

Lecture 2

KNN, ML Vocabulary, and K-Means

Our first models, supervised and unsupervised

EECS 189/289, Fall 2025 @ UC Berkeley

Joseph E. Gonzalez and Narges Norouzi

Roadmap

- Data Tools Overview
- K-Nearest Neighbors (KNN)
- Vocabulary
 - Training vs Inference,
 - Parametric vs Non-Parametric,
 - Generalization
 - Train/Validation/Test,
 - Overfitting vs Underfitting
 - Hyperparameters
 - Feature Engineering
- KNN for Regression
- K-Means Clustering



7824889

Data Tools Overview



7824889

- Data Tools Overview
- K-Nearest Neighbors (KNN)
- Vocabulary
 - Training vs Inference,
 - Parametric vs Non-Parametric,
 - Generalization
 - Train/Validation/Test,
 - Overfitting vs Underfitting
 - Hyperparameters
 - Feature Engineering
- KNN for Regression
- K-Means Clustering



7824889

Look at the Data First

How much data do you have? ($N = ?$)

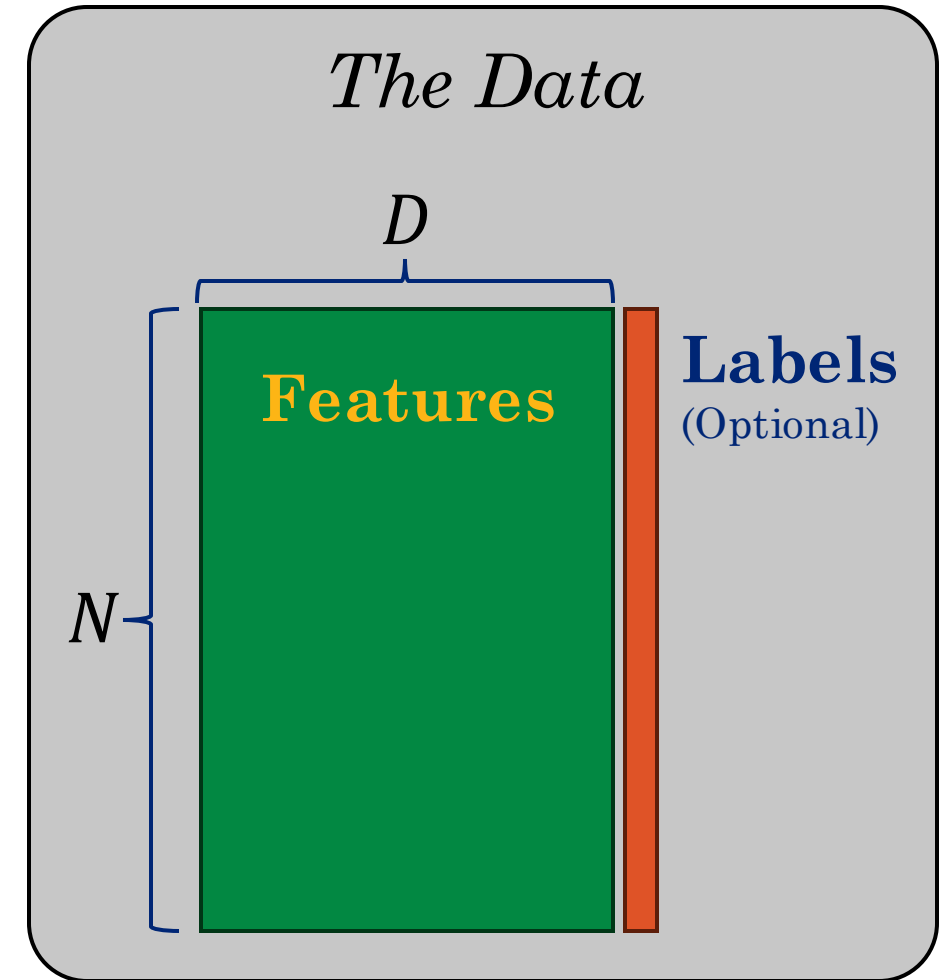
What are the **Features**?

- How many dimensions? ($D = ?$)
- What is the distribution?
- Are they all numeric?
- Are there missing values?

What are the **labels** (are there labels)?

- Are they discrete?
- What is the distribution?
- Are there missing labels or errors?

Look at the data!! (sample, read, plot...)



Today's Question



A field researcher hands you two measurements from a penguin: the length of its bill and the length of its flipper.

Can you tell them what species it is?

And before you answer: how would you know whether your answer is any good?



7824889



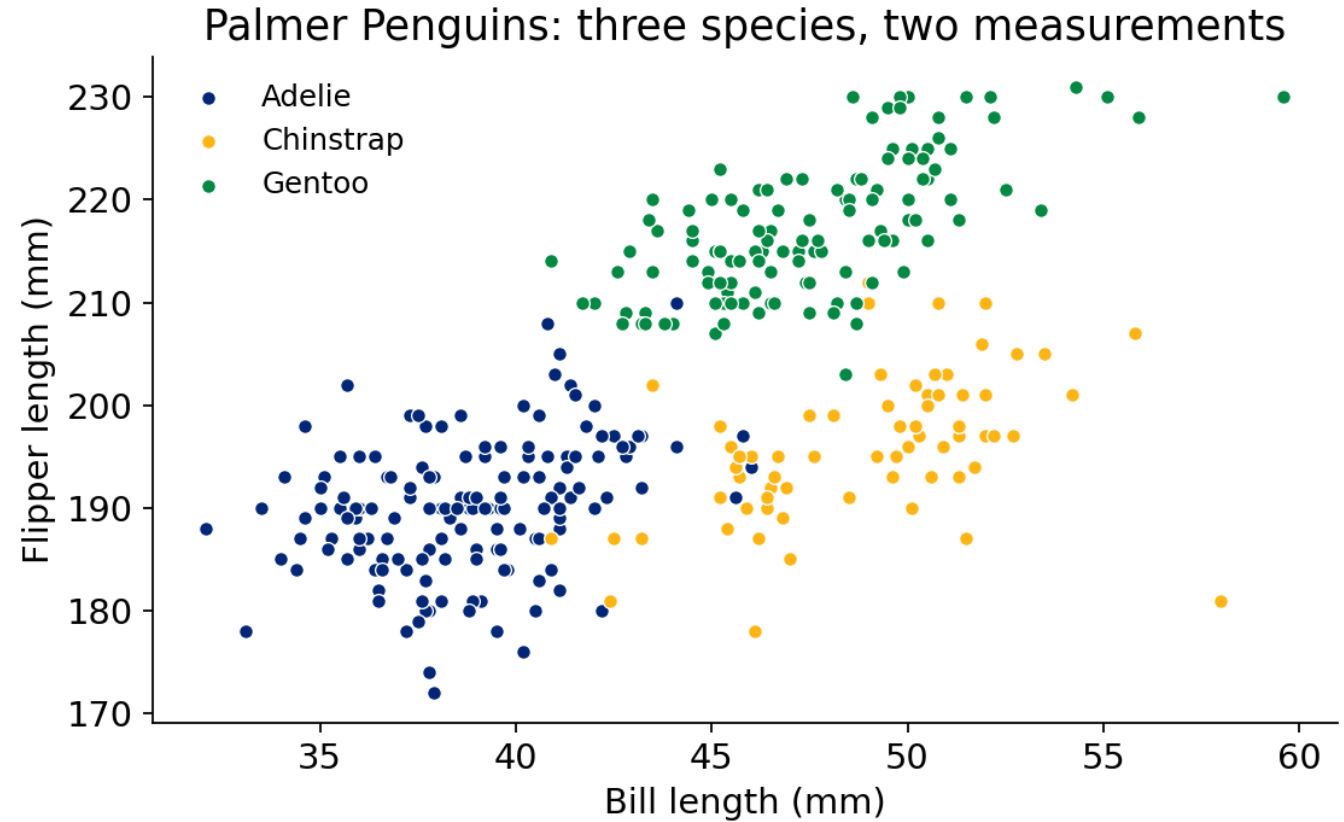
7824889

Demo

Palmer Penguins

Review:

- pandas,
- numpy, and
- plotting





7824889

The Simplest Model

We want to predict a penguin's species from two numbers.

Similar penguins should have similar species.

To label a new penguin, find the most similar penguin we have seen, and copy its label.



K-Nearest Neighbors (KNN)



7824889

- Data Tools Overview
- K-Nearest Neighbors (KNN)
- Vocabulary
 - Training vs Inference,
 - Parametric vs Non-Parametric,
 - Generalization
 - Train/Validation/Test,
 - Overfitting vs Underfitting
 - Hyperparameters
 - Feature Engineering
- KNN for Regression
- K-Means Clustering

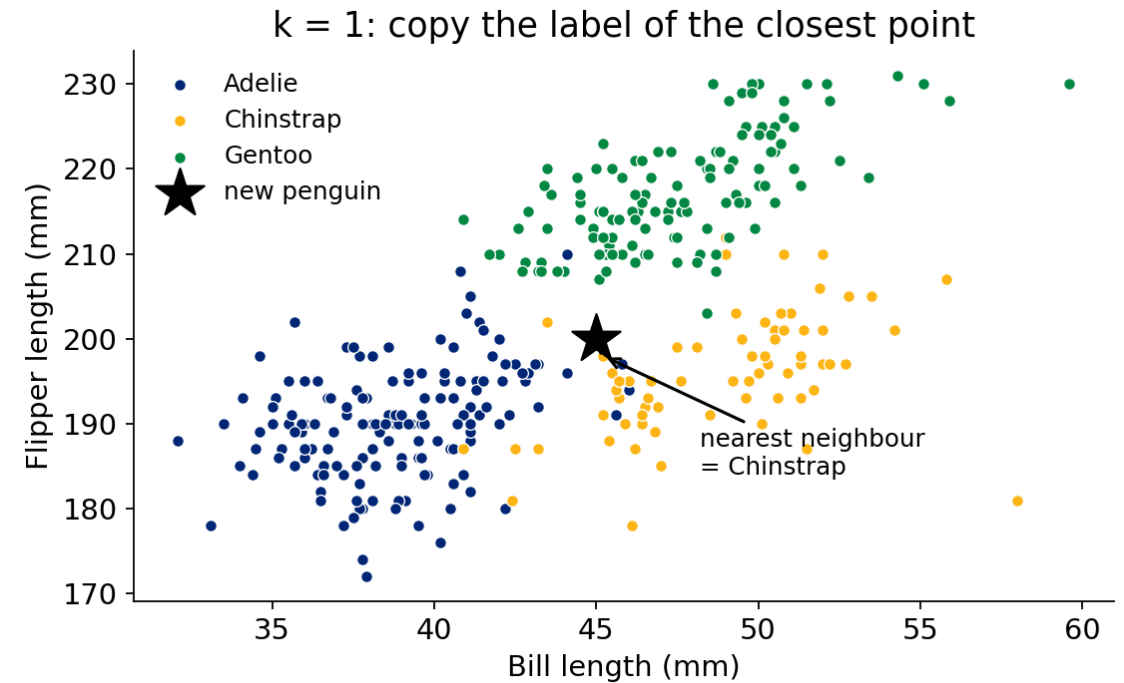


7824889

K-Nearest Neighbors, with $k = 1$

Given a new point, find the **nearest** point in the training data and copy its label.

- Similarity = Euclidean distance in feature space.



```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
```

Training vs Inference



7824889

- Data Tools Overview
- K-Nearest Neighbors (KNN)
- Vocabulary
 - Training vs Inference,
 - Parametric vs Non-Parametric,
 - Generalization
 - Train/Validation/Test,
 - Overfitting vs Underfitting
 - Hyperparameters
 - Feature Engineering
- KNN for Regression
- K-Means Clustering

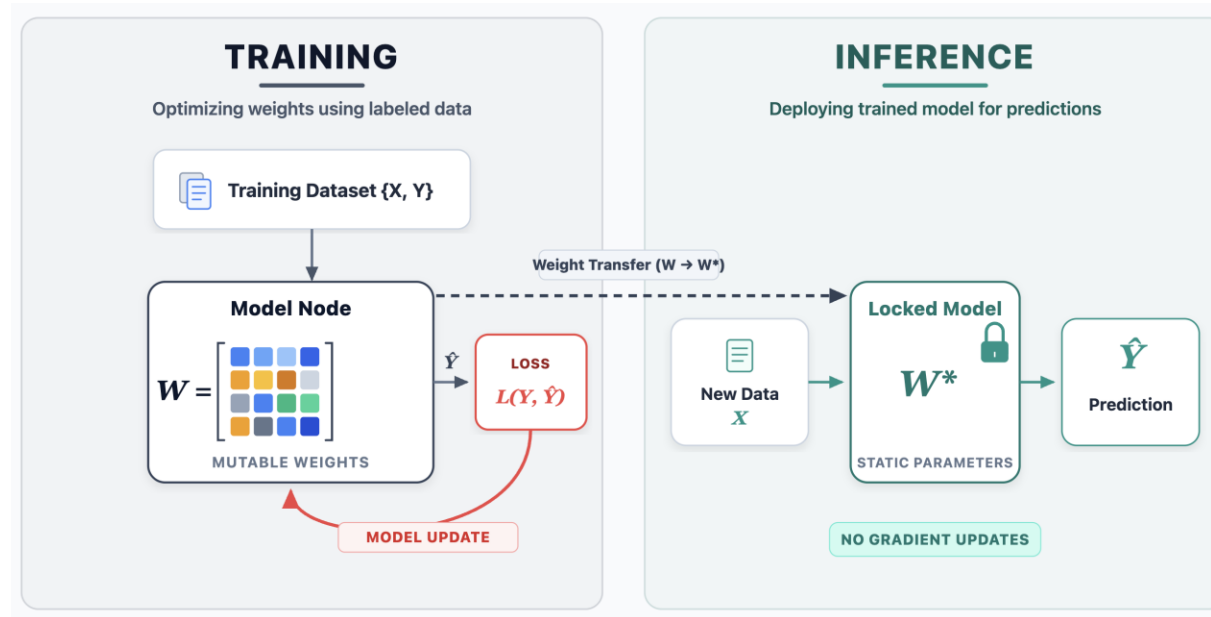


7824889

Training vs Inference

Training is the process of building the model from data.

Inference is the process of using the model to make a prediction.



- For KNN, training does almost nothing: it stores the data.
- All the work happens at inference, when we compute distances to every stored point.

Parametric vs Non-Parametric



7824889

- Data Tools Overview
- K-Nearest Neighbors (KNN)
- Vocabulary
 - Training vs Inference,
 - Parametric vs Non-Parametric,
 - Generalization
 - Train/Validation/Test,
 - Overfitting vs Underfitting
 - Hyperparameters
 - Feature Engineering
- KNN for Regression
- K-Means Clustering



7824889

Parametric vs. Non-Parametric Models

Parametric model families have a **fixed number of parameters** that does not depend on the size of the training data.

- *Example:* linear regression model $g_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{d=1}^D w_d x_d$

Non-parametric model families have “parameters” that **grow with the training data**.

- *Example:* **Nearest Neighbor Model**

$$g_{\mathcal{D}=\{(x_n, y_n)\}_{n=1}^N}(\mathbf{x}) = y_{i(\mathbf{x})}$$

$$\text{where } i(\mathbf{x}) = \arg \min_n \|\mathbf{x}_n - \mathbf{x}\|$$

The “parameters” are **all the training data**





7824889

Demo

KNN with $k = 1$

How accurate is it?

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X, y)          # 'training' = store the data

knn.score(X, y)

>>> 1.0

# ...100% accuracy. On the first try.

# What did we just do wrong?
```



Our $k=1$ model just scored 100% accuracy on the penguin data. What is the most likely explanation?



7824889

100% Accuracy

Our dumbest possible model just classified every penguin correctly.

Something is wrong. What?

- The nearest neighbour of every training point is **itself**, at distance zero.
- We asked the model questions it had already been given the answers to.

Accuracy on data the model has already seen tells us nothing about whether it has learned anything.

Generalization

- Data Tools Overview
- K-Nearest Neighbors (KNN)
- Vocabulary
 - Training vs Inference,
 - Parametric vs Non-Parametric,
 - Generalization
 - Train/Validation/Test,
 - Overfitting vs Underfitting
 - Hyperparameters
 - Feature Engineering
- KNN for Regression
- K-Means Clustering



7824889

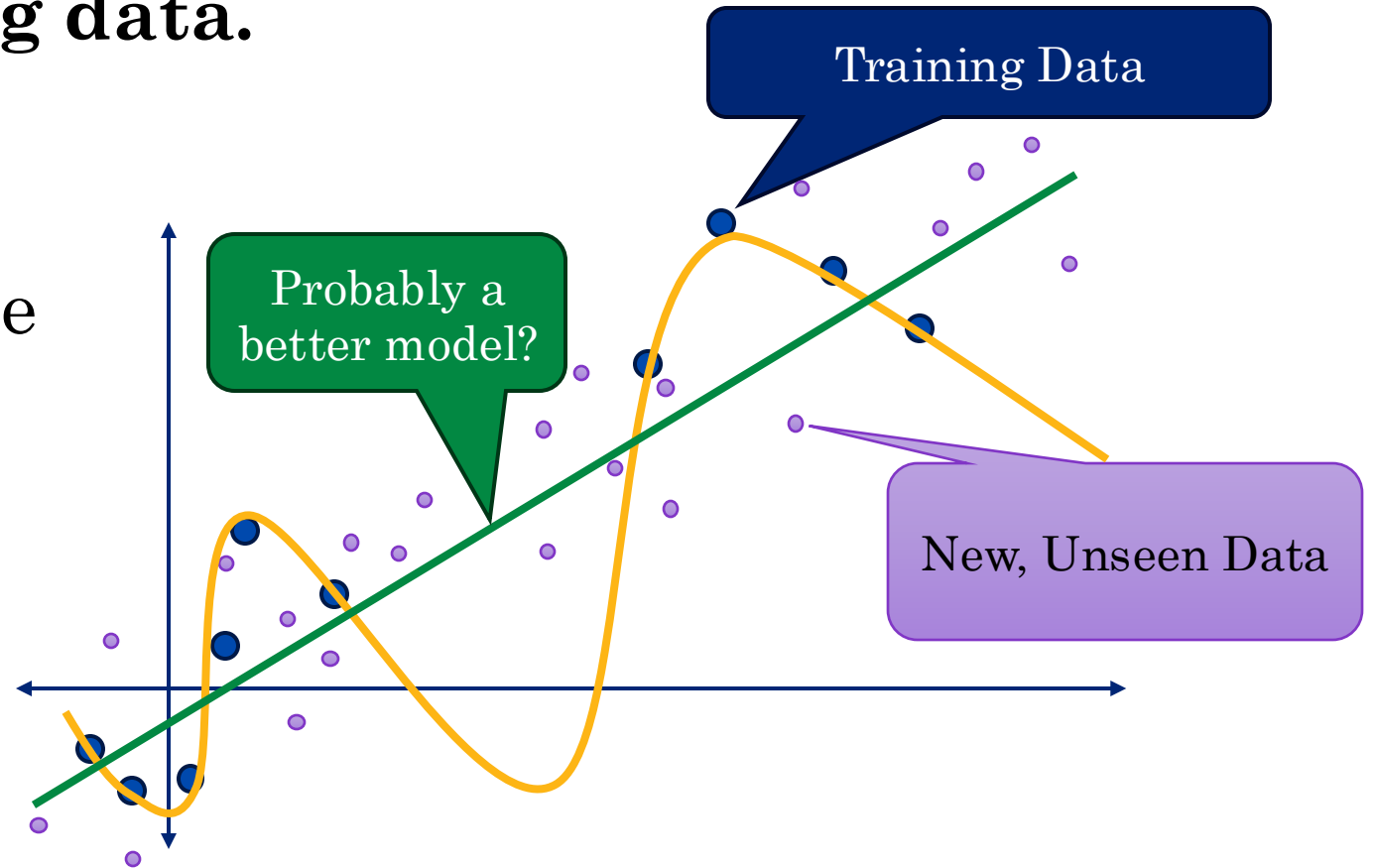
Generalization



7824889

Generalization in machine learning is the ability of a model to **perform well on new, unseen data** sampled from the **same distribution as its training data**.

To evaluate generalization, we need a method to evaluate the model's performance on **new data** not used for training.



Train/Validation/ Test



7824889

- Data Tools Overview
- K-Nearest Neighbors (KNN)
- Vocabulary
 - Training vs Inference,
 - Parametric vs Non-Parametric,
 - Generalization
 - Train/Validation/Test,
 - Overfitting vs Underfitting
 - Hyperparameters
 - Feature Engineering
- KNN for Regression
- K-Means Clustering



7824889

How will we evaluate the model?

We are about to **train** a model **using data**.

How will we know if the **model** has “**learned**” from our data?

- Could we measure how well our model fits the data?

The Exam Analogy:

*What would happen if we gave everyone access to the exam and solutions (**the data**) to study (**train**) for the exam?*

- Would everyone do well?
- Does this mean they learned the material in the class?

Why is **memorizing the data** (exam) not good?



7824889

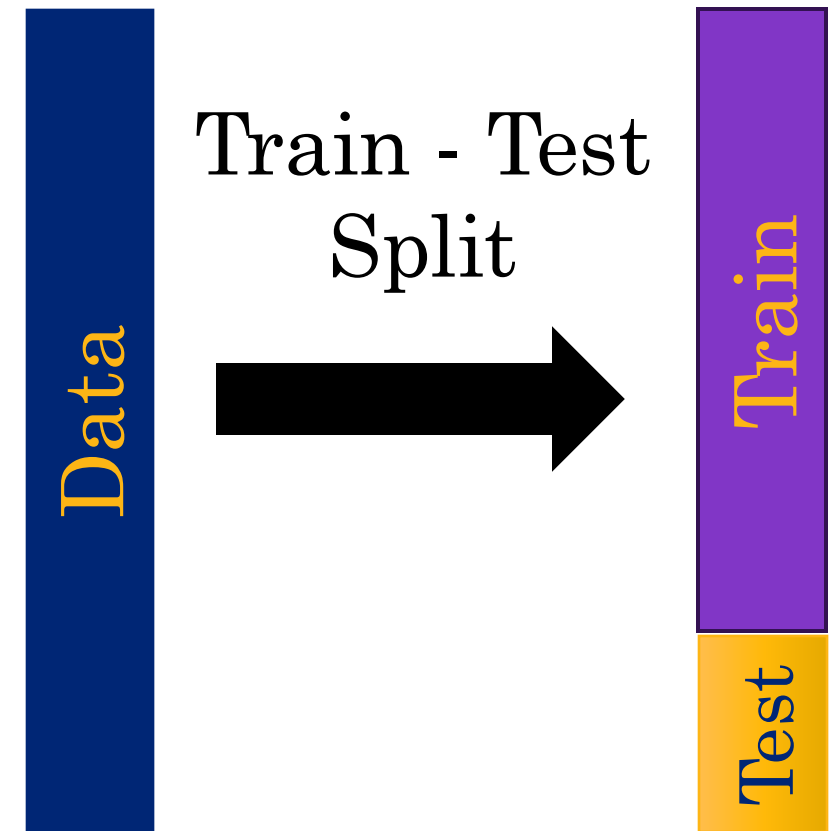
Evaluating generalization using the Train-Test Split

The **train-test split** is the standard technique we use to **evaluate generalization** in machine learning:

1. Shuffle the training data
2. Split into two parts:
 - **Larger Training Part** (~80%): used to develop and train the model.
 - **Smaller Testing Part** (~20%): used to evaluate generalization performance.

You should only use the **test dataset once** *after* developing and training the model.

If you use the test data to **tune the model**, the test data no longer measures **generalization**.





7824889

Evaluating generalization using the Train-Test Split

The **train-test split** is the standard technique we use to **evaluate generalization** in machine learning:

1. Shuffle the training data

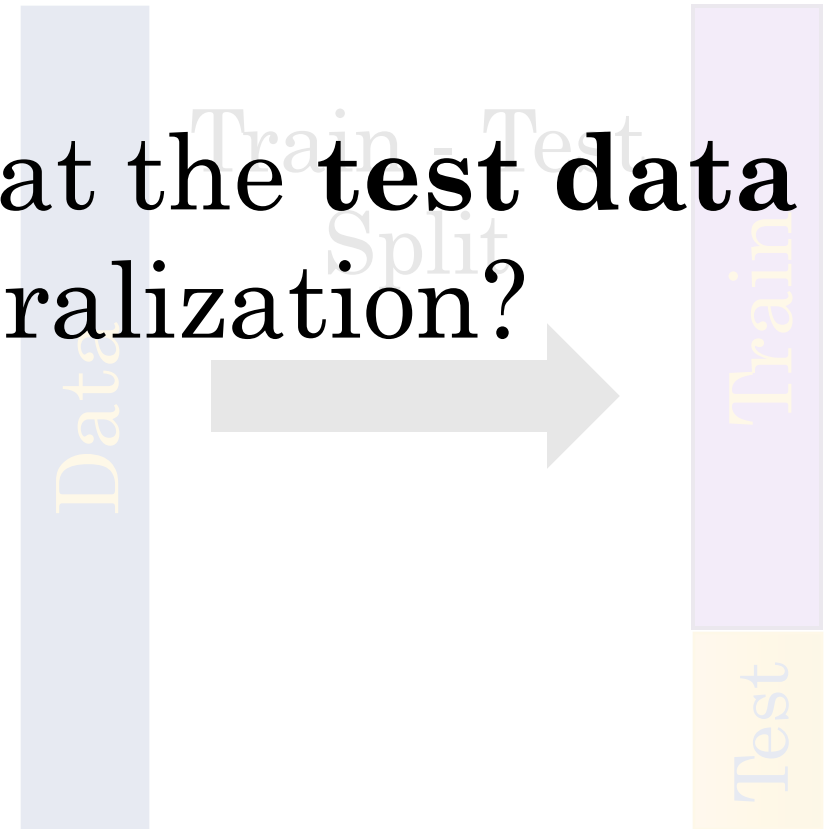
2. Split into two parts:

- **Larger Training Part (~80%)**: used to develop and train the model.
- **Smaller Testing Part (~20%)**: used to evaluate generalization performance.

But what if I *want* to peek at the test data to tune for better generalization?

You should only use the **test dataset once** *after* developing and training the model.

If you use the test data to **tune the model** the test data no longer measures **generalization**.

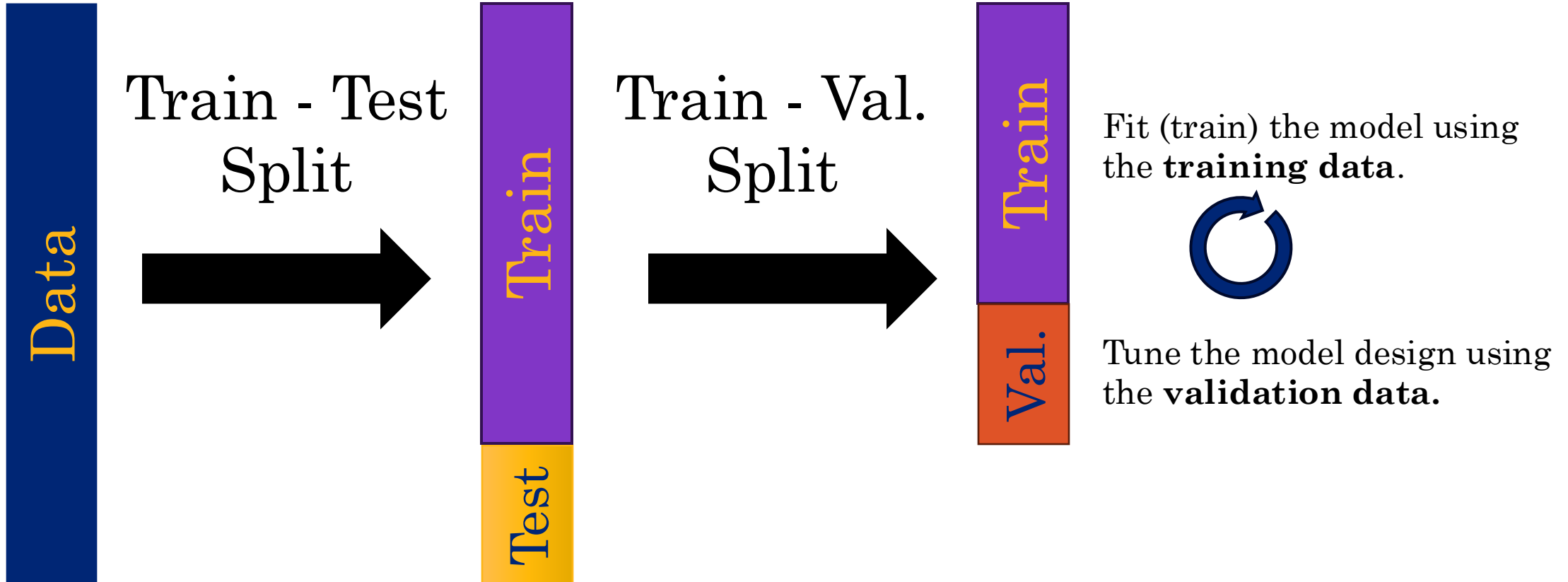


The Validation Split



7824889

The validation dataset is used to **evaluate generalization performance** during the **model development process**.





7824889

The Train-Validation-Test Split Analogy

You can think of the train, validation, and test splits as how you might study for an exam.

Train

Practice question and answer pairs that you use to study for the exam. (**Go to discussion!**)

Val.

Practice exam evaluates if your studying process is working (or if you need to study more).

Test

The exam is how we evaluate if you should pass the class.



You train a model, then try 30 different settings and pick the one with the best test accuracy. What have you actually measured?



7824889

Demo

Train/test split and the effect of k

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)

knn = KNeighborsClassifier(n_neighbors=1).fit(X_train, y_train)

train accuracy: 100.0%
test accuracy: 97.1%

# now sweep k = 1 ... 100 and watch both curves
```

Overfitting vs Underfitting



7824889

- Data Tools Overview
- K-Nearest Neighbors (KNN)
- Vocabulary
 - Training vs Inference,
 - Parametric vs Non-Parametric,
 - Generalization
 - Train/Validation/Test,
 - Overfitting vs Underfitting
 - Hyperparameters
 - Feature Engineering
- KNN for Regression
- K-Means Clustering

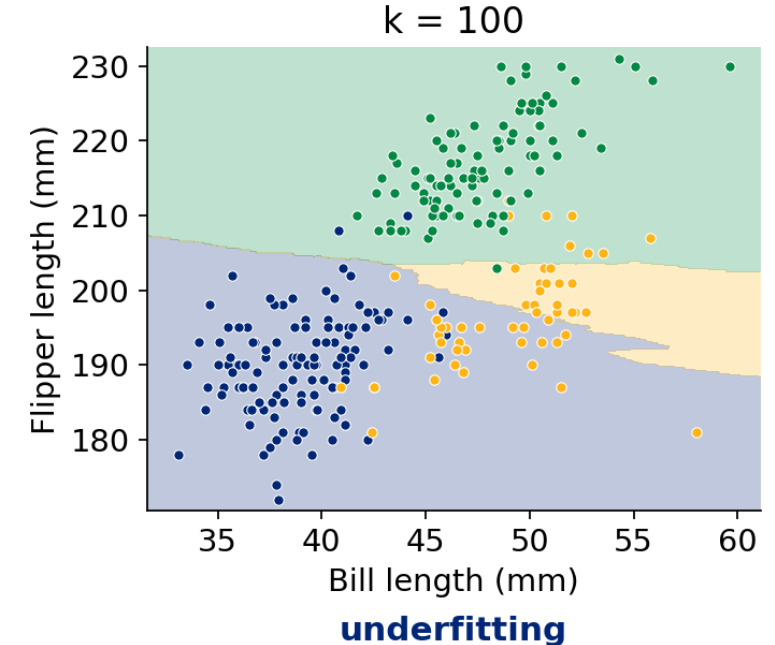
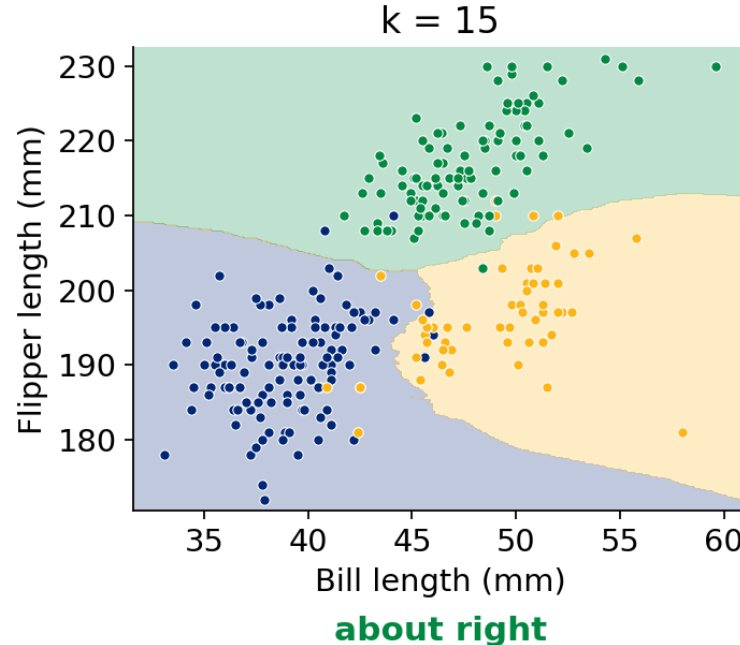
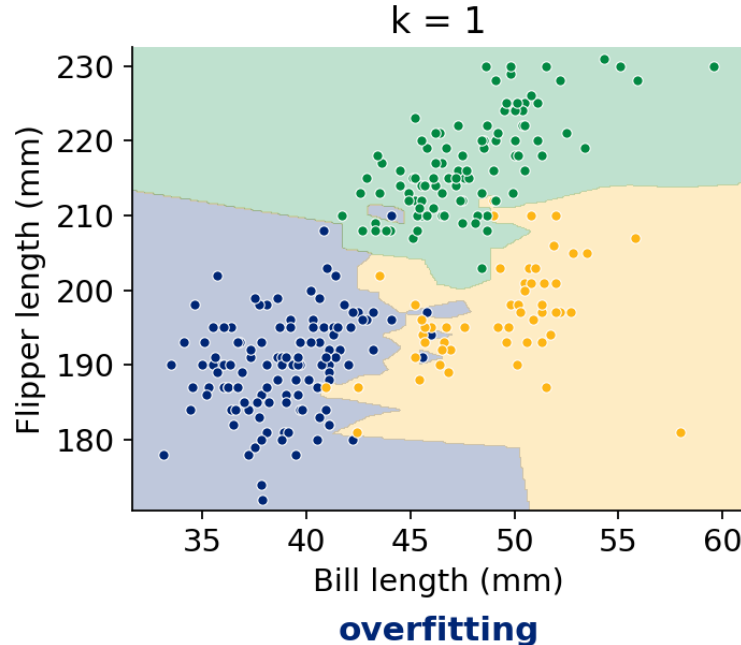


7824889

What k Controls

Let the k nearest neighbours vote, instead of just one.

- **Small k :** The boundary tries to capture every training point, including the unusual ones. This is **overfitting**.
- **Large k :** The boundary smooths until the model barely responds to the data at all. This is **underfitting**.





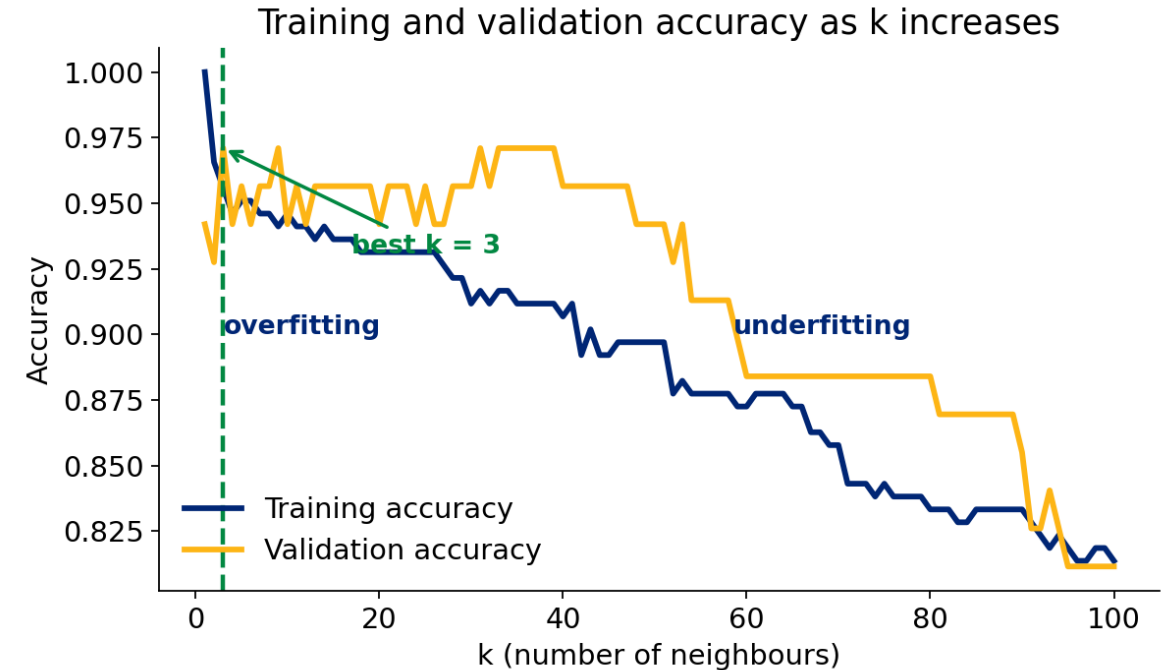
7824889

Choosing k Without Eyeballing It

Fit the model for every k and score it on the **validation** set.

- **Left side (Overfitting):** Training accuracy is perfect, validation accuracy is not.
- **Right side (Underfitting):** Both fall together.
- **The peak of the validation curve is the k we choose.**

On our data the best k is 3, and the final test accuracy is 97%.



Hyperparameters

- Data Tools Overview
- K-Nearest Neighbors (KNN)
- Vocabulary
 - Training vs Inference,
 - Parametric vs Non-Parametric,
 - Generalization
 - Train/Validation/Test,
 - Overfitting vs Underfitting
- Hyperparameters
 - Feature Engineering
- KNN for Regression
- K-Means Clustering



7824889



7824889

Hyperparameters

The **hyperparameters** are the parameters that are constant during the optimization (training) algorithm.

- Example λ in $\hat{w} = \arg \min_{w \in \Theta} \text{Error}[h_w; \mathcal{D}_{\text{training}}] + \lambda \text{Reg}[w]$

The **hyperparameters** are chosen to **improve generalization** performance as **measured on the validation dataset**.

- Hyperparameters are often selected using a **grid-search**

```
for hp1 in [0.1, 1, 10]:  
    for hp2 in [0.1, 1, 10]:  
        model = MyModel(hp1=hp1, hp2=hp2)  
        model.fit(X_train, y_train)  
        error[hp1, hp2] = error(y_val, model.predict(X_val))  
hp1_best, hp2_best = arg_min(error)
```

	0.1	1	10
0.1	0.8	0.7	0.7
1	0.4	0.2	0.3
10	0.5	0.6	0.6



7824889

Parameters vs Hyperparameters

A **parameter** is learned from the data during training.

A **hyperparameter** is chosen by us, before training.

- **k in KNN is a hyperparameter.**
- **KNN has no parameters at all, which is unusual.**

Feature Engineering



7824889

- Data Tools Overview
- K-Nearest Neighbors (KNN)
- Vocabulary
 - Training vs Inference,
 - Parametric vs Non-Parametric,
 - Generalization
 - Train/Validation/Test,
 - Overfitting vs Underfitting
 - Hyperparameters
 - Feature Engineering
- KNN for Regression
- K-Means Clustering



7824889

Example of Feature Engineering: Standardization

Feature engineering is a process of preparing your data and its features for effective modeling.

KNN is built entirely on distance, so it inherits whatever the units happen to be.

- Flipper length spans roughly 172 to 231 mm.
- Bill length spans roughly 32 to 60 mm.
- **Flipper length therefore dominates the distance, purely because its numbers are bigger.**

Standardization rescales every feature to mean 0 and standard deviation 1, so features contribute on equal terms.

KNN for Regression



7824889

- Data Tools Overview
- K-Nearest Neighbors (KNN)
- Vocabulary
 - Training vs Inference,
 - Parametric vs Non-Parametric,
 - Generalization
 - Train/Validation/Test,
 - Overfitting vs Underfitting
 - Hyperparameters
 - Feature Engineering
- KNN for Regression
- K-Means Clustering



7824889

The Other Kind of Prediction

So far we predicted a **category**: which species.
That is **classification**.

What if we want to predict a **number**, like body mass in grams?
That is **regression**.

Does our algorithm still work?

KNN Regression

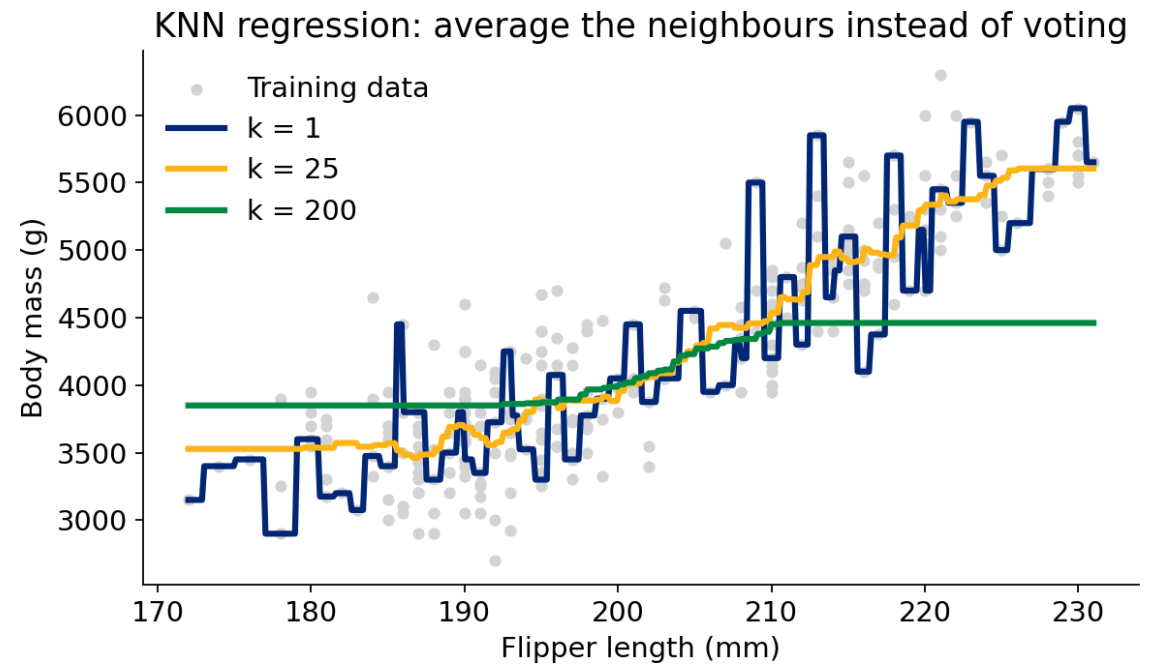


7824889

Find the k nearest neighbours, then **average** their values instead of taking a majority vote.

- $k = 1$: a jagged step function chasing every point.
- $k = 200$: nearly a flat line.

We see the same overfitting and underfitting issues.





7824889

Demo

KNN for regression Predicting body mass from flipper length

```
from sklearn.neighbors import KNeighborsRegressor

# same algorithm: average the neighbours
# instead of taking a majority vote

reg = KNeighborsRegressor(n_neighbors=25)
reg.fit(X_train, y_train)  # flipper length -> body mass

k= 1   train R2 = 0.643   test R2 = 0.457
k= 25  train R2 = 0.785   test R2 = 0.769
k=200  train R2 = 0.444   test R2 = 0.428
```

K-Means Clustering



7824889

- Data Tools Overview
- K-Nearest Neighbors (KNN)
- Vocabulary
 - Training vs Inference,
 - Parametric vs Non-Parametric,
 - Generalization
 - Train/Validation/Test,
 - Overfitting vs Underfitting
 - Hyperparameters
 - Feature Engineering
- KNN for Regression
- K-Means Clustering

What If There Were No Labels?

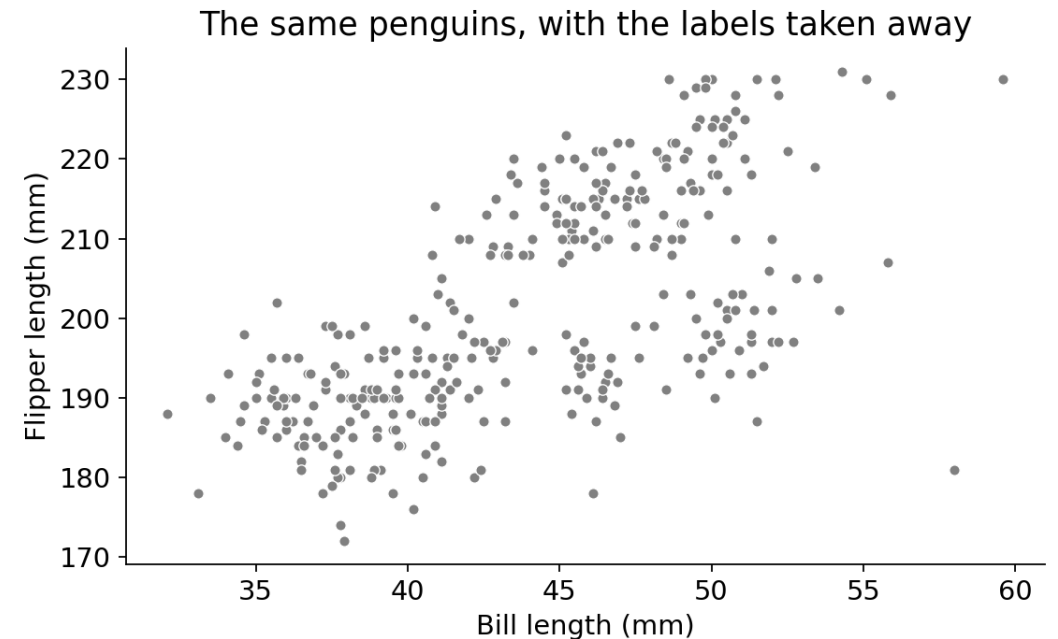


7824889

Everything so far was **supervised**: every penguin arrived with its species attached.

Now suppose the researcher hands you the same measurements with no labels at all, and asks: **are there natural groups in here?**

This is **unsupervised learning**, and specifically **clustering**.





7824889

K-Means Clustering

Input: A collection of data points $\mathcal{D} = \{x_1, \dots, x_N\}$ where $x_n \in \mathbb{R}^D$.

Output: K cluster centers $\mu_k \in \mathbb{R}^D$ and an **assignment** $z_n \in \{1, \dots, K\}$ of each data point to one of the cluster centers.

- The means μ_k are **model parameters** of the model (z_n can be computed from μ) that are **fit to the data**.
- K is a **hyperparameter** which we need to select.

Objective: Each data point should be close to its assigned center.

$$\arg \min_{\mu, z} \sum_{n=1}^N \|x_n - \mu_{z_n}\|_2^2$$

(L₂-Norm)²
Euclidean Distance Squared

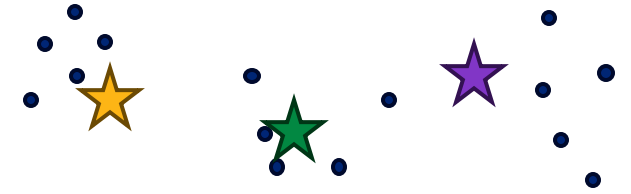
$$\|a\|_2^2 = \sum_{d=1}^D a_d^2$$



7824889

K-Means Cluster (Lloyd's) Algorithm

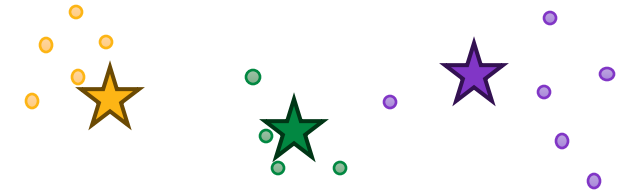
Initialization: Choose K points at random to be the initial cluster centers μ .



Iterate until convergence:

1. **Update Assignments:** Assign each point to its nearest cluster center.

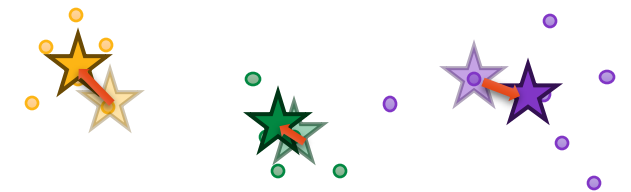
$$z = \arg \min_z \sum_{n=1}^N \|x_n - \mu_{z_n}\|_2^2$$



2. **Update Centers:** Recompute centers by averaging assigned points.

Why is this the mean of each cluster?

$$\mu = \arg \min_{\mu} \sum_{n=1}^N \|x_n - \mu_{z_n}\|_2^2$$





7824889

Convergence of K-Means

Lloyd's algorithm always converges, but only to a **local** optimum.

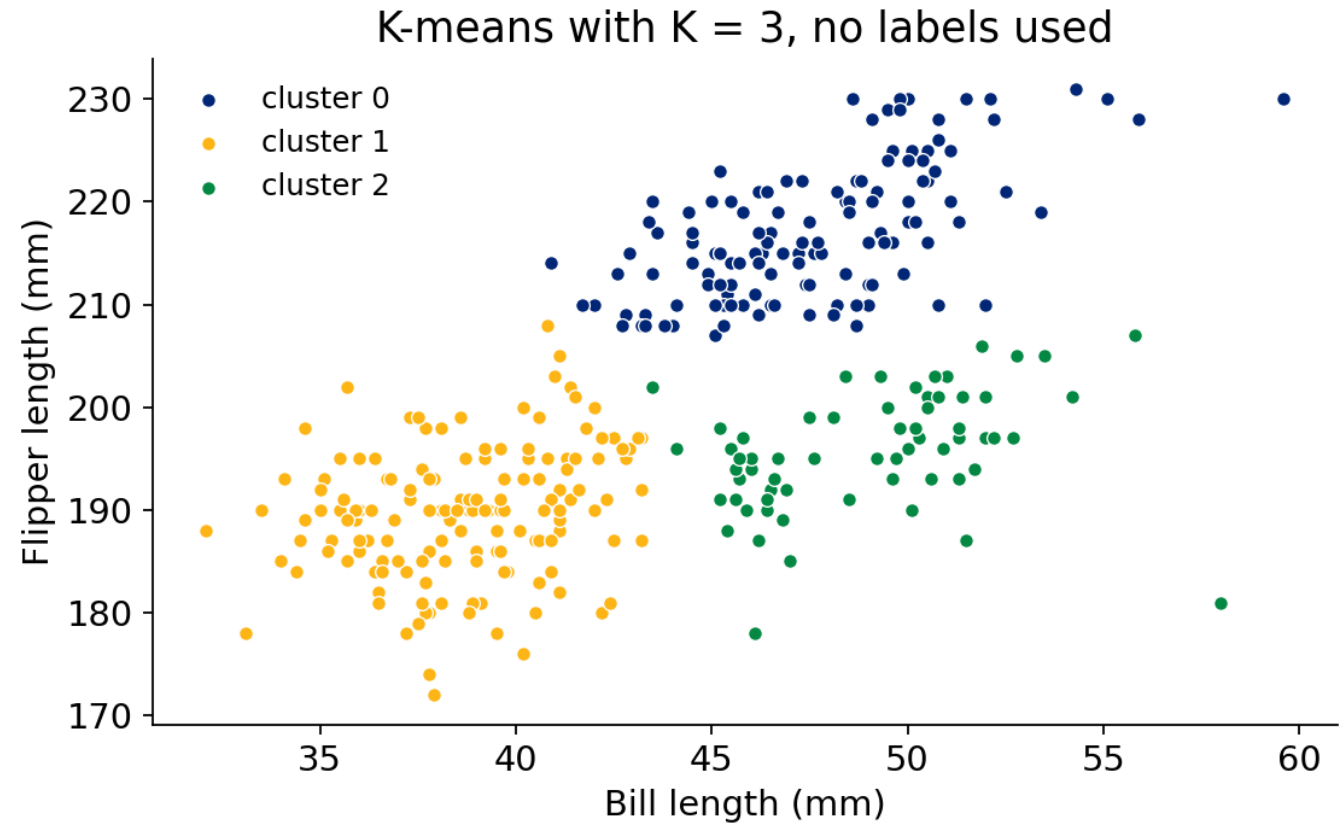
- Each step can only decrease the objective, and there are finitely many assignments.
- Different random initializations can give **different final clusterings**.
- This is why scikit-learn runs it several times and keeps the best: **n_init=10**.



7824889

Demo

K-means on the
penguins
Animated Lloyd's
algorithm





7824889

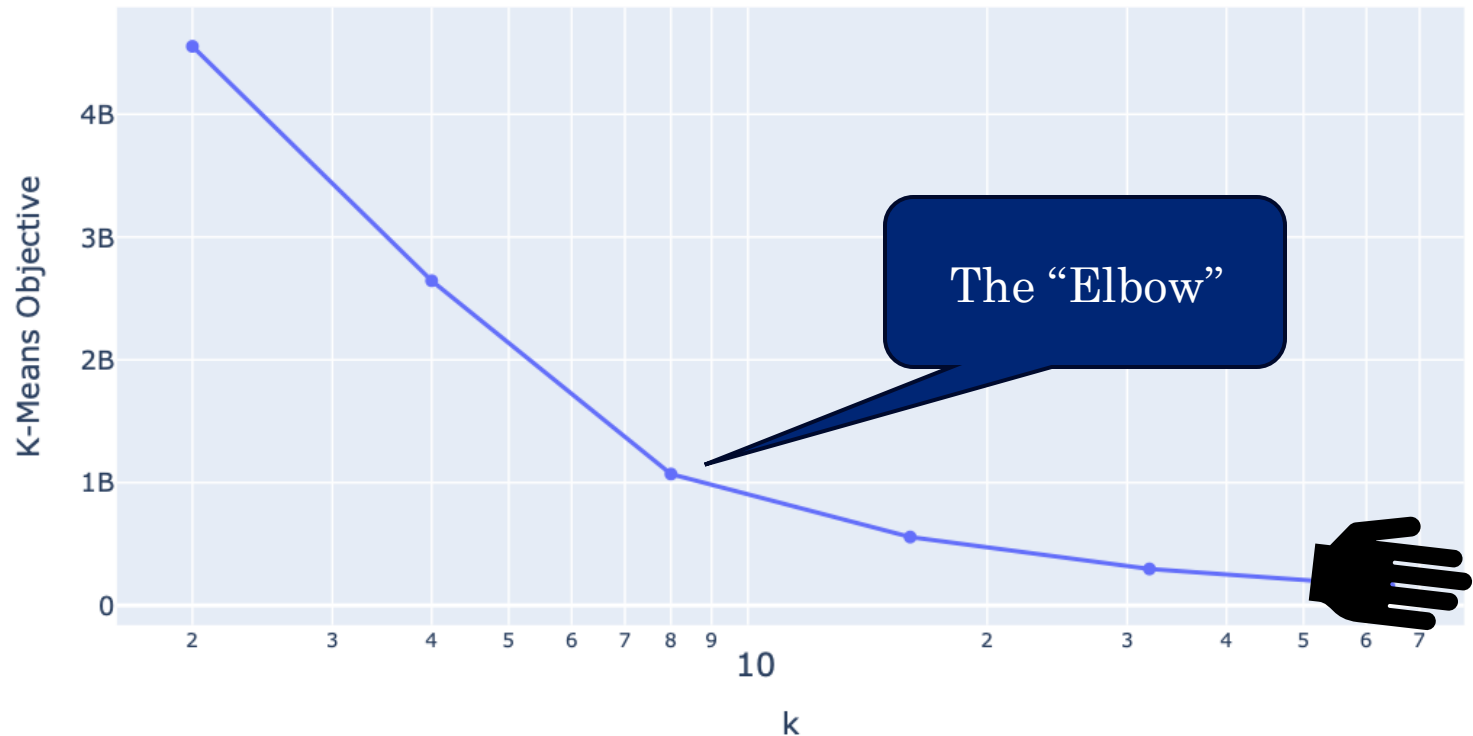
Choosing the Number of Clusters

There are several ways to select the **hyperparameter** K .

- Use **domain understanding** (e.g., three species)
- Use the “**elbow method**” to select the value of K with diminishing improvements in the objective:

$$\sum_{n=1}^N \|x_n - \mu_{z_n}\|_2^2$$

K-Means Objective vs Number of Clusters



Clusters Are Not Labels

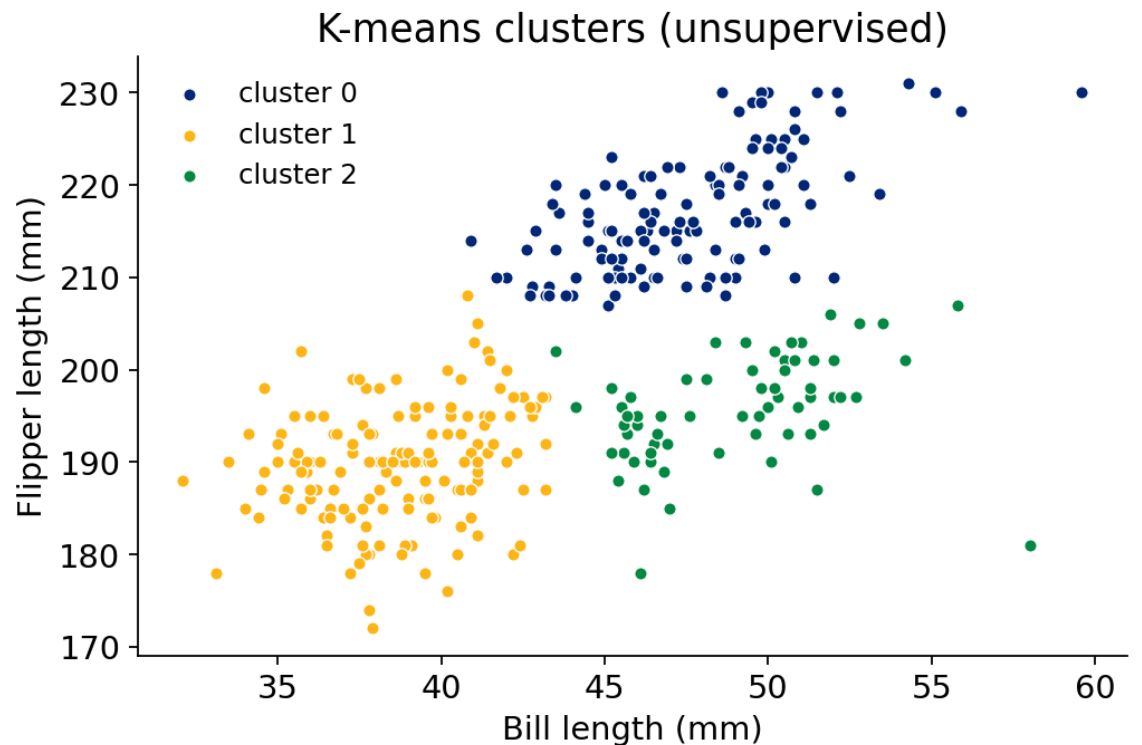
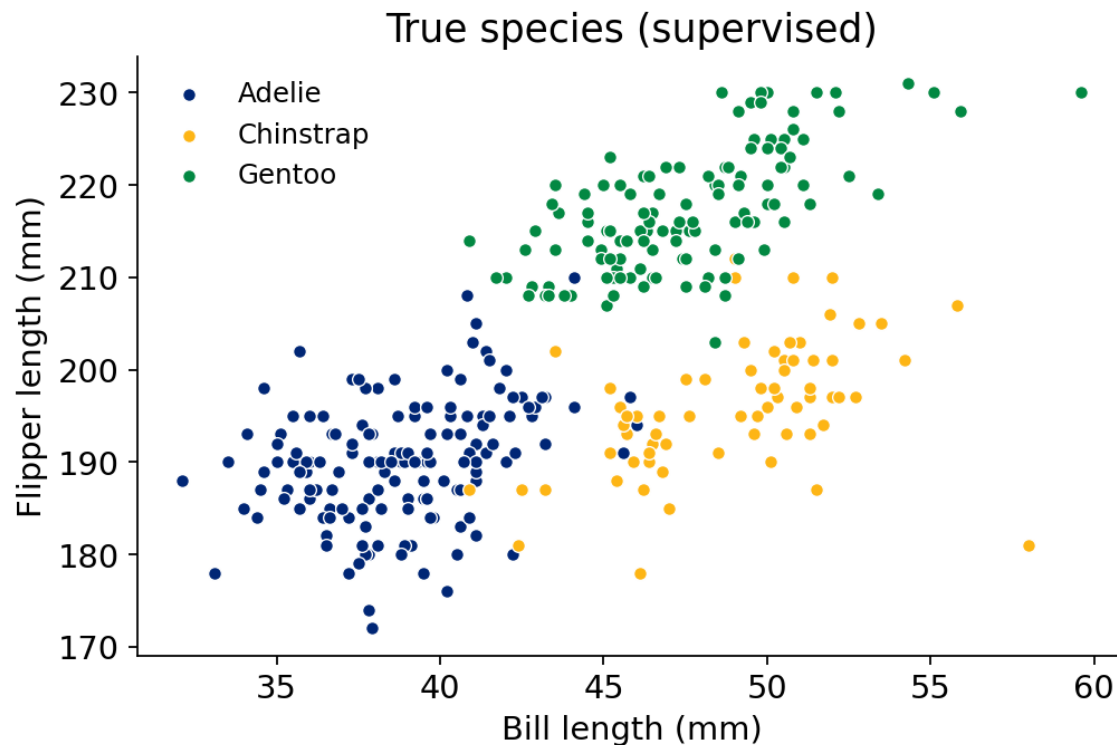


7824889

K-means rediscovered the species without ever seeing a label.

But it found penguins that sit close together in bill and flipper measurements. In this dataset those groups happen to line up with species.

Change the features and the clusters change.



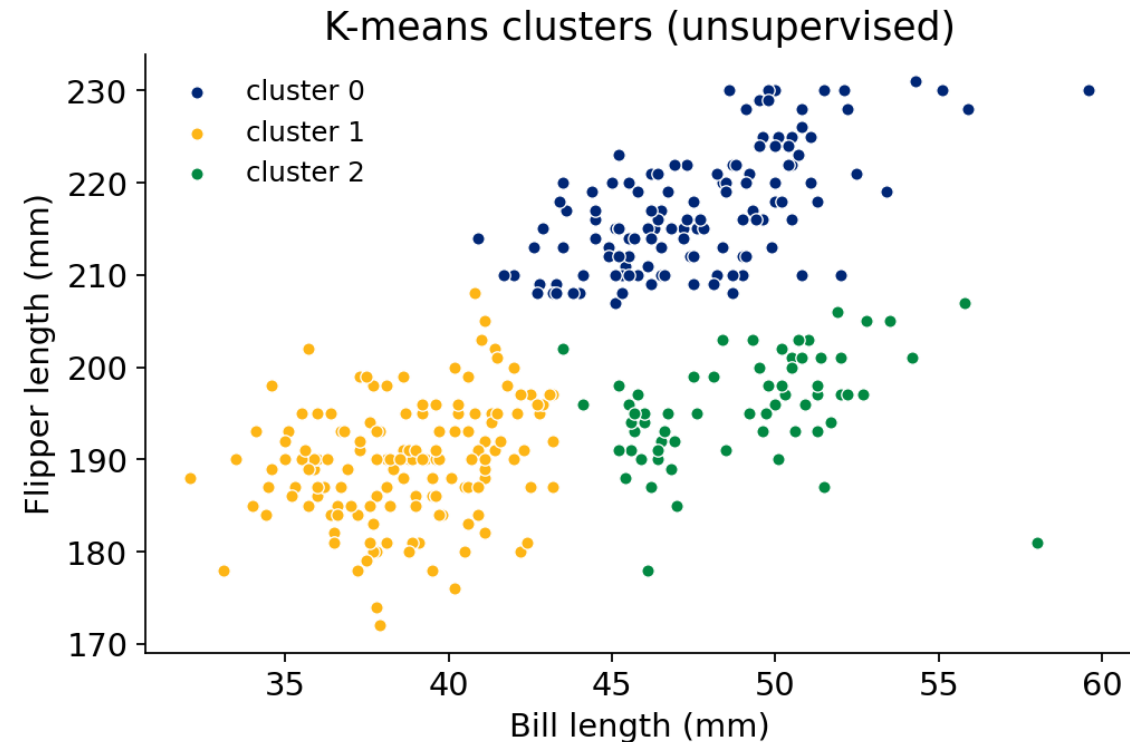
Interpreting the Clusters



7824889

We used k-means to compute a **cluster assignment** for each penguin.

- **Which species is each cluster?**
 - We do not know. We would have to ask for some labels.
- **Does each cluster represent a species?**
 - Maybe.
- **Use caution when interpreting clusters.**





K-means found three clusters that line up closely with the three penguin species. What does this tell us?

Lecture 2

KNN, ML Vocabulary, and K-Means

Credit: Joseph E. Gonzalez and Narges Norouzi

Reference Book Chapters: Chapter 1 (up to the end of 1.2), Chapter 15.1 (K-Means)