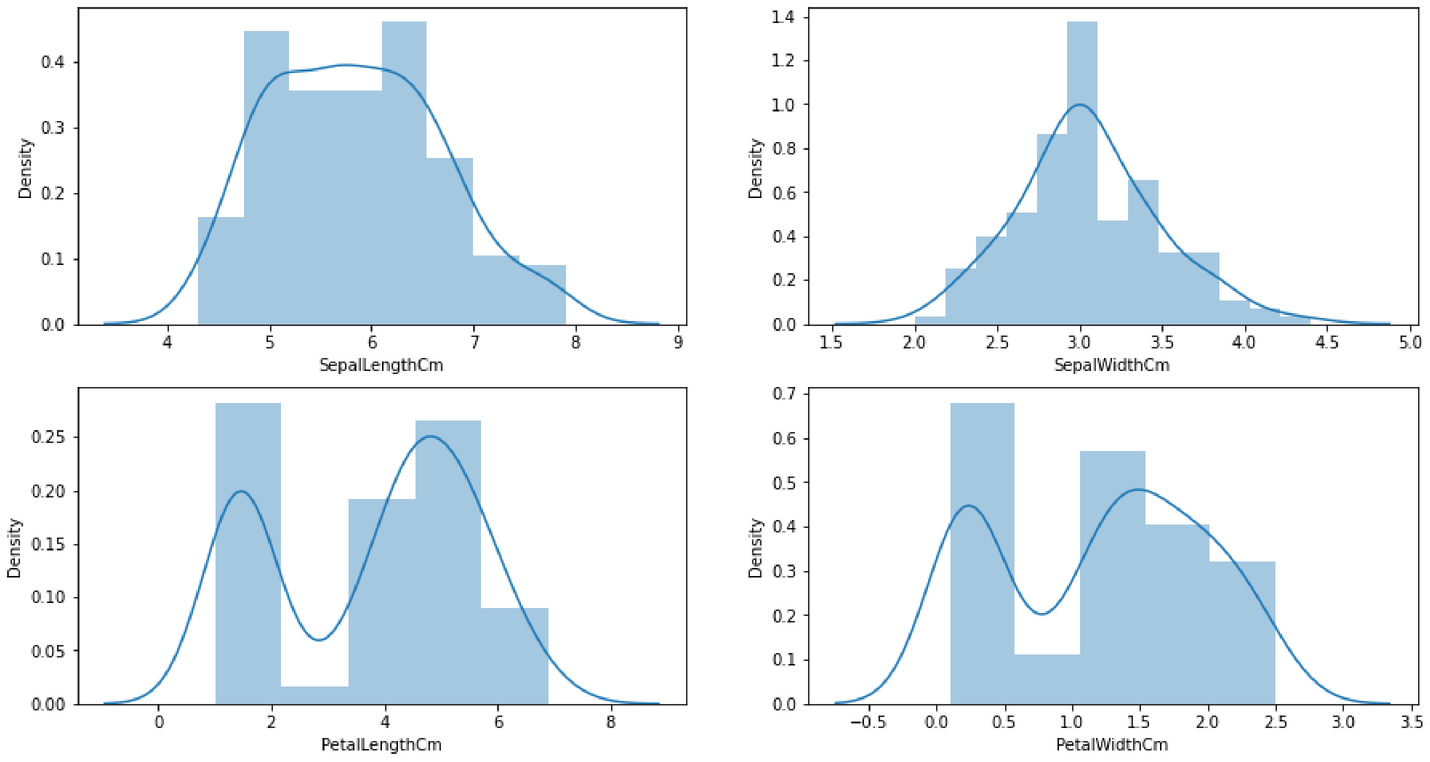
In [7]:
```
data_iris['Species'].value_counts()
```

Out[7]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

Visualization

In [8]:
```
# 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"])
```

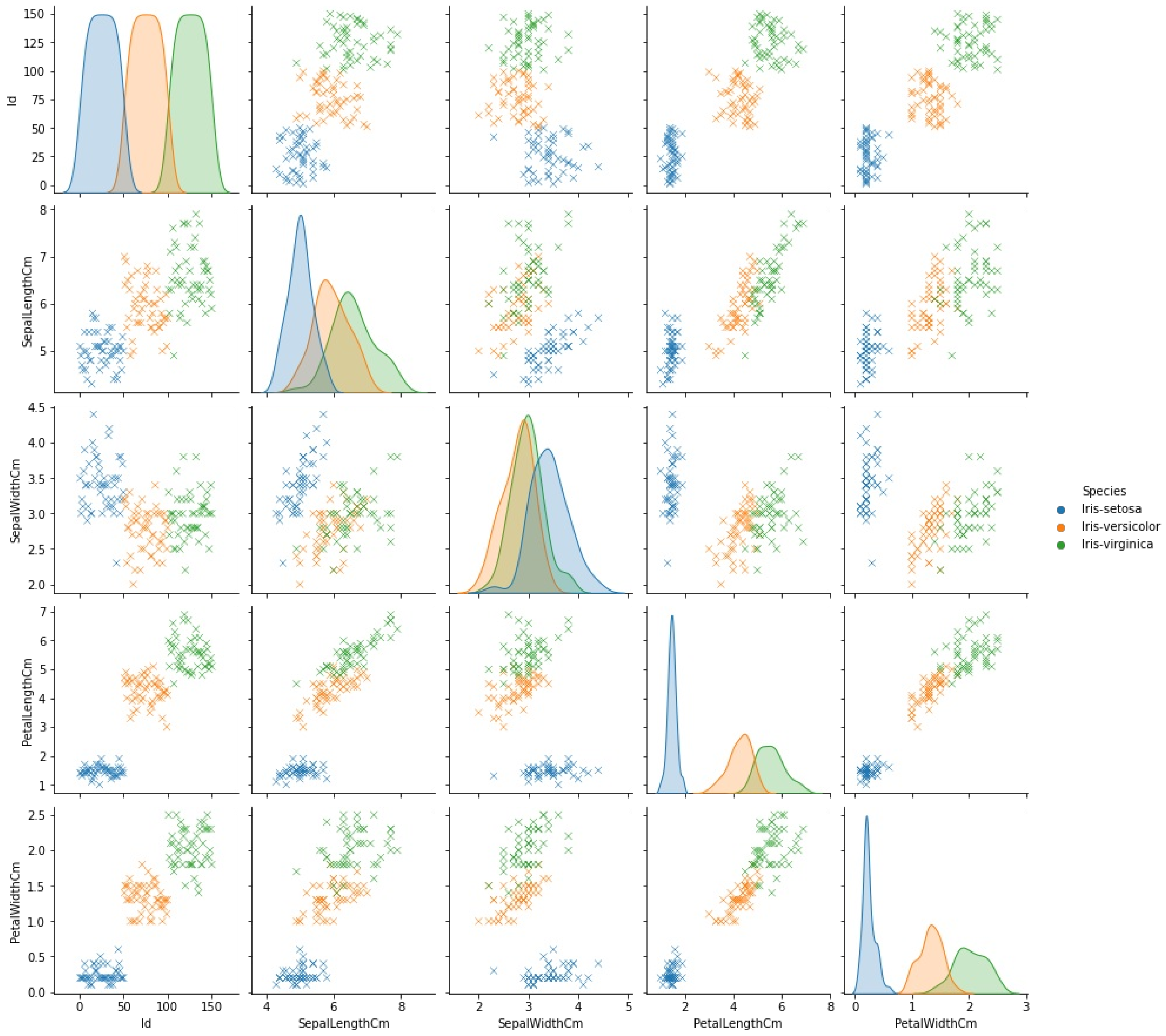
```
Out[8]:<AxesSubplot:xlabel='PetalWidthCm', ylabel='Density'>
```



Observation

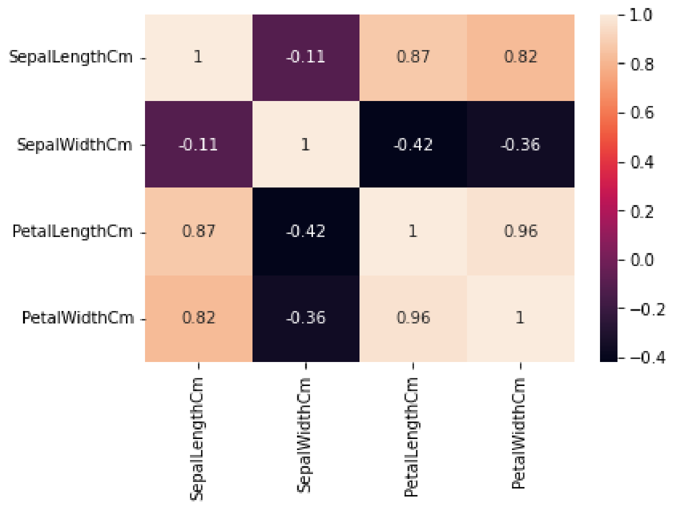
- Variables PetalLengthCm and PetalWidthCm have chance of internal grouping.Hence, For these variables we can do our further analysis.

In [9]:
```
#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).

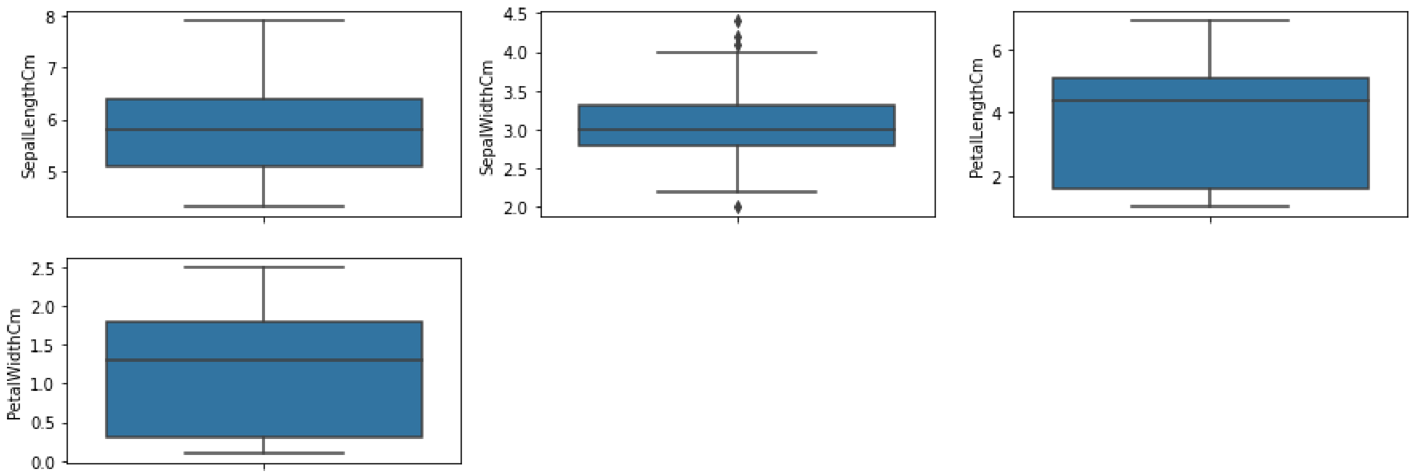
In [10]:
```
#Correlation Betweeen Characteristic
sns.heatmap(data_iris[["SepalLengthCm","SepalWidthCm","PetalLengthCm","PetalWidthCm"]].corr() , annot = True)
plt.show()
```



Outliers Treatment

In [11]:
```
# Checking Outliers
```

```
plt.figure(figsize =(15,8))

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()
```



In [12]:
```
# Soft Capping(SepalWidthCm)
#Percentile value of SepalWidthCm
q1 = data_iris['SepalWidthCm'].quantile(0.01)
q4 = data_iris['SepalWidthCm'].quantile(0.99)

# Capping Lower range outliers
data_iris['SepalWidthCm'][data_iris['SepalWidthCm']<=q1] = q1

# Capping Upper range outliers
data_iris['SepalWidthCm'][data_iris['SepalWidthCm']>=q4] = q4
```

In [13]:
```
#dropping the columns which are not required for the further analysis
data = data_iris.drop(["Id","Species"],axis= 1).values
```

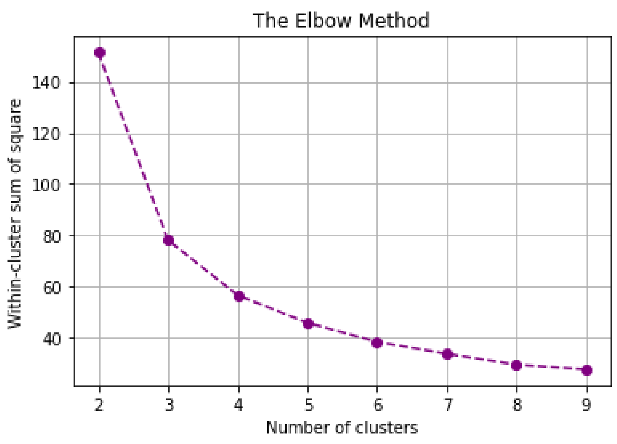
Clustering - [Using K-mean Clustering]

In [14]:
```
#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[14]:0.8238668328229162

In [28]:
```
# 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()
```

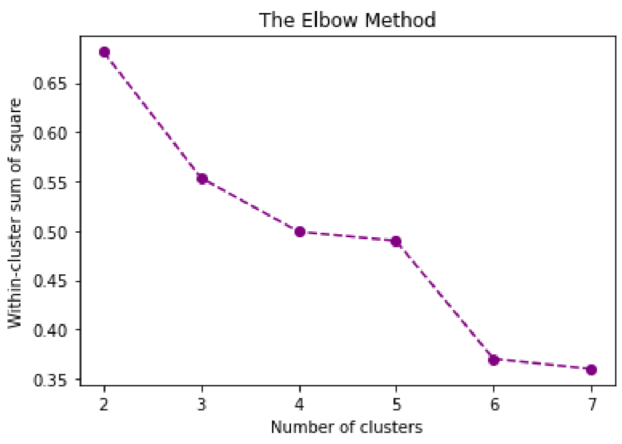


In [16]:
```
#Finding optimal clusters using sheloitte score
#Silhouette score measures that how a data point is similar to it's own cluster as compared to other cluster
range_n = [2,3,4,5,6,7]
ss = []
for i in range_n:
    kmeans = KMeans(n_clusters=i, max_iter = 500)
    kmeans.fit(data)

    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 0.6813827002308916
for n cluster 3 Silhouette score is 0.5535725524920355
for n cluster 4 Silhouette score is 0.4991092247792553
for n cluster 5 Silhouette score is 0.48982358704800855
for n cluster 6 Silhouette score is 0.3702142015220346
for n cluster 7 Silhouette score is 0.35999395076875734
```

In [17]:
```
Silhouette_score_df = pd.DataFrame(ss)
Silhouette_score_df.columns = ["Number of clusters", "Silhouette Score"]
```

In [27]:
```
plt.plot(Silhouette_score_df["Number of clusters"],Silhouette_score_df["Silhouette Score"],'go--', color = 'purple');
plt.title("The Elbow Method")
plt.xlabel("Number of clusters")
plt.ylabel("Within-cluster sum of square")
plt.show()
```



Observation

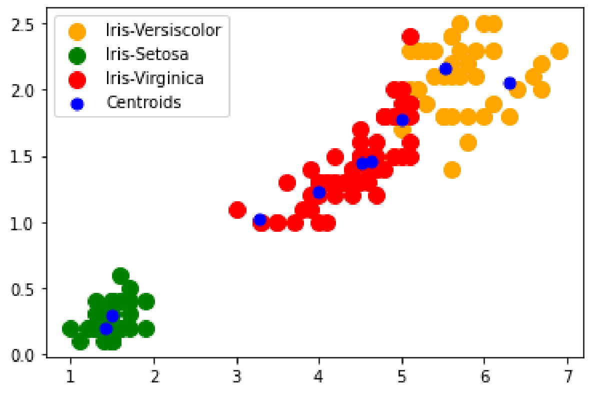
- From elbow curve we can observe that at i = 3 there is a break point and from silhouette score we can observe that for n cluster = 3 we are getting second highest silhouette score value. Therefore, we will take k value as 3 (Never choose Silhouette Score = 2.It is wrong practical implication)

```
In [19]: #Assuming k = 3
         kmeans = KMeans(n_clusters = 3,max_iter= 500 , random_state = 50)
         Cluster_Id= kmeans.fit_predict(data)
```

```
In [20]: Cluster_Id
```

```
Out[20]: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 [30]: #Visualizing Cluster using petal length and petal width column
         plt.scatter(data[Cluster_Id == 0 , 2 ], data[Cluster_Id == 0 , 3] ,s = 100 ,c = "orange", label = "Iris-Versiscolor" )
         plt.scatter(data[Cluster_Id == 1 , 2 ], data[Cluster_Id == 1 , 3] ,s = 100 ,c = "green", label = "Iris-Setosa" )
         plt.scatter(data[Cluster_Id == 2 , 2 ], data[Cluster_Id == 2 , 3] ,s = 100 ,c = "red", label = "Iris-Virginica" )

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

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



Conclusion:

- Therefore, the optimum number of clusters are .

Suggestions are always Welcome!

Thank You !