CS109A: Introduction to Data Science
Lecture 04: k-Nearest Neighbors Regression

Harvard University

Fall 2024

- **Course:** CS109A: Introduction to Data Science
- **Lecture:** Lecture 04: k-Nearest Neighbors (kNN) Regression
- **Instructor:** Pavlos Protopapas
- **Objective:** Understand statistical modeling fundamentals, the kNN algorithm, model evaluation using MSE and R^2, and the train/validation/test split

Key Summary

This lecture introduces **statistical modeling**—the process of finding mathematical relationships between variables to make predictions. We focus on **k-Nearest Neighbors (kNN)**, an intuitive algorithm that predicts values by looking at "similar" examples in the training data. We also learn how to evaluate models using **Mean Squared Error (MSE)** and **R-squared** **(R^2)**, and why we must split our data into **training, validation, and test** sets. By the end, you'll understand how to build, evaluate, and compare predictive models.

Contents

1 Key Terminology

Before diving into modeling, let's establish a common vocabulary:

Table 1: *Essential Modeling Terminology*

Term	**Also Known As**	**Description**
Response Variable (y)	Target, Dependent Variable, Outcome	The variable we want to predict
Predictor Variables (X)	Features, Independent Variables, Covariates	Variables used to make predictions
Design Matrix (X)	Data Matrix, Feature Matrix	Matrix of all predictors ($n \times p$)
Statistical Model ($\hat{f}$)	Estimator, Predictor	Our approximation of the true relationship
Hyperparameter	Tuning Parameter	Values set by humans before training (e.g., k in kNN)
Loss Function	Cost Function, Objective Function	Measures how wrong the model is
MSE	Mean Squared Error	Average of squared prediction errors
R^2	R-squared, Coefficient of Determination	How much better than the baseline model
Training Set	—	Data used to fit/train the model
Validation Set	Dev Set	Data used to select hyperparameters
Test Set	Holdout Set	Data used ONLY for final evaluation

2 Introduction to Statistical Modeling

2.1 The Prediction Problem

We often want to predict one variable based on others:

- Predict **TikTok views** based on video length, posting time, and past performance
- Predict **movie ratings** based on user history and demographics
- Predict **product sales** based on advertising budget

In this lecture, we'll use the **Advertising Dataset**:

- 200 markets (observations)
- 3 predictors: TV, Radio, Newspaper budgets (in $1,000s)
- 1 response: Sales (in 1,000 units)

Goal: Build a model to predict Sales given advertising budgets.

2.2 Response vs. Predictor Variables

Not all variables are equal. There's an asymmetry:

Definition:

Response and Predictor Variables

- **Response Variable** (y): The outcome we're trying to predict
  - Often harder to measure, more important, or influenced by other variables
  - Example: Sales
- **Predictor Variables** (X): The inputs we use to make predictions
  - Variables we can measure or control
  - Example: TV budget, Radio budget, Newspaper budget

2.3 Mathematical Notation

- y: Response vector of length n (one value per observation)
- X: Design matrix of size $n \times p$ (n observations, p predictors)
- **Convention**: Capital letters = matrices, lowercase = vectors

Example:

Design Matrix Structure With $n = 5$ observations and $p = 3$ predictors:

X **(Design Matrix,** 5×3**):**

TV	Radio	Newspaper
230.1	37.8	69.2
44.5	39.3	45.1
17.2	45.9	69.3
151.5	41.3	58.5
180.8	10.8	58.4

y **(Response Vector,** 5×1**):**

Sales
22.1
10.4
9.3
18.5
12.9

Warning

Shape Matters in Code!

In pandas and sklearn:

- `X.shape` returns `(n, p)` — 2D matrix
- `y.shape` returns `(n,)` (Series) or `(n, 1)` (DataFrame)

The difference between `(n,)` and `(n, 1)` can cause errors. Always check `.shape`!

3 What is a Statistical Model?

3.1 The True Relationship vs. Our Approximation

Analogy:

The Ice Cream Analogy Imagine there exists a **perfect ice cream** recipe—the ideal combination of flavors that no one has ever discovered.

- **True model** f: The perfect, unknown recipe that determines how inputs (ingredients) relate to outputs (taste)
- **Statistical model** $\hat{f}$: Our attempt to recreate that perfect recipe using the ingredients (data) we have

We'll never find the perfect recipe, but we try to get as close as possible!

Mathematically, we assume there exists a true relationship:

$$Y = f(X) + \epsilon$$

- $f(X)$: The systematic, predictable part (the "true" function)
- ϵ: Random noise (measurement error, missing variables, inherent randomness)

Statistical modeling is our attempt to estimate f with $\hat{f}$ using data.

3.2 Two Goals: Inference vs. Prediction

Table 2: *Inference vs. Prediction*

	Inference	**Prediction**
Goal	Understand the *relationship* between X and y	Get accurate *values* for y
Key Question	"How does TV budget *affect* sales?"	"What sales should we *expect* with \$150k TV budget?"
Model Type	Simple, interpretable (e.g., linear regression)	Complex, accurate (e.g., neural networks)
Is $\hat{f}$ interpretable?	**Yes**—we need to understand it	**No**—black box is fine
Analogy	Detective (understanding the crime)	Archer (hitting the target)

This lecture focuses on prediction. We'll cover inference with linear regression later.

4 The Simplest Model: The Mean

Before learning kNN, let's establish the **simplest possible model**—predicting the average:

$$\hat{y} = \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

This model ignores X entirely. No matter what the TV budget is, it always predicts the same value: the average sales (e.g., 12.5).

Key Information

Why Start with the Dumbest Model?
The mean model serves as our **baseline**. Any useful model should beat this baseline!
If your fancy machine learning model can't outperform "just guess the average," something is wrong.
We'll use this baseline to calculate R^2 later.

5 k-Nearest Neighbors (kNN) Algorithm

5.1 The Core Idea

Analogy:

The Doctor's Diagnosis A patient comes in with a stomach ache. The doctor thinks:
"This week, 10 other patients had stomach aches. They all ate from that food truck and got food poisoning. This patient probably has food poisoning too!"
The doctor **found similar past cases** (neighbors) and used their outcomes to make a prediction for the current case.

Definition:

k-Nearest Neighbors (kNN) **kNN** is a non-parametric algorithm that:

1. Finds the k training examples most similar to the query point
2. Averages their y values to make a prediction

Key insight: "Tell me who your neighbors are, and I'll tell you who you are."

5.2 kNN Step by Step (1D Example)

Given: Training data $\{(x_i, y_i)\}$ and a query point x_q

1. **Calculate distances**: Compute $D(x_q, x_i) = |x_q - x_i|$ for all training points
2. **Find k neighbors**: Select the k points with smallest distances
3. **Average**: Compute the prediction as the mean of neighbors' y values:

$$\hat{y}_q = \frac{1}{k} \sum_{i \in \text{Neighbors}_k} y_i$$

Example:

kNN with k=1 **Query**: What's the predicted sales when TV budget is \$150k?
Step 1: Calculate distances from \$150k to all training points
Step 2: Find the closest point (say, \$148k with sales = 18.2)
Step 3: Prediction: $\hat{y} = 18.2$ (just copy the nearest neighbor's value)

5.3 Effect of k on Model Complexity

The choice of k dramatically affects the model:

Warning

The Goldilocks Problem

- **k too small**: Model is too complex, memorizes noise (overfitting)
- **k too large**: Model is too simple, misses patterns (underfitting)

Table 3: *How k Affects kNN*

k Value	Behavior	Problem
k = 1 (small)	Copies nearest neighbor exactly. Very jagged, step-like predictions.	**Overfitting**: Too sensitive to noise
k = 10 (medium)	Averages 10 neighbors. Smoother curve that follows the trend.	Usually a good balance
k = n (large)	Averages ALL data points. Returns the global mean for any query.	**Underfitting**: Ignores local patterns

- **k just right**: Captures the true relationship without the noise

Finding the "just right" k requires model evaluation (next section).

5.4 k is a Hyperparameter

Definition:

Hyperparameter A **hyperparameter** is a value that:

- Is NOT learned from data
- Must be set by the human BEFORE training
- Controls the model's complexity or behavior

In kNN, k is the hyperparameter. We must choose it—the algorithm doesn't learn it.

Question: How do we find the best k? We need to evaluate different models!

6 Model Evaluation

6.1 What Does "Best" Mean?

Before comparing models, we must define "best." For prediction problems:

Best = Lowest prediction error

But how do we measure error?

6.2 Train, Validation, and Test Splits

Important:

The Golden Rule of Model Evaluation You **cannot** evaluate a model on the same data you trained it on!
Why? The model has "seen" that data—it could just memorize the answers.
We need to test on **unseen data** to measure how well the model **generalizes**.

Analogy:

The Exam Analogy

- **Training Set**: Practice problems with answer key. You study from these.
- **Validation Set**: Practice exam. You test yourself to see which study strategy works best.
- **Test Set**: The real final exam. You take it **once** to see your true ability.

If you memorize the practice exam answers instead of learning the material, you'll fail the real exam!

Table 4: *Data Split Purposes*

Set	Purpose	When Used
Training	Fit the model	During model training (kNN stores these points)
Validation	Choose hyperparameters	To compare $k = 1$ vs $k = 10$ vs $k = 70$
Test	Final evaluation	**ONLY ONCE** at the very end

Warning

Data Contamination
"There's a special place in hell for people who use the test set to choose hyperparameters."
—Professor Protopapas
If you peek at the test set while tuning, your final evaluation is **invalid**. The test set must remain **untouched** until the very end.

6.3 Mean Squared Error (MSE)

Definition:

Mean Squared Error **MSE** measures the average squared difference between predictions and actual values:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- y_i: Actual value
- $\hat{y}_i$: Predicted value
- $(y_i - \hat{y}_i)$: Residual (error for one point)

Why square the errors?

- Residuals can be positive or negative
- Without squaring, they might cancel out (e.g., +5 and -5 sum to 0)
- Squaring ensures all errors are positive and penalizes large errors more

Key Information

Why MSE (and not Mean Absolute Error)?
Short answer: MSE has nice mathematical properties (differentiable everywhere).
Deeper answer (covered in Lecture 7): If we assume the noise ϵ follows a Gaussian distribution (which the Central Limit Theorem suggests is often true), then minimizing MSE is mathematically optimal.

6.4 Choosing k Using Validation MSE

1. Split data into train and validation sets
2. For each candidate k (e.g., 1, 3, 5, 10, 20, 50):
   (a) Train kNN on training set
   (b) Compute predictions on validation set
   (c) Calculate validation MSE
3. Choose the k with the **lowest validation MSE**

6.5 R-squared (R^2): Is the Best Model Good Enough?

Analogy:

The Basketball Analogy Suppose Professor Protopapas claims to be the "best basketball player on the teaching team."
Would you sign him to the NBA?
No! Being the best among a small group doesn't mean you're actually good.
Similarly, having the best MSE among your models doesn't mean your model is actually useful.

Definition:

R-squared (R^2) R^2 measures how much better your model is compared to the baseline (mean) model:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{\text{MSE}_{\text{model}}}{\text{MSE}_{\text{baseline}}}$$

Interpretation:

- $R^2 = 1$: Perfect predictions ($\hat{y}_i = y_i$ for all i)
- $R^2 = 0$: Model is no better than predicting the mean
- $R^2 < 0$: Model is **worse** than the mean (something is wrong!)

Warning

R^2 Can Be Negative!
Despite the "squared" name, R^2 is NOT the square of anything—it's just a name.
If your model performs worse than the baseline (e.g., due to a bug or overfitting on the wrong data), R^2 will be negative.

7 High-Dimensional kNN

7.1 Multiple Predictors

With more than one predictor, we use **Euclidean distance**:

$$D(\mathbf{x}_q, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{p} (x_{q,j} - x_{i,j})^2}$$

This is just the Pythagorean theorem extended to p dimensions.

7.2 The Curse of Dimensionality

Warning

The Curse of Dimensionality

As the number of dimensions (p) increases:

- Data becomes **sparse**—points spread out in the high-dimensional space
- All points become roughly **equidistant** from each other
- The concept of "nearest neighbor" becomes meaningless

Consequence: kNN struggles in high dimensions unless you have massive amounts of data.

7.3 Feature Scaling

Important:

Scale Your Features! If TV budget is in thousands (\$0–\$300) but Newspaper budget is in dollars (\$0–\$300,000), the Newspaper dimension will dominate the distance calculation.

Solution: Standardize all features to have similar scales (covered in sections).

8 Key Takeaways

Key Summary

Summary of Lecture 04

Statistical Modeling Basics

- Response variable (y): What we predict
- Predictor variables (X): What we use to predict
- Goal: Find $\hat{f}$ that approximates the true relationship f

k-Nearest Neighbors

- Non-parametric algorithm: no assumed form for f
- Predicts by averaging the k closest training examples
- k is a hyperparameter: small k = complex/overfit, large k = simple/underfit

Model Evaluation

- Split data into **Train** (fit model), **Validation** (choose hyperparameters), **Test** (final evaluation)
- **MSE**: Average squared error—lower is better
- R^2: How much better than the baseline—higher is better (max 1)

Practical Considerations

- Never use test set for model selection
- In high dimensions, kNN struggles (curse of dimensionality)
- Always scale features when using distance-based methods

8.1 Learning Objectives Checklist

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

- ☐ Define response and predictor variables
- ☐ Represent data using design matrix X and response vector y
- ☐ Explain the difference between inference and prediction
- ☐ Describe kNN as a non-parametric algorithm
- ☐ Implement kNN in 1D: find neighbors, compute distances, average values
- ☐ Extend kNN to multiple dimensions using Euclidean distance
- ☐ Calculate MSE and R^2
- ☐ Explain the purpose of train/validation/test splits
- ☐ Recognize the curse of dimensionality and importance of feature scaling