us4e5oyl8

February 12, 2025

What is K-Means Clustering?

K-Means is an unsupervised learning algorithm used for clustering analysis. It groups similar data points into K clusters based on their features.

How K-Means Works: 1. Initialize K centroids randomly 2. Assign each data point to the closest centroid 3. Update centroids as the mean of assigned data points 4. Repeat steps 2-3 until convergence or max iterations

[1]:
```
# Import necessary packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
```

[21]:
```
# Load the Iris dataset
iris = load_iris()
data = pd.DataFrame(data=iris.data, columns=iris.feature_names)
data['target'] = iris.target
```

[22]:
```
data.head()
```

[22]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

[8]:
```
data.tail()
```

[8]:
```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8
```

[9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
```

[10]:
```
data.describe()
```

[10]:
```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000   
mean            5.843333          3.057333           3.758000   
std             0.828066          0.435866           1.765298   
min             4.300000          2.000000           1.000000   
25%             5.100000          2.800000           1.600000   
50%             5.800000          3.000000           4.350000   
75%             6.400000          3.300000           5.100000   
max             7.900000          4.400000           6.900000   

       petal width (cm)  
count        150.000000  
mean           1.199333  
std            0.762238  
min            0.100000  
25%            0.300000  
50%            1.300000  
75%            1.800000  
max            2.500000  
```

[11]:
```
data.isnull().sum()
```

[11]:
```
sepal length (cm)    0
sepal width (cm)     0
```

```
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

[23]:
```
# Create a K-Means model with 3 clusters (since Iris dataset has 3 classes)
kmeans = KMeans(n_clusters=3, random_state=42)
```

[24]:
```
# Fit the model to the dataset (excluding target column)
kmeans.fit(data[['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',␣
 ↪'petal width (cm)']])
```

[24]:
```
KMeans(n_clusters=3, random_state=42)
```

[25]:
```
# Predict cluster labels for each data point
labels = kmeans.labels_
```

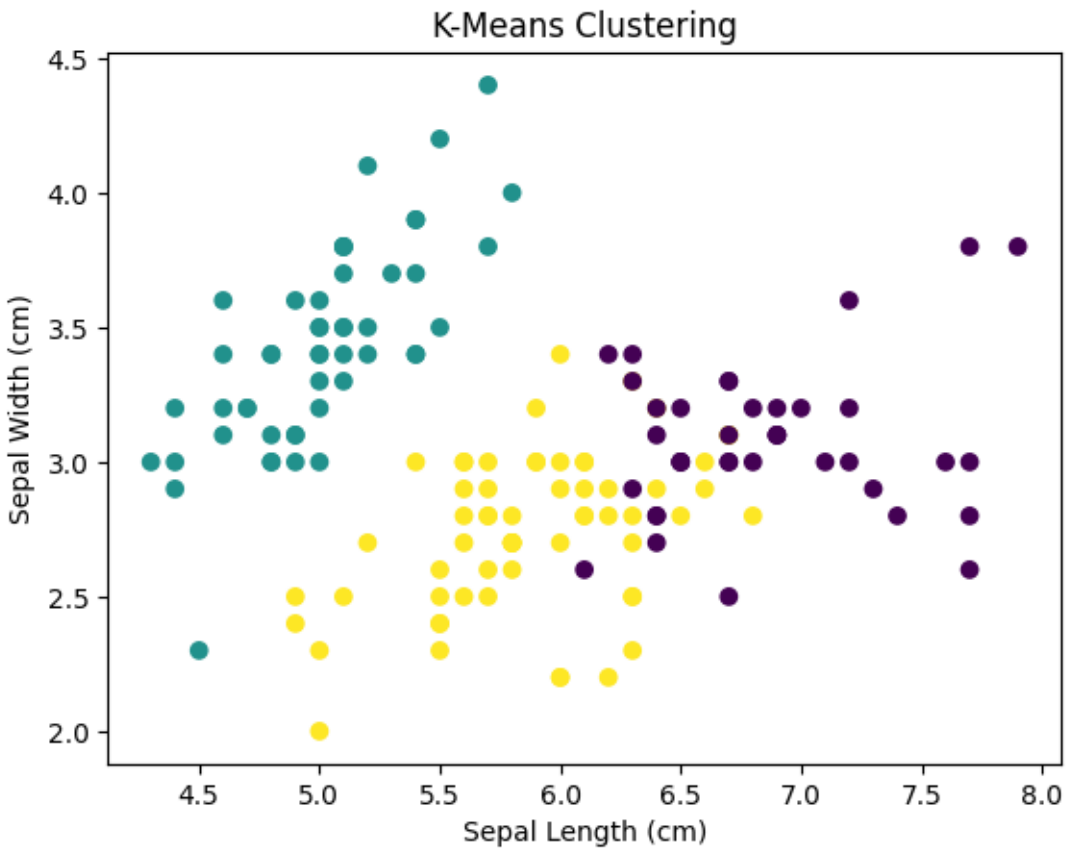
[26]:
```
# Add cluster labels to the dataframe
data['cluster'] = labels
```

[27]:
```
#Display first few rows of the updated dataframe
print(data.head())
```

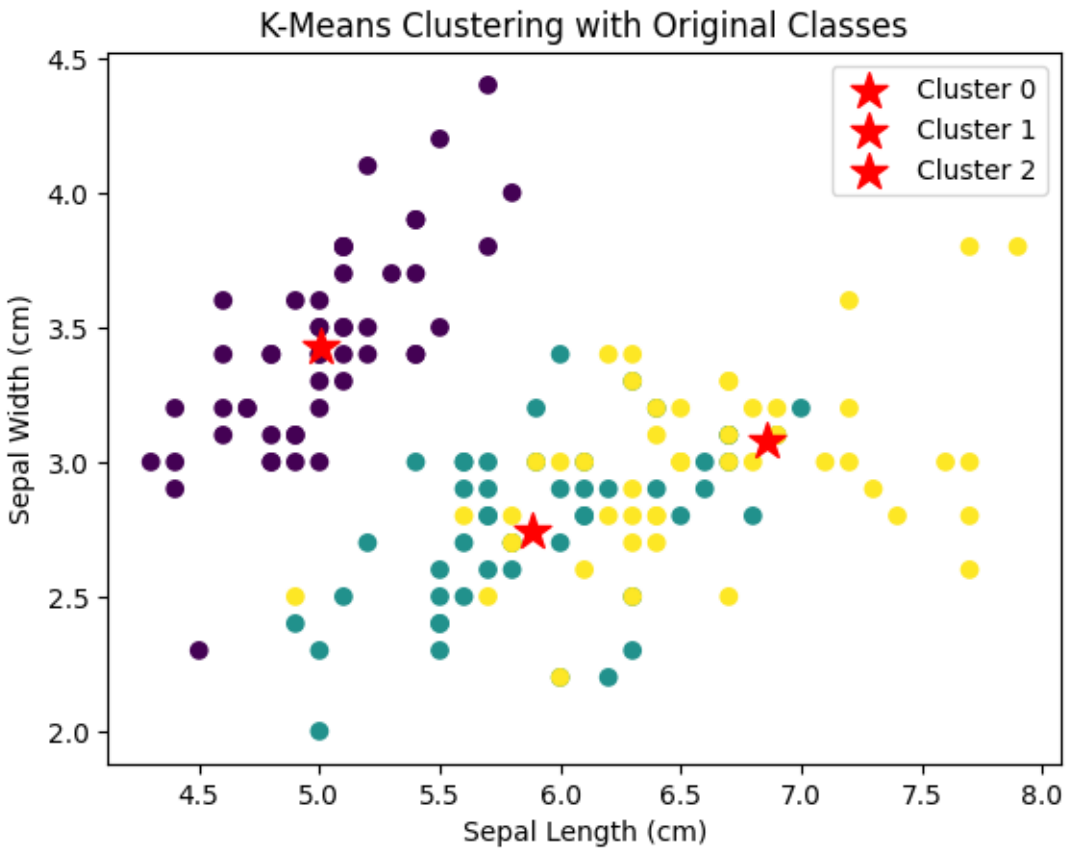
```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target  cluster
0       0        1
1       0        1
2       0        1
3       0        1
4       0        1
```

[28]:
```
#Cluster Plot
plt.scatter(data['sepal length (cm)'], data['sepal width (cm)'],␣
 ↪c=data['cluster'])
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('K-Means Clustering')
plt.show()
```



[29]:
```
#Cluster with Original Class Plot
plt.scatter(data['sepal length (cm)'], data['sepal width (cm)'],␣
 ↪c=data['target'], cmap='viridis')
for i in range(3):
    plt.scatter(kmeans.cluster_centers_[i,0], kmeans.cluster_centers_[i,1],␣
 ↪s=200, c='red', marker='*', label=f'Cluster {i}')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('K-Means Clustering with Original Classes')
plt.legend()
plt.show()
```



EVALUATION METRICS...

1. Silhouette Score
2. Calinski-Harabasz Index

[30]:
```
#Silhouette Score:
from sklearn.metrics import silhouette_score
silhouette = silhouette_score(data[['sepal length (cm)', 'sepal width (cm)',␣
 ↪'petal length (cm)', 'petal width (cm)']], data['cluster'])
print("Silhouette Score:", silhouette)
```

```
Silhouette Score: 0.551191604619592
```

[31]:
```
#Calinski-Harabasz Index:
from sklearn.metrics import calinski_harabasz_score
calinski = calinski_harabasz_score(data[['sepal length (cm)', 'sepal width␣
 ↪(cm)', 'petal length (cm)', 'petal width (cm)']], data['cluster'])
print("Calinski-Harabasz Index:", calinski)
```

```
Calinski-Harabasz Index: 561.5937320156642
```

These metrics evaluate the quality of clustering: - Silhouette Score: Values range from -1 to

1, where higher values indicate well-separated and cohesive clusters. - Calinski-Harabasz Index: Higher values indicate better clustering.