🦾 Validation of Supervised Machine Learning

Computational Communication Science II

May 12, 2025

📚 Agenda

- Recap & practice
- Validation concepts
- Validation metrics

Validation in Text-as-Data 📑

- **Why validate?**
  Ensure classifiers are **correct and generalizable**.

- **Internal vs. External**
  - *Internal*: coherence metrics, face-validity checks
  - *External*: compare to human labels, existing benchmarks

- **Key Recommendations**
  1. **Combine multiple methods** (e.g. coherence + human coding)
  2. **Use train/test splits & cross-validation**
  3. **Report all steps** (metrics, thresholds, failures)
  4. **Preregister** your validation plan
  5. **Perform sensitivity analyses** on choices (e.g. hyperparameters, preprocessing)

Naive Bayes Algorithm

- A type of supervised ML that makes predictions using probability
- Probability (how likely something is to happen) is at the heart of Naive Bayes
- Example: If it’s cloudy, there is a 70% chance of rain
- Analogy: Like guessing if a fruit is an apple or orange based on its color and shape
- Useful for text classification

How Does Naive Bayes Work?

Imagine we are predicting if an email is spam or not spam based on words like "free" or "friend."

- Step 1: Learn from examples:
  - Look at emails we already know are spam or not spam.
  - Count how often words like "free" or "friend" appear in spam vs. not spam.
- Step 2: For a new email, check its words.
- Step 3: Calculate which is more likely: spam or not spam.

The "Naive" Part

- Naive Bayes assumes all clues (like words in an email) are independent.
- Example: It assumes "free" and "win" don't affect each other's meaning.
- This is "naive" because, in reality, words can be related.
- Why it's okay: It's simple and still works well for many problems!

A Simple Example

Let's classify a fruit as Apple or Orange:

Fruit	Color	Shape	Taste
Apple	Red	Round	Sweet
Orange	Orange	Round	Tangy

New fruit: It is "Red, Round, Sweet", what is it?

Naive Bayes checks:

- How often is "Red" in Apples vs. Oranges?
- How often is "Round" in Apples vs. Oranges?
- How often is "Sweet" in Apples vs. Oranges?

Combines these probabilities to guess $\rightarrow$ Apple.

Why Use Naive Bayes?

- Simple: Easy to understand and build.
- Fast: Works quickly, even with lots of data.
- Works well: Great for text problems like spam detection or sentiment analysis.
- Good for a gentle introduction to supervised ML

🛠️ Code Demo/Along

Why split data into *train* and *test* sets?

A key challenge in SML is ensuring the model *generalizes* to new, unseen data.

What is a Train/Test split?

Training Set:

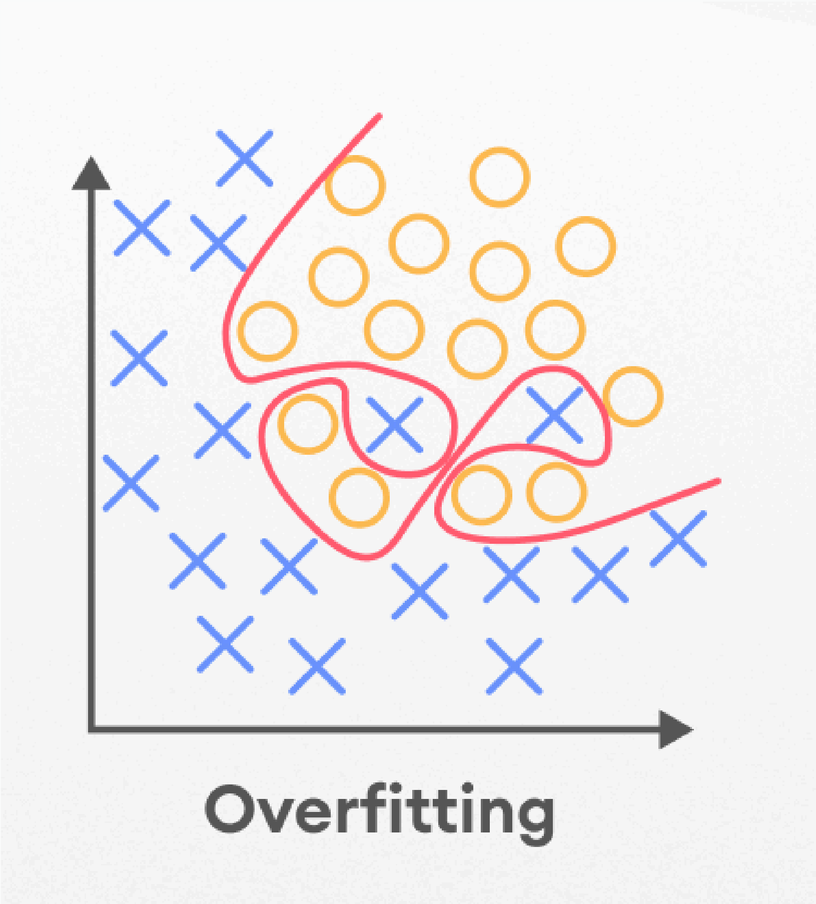
- The portion of data used to **train (or fit)** the model.
- The model "learns" from this data.

Test Set:

- A separate portion of data kept **untouched** during training.
- Used to **evaluate** the model's performance on unseen data.

Note: The Test Set acts as a proxy for unseen real-world data.

What if we use 100% data for training?



We run into *overfitting*, which means:

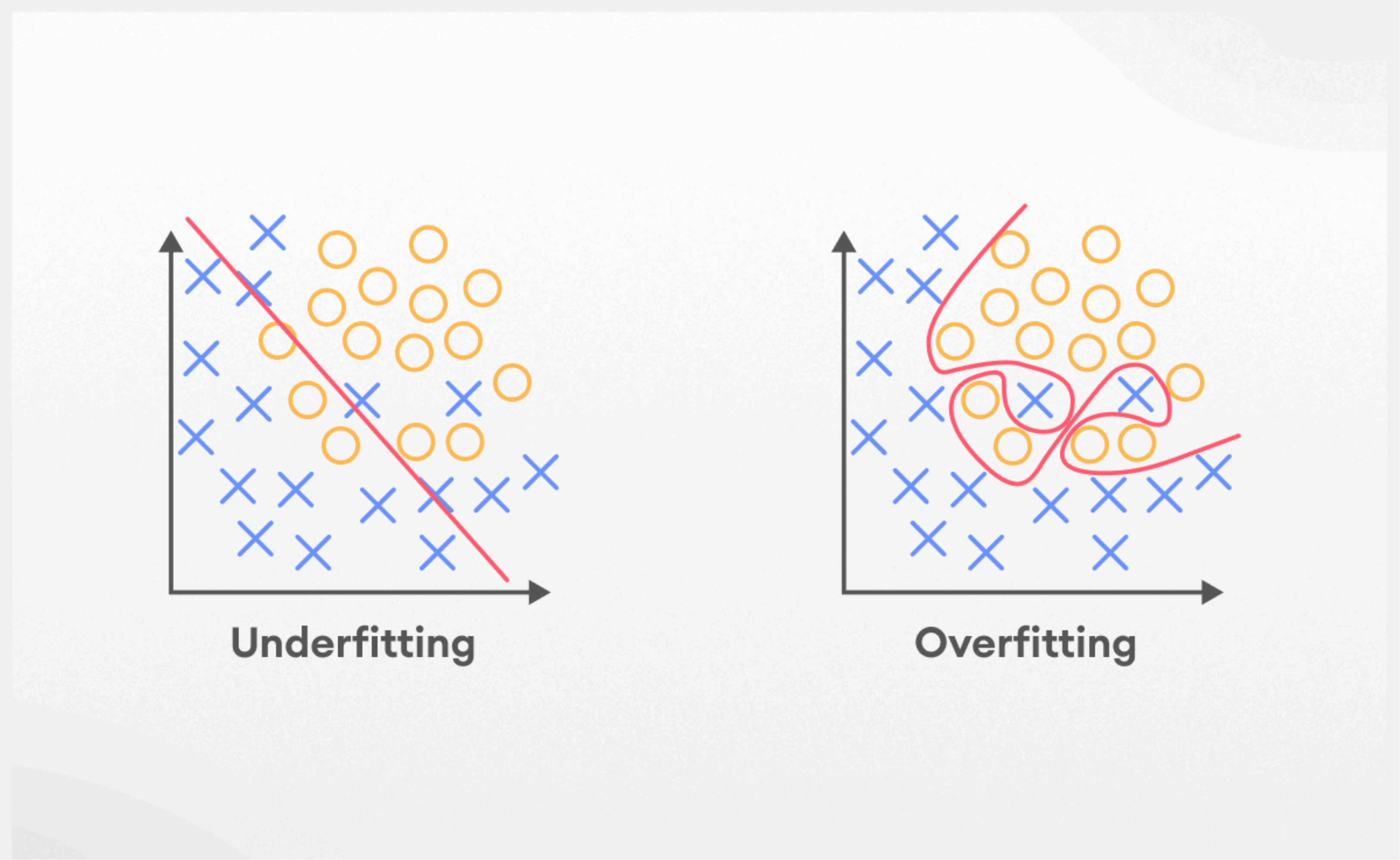
- The model performs very well on the training set 🙂
- But...it fails to predict new data accurately 🤮
- Our model will just memorize the current data, and not learn useful patterns (that help handle fresh data)

📚 A Simple Analogy

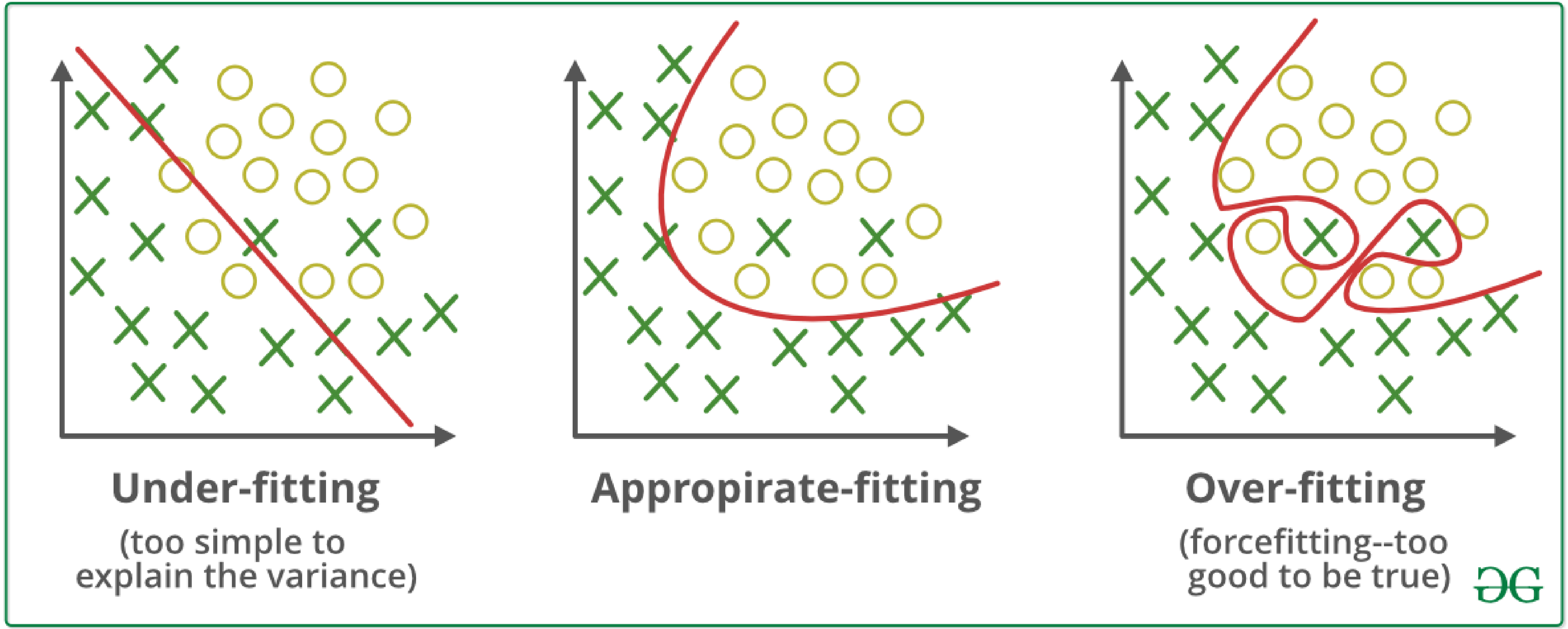
- Imagine you're studying for an exam:
  - **Training:** You practice a lot and memorize all exercise questions
  - **Testing:** You take an exam containing completely new questions
- Memorizing might help you ace the existing test, but true understanding is shown in the new exam
- The **train/test split** ensures that the model isn't just memorizing, but actually learning patterns that apply to new data

What if we use 1% data for training?

Underfitting can be a problem too 😖



⚖️ Balance is important



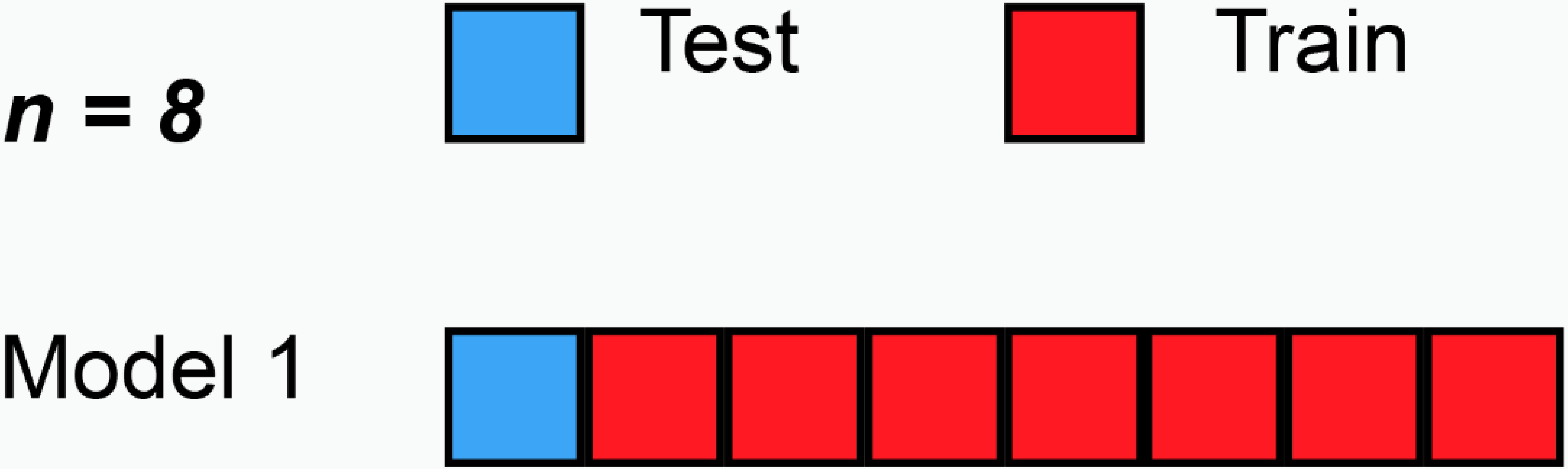
Why We Need Train/Test Splits

- **Avoiding Overfitting:** If we evaluate a model on the same data we trained it on, we might overestimate its performance.
- **Measuring Generalization:** The test set gives us a realistic estimate of how our model will perform on new data.
- **Fair Model Comparison:** When comparing different models, using the same test set ensures a fair evaluation.
- **Reliable Performance Metrics:** Metrics such as accuracy, precision, or recall computed on the test set better reflect true predictive power.

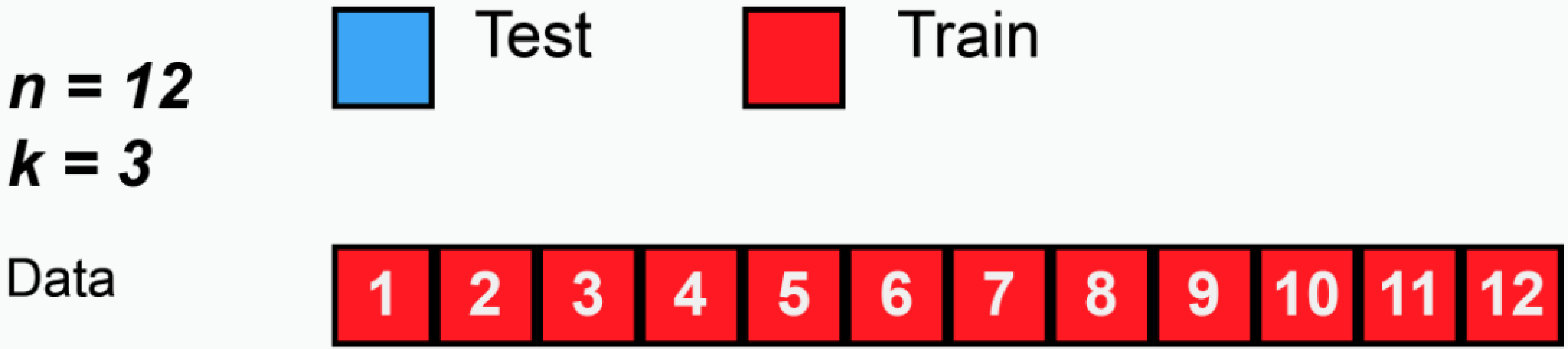
In Summary

- **Train/Test Splits** are crucial for ensuring your model generalizes well to unseen data.
- They help prevent overfitting by keeping test data separate from training.
- Always assess your model's performance on the test set to get a realistic idea of its real-world capabilities.

A Better Way $\rightarrow$ Cross Validation



An Even Better Way → k-fold Cross Validation



⏰

Confusion Matrices

A Confusion Matrix

	Actual: Dog	Actual: Not Dog
Predicted: Dog		
Predicted: Not Dog		

A Confusion Matrix

	Actual: Dog	Actual: Not Dog
Predicted: Dog	?	?
Predicted: Not Dog	?	?

A Confusion Matrix

	Actual: Dog	Actual: Not Dog
Predicted: Dog	True Positive (TP)	False Positive (FP)
Predicted: Not Dog	False Negative (FN)	True Negative (TN)

A Confusion Matrix

	Actual: Dog	Actual: Not Dog
Predicted: Dog	True Positive (TP)	False Positive (FP)
Predicted: Not Dog	False Negative (FN)	True Negative (TN)

True Positive (TP): It is the total counts having both predicted and actual values are Dog.

True Negative (TN): It is the total counts having both predicted and actual values are Not Dog.

False Positive (FP): It is the total counts having prediction as Dog while actually Not Dog.

False Negative (FN): It is the total counts having prediction as Not Dog while actually, it is Dog.

Actual	Dog	Dog	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Not Dog	Not Dog
Predicted	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Dog	Dog	Not Dog	Not Dog
Result										

Fill in the last row as TP, FP, FN, or TN.

Actual	Dog	Dog	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Not Dog	Not Dog
Predicted	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Dog	Dog	Not Dog	Not Dog
Result	TP	FN	TP	TN	TP	FP	TP	TP	TN	TN

Actual	Dog	Dog	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Not Dog	Not Dog
Predicted	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Dog	Dog	Not Dog	Not Dog
Result	TP	FN	TP	TN	TP	FP	TP	TP	TN	TN

True Positive Counts =

False Positive Counts =

True Negative Counts =

False Negative Counts =

	Actual: Dog	Actual: Not Dog
Predicted: Dog	True Positive (TP = 5)	False Positive (FP = 1)
Predicted: Not Dog	False Negative (FN = 1)	True Negative (TN = 3)

Accuracy?

What proportion of predictions are correct predictions?

$$\frac{TP+TN}{TP+TN+FP+FN}$$

	Actual: Dog	Actual: Not Dog
Predicted: Dog	True Positive (TP = 5)	False Positive (FP = 1)
Predicted: Not Dog	False Negative (FN = 1)	True Negative (TN = 3)

Precision?

What proportion of the images labeled as dogs were actually dogs?

$$\frac{TP}{TP+FP}$$

	Actual: Dog	Actual: Not Dog
Predicted: Dog	True Positive (TP = 5)	False Positive (FP = 1)
Predicted: Not Dog	False Negative (FN = 1)	True Negative (TN = 3)

Recall?

What proportion of the actual dog images were correctly identified as dogs?

$\frac{TP}{TP+FN}$

	Actual: Dog	Actual: Not Dog
Predicted: Dog	True Positive (TP = 5)	False Positive (FP = 1)
Predicted: Not Dog	False Negative (FN = 1)	True Negative (TN = 3)

F1-score?

$$\frac{2 \times Precision \times Recall}{Precision + Recall}$$

Aside: Connection with hypothesis testing

Type I error: Null hypothesis is true, but we reject it

Type II error: Null hypothesis is false, but we fail to reject it

What Metric to Use When?

Accuracy: Use when classes are roughly balanced and every prediction is equally important. *Example: Predicting if a student will pass or fail an exam when pass/fail cases are similar in number.*

Precision: Use when you want to minimize false positives (i.e., you don’t want to incorrectly predict a positive). *Example: In spam detection, you don’t want to classify legitimate emails as spam (high precision).*

Recall: Use when you want to minimize false negatives (i.e., you don’t want to miss any positive cases). *Example: In disease detection, you want to catch as many real cases as possible (high recall).*

F1 Score: Use when you want a balance between precision and recall, especially on **imbalanced** datasets. *Example: Fraud detection: you care both about catching fraud (recall) and avoiding false alarms (precision).*

Next week

Week 8: Coding in an Academic Context

Lecture: Monday, May 19, 2025

Lab Session: Tuesday, May 20, 2025

Readings:

- Bender et al. (2021): On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?
- Hube et al. (2019): Understanding and Mitigating Worker Biases in the Crowdsourced Collection of Subjective Judgments

🎓 Thank You!