Published in Towards Data Science

Ashutosh Bhardwaj Follow

May 26, 2020 · 3 min read · Listen

Silhouette Coefficient

Validating clustering techniques

After learning and applying several supervised ML algorithms like least square regression, logistic regression, SVM, decision tree etc. most of us try to have some hands-on unsupervised learning by implementing some clustering techniques like K-Means, DBSCAN or HDBSCAN.

We usually start with K-Means clustering. After going through several tutorials and Medium stories you will be able to implement k-means clustering easily. But as you implement it, a question starts to bug your mind: how can we measure its goodness of fit? Supervised algorithms have lots of metrics to check their goodness of fit like accuracy, r-square value, sensitivity, specificity etc. but what can we calculate to measure the accuracy or goodness of our clustering technique? The answer to this question is Silhouette Coefficient or Silhouette score.

Silhouette Coefficient:

Silhouette Coefficient or silhouette score is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1.

1: Means clusters are well apart from each other and clearly distinguished.

0: Means clusters are indifferent, or we can say that the distance between clusters is not significant.

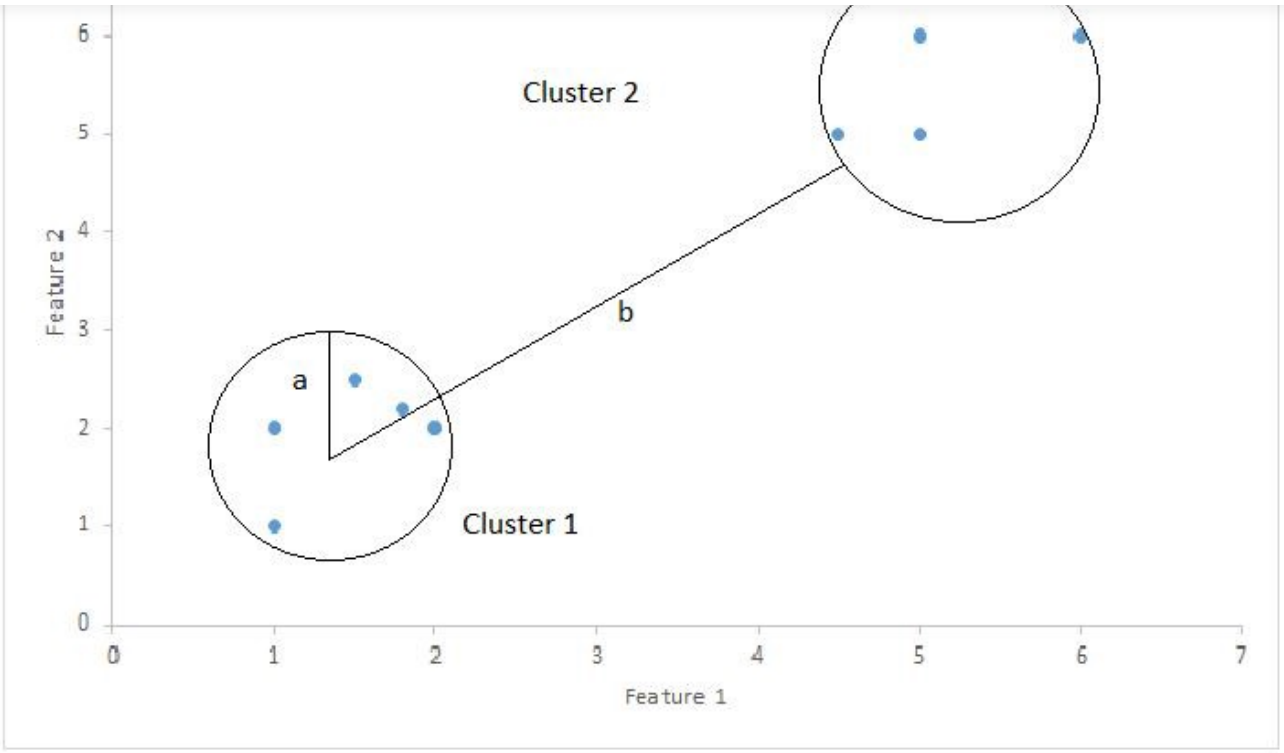

Image by author

Silhouette Score = (b-a)/max(a,b)

where

a= average intra-cluster distance i.e the average distance between each point within a cluster.

b= average inter-cluster distance i.e the average distance between all clusters.

Calculating Silhouette Score

Importing libraries:

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
%matplotlib inline
```

```
X= np.random.rand(50,2)
Y= 2 + np.random.rand(50,2)
Z= np.concatenate((X,Y))
Z=pd.DataFrame(Z) #converting into data frame for ease
```

Plotting the data:

```
sns.scatterplot(Z[0],Z[1])
```

Output

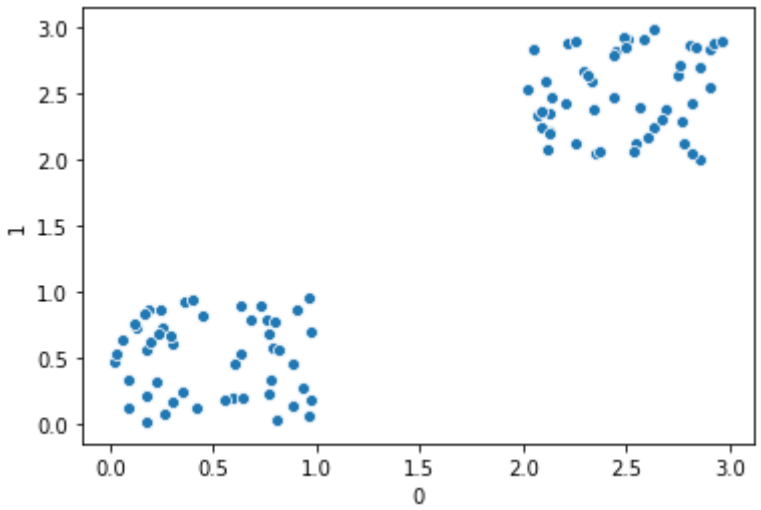

Image by author

Applying KMeans Clustering with 2 clusters:

```
KMean= KMeans(n_clusters=2)
KMean.fit(Z)
label=KMean.predict(Z)
```

Calculating the silhouette score:

We can say that the clusters are well apart from each other as the silhouette score is closer to 1.

To check whether our silhouette score is providing the right information or not let's create another scatter plot showing labelled data points.

```
sns.scatterplot(Z[0],Z[1],hue=label)
```

Output:

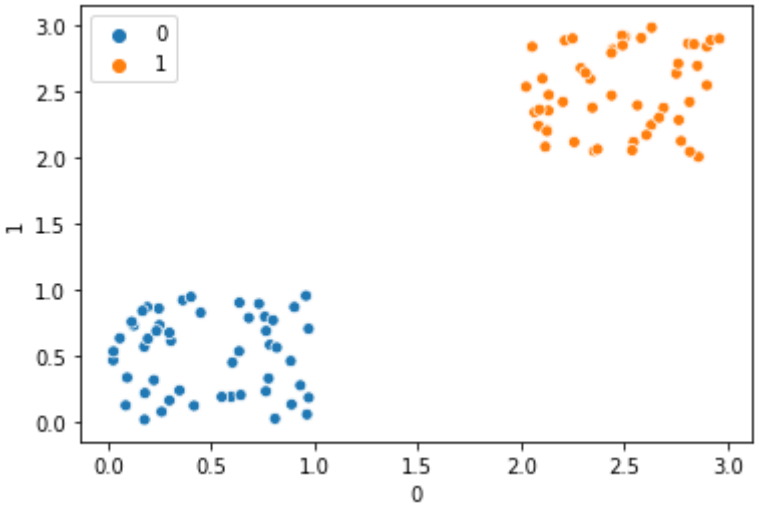

Image by author

It can be seen clearly in the above figure that each cluster is well apart from each other.

Let's try with 3 clusters:

```
KMean= KMeans(n_clusters=3)
KMean.fit(Z)
label=KMean.predict(Z)
print(f'Silhouette Score(n=3): {silhouette_score(Z, label)}')
sns.scatterplot(Z[0],Z[1],hue=label,palette='inferno_r')
```

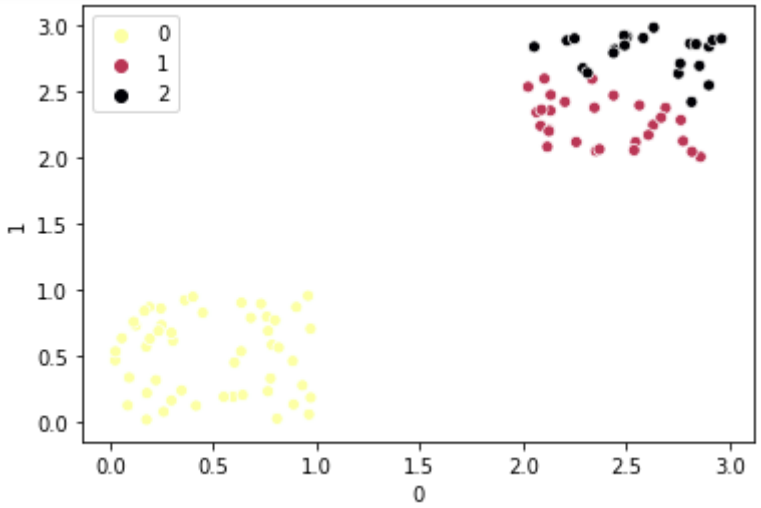

Image by author

As you can see in the above figure clusters are not well apart. The inter cluster distance between cluster 1 and cluster 2 is almost negligible. That is why the silhouette score for n= 3(0.596) is lesser than that of n=2(0.806).

When dealing with higher dimensions, the silhouette score is quite useful to validate the working of clustering algorithm as we can't use any type of visualization to validate clustering when dimensions are greater than 3.

We can also use the silhouette score to check the optimal number of clusters. In the above example, we can say that the optimal number of clusters is 2 as its silhouette score is greater than that of 3 clusters.