ARSLAN JAMIL

Choosing the Right k

K-Means Clustering

Introduction

Overview

K-Means is a machine learning algorithm (popular and unsupervised) used for grouping data into clusters based on feature similarity. The biggest challenge in K Means is deciding how many clusters best represent your data. In this tutorial, I will demonstrate how to implement it in Python, explain how it works and finally discussing practical methods like the Elbow Method and Silhouette Score to determine the optimal value of k.

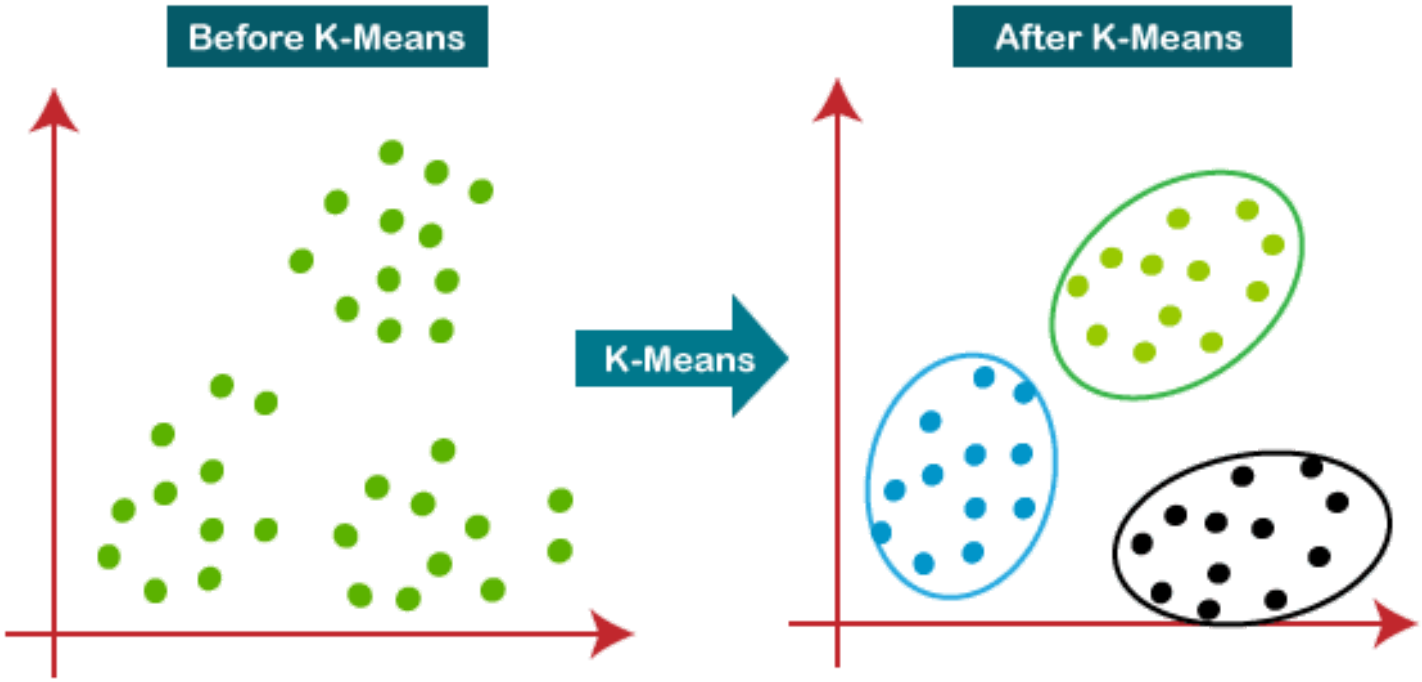

Figure 1: K-Means visual representation.

Understanding K Clustering

K-Means basically partitions the data into **k clusters**. Each data point belongs to the cluster with the nearest mean (also called as centroid). The algorithm follows these simple 4 steps:

1. **Initialize** k centroids randomly.

2. **Assign** each data point to the nearest centroid.

3. **Update** centroids as the mean of assigned points.

4. **Repeat** steps 2 and 3 until centroids do not change significantly.

K-Means minimizes WCSS (short for "**within-cluster sum of squares**") and this results in a compact and well-separated clusters.

K-Means in Python

I will use the popular Iris dataset to demonstrate the implementation in Python

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
# Load Iris data (using only first two features for easy visualization)
iris = load_iris()
X = iris.data[:, :2]
# Fit KMeans for a chosen value of k (e.g., k=3)
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
# Plotting
plt.figure(figsize=(8, 5))
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, cmap='viridis', s=50)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
c='red', s=200, alpha=0.75, marker='X', label='Centroids')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.title("K-Means Clustering (k=3) on Iris Data")
plt.legend()
plt.show()
```

Figure 2: Clusters formed by K-Means with k=3.

Choosing the Right k

A. Elbow Method

This Method helps to identify the optimal **k** by plotting the WCSS for a range of **k-values** and looking for an "**elbow**" which is a point after which WCSS reduction slows down.

```
wcss = []
ks = range(1, 11)
for k in ks:
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(X)
wcss.append(kmeans.inertia_)
plt.figure(figsize=(8, 5))
plt.plot(ks, wcss, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Within-cluster Sum of Squares (WCSS)')
plt.title('Elbow Method For Optimal k')
plt.show()
```

Figure 3: Elbow plot showing WCSS vs. number of clusters.

This **elbow** point indicates the best k and now after this point, if we add more clusters; it does not reduce the WCSS significantly. On the Iris dataset, the elbow typically appears at k=3.

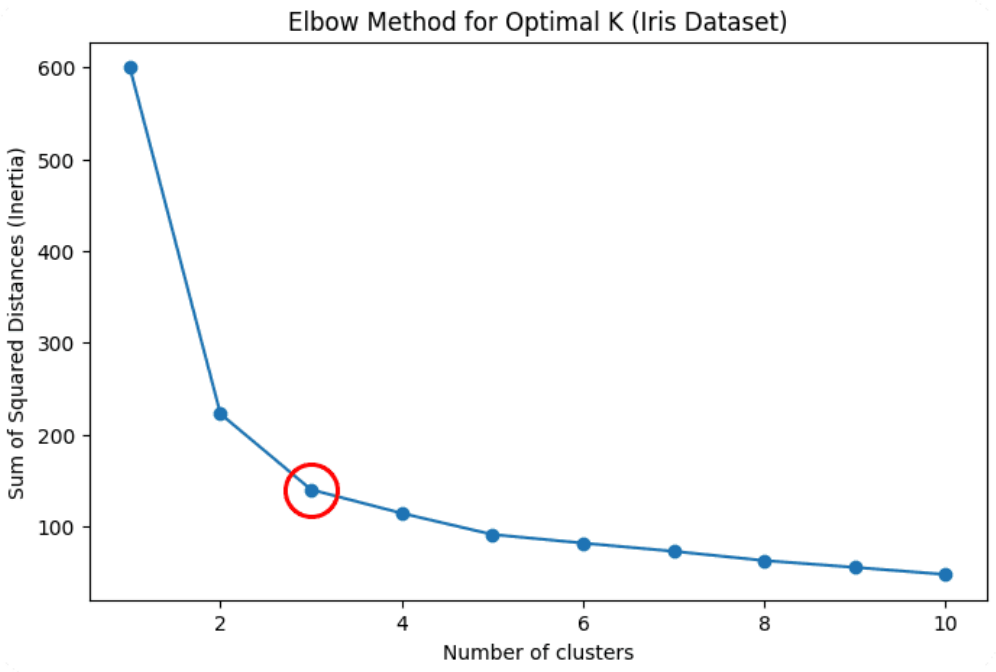

Figure 4: Iris dataset showing the elbow at k=3.

B. Silhouette Score

Now, The Silhouette Score measures how similar a datapoint is to its own cluster compared to other clusters it ranges from -1 (i.e poor) to +1 (i.e well-clustered). Following shows how we can compute the silhouette score for different values of k.

```python
from sklearn.metrics import silhouette_score
sil = []
for k in range(2, 11):
kmeans = KMeans(n_clusters=k, random_state=42)
labels = kmeans.fit_predict(X)
sil.append(silhouette_score(X, labels))
plt.figure(figsize=(8, 5))
plt.plot(range(2, 11), sil, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Method For Optimal k')
plt.show()
```

Figure 5: Silhouette scores for different k-values

The peak in the silhouette score suggests the best k .

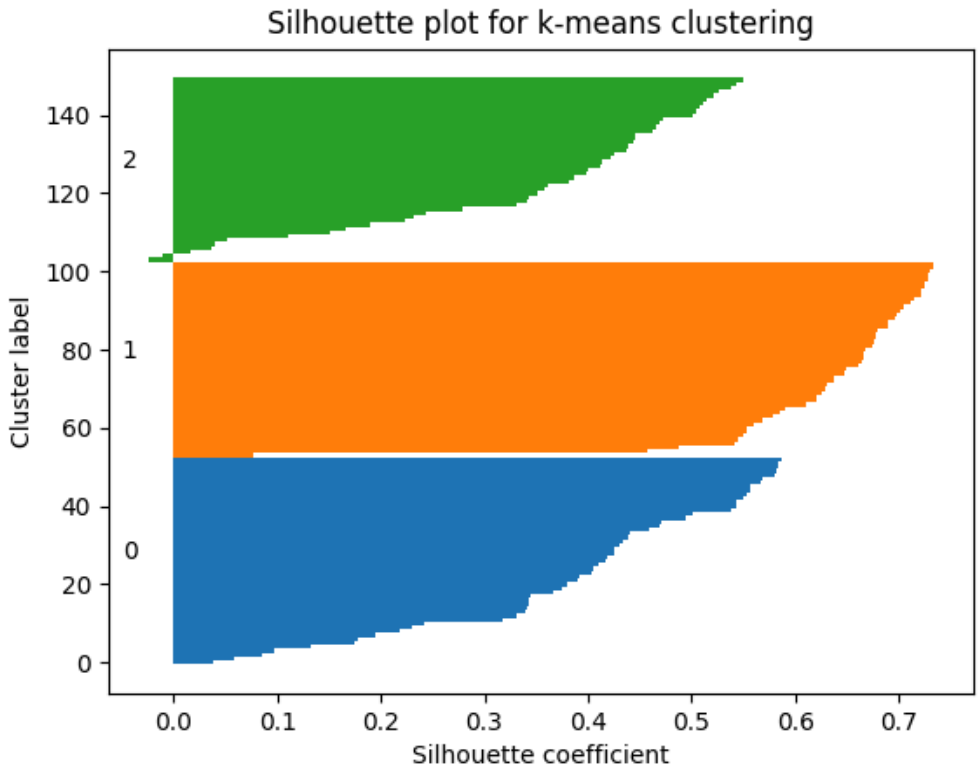

Figure 6: Silhouette scores for different k-values

Conclusion

I want you to remember the following key points from this tutorial

- K-Means is fast and widely used for clustering tasks.
- Choosing the right value of k is critical.
  - In Elbow Method, we look for the point where WCSS reduction slows (i.e the elbow).
  - In Silhouette Score, we pick the **k** with the highest score.
- For the Iris dataset (demonstrated in this tutorial), both the methods suggest **k=3**, which matches the number of species.

References

I want you to remember the following key points from this tutorial

- scikit-learn documentation on K-Means
- Elbow and Silhouette Methods explained
- "Pattern Recognition and Machine Learning" by Christopher Bishop
- K-Means Clustering: A Deep Dive into Unsupervised Learning