SEN608 · WEEK 10

Unsupervised Learning

Principal Components · K-Means · Hierarchical · Gaussian Mixture Models

Textbook: Bruce, Bruce & Gedeck (2026) "AI-Assisted Statistics for Data Scientists", 3rd ed. — Chapter 7

Assoc. Prof. Dr. Onur Polat

Hacettepe University · Graduate School of Informatics · Systems Engineering

Fall 2026-2027 · 2-hour graduate lecture · 32 slides

Today's Agenda

6 blocks · ~2 hours · Python + scikit-learn hands-on

01 **What is Unsupervised Learning?**

Supervised vs unsupervised, applications

02 **Principal Components Analysis**

Variance, eigendecomposition, scree plot

03 **K-Means Clustering**

WCSS, Lloyd's algorithm, choosing K

04 **Hierarchical Clustering**

Agglomerative, linkage, dendrogram

05 **Model-Based Clustering (GMM)**

Mixture models, EM, BIC

06 **Scaling & Mixed Data**

Standardization, Gower, categorical encoding

BLOCK 1

What is Unsupervised Learning?

Finding structure in data without a target variable

Supervised vs Unsupervised Learning

A fundamental split in machine learning

🎯 Supervised Learning

`Data: (X, y)` with a known target y

- Train a model to predict y from X
- Clear objective (MSE, accuracy ...)
- Easy to validate with train/test
- Examples: regression, classification, time series forecasting

🔍 Unsupervised Learning

`Data: only X` — no target

- Goal: **discover structure** (clusters, components, outliers)
- No single "right" answer
- Validation is harder — depends on domain
- Examples: PCA, clustering, anomaly detection

💡 Two main unsupervised tasks

1. **Dimensionality reduction** — summarize many features in a few. Tools: PCA, t-SNE, UMAP, autoencoders.
2. **Clustering** — group observations into similar subsets. Tools: K-Means, hierarchical, GMM, DBSCAN.

Applications in Business and Research

Why unsupervised learning matters

Field	Task	Technique
Marketing	Customer segmentation — group buyers by behavior	K-Means, GMM
Finance	Portfolio construction — identify return-pattern clusters	Hierarchical, PCA
Genomics	Group patients by gene expression profile	Hierarchical, GMM
Image analysis	Compress / denoise images	PCA, autoencoders
Anomaly detection	Fraud, intrusion, equipment failures	GMM, Isolation Forest
Text / NLP	Topic modeling across documents	LDA, clustering of embeddings
Factor investing	Extract principal risk drivers from asset returns	PCA

BLOCK 2

Principal Components Analysis

Summarize many correlated variables with a few uncorrelated ones

Why PCA?

The curse of correlated, high-dimensional data

The problem

With 50 correlated features, we face: slow computation, noisy covariance estimates, multicollinearity in downstream regressions, and impossible visualization. Many of those features likely carry redundant information — their variance can be summarized by a much smaller set of orthogonal directions.

🎯 The PCA idea (Pearson 1901, Hotelling 1933)

- Find a set of **orthogonal linear combinations** of the original variables that capture the maximum variance
- PC1 captures the direction of largest variance; PC2 the largest variance orthogonal to PC1; and so on
- Retain only the first few PCs → **massive dimensionality reduction** with minimal information loss
- The PCs are **uncorrelated by construction** — solves multicollinearity

The Math of PCA

Eigendecomposition of the covariance matrix

Step 1 — Center the data and compute the covariance matrix

$$\Sigma = \frac{1}{n-1}\mathbf{X}_c^\top \mathbf{X}_c, \qquad \mathbf{X}_c \text{ centered}$$

Step 2 — Solve the eigenvalue problem

$$\Sigma\,\mathbf{v}_k = \lambda_k\,\mathbf{v}_k, \qquad \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \geq 0$$

Step 3 — Project the data onto each eigenvector

$$\mathrm{PC}_k = \mathbf{X}_c\,\mathbf{v}_k$$

- **v_k** is the **k-th loading vector** (direction in feature space); **λ_k** is the variance along that direction
- Modern software uses SVD on X directly — numerically more stable, same result

How Many Components to Keep?

The variance-explained criterion

Fraction of total variance captured by PC_k

$$\text{Variance explained by } \text{PC}_k = \frac{\lambda_k}{\sum_{j=1}^{p} \lambda_j}$$

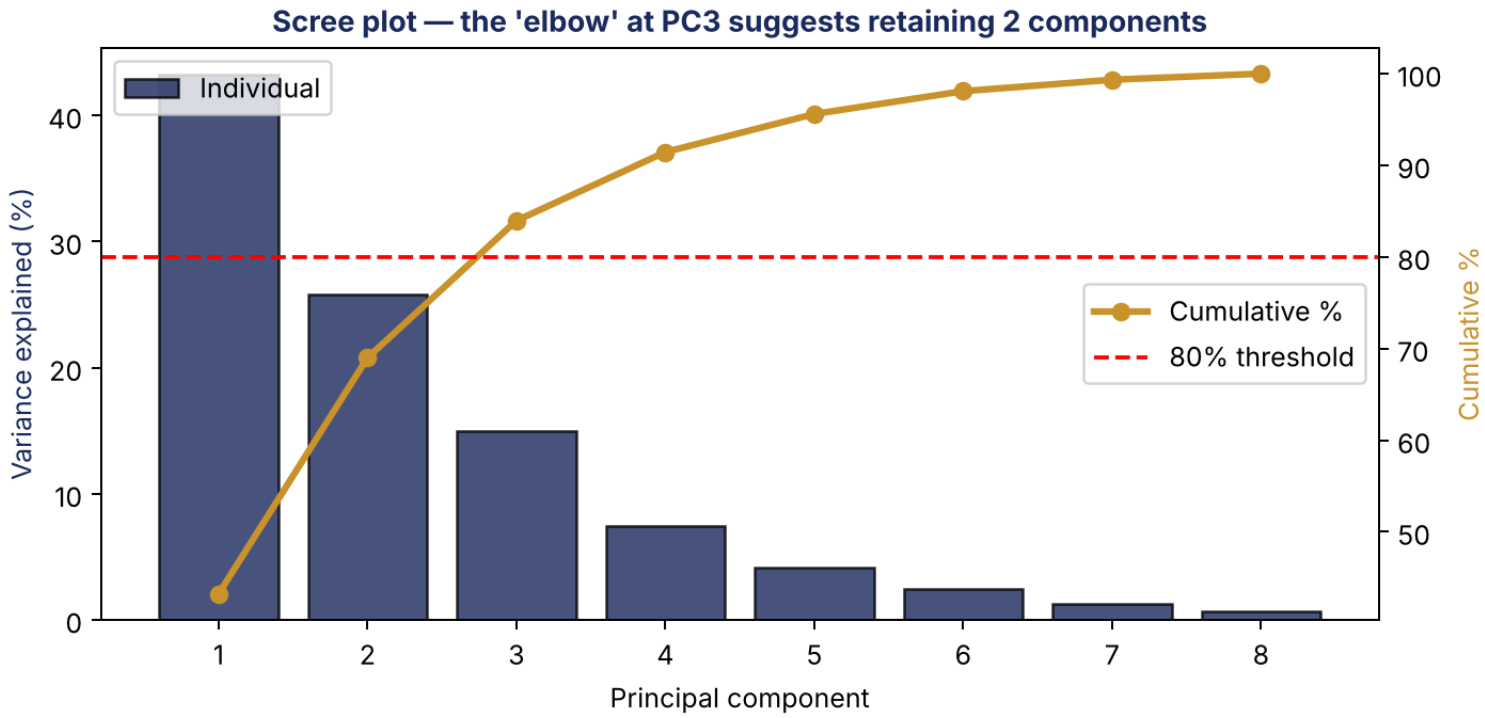
Three common rules of thumb

1. **Cumulative threshold:** keep enough PCs to reach ~80% or ~90% of total variance
2. **Scree-plot elbow:** plot λ_k vs k and retain PCs before the "bend" where marginal variance flattens
3. **Kaiser rule:** if data were standardized, keep PCs with λ_k > 1 (they explain more variance than one original variable)

⚠ **Always standardize the data before PCA when variables are on different scales** — otherwise large-scale variables dominate the components.

Scree Plot — Visualizing Variance Explained

Where does the "elbow" bend?



How to read a scree plot

- Bars = individual variance of each PC; Line = cumulative variance
- Look for where the bars drop sharply then flatten — this is the "elbow"
- In the plot above: PC1 + PC2 explain ~75% of total variance → retain 2 components is a reasonable choice

Interpreting the Principal Components

What does each PC mean?

Loadings = the coefficients of the original variables in each PC

- A loading close to +1 on PC1 for variable X_j → X_j contributes strongly in the positive direction to PC1
- A loading close to −1 → strong negative contribution
- A loading close to 0 → little contribution
- Interpret each PC by looking at which variables have the largest absolute loadings — try to give them a name ("growth factor", "size factor", etc.)

📊 Classical finance example — PCA of yield curves

When PCA is applied to daily changes in Treasury yields across maturities (1M, 3M, 1Y, 2Y, 5Y, 10Y, 30Y), three PCs typically explain over 95% of variance:

- **PC1 (≈ 70-80%): LEVEL** — all maturities move together
- **PC2 (≈ 15-20%): SLOPE** — short-end up, long-end down (or vice versa)
- **PC3 (≈ 3-5%): CURVATURE** — belly up, ends down (or vice versa)

BLOCK 3

K-Means Clustering

Partition data into K groups by minimizing within-cluster variance

The K-Means Objective

Minimize Within-Cluster Sum of Squares (WCSS)

Objective function

$$\text{WCSS}(C_1, \ldots, C_K) = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - \boldsymbol{\mu}_k\|^2$$

Optimal centroid (given cluster assignments)

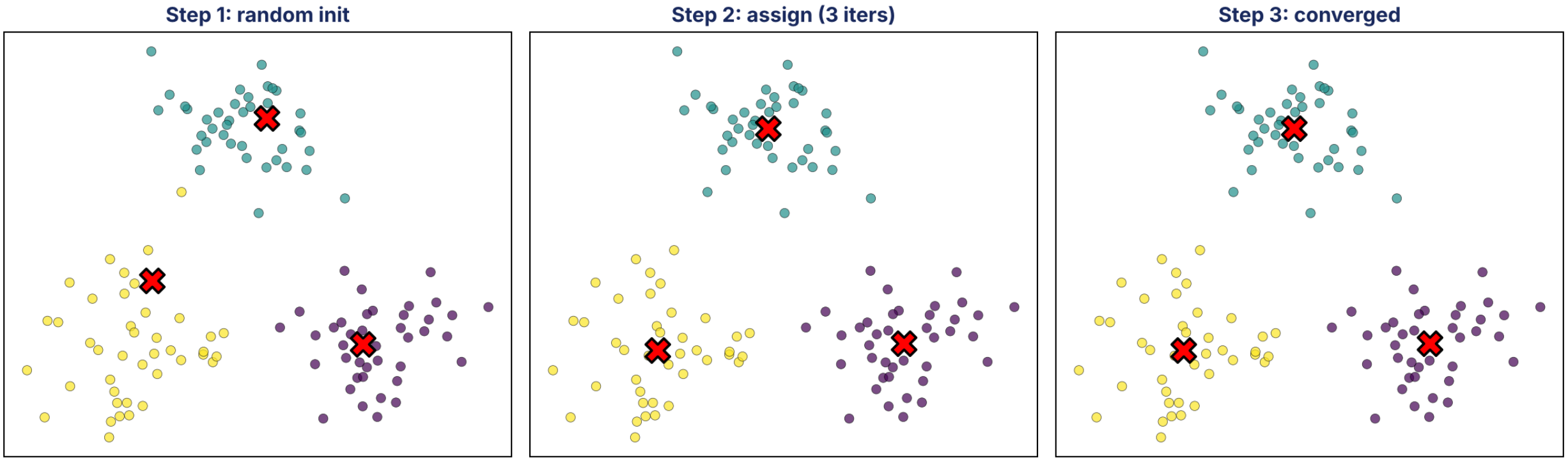
$$\boldsymbol{\mu}_k = \frac{1}{|C_k|} \sum_{x \in C_k} x$$

Key points

- Assumes clusters are roughly **spherical and of similar size** — squared Euclidean distance drives everything
- NP-hard problem in general; Lloyd's algorithm finds a LOCAL optimum
- Scale matters: **always standardize features** before running K-Means

Lloyd's Algorithm

The iterative procedure every K-Means library implements



Algorithm

1. Initialize K centroids (random or k-means++)
2. Assign each point to the nearest centroid (Euclidean distance)
3. Recompute each centroid as the mean of its assigned points
4. Repeat steps 2-3 until assignments stop changing (or max iterations reached)

Choosing K — the Number of Clusters

Three widely used criteria

① Elbow method

Plot WCSS vs K. Pick the K where the curve "bends" — adding one more cluster brings little marginal gain.

✓ Intuitive, graphical
✗ Subjective — elbow sometimes unclear

② Silhouette score

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \qquad s(i) \in [-1, 1]$$

a(i) = mean dist to own-cluster points; b(i) = mean dist to nearest other cluster. Values near +1 = well clustered.

✓ Quantitative
✓ Can compare K values

③ Business judgment

Interpretability trumps statistics. If your marketing team can only act on 4 segments, choose K=4 regardless of WCSS.

✓ Actionable
✓ Domain-aware
⚠ Don't let analysis override reality

K-Means Limitations

When K-Means is the wrong tool

K-Means struggles when:

- **Clusters are non-spherical** (elongated, crescent-shaped) — Euclidean geometry fails. Try DBSCAN, spectral clustering, GMM with full covariance.
- **Clusters have very different sizes or densities** — K-Means prefers balanced partitions.
- **Many outliers are present** — the mean is sensitive. Consider K-Medoids (PAM) or robust alternatives.
- **The true K is unknown and the elbow is ambiguous** — combine elbow + silhouette + business logic.
- **Mixed data types (numeric + categorical)** — Euclidean distance is ill-defined. Use K-Prototypes or Gower distance with hierarchical clustering.

BLOCK 4

Hierarchical Clustering

Build a tree — no pre-specified K required

Agglomerative Hierarchical Clustering

Bottom-up: start with n singletons, merge until one cluster remains

The algorithm

1. Start with every observation in its own cluster (n clusters)
2. Compute a distance matrix between all cluster pairs
3. Merge the two closest clusters into one
4. Recompute distances between the new cluster and all remaining clusters (via a LINKAGE rule)
5. Repeat 3-4 until a single cluster contains all observations

Advantages

- **No need to specify K in advance** — cut the dendrogram at a chosen height to get any K
- **Deterministic** — no random initialization
- **Works with any distance metric** (Gower for mixed data, cosine for text, correlation for returns)
- Produces a rich diagnostic (the dendrogram)
- *⚠ $O(n^2 \log n)$ time and $O(n^2)$ memory — does not scale beyond ~10 000 observations*

Linkage Methods

How to measure distance between two clusters

Single linkage — nearest neighbor (produces "chaining", elongated clusters)

$$d_{\text{single}}(A,\ B) = \min_{x \in A,\ y \in B} d(x,\ y)$$

Complete linkage — farthest neighbor (produces compact, spherical clusters)

$$d_{\text{complete}}(A,\ B) = \max_{x \in A,\ y \in B} d(x,\ y)$$

Average linkage — mean pairwise distance (balanced)

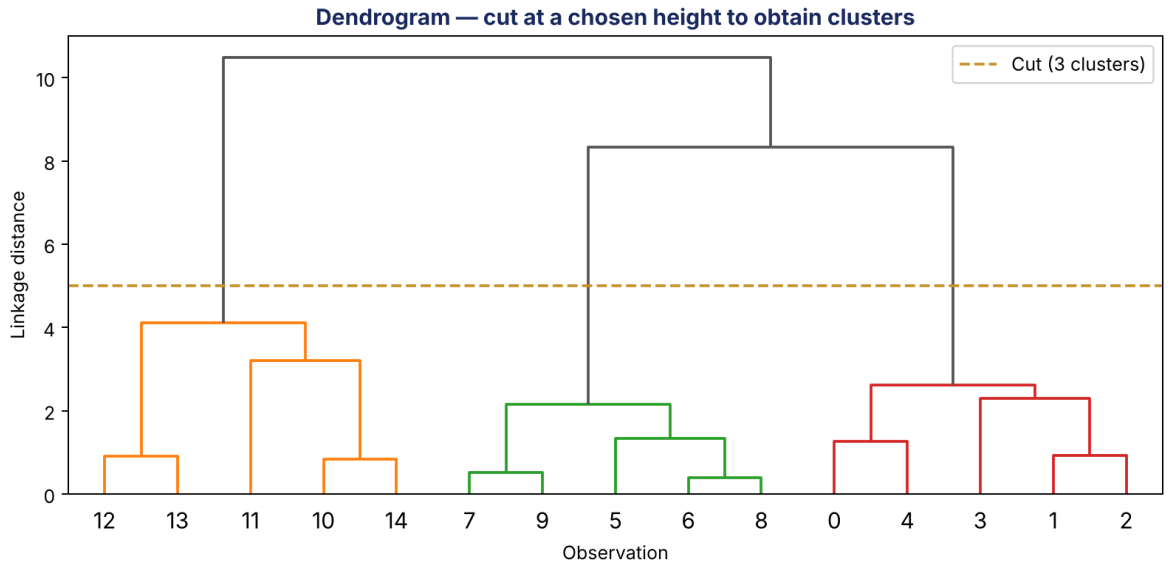
$$d_{\text{average}}(A,\ B) = \frac{1}{|A|\,|B|} \sum_{x \in A} \sum_{y \in B} d(x,\ y)$$

Ward's method — minimize within-cluster variance increase (DEFAULT for most purposes)

$$d_{\text{Ward}}(A,\ B) = \frac{|A|\,|B|}{|A| + |B|} \|\bar{A} - \bar{B}\|^2$$

Reading a Dendrogram

The output tree — cut it to get clusters



How to read the dendrogram

- Horizontal axis = individual observations (leaves)
- Vertical axis = linkage distance at which two clusters were merged (merge height)
- Longer vertical lines = greater separation between the clusters being merged
- **To get K clusters:** draw a horizontal line that crosses exactly K vertical branches (dashed gold line in the picture above)

BLOCK 5

Model-Based Clustering (GMM)

Clusters as mixtures of Gaussian distributions

The Gaussian Mixture Model

A probabilistic approach to clustering

Density of the whole dataset

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \boldsymbol{\mu}_k, \Sigma_k), \qquad \sum_k \pi_k = 1$$

Interpretation

- Each observation is assumed to come from **one of K Gaussian distributions** — we just don't know which one
- **π_k :** the mixing weight = prior probability that a random observation belongs to component k
- **μ_k, Σ_k :** the mean and covariance of component k — **each cluster has its own shape** (elongated, tilted, elliptical)
- Unlike K-Means, GMM gives **soft assignments** — the probability that observation i comes from component k
- Classifies by the maximum posterior responsibility γ_ik

The EM Algorithm

Expectation-Maximization for fitting GMMs

E-step — responsibility (posterior cluster probability)

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(x_i \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i \mid \boldsymbol{\mu}_j, \Sigma_j)}$$

The EM loop (Dempster, Laird & Rubin, 1977)

- **E-step:** Given current parameters, compute responsibility γ_ik — probability that point i came from component k
- **M-step:** Re-estimate parameters as weighted averages using the responsibilities:
  - `π_k = (Σ_i γ_ik) / n`
  - `μ_k = weighted mean with weights γ_ik`
  - `Σ_k = weighted sample covariance`

Iterate E and M until the log-likelihood stops increasing (typically < 100 iterations).

GMM vs K-Means

When to prefer each

Property	K-Means	GMM
Cluster shape	Spherical (equal size)	Any elliptical shape
Assignment	Hard (one label per point)	Soft (probabilities per cluster)
Hyperparameters	K only	K + covariance structure
Choosing K	Elbow / silhouette	BIC / AIC (information criteria)
Scalability	Very fast	Slower (EM iterations)
Outlier handling	Poor	Better (low responsibility values)
Interpretability	Very intuitive	Needs statistical background

Choosing K and Covariance Type for GMM

Use the Bayesian Information Criterion (BIC)

$$\mathrm{BIC} = -2 \ln \hat{\mathcal{L}} + k \ln n$$

How BIC works

- **`ln ℒ̂ :`** maximized log-likelihood of the GMM at convergence
- **`k :`** number of free parameters in the model (depends on K AND covariance type)
- **`n :`** sample size — the penalty term grows with n

LOWER BIC = better model. Fit GMM for K = 1, 2, 3, ..., and multiple covariance types ("spherical", "diag", "tied", "full"); pick the combination with the lowest BIC.

BIC penalizes model complexity more strongly than AIC — preferred when sample is large.

BLOCK 6

Scaling and Mixed Data

Practical preprocessing for every unsupervised method

Why Scaling Matters So Much

Variance-based methods are not scale-invariant

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}, \qquad \text{so that } \bar{z}_j = 0,\ s_{z_j}^2 = 1$$

The scale problem in unsupervised learning

- **PCA:** a feature measured in Turkish lira (range 0-100 000) will DOMINATE a feature measured in percent (range 0-10). PC1 just tracks the big-scale variable.
- **K-Means / GMM:** Euclidean distance is driven by large-scale variables. Clusters form along those axes.

⟹ **Always standardize numeric features** (StandardScaler) before PCA, K-Means, or GMM — unless you have a strong reason not to.

- An alternative is min-max scaling to [0, 1]. For heavy-tailed data (financial returns, firm sizes) consider taking logarithms first.

Handling Categorical and Mixed Data

What to do when your features are not all numeric

Three main strategies

1. One-hot encode categorical variables before PCA / K-Means

⚠ *Creates sparse binary variables with very different variance → scale them too*

⚠ *High-cardinality categoricals (hundreds of values) blow up dimensionality*

2. Use Gower distance + hierarchical clustering (no K-Means compatible)

– Gower = weighted mean of per-feature dissimilarities, chosen by type: Manhattan for numeric, matching for categorical

– Industry standard for market-research and customer segmentation

3. K-Prototypes — extends K-Means

– Uses Euclidean for numeric features AND matching dissimilarity for categorical features in one objective

– Python package: `kmodes.KPrototypes`

Validating Unsupervised Models

Why there is no "true answer" to check against

⚠ The fundamental challenge

Unlike supervised learning, there is no held-out test label to verify against. Different algorithms, distance metrics, and preprocessing choices can produce very different clusterings — all plausible. Validation requires domain context.

Validation strategies

- **Internal metrics:** Silhouette, Davies-Bouldin, Calinski-Harabasz, BIC
- **External validation:** if you have a holdout label (even a noisy one), compute Adjusted Rand Index or NMI
- **Stability analysis:** bootstrap or subsample the data and check whether cluster assignments are consistent
- **Domain validation:** do the discovered clusters make business/scientific sense? Can an expert name and describe them?

Takeaways and Homework

What to remember and what to practice

🧠 Key concepts to remember

1. Unsupervised learning finds **structure without a target** — dimensionality reduction + clustering
2. PCA = **eigendecomposition of covariance** → orthogonal components ranked by variance
3. K-Means minimizes WCSS; fast and simple but requires **spherical, similar-size clusters**
4. Hierarchical clustering produces a **dendrogram; Ward linkage** is a good default
5. GMM allows **elliptical clusters + soft assignments;** BIC picks K and covariance structure
6. **ALWAYS standardize numeric features** before any variance-based unsupervised method

💻 Colab practice

- Apply PCA to a multi-asset return panel (S&P 500 sectors or BIST-100 sectors) — identify the first 3 components
- K-Means on the Iris dataset — compare WCSS and silhouette across K = 2 to 10
- Hierarchical clustering of S&P sector returns — plot the dendrogram with Ward linkage
- GMM with BIC selection on the Wine dataset — compare with K-Means on the same data
- (Challenge) Repeat PCA with and without standardization — see how the first component changes

Thank You

Questions?

onurpolat@hacettepe.edu.tr