Dimensionality Reduction & Clustering

Principal Component Analysis, Factor Models, and K-Means

MSc Banking and Finance

A.U.E.B

October 21, 2025

Today's Agenda

The Curse of Dimensionality

Motivation: The High-Dimensional Problem

Consider a DeFi portfolio manager tracking 50 tokens with 8 metrics each:

- Daily returns
- Volatility (standard deviation)
- Trading volume
- Market capitalization
- Total Value Locked (TVL)
- Liquidity score
- Correlation with ETH
- Sentiment score

Challenge: 50 tokens $\times$ 8 features $=$ 400 numbers to analyze

Solution: Dimensionality reduction techniques compress information while retaining essential patterns

Why Dimensionality Reduction Matters

Problems with High Dimensions:

- Computational complexity
- Overfitting in models
- Difficult visualization
- Noisy features dilute signal
- Multicollinearity issues

Benefits of Reduction:

- Faster computation
- Better generalization
- Clearer insights
- Remove redundancy
- Improved model performance

Key Insight

Most financial data contains redundancy. Multiple variables often measure similar underlying phenomena (e.g., various measures of size, risk, or liquidity).

Principal Component Analysis: Overview

Definition: PCA is an orthogonal linear transformation that converts correlated variables into a set of linearly uncorrelated variables called principal components.

Objective: Find directions of maximum variance in high-dimensional data

Key Properties:

- PC1 captures most variance
- Each PC is orthogonal
- PCs ordered by variance
- Unsupervised method
- Linear transformation

Applications:

- Data compression
- Noise reduction
- Visualization (2D/3D)
- Feature extraction
- Preprocessing for ML

PCA Intuition: A Simple Analogy

Example: Customer Measurements

Suppose we measure 10 physical characteristics of coffee shop customers:

- Height, weight, shoe size, arm length, leg length, hand size, finger length, head circumference, shoulder width, foot length

Observation: These variables are highly correlated

PCA discovers: One dominant factor explains most variation: **body size**

Possibly a second factor: **body proportions** (tall-thin vs short-wide)

Insight

PCA automatically identifies that 10 variables essentially capture 1-2 underlying dimensions

Mathematical Formulation

Setup: Data matrix $\mathbf{X}$ with n observations and p features

Step 1: Standardization

$$\mathbf{Z} = \frac{\mathbf{X} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}$$

Step 2: Covariance Matrix

$$\boldsymbol{\Sigma} = \frac{1}{n-1}\mathbf{Z}^T\mathbf{Z}$$

Step 3: Eigendecomposition

$$\boldsymbol{\Sigma} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T$$

where $\mathbf{V}$ contains eigenvectors (loadings) and $\boldsymbol{\Lambda}$ contains eigenvalues (variances)

Step 4: Principal Components

$$\mathbf{PC} = \mathbf{Z}\mathbf{V}$$

Variance Explained

Key Concept: Each eigenvalue λ_i represents the variance of PC_i

Proportion of variance explained:

$$\text{Var Explained}_i = \frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j}$$

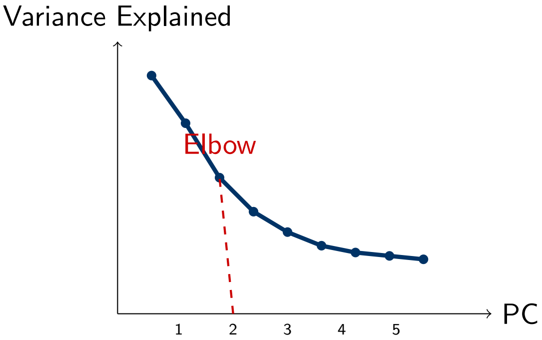
Example with 50 DeFi tokens:

Component	Eigenvalue	Var %	Cumulative %
PC1	3.6	45%	45%
PC2	2.0	25%	70%
PC3	1.2	15%	85%
PC4	0.8	10%	95%
Rest	0.4	5%	100%

Interpretation: First 3 components capture 85% of total variance

The Scree Plot

Purpose: Visual tool for selecting the number of components



Decision Rule:

- Look for the "elbow"
- Sharp drops $\rightarrow$ informative
- Flat region $\rightarrow$ noise
- Aim for 80-90% cumulative variance

In this example:

- Elbow at PC 3-4
- Retain 3 components
- Reduces from 8 to 3 dimensions

Interpreting Principal Components

Loadings: Coefficients linking original variables to PCs

Example: DeFi DEX Analysis

Variable	PC1	PC2	PC3
Trading Volume	0.42	-0.15	0.08
TVL	0.45	-0.12	0.05
ETH Correlation	0.38	-0.10	0.15
Daily Return	0.10	0.52	-0.35
Volatility	0.08	0.48	-0.40
Liquidity	0.35	0.05	0.60
Market Cap	0.44	-0.08	0.12
Sentiment	0.25	0.35	0.45

Interpretation:

- PC1: Market size factor (volume, TVL, market cap)
- PC2: Risk-return factor (return, volatility)
- PC3: Liquidity-sentiment factor

PCA in Practice: Implementation

Python Implementation:

Code Example

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import pandas as pd

# Load and standardize data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Fit PCA
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)

# Variance explained
print(pca.explained_variance_ratio_)
```

Important: Always standardize before PCA to ensure features are on comparable scales

Factor Analysis: Motivation

Question: Why are financial assets correlated?

Observation:

- DEX tokens tend to move together
- Lending protocols correlate with each other
- Most tokens correlate with Bitcoin/Ethereum

Hypothesis: Hidden common factors drive these correlations

Factor Analysis Goal

Identify latent (unobserved) factors that explain correlations among observed variables. Unlike PCA, Factor Analysis explicitly models shared variance vs unique variance.

Factor Analysis vs PCA

Principal Component Analysis:

- Finds directions of maximum variance
- Explains total variance (signal + noise)
- Mathematical/statistical tool
- No underlying causal model
- Data compression

Use cases:

- Visualization
- Dimensionality reduction
- Preprocessing

Factor Analysis:

- Finds hidden causal factors
- Explains shared variance only
- Theoretical model
- Assumes latent structure
- Understanding relationships

Use cases:

- Risk modeling
- Factor investing
- Structural analysis

Key Difference

PCA: "What patterns exist?" Factor Analysis: "What causes these patterns?"

Factor Model Specification

Mathematical Model:

$$X_i = \sum_{k=1}^{m} \beta_{ik} F_k + \epsilon_i$$

where:

- X_i = observed variable (e.g., token return)
- F_k = latent factor k (unobserved)
- β_{ik} = factor loading (sensitivity to factor k)
- ϵ_i = unique component (idiosyncratic)

Variance Decomposition:

$$\mathrm{Var}(X_i) = \underbrace{\sum_{k=1}^{m} \beta_{ik}^2}_{\text{Communality } h_i^2} + \underbrace{\mathrm{Var}(\epsilon_i)}_{\text{Uniqueness } u_i}$$

Constraint: $h_i^2 + u_i = 1$ (total variance = 100%)

Example: Multi-Factor Model for UNI Token

Empirical Application:

$$r_{\text{UNI}} = \underbrace{0.6 \times F_{\text{Market}}}_{\text{45\% var}} + \underbrace{0.4 \times F_{\text{DEX}}}_{\text{30\% var}} + \underbrace{0.2 \times F_{\text{Liquidity}}}_{\text{10\% var}} + \underbrace{\epsilon}_{\text{15\% var}}$$

Interpretation:

- 45% driven by overall market
- 30% by DEX sector
- 10% by liquidity conditions
- 15% UNI-specific

Risk Management:

- Communality: 85%
- Uniqueness: 15%
- Mostly systematic risk
- Limited diversification benefit

Portfolio Implication

High communality means UNI offers little diversification. Need assets with different factor exposures.

Factor Analysis Results: 20 DeFi Tokens

Identified Factors:

Token	F1: Market	F2: DEX	F3: Lending
UNI	0.65	**0.82**	0.15
SUSHI	0.60	**0.78**	0.20
CAKE	0.55	**0.70**	0.10
AAVE	0.70	0.12	**0.85**
COMP	0.68	0.18	**0.80**
MKR	0.62	0.25	**0.75**
LINK	0.75	0.30	0.40

Factor Interpretation:

- Factor 1 (40% variance): Overall market sentiment
- Factor 2 (25% variance): DEX-specific risk
- Factor 3 (20% variance): Lending protocol risk

Portfolio Construction with Factor Analysis

Strategy: Diversify across factors, not just tokens

Poor Diversification:

- 50% UNI
- 30% SUSHI
- 20% CAKE

Problem: All heavily loaded on DEX factor

Factor exposures:

- Market: 0.60
- DEX: 0.77
- Lending: 0.15

Better Diversification:

- 40% UNI (DEX)
- 40% AAVE (Lending)
- 20% LINK (Infrastructure)

Benefit: Balanced factor exposure

Factor exposures:

- Market: 0.70
- DEX: 0.41
- Lending: 0.47

Key Insight

True diversification requires understanding factor structure, not just holding many assets

K-Means Clustering: Overview

Objective: Partition observations into K clusters such that observations within each cluster are similar

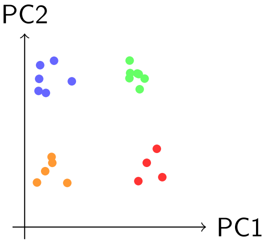
Application: Group 50 DeFi tokens into 4 risk categories

Method:

- Unsupervised learning
- Distance-based clustering
- Iterative algorithm
- Requires pre-specified K

Use Cases:

- Market segmentation
- Risk categorization
- Pattern discovery
- Data exploration



K-Means Algorithm

Iterative Procedure:

1. **Initialize:** Randomly select K cluster centers
2. **Assignment:** Assign each observation to nearest center
3. **Update:** Compute new centers as mean of assigned observations
4. **Iterate:** Repeat steps 2-3 until convergence

Objective Function:

$$\min \sum_{k=1}^{K} \sum_{i \in C_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2$$

where:

- C_k = set of observations in cluster k
- $\boldsymbol{\mu}_k$ = centroid of cluster k
- $\|\cdot\|$ = Euclidean distance

Properties: Algorithm converges to local minimum, may require multiple runs

Distance Calculation Example

Setup: Three tokens with two features (Return, Volatility)

Token Data:

- UNI: (0.15%, 5%)
- AAVE: (0.18%, 6%)
- DOGE: (0.50%, 15%)

Proposed Clustering:

- Cluster 1: {UNI, AAVE}
- Cluster 2: {DOGE}

Distance Calculations:

Cluster 1 center: (0.165%, 5.5%)

UNI to Cluster 1:

$$d = \sqrt{(0.15 - 0.165)^2 + (5 - 5.5)^2}$$
$$= 0.50$$

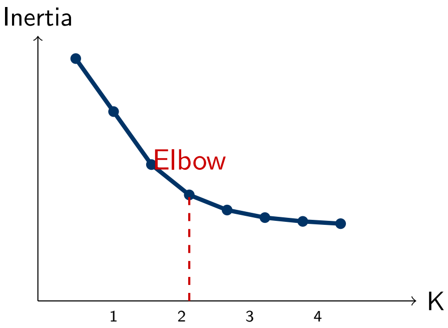
DOGE to Cluster 1:

$$d = \sqrt{(0.50 - 0.165)^2 + (15 - 5.5)^2}$$
$$= 9.51$$

Conclusion: DOGE is clearly different from the UNI/AAVE cluster

Choosing the Number of Clusters K

The Elbow Method:



In this example: $K = 4$ appears optimal

Decision Criteria:

- Steep decline: informative
- Flat region: diminishing returns
- Choose K at the "elbow"

Alternative: Silhouette Score

- Measures cluster quality
- Range: [-1, +1]
- Higher is better
- Evaluates separation

K-Means Results: DeFi Portfolio

Clustering 50 tokens into 4 groups:

Cluster	Size	Avg Return	Avg Vol	Sharpe	Label
0	15	0.05%	2.1%	1.2	Blue Chips
1	18	0.15%	3.5%	1.1	Growth
2	12	0.25%	6.2%	0.8	Speculative
3	5	0.30%	8.5%	0.4	High Risk

Cluster Characteristics:

- Cluster 0: Large-cap stablecoins
- Cluster 1: Established protocols
- Cluster 2: Emerging projects
- Cluster 3: Volatile/micro-cap

Portfolio Allocation:

- 50% Cluster 0 (stability)
- 30% Cluster 1 (growth)
- 15% Cluster 2 (upside)
- 5% Cluster 3 (high risk/reward)

Strategic Insight

Clustering provides objective risk categorization for systematic portfolio construction

Practical Considerations

Best Practices for K-Means:

1. **Standardization:** Always standardize features to ensure equal weighting
2. **Multiple Runs:** Run algorithm 10-20 times with different initializations
3. **Validation:** Use silhouette score to validate cluster quality
4. **Interpretation:** Examine cluster centroids for business meaning
5. **Stability:** Test robustness across different time periods

Common Pitfalls:

- Forgetting to standardize (features with larger scales dominate)
- Single random initialization (may find poor local optimum)
- Choosing K arbitrarily (use elbow method or silhouette analysis)
- Over-interpreting clusters (they may not have economic meaning)

PCA + K-Means: A Powerful Combination

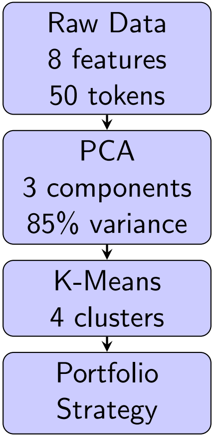
Motivation: Why combine PCA with K-Means?

Benefits:

1. Noise reduction
2. Computational efficiency
3. Better cluster stability
4. Easier visualization
5. Removes multicollinearity

Procedure:

1. Apply PCA
2. Retain top k components
3. Apply K-Means on PCs
4. Analyze clusters



PCA + K-Means: Empirical Results

Comparative Performance Analysis:

Method	Silhouette Score	Time (sec)	Stability
K-Means only	0.42	5.2	Low
PCA + K-Means	**0.58**	**1.8**	High

Advantages of PCA preprocessing:

- Higher silhouette score (better cluster quality)
- Faster computation (fewer dimensions)
- More stable across random initializations
- Removes redundant information

Industry Practice

PCA + K-Means is standard in portfolio management for handling high-dimensional asset data

Case Study: Building a $1M DeFi Portfolio

Analysis Pipeline:

Step 1: Data Collection

- 50 tokens
- 12-month history
- 8 features per token

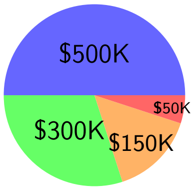
Step 2: PCA

- Extract 3 components
- 87% variance explained
- Interpret factors

Step 3: K-Means

- Optimal K = 4
- Identify risk clusters
- Label categories

Final Allocation:



Blue:	Stable
Green:	Growth
Orange:	Speculative
Red:	High Risk

Expected Performance:

- Return: 12-18% APY
- Sharpe: 1.2
- Max DD: -25%

Python Implementation: Complete Pipeline

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

# Step 1: Standardize
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 2: PCA
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)
print(f"Variance: {pca.explained_variance_ratio_}")

# Step 3: K-Means
kmeans = KMeans(n_clusters=4, n_init=10, random_state=42)
clusters = kmeans.fit_predict(X_pca)

# Step 4: Analysis
df['cluster'] = clusters
print(df.groupby('cluster').mean())
```

Common Mistakes to Avoid

Mistake 1: Not Standardizing

Problem:

- Trading volume (millions)
- Returns (percentages)
- Large values dominate

Solution: $z = \frac{x-\mu}{\sigma}$

Mistake 2: Too Many PCs
Solution:

- Use scree plot
- Aim for 80-90% variance
- Balance information vs complexity

Mistake 3: Random K Selection
Problem:

- Too few: miss patterns
- Too many: overfit noise

Solution:

- Elbow method
- Silhouette analysis
- Domain knowledge

Mistake 4: Single K-Means Run
Solution:

- Run 10-20 times
- Use `n_init` parameter
- Select best result

Validation and Robustness Checks

Ensuring Reliable Results:

1. **Temporal Stability:** Test on different time periods
    - Do clusters remain stable?
    - Are factor loadings consistent?
2. **Cross-Validation:** Out-of-sample testing
    - Train on 80% of data
    - Validate on remaining 20%
3. **Sensitivity Analysis:** Vary parameters
    - Different numbers of PCs
    - Different K values
    - Different standardization methods
4. **Economic Interpretation:** Do results make sense?
    - Are clusters interpretable?
    - Do factors align with theory?

Real-World Applications

1. **Portfolio Management**
    - Risk-based asset allocation
    - Systematic rebalancing rules
    - Factor exposure monitoring
2. **Risk Management**
    - Multi-factor risk models
    - Stress testing scenarios
    - Correlation structure analysis
3. **Trading Strategies**
    - Statistical arbitrage across clusters
    - Factor momentum strategies
    - Mean-reversion within clusters
4. **Research and Analytics**
    - Market structure analysis
    - Protocol categorization
    - Competitive landscape mapping

Limitations and Considerations

Technical Limitations:

- **Linearity Assumption:** PCA and Factor Analysis assume linear relationships
- **Euclidean Distance:** K-Means assumes spherical clusters
- **Stationarity:** Methods assume stable relationships over time
- **Sample Size:** Require sufficient observations relative to features

Market Considerations:

- DeFi markets are highly dynamic
- Correlations can break down during stress
- New protocols constantly emerge
- Regulatory changes affect factor structure

Recommendation

Regularly update models and validate results. In DeFi, monthly retraining is advisable.

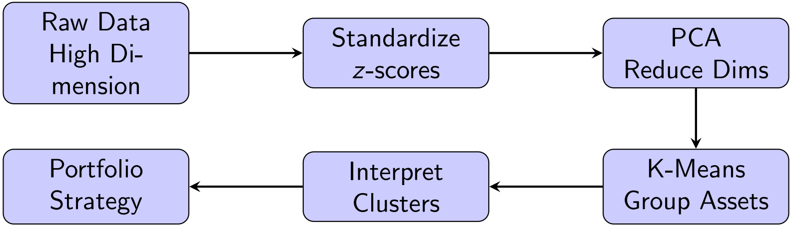
Key Takeaways

1. **Principal Component Analysis (PCA)**
   - Reduces dimensions while preserving 80-90% of variance
   - Use scree plot to select components
   - Interpret via factor loadings
   - Essential preprocessing step

2. **Factor Analysis**
   - Identifies latent causal factors
   - Separates common vs unique variance
   - Critical for multi-factor risk models
   - Enables factor-based diversification

3. **K-Means Clustering**
   - Groups similar observations
   - Use elbow method to choose K
   - Always standardize first
   - Run multiple initializations

Integration: The Complete Workflow



Best Practice

This pipeline is standard in quantitative finance for processing high-dimensional asset data and constructing systematic portfolios.