Lesson 31: PCA Dimensionality Reduction

Data Science with Python – BSc Course

Data Science Program

BSc Course

45 Minutes

Previously on L30: Hierarchical Clustering

What We Learned:

- Dendrograms reveal nested cluster structure at every scale
- Ward linkage minimizes within-cluster variance
- Correlation-based clustering groups co-moving assets
- HRP uses the hierarchy for portfolio allocation

We can cluster observations and features. But what if we have 100 correlated features?

The Curse of Too Many Features

Situation: You have 100 features for each stock.

What goes wrong?

- Many features are **correlated** (P/E and earnings yield, volatility and beta)
- Models become **slow** – more features = more computation
- Noise overwhelms signal – **overfitting** with sparse high-dimensional data
- Visualization impossible beyond 3D

Complication: We can't just drop features randomly – we might lose important information hidden in the correlations.

Question: Can we compress 100 features into a handful while keeping the essential patterns?

High dimensions: slow, noisy, overfitting. We need a principled way to compress.

Learning Objectives

The Problem: 100 features, many correlated. Models are slow, noisy, and overfit. Can we compress without losing the big picture?

After this lesson, you will be able to:

- Explain PCA as finding new axes of maximum variance
- Apply PCA with sklearn (StandardScaler $\rightarrow$ PCA $\rightarrow$ fit_transform)
- Interpret scree plots and explained variance ratios
- Use loadings and biplots to understand what components mean

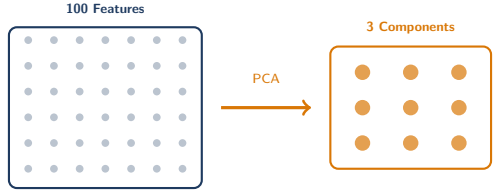
---

Finance Application: Factor discovery, risk decomposition, noise reduction

The Photo Compression Idea

Analogy: 100-Megapixel Photo → JPEG

- Raw photo stores every pixel – massive, redundant
- JPEG keeps shapes and colors, drops tiny details
- Result: 10× smaller, still looks great



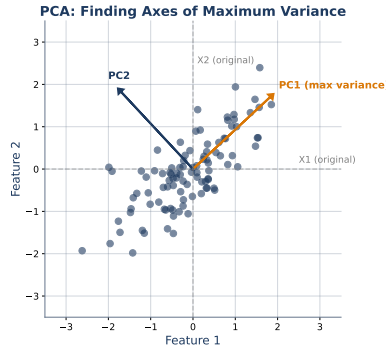
PCA does this for data: compress correlated features into fewer components.

PCA = JPEG for your feature matrix. Lose some detail, keep the essence.

PCA Concept: Finding New Axes

Rotating the Coordinate System

- Original axes may be correlated – PCA finds new axes aligned with spread
- PC1: maximum variance direction. PC2: perpendicular, next most.

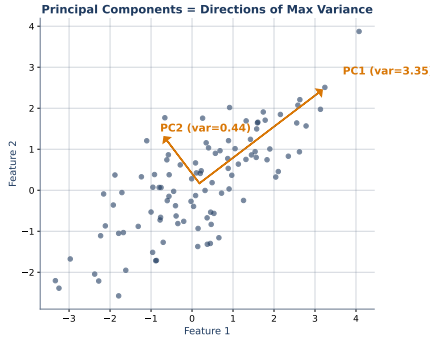


PCA rotates the coordinate system to align with the data's natural directions

Principal Component Directions

Each PC = weighted combination of original features

- PC1 = $w_1 \times$ Return $+ w_2 \times$ Volatility $+ \ldots$
- Weights maximize variance; all PCs are orthogonal (uncorrelated)

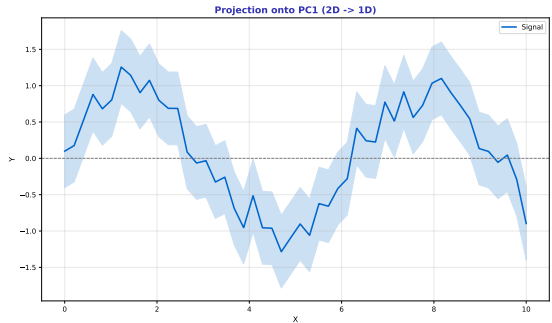


Orthogonal = perpendicular = uncorrelated. Each PC captures unique information.

Projection: 2D to 1D

Dimensionality Reduction in Action

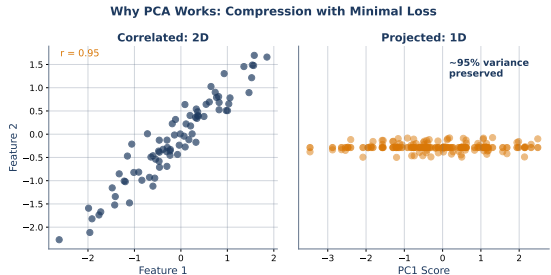
- Project each point onto PC1 – collapse 2D to 1D
- If PC1 captures 90% of variance, we keep 90% of information



Projection = shadow of data on PC1. More variance on PC1 = better shadow.

Correlation Is Compressible

- Correlated features carry overlapping information
- PCA exploits this redundancy – fewer components, same signal

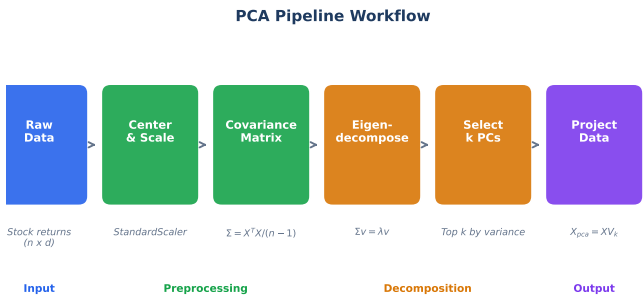


High correlation = high compressibility. PCA works best on correlated features.

Self-Study: PCA Workflow Overview

End-to-End Steps

- Standardize → Covariance matrix → Eigendecomposition → Select k → Project
- Each step has a clear purpose in the pipeline

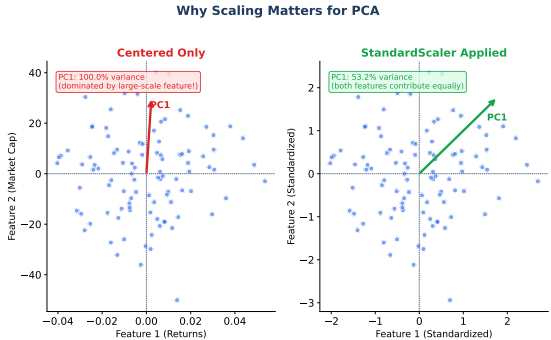


Self-study: Complete PCA pipeline from raw data to reduced dimensions

Self-Study: Centering vs Scaling

Why Standardize Before PCA?

- Centering: subtract mean (PCA assumes zero-centered data)
- Scaling: divide by std (critical when features differ in units)

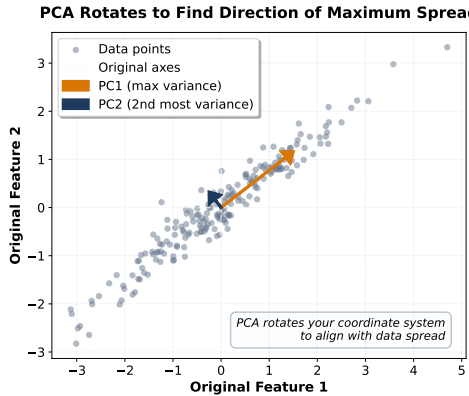


Self-study: Always use StandardScaler before PCA unless features share the same scale

Self-Study: PCA Intuition

Another Way to See It

- PCA finds directions where data "stretches" most – like the longest axis of an ellipse

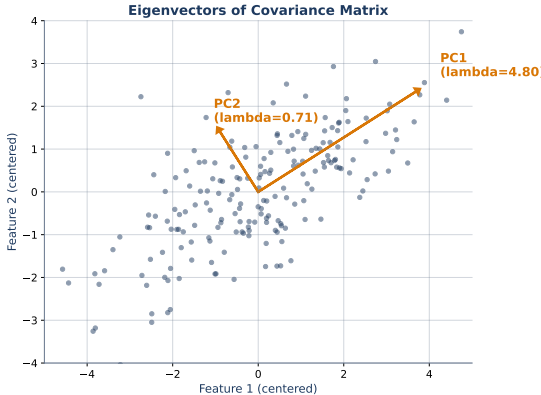


Self-study: PCA = finding the natural axes of your data ellipse

Self-Study: Eigenvectors of the Covariance Matrix

The Math Connection

- Covariance matrix Σ encodes pairwise feature relationships
- Its eigenvectors = PC directions; eigenvalues = variance per PC

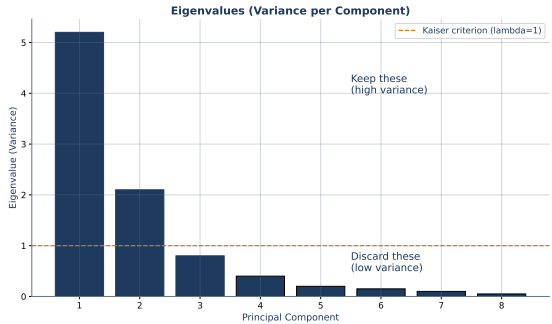


Self-study: Eigenvectors of Σ = principal component directions

Self-Study: Eigenvalues as Variance

Eigenvalue = Variance Along That PC

- Large eigenvalue = keep it. Small = safe to drop.
- Sum of all eigenvalues = total variance in dataset.

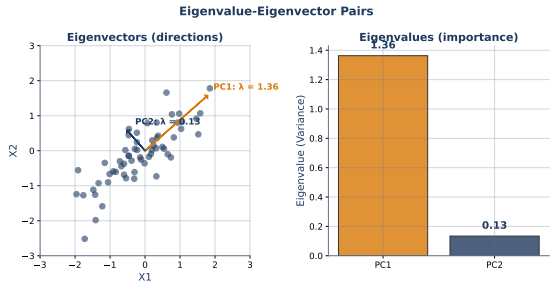


Self-study: Eigenvalue ratio = fraction of total variance explained by that PC

Eigenvectors: The Data's Natural Axes

Where Do PC Directions Come From?

- Covariance matrix encodes all feature relationships
- Its eigenvectors = PC directions; eigenvalues = variance per PC



Eigenvector = direction of stretch. Eigenvalue = amount of stretch. Larger = more important.

PCA in sklearn

Three Lines of Code

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

1. `X_scaled = StandardScaler().fit_transform(X)`
2. `pca = PCA(n_components=3)`
3. `X_pca = pca.fit_transform(X_scaled)`

Key Attributes After Fitting:

- `pca.explained_variance_ratio_` – fraction of variance per PC
- `pca.components_` – PC directions (weights for each feature)
- `pca.n_components_` – number of components kept

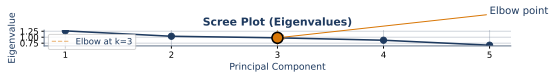
Critical: Always scale first! Unscaled features with large values dominate.

StandardScaler → PCA → fit_transform. Always scale first!

Scree Plot: How Many Components?

Plot Eigenvalues in Decreasing Order

- Look for the "elbow" where values level off – cut there
- Named after geological scree (rubble at a mountain base)

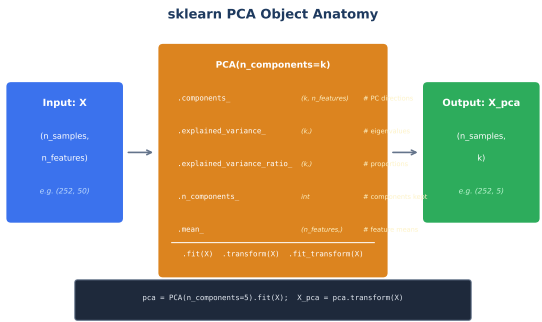


Mountain peak = important PCs. Rubble (scree) = noise. Cut at the elbow.

Self-Study: sklearn PCA Implementation Details

Under the Hood

- sklearn uses SVD (not eigendecomposition) for numerical stability
- `PCA(n_components=0.95)` keeps enough PCs for 95% variance

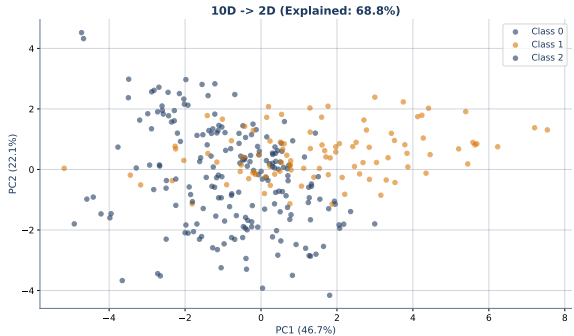


Self-study: sklearn PCA uses SVD internally – same result, better numerics

Self-Study: Choosing n_components

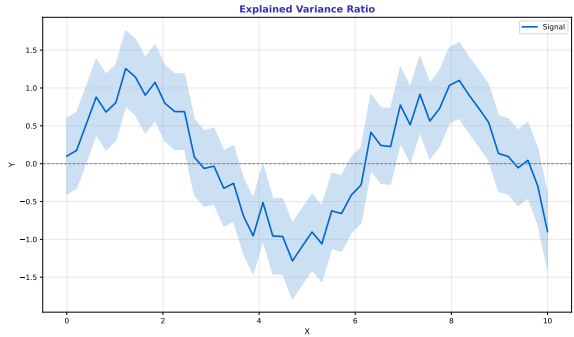
Three Ways to Set n_components

- **Integer:** `PCA(n_components=5)` – exactly 5 PCs
- **Float:** `PCA(n_components=0.95)` – enough for 95% variance



Self-study: Recommended – fit with all, then choose k from scree/variance

Self-Study: Explained Variance Ratio (Bar Chart)

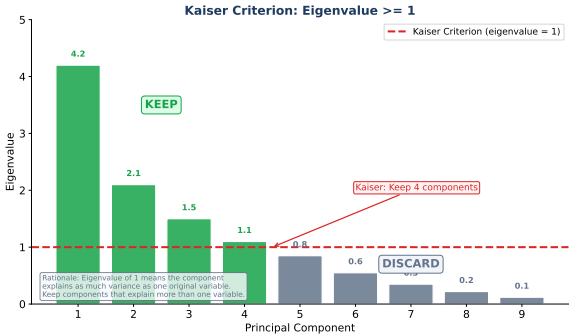


Self-study: Each bar shows the percentage of total variance explained by that component

Self-Study: Kaiser Criterion

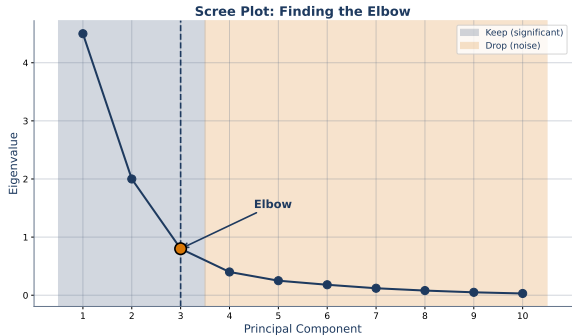
Kaiser Rule: Keep Eigenvalues ≥ 1

- Each standardized feature has variance $= 1$; drop PCs below that
- Quick heuristic – not always reliable, use with scree plot



Self-study: Kaiser = keep eigenvalues ≥ 1. Use as rough guide, not gospel.

Self-Study: Scree Plot Variant



Self-study: Components after the elbow contribute little – safe to drop

Checkpoint: Variance Math

Quick Exercises – Think Before You Peek

Q1: PC1 explains 60%, PC2 explains 25%.
Together they explain **??%** of variance.

Q2: Individual ratios: 50%, 25%, 15%, 7%, 3%.
How many components for 95% variance?

Answers:

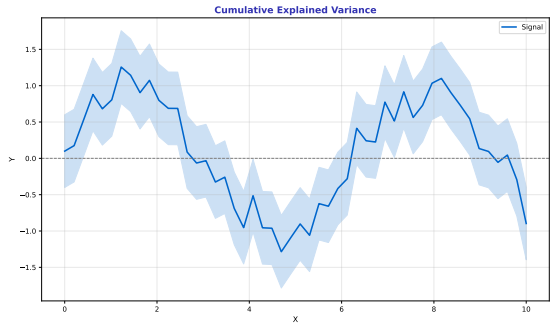
- Q1: $60 + 25 = \mathbf{85\%}$ (cumulative = sum of individual ratios)
- Q2: $50{+}25{+}15{+}7 = 97\% \geq 95\% \Rightarrow$ **4 components**

Cumulative variance = running sum of explained variance ratios. Simple addition!

Cumulative Explained Variance

Running Total of Information Retained

- Cumulative curve: total variance captured by first k PCs
- Read off: "How many components for 95% variance?"

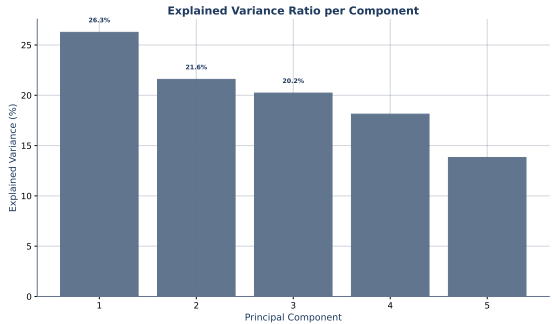


Steep curve = good compression. Flat curve = data is complex, many PCs needed.

Explained Variance Ratio

Individual + Cumulative View

- Bars: each component's share. Line: cumulative sum.
- Decision rule: keep PCs until cumulative ≥ 0.95

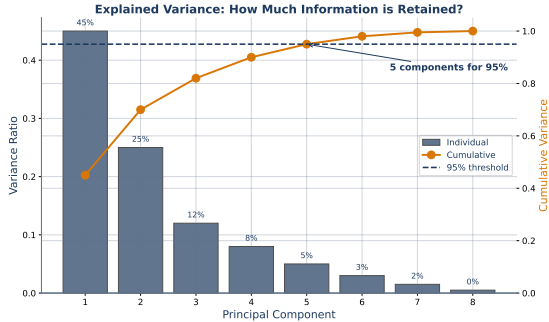


95% variance threshold is convention. Adjust: visualization (80%), reconstruction (99%).

Self-Study: Explained Variance Concept

What "Explained Variance" Actually Means

- Total variance = sum of all eigenvalues = total "spread" in data
- Each PC's ratio = its eigenvalue / total = fraction of spread it captures
- High ratio on first few PCs = data lives in a low-dimensional subspace

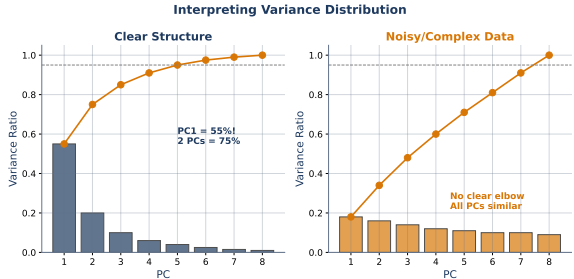


Self-study: Explained variance ratio = eigenvalue / sum of eigenvalues

Self-Study: Practical Interpretation

Reading Variance Plots in Practice

- Steep drop: first few PCs dominate – strong structure, easy to compress
- Gradual decline: many PCs needed – data is high-dimensional / noisy

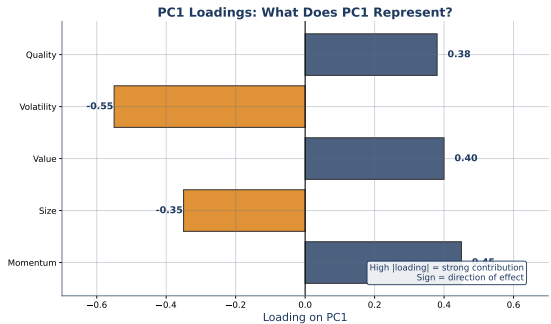


Self-study: Steep = structured data. Flat = noisy / genuinely high-dimensional.

Loadings: What Do Components Mean?

Interpreting the "Black Box"

- Loadings = correlation between original features and each PC
- High loading = strong influence; sign shows direction

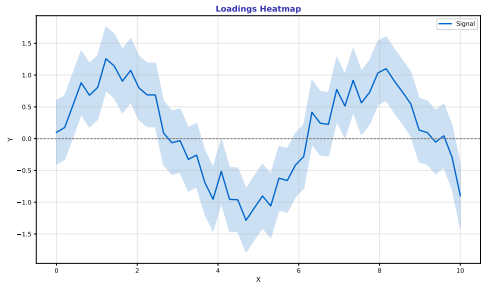


Loadings turn abstract PCs into interpretable factors. "PC1 = market exposure."

Loadings Heatmap

Which Features Drive Which Components?

- Rows = features, columns = PCs. Dark = strong influence.
- Read column-wise to see what each PC captures.

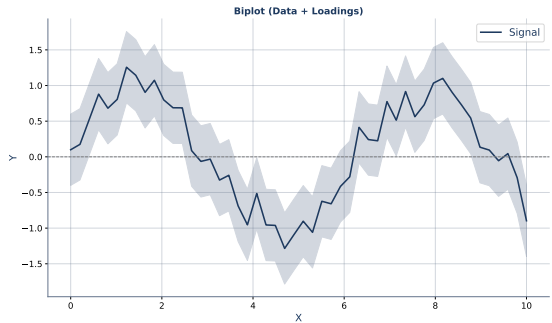


Heatmap pattern reveals what each PC "means" in terms of original features.

Biplot: Data and Loadings Together

The Most Informative PCA Visualization

- Points: observations on PC1 vs PC2. Arrows: feature directions.
- Nearby points are similar; arrows show why.

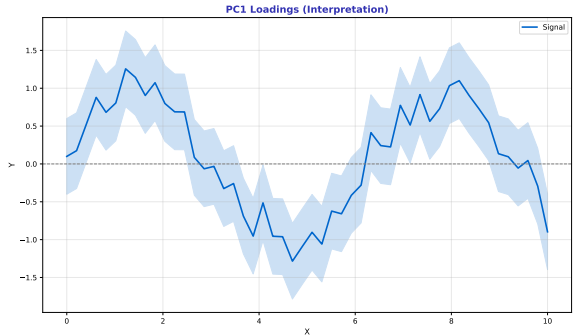


Biplot = scores (points) + loadings (arrows). Arrow direction = feature's PC alignment.

Self-Study: PC1 Loadings Interpretation

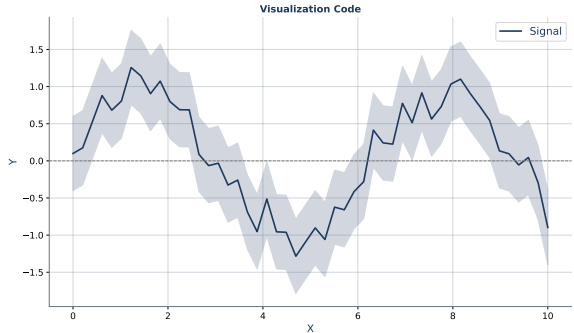
What Does PC1 Represent?

- All stocks load positively on PC1 $\Rightarrow$ PC1 = "market factor"
- Mixed signs on PC2 $\Rightarrow$ PC2 = "sector rotation" or "growth vs value"



Self-study: Finance – PC1 is almost always the market factor

Self-Study: PCA Visualization Code

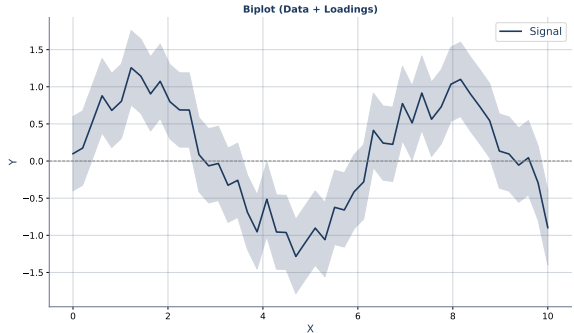


Self-study: Use matplotlib scatter for scores, quiver for loading arrows

Self-Study: Biplot Interpretation

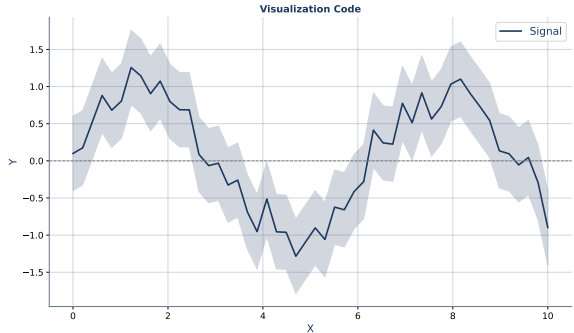
Reading the Biplot

- Near each other = similar. Near arrow tip = high on that feature.



Self-study: Biplot reveals both structure in data and feature relationships

Self-Study: Complete Visualization Pipeline

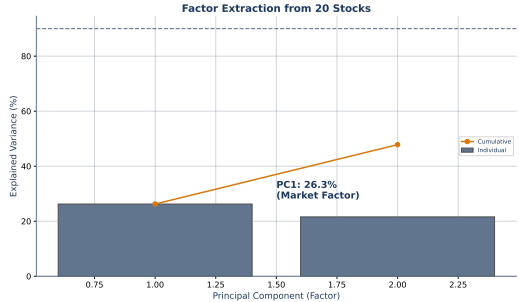


Self-study: Color points by cluster or category to reveal hidden structure

Finance: Factor Extraction from Returns

PCA on Stock Returns = Statistical Factors

- PC1 $\approx$ market factor (all stocks move together)
- PC2–PC3 often capture sector or style effects
- Compare: PCA is data-driven, Fama-French (L24) is theory-driven

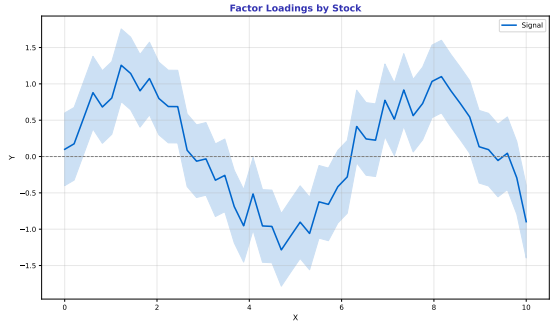


PCA discovers latent factors from returns. Compare with Fama-French from L24.

Finance: Factor Loadings by Stock

Each Stock's Exposure to Statistical Factors

- Loadings show each stock's factor exposure
- Tech loads on PC2 (growth); utilities load oppositely (defensive)

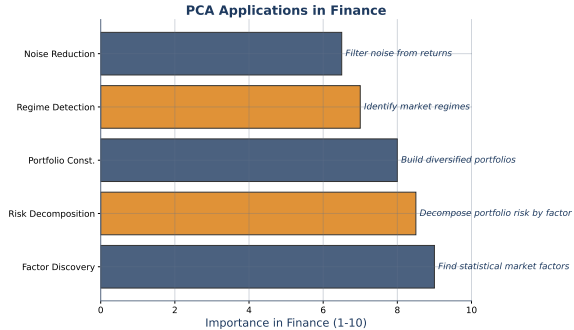


Loadings = factor exposures. Portfolio managers use these for risk budgeting.

Self-Study: Finance Applications of PCA

Beyond Factor Extraction

- Risk decomposition, regime detection, noise filtering via top PCs

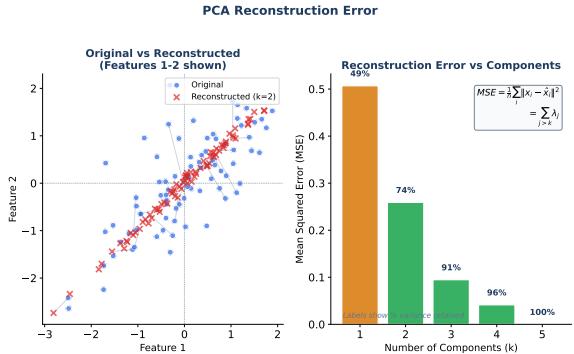


Self-study: PCA is central to quantitative finance risk models

Self-Study: Reconstruction Error

What PCA Loses

- `pca.inverse_transform(X_pca)` reconstructs from k PCs
- Lower k = more compression but higher reconstruction error

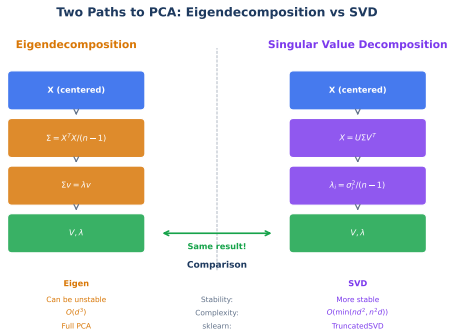


Self-study: Reconstruction error decreases as you keep more components

Self-Study: SVD vs Eigendecomposition

Two Paths to the Same Answer

- Eigendecomposition: $\Sigma v = \lambda v$ (textbook). SVD: $X = U\Sigma V^T$ (sklearn).
- Both give same PCs; SVD is numerically more stable.

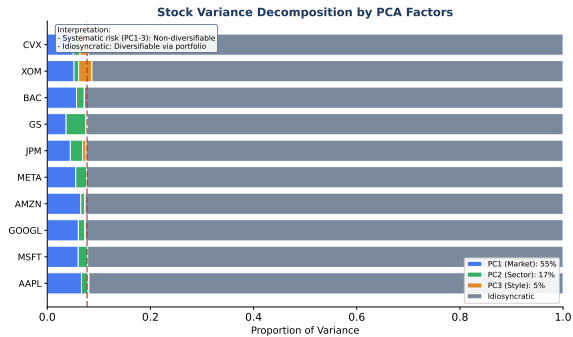


Self-study: Both give the same PCs. SVD is preferred in practice.

Self-Study: Portfolio Variance Decomposition

Attributing Risk to Principal Factors

- PC1 (market) often explains 60–80% of portfolio risk
- Remaining PCs capture sector, style, and idiosyncratic risk

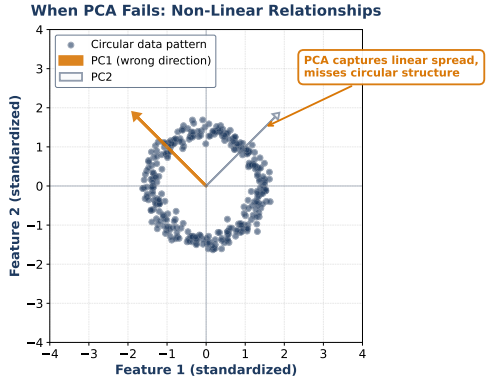


Self-study: Risk managers decompose portfolio variance into factor contributions

Self-Study: PCA Limitations in Detail

When PCA Falls Short

- Linear only – misses curved structure. Sensitive to outliers.
- Alternatives: t-SNE, UMAP, Kernel PCA for nonlinear patterns



Self-study: PCA is linear. For nonlinear structure, consider t-SNE or UMAP.

PCA Limitations

When PCA Struggles

1. Non-Linear Relationships

- PCA finds linear directions only
- Curved or clustered structure may be missed entirely
- Alternative: Kernel PCA, t-SNE, UMAP for nonlinear patterns

2. Scaling Sensitivity

- Unscaled data $\Rightarrow$ PC1 dominated by largest-variance feature
- Always standardize unless features share the same units

3. Interpretability

- PCs are mathematical constructs, not natural concepts
- "PC1 $= 0.35\times$Return $+ 0.28\times$Vol $+ \ldots$" – what does it *mean*?
- Loadings and domain knowledge help, but naming is subjective

---

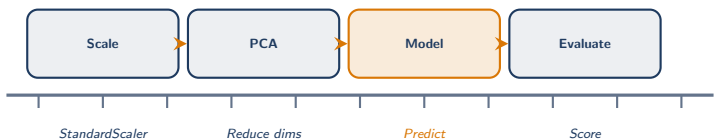
PCA is powerful but linear. Check assumptions before trusting the compression.

What's Next: The ML Assembly Line

Closing: From Components to Complete Pipelines

We now have all the building blocks:

- Preprocessing (scaling, encoding)
- Dimensionality reduction (PCA)
- Models (regression, classification, clustering)



Next lesson (L32): Putting it all together in sklearn `Pipeline` – reproducible, leak-free, deployable.

L32 preview: sklearn Pipeline chains Scale → PCA → Model into one object.

Hands-On Exercise (25 min)

Task: Discover Factors in Stock Returns

1. Load daily returns for 20–30 stocks (1 year of data)
2. Standardize returns with `StandardScaler`
3. Fit `PCA()` and plot a scree plot
4. How many components for 90% variance?
5. Examine PC1 loadings – what does it represent?
6. Create a biplot: project stocks to PC1 vs PC2, color by sector

Deliverable: Scree plot + biplot with sector colors.

Extension: Correlate PC1 scores with S&P 500 returns. Is PC1 the "market factor"?

Tip: Use `yfinance` to download stock prices, then `pct_change()` for returns.

Lesson Summary

Problem Solved: We can now compress many correlated features into a few uncorrelated components while preserving most of the information.

Key Takeaways:

- PCA finds orthogonal axes of maximum variance (new coordinate system)
- Scree plot + cumulative variance guide how many components to keep
- **Always standardize** before PCA – scaling matters!
- Loadings and biplots reveal what each component means
- Finance: PCA extracts statistical factors from return matrices

Next Lesson: L32 – ML Pipeline (putting it all together)

Memory: PCA = rotate to max-variance axes. Always scale. Loadings = interpretation.