K-Nearest Neighbors: Theory and Practice

Full Technical Lecture

Methods and Algorithms

MSc Data Science

A Bank Receives 10,000 Transactions Per Hour — Which Are Fraud?

The Scene

Banks process massive transaction volumes every hour, yet only about 0.1% are fraudulent — each costing thousands.

- **Situation:** 10,000 transactions per hour
- **Complication:** Only **10 are fraud**, but each costs **$**5,000+
- **Question:** Can past fraud cases help catch new ones?

Core Idea

What if we find the **most similar** past transactions and let them vote?

KNN: find the most similar past cases and let them vote on the new one

What Will You Learn in This Lecture?

Learning Objectives

1. **Derive** the distance-based classification rule and analyze its error bounds
2. **Evaluate** how K controls the bias–variance tradeoff in practice
3. **Analyze** why feature scaling and dimensionality affect KNN performance
4. **Apply** cross-validation to select optimal K for financial datasets

Prerequisites

Basic probability, linear/logistic regression concepts (L01–L02).

Why It Matters

KNN is the foundation of all distance-based methods and instance-based reasoning.

Bloom's levels: derive (4), evaluate (5), analyze (4), apply (3)

What Is KNN in Plain English?

The simplest classifier imaginable

Find the K training points closest to the query. Let them vote. Majority wins.

- No training phase — just store the data
- Non-parametric: no assumed functional form
- Each prediction has an intuitive explanation

Analogy: "Asking your 5 nearest neighbors which restaurant to go to."

Key Vocabulary

- **Query point:** the new observation to classify
- **Neighbor:** a training point close to the query
- **Distance:** how we measure "close"
- **Majority vote:** most common label among K neighbors
- K: the number of neighbors to consult

No formulas needed yet — KNN is purely intuitive at this stage

Where Does KNN Fit in Machine Learning?

Parametric Models

- Learn fixed coefficients (e.g., logistic regression)
- Fast prediction — just plug in
- Model complexity is fixed after training

Example: $\hat{y} = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2)$

Non-Parametric Models

- Store all training data (e.g., KNN)
- Flexible decision boundaries
- Complexity grows with data size

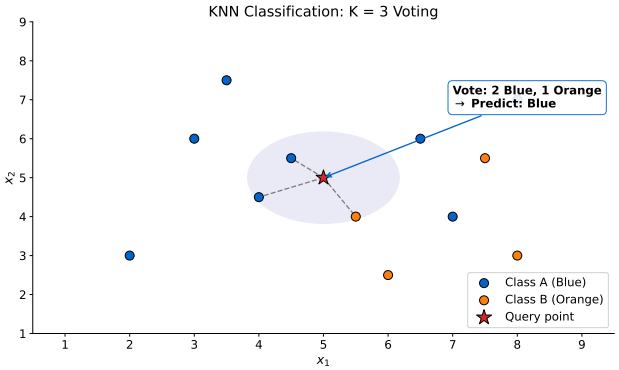
Example: Find K nearest, vote on class

Bridge

KNN is the simplest non-parametric classifier. Understanding it builds intuition for all distance-based methods.

Parametric = learn parameters. Non-parametric = memorize data.

How Does KNN Classify a New Point?



The Algorithm

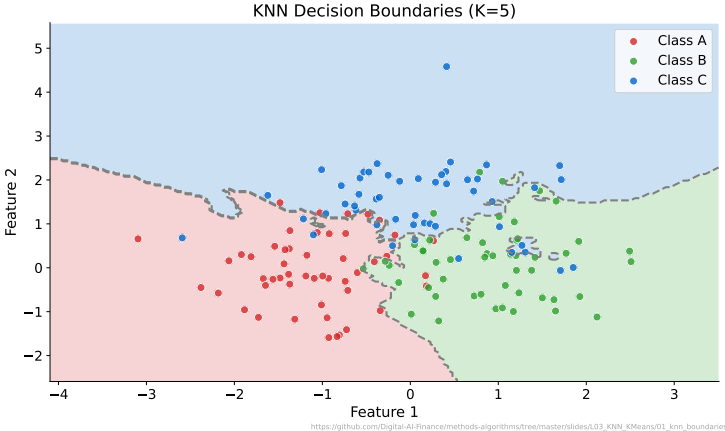
1. Compute distance from query to **all** training points
2. Sort by distance (ascending)
3. Take the K nearest neighbors
4. Count class votes $\rightarrow$ assign majority class

Key Insight

No "learning" happens — prediction = a lookup + a vote.

KNN is a lazy learner: all computation happens at prediction time

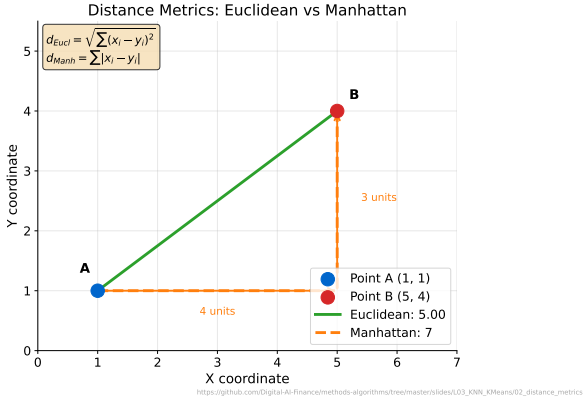
What Do Different Decision Boundaries Look Like?



K=1: jagged, overfits noise. K=15: smooth, may underfit. K=5: balanced tradeoff.

Decision boundaries become smoother as K increases

How Do We Measure Distance?



Three Distance Functions

Euclidean (straight line):

$$d_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{p}(x_i - y_i)^2}$$

Manhattan (city blocks):

$$d_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{p}|x_i - y_i|$$

Minkowski (generalized):

$$d_q(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{p}|x_i - y_i|^q\right)^{1/q}$$

Euclidean (q=2) and Manhattan (q=1) are special cases of Minkowski

Worked Example: Which Transaction Is the Nearest Neighbor?

Training data (4 past transactions):

ID	Amount ($)	Time (hr)	Foreign	Label
#1	500	13	1	Fraud
#2	420	15	1	Fraud
#3	200	10	0	Legit
#4	480	12	1	Legit

Query: Amount=450, Time=14, Foreign=1 (Euclidean distance)

ID	Distance
#1	$\sqrt{(450-500)^2 + (14-13)^2 + (1-1)^2} = \sqrt{2501} \approx 50.0$
#2	$\sqrt{(450-420)^2 + (14-15)^2 + (1-1)^2} = \sqrt{901} \approx 30.0$
#3	$\sqrt{(450-200)^2 + (14-10)^2 + (1-0)^2} = \sqrt{62517} \approx 250.0$
#4	$\sqrt{(450-480)^2 + (14-12)^2 + (1-1)^2} = \sqrt{904} \approx 30.1$

$K=3$: Neighbors #2, #4, #1 → 2 Fraud, 1 Legit → **Predict FRAUD**

Note: Amount dominates the distance — we will fix this with feature scaling (Slide 12)

The KNN Decision Rule Formally

KNN Classifier

$$\hat{y}(\mathbf{x}) = \arg\max_{c \in \mathcal{C}} \sum_{i \in N_K(\mathbf{x})} \mathbb{1}(y_i = c)$$

where $N_K(\mathbf{x})$ = set of K nearest neighbors of $\mathbf{x}$.

Probability Estimate

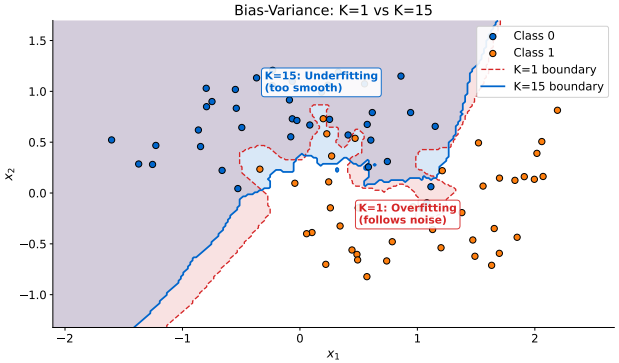
$$\hat{P}(Y{=}c \mid \mathbf{x}) = \frac{1}{K} \sum_{i \in N_K(\mathbf{x})} \mathbb{1}(y_i = c)$$

Translation: Count the votes among the K nearest neighbors. The class with the most votes wins.

Example from Slide 9: $\hat{P}(\text{Fraud} \mid \mathbf{x}_q) = \frac{2}{3} = 0.67, \quad \hat{P}(\text{Legit} \mid \mathbf{x}_q) = \frac{1}{3} = 0.33$

The indicator function $\mathbb{1}(\cdot)$ returns 1 if true, 0 otherwise

How Does K Control Bias and Variance?



The Tradeoff

Small K (e.g., 1):

- Low bias, **high variance**
- Complex, noisy boundaries

Large K (e.g., 50):

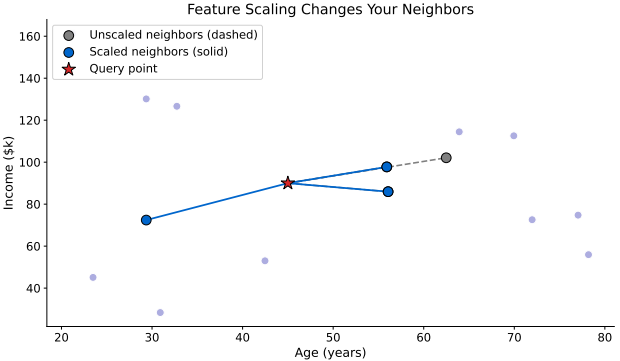
- **High bias**, low variance
- Smooth, potentially wrong boundaries

Goal

Find the K that minimizes **test error**.

This is the fundamental tradeoff in all of machine learning — not just KNN

Why Must We Scale Features?



The Problem
If Amount $\in [0, 10000]$ and Time $\in [0, 24]$, distance is **dominated by Amount**.

The Solution
StandardScaler: transform each feature to mean$= 0$, std$= 1$:

$$z_i = \frac{x_i - \bar{x}_i}{\sigma_i}$$

Rule

Always scale features *before* computing distances.

Fit the scaler on training data only — transform test data with the same parameters

The Curse of Dimensionality

The Problem
In high dimensions, **all points become roughly equidistant**.

- Volume of unit hypersphere $\to 0$ as $d \to \infty$
- Data needed grows exponentially: $n \propto k^d$
- Nearest neighbor is barely closer than the farthest

Practical Guidance
KNN works well when $p < 20$ features.

- **Mitigation 1:** PCA to reduce dimensions ($\to$ L05)
- **Mitigation 2:** Feature selection (keep only informative features)
- **Mitigation 3:** Use approximate nearest neighbors (e.g., Annoy, FAISS)

Key Insight

Distance loses meaning in high dimensions — "closeness" stops discriminating.

This curse affects ALL distance-based methods, not just KNN

Error Bound: How Good Can KNN Be?

Cover & Hart Theorem (1967)

As $n \to \infty$, the 1-NN error rate $R_{\text{1-NN}}$ satisfies:

$$R^* \leq R_{\text{1-NN}} \leq 2R^*(1 - R^*)$$

where R^* is the Bayes optimal error rate.

What does this mean?

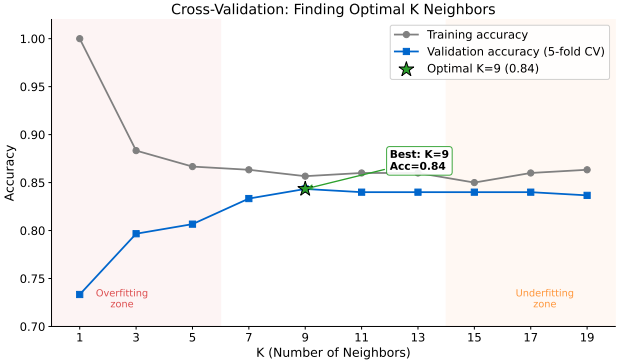
- 1-NN error is at most **twice** the theoretical minimum
- If Bayes error is 5%: $R_{\text{1-NN}} \leq 2 \times 0.05 \times 0.95 = 9.5\%$
- Remarkable guarantee for such a simple algorithm

Caveat

Requires $n \to \infty$ and fixed dimensionality. In practice, performance depends on K, scaling, and p.

Cover & Hart (1967), "Nearest neighbor pattern classification," IEEE Trans. Info. Theory

How Do We Select the Best K?



5-Fold Cross-Validation

1. For $K = 1, 3, 5, \ldots, 21$
2. Split training data into 5 folds
3. Train on 4 folds, test on 1
4. Average accuracy across folds
5. Pick K with highest mean CV accuracy

Practical Tips

Use **odd** K for binary tasks (avoids ties). Try $K \in [1, \sqrt{n}]$.

Cross-validation prevents overfitting to the training set when tuning K

Weighted KNN: Not All Neighbors Are Equal

Standard KNN
All K neighbors vote equally:

- Neighbor at distance 1 counts the same as neighbor at distance 10
- Sensitive to outliers among the K neighbors

Weighted KNN
Closer neighbors get more weight:

$$w_i = \frac{1}{d(\mathbf{x}, \mathbf{x}_i)^2}$$

- Nearby neighbors dominate the vote
- Less sensitive to large K

Worked example: 3 neighbors at distances d=1, 3, 5

Distance	Weight $1/d^2$	Label	Weighted Vote
1.0	1.000	Fraud	1.000
3.0	0.111	Legit	0.111
5.0	0.040	Legit	0.040

Fraud: 1.00 vs. Legit: 0.15 → **Predict FRAUD**

In sklearn: `KNeighborsClassifier(weights='distance')`

Interpreting KNN Predictions

Why KNN is inherently interpretable

For a fraud detection prediction with K=5:

Neighbor	Amount	Time	Foreign	Distance	Label
#1	$480	14:30	Yes	0.12	Fraud
#2	$460	13:50	Yes	0.18	Fraud
#3	$445	14:10	Yes	0.21	Fraud
#4	$470	15:00	No	0.35	Legit
#5	$500	13:00	Yes	0.40	Legit

$\hat{P}(\text{Fraud}) = 3/5 = 0.60$ → **Flagged as Fraud**

Explanation Template

"This transaction was flagged because 3 of its 5 most similar past transactions were confirmed fraud — all had similar amount, time, and foreign status."

KNN gives case-based explanations: show the evidence, not abstract coefficients

KNN Regression Rule

$$\hat{y}(\mathbf{x}) = \frac{1}{K} \sum_{i \in N_K(\mathbf{x})} y_i$$

Predict the **average** of the K nearest neighbors' target values.

Worked example: Predict house price from 3 nearest neighbors

Neighbor	Size (sqm)	Distance	Price (k)
#1	82	0.15	320
#2	78	0.22	290
#3	85	0.31	345

$\hat{y} = \frac{320+290+345}{3} = 318.3$k

Weighted: $\hat{y}_w = \frac{320/0.15^2+290/0.22^2+345/0.31^2}{1/0.15^2+1/0.22^2+1/0.31^2} \approx 314.5$k

In sklearn: `KNeighborsRegressor(n_neighbors=3, weights='distance')`

Computational Complexity: The Speed Problem

Brute Force

$O(n \times p)$ per query — must scan ALL training points.

- 1M training, 100 features = 100M operations per prediction
- **Training**: $O(1)$ — just store the data
- **Prediction**: $O(n \times p)$ — expensive!

Acceleration Structures

- **KD-Trees**: $O(p \log n)$ per query (low p)
- **Ball Trees**: $O(p \log n)$ per query (any metric)
- **Approximate NN**: sub-linear (FAISS, Annoy)

sklearn

`algorithm='auto'` picks the best structure for your data.

Bottom line: KNN prediction is *slow* compared to parametric models. This limits real-time applications.

KNN trades training speed for prediction speed — opposite of neural networks

When to Use KNN — and When Not To

Strengths

- ✓ No training phase needed
- ✓ Handles multi-class naturally
- ✓ Non-linear decision boundaries
- ✓ Intuitive case-based explanations
- ✓ Simple to implement and tune

Limitations

- × Slow prediction ($O(np)$)
- × Curse of dimensionality ($p > 20$)
- × Requires feature scaling
- × Stores entire training set in memory
- × Sensitive to irrelevant features

Rule of Thumb

Use KNN when: few features ($p < 20$), moderate n, need interpretability. Avoid when: large n, high p, real-time prediction required.

KNN is an excellent baseline — always try it before complex models

KNN vs Logistic Regression for Fraud Detection

Criterion	KNN	Logistic Regression
Interpretability	Case-based (show neighbors)	Coefficient-based (log-odds)
Non-linearity	Yes (inherently)	No (needs feature engineering)
Training speed	$O(1)$ — just store	$O(np)$ — optimization
Prediction speed	$O(np)$ — slow	$O(p)$ — fast
Feature scaling	**Required**	Not required
High dimensions	Degrades ($p > 20$)	Handles well (with regularization)
Real-time use	Limited	Excellent

When to Choose Which?

KNN: Few features, need case-based audit trail, non-linear patterns. **LR:** Many features, real-time scoring, regulatory transparency.

In banking: logistic regression for real-time scoring, KNN for post-hoc investigation

Hands-On: Build a Fraud Detector with KNN

Complete sklearn workflow

```
# Step 1: Scale features
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score

scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s  = scaler.transform(X_test)

# Step 2: Fit and evaluate
knn = KNeighborsClassifier(n_neighbors=5, weights='distance')
scores = cross_val_score(knn, X_train_s, y_train, cv=5)
print(f"CV Accuracy: {scores.mean():.3f} +/- {scores.std():.3f}")

# Step 3: Final prediction
knn.fit(X_train_s, y_train)
y_pred = knn.predict(X_test_s)
```

Full notebook: `L03_knn.ipynb` on Colab

Remember: fit scaler on training data only, then transform both train and test

Key Takeaways

Concepts

- Distance metrics (Euclidean, Manhattan, Minkowski)
- Majority vote classification rule
- Bias–variance tradeoff via K
- Curse of dimensionality
- Cover–Hart error bound

Practice

- **Always** scale features before KNN
- Use cross-validation to select K
- Consider weighted KNN for noisy data
- Watch out for high dimensions ($p > 20$)
- Use KD-Trees / Ball Trees for large n

One Sentence Summary

KNN classifies by finding the most similar past observations and letting them vote — simple, powerful, and inherently interpretable.

KNN is the starting point for understanding all distance-based and instance-based methods

What's Next: Unsupervised Learning with K-Means

From labeled to unlabeled data

What we just learned (KNN):

- Uses **labels** to classify
- Supervised: given correct answers
- Goal: predict the label of new points

What's coming (K-Means):

- Has **no labels** at all
- Unsupervised: discover structure
- Goal: find natural groups in data

Teaser

Same distance idea, completely different goal: K-Means finds customer segments without predefined groups. Can a bank discover its own customer types?

Next lecture: K-Means Clustering — from distance-based classification to distance-based grouping

Closing Thought

"Sometimes the simplest ideas work best — ask the 5 nearest transactions."

XKCD #1838 by Randall Munroe (CC BY-NC 2.5)