L03: K-Nearest Neighbors & K-Means

Overview

Methods and Algorithms

MSc Data Science

Spring 2026

Outline

Can a Machine Learn from Its Neighbors?

- Banks have millions of past transactions — some fraudulent, most legitimate
- How do we classify a brand-new transaction as fraud or not?
- How do we find natural customer groups when nobody has labeled them?

XKCD #1838 by Randall Munroe (CC BY-NC 2.5) — Today: learning from neighbors and discovering groups

What Is Classification?

Classification is like **sorting email into spam or not-spam**: we learn from labeled examples, then predict labels for new ones.

Everyday examples:

- **Spam detection:** past emails labeled spam/not-spam train a filter for new emails
- **Medical diagnosis:** patient records with known conditions help diagnose new patients
- **Fraud flagging:** historical transactions labeled fraud/legit guide future alerts

---

Classification = supervised learning: we learn from labeled examples

What Is Clustering?

Clustering is like **organizing a messy bookshelf by topic** — there are no labels, you just find natural groups based on similarity.

Everyday examples:

- **Customer segments:** group shoppers by purchasing behavior without predefined categories
- **News article grouping:** bundle related stories together automatically
- **Song playlists:** streaming services group songs by style and mood

Clustering = unsupervised learning: discover structure without labels

Classification vs Clustering Side by Side

Classification

- Has labeled training data
- Goal: **predict** a label for new data
- Supervised learning — a teacher provides answers

Clustering

- No labels at all
- Goal: **discover** natural groups
- Unsupervised learning — no teacher, just patterns

KNN solves classification; K-Means solves clustering — same letter K, different meanings

Why Do Similar Things Behave Similarly?

This is an intuition we use every day:

- **Medicine:** patients with similar symptoms tend to have similar diagnoses
- **Real estate:** houses in similar neighborhoods tend to have similar prices
- **Banking:** borrowers with similar financial profiles tend to have similar default rates

The core idea: **nearness in data space implies similarity in outcome.**

KNN turns this everyday intuition into a precise algorithm

Why Do Banks Need Customer Segments?

- **Targeted products:** premium cards for high-value customers, starter accounts for new ones
- **Risk management:** group loans by risk profile to set appropriate interest rates
- **Marketing:** design retention campaigns for at-risk customers before they leave

Real-world impact: A major UK bank identified 6 customer segments, increasing product cross-sell by 15%.

Segmentation transforms raw data into actionable business strategy

What Will You Learn Today?

By the end of this lecture, you will be able to answer:

1. How does KNN use neighbors to classify new data?
2. How does K-Means find clusters step by step?
3. How do you choose the right value of K for either method?
4. When should you use KNN vs K-Means?

Road Map: Problem $\rightarrow$ Method $\rightarrow$ Solution $\rightarrow$ Practice

---

By end of lecture: classify with KNN, cluster with K-Means, choose K

The Two Problems We Solve Today

Fraud Detection

- Labeled transactions (fraud / legitimate)
- Goal: predict whether a new transaction is fraud
- This is **classification**

Customer Segmentation

- No labels — just raw customer data
- Goal: discover natural groups of customers
- This is **clustering**

Important: K in KNN = number of neighbors; K in K-Means = number of clusters — completely different!

Same letter K, fundamentally different meanings — watch for this!

A Concrete Example: 5 Customers

Training Data

Age	Income	Label
25	30k	Legit
30	50k	Legit
45	80k	Fraud
50	90k	Fraud
35	40k	Legit

The Question

A new Customer #6 arrives: Age = 42, Income = 75k.

Which group does Customer #6 belong to?

Imagine plotting these 5 points on a scatter plot — Customer #6 sits near the fraud cases. KNN formalizes this intuition.

This small example is the starting point — KNN scales this to millions

How Does KNN Work? Three Simple Steps

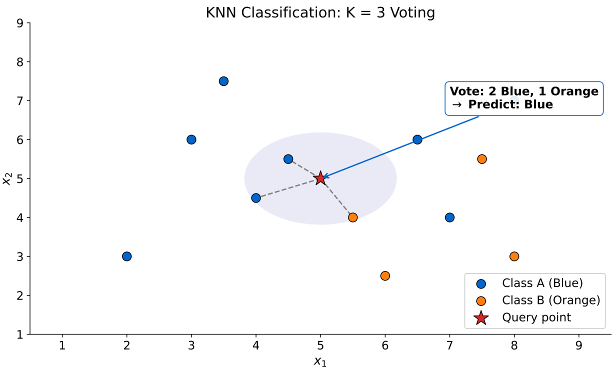
1. **Measure distance** to every past example in the training data
2. **Find the K=3 closest neighbors** — the 3 most similar past cases
3. **Take a majority vote** among those neighbors

Worked example: Customer #6 has 3 nearest neighbors: 2 fraud, 1 legit $\rightarrow$ **predict fraud** (majority wins).

KNN is called a "lazy learner" (delays all computation until prediction time) — it stores all training data and only computes distances when asked.

KNN is "lazy": it stores all data and only computes at prediction time

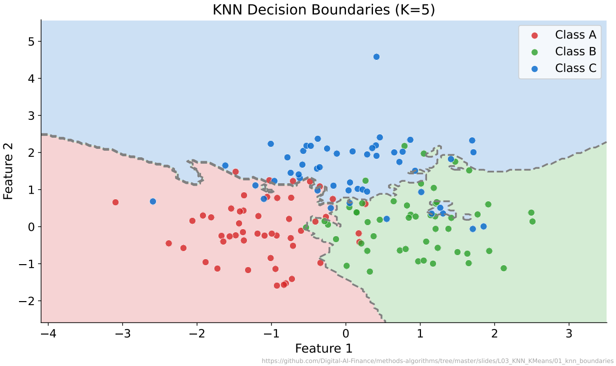
KNN Step by Step: A Worked Example



- We measure distance from the new point (star) to all training points
- The 3 closest neighbors are 2 blue, 1 orange $\rightarrow$ majority vote = blue
- If we used K=5 neighbors, the vote might change — K matters!

The choice of K directly controls how many neighbors vote

What Happens When K Changes?



- K=1: jagged boundary follows every point — may memorize noise (overfitting)
- K=15: smooth boundary — may miss important patterns (underfitting)
- Sweet spot: use cross-validation (testing on held-out data) to find the best K

Small K = flexible but noisy; large K = smooth but may miss detail

How Do We Measure Distance?

Euclidean Distance

The straight-line distance between two points:

$$d = \sqrt{\sum_{i=1}^{p}(a_i - b_i)^2}$$

where a and b are two data points with p features.

Worked Example

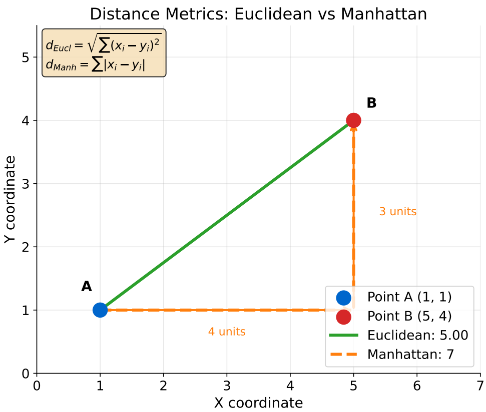
Customer A: age = 30, income = 50k
Customer B: age = 25, income = 55k

$$d = \sqrt{(30-25)^2 + (50-55)^2}$$
$$d = \sqrt{25 + 25} = \sqrt{50} \approx 7.07$$

Warning: Always standardize features (rescale to similar ranges) first — otherwise income (thousands) dominates age (decades)!

Always standardize features first — otherwise income (thousands) dominates age (decades)

What Do Different Distance Measures Look Like?



https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L03_KNN_KMeans/02_distance_metrics

- Euclidean draws circular neighborhoods — straight-line distance
- Manhattan draws diamond neighborhoods — block-by-block distance
- The choice of distance metric (way of measuring) changes which points count as "nearest"

Different metrics = different neighborhoods = different predictions

How Does K-Means Find Clusters?

Three steps, repeated until nothing changes:

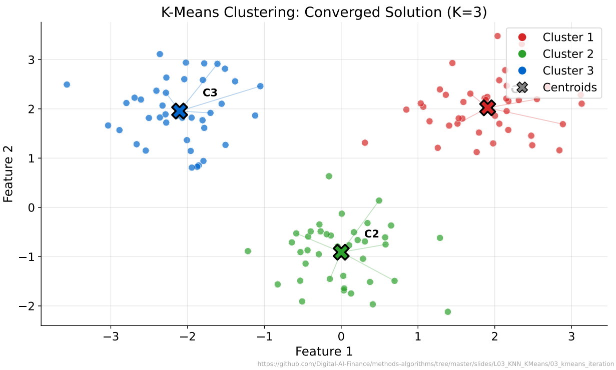
1. **Pick** $K=3$ **starting points** called centroids (cluster centers)
2. **Assign each data point** to its nearest centroid
3. **Move each centroid** to the center (average position) of its group

Repeat steps 2–3 until centroids stop moving.

Analogy: Like rearranging seats at a party until everyone is closest to their table center.

K-Means is like rearranging seats at a party until everyone is closest to their table center

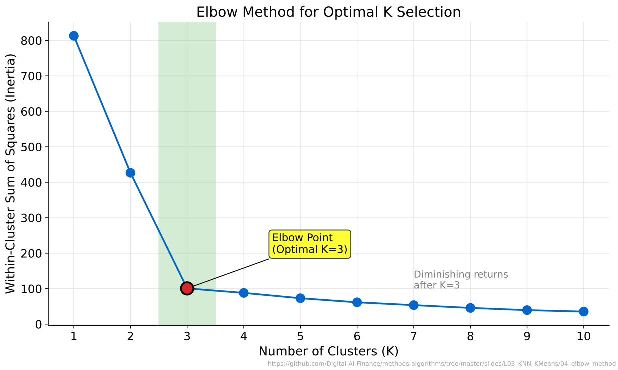
Watching K-Means Iterate



- Stars show centroids (cluster centers) moving toward their groups
- Colors show which centroid each point is assigned to
- After a few rounds, centroids stop moving — we have found our clusters

Convergence (stopping) is guaranteed — the algorithm always finishes

How Do We Choose K Clusters? The Elbow Method



- WCSS (total distance from points to their centroids) always decreases with more clusters
- The "elbow" is where adding more clusters stops helping much
- Here $K=3$ clusters looks like the best choice — diminishing returns after that

The elbow is sometimes hard to see — that is why we also use silhouette analysis

Why Does Starting Position Matter?

K-Means++ in plain English: Instead of picking random starting centroids, spread them out — pick each new centroid far from existing ones.

Why it matters: Random starts can lead to bad results because centroids may clump together in one region of the data.

Analogy: Imagine choosing 3 meeting points in a city — you would spread them out, not put all 3 on the same street.

K-Means++ is the default in Python's scikit-learn library — you get it automatically.

K-Means++ is the default in scikit-learn — always use it

K-Means Has Limitations — Know Them!

Strengths

[+] Fast, simple, and easy to interpret

[+] Works well on round (spherical) clusters of similar size

[+] Scales to large datasets

Limitations

[-] Assumes round clusters of similar size — struggles with elongated shapes

[-] Sensitive to outliers (extreme values) — one extreme point shifts the centroid

[-] Must specify K clusters in advance (unlike DBSCAN, a density-based method)

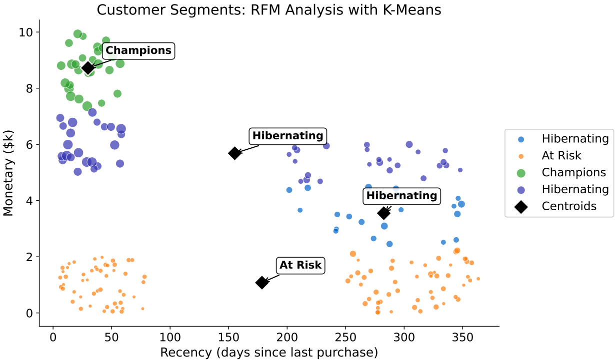
If clusters look elongated or have very different sizes, K-Means may not be the right tool

How Do KNN and K-Means Compare?

Property	KNN	K-Means
Task	Classification	Clustering
Learning	Supervised (has labels)	Unsupervised (no labels)
K means	Number of neighbors	Number of clusters
Training	None — lazy learner	Iterative centroid updates
Output	Class label	Cluster ID

Despite sharing K, these solve fundamentally different problems

Finance: Customer Segmentation with K-Means



- RFM = Recency (days since last purchase), Frequency (number of purchases), Monetary (total spend)
- K-Means groups customers: Champions (high R,F,M) get loyalty rewards; At-Risk (low R) get retention campaigns
- **Example:** Customer A: R=5, F=20, M=\$5000 $\rightarrow$ Champion. Customer B: R=180, F=2, M=\$50 $\rightarrow$ At-risk

Each segment gets tailored products and communication — segments drive strategy

Finance: Can KNN Detect Fraud?

The class imbalance (unequal groups) problem: Fraud is less than 1% of transactions.

If KNN predicts "no fraud" for everything → 99% accuracy but catches **zero** fraud cases!

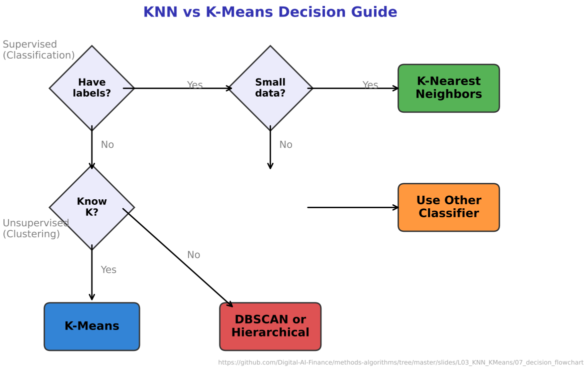
Solutions:

- **Oversample** fraud cases — duplicate rare examples to balance the training data
- **Weight fraud neighbors** more heavily — give more influence to minority class votes
- **Measure with Precision-Recall** instead of accuracy — reward catching actual fraud

---

In fraud detection, missing a fraud case is far more costly than a false alarm

Which Method Should You Choose?



- Have labels? $\rightarrow$ supervised methods like KNN
- No labels, round clusters? $\rightarrow$ K-Means
- Weird-shaped clusters? $\rightarrow$ DBSCAN (density-based) or hierarchical clustering

Start simple (K-Means or KNN), add complexity only if results are poor

Hands-on Exercise

1. **KNN Classification:** Apply KNN to classify customers — vary K neighbors and see how predictions change
2. **K-Means Segmentation:** Segment customers with K-Means — interpret what each group means in business terms
3. **Choosing K:** Compare the elbow method and silhouette analysis to choose K clusters

---

Exercises progress from guided implementation to open-ended analysis

Key Takeaways

KNN

- Non-parametric (no fixed model), lazy learner
- Small K = flexible but noisy; large K = smooth but may miss detail
- **Scale your features!**

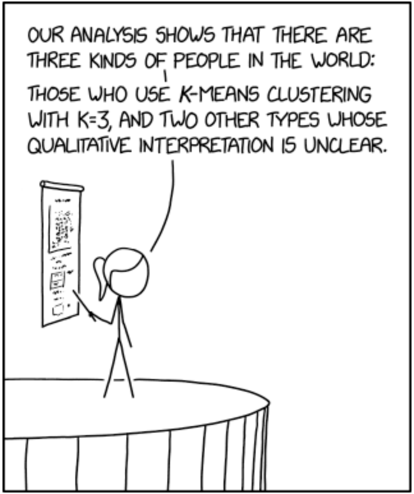
K-Means

- Iterative assign-and-update, always converges (stops)
- Use K-Means++ for smart starting
- Check K with elbow and silhouette methods

Both methods: Feature scaling is critical, and K means different things in each algorithm.

Get the distance right, get the result right — both methods depend on distance

Until Next Time...



"Even K-Means would struggle to cluster the ways students misuse K-Means."

Next session: L04 — Random Forests

XKCD #2731 by Randall Munroe (CC BY-NC 2.5) — Clustering is easy; knowing when to cluster is hard