Practical No.8

Aim:- K-Means Clustering

- Apply the K-Means Algorithm to group similar data points into clusters.
- Determine optimal number of clusters using elbow method or silhouette analysis.
- Visualize the clustering results and analyse the cluster characteristics.

In [7]:
```
#Importing  Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn.cluster as cluster
from sklearn.cluster import KMeans
import seaborn as sns
import sklearn.metrics as metrics
```

In [8]:
```
dataset = pd.read_csv('Iris.csv')
x = dataset.iloc[:,[0,1,2,3]].values
```

In [9]:
```
print(x)
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
```

In [10]:
```
K = range(1,10)
# within-cluster-sum-of-square
wss = []
for k in K:
    kmeans=cluster.KMeans(n_clusters=k,init="k-means++")
    kmeans=kmeans.fit(x)
    wss_iter = kmeans.inertia_
    wss.append(wss_iter)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py
on Windows with MKL, when there are less chunks than available thread
P_NUM_THREADS=1.
  warnings.warn(
```

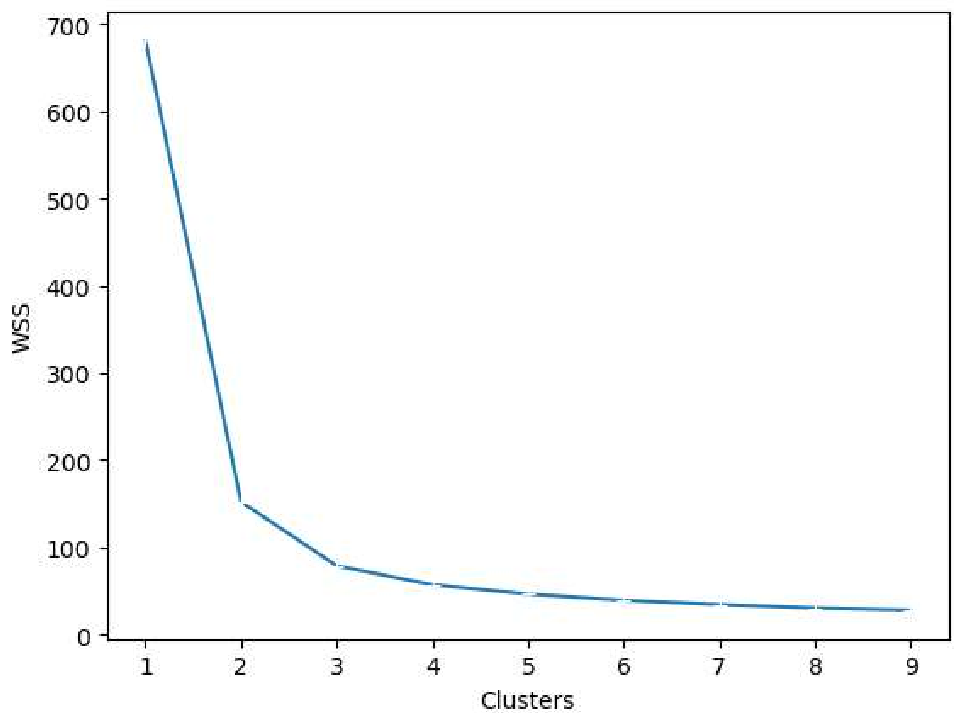
In [11]:
```python
#storing number of clusters along with their WSS in DataFrame
mycenters = pd.DataFrame({'Clusters':K,'WSS':wss})
mycenters
```

Out[11]:

	Clusters	WSS
0	1	681.370600
1	2	152.347952
2	3	78.851441
3	4	57.228473
4	5	46.446182
5	6	39.306107
6	7	34.409010
7	8	30.410173
8	9	27.861429

In [12]:
```python
#Plot Elbow Plot
sns.lineplot(x = 'Clusters',y='WSS',data = mycenters,marker = "+")
```

Out[12]: <AxesSubplot:xlabel='Clusters', ylabel='WSS'>



In [13]:
```
#Answer : 3 Clusters identified From Elbow Plot
```

In [15]:
```
#Silhouette Method to identify clusters.
SK = range(3,10)
sil_score = []
for i in SK:
    labels=cluster.KMeans(n_clusters=i,init="k-means++",random_state=100).fit(x).labels_
    score = metrics.silhouette_score(x,labels,metric="euclidean",sample_size=1000,random_state=100)
    sil_score.append(score)
    print("Silhouette score for k(clusters) = "+ str(i)+" is "
         +str(metrics.silhouette_score(x,labels,metric="euclidean",sample_size=150,random_state=100)))
```

```
Silhouette score for k(clusters) = 3 is 0.5528190123564096
Silhouette score for k(clusters) = 4 is 0.49805050499728737
Silhouette score for k(clusters) = 5 is 0.4887488870931056
Silhouette score for k(clusters) = 6 is 0.3648340039670025
Silhouette score for k(clusters) = 7 is 0.3566882476581695
Silhouette score for k(clusters) = 8 is 0.3471194328049034
Silhouette score for k(clusters) = 9 is 0.32551324341596094
```

In [16]:
```
sil_centers = pd.DataFrame({'Clusters':SK,'Sil Score': sil_score})
sil_centers
```

Out[16]:

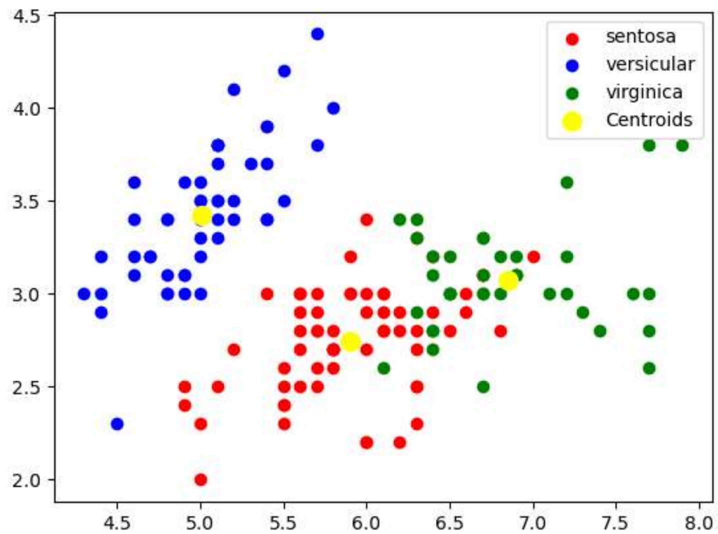
	Clusters	Sil Score
0	3	0.552819
1	4	0.498051
2	5	0.488749
3	6	0.364834
4	7	0.356688
5	8	0.347119
6	9	0.325513

In [17]:
```
#Answer: Max Silhouette Score as k = 3, Hence 3 Clusters is the right option.
```

In [19]:
```
#Perform K-Means Clustering with 3 Clusters.
kmeans = cluster.KMeans(n_clusters=3,init="k-means++")
y_kmeans = kmeans.fit_predict(x)
```

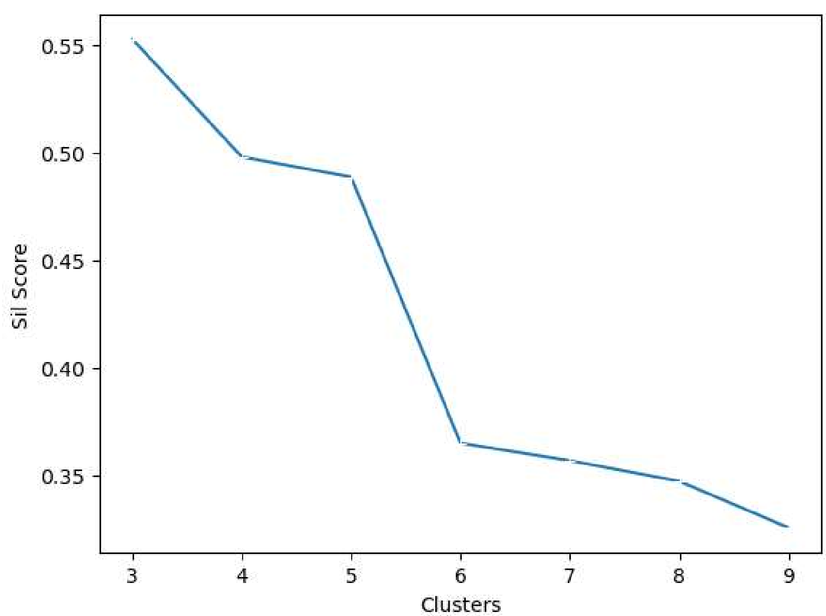
In [22]:
```
#Visulaization of clusters.
plt.scatter(x[y_kmeans == 0,0],x[y_kmeans == 0,1], c = 'red', label = 'sentosa')
plt.scatter(x[y_kmeans == 1,0],x[y_kmeans == 1,1], c = 'blue', label = 'versicular')
plt.scatter(x[y_kmeans == 2,0],x[y_kmeans == 2,1], c = 'green', label = 'virginica')

#PLotting the centroids of the clusters
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],s=100,c='yellow',label='Centroids')
plt.legend()
```

Out[22]: <matplotlib.legend.Legend at 0x28369e00c40>



In [23]:
```
sns.lineplot(x='Clusters',y='Sil Score',data = sil_centers,marker="+")
```

Out[23]: <AxesSubplot:xlabel='Clusters', ylabel='Sil Score'>



In [24]:
```
#Answer : Max Silhouette Score as k = 3, Hence 3 Clusters is the right option.
```