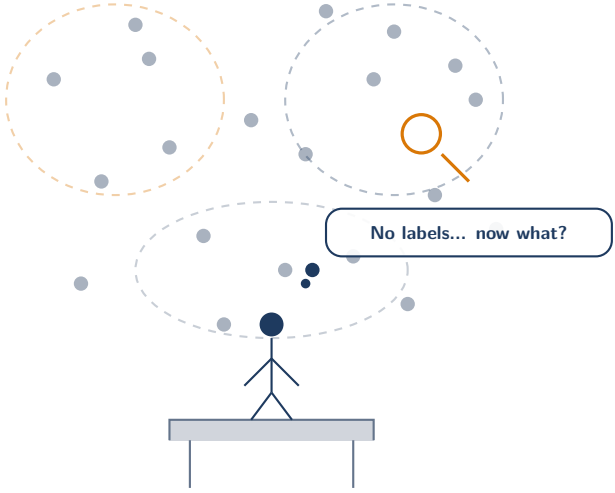
Unsupervised Learning: The Complete Picture

Data Science with Python – BSc Course

90–120 Minutes


No labels... now what?

Learning Objectives

After this session you will be able to:

1. Explain unsupervised learning vs supervised learning
2. Apply K-Means and interpret elbow/silhouette diagnostics
3. Interpret dendrograms and choose linkage methods
4. Calculate explained variance ratios and interpret PCA loadings
5. Design ML pipelines that prevent data leakage

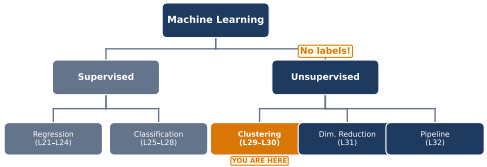
Example: By session end you can cluster 500 S&P 500 stocks into $K=5$ regime groups, keep top 3 PCs explaining 85% variance, and wrap it all in a leakage-free sklearn Pipeline.

Covers L29 K-Means, L30 Hierarchical Clustering, L31 PCA, L32 ML Pipeline

The Big Picture: No Labels, Hidden Structure

- Supervised: learn $f(X) \rightarrow Y$ from labeled data
- Unsupervised: **no target** Y — discover hidden structure in X alone
- Applications: customer segmentation, anomaly detection, feature extraction

Example: A 500-stock universe has no labels. K-Means finds 5 regime clusters; PCA compresses 50 features to 3 factors — all without a target Y. **Read the chart:** The taxonomy shows where unsupervised methods sit relative to supervised ones — clustering groups observations, dimensionality reduction compresses features.

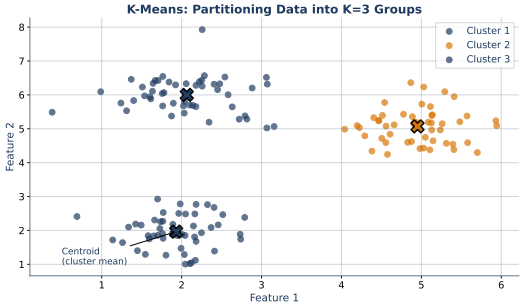


A fund manager segments 500 stocks by return patterns — no analyst labels needed

Three Core Tasks

- **Clustering** — group similar observations (K-Means, Hierarchical)
- **Dimensionality reduction** — compress features, retain variance (PCA)
- **Pipeline** — automate preprocessing, prevent leakage

Example: Cluster 500 ETFs into K=4 volatility groups, compress 50 price features to 3 PCs, and wrap both steps in a Pipeline so CV refits per fold. **Read the chart:** Each colored dot is one observation; points sharing a color share a nearest centroid. The visible coloring shows clustering in action — this is the visible-structure theme we will thread through the whole session.

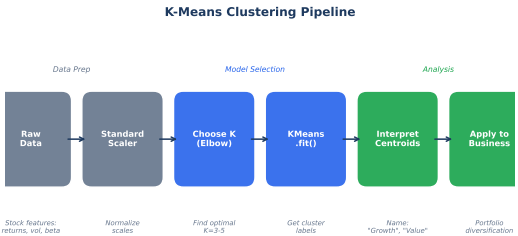


A quant desk clusters ETFs by volatility to build diversified baskets automatically

K-Means: Algorithm Overview

- Pick K initial **centroids** (cluster centre points) at random
- **Assign** each point to the nearest centroid
- **Update** each centroid to the mean of its cluster
- Repeat until assignments stop changing

Example: Three points $(1,1),(2,3),(1,4)$ belong to one cluster. New centroid $= \left(\frac{1+2+1}{3}, \frac{1+3+4}{3}\right) = (1.33, 2.67)$. **Read the chart:** The workflow shows the assign-update loop. Arrows cycle between the two steps until no point switches cluster.

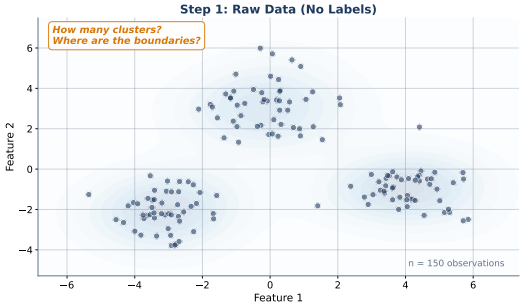


Bloomberg clusters 3,000 bonds by yield and duration using exactly this loop

Step by Step (1/2): Raw Data and Initial Assignment

- Start with raw unlabeled observations
- Randomly place K centroids in feature space
- Assign every point to the closest centroid (**Euclidean distance**: straight-line distance in feature space)

Example: K=3 with initial centroids $(1,1),(5,5),(8,2)$. Point $(3,3)$ has distances $\sqrt{8},\sqrt{8},\sqrt{26} = 2.83, 2.83, 5.10$ — a tie between centroids 1 and 2. **Read the chart:** Raw data points have no color yet. After initial assignment, each point inherits the color of its nearest centroid — that coloring is the first visible structure K-Means creates.

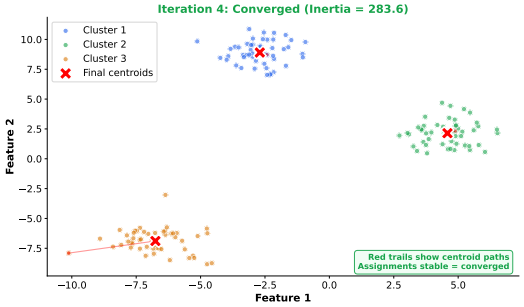


Scikit-learn's `KMeans(n_init=10)` **runs 10 random starts and keeps the best**

Step by Step (2/2): Convergence

- Centroids move toward cluster means each iteration
- Convergence: assignments no longer change between iterations
- Typical convergence in 10–20 iterations

Example: Centroid moves $(3.1, 2.8) \to (2.95, 3.02) \to (2.95, 3.02)$. Zero movement between iterations 2 and 3 $\Rightarrow$ converged. **Read the chart:** The red trail records centroid positions across iterations. By iteration 4 the trail collapses into a single point per cluster — that is convergence, and the coloured cluster assignments stabilize.



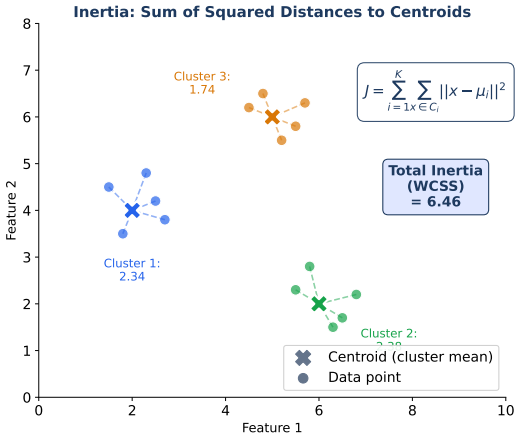
On 252 daily stock returns, K-Means typically converges in under 15 iterations

Inertia: Within-Cluster Sum of Squares

Inertia (also called **WCSS**): total squared distance from each point to its cluster centroid.

$$J = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$

Example: Cluster A with points $(1,1), (2,1)$ and centroid $(1.5,1)$: inertia $= (1-1.5)^2 + (2-1.5)^2 = 0.25 + 0.25 = 0.5$.
Read the chart: Each line segment shows one point's distance to its centroid. Inertia is the sum of all squared lengths.

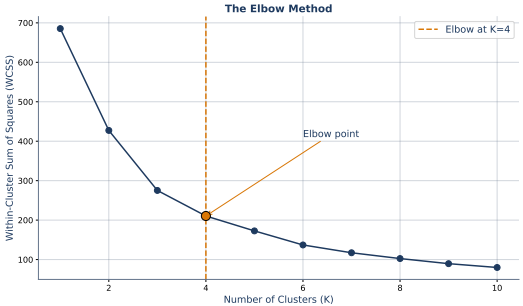


A portfolio cluster with inertia 0.02 means stocks are tightly grouped by risk profile

Choosing K: The Elbow Method

- Plot inertia vs K for $K = 1, 2, \ldots$
- Look for the "elbow" — where marginal gain flattens
- Not always clear-cut; combine with silhouette analysis

Example: Inertia drops $5.2 \to 2.1 \to 1.6 \to 1.5$ for K=1, 2, 3, 4. Biggest drop K=2→3 (0.5), tiny drop after ⇒ elbow at K=3.
Read the chart: The curve drops steeply then bends. The elbow at K=3 marks the point where adding clusters yields diminishing returns.



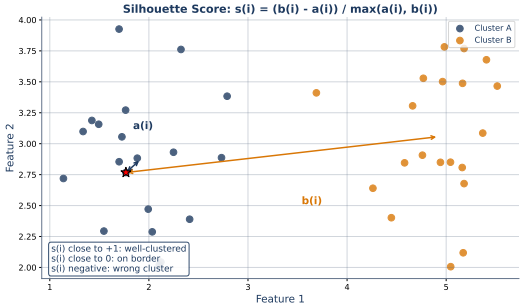
An asset manager tests K=2 to K=10 on sector ETFs; the elbow at K=4 matches known macro regimes

Silhouette Score: Cluster Quality

$$s(i) = \frac{b(i) - a(i)}{\max(a(i),\ b(i))}$$

- $a(i)$: mean distance to points in the **same** cluster
- $b(i)$: mean distance to points in the **nearest other** cluster
- Range $[-1, 1]$: higher is better; > 0.5 indicates reasonable structure

Example: Point P has $a(P)$=0.5, $b(P)$=1.2.
$s(P) = (1.2 - 0.5)/\max(0.5, 1.2) = 0.7/1.2 = 0.58$ — well-clustered. **Read the chart:** Each bar is one point's silhouette value. Wide positive bars mean clear membership; negative bars signal misassignment.

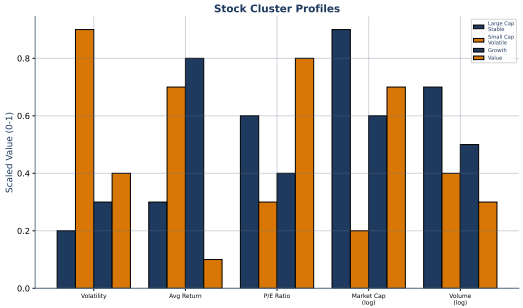


Silhouette 0.6 on stock clusters: each stock is clearly closer to its own group than the next nearest

Finance Application: Stock Cluster Profiles

- Cluster stocks by return, volatility, and volume features
- Each cluster represents a distinct risk-return profile
- Portfolio managers use clusters for diversification

Example: AAPL (return 0.12, vol 0.22), MSFT (0.10, 0.18), JPM (0.05, 0.30). AAPL and MSFT cluster together (tech-growth); JPM lands in the high-vol group. **Read the chart:** Colored groups are clusters in return-volatility space. High-return/high-volatility stocks land in one group; low-risk/low-return in another — visible structure in 2D.

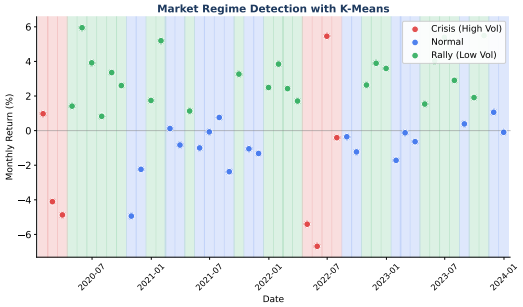


JPMorgan's LOXM system clusters order-flow patterns to optimize trade execution

Finance Application: Market Regime Detection

- Cluster time windows by volatility and return
- Identify bull, bear, and crisis regimes automatically
- Regime labels guide dynamic asset allocation

Example: In 2008 Q4 the cluster label switches from "bull" (vol < 0.15, return > 0) to "crisis" (vol > 0.35, return < -0.15). Allocation flips equities $\rightarrow$ treasuries. **Read the chart:** Colors mark detected regimes over time. Red periods are high-volatility crisis regimes; green periods are calm bull markets.

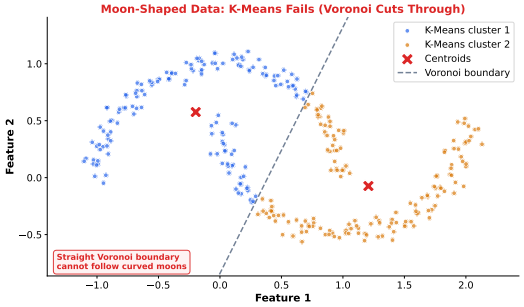


A hedge fund shifts from equities to treasuries when the cluster label switches to "crisis"

K-Means Limitations

- Assumes spherical, equally sized clusters
- Struggles with non-convex or elongated shapes
- Sensitive to outliers and initialization

Example: On 300 moon-shaped crypto-style returns, K-Means with $K=2$ reports inertia 45 but splits each crescent in half — the straight Voronoi boundary cannot follow the curves. **Read the chart:** Colored dots show K-Means cluster assignments on moon-shaped data. The dashed Voronoi boundary between the two centroids (red X) cuts straight across — slicing each natural crescent into two halves.

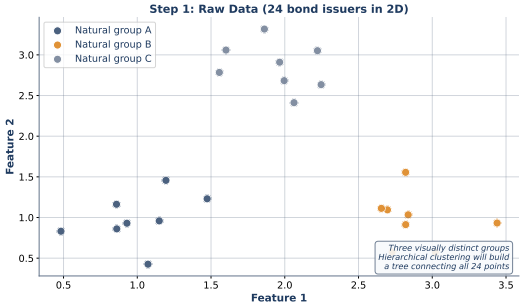


Cryptocurrency returns often form non-spherical clusters — hierarchical or DBSCAN handles them better

Hierarchical Clustering: Bottom-Up Approach

- **Agglomerative** (bottom-up): starts with each point alone, merges upward
- Repeatedly merge the two closest clusters
- Result: a **dendrogram** (tree diagram showing all merge levels)

Example: 5 bond issuers start as 5 singletons. First merge joins closest pair (e.g. AAA1,AAA2) → 4 clusters. After 4 merges total, all issuers join into 1 root cluster. **Read the chart:** The 2D scatter shows 24 points forming three visually distinct natural groups (navy / amber / slate). Hierarchical clustering will connect every point into one tree, preserving this visible structure at different cut levels.

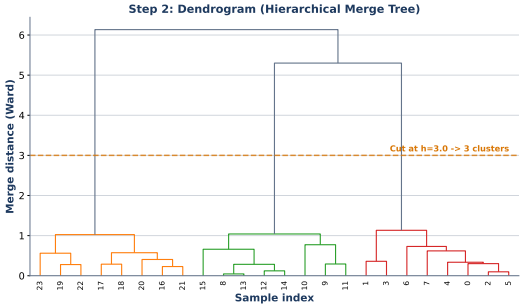


A credit analyst clusters 200 issuers by spread curves — the dendrogram reveals natural tiers

Hierarchical Output: The Dendrogram Tree

- Each merge is a horizontal bar whose **height = merge distance**
- Bottom: individual points; Top: single root cluster
- Horizontal cut at height h yields a partition (coloured below the cut)

Example: Using Ward linkage on the 24 scattered points from the prior slide, a cut at h=3.0 yields the 3 natural groups (navy / amber / slate). **Read the chart:** Bottom leaves are individual points; vertical lines join them as merges happen. The orange dashed line at h=3.0 is the cut; the colouring below it reproduces the three visible groups from slide 15.



Same 24 bond issuers — the tree preserves the visible 3-group structure for any cut height

K-Means vs Hierarchical Clustering

- K-Means: fast, needs K upfront, spherical assumption
- Hierarchical: slower, produces full tree, flexible shapes
- Both need feature scaling; neither handles high dimensions well alone

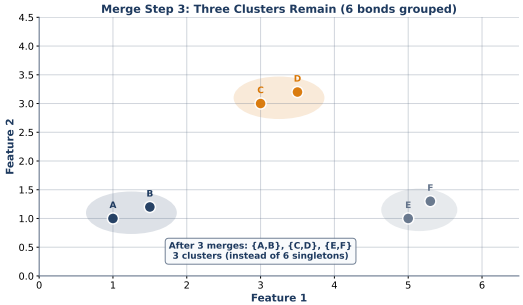
Example: For 10 000 intraday trades K-Means takes $\sim$0.3 s; hierarchical takes $\sim$4.2 min. For 200 bonds, hierarchical takes $\sim$0.1 s and returns the full tree. **Read the chart:** The horizontal bar chart scores each method on speed, K-selection flexibility, hierarchy view, and scalability. K-Means (navy) wins speed and large-data axes; Hierarchical (amber) wins on "no-K-needed" and hierarchy.

For 10,000 trades use K-Means; for 200 bonds where hierarchy matters, use agglomerative

Agglomerative Merging Steps

- Step 1: Compute pairwise distance matrix
- Step 2: Merge closest pair; update distances
- Step 3: Repeat until one cluster remains

Example: 6 bonds A–F with closest pair (A, B) at distance 0.3 merges first. The distance matrix drops from 6×6 to 5×5; continue until 1 cluster remains (5 total merges). **Read the chart:** Points A–F are colored by their cluster membership after 3 merges. Three ellipses enclose the three remaining clusters {A,B}, {C,D}, {E,F} — the visible-structure partition at this cut height.

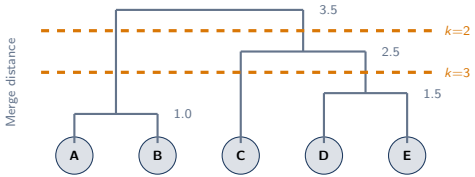


The merge order is recorded in the dendrogram — cutting at any height yields a partition

Cutting the Dendrogram

- Cut at different heights → different number of clusters
- Higher cut = fewer clusters
- The dendrogram lets you choose k *after* fitting

Example: Cut at h=2 yields 3 clusters $\{A, B\}, \{C\}, \{D, E\}$. Cut at h=3 yields 2 clusters $\{A, B\}, \{C, D, E\}$. One tree, many partitions.



A risk manager cuts the bond dendrogram at 3 clusters for reporting but at 7 for granular hedging

Linkage Methods: Choosing a Merge Criterion

- **Single**: min distance — finds elongated shapes, sensitive to noise
- **Complete**: max distance — compact clusters, sensitive to outliers
- **Ward**: minimizes variance increase — most popular for balanced clusters

Example: On 200 currency pairs, *single* linkage produces 1 mega-cluster plus outliers. *Ward* produces 4 balanced clusters matching regional groups (USD, EUR, EM-Asia, LATAM).
Read the chart: The decision matrix maps data characteristics (rows: noise level, shape, scalability) to recommended linkage (columns). Ward sits in the "balanced default" row — pick it unless you have a specific reason not to.

Linkage Method Selection Guide

Linkage	Formula	Best For	Caution
Ward	$\min \Delta \sigma^2$	Compact, spherical equal-sized clusters	Assumes spherical
Complete	$\max\{d(a,b)\}$	Tight clusters outlier sensitive	May miss elongated
Average	$\frac{1}{mn}\sum d$	Balanced approach robust	Slower convergence
Single	$\min\{d(a,b)\}$	Elongated clusters chains	Chaining effect

Default: Use Ward for most applications. Use Single only for elongated/chain-like data.

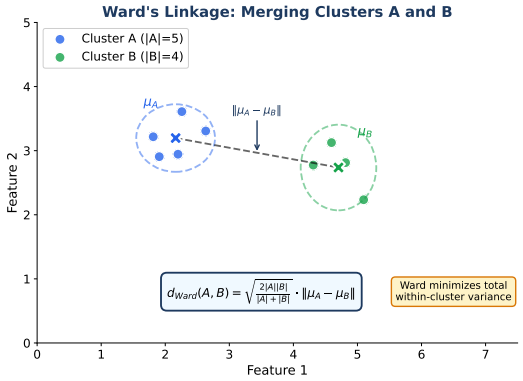
Ward linkage on currency pairs produces the most balanced clusters for FX portfolio grouping

Ward Linkage Formula

$$\Delta(C_i, C_j) = \frac{n_i\, n_j}{n_i + n_j} \|\mu_i - \mu_j\|^2$$

- Merges the pair causing the smallest variance increase
- Produces compact, spherical clusters — the hierarchical analog of K-Means

Example: Cluster A (n_A=3, mean 0.08), cluster B (n_B=2, mean 0.12). $\Delta = \frac{3\cdot 2}{3+2} \cdot (0.08-0.12)^2 = 1.2 \cdot 0.0016 = 0.00192$.
Read the chart: The visualization shows two candidate merges; the one with lower Δ wins. Color intensity encodes variance contribution.



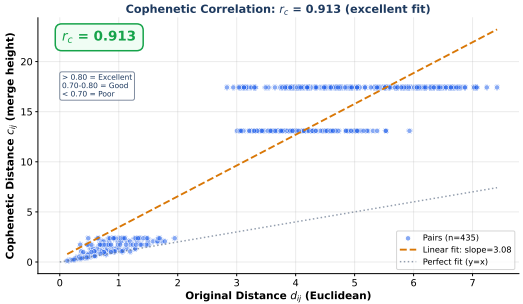
Ward merges two stock clusters only if doing so adds the least variance to the combined group

Cophenetic Correlation: Dendrogram Quality

- **Cophenetic correlation**: correlation between actual pairwise distances and the heights at which pairs merge in the dendrogram
- Values > 0.7 suggest a reliable hierarchical structure

Example: On 30 equity clusters r_c=0.82>0.70 confirms the dendrogram faithfully encodes return distances. Below 0.70, consider a different linkage or reject the tree. **Read the chart:**

Each blue dot is one pair of observations. X-axis = true Euclidean distance, Y-axis = dendrogram merge height. Tight scatter along the dashed perfect-fit line means high fidelity; the annotated r_c value quantifies it.

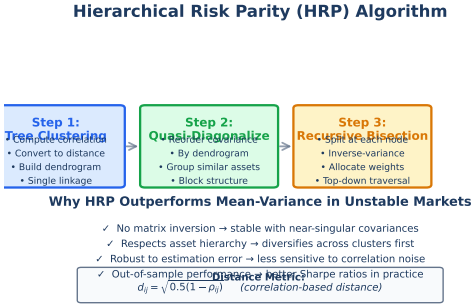


Cophenetic 0.82 on equity clusters confirms the dendrogram faithfully represents return distances

Hierarchical Risk Parity (HRP): Algorithm Steps

- Step 1: Hierarchical clustering on correlation matrix
- Step 2: **Quasi-diagonalize**: reorder the covariance matrix so correlated assets sit next to each other
- Step 3: Recursive bisection to allocate weights

Example: 20 assets with correlations 0.1–0.9. Step 1 dendrogram groups AAPL, MSFT, GOOGL (tech, $\rho \approx 0.7$). Step 2 reorders covariance so tech sits adjacent. Step 3 allocates $\sim$45% to the tech sub-tree. **Read the chart:** The three coloured boxes show the sequential HRP pipeline: tree clustering (blue) $\rightarrow$ quasi-diagonalize (green) $\rightarrow$ recursive bisection (amber). Below, four checkmarks list HRP's advantages; the bottom box anchors the distance metric $d_{ij} = \sqrt{0.5(1-\rho_{ij})}$.



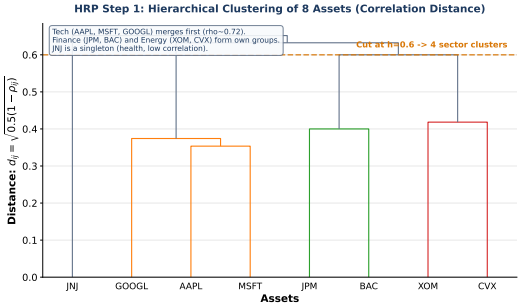
HRP avoids inverting the covariance matrix — more stable when correlations shift in a crisis

Finance Application: HRP Portfolio

- Hierarchical Risk Parity avoids inverting the covariance matrix
- Outperforms Markowitz in out-of-sample tests (de Prado, 2016)
- Naturally diversifies across correlation-based clusters

Example: 2005–2020 backtest on 10 sector ETFs: HRP Sharpe 1.2 vs Markowitz 0.7 vs equal-weight 0.9. HRP wins without requiring a full covariance inversion. **Read the chart:**

Dendrogram of 8 example assets clustered by correlation distance. Tech (AAPL, MSFT, GOOGL) merges first at low distance; Finance (JPM, BAC) and Energy (XOM, CVX) form their own groups. The orange cut line at h=0.6 produces 4 sector clusters that HRP then weights inversely by variance.

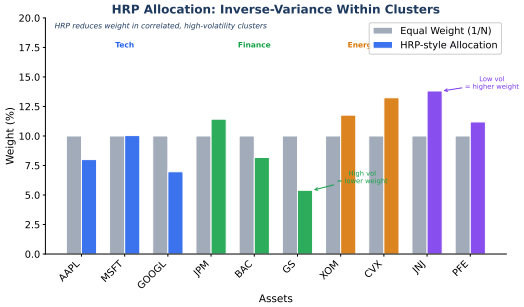


HRP is the most cited unsupervised-learning application in modern portfolio theory

HRP: Weight Allocation Across Clusters

- Recursive bisection splits the dendrogram into left/right sub-trees
- Weights are inversely proportional to sub-tree variance
- Final weights reflect both correlation structure and individual risk

Example: Utilities sub-tree variance σ_U^2=0.08, tech σ_T^2=0.22. $w_U = (1/0.08)/\big((1/0.08) + (1/0.22)\big) = 12.5/17.0 = 0.735$ (73.5% utilities, 26.5% tech). **Read the chart:** Bar heights show final weights per asset. Assets in low-variance sub-trees receive larger allocations — the tree structure drives every weight.

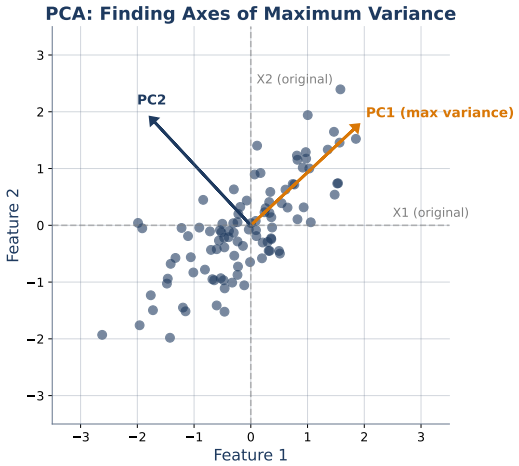


A 20-asset HRP portfolio rebalances monthly, using fresh dendrograms each time

The Curse of Dimensionality

- More features $\neq$ better models — high dimensions cause sparsity and overfitting
- PCA compresses features while retaining most variance

Example: A 50-feature equity model on 500 observations gives only 10 points per feature — far too sparse. PCA reduces to 5 PCs explaining 88% variance before downstream modelling.
Read the chart: The concept diagram shows many input features on the left compressing into a few principal components on the right. PCA is the arrow: high-dimensional feature space $\rightarrow$ low-dimensional PC space preserving variance.

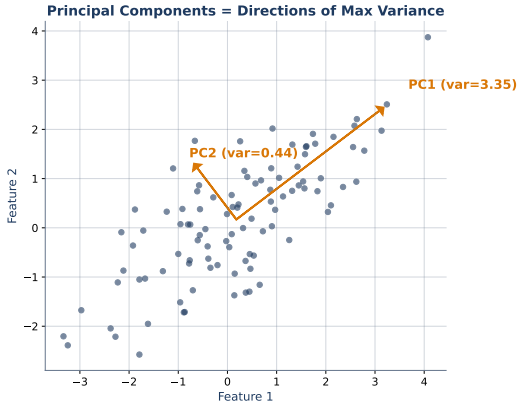


A 50-feature equity model overfits on 500 observations — PCA reduces to 5 factors first

Principal Components: Directions of Maximum Variance

- PC1: direction of greatest variance in the data
- PC2: orthogonal to PC1, captures next most variance
- Each PC is a linear combination of original features

Example: For daily S&P 500 returns, PC1 loadings are AAPL 0.12, MSFT 0.11, JPM 0.10 — all positive. PC1 is the "market factor" capturing broad up/down moves. **Read the chart:** The arrows show PC1 and PC2 overlaid on the data cloud. PC1 aligns with the longest axis of the ellipse; PC2 is perpendicular and shorter (less variance).

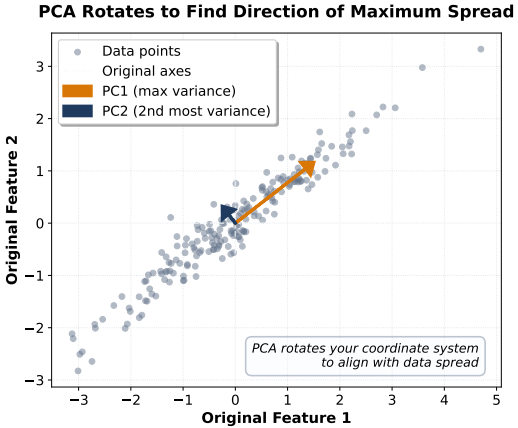


PC1 on daily S&P 500 returns is the "market factor" — it captures broad up/down moves

PCA Intuition

- Rotate axes to align with data spread
- Project onto top k axes, discard the rest
- Reconstruction error = discarded variance

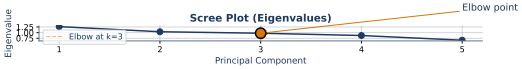
Example: 2D data with eigenvalues λ_1=5.1 (PC1 direction) and λ_2=0.3 (PC2). Keep PC1 only $\Rightarrow$ retain $5.1/5.4 = 94\%$ variance; drop PC2 as noise. **Read the chart:** Grey dots are centered 2D data. The orange arrow (PC1) points along the longest axis of the elliptical cloud — the direction of maximum spread. The navy arrow (PC2) is perpendicular and noticeably shorter, reflecting its smaller captured variance.



Rotating 30 stock-return features into 3 PCs discards noise while keeping systematic risk

Scree Plot and Explained Variance Ratio

$$\text{EVR}_k = \frac{\lambda_k}{\sum_{j=1}^{p} \lambda_j}$$



- Plot eigenvalues (or EVR) in descending order
- Look for an "elbow" where additional PCs add little variance
- Common threshold: cumulative EVR $\geq 80\%$ or 90%

Example: Eigenvalues $[5.1, 2.3, 1.0, 0.4, 0.2]$, total 9.0. $\text{EVR}_1 = 5.1/9.0 = 56.7\%$; cumulative 2 PCs $= 82.2\%$; 3 PCs $= 93.3\%$. **Read the chart:** Bars show individual EVR per component; the line shows cumulative. The elbow at PC3 means 3 components capture most variance.

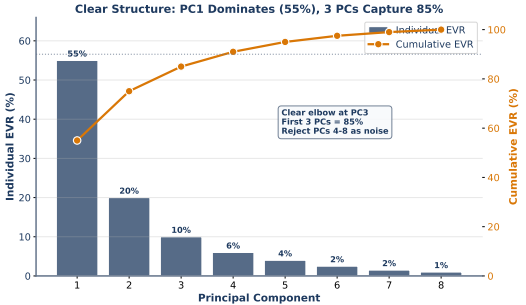
On 50 equity features, 3 PCs explain 85% of variance — the other 47 are mostly noise

Interpreting PCA: Practical Guidelines

- Standardize features first (mean 0, std 1)
- Interpret PCs through loadings, not raw coefficients
- Domain knowledge is essential for labeling components

Example: Clear structure: PC1 alone captures 55%, 3 PCs reach 85% (elbow at PC3). Noisy data: all 8 PCs sit near 15% each — no elbow, no compression benefit. **Read the chart:**

Navy bars show individual EVR per PC; the amber line shows cumulative EVR with a dotted 90%-threshold. PC1 dominates at 55%, PC2 adds 20%, then diminishing returns — a textbook "clear structure" signal.

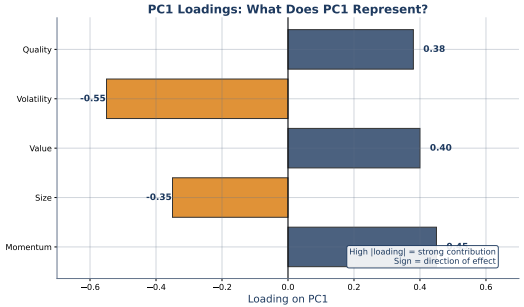


A quant labels PC1 "market" and PC2 "value-vs-growth" based on loading patterns

Loadings and Biplots

- Loadings = correlations between original features and PCs
- Biplot: overlay scores (observations) with loading arrows (features)
- Arrow direction shows which features drive each PC

Example: Bank-stock biplot: `NIM` (net interest margin) arrow points NE along PC1 (banking-earnings factor); `NPL ratio` arrow points SW (opposite direction). Long arrows dominate their PC. **Read the chart:** Dots are observations projected onto PC1/PC2. Arrows are feature loadings. Long arrows that point along PC1 are strong drivers of that component; arrows pointing opposite directions are anti-correlated features.

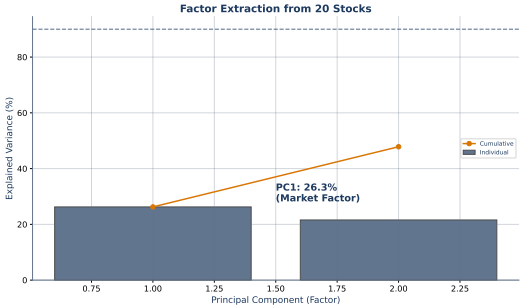


A biplot of bank stocks reveals that "NIM" and "loan growth" arrows point opposite to "NPL ratio"

Finance Application: Factor Extraction

- Apply PCA to a panel of stock returns
- PC1 $\approx$ market factor; PC2 $\approx$ sector/size factor
- Replaces ad-hoc factor construction with data-driven factors

Example: 30-stock panel: PC1 loadings AAPL 0.45, MSFT 0.41, JNJ 0.12 $\Rightarrow$ PC1 is the tech factor. PC2 flips sign between tech ($+0.30$) and utilities (-0.25) $\Rightarrow$ sector-tilt factor. **Read the chart:** PC1 loadings are uniformly positive across all stocks — the market factor. PC2 loadings split positive (tech) and negative (utilities), revealing a sector tilt.

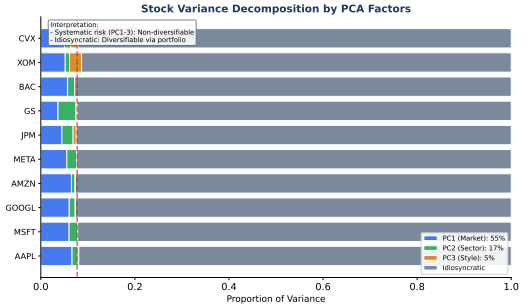


BlackRock uses PCA-based factors alongside Fama-French for multi-factor portfolio construction

Finance Application: Portfolio Variance Decomposition

- Decompose portfolio variance into PC contributions
- Identify concentration risk: is 90% of variance driven by one factor?
- Risk managers use PCA to monitor factor exposure

Example: A 10-asset fund with PC1 variance share 88%, PC2 7%, PCs 3–10 totalling 5%. The fund is effectively a 1-factor market bet disguised as diversified. **Read the chart:** The stacked bar shows each PC's share of total portfolio variance. If PC1 dominates, the portfolio is effectively a single-factor bet.

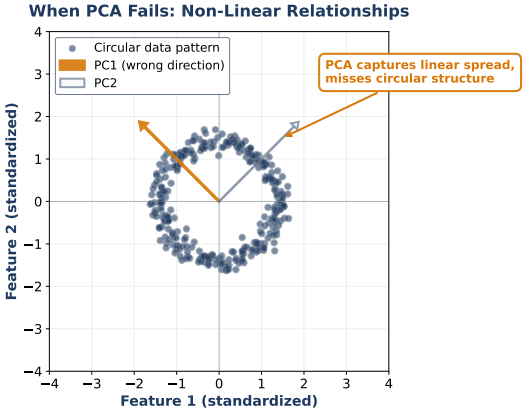


A risk officer flags a "diversified" fund whose variance is 88% PC1 — it is a hidden market bet

PCA Limitations

- Linear only — cannot capture nonlinear relationships
- Sensitive to feature scaling (always standardize first)
- Components are orthogonal, which may not match domain structure

Example: On an option portfolio with convex payoffs, PCA reconstruction error is $\sim 2\times$ larger than on linear equity returns. Use Kernel PCA or autoencoders for curved manifolds. **Read the chart:** PCA projects the curved manifold onto a flat plane, losing the nonlinear structure entirely. The reconstruction error is large where curvature is strongest.

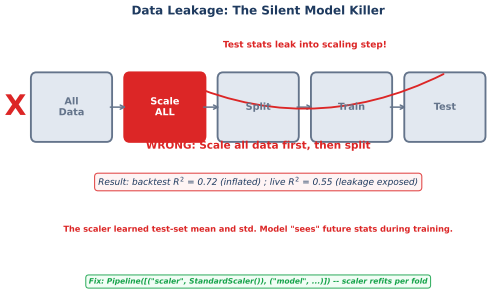


Option-pricing features have nonlinear payoff structures — PCA misses the curvature

Data Leakage: The Silent Model Killer

- Leakage: test information bleeds into training
- Scaling before train/test split uses test-set mean and std — the model "peeks" at future data
- Result: inflated accuracy that collapses in production

Example: Fitting `StandardScaler` on train+test combined inflates backtest R^2 from 0.55 to 0.72 — a 17-point leakage bonus that vanishes in production. **Read the chart:** The "WRONG" workflow (top) starts with *all data*, scales ALL of it, then splits. The curved red arrow traces test statistics flowing back into the scaling step — that is the leak. The green caption at the bottom gives the fix.

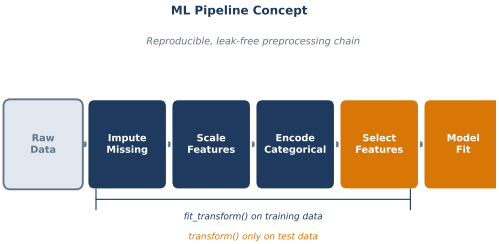


A trading model with leakage shows 15% annual alpha in backtest but 0% live — the leakage inflated returns

sklearn Pipeline: Encapsulated Workflow

- Chain preprocessing and modeling into one object
- `Pipeline([('scaler', StandardScaler()), ('model', KMeans())])`
- Guarantees correct fit/transform order during cross-validation

Example: `Pipeline([('scaler', StandardScaler()), ('pca', PCA(3)), ('km', KMeans(5))])` runs scaling → PCA → K-Means on every fold, refitting the scaler each time — leakage eliminated by construction. **Read the chart:** The pipeline diagram shows data flowing left-to-right through each step. `fit` is called on training data; `transform` on test data — no leakage possible.



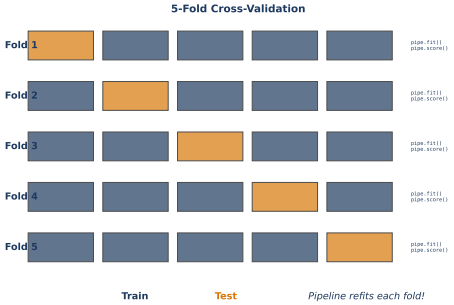
Every production model at Goldman Sachs is wrapped in a pipeline to prevent leakage by construction

Cross-Validation: Robust Evaluation

$$\text{CV Score} = \frac{1}{K}\sum_{k=1}^{K} \text{Score}_k$$

- Split data into K folds; train on $K-1$, test on 1
- Average score across folds reduces variance of estimate
- Pipeline ensures no leakage across folds

Example: 5-fold CV on 500 bonds: 5 folds of 100 each, train on 400, validate on 100, rotate. The scaler refits on each fold's 400 training bonds only. **Read the chart:** Each row is one fold. Blue = training, orange = test. The pipeline refits from scratch on each training set — no information carries over.

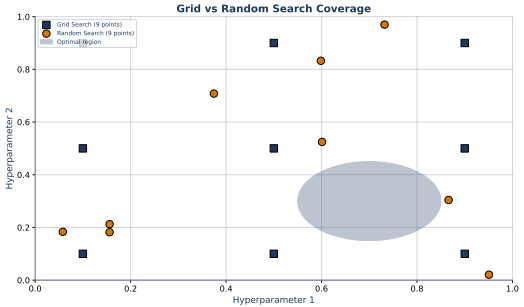


5-fold CV on a credit-scoring pipeline gives a Sharpe estimate with standard error ± 0.12

Hyperparameter Tuning: Grid vs Random Search

- Grid search: exhaustive, $O(n^d)$ — expensive in high dimensions
- Random search: samples parameter space, often finds good regions faster
- Random dominates when only a few parameters matter (Bergstra & Bengio, 2012)

Example: Grid $5 \times 4 = 20$ combos. Random draws 10 samples. Random often lands within 1% of grid's best score at half the compute cost. **Read the chart:** Grid points form a regular lattice; random points scatter. With two parameters but only one that matters, random search covers more of the important axis.

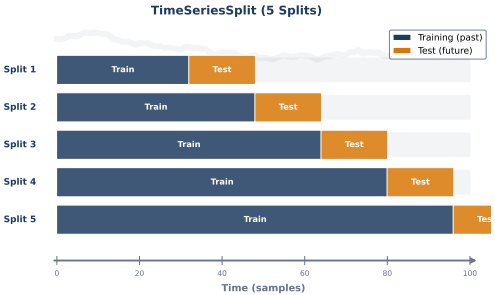


A quant tunes regularization strength and learning rate — random search finds optimal α 3x faster

Finance: TimeSeriesSplit for Temporal Data

- Standard K-fold shuffles time — violates temporal order
- TimeSeriesSplit: always train on past, test on future
- Expanding window mimics real deployment conditions

Example: A momentum strategy backtested with shuffled 5-fold CV shows 20% alpha. With TimeSeriesSplit the same strategy shows 3% — shuffling was leaking future returns backward.
Read the chart: Each row expands the training window forward in time. The test window always sits to the right of training — no future data leaks backward.



A momentum strategy backtested with shuffled CV shows 20% alpha; TimeSeriesSplit shows 3%

Clustering vs PCA: Side by Side

	Clustering	**PCA**
Goal	Group similar observations	Compress features
Output	Cluster labels $(1, 2, \ldots K)$	Principal components
Finance use	Regime detection, HRP	Factor extraction, risk
Key metric	Silhouette / inertia	Explained variance ratio

- Both require standardized features
- Often combined: PCA to reduce dimensions, then cluster in PC space

Example: A risk desk runs PCA on 50 equity features (retaining 5 PCs at 88% EVR), then K-Means with K=4 on those 5 PCs to detect regimes — combining compression and grouping.

A risk desk runs PCA on 50 equity features, then clusters the top 5 PCs to detect regimes

Formula Reference: Clustering

Inertia (WCSS)

$$J = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$

Centroid Update

$$\mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i$$

Euclidean Distance

$$d(x, y) = \sqrt{\sum_{j=1}^{p} (x_j - y_j)^2}$$

Silhouette

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Ward Linkage

$$\Delta(C_i, C_j) = \frac{n_i n_j}{n_i + n_j} \|\mu_i - \mu_j\|^2$$

Correlation Distance

$$d_{\text{corr}}(x, y) = 1 - \rho(x, y)$$

Standardization

$$z = \frac{x - \bar{x}}{s}$$

All clustering formulas on one slide — save this for exam review

Formula Reference: PCA and Cross-Validation

Covariance Matrix

$$\Sigma = \frac{1}{n-1}(X - \bar{X})^T (X - \bar{X})$$

Eigenvalue Equation

$$\Sigma v_k = \lambda_k v_k$$

Explained Variance Ratio

$$\text{EVR}_k = \frac{\lambda_k}{\sum_{j=1}^{p} \lambda_j}$$

Cumulative EVR

$$\text{CEVR}_m = \sum_{k=1}^{m} \text{EVR}_k$$

Projection

$$Z = XW_k \quad (n \times k)$$

CV Score

$$\text{CV} = \frac{1}{K} \sum_{k=1}^{K} \text{Score}_k$$

PCA formulas and cross-validation — save this for exam review

Common Misconceptions (1–3)

Myth 1: K-Means is always the best clustering algorithm

Reality: K-Means assumes spherical clusters. Hierarchical clustering handles arbitrary shapes and provides a dendrogram for exploration.

Myth 2: More clusters are always better

Reality: Inertia always decreases with more clusters. Use the elbow method and silhouette score to find the meaningful number.

Myth 3: PCA finds the most important features

Reality: PCA finds directions of maximum variance. High-variance directions are not necessarily the most predictive or important.

Example: 2008 credit default data: K-Means forces issuers into spherical clusters and misses the AAA/AA/A/BBB hierarchy. Hierarchical clustering recovers those nested tiers naturally.

A stock's highest-variance feature is often price level, not the return signal you care about

Common Misconceptions (4–6)

Myth 4: Hierarchical clustering is always better than K-Means

Reality: Hierarchical is $O(n^2)$ in memory. For large datasets K-Means with $O(nK)$ complexity is far more practical.

Myth 5: Retaining 90% variance means the remaining 10% is noise

Reality: The discarded variance may contain task-relevant signal. Always validate downstream task performance after dimensionality reduction.

Myth 6: PCA works fine without scaling

Reality: Without standardization, features with larger scales dominate the covariance matrix and bias the principal components.

Example: Market cap ($10 bn) vs P/E ratio (∼15): variance ratio 10^8:1. Without scaling, PCA treats market cap as "the signal" and P/E as noise — that is the bias.

Market cap in millions dwarfs P/E ratios around 15 — without scaling, PCA sees only market cap

Self-Assessment (1/2)

Q1: Calculate Inertia
Points: $(1, 2)$, $(3, 4)$, $(2, 3)$. Centroid: $(2, 3)$.
Compute $J = \sum \|x_i - \mu\|^2$.

Self-Assessment (1/2)

Q1: Calculate Inertia
Points: $(1, 2)$, $(3, 4)$, $(2, 3)$. Centroid: $(2, 3)$.
Compute $J = \sum \|x_i - \mu\|^2$.

$J = (1-2)^2+(2-3)^2 + (3-2)^2+(4-3)^2 + (2-2)^2+(3-3)^2 = 2 + 2 + 0 = 4$

Q2: Calculate Silhouette
$a(i) = 3$, $b(i) = 8$. Compute $s(i)$ and interpret.

Self-Assessment (1/2)

Q1: Calculate Inertia
Points: $(1,2)$, $(3,4)$, $(2,3)$. Centroid: $(2,3)$.
Compute $J = \sum \|x_i - \mu\|^2$.

$J = (1-2)^2+(2-3)^2 + (3-2)^2+(4-3)^2 + (2-2)^2+(3-3)^2 = 2+2+0 = 4$

Q2: Calculate Silhouette
$a(i) = 3$, $b(i) = 8$. Compute $s(i)$ and interpret.

$s(i) = \frac{8-3}{\max(3,8)} = \frac{5}{8} = 0.625$ — well-clustered (close to 1).

Exam tip: show every substitution step — partial credit depends on visible work

Self-Assessment (2/2)

Q3: Choose Number of PCs
Eigenvalues: [5.1, 2.3, 0.8, 0.5, 0.3]. Total = 9.0.
How many PCs for $\geq 80\%$ explained variance?

Self-Assessment (2/2)

Q3: Choose Number of PCs
Eigenvalues: [5.1, 2.3, 0.8, 0.5, 0.3]. Total = 9.0.
How many PCs for $\geq 80\%$ explained variance?

EVR: 56.7%, 25.6%, 8.9%, 5.6%, 3.3%. Cumulative: 56.7%, 82.2%, ... $\Rightarrow$ **2 PCs** for 82.2%.

Q4: Spot the Leakage
A colleague scales the entire dataset, then splits into train and test. What is wrong?

Self-Assessment (2/2)

Q3: Choose Number of PCs
Eigenvalues: [5.1, 2.3, 0.8, 0.5, 0.3]. Total = 9.0.
How many PCs for $\geq 80\%$ explained variance?

EVR: 56.7%, 25.6%, 8.9%, 5.6%, 3.3%. Cumulative: 56.7%, 82.2%, ... $\Rightarrow$ **2 PCs** for 82.2%.

Q4: Spot the Leakage
A colleague scales the entire dataset, then splits into train and test. What is wrong?

The scaler used test-set statistics during fitting — data leakage. Fit the scaler on training data only, then transform test data.

In a trading backtest, leakage means your scaler "saw" tomorrow's volatility — results are worthless

What Comes Next

Lesson companions for deeper study:

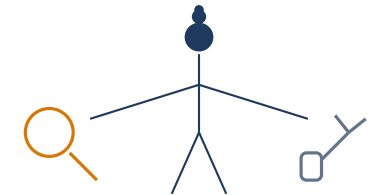
- L29: K-Means Clustering — full algorithm, sklearn implementation
- L30: Hierarchical Clustering — linkage, dendrograms, HRP
- L31: PCA — eigenvalues, loadings, biplots, factor models
- L32: ML Pipeline — leakage, pipelines, cross-validation, tuning

Coming up: Module 7 — Deep Learning (L33–L36)

- Neural network fundamentals and backpropagation
- CNNs, RNNs, and practical training workflows

This overview connects the dots — the individual lessons provide the depth

Find the structure. Compress the noise. Automate the rest.



Hidden structure revealed