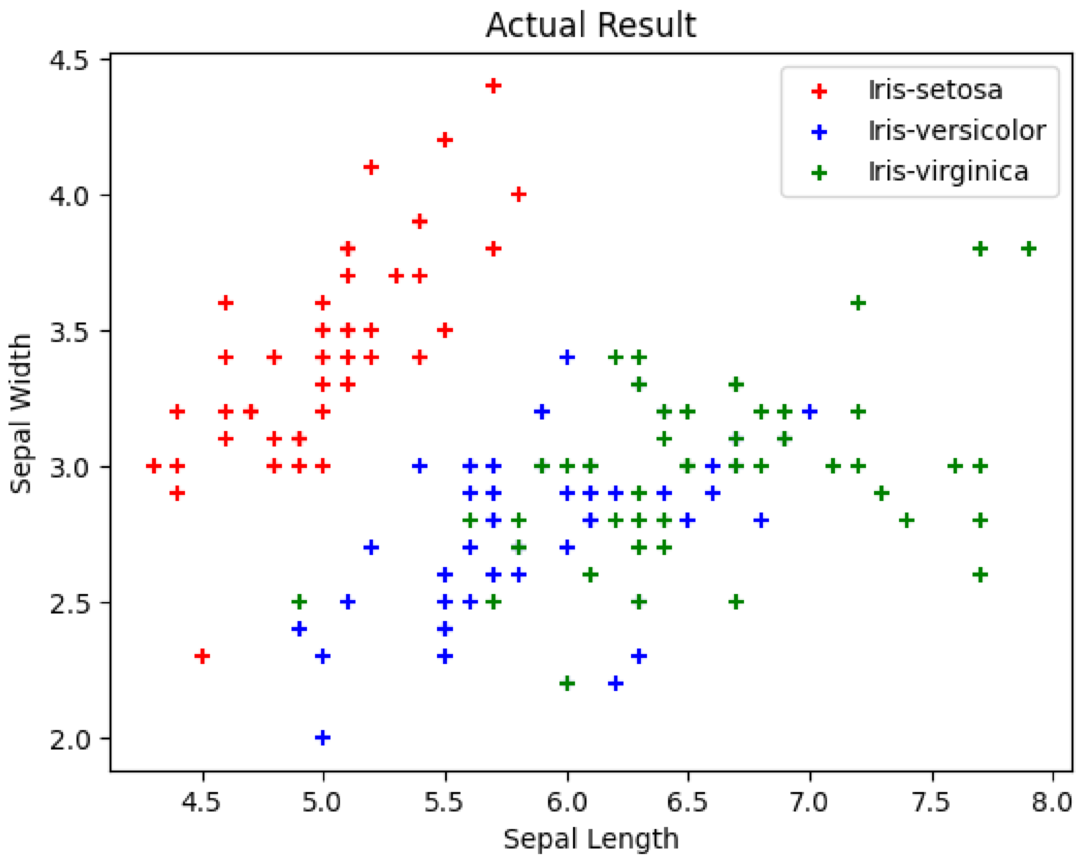
# ML Assignment 4: Iris Dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import silhouette_score, calinski_harabasz_score,
davies_bouldin_score

!gdown 17bW5DYVUVShliNePIuBfawk_y7F0q07F    # iris dataset
```

## Clustering in Iris Dataset

```
df_iris = pd.read_csv("iris.csv", names=['sepal_length',
'sepal_width', 'petal_length', 'petal_width', 'species'])

X = df_iris.drop('species', axis=1)
y = df_iris.species

# Actual Clustering Result
newDf0 = df_iris[df_iris.species=="Iris-setosa"]
newDf1 = df_iris[df_iris.species=="Iris-versicolor"]
newDf2 = df_iris[df_iris.species=="Iris-virginica"]

plt.title("Actual Result")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.scatter(newDf0.sepal_length, newDf0.sepal_width, color="red",
marker="+", label="Iris-setosa")
plt.scatter(newDf1.sepal_length, newDf1.sepal_width, color="blue",
marker="+", label="Iris-versicolor")
plt.scatter(newDf2.sepal_length, newDf2.sepal_width, color="green",
marker="+", label="Iris-virginica")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f3c26c6d3f0>
```

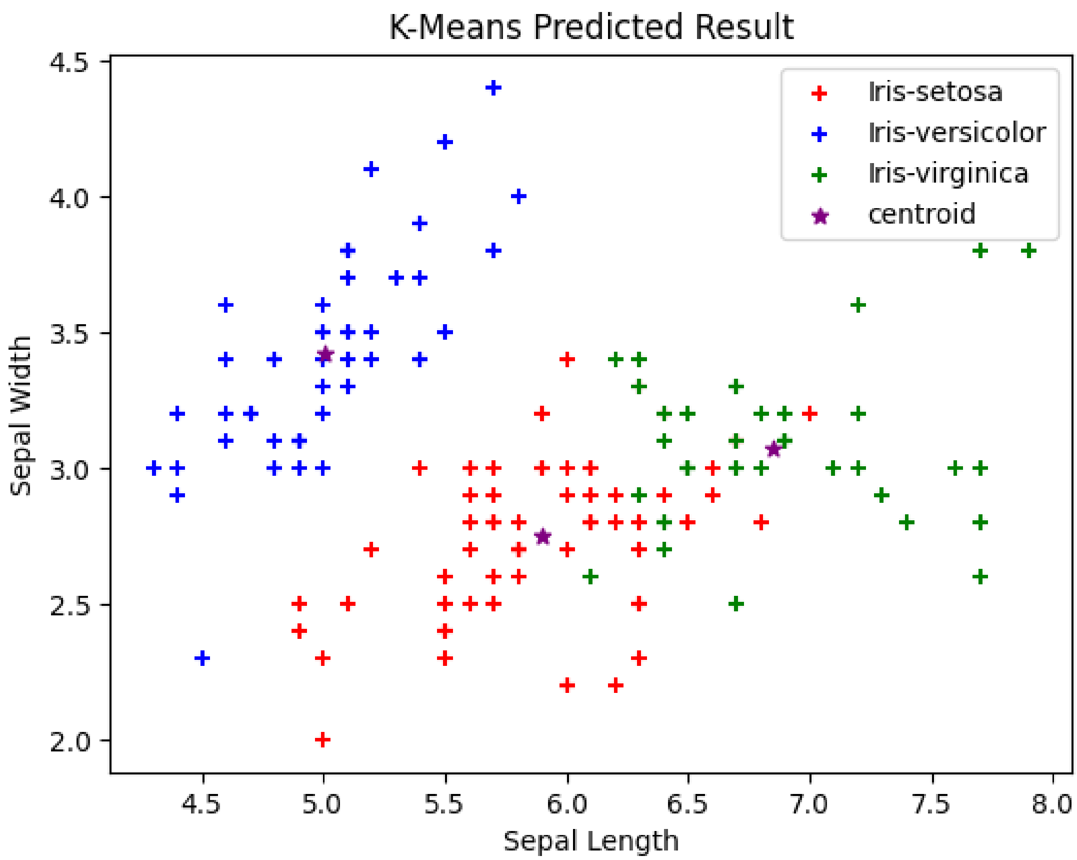



## Partition Based: K-means Clustering in Iris Dataset

```
# Clustering using K-means algorithm
from sklearn.cluster import KMeans
km = KMeans(n_clusters=3, n_init=10)

y_predicted = km.fit_predict(X)

newDf = df_iris
newDf["cluster"] = y_predicted
newDf0 = newDf[newDf.cluster==0]
newDf1 = newDf[newDf.cluster==1]
newDf2 = newDf[newDf.cluster==2]

plt.title("K-Means Predicted Result")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.scatter(newDf0.sepal_length, newDf0.sepal_width, color="red",
marker="+", label="Iris-setosa")
plt.scatter(newDf1.sepal_length, newDf1.sepal_width, color="blue",
marker="+", label="Iris-versicolor")
plt.scatter(newDf2.sepal_length, newDf2.sepal_width, color="green",
marker="+", label="Iris-virginica")
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1],
```

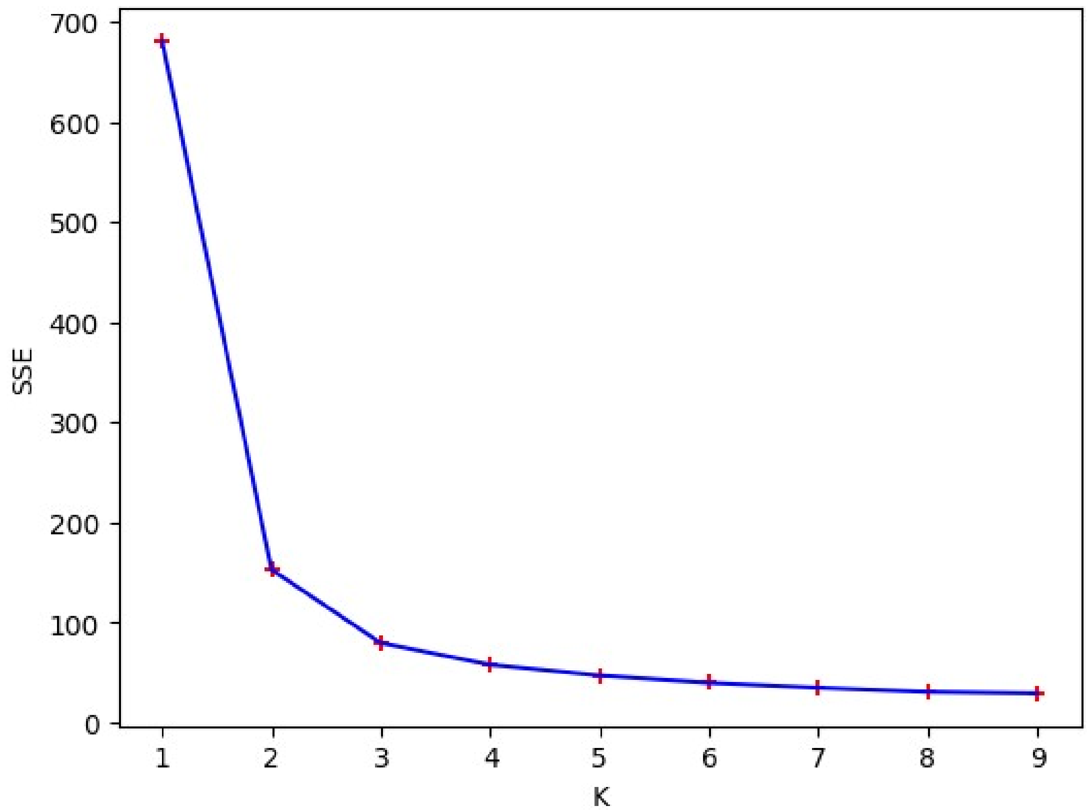
```
color="purple", marker="*", label="centroid")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f3c26651bd0>
```




```
# Visualisation of SSE (Sum of Squared Errors) & Elbow Graph:
sse = []
k_range = range(1, 10)

for k in k_range:
    km = KMeans(n_clusters=k, n_init=10)
    km.fit_predict(X)
    sse.append(km.inertia_)

plt.xlabel("K")
plt.ylabel("SSE")
plt.scatter(k_range, sse, color="red", marker="+")
plt.plot(k_range, sse, color="blue")
# We can see here, our elbow is at K=3
```

```
[<matplotlib.lines.Line2D at 0x7f3c26248a90>]
```




```python
# Evaluating Metrics
silhouette_result = silhouette_score(X, km.labels_)
print("Silhouette Score: ", silhouette_result)

calinski_result = calinski_harabasz_score(X, km.labels_)
print("Calinski Harabasz Score: ", calinski_result)

davies_result = davies_bouldin_score(X, km.labels_)
print("Davies Bouldin Score: ", davies_result)

# Evaluating Cohesion & Separation
labels = km.labels_
centroids = km.cluster_centers_
SSE = np.sum((X - centroids[labels])**2)
overall_centroid = np.mean(X, axis=0)

SSB = np.sum([np.sum((X[labels == i] - centroids[i])**2) for i in
range(3)])

N = X.shape[0]
cohesion_scores = SSE/N
cohesion = np.mean(cohesion_scores)
separation = SSB/N
```

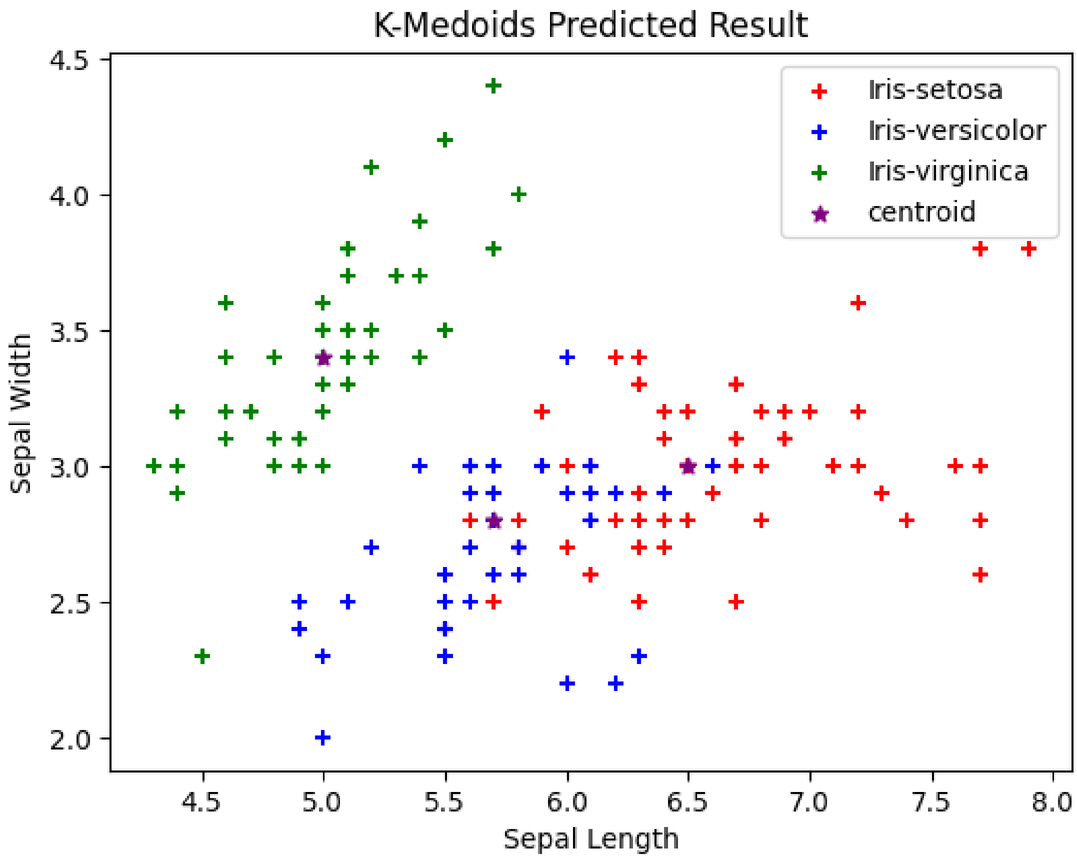
```
print(f"\nCohesion Score: {cohesion}")
print(f"Separation Score: {separation}")
```

```
Silhouette Score:  0.34597762034129553
Calinski Harabasz Score:  401.8511977911363
Davies Bouldin Score:  1.0253347541939601

Cohesion Score: 0.04767664257307373
Separation Score: 0.08704928634167765
```

### Partition Based: K-medoids Clustering in Iris Dataset

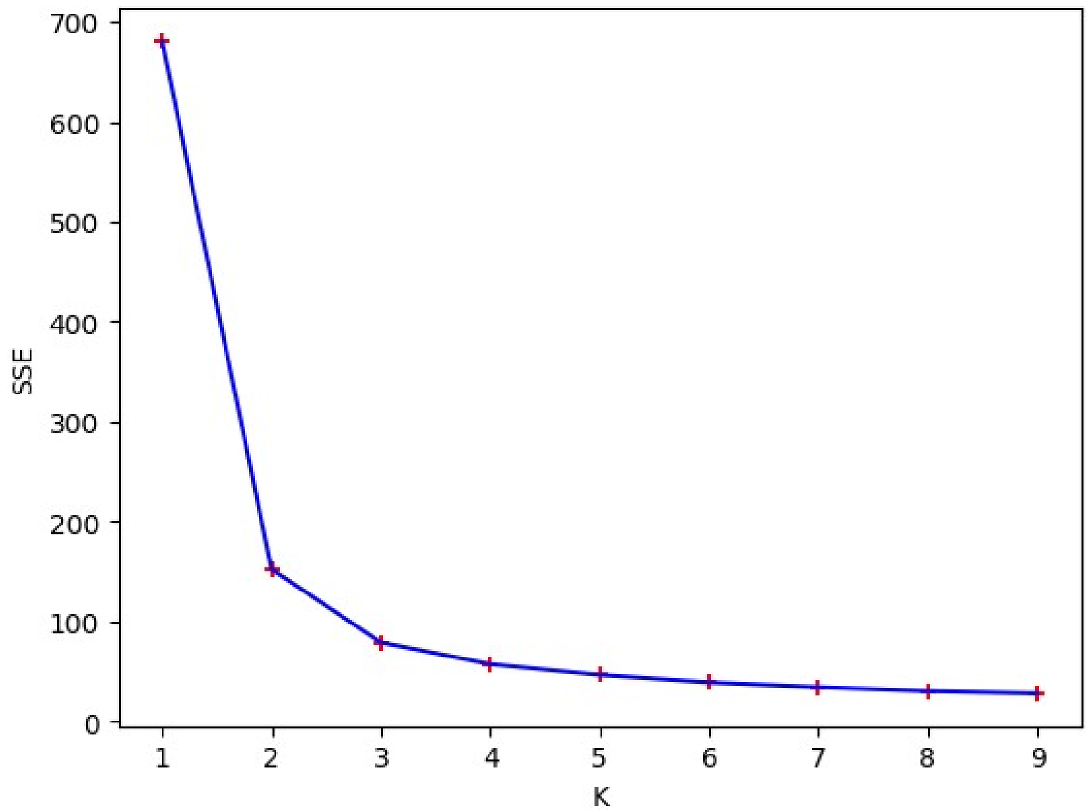
```
!pip install scikit-learn-extra

# Clustering using K-medoids algorithm
from sklearn_extra.cluster import KMedoids
km = KMedoids(n_clusters=3)

y_predicted = km.fit_predict(X)

newDf = df_iris
newDf["cluster"] = y_predicted
newDf0 = newDf[newDf.cluster==0]
newDf1 = newDf[newDf.cluster==1]
newDf2 = newDf[newDf.cluster==2]

plt.title("K-Medoids Predicted Result")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.scatter(newDf0.sepal_length, newDf0.sepal_width, color="red",
marker="+", label="Iris-setosa")
plt.scatter(newDf1.sepal_length, newDf1.sepal_width, color="blue",
marker="+", label="Iris-versicolor")
plt.scatter(newDf2.sepal_length, newDf2.sepal_width, color="green",
marker="+", label="Iris-virginica")
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1],
color="purple", marker="*", label="centroid")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f3c262b73d0>
```




```python
# Visualisation of SSE (Sum of Squared Errors) & Elbow Graph:
sse = []
k_range = range(1, 10)

for k in k_range:
    km = KMeans(n_clusters=k, n_init=10)
    km.fit_predict(X)
    sse.append(km.inertia_)

plt.xlabel("K")
plt.ylabel("SSE")
plt.scatter(k_range, sse, color="red", marker="+")
plt.plot(k_range, sse, color="blue")
# We can see here, our elbow is at K=3
```

```
[<matplotlib.lines.Line2D at 0x7f3c261579a0>]
```



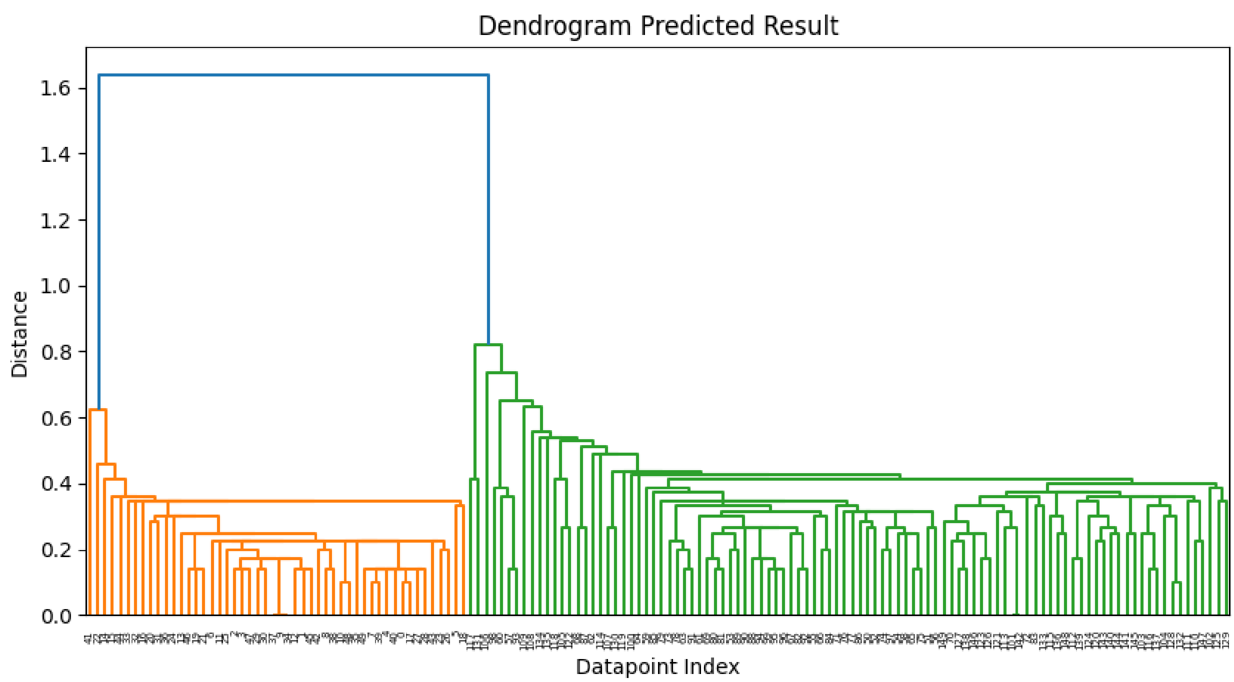

```
# Evaluating Metrics
silhouette_result = silhouette_score(X, km.labels_)
print("Silhouette Score: ", silhouette_result)

calinski_result = calinski_harabasz_score(X, km.labels_)
print("Calinski Harabasz Score: ", calinski_result)

davies_result = davies_bouldin_score(X, km.labels_)
print("Davies Bouldin Score: ", davies_result)

# Evaluating Cohesion & Separation
labels = km.labels_
centroids = km.cluster_centers_
SSE = np.sum((X - centroids[labels])**2)
overall_centroid = np.mean(X, axis=0)

SSB = np.sum([np.sum((X[labels == i] - centroids[i])**2) for i in
range(3)])

N = X.shape[0]
cohesion_scores = SSE/N
cohesion = np.mean(cohesion_scores)
separation = SSB/N
```

```
print(f"\nCohesion Score: {cohesion}")
print(f"Separation Score: {separation}")
```

```
Silhouette Score:  0.3435514923440904
Calinski Harabasz Score:  411.2774030706613
Davies Bouldin Score:  0.973553876855317

Cohesion Score: 0.04662882885901002
Separation Score: 0.06856001899335233
```

### Hierarchical: Dendrogram Clustering in Iris Dataset

```
# Clustering using Dendrogram Clustering algorithm
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
Z = linkage(X, method='single')

# Create and plot the dendrogram
plt.figure(figsize=(10, 5))
dn = dendrogram(Z)

plt.title('Dendrogram Predicted Result')
plt.xlabel('Datapoint Index')
plt.ylabel('Distance')
plt.show()
```



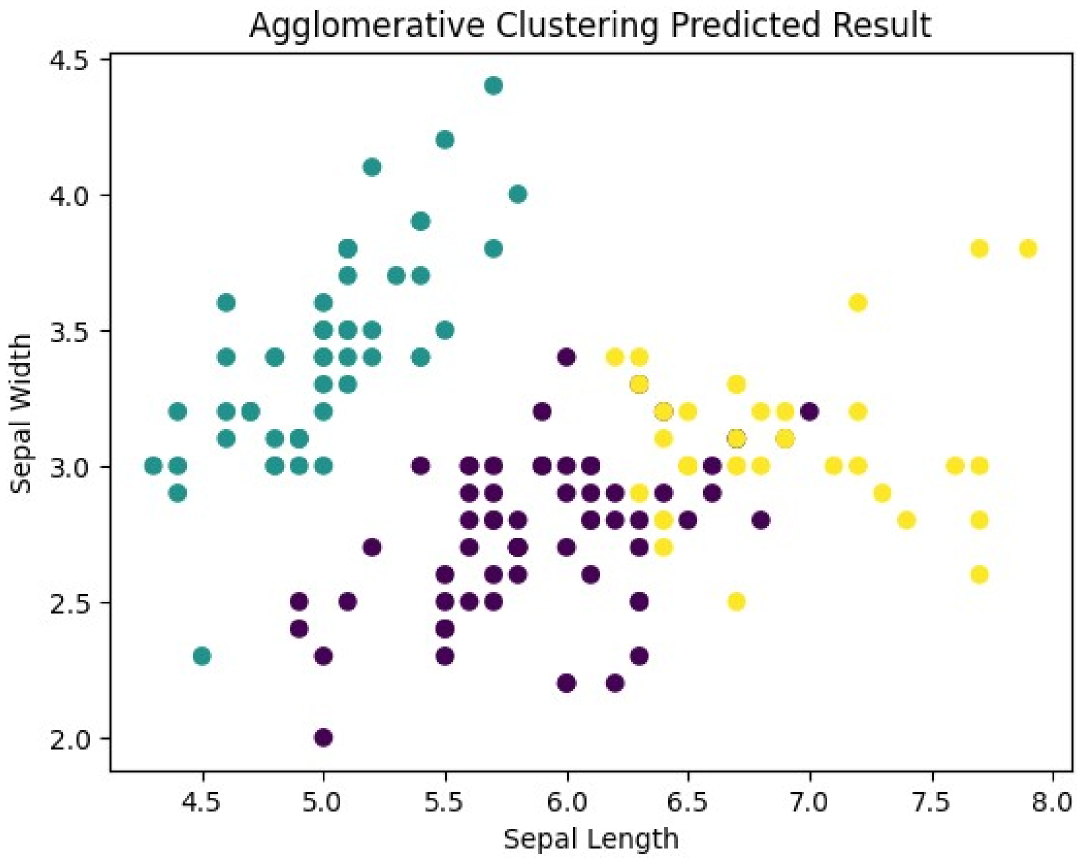

```
# Evaluating Metrics
labels = fcluster(Z, 3, criterion='maxclust')
```

```
silhouette_result = silhouette_score(X, labels)
print("Silhouette Score: ", silhouette_result)

calinski_result = calinski_harabasz_score(X, labels)
print("Calinski Harabasz Score: ", calinski_result)

davies_result = davies_bouldin_score(X, labels)
print("Davies Bouldin Score: ", davies_result)
```

```
Silhouette Score:  0.5118387098922373
Calinski Harabasz Score:  277.4926776474616
Davies Bouldin Score:  0.4474384341989626
```

## Hierarchical: AGNES Clustering in Iris Dataset

```
# Clustering using AGNES Clustering algorithm
from sklearn.cluster import AgglomerativeClustering

agg_cluster = AgglomerativeClustering(n_clusters=3, linkage='ward')
agg_cluster.fit(X)

plt.scatter(df_iris.sepal_length, df_iris.sepal_width,
c=agg_cluster.labels_, cmap='viridis')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('Agglomerative Clustering Predicted Result')
plt.show()
```



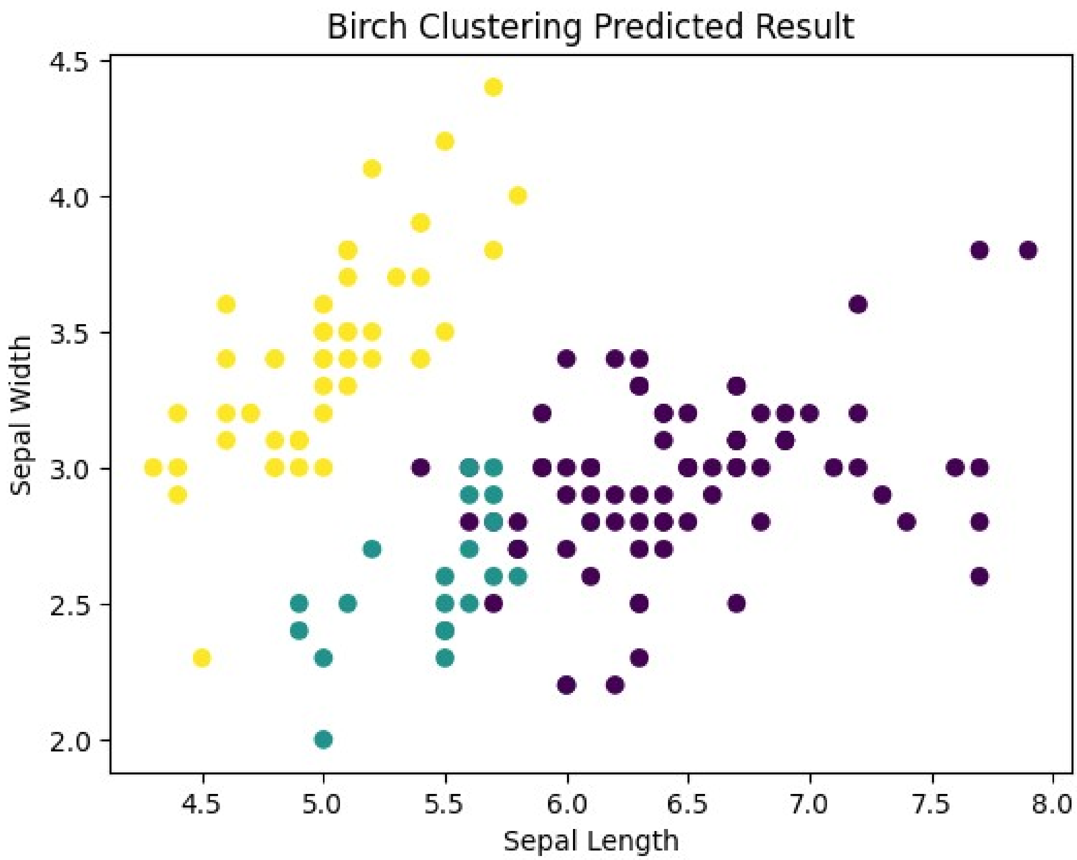

```
# Evaluating Metrics
labels = fcluster(Z, 3, criterion='maxclust')

from sklearn.metrics import silhouette_score
silhouette_result = silhouette_score(X, labels)
print("Silhouette Score: ", silhouette_result)

from sklearn.metrics import calinski_harabasz_score
calinski_result = calinski_harabasz_score(X, labels)
print("Calinski Harabasz Score: ", calinski_result)

from sklearn.metrics import davies_bouldin_score
davies_result = davies_bouldin_score(X, labels)
print("Davies Bouldin Score: ", davies_result)
```

```
Silhouette Score:  0.5118387098922373
Calinski Harabasz Score:  277.4926776474616
Davies Bouldin Score:  0.4474384341989626
```

## Hierarchical: BIRCH Clustering in Iris Dataset

```
# Clustering using BIRCH Clustering algorithm
from sklearn.cluster import Birch
```

```python
birch_cluster = Birch(n_clusters=3)
birch_cluster.fit(X)

plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df_iris.sepal_length, df_iris.sepal_width,
c=birch_cluster.labels_, cmap='viridis')
plt.title('Birch Clustering Predicted Result')
plt.show()
```

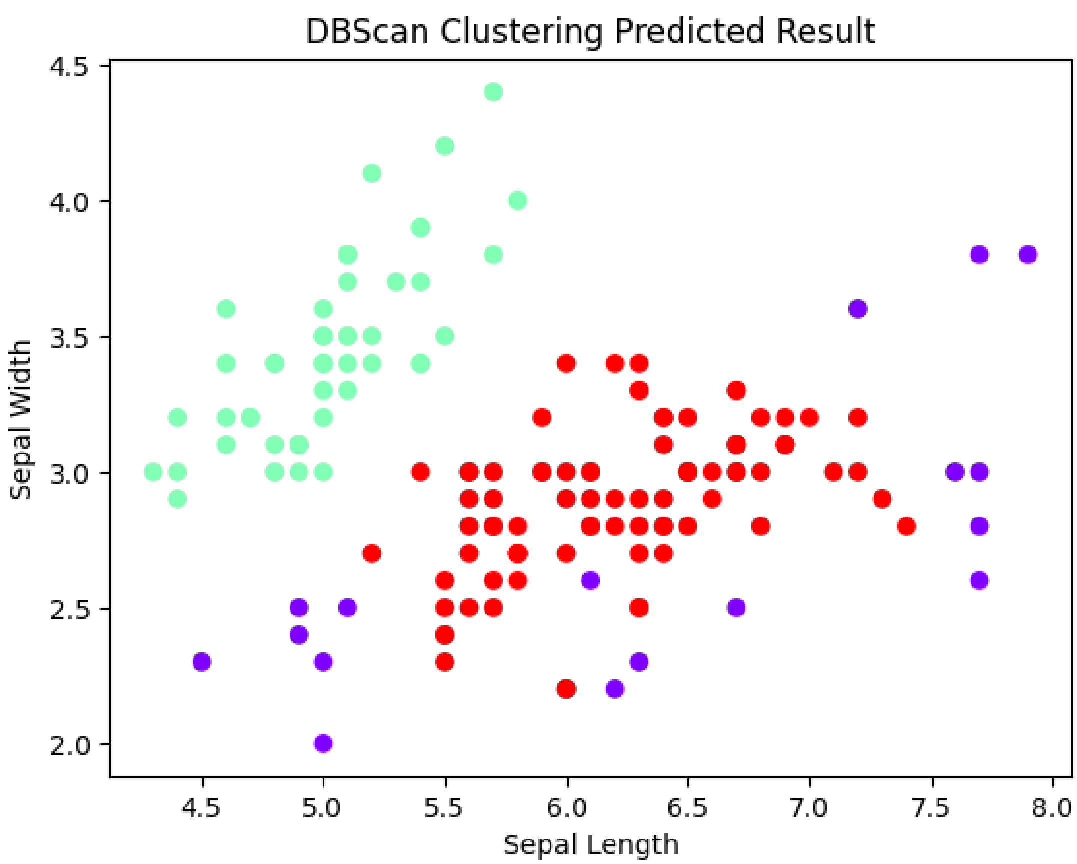



```python
# Evaluating Metrics
labels = birch_cluster.fit_predict(X)

silhouette_result = silhouette_score(X, labels)
print("Silhouette Score: ", silhouette_result)

calinski_result = calinski_harabasz_score(X, labels)
print("Calinski Harabasz Score: ", calinski_result)

davies_result = davies_bouldin_score(X, labels)
print("Davies Bouldin Score: ", davies_result)
```

```
Silhouette Score:  0.5016992571068448
Calinski Harabasz Score:  457.54177598067594
Davies Bouldin Score:  0.6262973013286385
```

### Density Based: DBSCAN Clustering in Iris Dataset

```
# Clustering using DBSCAN Clustering algorithm
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.5, algorithm='auto', metric='euclidean')
y = dbscan.fit_predict(X)

plt.scatter(df_iris.sepal_length, df_iris.sepal_width,
c=dbscan.labels_, cmap='rainbow')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('DBScan Clustering Predicted Result')
plt.show()
```



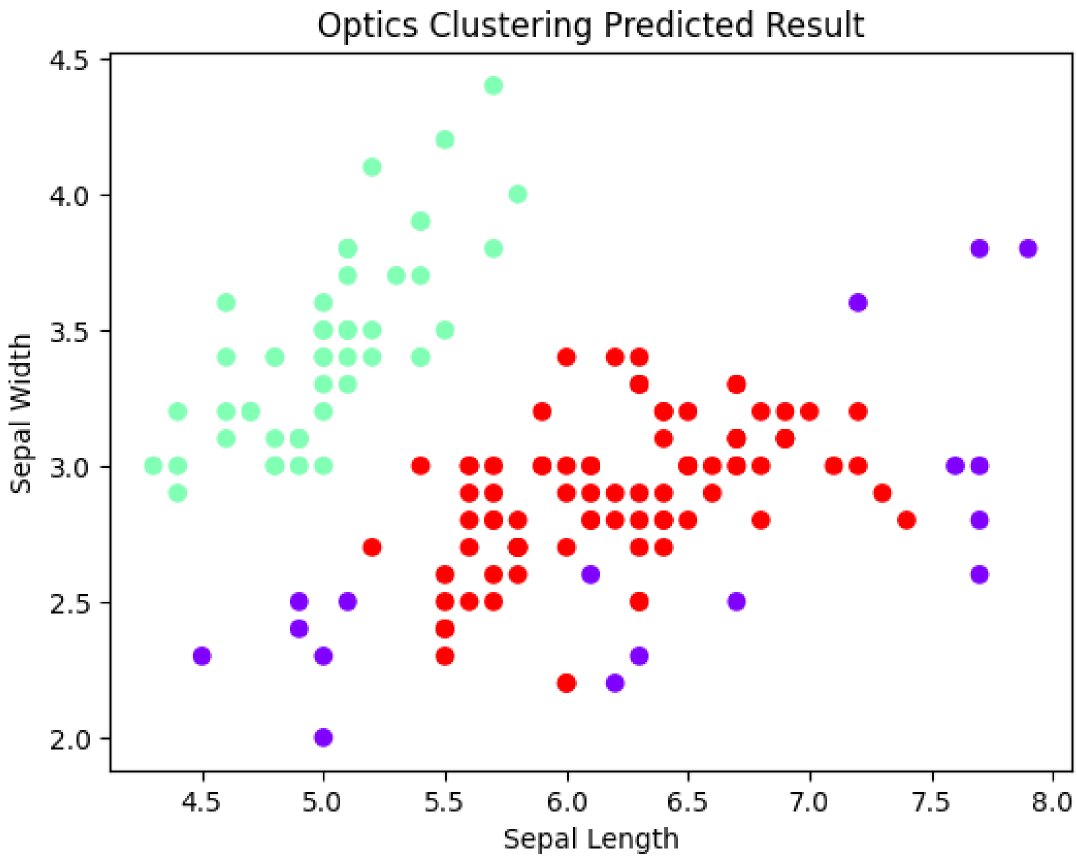

```
# Evaluating Metrics
silhouette_result = silhouette_score(X, dbscan.labels_)
print("Silhouette Score: ", silhouette_result)
```

```
calinski_result = calinski_harabasz_score(X, dbscan.labels_)
print("Calinski Harabasz Score: ", calinski_result)

davies_result = davies_bouldin_score(X, dbscan.labels_)
print("Davies Bouldin Score: ", davies_result)
```

```
Silhouette Score:  0.485842354600955
Calinski Harabasz Score:  219.87022703461665
Davies Bouldin Score:  7.222826995273629
```

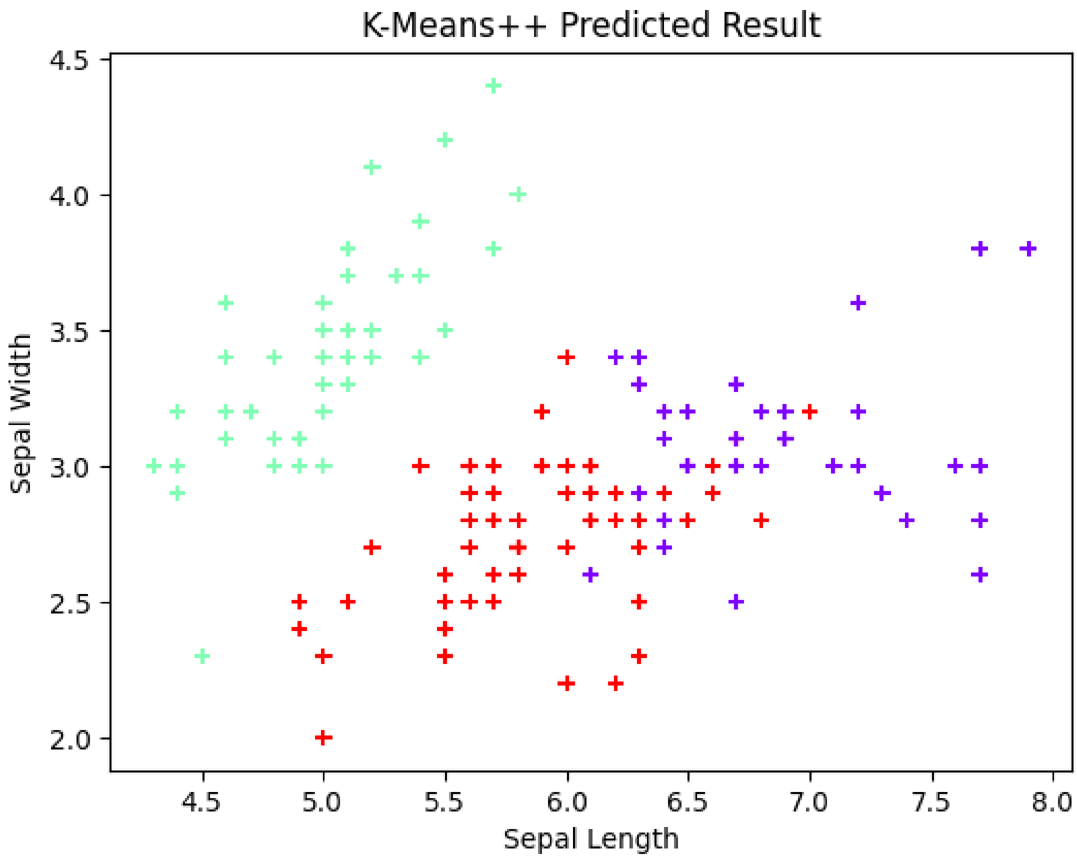
## Density Based: Optics Clustering in Iris Dataset

```
# Clustering using Optics Clustering algorithm
from sklearn.cluster import OPTICS

optics_cluster = OPTICS(min_samples=5, xi=0.05,
cluster_method='dbscan')
optics_cluster.fit(X)

plt.scatter(df_iris.sepal_length, df_iris.sepal_width,
c=dbscan.labels_, cmap='rainbow')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('Optics Clustering Predicted Result')
plt.show()
```

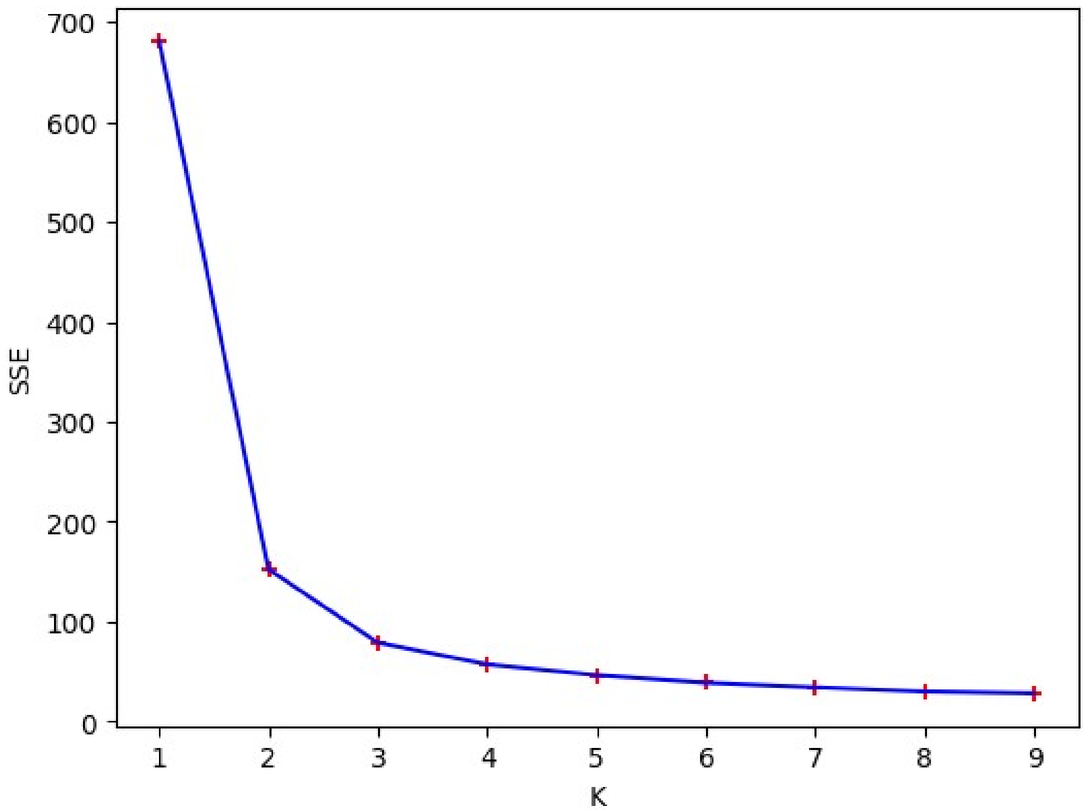



## K-means++ Clustering in Iris Dataset

```
# Clustering using K-means++ algorithm
from sklearn.cluster import KMeans
km = KMeans(init='k-means++', n_clusters=3, n_init=10, max_iter=300,
random_state=42)
km = KMeans(n_clusters=3, n_init=10)

y_predicted = km.fit_predict(X)

plt.title("K-Means++ Predicted Result")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.scatter(df_iris.sepal_length, df_iris.sepal_width, c=km.labels_,
cmap='rainbow', marker="+")
plt.show()
```




```
# Visualisation of SSE (Sum of Squared Errors) & Elbow Graph:
sse = []
k_range = range(1, 10)

for k in k_range:
    km = KMeans(n_clusters=k, n_init=10)
    km.fit_predict(X)
    sse.append(km.inertia_)

plt.xlabel("K")
plt.ylabel("SSE")
plt.scatter(k_range, sse, color="red", marker="+")
plt.plot(k_range, sse, color="blue")
# We can see here, our elbow is at K=3

[<matplotlib.lines.Line2D at 0x7f3c25f7cd60>]
```



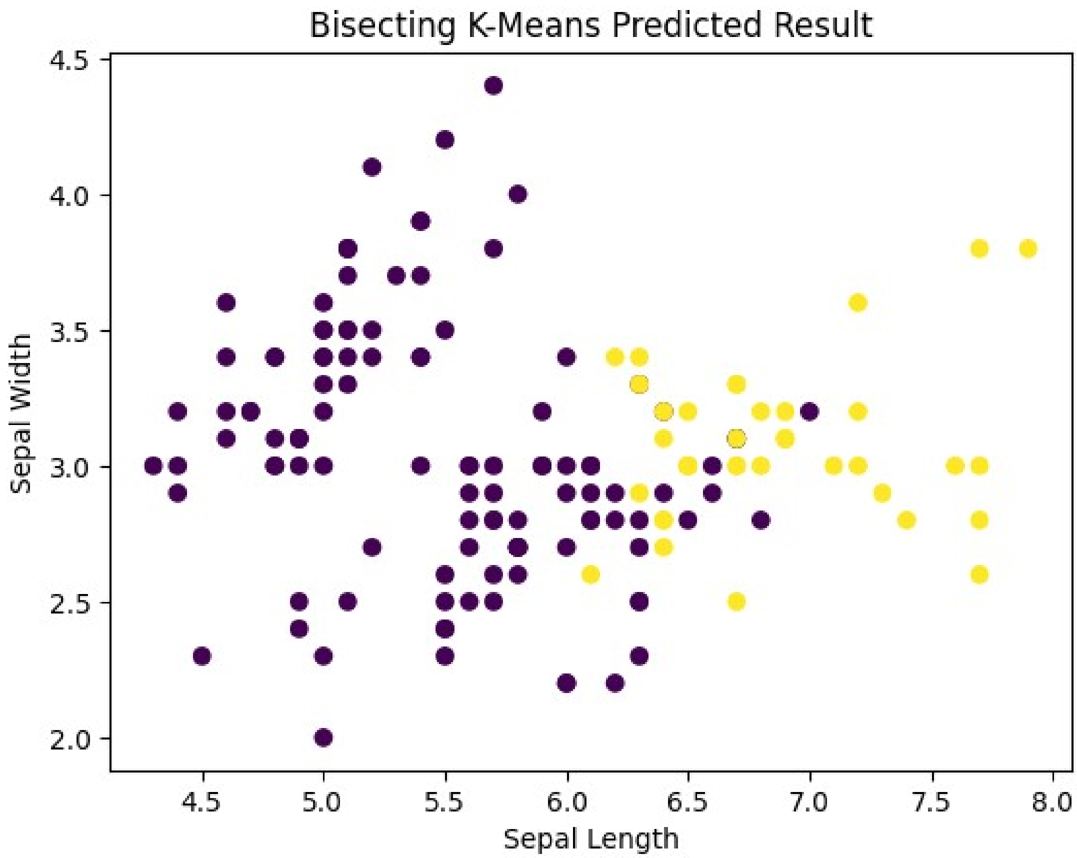

```
# Evaluating Metrics
silhouette_result = silhouette_score(X, km.labels_)
print("Silhouette Score: ", silhouette_result)

calinski_result = calinski_harabasz_score(X, km.labels_)
print("Calinski Harabasz Score: ", calinski_result)

davies_result = davies_bouldin_score(X, km.labels_)
print("Davies Bouldin Score: ", davies_result)

# Evaluating Cohesion & Separation
labels = km.labels_
centroids = km.cluster_centers_
SSE = np.sum((X - centroids[labels])**2)
overall_centroid = np.mean(X, axis=0)

SSB = np.sum([np.sum((X[labels == i] - centroids[i])**2) for i in
range(3)])

N = X.shape[0]
cohesion_scores = SSE/N
cohesion = np.mean(cohesion_scores)
separation = SSB/N
```

```
print(f"\nCohesion Score: {cohesion}")
print(f"Separation Score: {separation}")
```

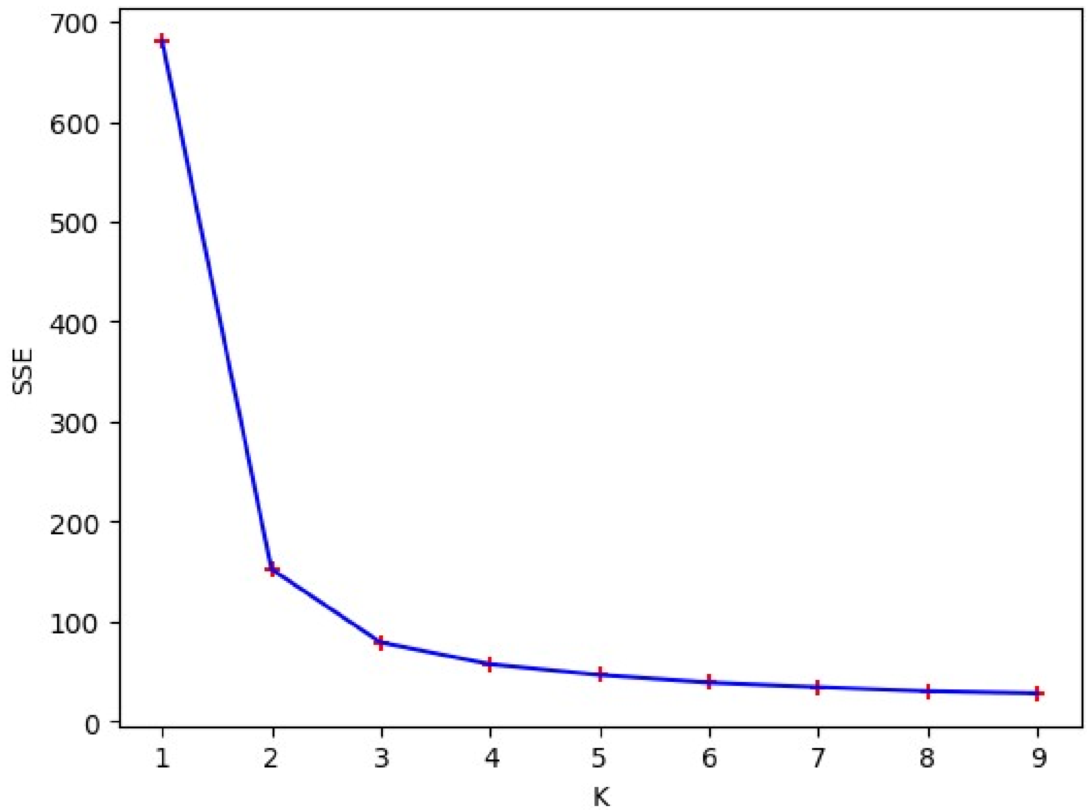
```
Silhouette Score:  0.35859288904946124
Calinski Harabasz Score:  407.19542232102003
Davies Bouldin Score:  0.958208320286649

Cohesion Score: 0.04707687224812225
Separation Score: 0.09227952602952604
```

## Bisecting K-means Clustering in Iris Dataset

```
# Clustering using Bisecting K-means algorithm
from sklearn.cluster import KMeans
km = KMeans(n_clusters=1, n_init=10, random_state=0).fit(X)

K=3
for i in range(K-1):
    largest_cluster = np.argmax(np.bincount(km.labels_))
    largest_cluster_mask = (km.labels_ == largest_cluster)
    X_split = X[largest_cluster_mask]
    km.labels_[largest_cluster_mask] = KMeans(n_clusters=2, n_init=10,
random_state=0).fit(X_split).labels_


plt.title("Bisecting K-Means Predicted Result")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.scatter(df_iris.sepal_length, df_iris.sepal_width, c=km.labels_,
cmap='viridis')
plt.show()
```




```
# Visualisation of SSE (Sum of Squared Errors) & Elbow Graph:
sse = []
k_range = range(1, 10)

for k in k_range:
    km = KMeans(n_clusters=k, n_init=10)
    km.fit_predict(X)
    sse.append(km.inertia_)

plt.xlabel("K")
plt.ylabel("SSE")
plt.scatter(k_range, sse, color="red", marker="+")
plt.plot(k_range, sse, color="blue")
# We can see here, our elbow is at K=3

[<matplotlib.lines.Line2D at 0x7f3c2665c7c0>]
```




```
# Evaluating Metrics
silhouette_result = silhouette_score(X, km.labels_)
print("Silhouette Score: ", silhouette_result)

calinski_result = calinski_harabasz_score(X, km.labels_)
print("Calinski Harabasz Score: ", calinski_result)

davies_result = davies_bouldin_score(X, km.labels_)
print("Davies Bouldin Score: ", davies_result)

# Evaluating Cohesion & Separation
labels = km.labels_
centroids = km.cluster_centers_
SSE = np.sum((X - centroids[labels])**2)
overall_centroid = np.mean(X, axis=0)

SSB = np.sum([np.sum((X[labels == i] - centroids[i])**2) for i in
range(3)])

N = X.shape[0]
cohesion_scores = SSE/N
cohesion = np.mean(cohesion_scores)
separation = SSB/N
```

```
print(f"\nCohesion Score: {cohesion}")
print(f"Separation Score: {separation}")
```

```
Silhouette Score:  0.34093973210740597
Calinski Harabasz Score:  409.94078898037156
Davies Bouldin Score:  1.0060550389051572

Cohesion Score: 0.046774595314776485
Separation Score: 0.06472950617283951
```

## ML Assignment 4: Wine Dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import silhouette_score, calinski_harabasz_score,
davies_bouldin_score
```

```
!gdown 1w57c07HLl33Dcwxae_j41Gm8SiqJYpih    # wine dataset
```

```
df = pd.read_csv("wine.csv",
names=['class',"Alcohol","Malicacid","Ash","Alcalinity_of_ash","Magnes
ium","Total_phenols","Flavanoids","Nonflavanoid_phenols","Proanthocyan
ins","Color_intensity","Hue","0D280_0D315_of_diluted_wines","Proline"]
)
X = df.drop('class',axis=1)
y = df["class"]
df.head()
```

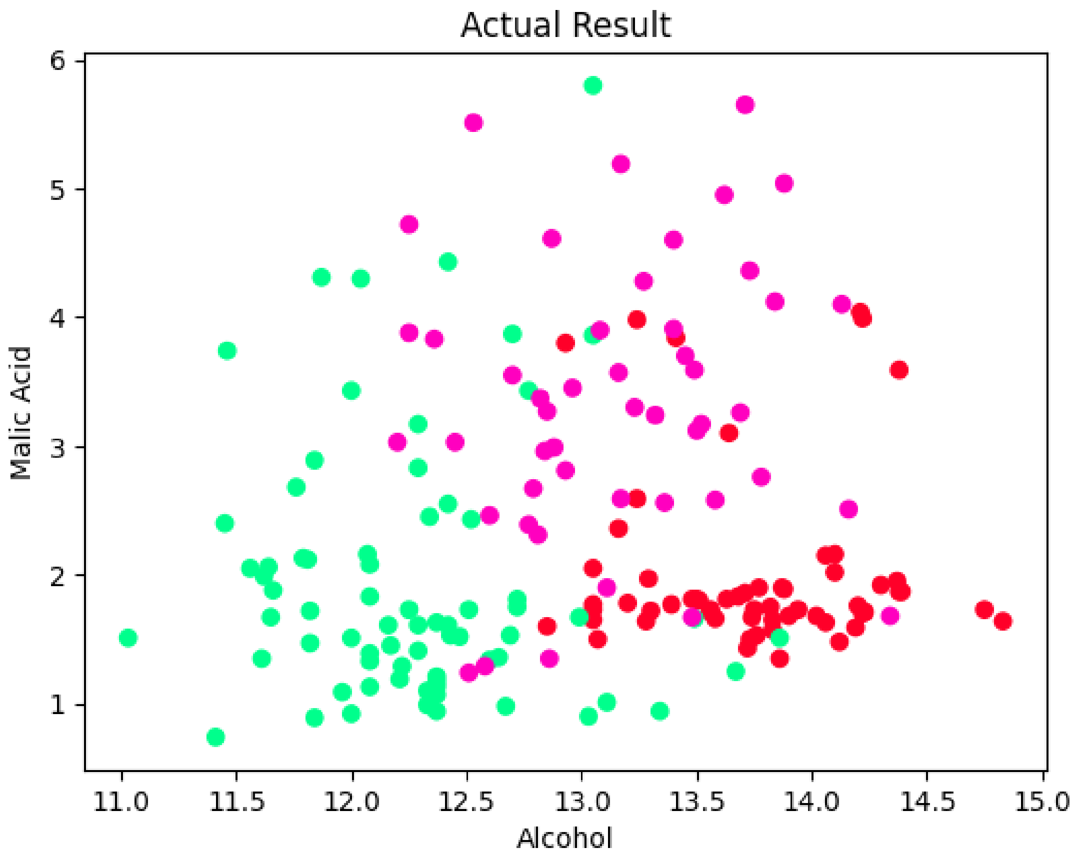
| | class | Alcohol | Malicacid | Ash | Alcalinity_of_ash | Magnesium |
|---|---|---|---|---|---|---|
| 0 | 1 | 14.23 | 1.71 | 2.43 | 15.6 | 127 |
| 1 | 1 | 13.20 | 1.78 | 2.14 | 11.2 | 100 |
| 2 | 1 | 13.16 | 2.36 | 2.67 | 18.6 | 101 |
| 3 | 1 | 14.37 | 1.95 | 2.50 | 16.8 | 113 |
| 4 | 1 | 13.24 | 2.59 | 2.87 | 21.0 | 118 |

| | Total_phenols | Flavanoids | Nonflavanoid_phenols | Proanthocyanins |
|---|---|---|---|---|
| 0 | 2.80 | 3.06 | 0.28 | 2.29 |
| 1 | 2.65 | 2.76 | 0.26 | 1.28 |
| 2 | 2.80 | 3.24 | 0.30 | 2.81 |
| 3 | 3.85 | 3.49 | 0.24 | 2.18 |
| 4 | 2.80 | 2.69 | 0.39 | 1.82 |

| | Color_intensity | Hue | 0D280_0D315_of_diluted_wines | Proline |
|---|---|---|---|---|
| 0 | 5.64 | 1.04 | 3.92 | 1065 |
| 1 | 4.38 | 1.05 | 3.40 | 1050 |
| 2 | 5.68 | 1.03 | 3.17 | 1185 |
| 3 | 7.80 | 0.86 | 3.45 | 1480 |
| 4 | 4.32 | 1.04 | 2.93 | 735 |

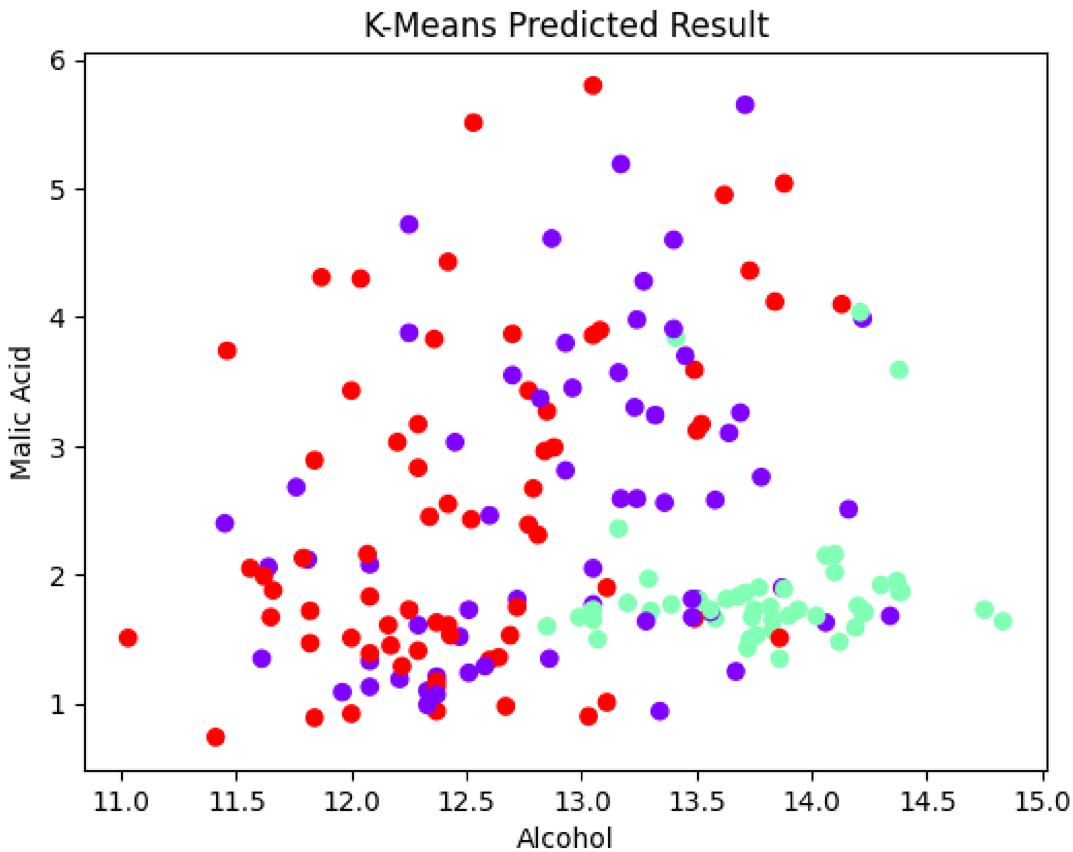
```
# Actual Clustering Result
plt.title("Actual Result")
plt.xlabel('Alcohol')
plt.ylabel('Malic Acid')
plt.scatter(df.Alcohol, df.Malicacid, c=df["class"],
cmap='gist_rainbow')
```

```
<matplotlib.collections.PathCollection at 0x7a82044b9960>
```

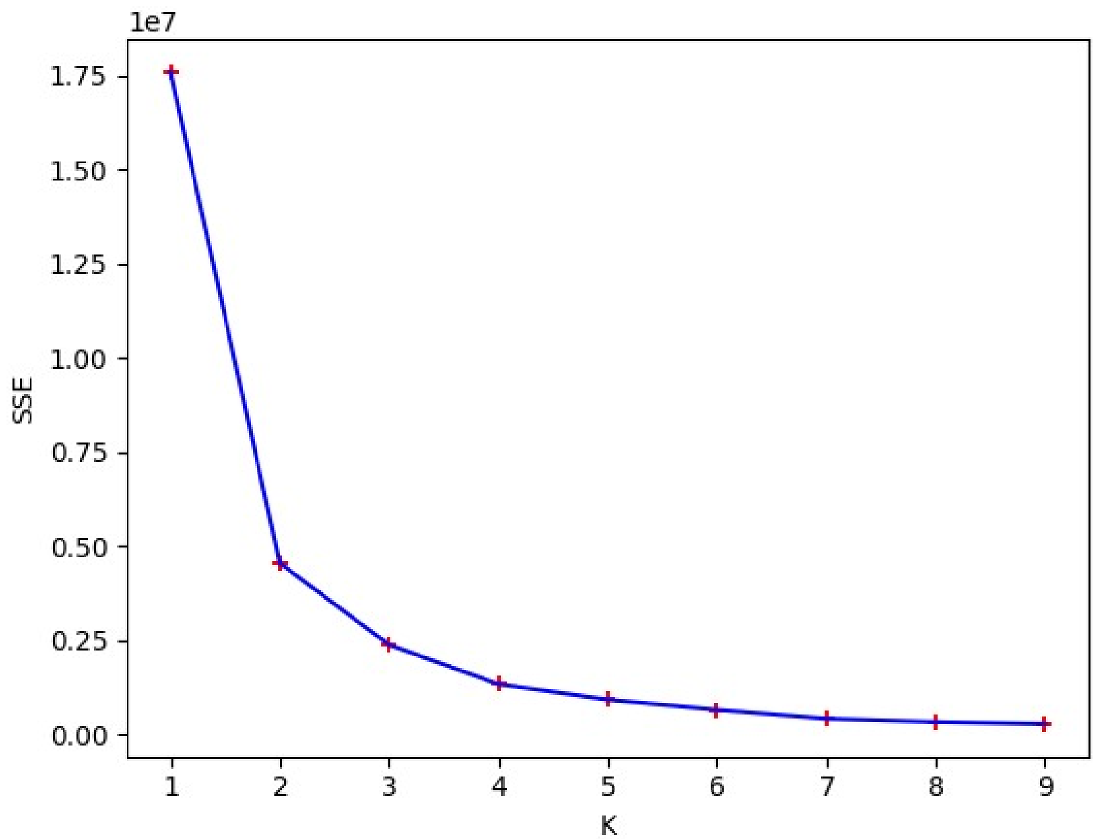



### Partition Based: K-means Clustering in Wine Dataset

```
# Clustering using K-means algorithm
from sklearn.cluster import KMeans
km = KMeans(init="random", n_clusters=3, n_init=10, max_iter=300,
random_state=42)
y_predicted = km.fit_predict(X)

plt.title("K-Means Predicted Result")
plt.xlabel("Alcohol")
plt.ylabel("Malic Acid")
plt.scatter(df.Alcohol, df.Malicacid, c=km.labels_, cmap='rainbow')
plt.show()
```




```
# Visualisation of SSE (Sum of Squared Errors) & Elbow Graph:
sse = []
k_range = range(1, 10)

for k in k_range:
    km = KMeans(n_clusters=k, n_init=10)
    km.fit_predict(X)
    sse.append(km.inertia_)

plt.xlabel("K")
plt.ylabel("SSE")
plt.scatter(k_range, sse, color="red", marker="+")
plt.plot(k_range, sse, color="blue")
# We can see here, our elbow is at K=3

[<matplotlib.lines.Line2D at 0x7a8204383100>]
```



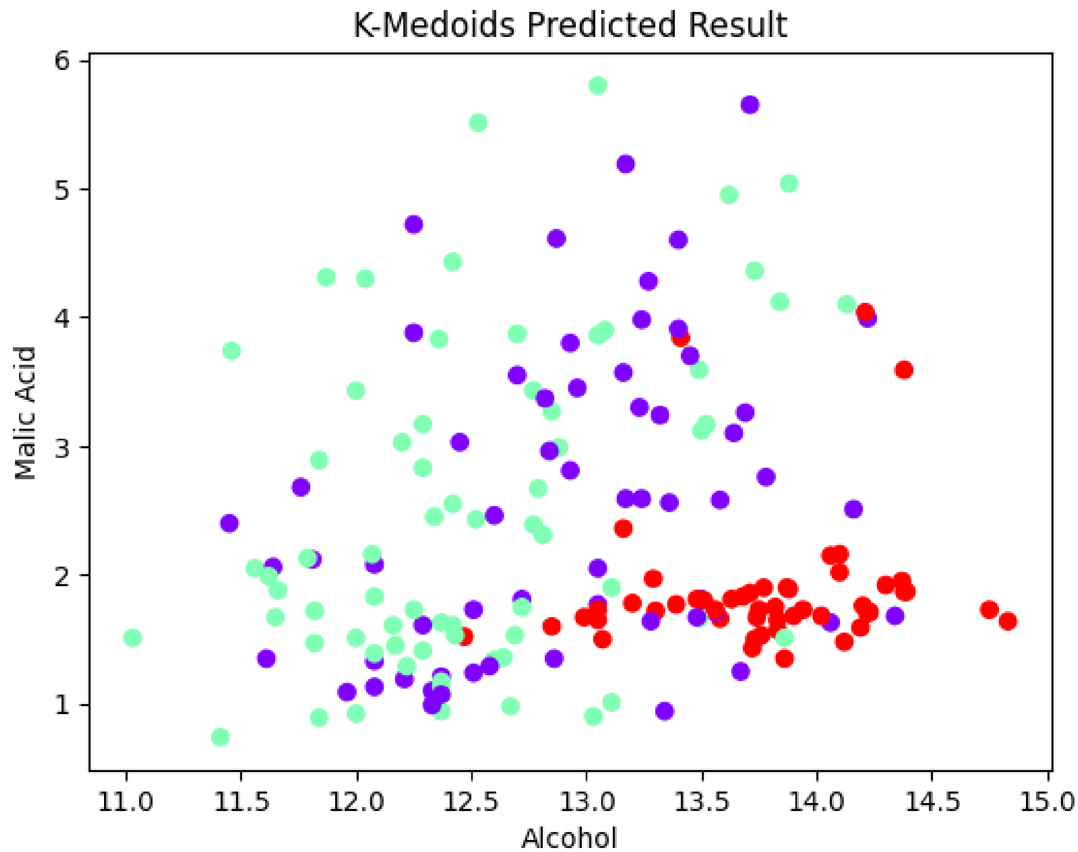

```
# Evaluating Metrics
silhouette_result = silhouette_score(X, km.labels_)
print("Silhouette Score: ", silhouette_result)

calinski_result = calinski_harabasz_score(X, km.labels_)
print("Calinski Harabasz Score: ", calinski_result)

davies_result = davies_bouldin_score(X, km.labels_)
print("Davies Bouldin Score: ", davies_result)

# Evaluating Cohesion & Separation
labels = km.labels_
centroids = km.cluster_centers_
SSE = np.sum((X - centroids[labels])**2)
overall_centroid = np.mean(X, axis=0)

SSB = np.sum([np.sum((X[labels == i] - centroids[i])**2) for i in
range(3)])

N = X.shape[0]
cohesion_scores = SSE/N
cohesion = np.mean(cohesion_scores)
separation = SSB/N
```

```python
print(f"\nCohesion Score: {cohesion}")
print(f"Separation Score: {separation}")
```

```
Silhouette Score:  0.5307235924738344
Calinski Harabasz Score:  1350.458318826902
Davies Bouldin Score:  0.5163732495928284

Cohesion Score: 117.09374643108792
Separation Score: 607.9526439562346
```

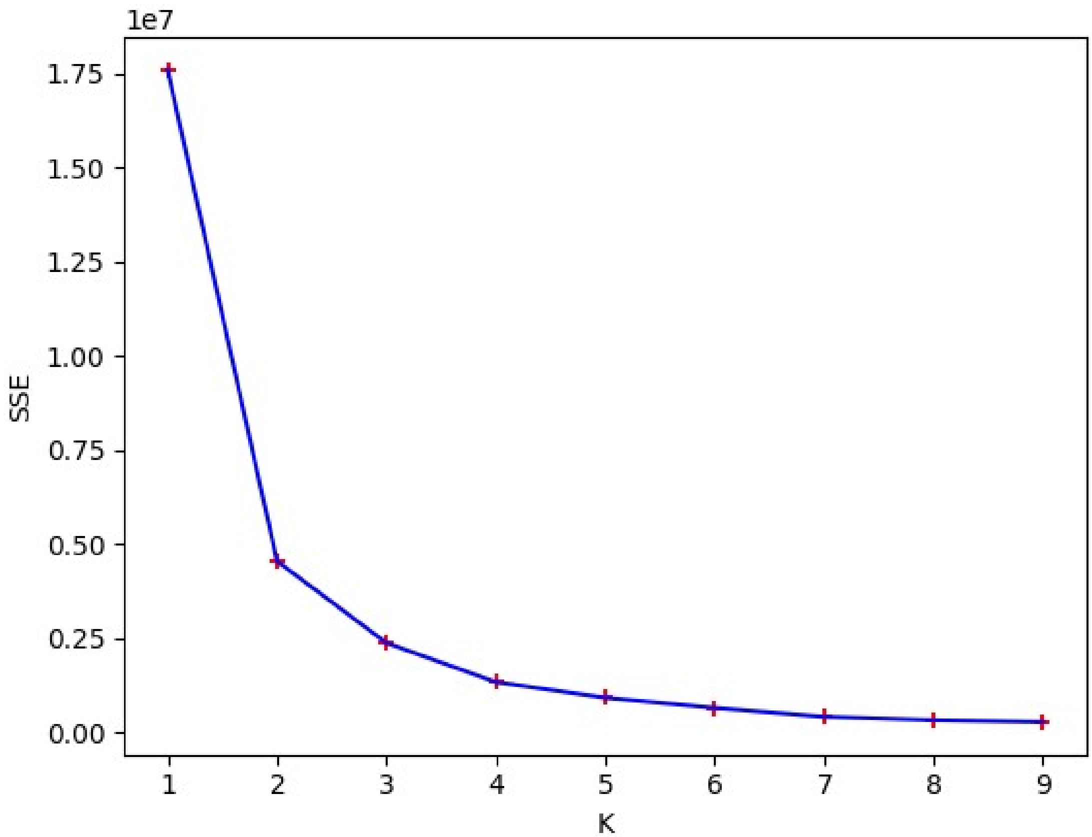
## Partition Based: K-medoids Clustering in Wine Dataset

```python
!pip install scikit-learn-extra

# Clustering using K-medoids algorithm
from sklearn_extra.cluster import KMedoids
km = KMedoids(n_clusters=3)

y_predicted = km.fit_predict(X)

plt.title("K-Medoids Predicted Result")
plt.xlabel("Alcohol")
plt.ylabel("Malic Acid")
plt.scatter(df.Alcohol, df.Malicacid, c=km.labels_, cmap='rainbow')
plt.show()
```




```
# Visualisation of SSE (Sum of Squared Errors) & Elbow Graph:
sse = []
k_range = range(1, 10)

for k in k_range:
    km = KMeans(n_clusters=k, n_init=10)
    km.fit_predict(X)
    sse.append(km.inertia_)

plt.xlabel("K")
plt.ylabel("SSE")
plt.scatter(k_range, sse, color="red", marker="+")
plt.plot(k_range, sse, color="blue")
# We can see here, our elbow is at K=3
```

```
[<matplotlib.lines.Line2D at 0x7a8254c14490>]
```



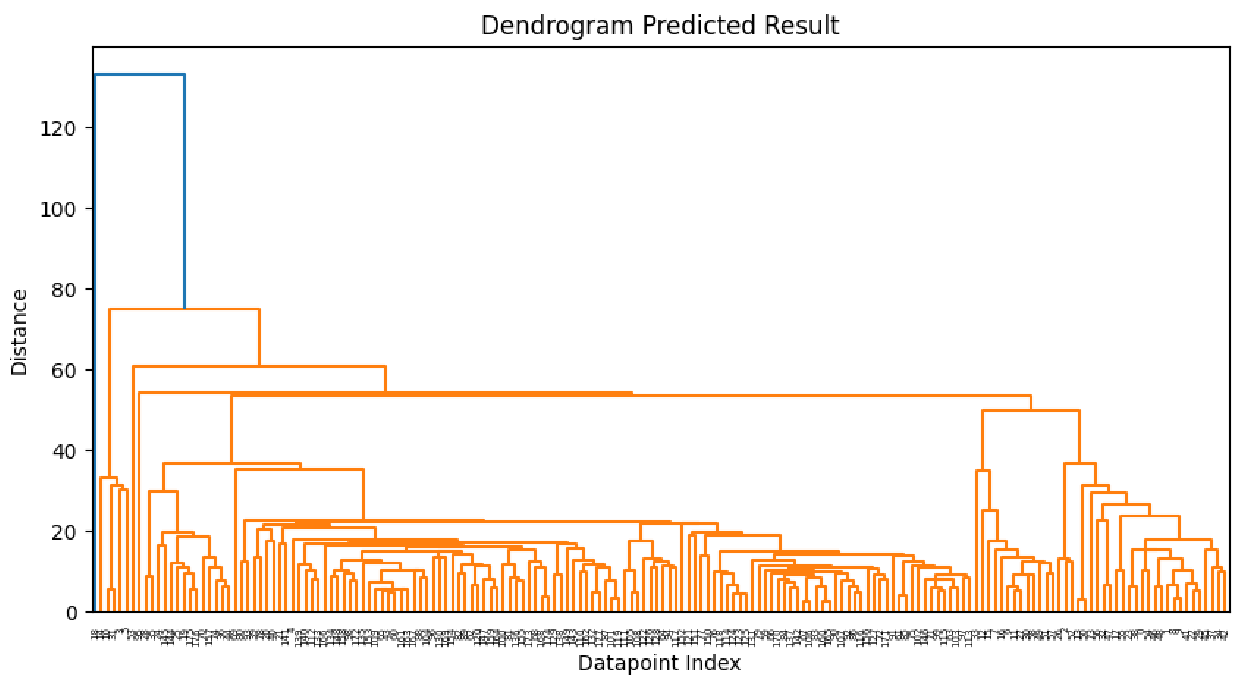

```
# Evaluating Metrics
silhouette_result = silhouette_score(X, km.labels_)
print("Silhouette Score: ", silhouette_result)

calinski_result = calinski_harabasz_score(X, km.labels_)
print("Calinski Harabasz Score: ", calinski_result)

davies_result = davies_bouldin_score(X, km.labels_)
print("Davies Bouldin Score: ", davies_result)

# Evaluating Cohesion & Separation
labels = km.labels_
centroids = km.cluster_centers_
SSE = np.sum((X - centroids[labels])**2)
overall_centroid = np.mean(X, axis=0)

SSB = np.sum([np.sum((X[labels == i] - centroids[i])**2) for i in
range(3)])

N = X.shape[0]
cohesion_scores = SSE/N
cohesion = np.mean(cohesion_scores)
separation = SSB/N
```

```
print(f"\nCohesion Score: {cohesion}")
print(f"Separation Score: {separation}")
```

```
Silhouette Score:  0.5382358200331198
Calinski Harabasz Score:  1340.298246818952
Davies Bouldin Score:  0.5274536247334654

Cohesion Score: 117.96759730604572
Separation Score: 617.91903929374
```

### Hierarchical: Dendrogram Clustering in Wine Dataset

```
# Clustering using Dendrogram Clustering algorithm
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
Z = linkage(X, method='single')

# Create and plot the dendrogram
plt.figure(figsize=(10, 5))
dn = dendrogram(Z)

plt.title('Dendrogram Predicted Result')
plt.xlabel('Datapoint Index')
plt.ylabel('Distance')
plt.show()
```



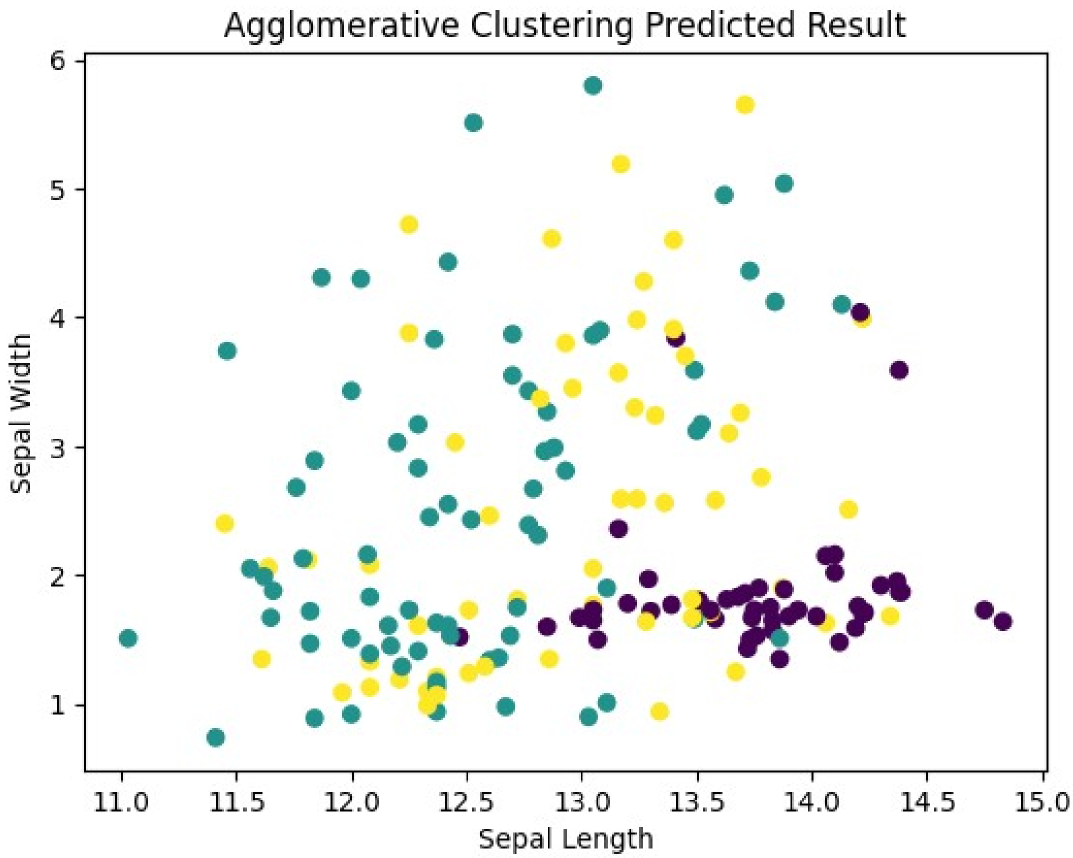

```
# Evaluating Metrics
labels = fcluster(Z, 3, criterion='maxclust')
from sklearn.metrics import silhouette_score
```

```
silhouette_result = silhouette_score(X, labels)
print("Silhouette Score: ", silhouette_result)

from sklearn.metrics import calinski_harabasz_score
calinski_result = calinski_harabasz_score(X, labels)
print("Calinski Harabasz Score: ", calinski_result)

from sklearn.metrics import davies_bouldin_score
davies_result = davies_bouldin_score(X, labels)
print("Davies Bouldin Score: ", davies_result)
```

```
Silhouette Score:  0.4879820335189063
Calinski Harabasz Score:  24.42036238154286
Davies Bouldin Score:  0.30814096183494405
```

### Hierarchical: AGNES Clustering in Wine Dataset

```
# Clustering using AGNES Clustering algorithm
from sklearn.cluster import AgglomerativeClustering

agg_cluster = AgglomerativeClustering(n_clusters=3, linkage='ward')
agg_cluster.fit(X)

plt.scatter(df.Alcohol, df.Malicacid, c=agg_cluster.labels_,
cmap='viridis')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('Agglomerative Clustering Predicted Result')
plt.show()
```

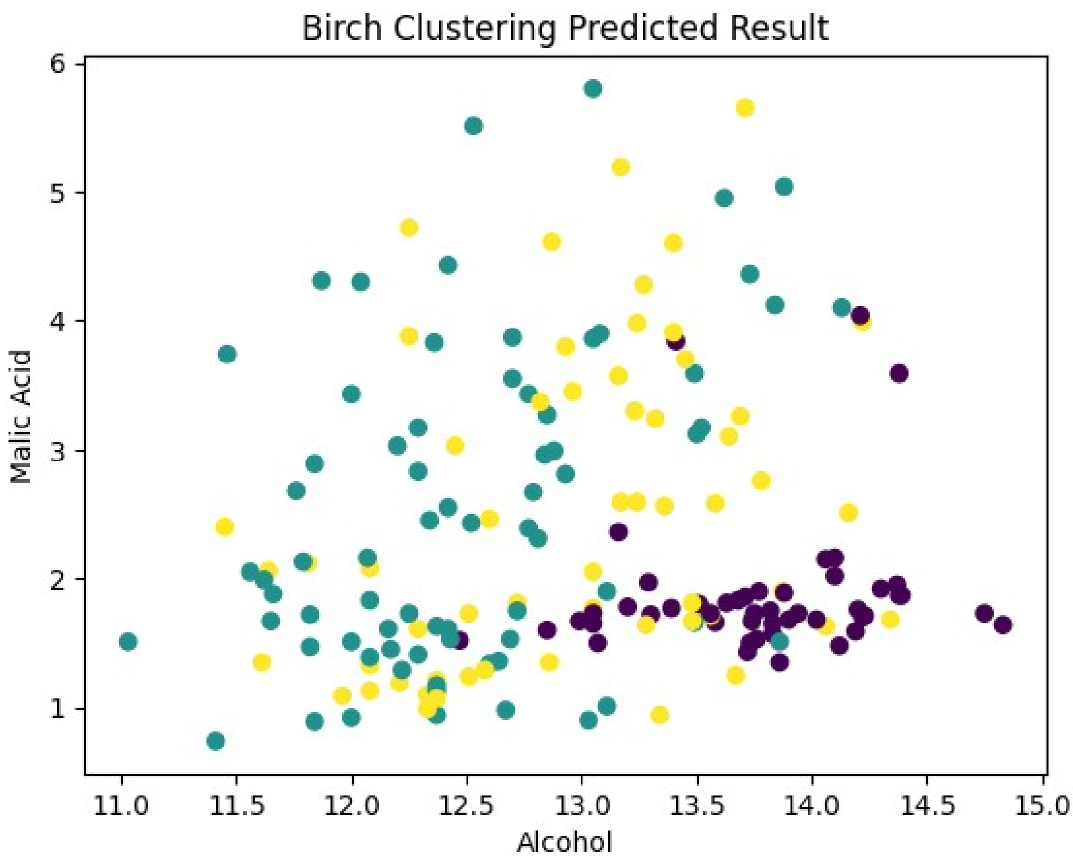



```
# Evaluating Metrics
labels = fcluster(Z, 3, criterion='maxclust')

silhouette_result = silhouette_score(X, labels)
print("Silhouette Score: ", silhouette_result)

calinski_result = calinski_harabasz_score(X, labels)
print("Calinski Harabasz Score: ", calinski_result)

davies_result = davies_bouldin_score(X, labels)
print("Davies Bouldin Score: ", davies_result)

Silhouette Score:  0.4879820335189063
Calinski Harabasz Score:  24.42036238154286
Davies Bouldin Score:  0.30814096183494405
```

## Hierarchical: BIRCH Clustering in Wine Dataset

```
# Clustering using BIRCH Clustering algorithm
from sklearn.cluster import Birch

birch_cluster = Birch(n_clusters=3)
birch_cluster.fit(X)
```

```
plt.xlabel('Alcohol')
plt.ylabel('Malic Acid')
plt.scatter(df.Alcohol, df.Malicacid, c=birch_cluster.labels_,
cmap='viridis')
plt.title('Birch Clustering Predicted Result')
plt.show()
```

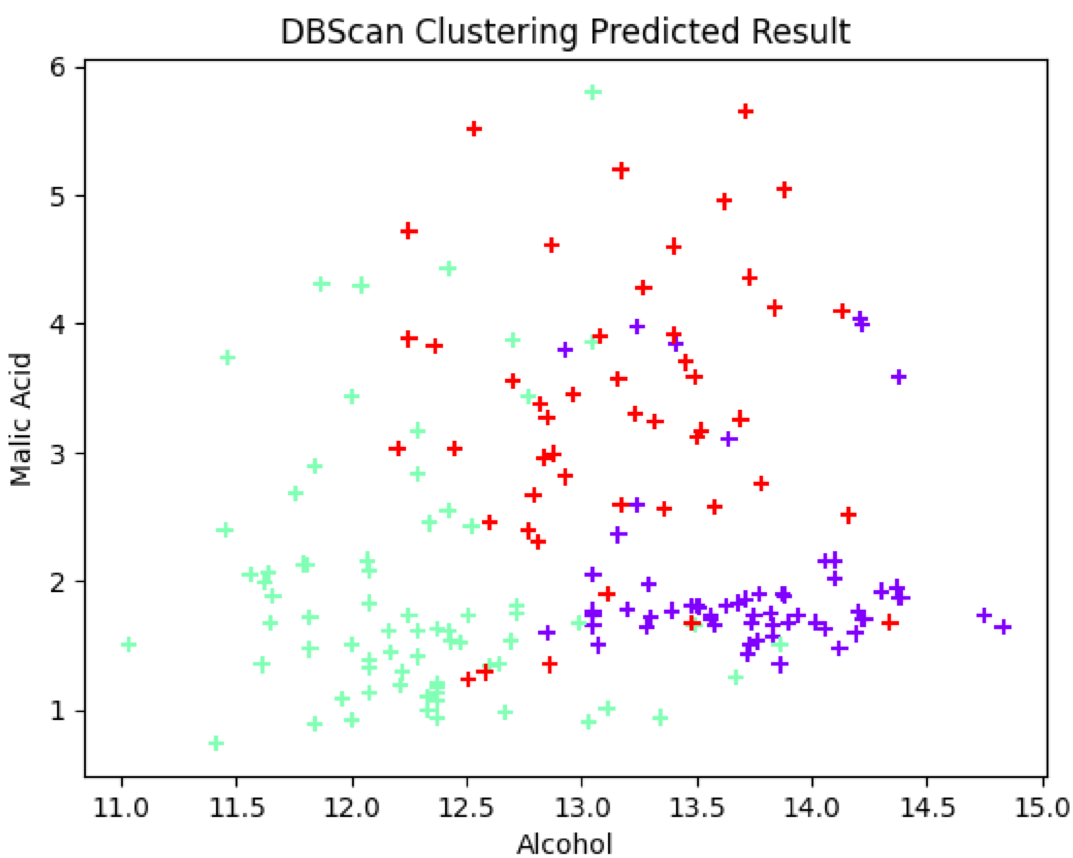



```
# Evaluating Metrics
labels = birch_cluster.fit_predict(X)

from sklearn.metrics import silhouette_score
silhouette_result = silhouette_score(X, labels)
print("Silhouette Score: ", silhouette_result)

from sklearn.metrics import calinski_harabasz_score
calinski_result = calinski_harabasz_score(X, labels)
print("Calinski Harabasz Score: ", calinski_result)

from sklearn.metrics import davies_bouldin_score
davies_result = davies_bouldin_score(X, labels)
print("Davies Bouldin Score: ", davies_result)
```

```
Silhouette Score:  0.5644796401732071
Calinski Harabasz Score:  552.851711505718
Davies Bouldin Score:  0.5357343073560251
```

### Density Based: DBSCAN Clustering in Wine Dataset

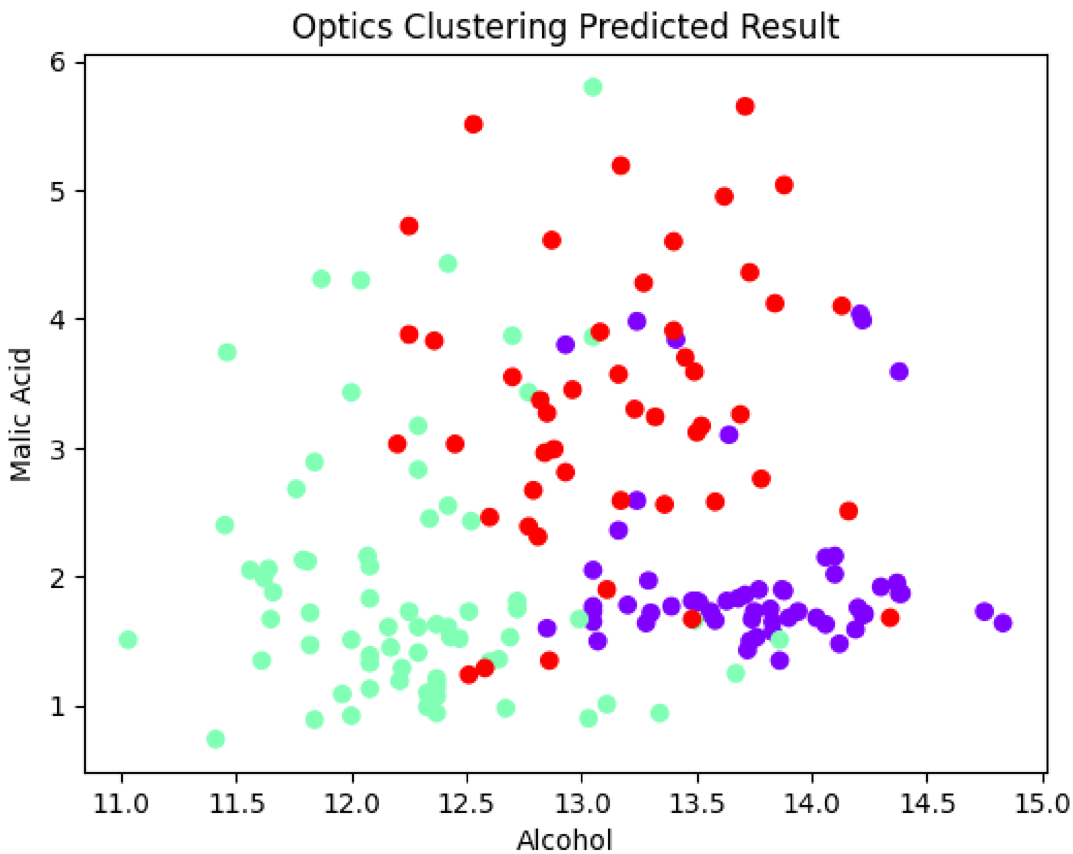
```
# Clustering using DBSCAN Clustering algorithm
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=0.5, algorithm='auto', metric='euclidean')
y = dbscan.fit_predict(X)

plt.title('DBScan Clustering Predicted Result')
plt.xlabel('Alcohol')
plt.ylabel('Malic Acid')
plt.scatter(df.Alcohol, df.Malicacid, c=df["class"], cmap='rainbow', 
marker="+")
plt.show()
```




### Density Based: Optics Clustering in Wine Dataset

```
# Clustering using Optics Clustering algorithm
from sklearn.cluster import OPTICS
```

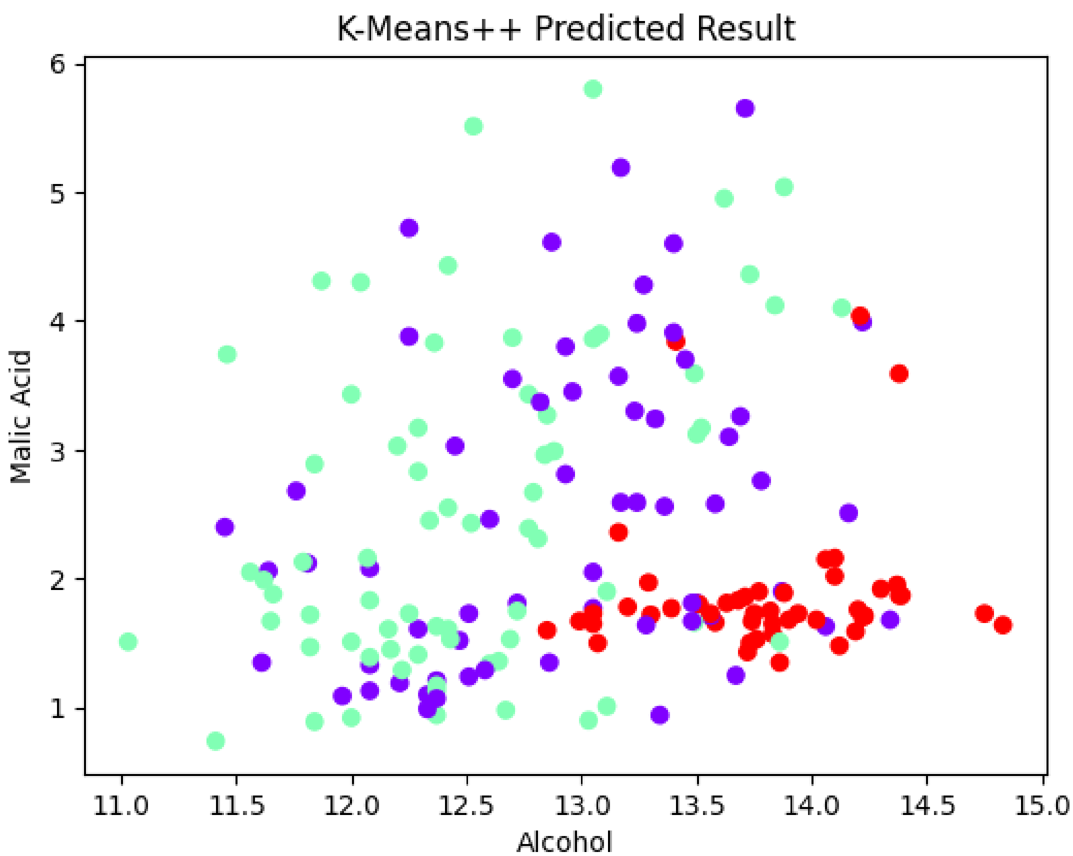
```
optics_cluster = OPTICS(min_samples=5, xi=0.05,
cluster_method='dbscan')
optics_cluster.fit(X)

plt.scatter(df.Alcohol, df.Malicacid, c=df["class"], cmap='rainbow')
plt.xlabel('Alcohol')
plt.ylabel('Malic Acid')
plt.title('Optics Clustering Predicted Result')
plt.show()
```



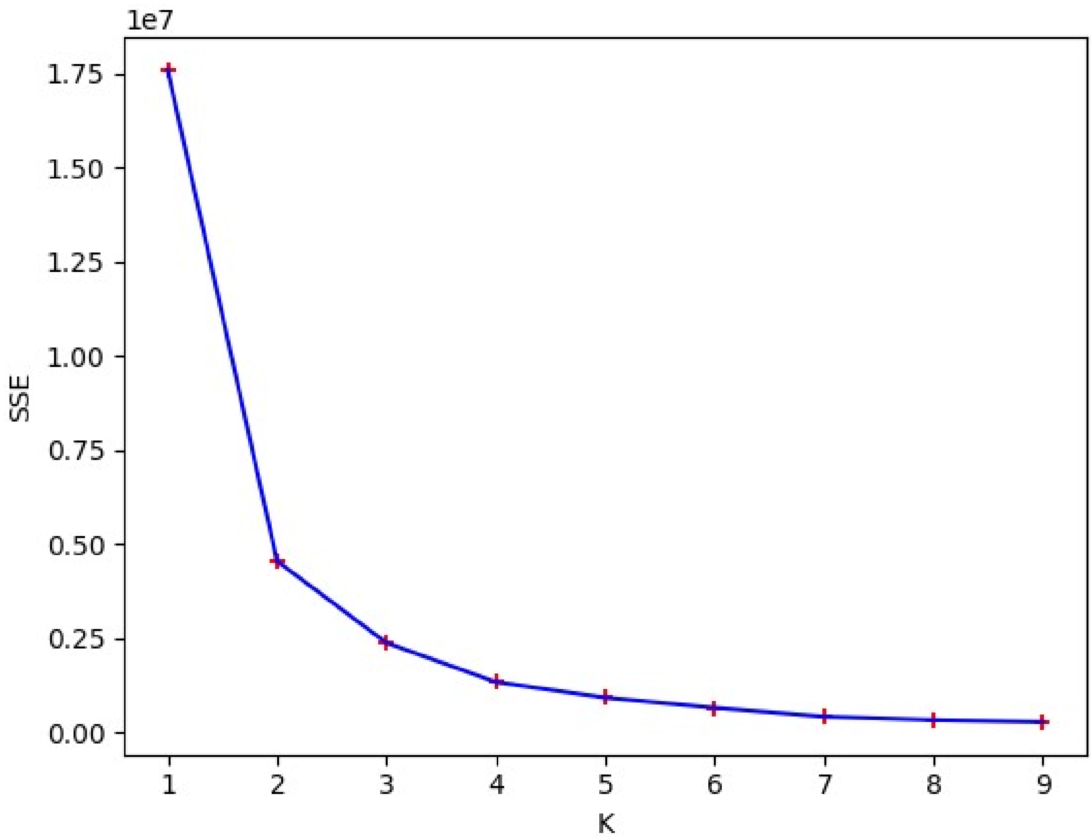

## K-means++ Clustering in Wine Dataset

```
# Clustering using K-means++ algorithm
from sklearn.cluster import KMeans
km = KMeans(init='k-means++', n_clusters=3, n_init=10, max_iter=300,
random_state=42)
km = KMeans(n_clusters=3, n_init=10)

y_predicted = km.fit_predict(X)

plt.title("K-Means++ Predicted Result")
plt.xlabel("Alcohol")
```

```
plt.ylabel("Malic Acid")
plt.scatter(df.Alcohol, df.Malicacid, c=km.labels_, cmap='rainbow')
plt.show()
```




```
# Visualisation of SSE (Sum of Squared Errors) & Elbow Graph:
sse = []
k_range = range(1, 10)

for k in k_range:
    km = KMeans(n_clusters=k, n_init=10)
    km.fit_predict(X)
    sse.append(km.inertia_)

plt.xlabel("K")
plt.ylabel("SSE")
plt.scatter(k_range, sse, color="red", marker="+")
plt.plot(k_range, sse, color="blue")
# We can see here, our elbow is at K=3

[<matplotlib.lines.Line2D at 0x7a8204295270>]
```



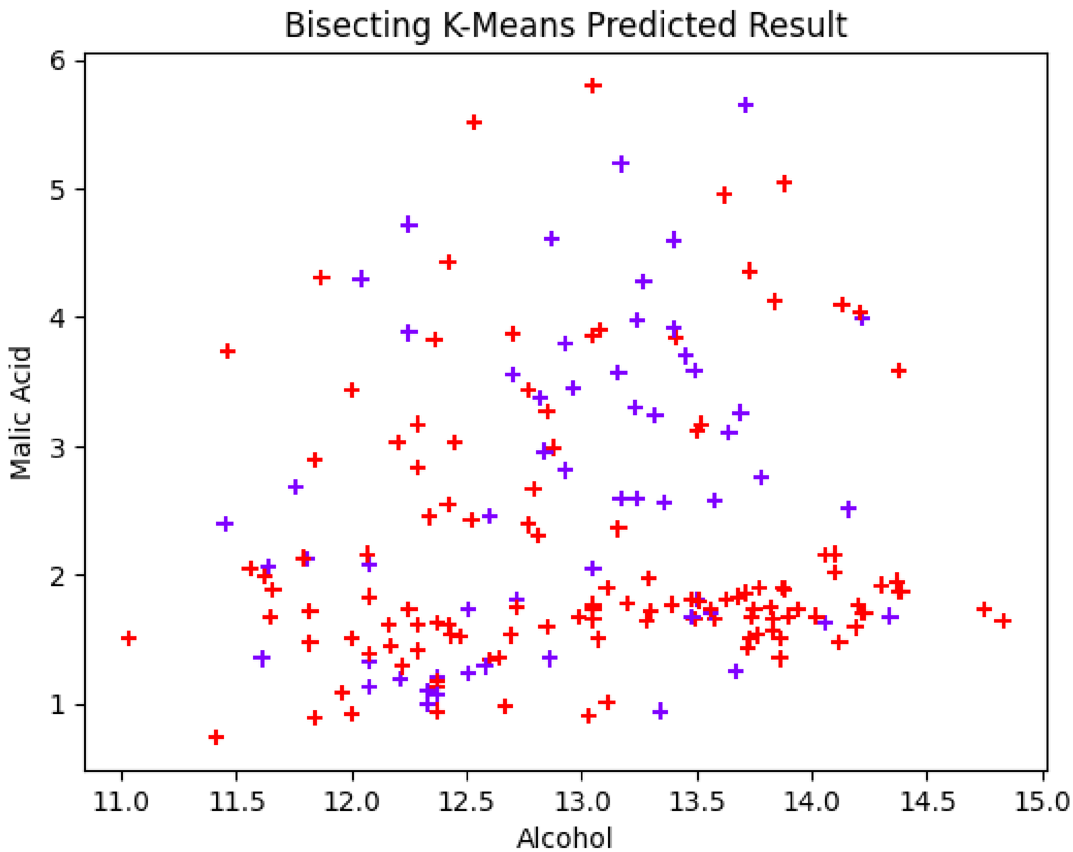

```
# Evaluating Metrics
from sklearn.metrics import silhouette_score
silhouette_result = silhouette_score(X, km.labels_)
print("Silhouette Score: ", silhouette_result)

from sklearn.metrics import calinski_harabasz_score
calinski_result = calinski_harabasz_score(X, km.labels_)
print("Calinski Harabasz Score: ", calinski_result)

from sklearn.metrics import davies_bouldin_score
davies_result = davies_bouldin_score(X, km.labels_)
print("Davies Bouldin Score: ", davies_result)

# Evaluating Cohesion & Separation
labels = km.labels_
centroids = km.cluster_centers_
SSE = np.sum((X - centroids[labels])**2)
overall_centroid = np.mean(X, axis=0)

SSB = np.sum([np.sum((X[labels == i] - centroids[i])**2) for i in
range(3)])

N = X.shape[0]
cohesion_scores = SSE/N
```

```
cohesion = np.mean(cohesion_scores)
separation = SSB/N

print(f"\nCohesion Score: {cohesion}")
print(f"Separation Score: {separation}")
```

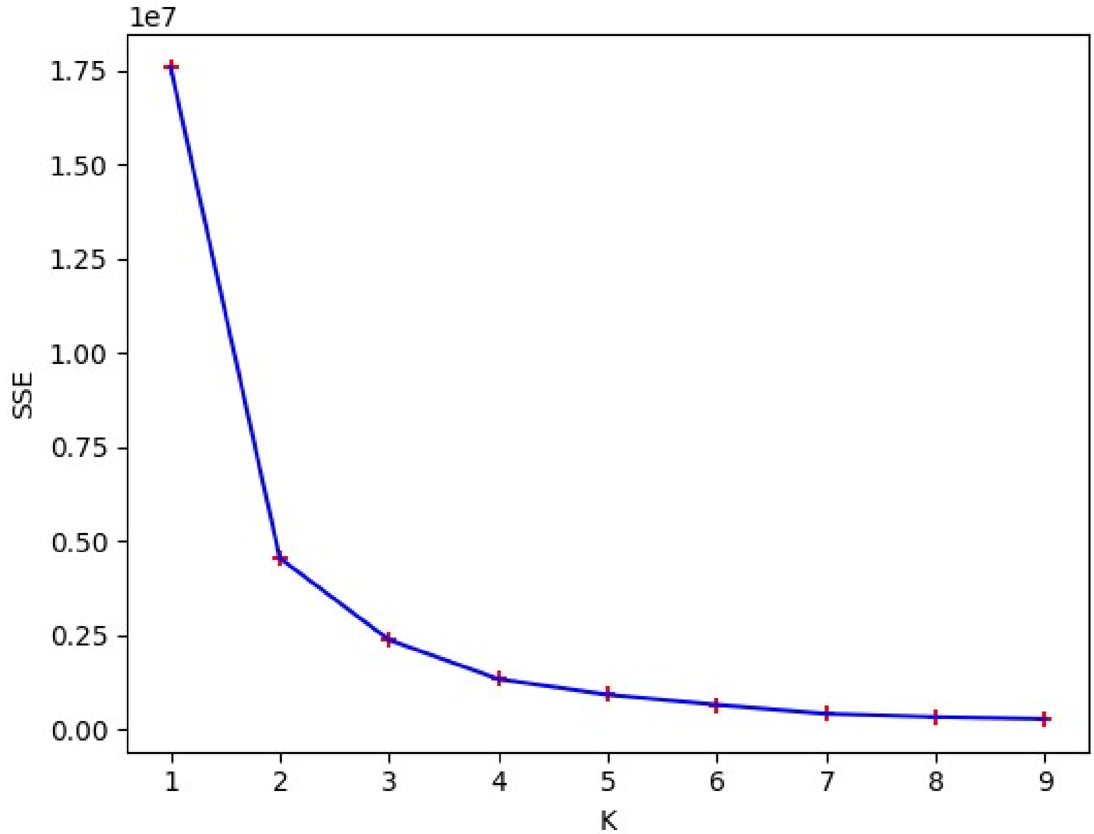
```
Silhouette Score:  0.5287268772337207
Calinski Harabasz Score:  1354.5755834453266
Davies Bouldin Score:  0.530361984915425

Cohesion Score: 116.74330248636556
Separation Score: 395.5195298071477
```

## Bisecting K-means Clustering in Wine Dataset

```
# Clustering using Bisecting K-means algorithm
from sklearn.cluster import KMeans
km = KMeans(n_clusters=1, n_init=10, random_state=0).fit(X)

K=3
for i in range(K-1):
    largest_cluster = np.argmax(np.bincount(km.labels_))
    largest_cluster_mask = (km.labels_ == largest_cluster)
    X_split = X[largest_cluster_mask]
    km.labels_[largest_cluster_mask] = KMeans(n_clusters=2, n_init=10,
random_state=0).fit(X_split).labels_


plt.title("Bisecting K-Means Predicted Result")
plt.xlabel("Alcohol")
plt.ylabel("Malic Acid")
plt.scatter(df.Alcohol, df.Malicacid, c=km.labels_, cmap='rainbow',
marker="+")
plt.show()
```




```
# Visualisation of SSE (Sum of Squared Errors) & Elbow Graph:
sse = []
k_range = range(1, 10)

for k in k_range:
    km = KMeans(n_clusters=k, n_init=10)
    km.fit_predict(X)
    sse.append(km.inertia_)

plt.xlabel("K")
plt.ylabel("SSE")
plt.scatter(k_range, sse, color="red", marker="+")
plt.plot(k_range, sse, color="blue")
# We can see here, our elbow is at K=3

[<matplotlib.lines.Line2D at 0x7a82041950f0>]
```




```python
# Evaluating Metrics
silhouette_result = silhouette_score(X, km.labels_)
print("Silhouette Score: ", silhouette_result)

calinski_result = calinski_harabasz_score(X, km.labels_)
print("Calinski Harabasz Score: ", calinski_result)

davies_result = davies_bouldin_score(X, km.labels_)
print("Davies Bouldin Score: ", davies_result)

# Evaluating Cohesion & Separation
labels = km.labels_
centroids = km.cluster_centers_
SSE = np.sum((X - centroids[labels])**2)
overall_centroid = np.mean(X, axis=0)

SSB = np.sum([np.sum((X[labels == i] - centroids[i])**2) for i in
range(3)])

N = X.shape[0]
cohesion_scores = SSE/N
cohesion = np.mean(cohesion_scores)
separation = SSB/N
```

```python
print(f"\nCohesion Score: {cohesion}")
print(f"Separation Score: {separation}")
```

```
Silhouette Score:  0.527999057875864
Calinski Harabasz Score:  1349.5503166007632
Davies Bouldin Score:  0.5215893651849661

Cohesion Score: 117.17131504353662
Separation Score: 435.555516861042
```

## CLUSTERING ALGORITHMS

| Type of Algorithm | Algorithm | Dataset | Silhouette Score | Calinski Harabasz Score | Davies Bouldin Score | Cohesion | Separation |
|---|---|---|---|---|---|---|---|
| Partition Based | K-means | IRIS PLANT DATASET | 0.34597762 | 401.8511978 | 1.025334754 | 0.047676643 | 0.087049286 |
| | | WINE DATASET | 0.530723592 | 1350.458319 | 0.51637325 | 117.0937464 | 607.952644 |
| | K-medoids | IRIS PLANT DATASET | 0.343551492 | 411.2774031 | 0.973553877 | 0.046628829 | 0.068560019 |
| | | WINE DATASET | 0.53823582 | 1340.298247 | 0.527453625 | 117.9675973 | 617.9190393 |
| Hierarchical | Dendrogram | IRIS PLANT DATASET | 0.51183871 | 277.4926776 | 0.447438434 | - | - |
| | | WINE DATASET | 0.487982034 | 24.42036238 | 0.308140962 | - | - |
| | AGNES | IRIS PLANT DATASET | 0.51183871 | 277.4926776 | 0.447438434 | - | - |
| | | WINE DATASET | 0.487982034 | 24.42036238 | 0.308140962 | - | - |
| | BIRCH | IRIS PLANT DATASET | 0.501699257 | 457.541776 | 0.626297301 | - | - |
| | | WINE DATASET | 0.56447964 | 552.8517115 | 0.535734307 | - | - |
| Density Based | DBSCAN | IRIS PLANT DATASET | 0.485842355 | 219.870227 | 7.222826995 | - | - |
| | | WINE DATASET | - | - | - | - | - |
| | OPTICS | IRIS PLANT DATASET | - | - | - | - | - |
| | | WINE DATASET | - | - | - | - | - |
| Additional | K-means++ | IRIS PLANT DATASET | 0.358592889 | 407.1954223 | 0.95820832 | 0.047076872 | 0.092279526 |
| | | WINE DATASET | 0.528726877 | 1354.575583 | 0.530361985 | 116.7433025 | 395.5195298 |
| | Bisecting K-means | IRIS PLANT DATASET | 0.340939732 | 409.940789 | 1.006055039 | 0.046774595 | 0.064729506 |
| | | WINE DATASET | 0.527999058 | 1349.550317 | 0.521589365 | 117.171315 | 435.5555169 |