Lesson 29: K-Means Clustering

Data Science with Python – BSc Course

Data Science Program

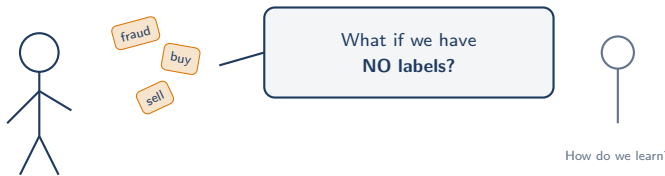
BSc Course

45 Minutes

The End of Labels

Supervised learning needed labels for every example.

- L21–L28: every row had a target – price, class, fraud/not-fraud
- But real-world data is often **unlabeled**
- Can we still learn something useful?

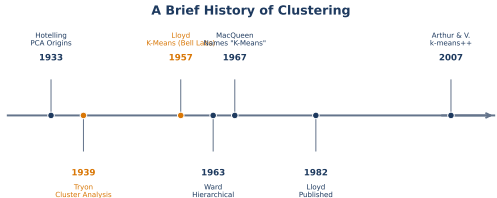


Welcome to unsupervised learning – discovering structure without being told what to look for

Brief History of Clustering

Clustering has a rich history across many disciplines

- 1939: Tryon applied cluster analysis in psychology research
- 1957: Lloyd invented K-Means at Bell Labs (published 1982)
- 1967: MacQueen coined the name "K-Means"

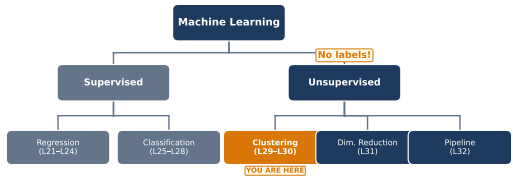


K-Means was invented at Bell Labs in 1957 but not formally published until 25 years later

Where Unsupervised Fits

ML has two main paradigms

- **Supervised** (L21–L28): predict a known target
- **Unsupervised** (L29–L32): discover hidden structure
- Today: clustering – grouping similar observations



Unsupervised learning finds patterns without any labels – clustering is its most common task

Learning Objectives

The Problem: We have 500 stocks with dozens of features. How do we group similar stocks without predefined categories?

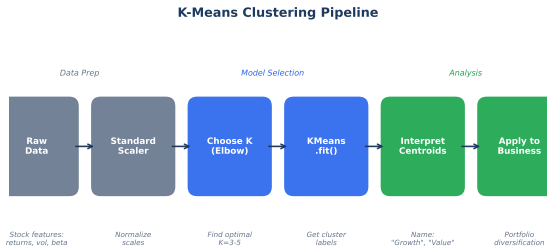
After this lesson, you will be able to:

1. Explain the K-Means algorithm step by step
2. Choose optimal K using the elbow method and silhouette score
3. Interpret cluster centroids to understand group characteristics
4. Apply K-Means to segment financial assets by behavior

Prerequisites: L13 (descriptive statistics), L21 (distance concepts)

Finance application: stock segmentation, customer clustering, market regime detection

K-Means Workflow Overview

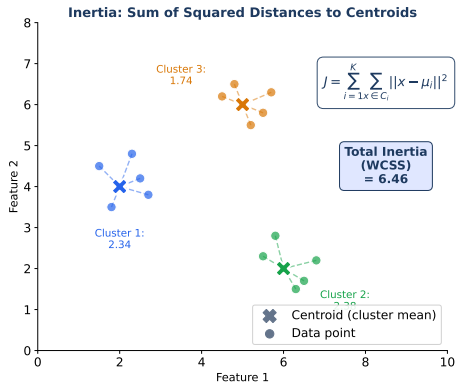


Self-study: end-to-end K-Means workflow from raw data to cluster interpretation

Inertia Formula (WCSS)

Inertia = Within-Cluster Sum of Squares

- Inertia $= \sum_{i=1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2$
- Lower inertia = tighter, more compact clusters

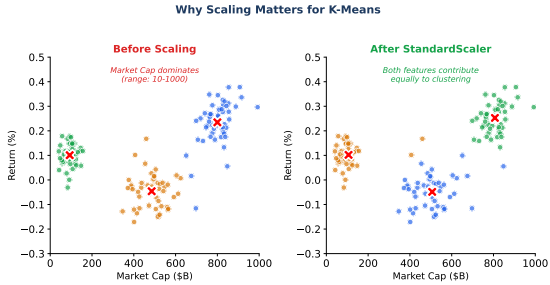


Self-study: inertia measures total squared distance from each point to its cluster center

Why Scaling Matters

Features on different scales distort distance calculations

- Market cap ($1B–$2000B) vs daily return (-5% to +5%)
- Without scaling: large-range features dominate
- Solution: `StandardScaler` transforms each feature to mean=0, std=1

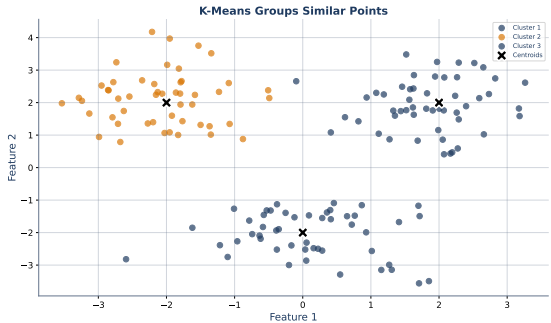


Self-study: always standardize before K-Means – otherwise large-scale features dominate

K-Means Groups Similar Points

Points within the same cluster share similar characteristics

- Clustering reveals natural groupings not obvious from raw data
- Works in 2D (as shown) and in high-dimensional spaces

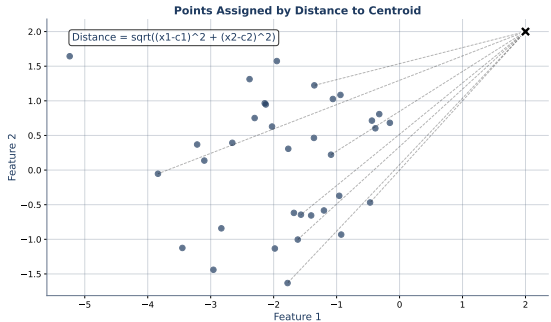


Self-study: points within a cluster are more similar to each other than to points in other clusters

Points Assigned by Distance

Each point goes to the cluster with the nearest centroid

- "Nearest" means smallest Euclidean distance
- This creates natural boundaries between clusters



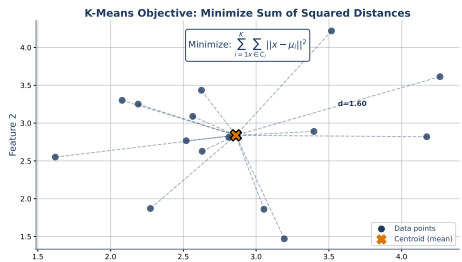
Self-study: Euclidean distance is the default metric for K-Means assignment

K-Means Objective Function

Minimize within-cluster sum of squares:

$$\min \sum_{i=1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2$$

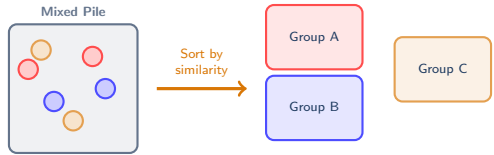
- C_i = set of points in cluster i, μ_i = centroid of cluster i
- Each point assigned to cluster with nearest centroid
- Centroids recomputed as cluster means



Self-study: WCSS (inertia) is the objective K-Means minimizes iteratively

What is Clustering?

Sorting without instructions – like organizing laundry



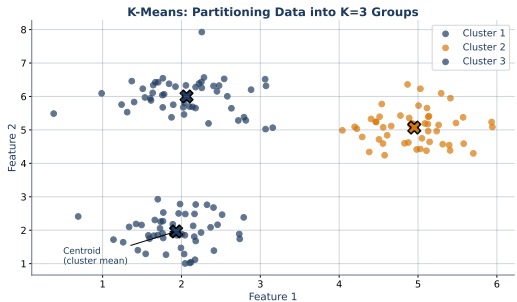
- **Goal:** group similar items together, separate dissimilar ones
- **Key insight:** the algorithm discovers groups on its own
- **Finance:** group stocks by behavior, not by sector label

Clustering = grouping by similarity. No one tells the algorithm what the groups should be.

K-Means: Core Idea

Partition data into exactly K groups

- You choose K (number of clusters) in advance
- Each point belongs to the cluster with the nearest center
- Centers (centroids) are updated iteratively until stable

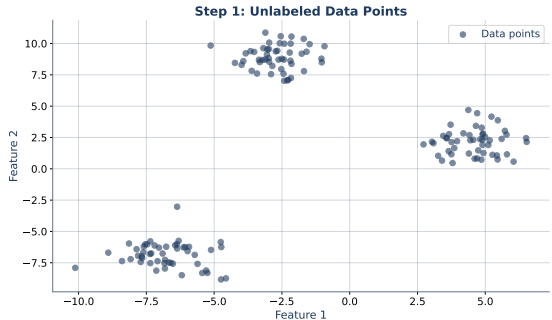


K-Means finds compact, spherical clusters by minimizing within-cluster variance (inertia)

Step 1: Raw Data

Starting point – unlabeled observations

- No colors, no groups – just scattered points
- Goal: discover natural structure hidden in the data

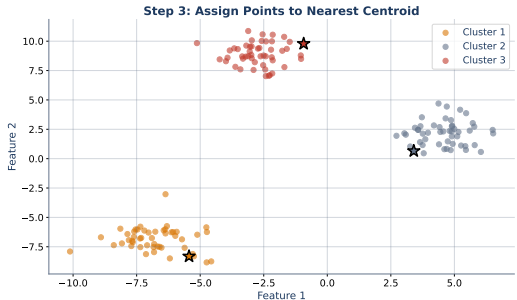


Before clustering: no structure visible. The algorithm will find groups automatically.

Steps 2–3: Initialize & Assign

Place K centroids, then assign every point

- **Step 2:** Place K initial centroids (randomly or via k-means++)
- **Step 3:** Assign each point to its nearest centroid
- This creates a Voronoi partition of the feature space

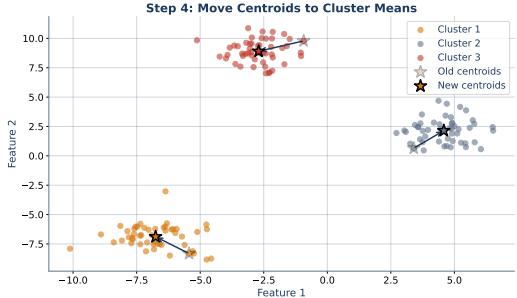


Each point is assigned to the cluster whose centroid is closest (Euclidean distance)

Step 4: Recompute & Repeat

Move centroids to cluster means, then reassign

- New centroid = mean position of all points in that cluster
- Repeat Steps 3–4 until assignments stop changing
- Convergence is guaranteed (finite iterations)

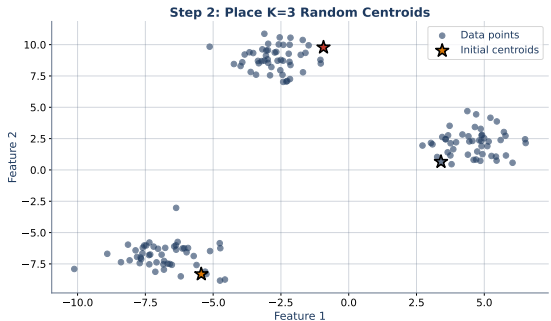


K-Means converges when no point changes cluster. Typically 10–20 iterations.

Step 2: Initialize Centroids

Initial placement affects the final result

- Random initialization can lead to different clusterings
- k-means++ spreads initial centroids for better convergence

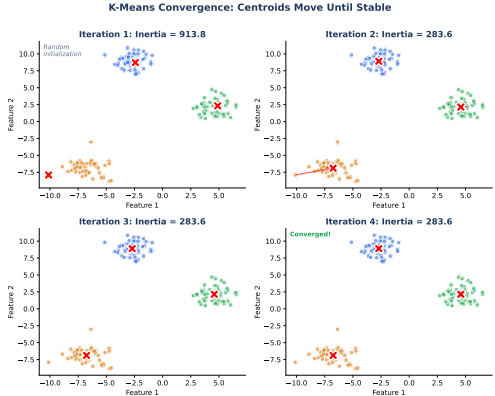


Self-study: k-means++ (sklearn default) selects initial centers that are well-separated

Convergence Iterations

How quickly does K-Means converge?

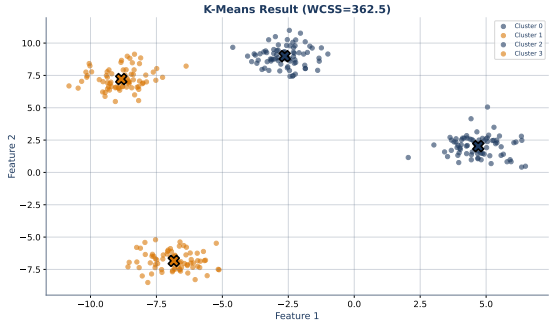
- Inertia decreases each iteration until stable
- Most improvement happens in the first few iterations



Self-study: convergence typically takes 10–20 iterations for well-separated clusters

K-Means Initialization Methods

Comparing random vs k-means++ initialization



Self-study: k-means++ (default in sklearn) avoids poor starting positions

How Many Groups?

K-Means requires you to choose K in advance

This is the central challenge:

- Too few clusters $\rightarrow$ merge distinct groups
- Too many clusters $\rightarrow$ split natural groups apart
- No single "correct" answer – it depends on the question

Two complementary methods:

1. **Elbow method** – plot inertia vs K, look for the bend
2. **Silhouette score** – measure how well-separated clusters are

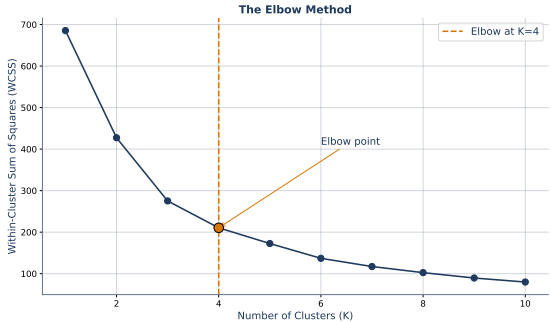
Finance example:
Are there 3 market regimes (bull/bear/sideways) or 5 (adding crash and recovery)?
The data can support multiple valid answers.

Choosing K is the hardest part of K-Means. Use elbow + silhouette + domain knowledge.

The Elbow Method

Plot inertia vs K – look for diminishing returns

- Inertia always decreases as K increases
- The "elbow" = where adding more clusters stops helping much



The elbow is subjective – combine with domain knowledge and silhouette score

Checkpoint: Test Your Understanding

Q1: What does inertia measure?

Q2: Why must we standardize before K-Means?

Q3: If the elbow plot has no clear bend, what does that suggest about the data?

Quick answers:

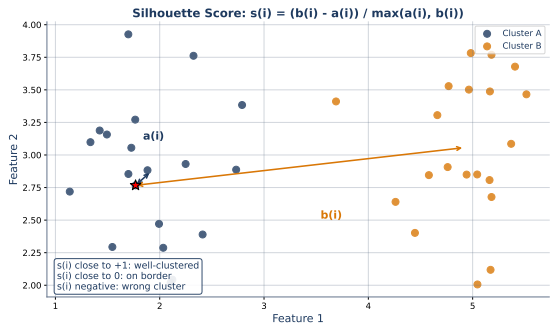
1. Inertia = total squared distance from points to their cluster centers
2. Features on different scales would dominate distance calculations
3. The data may not have well-defined cluster structure

Pause here – make sure you can answer all three before continuing

Silhouette Score

Is each point in the right cluster?

- Range: -1 to $+1$. Higher = better-separated clusters
- Compares distance to own cluster vs nearest other cluster
- $s > 0.5$: reasonable structure; $s < 0$: wrong cluster

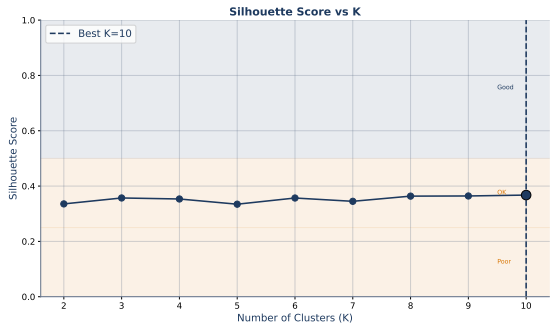


Silhouette answers: "Am I closer to my own cluster or to the next-nearest one?"

Silhouette Score vs K

Choose K that maximizes the average silhouette

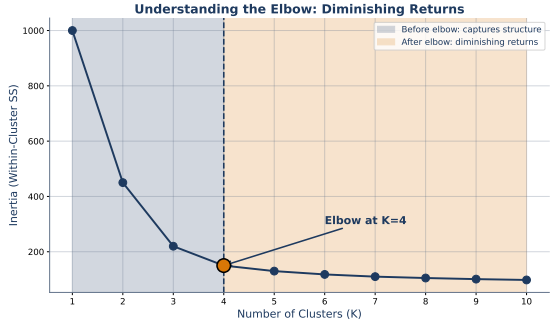
- Compute silhouette for K=2, 3, 4, ...
- Peak = best trade-off between cohesion and separation
- Combine with elbow method for robust choice



Silhouette peak + elbow bend at same K = strong evidence for that number of clusters

Understanding the Elbow

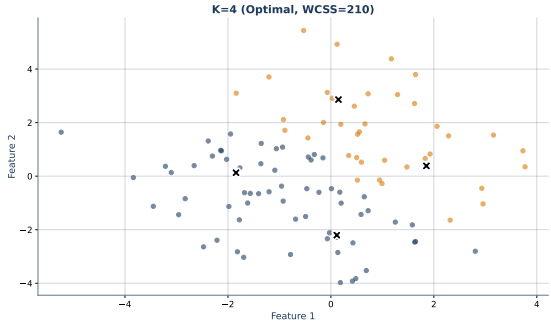
Before the elbow: real structure. After: just splitting.



Self-study: adding clusters before the elbow captures real groups; after it, gains are marginal

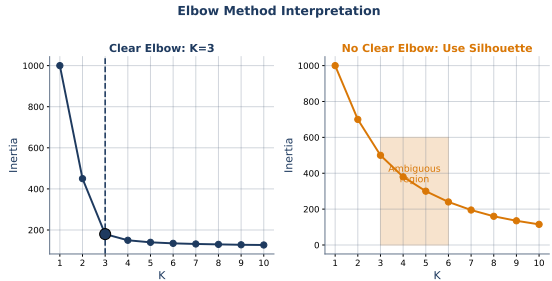
Python Implementation

Computing the elbow curve in sklearn



Self-study: loop over K values, store `kmeans.inertia_`, plot the curve

Elbow Method Variant

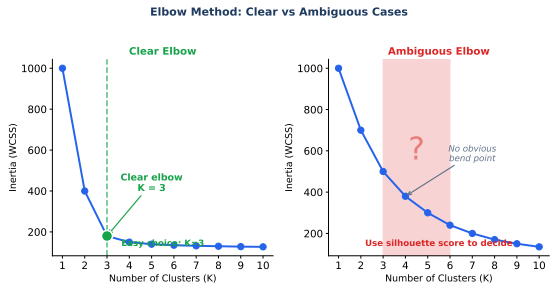


Self-study: no clear elbow? Data may not have well-defined clusters.

When the Elbow is Ambiguous

Real data often has no sharp bend

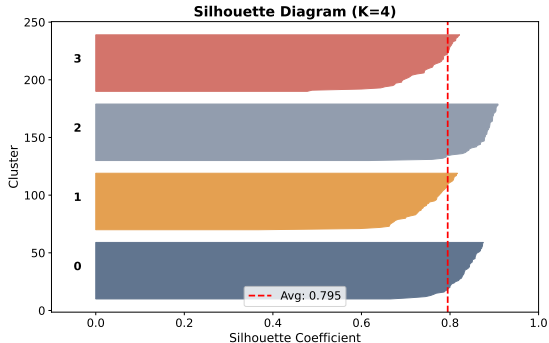
- Try multiple K candidates and compare interpretability
- Use domain knowledge to narrow the range



Self-study: ambiguous elbows are common – combine methods for a robust decision

Silhouette Diagram (K=4)

Per-point silhouette scores reveal cluster quality

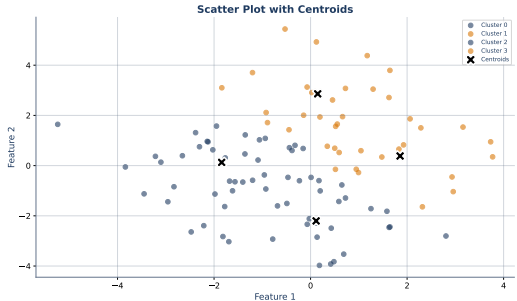


Self-study: wide bars = cohesive clusters; negative scores = misclassified points

Visualizing Clusters

Scatter plot – the first sanity check

- Color points by cluster assignment
- Mark centroids with a distinct symbol
- For high-dimensional data: reduce to 2D with PCA first

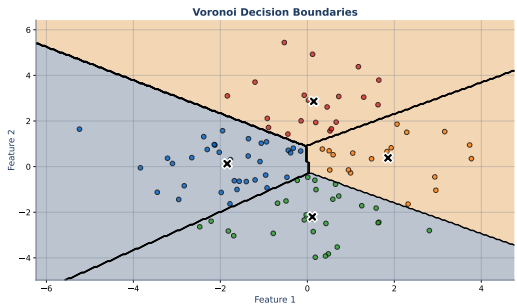


Always visualize clusters to check whether the groupings make intuitive sense

Voronoi Boundaries

Where one cluster ends and another begins

- Each region contains all points closest to that centroid
- Boundaries are straight lines (linear decision boundaries)
- Like territorial borders based on distance to capitals

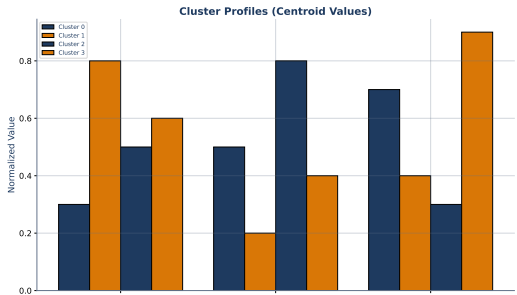


Voronoi partitions create linear boundaries – K-Means cannot capture non-convex shapes

Interpreting Centroids

Centroids tell you what each cluster "looks like"

- Centroid = average feature values for all points in that cluster
- Compare across clusters to understand what distinguishes them
- Use `scaler.inverse_transform()` to get original-scale values

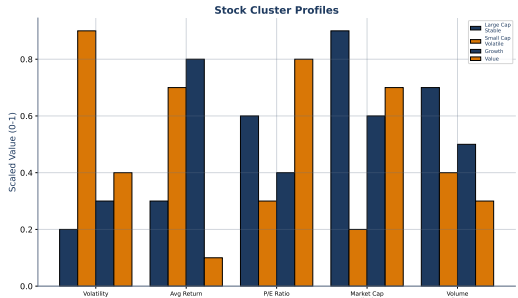


Bar chart of centroid features reveals what makes each cluster distinct

Finance: Stock Cluster Profiles

What does each stock group look like?

- Features: return, volatility, beta, market cap, P/E ratio
- Centroids reveal natural groupings beyond sector labels
- Name clusters: "Blue Chips", "Growth", "Speculative"

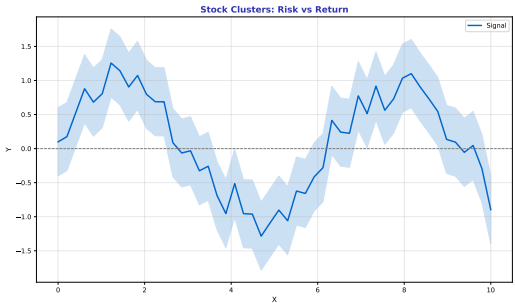


Cluster names come from interpreting centroids: high vol + low cap = "Speculative"

Finance: Risk-Return Map

Stocks clustered by volatility and return

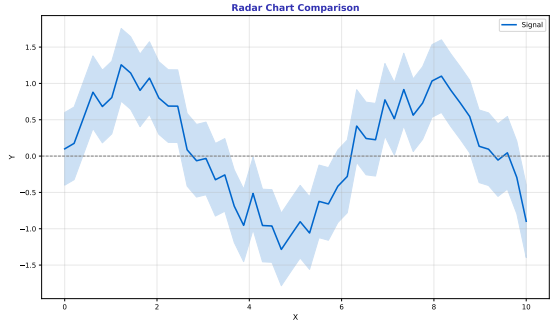
- Each cluster occupies a different risk-return region
- Diversification strategy: pick one stock per cluster
- Clusters often cut across traditional sector boundaries



Portfolio tip: select one stock per cluster for maximum diversification

Radar Chart Comparison

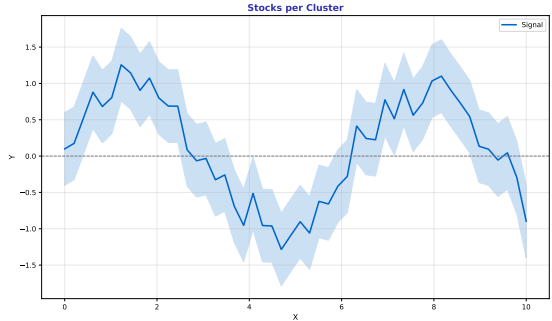
Multi-dimensional cluster profiles at a glance



Self-study: radar charts show all feature dimensions for each cluster simultaneously

Stocks per Cluster

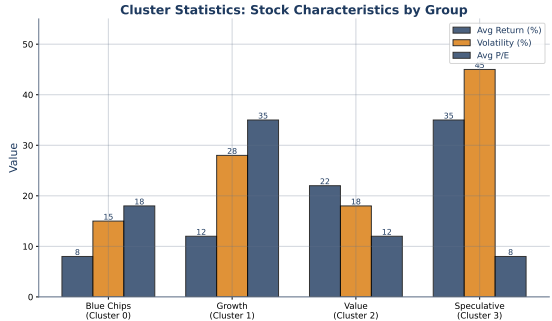
Check cluster balance – very uneven sizes may indicate poor K



Self-study: if one cluster has 90% of points, K may be too large or data lacks structure

Cluster Statistics

Summary statistics validate that clusters are financially meaningful

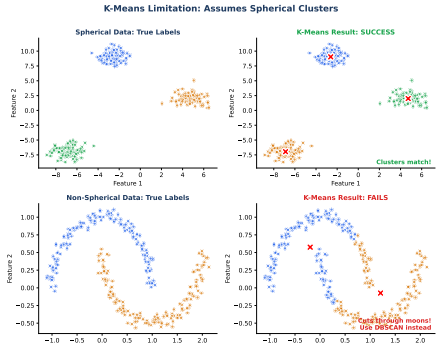


Self-study: compare mean return, volatility, beta across clusters to check interpretability

K-Means Limitations

When K-Means struggles

- Assumes spherical, equally-sized clusters
- Sensitive to outliers; cannot handle non-convex shapes

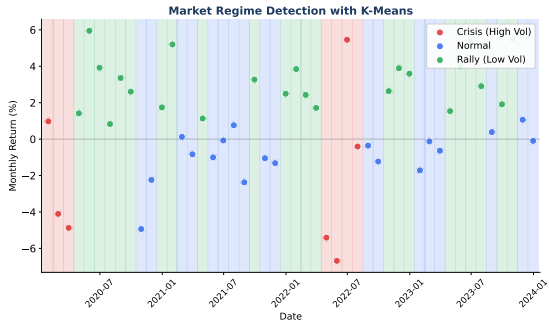


Works best for compact clusters. Hierarchical clustering (L30) handles complex shapes.

Market Regime Detection

Clustering identifies market regimes from return data

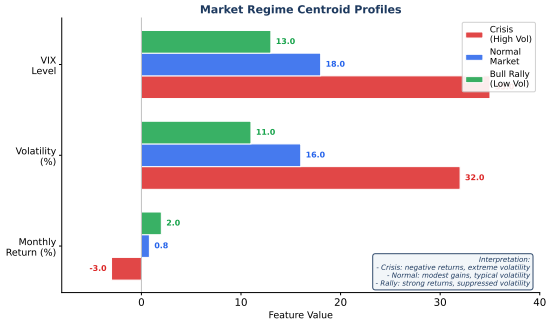
- Bull, bear, and sideways markets as clusters
- Features: rolling return, rolling volatility, volume



Self-study: market regime detection is a popular application of K-Means in finance

Regime Characteristics

What distinguishes each market regime?



Self-study: regime clusters help traders adjust strategy to current market conditions

What's Next: The Clustering Family

Next: L30 Hierarchical Clustering

K-Means gives you flat groups. But what if groups have sub-groups?

- Hierarchical clustering builds a tree of nested clusters
- No need to choose K in advance – the dendrogram shows all levels
- We will compare both methods in L30

K-Means = fast, flat partitioning. Hierarchical = slower, richer structure. Both are essential.

Hands-On Exercise (25 min)

Task: Cluster Stocks by Behavior

1. Calculate features: 1-year return, volatility, beta for 50 stocks
2. Standardize features using `StandardScaler`
3. Run K-Means with K = 2, 3, 4, 5 – plot the elbow curve
4. Choose best K using elbow + silhouette score
5. Interpret centroids: what characterizes each cluster?

Deliverable: Elbow plot + scatter plot of clusters with labels

Extension: Compare clusters to GICS sectors – do behavioral clusters align with industry classifications?

Remember: standardize first, then cluster. Use `random_state=42` for reproducibility.

Summary

Problem solved: Discovered natural groups in data without labels

Key takeaways:

1. K-Means iterates: assign points to nearest centroid $\rightarrow$ recompute centroids $\rightarrow$ repeat
2. Choose K with the elbow method (inertia) and silhouette score
3. Always standardize features before clustering
4. Interpret centroids to name and understand clusters

Finance insight:
Stock clusters often cut across sectors – behavioral groupings reveal diversification opportunities that sector labels miss.

Next lesson: L30 Hierarchical Clustering – clusters within clusters

---

K-Means: fast, simple, effective. But choose K carefully and always standardize first.