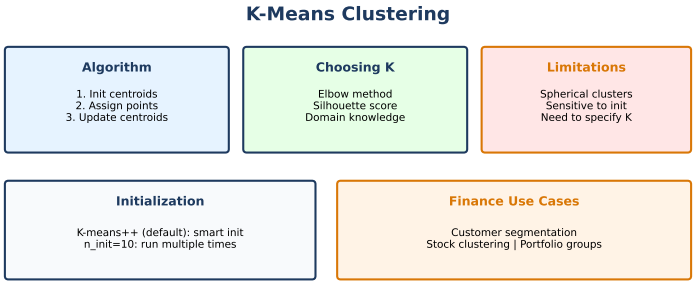
Lesson 29 Summary: K-Means Clustering

Data Science with Python – Key Concepts

Data Science Program

K-Means Overview



KMeans(n_clusters=K).fit(X) | *model.labels_* | *model.cluster_centers_*

K-Means is the most popular clustering algorithm

The Algorithm

Iterative optimization:

- **Step 1:** Initialize K centroids (randomly or K-means++)
- **Step 2:** Assign each point to nearest centroid
- **Step 3:** Update centroids as cluster means
- **Repeat:** Steps 2-3 until convergence

Algorithm converges when assignments stop changing

Choosing K

Methods to find optimal clusters:

- **Elbow method:** Plot inertia vs K, find bend
- **Silhouette score:** Measures cluster cohesion
- **Domain knowledge:** Business requirements

There is no single correct K; use multiple methods

Initialization Matters

K-means++ (default):

- **Problem:** Random init can give poor results
- **Solution:** Smart initialization spreads centroids
- **n_init:** Run multiple times, keep best

K-means++ is much better than random initialization

Feature Scaling

Critical for K-means:

- **Why:** Distance-based algorithm
- **Without scaling:** Large-scale features dominate
- **Solution:** StandardScaler before clustering

Always scale features before K-means

Limitations

When K-means struggles:

- **Non-spherical:** Assumes circular clusters
- **Varying sizes:** All clusters treated equally
- **Outliers:** Can distort centroids

Consider DBSCAN or hierarchical for complex shapes

Inertia and Silhouette

Evaluation metrics:

- **Inertia:** Sum of squared distances to centroids
- **Silhouette:** Cohesion vs separation (-1 to 1)
- **Higher silhouette:** Better-defined clusters

Silhouette close to 1 indicates well-separated clusters

Finance Applications

Clustering use cases:

- **Customer segmentation:** Group by behavior
- **Stock clustering:** Similar price patterns
- **Regime detection:** Market states

Clustering reveals hidden structure in financial data

Quick Reference

Essential Commands:

Task	Code
Fit model	`KMeans(n_clusters=K).fit(X)`
Get labels	`model.labels_`
Get centroids	`model.cluster_centers_`
Inertia	`model.inertia_`
Silhouette	`silhouette_score(X, labels)`

K-means is fast, simple, and effective for spherical clusters