ELEC3544 · Fall 2026

Data Science with Foundation Models

Yang Liu

The University of Hong Kong



Unsupervised Learning and Embeddings

From feature engineering to representations learned from data

What You Have Built So Far — and What Comes Next

Module 2 Progress

L5 Regression

Linear regression · Ridge · Lasso · R^2 · Cross-validation

L6 Classification

Logistic regression · Decision trees · Confusion matrix
Precision / Recall / F1 · ROC–AUC · Threshold as a business decision

↑ *Both assume a labeled target column y is available.*

Two Gaps This Lecture Fills

Gap 1 — Polynomial Features

The solver can fit curves, but we have not seen how to construct the features that make that possible. Session 1 fills this gap.

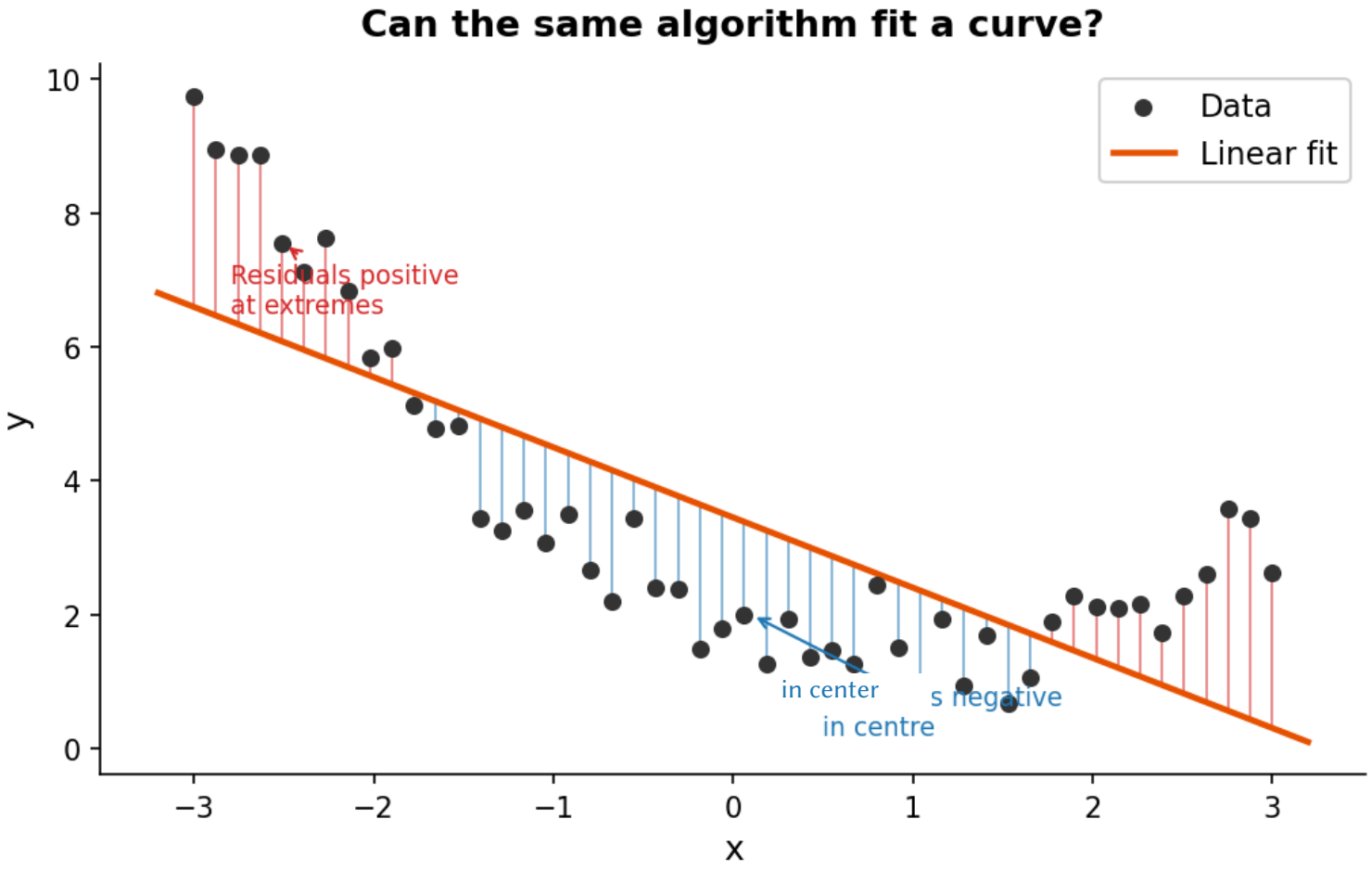
Gap 2 — Supervised Assumes Labels

Every model in L5 and L6 required a labeled target *y*. But labels are often unavailable, expensive, or beside the point. Sessions 2 and 3 address this directly.

Learning Objectives

O1	Articulate two motivations for unsupervised learning; distinguish structure discovery from supervised prediction
O2	Construct polynomial features as new columns in X; apply the unchanged linear regression solver; explain parameter linearity
O3	State the k-means objective J; explain non-convexity and initialization sensitivity; explain why minimizing J alone cannot select k
O4	Apply k-means; evaluate with the elbow method and silhouette scores; justify a choice of k
O5	Explain how representation choice shapes what k-means discovers; select features appropriate to the analytic question
O6	Describe FM embeddings as learned representations; explain how cosine similarity and normalization affect retrieval.
O7	Visualize an embedding space using PCA projection; acknowledge the linear limitation of the projection
O8	Implement a similarity search system: encode a corpus, rank by cosine similarity, verify the results
O9	**Connect polynomial features, k-means representations, and FM embeddings as three instances of the same underlying principle**

[QUESTION] Can the Same Algorithm Fit a Curve?



? Let's Find Out

Linear regression draws a straight line. The data on the left has a clear curve. After two lectures of supervised learning, the question is simple: does the algorithm have to change, or does something else?

We already have a regression solver from L5. The question is whether we can use it unchanged.

Polynomial Features Are Just New Columns in X

EQ1 — Feature Transformation

$$\mathbf{x} \rightarrow \begin{bmatrix} 1 & x & x^2 & x^3 & \cdots & x^d \end{bmatrix}$$

x: original scalar feature · d: polynomial degree (analyst's choice) · Intercept 1 included by convention

EQ2 — Polynomial Regression Model

$$\hat{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d$$

β_0, …, β_d: OLS coefficients for the polynomial feature columns

Key Insight: Parameter Linearity

The model is nonlinear in *x* but linear in the *β* coefficients. The solver, the loss function, and the coefficient interpretation framework are all unchanged. Linearity is a property of the parameters — not of the shape of the fit.

CODE1 — Design Matrix Construction

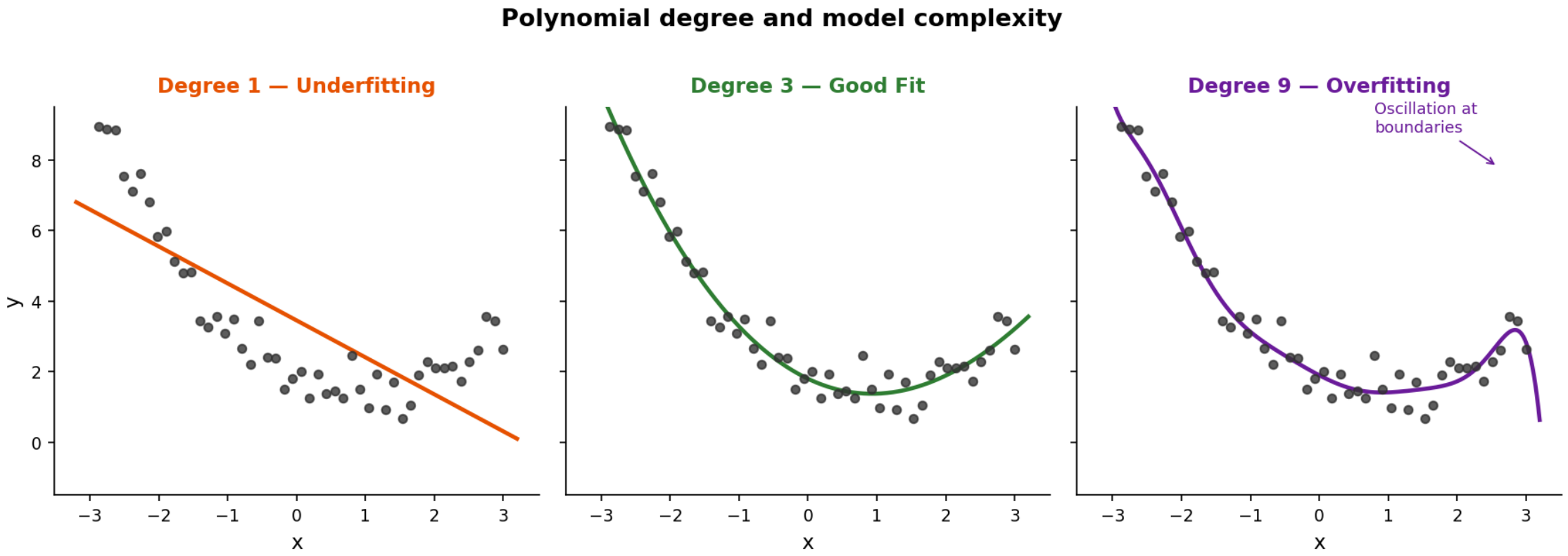
```
from sklearn.preprocessing import (
    PolynomialFeatures
)
import numpy as np

x = np.array([1,2,3,4,5,6]).reshape(-1,
1)
poly = PolynomialFeatures(degree=3)
X_poly = poly.fit_transform(x)
print(X_poly)
# columns: [1, x, x**2, x**3]
```

✓ VERIFY: Execute

Run **CODE1** in Colab. Check rows 1–3: are the powers of *x* correct?

More Features, More Risk — Regularization Returns



Key Insight: Regularization Is Not Specific to Tabular Data

Each additional polynomial term is another column in X — another degree of freedom to overfit. Ridge and Lasso from L5 apply unchanged: they penalize large β coefficients regardless of whether those coefficients multiply x, x^2, or x^9. Selecting the polynomial degree d is itself a model selection problem; more is not better.

🤖 CTA Demo 1: The Same Solver, Radically Different Fits

What We'll Explore

We have two **representations** of the same dataset: raw x and augmented $[x, x^2]$. We will ask the CTA to fit linear regression on each using the identical `LinearRegression` solver. The question is whether the representation change alone accounts for the improved fit — or whether the solver itself must change.

Key Questions for the CTA:

1. Fit `LinearRegression` on raw x. Print the design matrix and residuals. What does the residual pattern reveal?
2. Add x^2 as an explicit new column and refit using the identical solver. How does the fit change? Print both design matrices side by side.
3. Follow-up: "The model is still called linear regression. Why is that not a contradiction?"

Collaboration Pattern: Iterative Refinement · VERIFY: Manually verify the x^2 column for several rows before running the full demo

Feature Engineering is a Representation Decision

The Unifying Principle (Objective 9)

Every feature engineering choice encodes an assumption about what structure matters in the data. When we add x^2 as a column, we assume curvature is relevant. When we normalize, we assume scale is not. The algorithm sees only the matrix we hand it — the representation is already a model of the world.

In the Supervised Setting

The label *y* anchors the choice: a feature is useful if it helps predict the outcome. This gave us polynomial features in Demo 1 — and Ridge/Lasso to control for the features that add noise.

What If There Is No Label?

Without *y*, there is no outcome to anchor the choice. The representation still determines what structure the algorithm can find — but now the analyst must justify the choice on different grounds.

Key Insight: This Principle Recurs Three Times in L7

Polynomial features shape what a supervised model can fit. Feature choice shapes what k-means can find. FM embeddings are a third, qualitatively different answer to the same question. Objective 9 asks you to connect all three as instances of a single idea.

[QUESTION] What If There Is No Label to Predict?

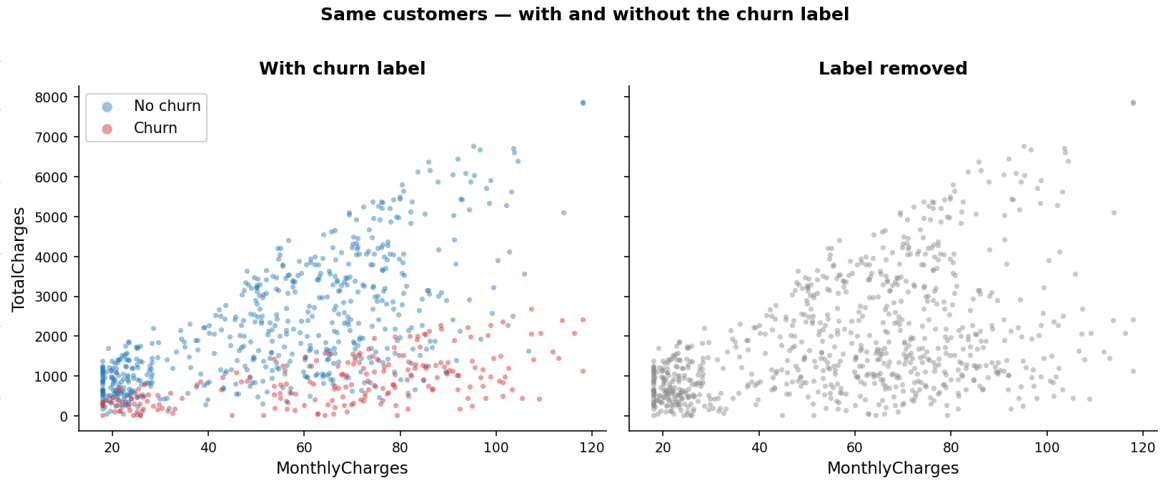
Streaming...	Contract	Paperless...	PaymentM...	MonthlyCh...	TotalCharg...	Churn
No	Month-to-month	Yes	Electronic check	29.85	29.85	No
No	One year	No	Mailed check	56.95	1889.5	No
No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
No	One year	No	Bank transfer (automatic)	42.3	1840.75	No
No	Month-to-month	Yes	Electronic check	70.7	151.65	Yes
Yes	Month-to-month	Yes	Electronic check	99.65	820.5	Yes
No	Month-to-month	Yes	Credit card (automatic)	89.1	1949.4	No
No	Month-to-month	No	Mailed check	29.75	301.9	No
Yes	Month-to-month	Yes	Electronic check	104.8	3046.05	Yes
No	One year	No	Bank transfer (automatic)	56.15	3487.95	No
No	Month-to-month	Yes	Mailed check	49.95	587.45	No
No internet service	Two year	No	Credit card (automatic)	18.95	326.8	No
Yes	One year	No	Credit card (automatic)	100.35	5681.1	No
Yes	Month-to-month	Yes	Bank transfer (automatic)	103.7	5036.3	Yes
Yes	Month-to-month	Yes	Electronic check	105.5	2686.05	No

y

? Let's Find Out

The left panel shows the same customer data you have worked with since L4, but with churn labels removed on the right. What can we say about the structure of this data when the outcome column is gone?

In L5 and L6, the label column y defined which patterns were worth finding. Without it, the question of what counts as "interesting structure" is genuinely open.



Two Independent Reasons to Clustering

Reason 1 — Structure Discovery

Clustering is a legitimate first-choice goal, even when labels exist.

The question is not "what predicts *y*?" but "what patterns are present in the data?"

A retailer might cluster customers to discover natural segments before any hypothesis is formed. A biologist might cluster gene expression profiles without a disease label to guide grouping.

Pursued deliberately even when labels are available — not a fallback.

Reason 2 — Labeling Cost

Annotation is expensive, slow, and sometimes impossible at scale.

Medical imaging datasets may have thousands of scans but only hundreds with expert labels. Web-scale text corpora cannot be fully annotated by hand.

Clustering allows useful structure to be discovered from the raw, unlabeled data — and the discovered clusters can then seed or reduce the annotation effort.

A practical constraint, independent of whether structure discovery is the goal.

The two reasons are independent. Neither subsumes the other.

Unsupervised Learning Asks a Different Question

Definition: Unsupervised Learning

A class of methods that finds patterns, structure, or compact representations in data X without reference to a target variable y. The algorithm has no outcome to predict — only the feature matrix.

	Supervised Learning	Unsupervised Learning
Question asked	What will y be for a new observation?	What patterns exist in X?
Uses y?	Yes — y defines what counts as a good fit	No — y is absent or ignored
Evaluation	Loss against held-out y (MSE, accuracy…)	Requires separate criteria (elbow, silhouette…)
Examples	Linear regression, logistic regression, decision trees	K-means clustering, FM similarity search

Session 1 Summary

Features as Modeling Decisions

1

Polynomial features extend linear regression without changing the solver

Adding x^2, x^3, ... as explicit columns in X lets the same `LinearRegression` solver fit curves.
The model remains linear in its parameters β. Ridge and Lasso apply unchanged as complexity grows with polynomial degree.

2

Feature engineering is a representation decision

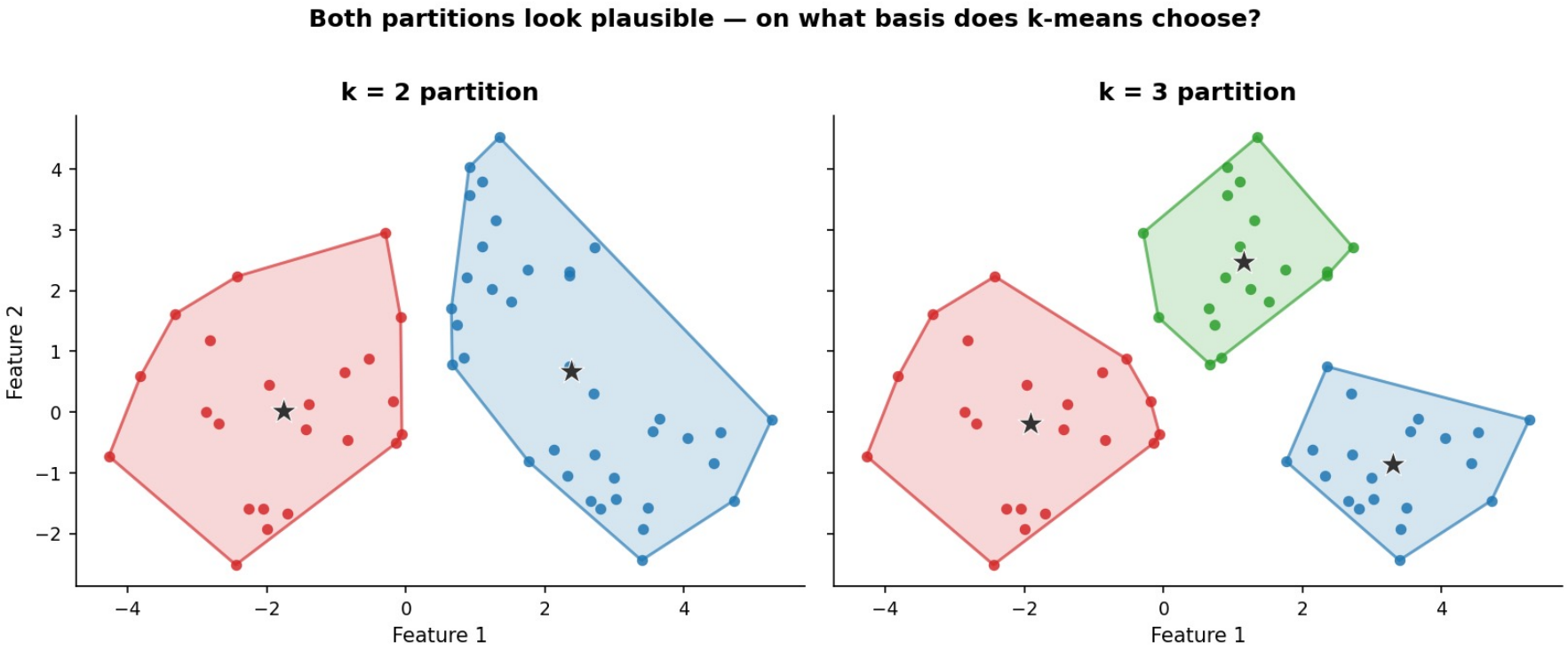
Every feature choice encodes an assumption about what structure matters.
In the supervised setting, the label y anchors that choice. This is the first instance of Objective 9's unifying theme.

3

Unsupervised learning addresses a genuinely different question

Without a target y, the algorithm finds patterns in X itself.
This is not a fallback when labels are unavailable — it is a legitimate and deliberate choice. Session 2 introduces the method and its fundamental difficulties.

[QUESTION] How Does K-Means Decide What a Cluster Is?



? Let's Find Out

The data on the left could plausibly be divided into two groups or three. Both partitions look reasonable. K-means will produce one definite answer — but on what basis does it choose?

SESSION 2 — K-Means: Optimization Without Ground Truth

Session 1 established why we might want to find structure without labels. Session 2 now works through the algorithm that does exactly that — and the fundamental difficulties it creates.

K-Means Minimizes Within-Cluster Variance — and Updates Until It Cannot Improve

EQ3 — Within-Cluster Sum of Squares (WCSS)

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

J: within-cluster sum of squares (lower is tighter) · Non-increasing at every step

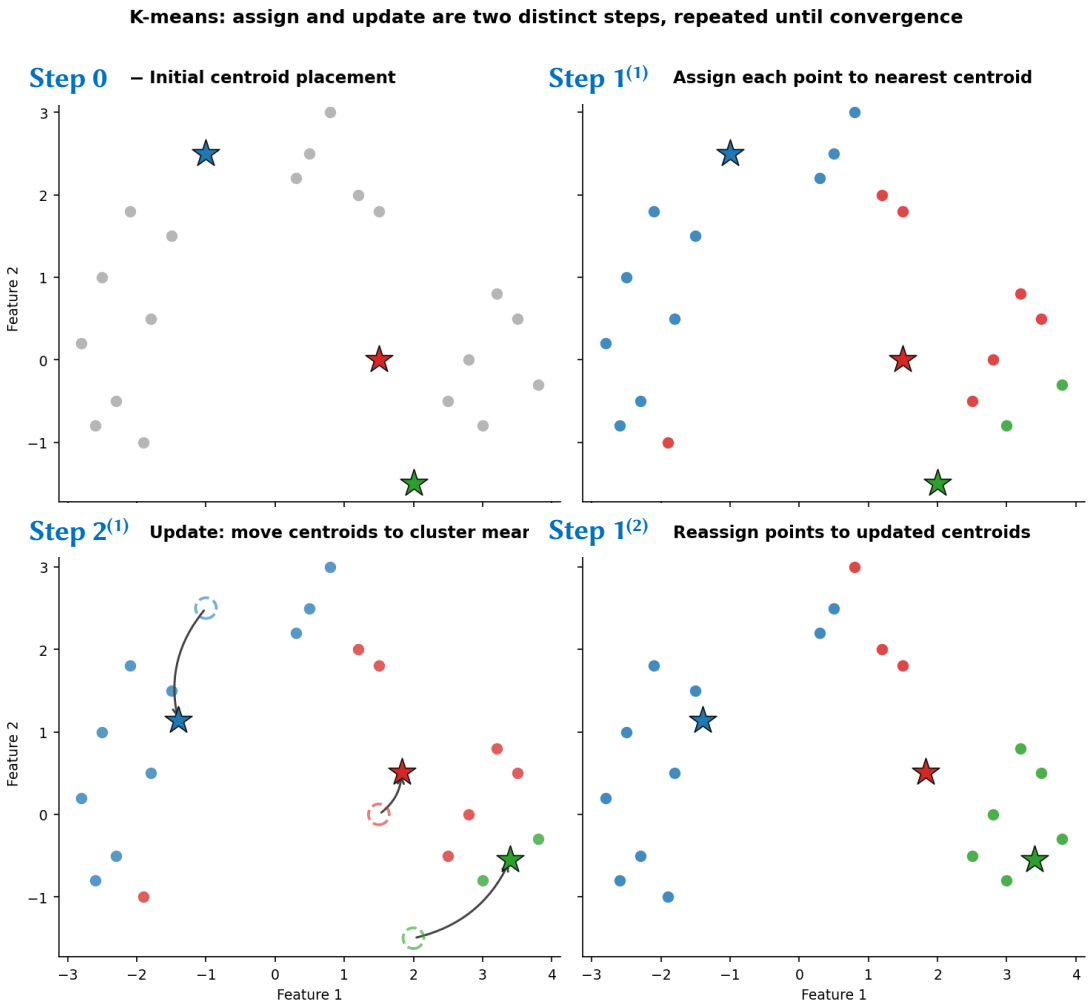
Symbol	Type	Meaning
K	Integer ≥ 1	Number of clusters — chosen by the analyst before running
C_k	Set of points	All observations assigned to cluster k
μ_k	$\mathbb{R}^p$	Centroid of cluster k — the coordinate-wise mean of all $x \in C_k$
$\|\cdot\|$	Euclidean	L2 norm introduced in L2; distance from a point to its centroid

The Two-Step Algorithm

Assign	Each point is assigned to the nearest centroid: $\text{argmin}_k \|x - \mu_k\|$
Update	Each centroid is recomputed as the mean of its assigned points: $\mu_k \leftarrow \text{mean}(C_k)$

Convergence: J is non-increasing at every step.

The algorithm halts when assignments stop changing (equivalently, when J ceases to decrease). Convergence is guaranteed — to a local minimum.



🤖 CTA Demo 2: What K-Means Is Actually Optimizing

What We'll Explore

Implement k-means using NumPy, then run 10 random initializations. Track the non-increasing objective J within each run and compare the final partitions across runs. Finally, ask which cluster count minimizes J—and why that alone is insufficient.

Turn 1

1. Implement k-means in NumPy: assign points to their nearest centroid, then update each centroid to its cluster mean. Handle empty clusters. Print assignments and J at each iteration.

Turn 2

2. Run 10 random initializations on the same data. Report the final J and compare the resulting partitions, allowing for permuted cluster labels. How often is the same grouping found?

Turn 3

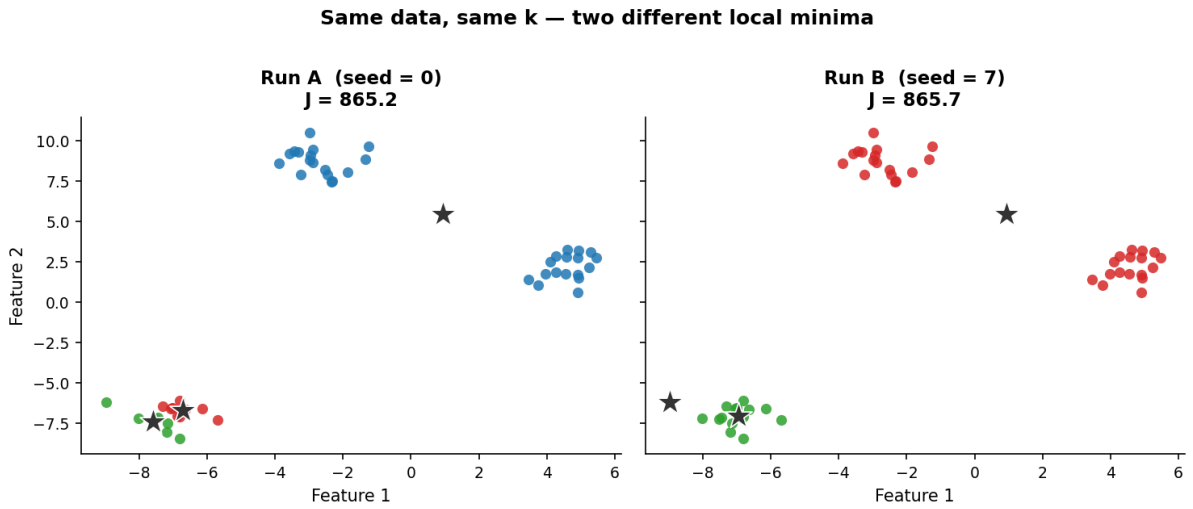
3. Follow-up: "If we just want to minimize J, what value of k should we choose?"

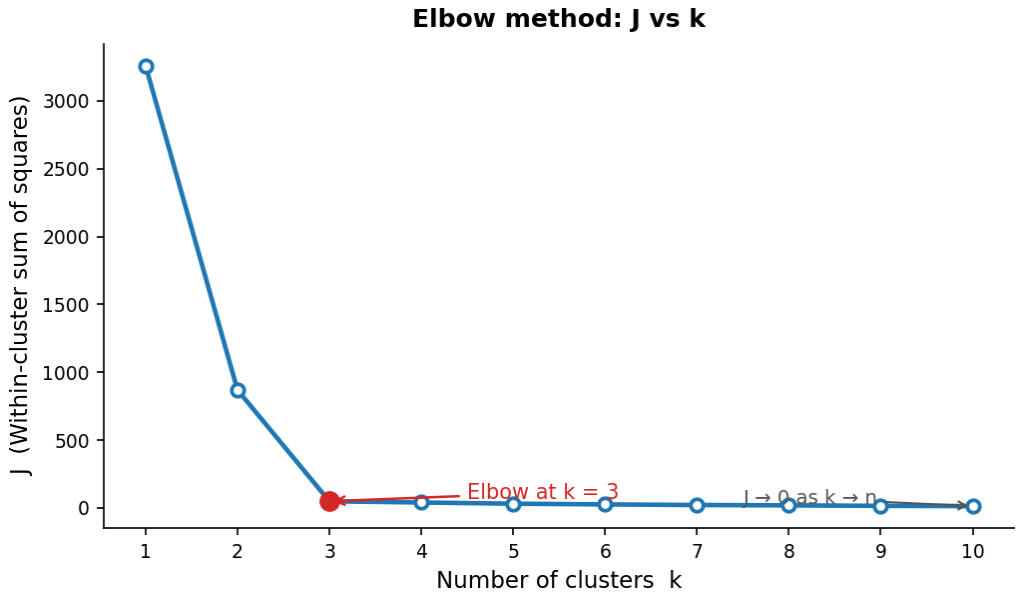
✓ VERIFY: Execute

On three points with known cluster assignments, compute J by hand for one iteration. Confirm that it matches the code output.

Collaboration Pattern: Verification Through Computation · Objectives: O3, O4

Local Minima and Why J Cannot Select K





Why Different Runs Produce Different Answers

Each step cannot increase J, but different initializations can lead to different local solutions. Multiple restarts help without guaranteeing the global minimum. Selecting the cluster count is a separate issue: one cluster per observation gives $J = 0$.

Key Insight: $J = 0$ When $k = n$

Setting $k = n$ assigns every point its own cluster: $J = 0$ exactly.

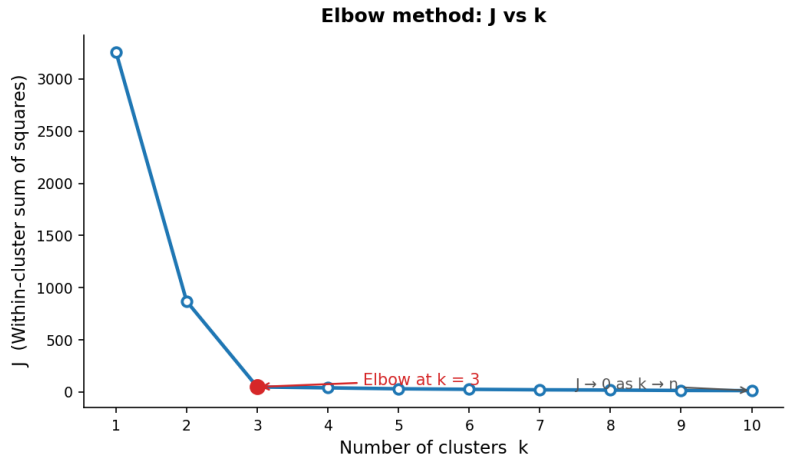
This is why minimizing J alone is logically insufficient to select k — the trivial solution always wins on the training objective.

Two Metrics, Two Perspectives on Cluster Quality

Metric 1 — The Elbow Method

Plot J vs k. Find the "elbow" where further increases yield diminishing reductions in J. This is a heuristic for diminishing returns, not a proof of correctness.

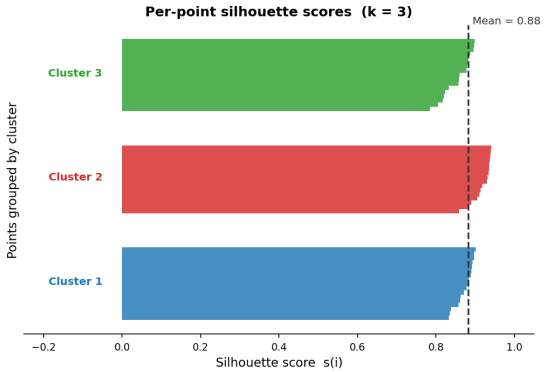
Recall: Where else did we use the elbow method?
What do they have in common?

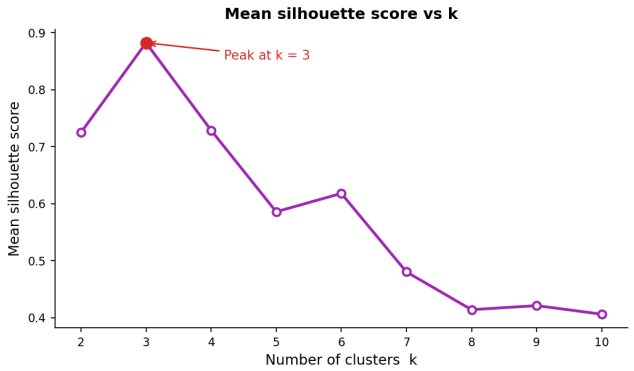


Metric 2 — Silhouette Score

$s(i)$

- near +1: well matched to its cluster.
- near 0: clusters overlap.
- near −1: the point is much closer, on average, to another cluster than to its assigned cluster.





EQ4 — Silhouette Score for Point *i*

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

$a(i)$: mean distance to other points in its cluster · $b(i)$: lowest mean distance to another cluster

Misconception to Correct

A negative score means another cluster is closer on average. It does not prove a coding error; inspect the representation, distance measure, and clustering.

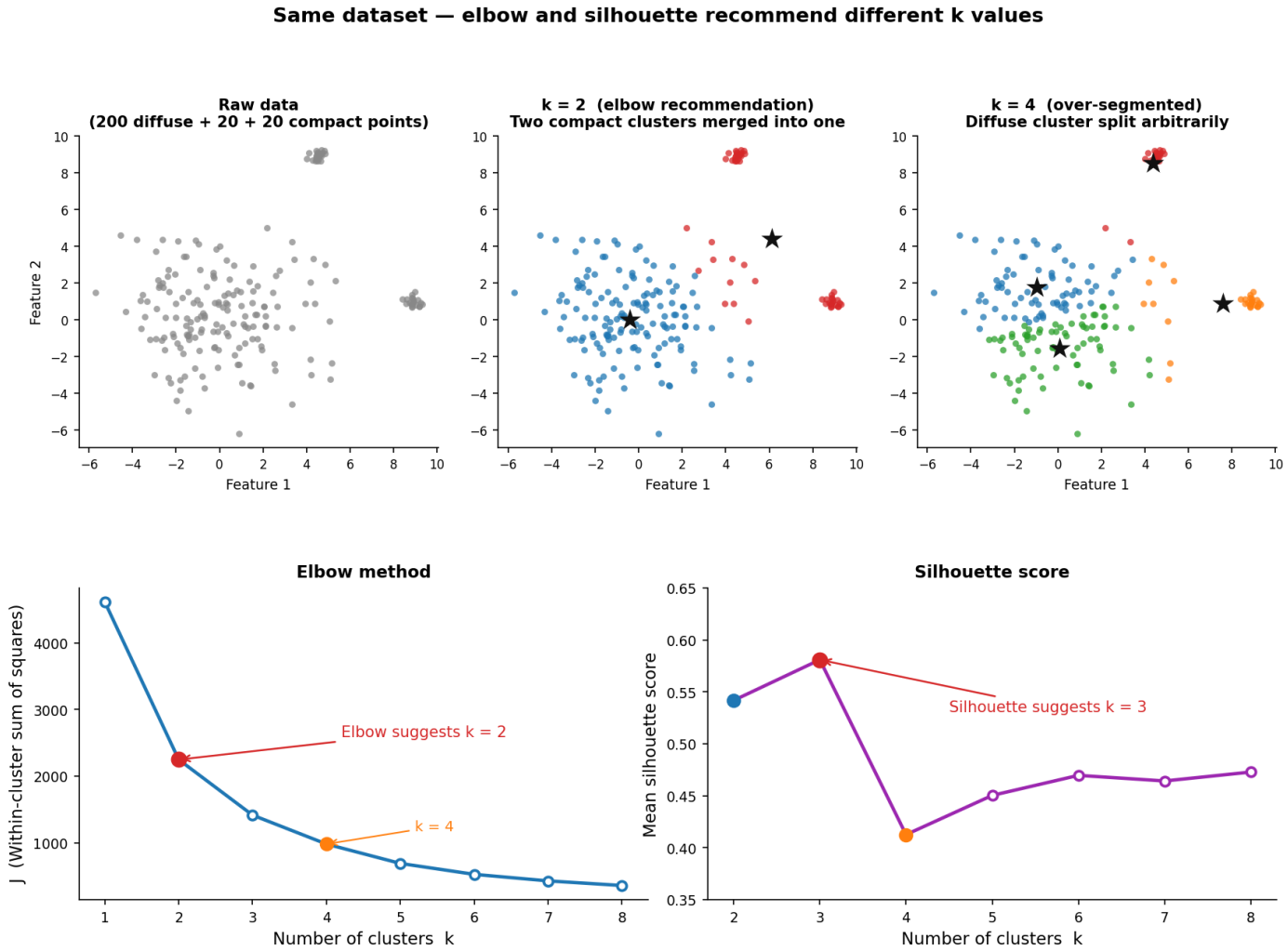
Neither Metric Is Authoritative Without Analytic Purpose

Why They Disagree Here

The elbow reflects reductions in *J*. The silhouette compares within-cluster and between-cluster distances. They reward different features of the geometry, so their preferred cluster counts can differ.

What Resolves the Disagreement

The analyst's question. Decide which groupings are useful for the application, then inspect the clusters in the original features. These plots alone do not establish high business value. Use domain knowledge and external checks.



We now have two principled tools for evaluating a clustering — and we know that neither is sufficient without a clear analytic purpose.

But there is a prior question we have been setting aside: we have been treating the features as given inputs.

What if the features themselves are a modeling decision, one that shapes what k-means can possibly find?

[QUESTION] Does Changing the Features Change What You Find?



? Let's Find Out

Both panels show the same customers. The left uses raw charge amounts; the right uses proportions. Will k-means find the same clusters in both spaces? Which grouping best answers our question?

This is the same Telco dataset from L4 and L6. The representation is the only thing that changed.

CTA Demo 3: Does the Question Change the Features?

What We'll Explore

Use the same Telco customers in two representations: raw charges and ratio features. Run k-means on each and inspect PCA projections. Which clustering is useful? Assess that against the application's purpose and domain context.

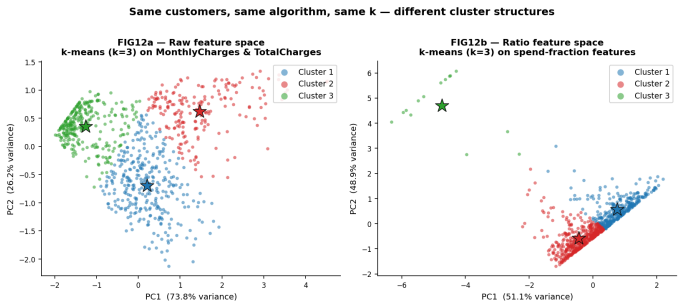
1. **Turn 1**
   Run k-means (k=3) on the raw feature set. Produce a PCA-projected scatter plot with cluster color assignments.

2. **Turn 2**
   Re-run on the ratio feature set. Produce a second PCA scatter plot. Compare the two visualizations.

3. **Turn 3**
   Challenge: "Which clustering is correct?" The CTA cannot answer without external criteria.

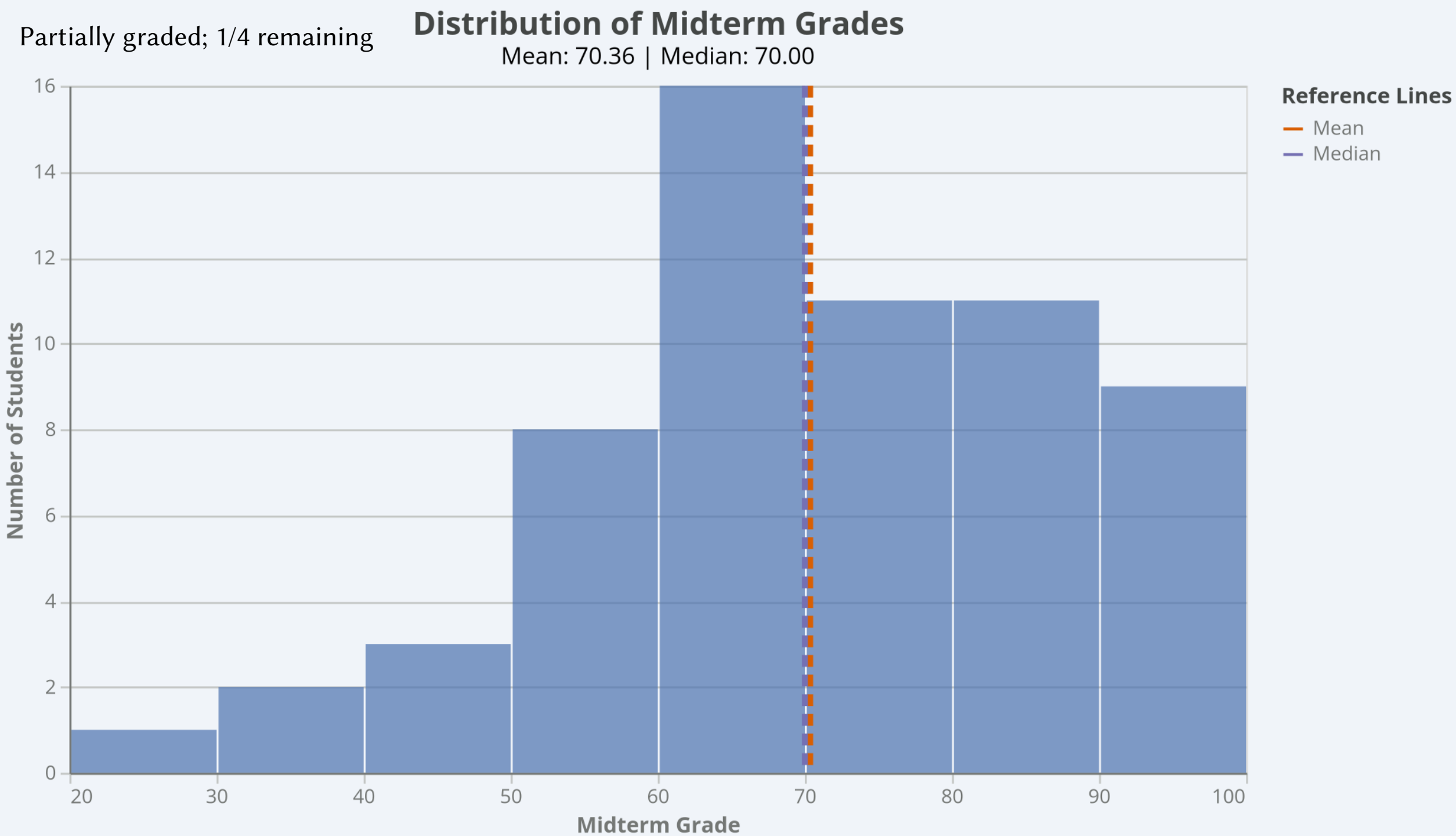
4. **Turn 4**
   Follow-up: "How would a data scientist decide which representation to use?"

ref only



Collaboration Pattern: Challenging and Probing · Objectives: O5, O9

Partially graded; 1/4 remaining



Representation Encodes the Analyst's Question

Formal Statement of Objective 5

The choice of features determines which notion of similarity k-means optimizes. Raw monthly charges make customers with similar absolute spend look alike. Ratio features make customers with similar spending behavior look alike. The algorithm faithfully minimizes J in whichever space it is given — it cannot know which space is appropriate.

Raw Feature Clustering

Clusters group customers by the absolute amounts they pay across service categories. Useful for: volume-based segmentation, identifying high-spend and low-spend tiers, capacity planning.

Analytic question implied: Who spends how much?

Ratio Feature Clustering

Clusters group customers by how they allocate spending across categories, independent of total spend. Useful for: behavioral profiling, product bundle design, churn risk profiling.

Analytic question implied: How does each customer prioritize services?

Key Insight: Both Clusterings Are Valid

Neither is objectively correct. Both are internally consistent solutions that minimize J in their respective feature spaces. Validity is not a property of the clustering alone — it is a relationship between the clustering, the features, and the analytic question the analyst brings to the data.

Human Judgment Selects the Representation

What the CTA Cannot Do

In Demo 3, the CTA produced and explained both clusterings. Choosing between them requires a business question and domain evidence. The CTA reasons about representations; the analyst judges their practical relevance.

Volume Segmentation Question

Goal: identify customers by total monthly spend to target promotions by budget tier.

⟶ Raw features are appropriate. Ratio features obscure the spend signal.

Behavioral Segmentation Question

Goal: identify distinct customer profiles based on how they allocate spending, regardless of total bill size.

⟶ Ratio features are appropriate. Raw features conflate behavior with volume.

Misconception: Feature Engineering Is Data Dredging

Ratio features can express a stated question about spending patterns rather than volume. Justify that choice before examining the result. Data dredging means repeatedly searching for a desired pattern without accounting for the search.

The Same Principle Has Appeared Twice Already

OBJECTIVE 9 — Building the Unifying Thread

Instance 1 — Demo 1 (Session 1)

Adding x^2 as a new column changed what the linear regression solver could fit. The algorithm and the loss function were identical; the representation was the only variable. A curved relationship became visible to the model because the feature space was extended to include curvature.

Instance 2 — Demo 3 (Session 2)

Switching from raw charges to ratio features changed what k-means found. The objective function and the algorithm were identical; the representation was the only variable. A behavioral structure became visible to the algorithm because the feature space encoded proportions.

The Shared Principle

The features presented to an algorithm determine what structure it can discover. This is true whether the algorithm is supervised or unsupervised, parametric or non-parametric. The representation is always a modeling decision.

Both instances used hand-crafted features — features we chose and constructed ourselves. Session 3 asks: what if, instead of choosing a representation ourselves, we could learn one from data?

Session 2 Summary

SESSION 2 — K-Means: Optimization Without Ground Truth

1

Finding a good clustering and selecting its cluster count are separate problems

J is non-increasing during a run, which may end at a local minimum. Multiple restarts help but do not guarantee the global optimum. The objective alone cannot select the cluster count: one cluster per observation gives $J = 0$.

2

The elbow method and silhouette score are complementary heuristics — neither is authoritative

The elbow identifies where reductions in J diminish. The silhouette compares within-cluster and between-cluster distances. Use the application's purpose and external checks to interpret them. A negative silhouette means another cluster is closer on average, not necessarily a coding error.

3

Representation choice is a modeling decision that requires human judgment

Changing the features changes what k-means finds — even with the same algorithm, the same k, and the same objective function.

The analyst's question determines which representation is appropriate.

This is the second instance of the unifying principle that Session 3 will complete.

[QUESTION] Can a Representation Encode Meaning — Not Just Values?

Language <-> Semantics (Meaning) is NOT a one-to-one mapping

Surface-similar, semantically different

"The bank approved the loan."

"The river overflowed its bank."

Same word "bank" — completely different meaning

Surface-different, semantically equivalent

"The flight was canceled."

"They called off the journey by air."

Different words — identical meaning

? Let's Find Out

A vector like `[monthly_charges, tenure]` captures numbers.
Similar wording can have different meanings; different wording can have the same meaning.
Can a learned representation capture this distinction?

Session 3 asks: what if, instead of choosing or engineering a representation ourselves, we could learn one from data?

SESSION 3 — FM Embeddings: Representations Learned From Data

FM Embeddings Map Text to Vectors in a Semantic Space

EQ5 — The Embedding Function

$$f : \text{text} \rightarrow \mathbb{R}^d$$

f: fixed encoder · text: truncated beyond 256 word pieces by default · $\mathbb{R}^d$: vector space · d = 384 for all-MiniLM-L6-v2

Learned Semantic Similarity

Training encourages related sentences to have nearby vectors. This supports semantic search, but task-specific checks remain necessary.

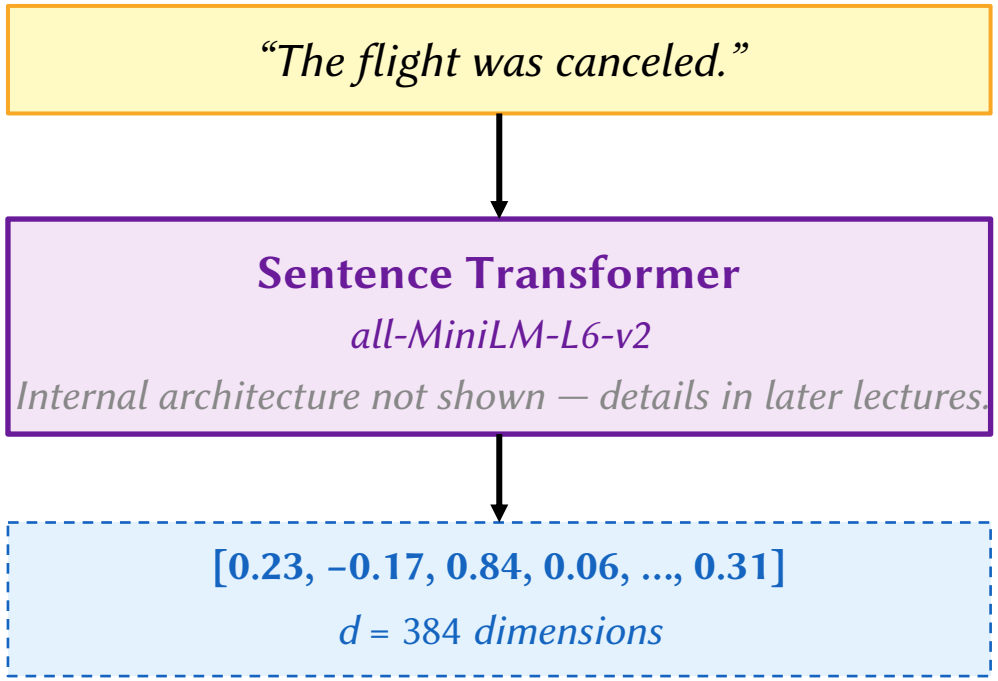
Meaning Is Distributed Across Coordinates

A single coordinate usually has no simple meaning. Semantic information is distributed across coordinates and relationships between vectors.

f is Fixed in This Demo

The encoder stays fixed in this demo and outputs 384 dimensions. Similar texts tend to have nearby vectors, but errors and domain shifts occur.

How an Embedding Is Produced



Same idea applies to image embeddings and unified text-image embeddings!

Cosine and Normalized Euclidean Distance

EQ6 — Cosine Similarity (Reactivated from L2)

$$\cos(\theta) = \frac{\mathbf{u}\cdot\mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$$

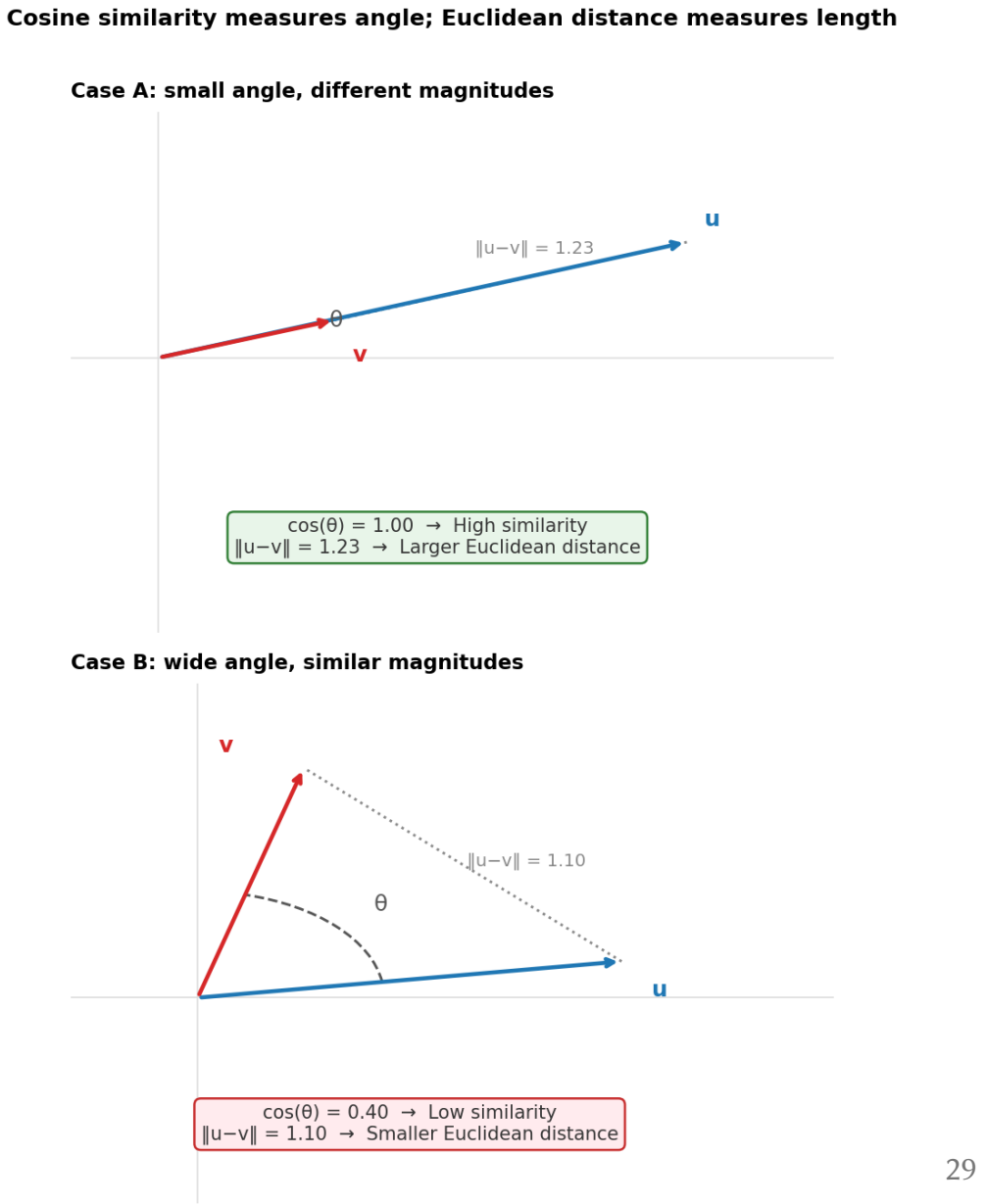
u, v: two embedding vectors · u·v: dot product · ‖u‖, ‖v‖: Euclidean norms · θ: angle between vectors · Range: [−1, +1]

Why Divide by the Norms?

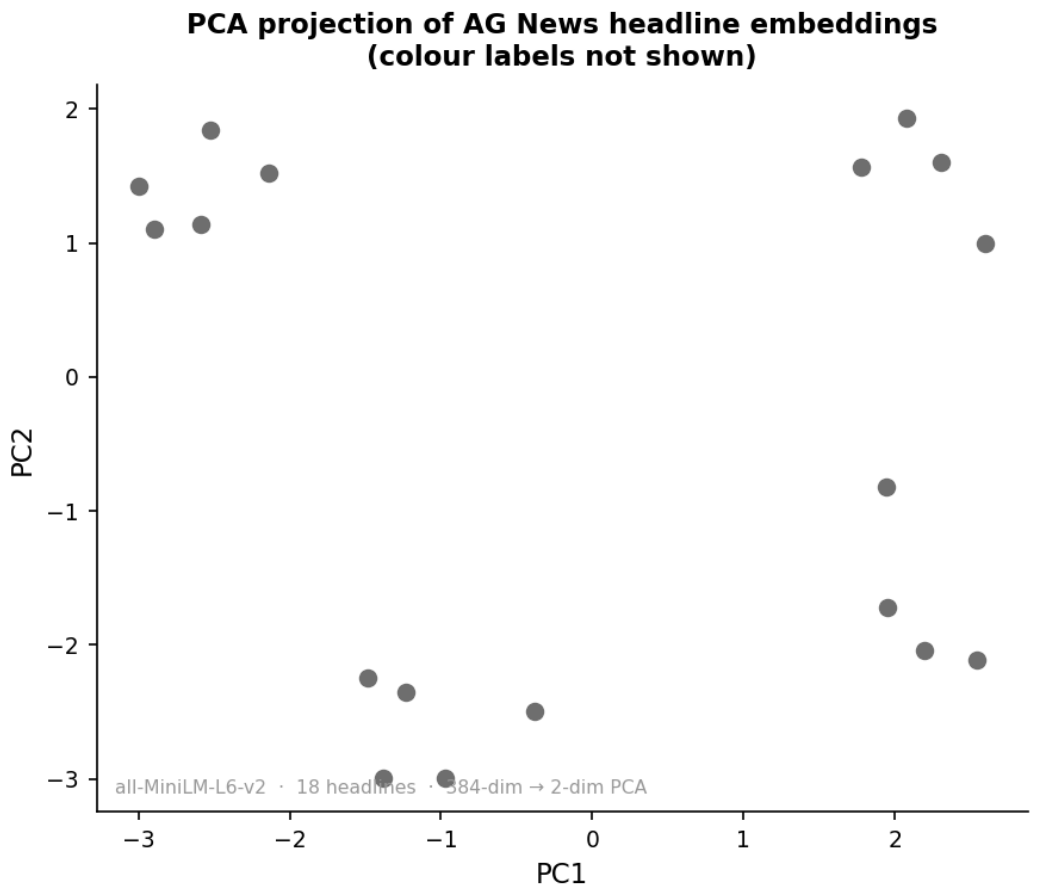
Cosine similarity compares direction, removing magnitude. This matches the sentence-pair training objective used by all-MiniLM-L6-v2. The appropriate measure depends on the model and task.

Same Ranking After Unit Normalization

For unit vectors, $\|u - v\|^2 = 2 - 2\cos(\theta)$. Descending cosine similarity and ascending Euclidean distance give identical rankings. Neither measure automatically avoids concentration in high dimensions.



PCA Reveals Structure in the Embedding Space



ref only

PCA on the Embedding Matrix

Each row of the embedding matrix is a 384-dimensional vector. PCA finds the two directions of maximum variance and projects the points onto them — compressing 384 dimensions into a 2D scatter we can inspect visually.

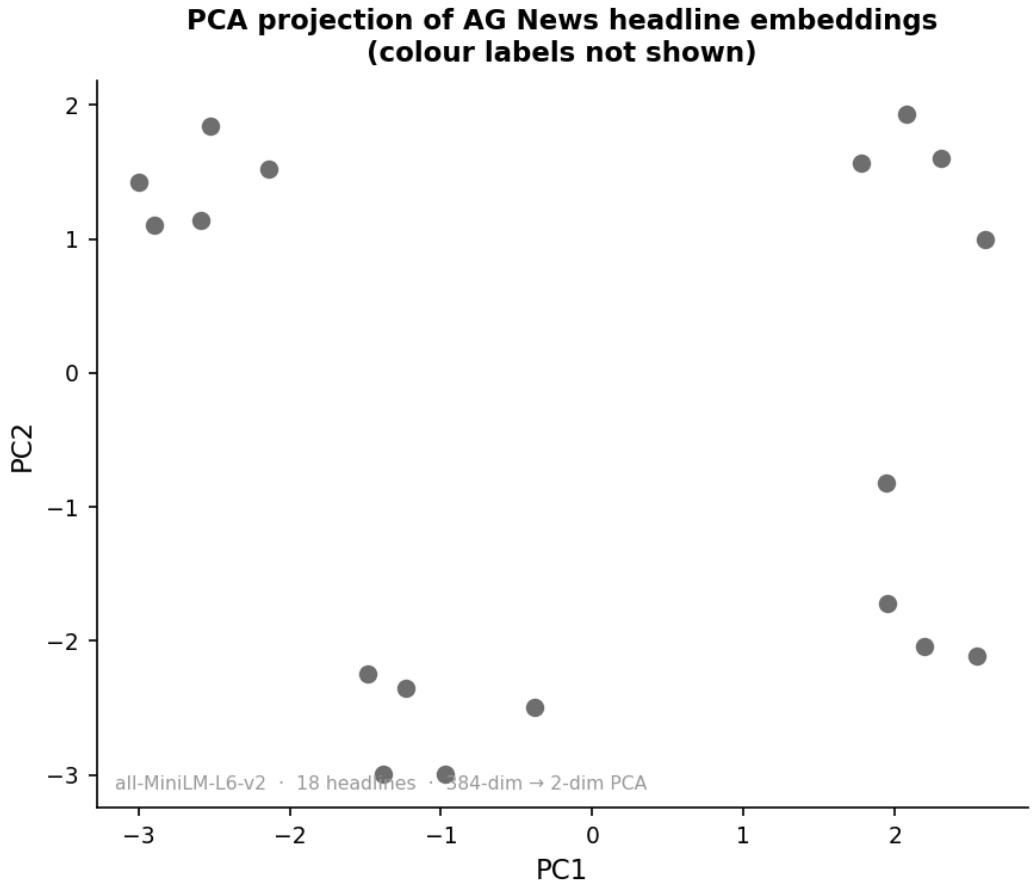
What Structure Would Confirm Our Hypothesis?

If FM embeddings encode semantic meaning, headlines about World news, Sports, Business, and Technology should cluster into recognizable regions of the PCA projection — even though the model was given no topic labels.

Acknowledged Limitation

PCA captures only linear structure. Semantic relationships in a 384-dimensional embedding space may involve curved or non-linear geometry that a 2D PCA projection cannot fully reveal. The projection is a useful inspection tool, not a complete picture of the embedding space.

[QUESTION] Does the Embedding Space Encode Structure We Recognize?

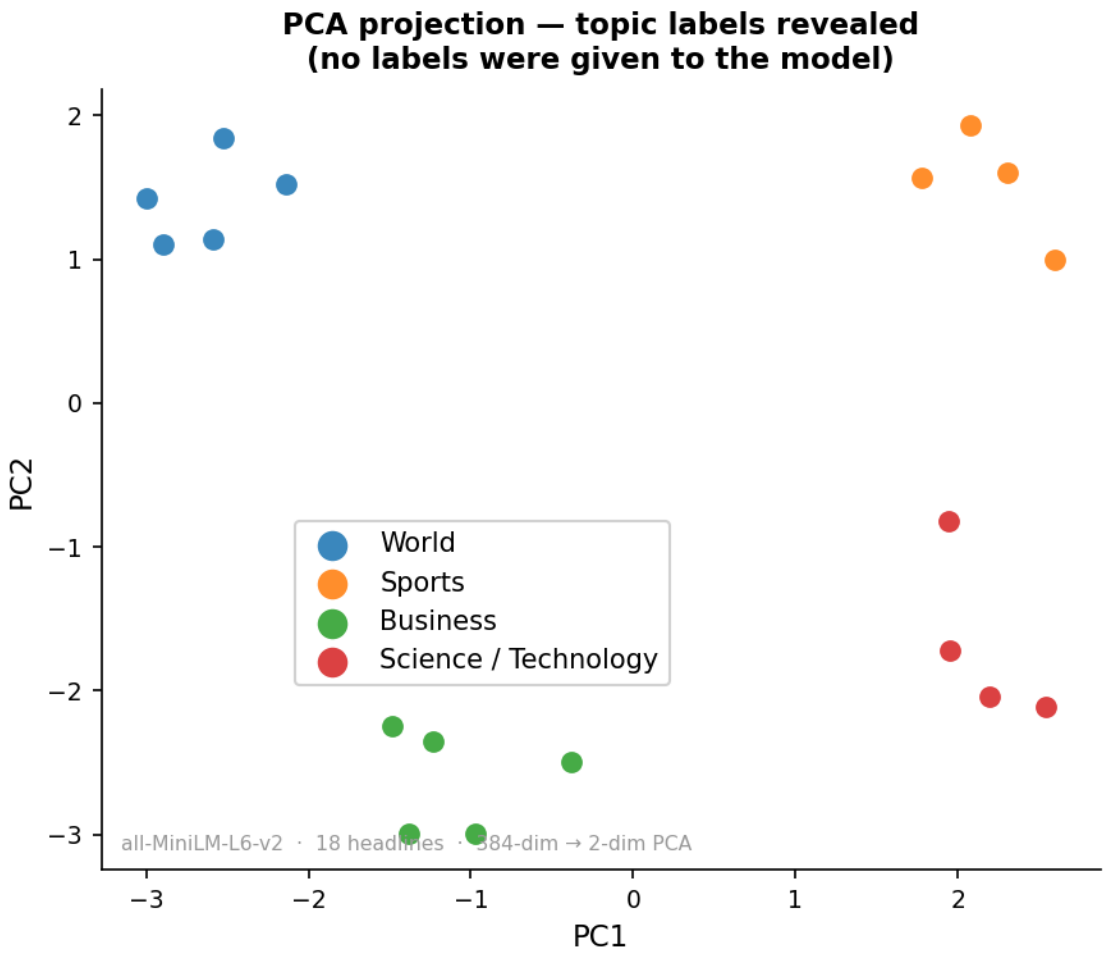


ref only

? Let's Find Out

The projection on the left was produced by applying PCA to the raw embedding vectors — no labels were provided to the algorithm.
Do you see distinct groups? Where would you draw the boundaries, and what do you think each group represents?

The Clusters Correspond to Semantic Topics



ref only, let's demo it.

Geometry Can Reflect Meaning

This reference plot illustrates groups that align with topic categories. The demo uses topic labels only to color the plot, not to encode the texts or fit PCA. Check whether the actual results show this structure.

What This Demonstrates

Training encourages semantically related texts to have nearby representations. A 2D projection may reveal this structure, but clear topic clusters are not guaranteed.

Connection to Objective 9

Polynomial features and the k-means examples required hand-crafted feature choices. Here, we reuse a pretrained representation without task-specific feature engineering. Its usefulness on a new corpus still needs evaluation.

🤖 CTA Demo 4: Building a Semantic Search System

What We'll Explore

Encode headlines, visualize with PCA, and rank query matches by cosine similarity. Unit-normalize the vectors and compare Euclidean rankings: they should agree. No task-specific training is needed; check whether the retrieved texts answer the query.

1. **Turn 1**
   Encode corpus and query with `SentenceTransformer`; plot PCA with topic colors; return the top 3 matches and `cosine_similarity()` scores.

2. **Turn 2**
   Unit-normalize query and document vectors. Compare descending cosine scores with ascending Euclidean distances. Do the rankings agree?

3. **Turn 3**
   Ask the CTA to derive why cosine similarity and Euclidean distance give the same ranking for unit-normalized vectors.

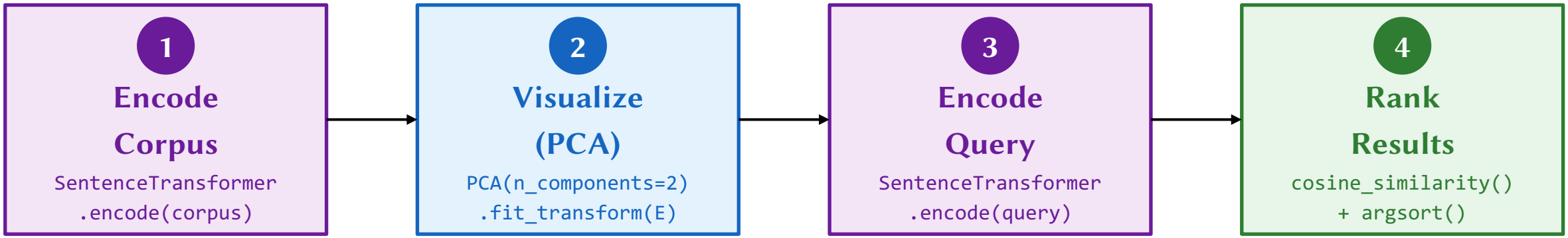
✓ VERIFY: Execute

Check the 384-dimensional cosine score using a direct dot-product calculation. Then verify the formula by hand on a small 2D example.

Collaboration Pattern: Structured Requests · Objectives: O6, O7, O8, O9 · CTA writes code; students run in Colab

The Pipeline: Encode, Visualize, Query, Rank

Similarity Search Pipeline



What Makes This Possible

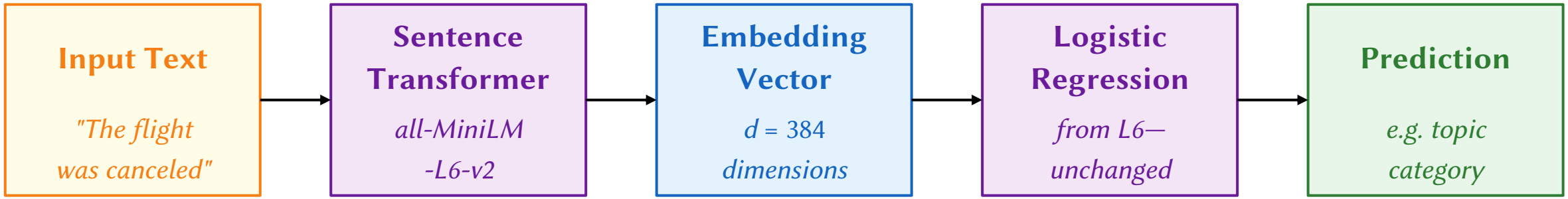
No new labeled data or fine-tuning is needed for this demo. Steps 1 and 3 use the same pretrained encoder. Rank the original 384-dimensional vectors; PCA is only for visualization. Domain knowledge is still needed to assess whether the results are useful.

Scope Note

This is a conceptual demonstration of the search mechanism on 15–20 documents held in memory. Production similarity search at scale uses approximate nearest-neighbor indices (e.g. FAISS, Pinecone).

Embeddings Transfer — and Feed Supervised Models

Embeddings as Input Features to a Downstream Model



No Labels Needed for Encoding

Steps 1–3 require no labeled data. The FM representations transfer because they were trained on diverse data at scale — this is the property that makes FM embeddings qualitatively different from hand-crafted features.

A Supervised Classifier

Step 4 uses the logistic regression method from L6. The embeddings are input features; the classifier learns from labeled examples. Evaluate on held-out data to test whether these features are useful.

Transfer is the Key Property

We do not fine-tune the encoder in this demo. Pretraining supports transfer to new texts, but performance must be evaluated on the target task and domain.

Three Instances of the Same Principle

OBJECTIVE 9 — The Unifying Thread Completed

	Polynomial Features	K-Means Representation	FM Embeddings
How chosen	Analyst selects degree *d* explicitly	Analyst selects which features to include and how to construct them	Learned automatically during pretraining on massive corpora; analyst selects model
Prior knowledge encoded	Curvature up to degree *d*; nothing higher unless added	Domain assumption in feature choice (volume vs behavior vs other)	Semantic relationships in language as learned from billions of text examples
Structure discoverable	Polynomial relationships in *x* up to degree *d*	Clusters defined by proximity in the chosen feature space	Semantic similarity in new texts; transfer must be evaluated
Human judgment required	Select degree *d*; decide if curvature is a valid assumption	State the analytic question; decide which representation it implies	Select or verify model; interpret what the embedding space encodes

Homework 3, Group Project checkpoint (brief statements of progress)

Group Project

- Presentations Nov 18 and Nov 25. ~10 min for each group
- Report due Nov 17 at 11:59PM; requirements will be released

See Moodle for the confirmed schedule.

Check the current course announcements for submission deadlines, release dates, and group project details.

Questions?

CONTACT INFORMATION



yangliu.mit.edu

Instructor: Prof. Yang Liu

Email: yliu12@hku.hk

Office Hours: 5:00 - 5:50 PM, Thursdays

Location: CB-714, Chow Yei Ching Building

Let me know if you want remote!

🤖 **CTA is your 24/7 interactive and responsive learning partner!**

TA contacts

Tuesday 4-5PM

Yushi Du:
u3012725@connect.hku.hk
https://us04web.zoom.us/j/78561597481

Zhengyang Hu:
u3010250@connect.hku.hk
https://hku.zoom.us/j/9021481973

Thursday 3-4PM

Jun Dai:
jundai332@connect.hku.hk
https://rochester.zoom.us/j/97202452759

Lulu Cai:
u303011209@connect.hku.hk
https://hku.zoom.us/j/5010926726

TA office hours on Zoom! Two slots (1 hour each) on Tuesdays and Thursdays