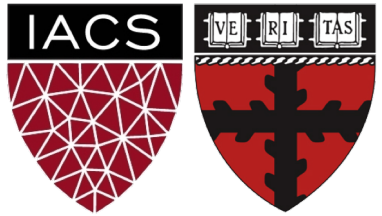
Lecture 13: Data, Models, and Code

Addressing common issues

Harvard IACS

CS109A

Pavlos Protopapas, Kevin Rader, and Chris Tanner



ANNOUNCEMENTS

- **Homework 2 grades** will be released soon
- **Homework 3** has been released. **Due Oct 7 (Wed) @ 11:59pm**
- **Standard Sections** will be:
  - today at **1:30pm-2:45pm** and
  - **Monday 8:30pm-9:45pm**
- After lecture, please update your Zoom to the latest version (5.3.1)

Learning Objectives

- Understand all concepts that are addressed in HW3, along with how they relate to one another
- Be able to soundly **prepare** machine learning experiments, in terms of data splits and formatting
- Know common strategies in debugging code
- Feel prepared tackling the remaining course content / gain confidence!

Agenda

Review of Models

Review of Data

Debugging

Agenda

Review of Models

Review of Data

Debugging

What is the most difficult concept for you right now?

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column

Your Data X

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column

Your Data X

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column

Your Data X

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column

Your Data X

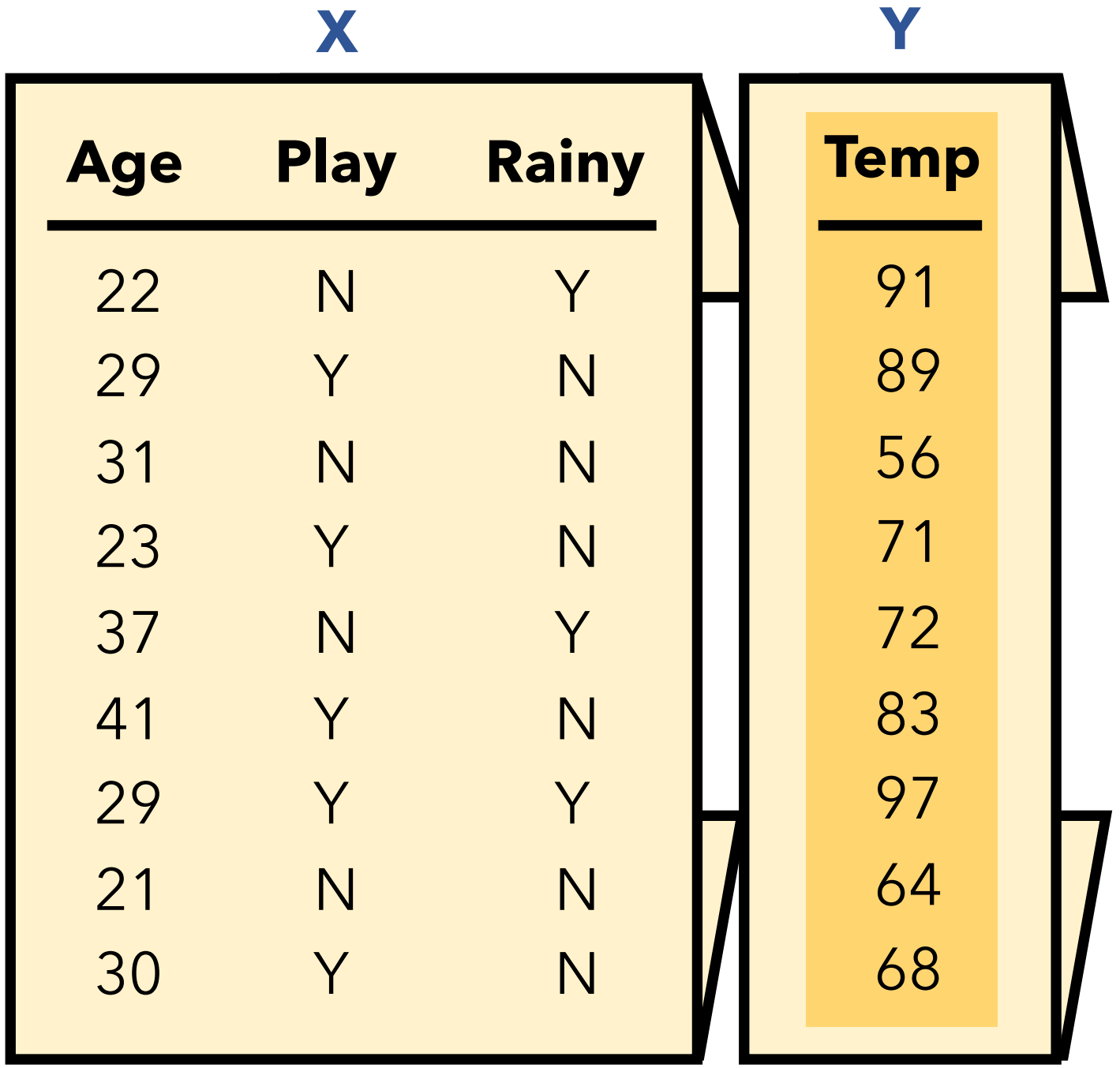
Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column (e.g. **Temp**)

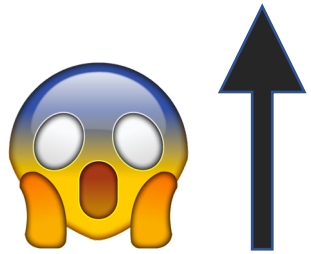
Your Data X

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column (e.g. **Temp**)
- Let's divide our data and learn how data **X** is related to data **Y**
- Assert that: $\boldsymbol{Y} = f(\boldsymbol{X}) + \varepsilon$



- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column (e.g. **Temp**)
- Let's divide our data and learn how data **X** is related to data **Y**
- Assert that: $\boldsymbol{Y} = f(\boldsymbol{X}) + \varepsilon$

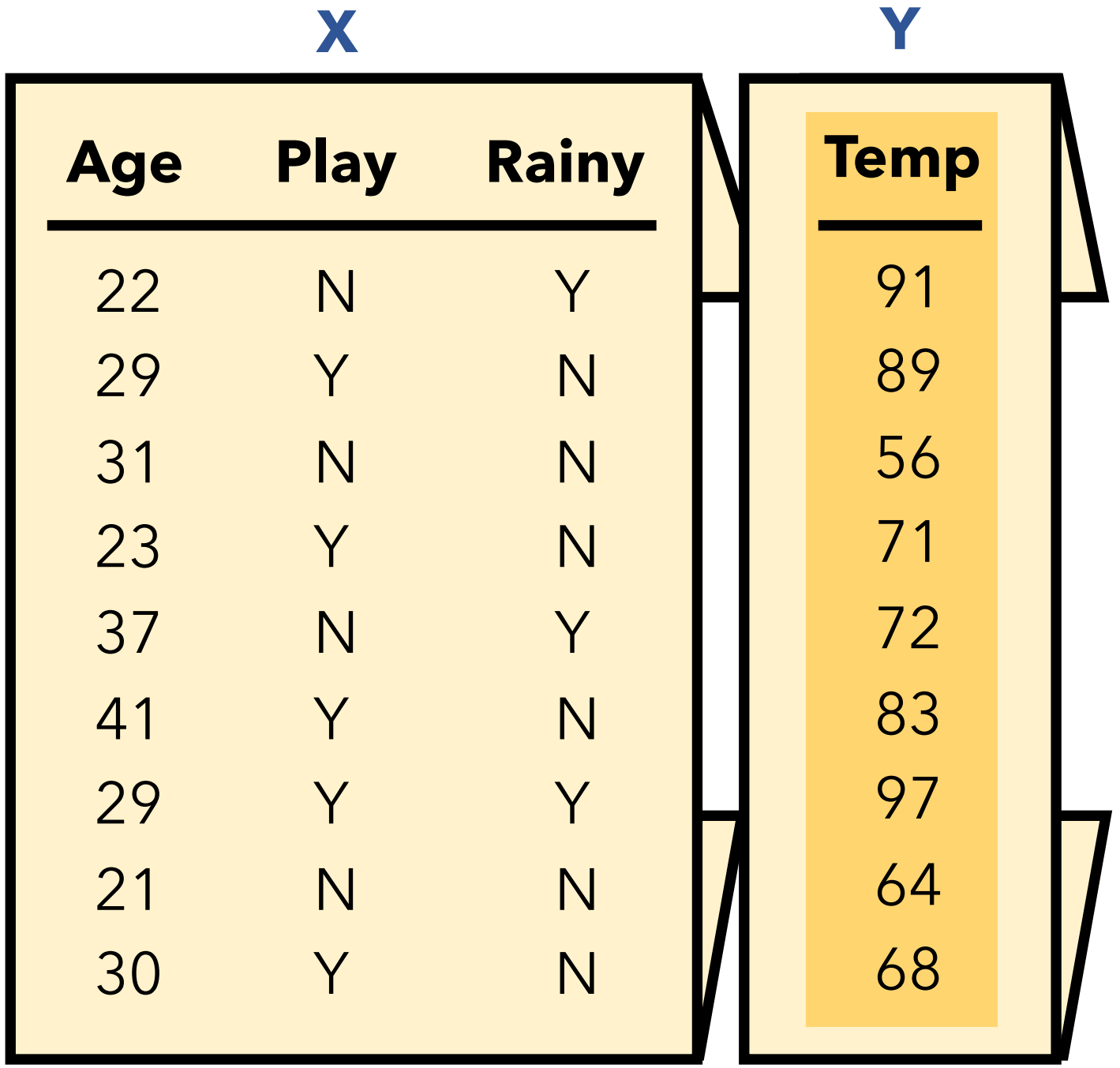


X			**Y**
Age	**Play**	**Rainy**	**Temp**
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column (e.g. **Temp**)
- Let's divide our data and learn how data **X** is related to data **Y**
- Assert that: $\boldsymbol{Y} = f(\boldsymbol{X}) + \varepsilon$
- Want a model f that is:
  - Supervised

X			**Y**
Age	**Play**	**Rainy**	**Temp**
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column (e.g. **Temp**)
- Let's divide our data and learn how data **X** is related to data **Y**
- Assert that: $Y = f(X) + \varepsilon$
- Want a model f that is:
  - Supervised



Def:

Supervised models use target data, **Y**, to provide feedback so that your model can learn the relationship between **X** and **Y**.

$$\hat{Y} = f(X)$$

- Supervised

	X		Y
	Play	Rainy	Temp
	N	Y	91
	Y	N	89
	N	N	56
	Y	N	71
	N	Y	72
	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column (e.g. **Temp**)
- Let's divide our data and learn how data **X** is related to data **Y**
- Assert that: $\boldsymbol{Y} = f(\boldsymbol{X}) + \varepsilon$
- Want a model f that is:
  - Supervised
  - Predicts real numbers (regression model)

X			**Y**
Age	**Play**	**Rainy**	**Temp**
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

- Given some data such that each
- Predicts real numbers (regression model)

Def:

Regression models are **supervised** models, whereby **Y** are *continuous* values.

X		Y
	Rainy	Temp
	Y	91
	N	89
	N	56
	N	71
	Y	72
	N	83
	Y	97
21	N	64
30	Y	68

- Given some data such that each
- Predicts real numbers (regression model)

Def:

Regression models are **supervised** models, whereby **Y** are *continuous* values.

Classification models are **supervised** models, whereby **Y** are *categorical* values.

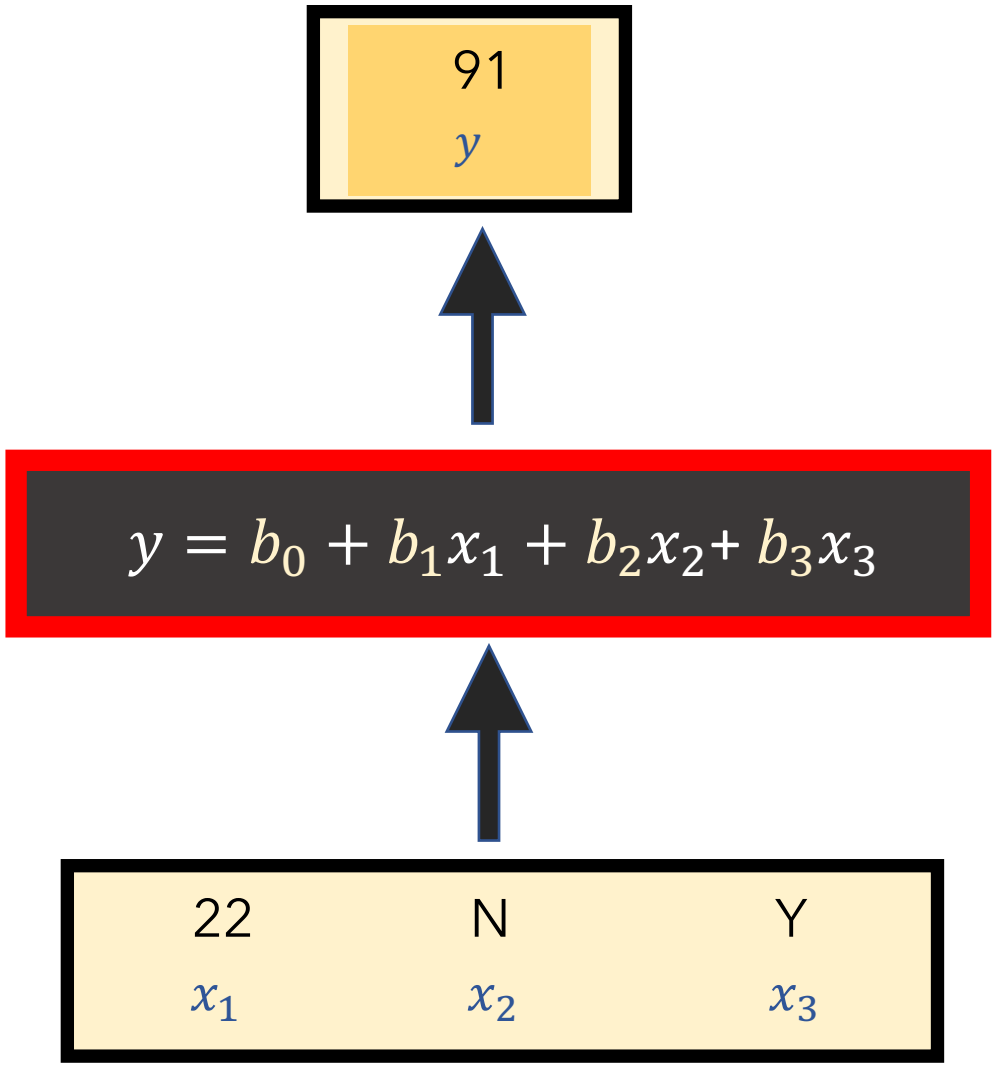
X		Y
	Rainy	Temp
	Y	91
	N	89
	N	56
	N	71
	Y	72
	N	83
	Y	97
21	N	64
30	Y	68

Linear Regression

High-level

Linear Regression

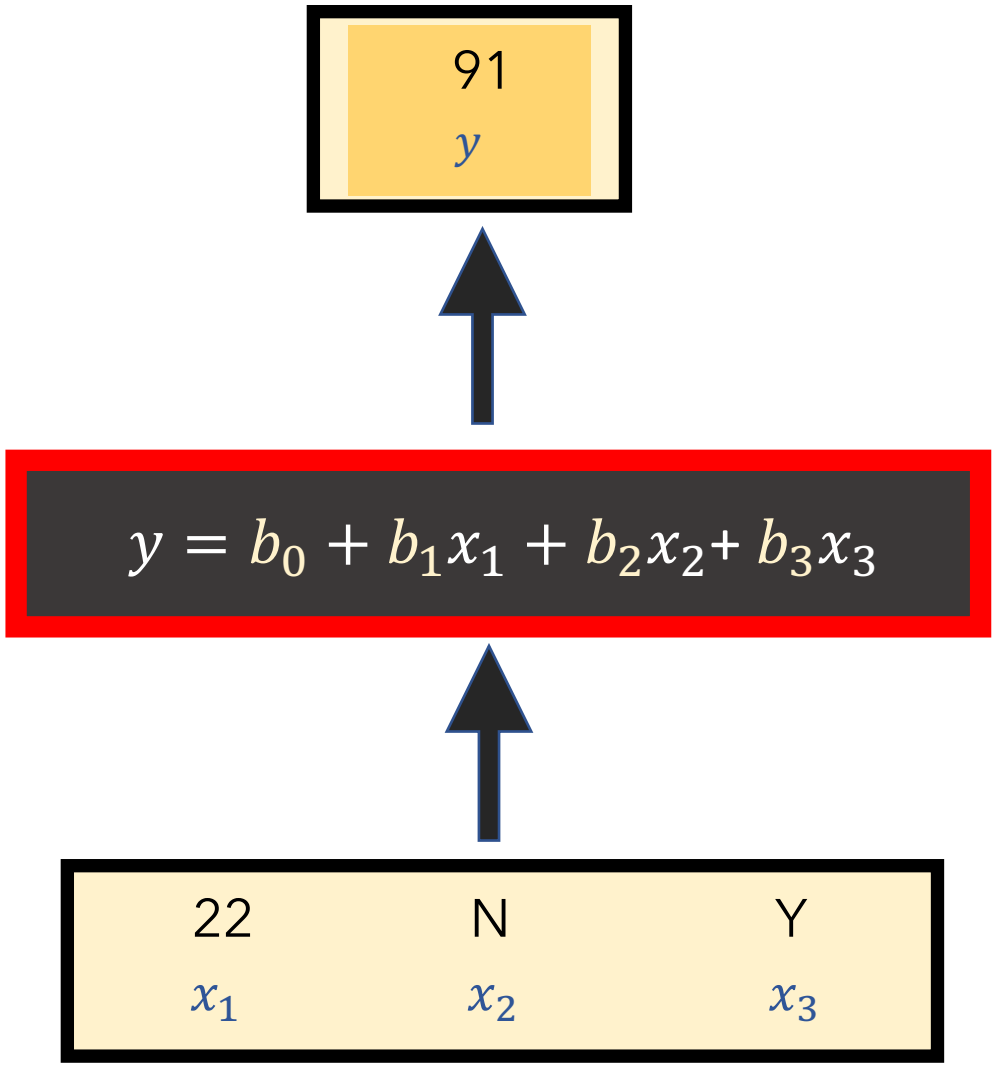
High-level



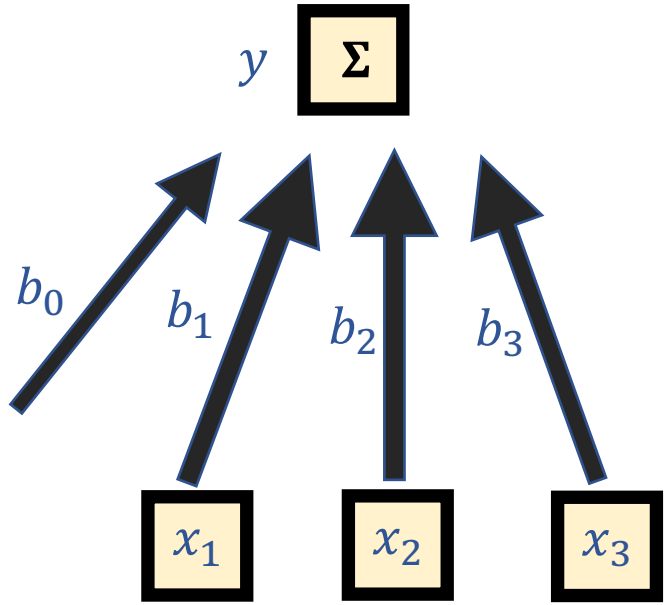
Mathematically

Linear Regression

High-level



Mathematically

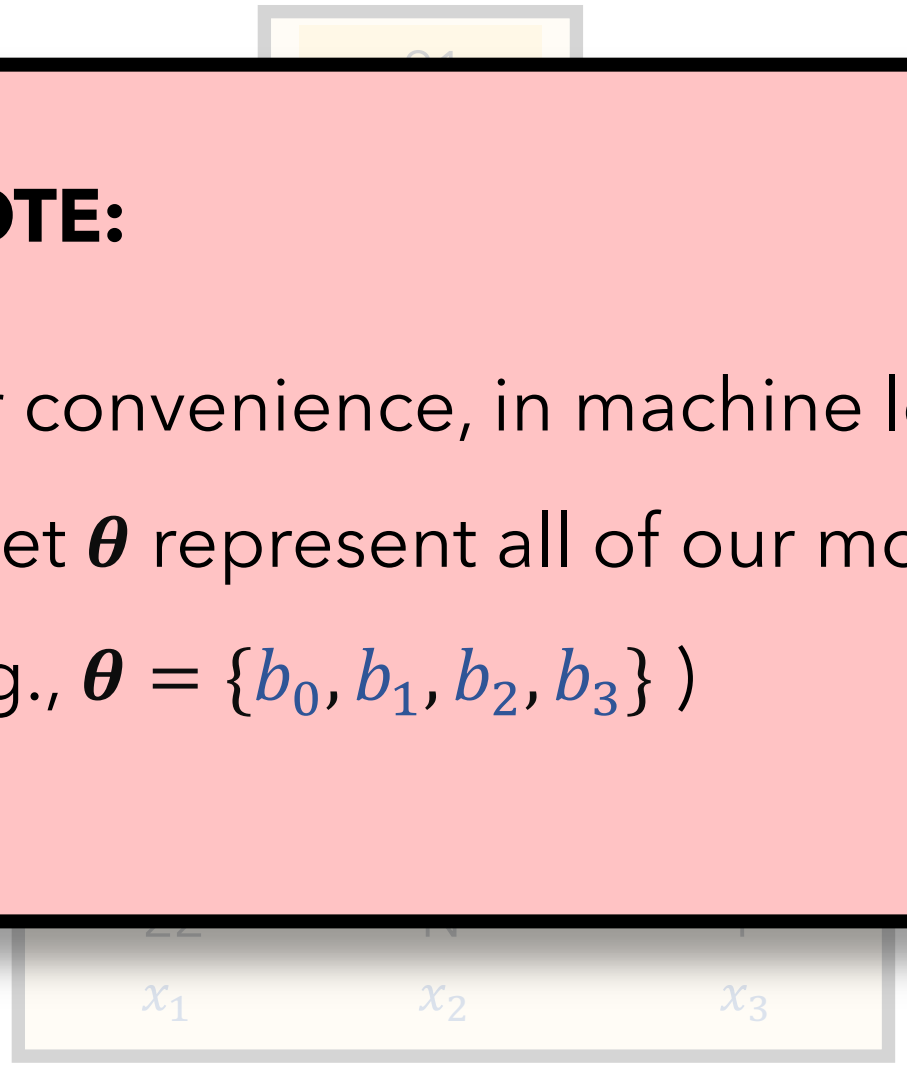


Graphically
(NN format)

Linear Regression

NOTE:

For convenience, in machine learning we tend to let $\boldsymbol{\theta}$ represent all of our model's parameters (e.g., $\boldsymbol{\theta} = \{b_0, b_1, b_2, b_3\}$)



High-level

Mathematically

Graphically
(NN format)

IMPORTANT

When **training** any supervised model,
be mindful of what you select for:

---

1. Our **loss function** (aka cost function)

   Measures how bad our current
   parameters $\boldsymbol{\theta}$ are

2. Our **optimization** algorithm?

   Determines how we update our parameters $\boldsymbol{\theta}$
   so that our model better fits our training data

IMPORTANT

When **training** any supervised model, be mindful of what you select for:

1. Our **loss function** (aka cost function)

   Measures how bad our current parameters $\boldsymbol{\theta}$ are

2. Our **optimization** algorithm?

   Determines how we update our parameters $\boldsymbol{\theta}$ so that our model better fits our training data

When **testing** our model's predictions, be mindful of what you select for:

3. Our **evaluation metric**

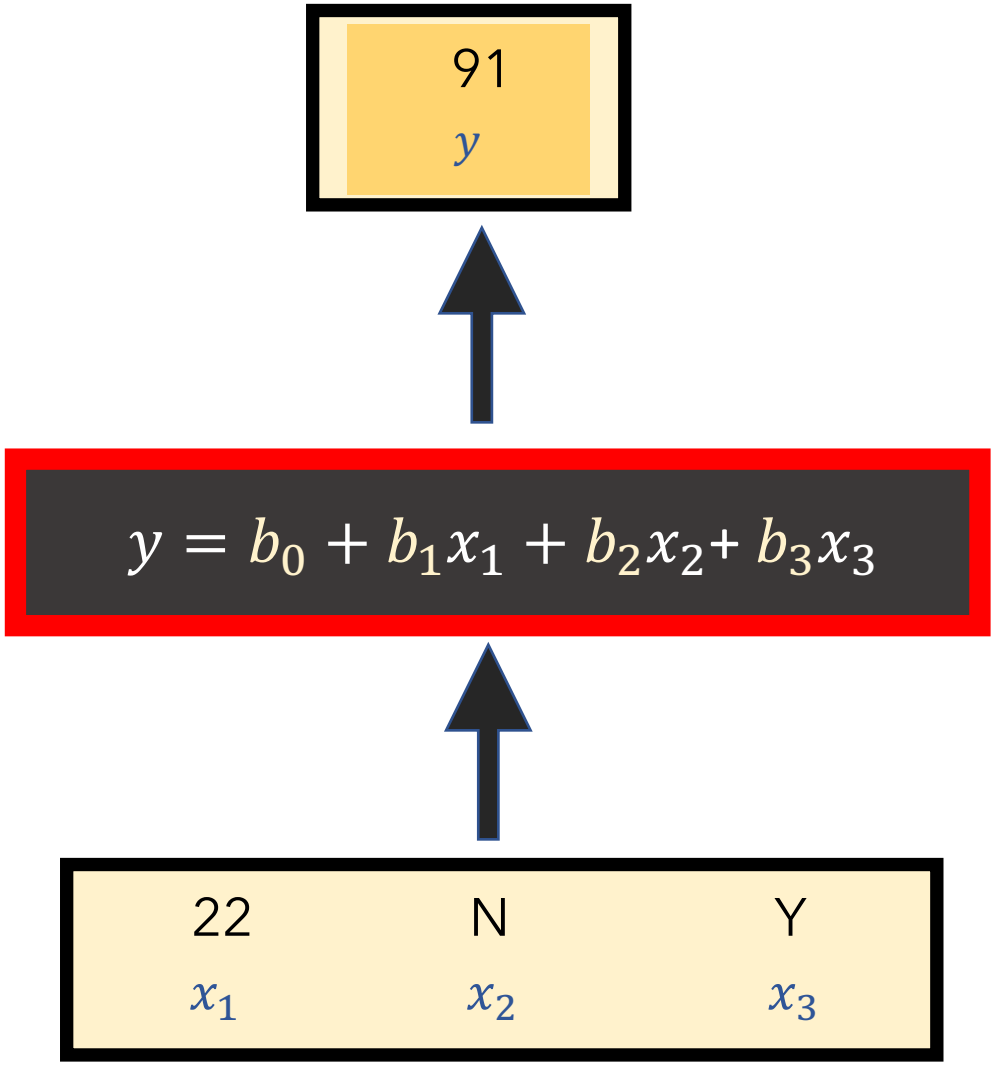
   Determines our model's performance (e.g., Mean Squared Error (MSE), R^2, $F1$ score, etc.)

Linear Regression

Q1 When training our model, how do we measure its m predictions $\hat{y}$?

A1 Cost function $J(\boldsymbol{\theta}) = \underbrace{\frac{1}{2}\sum_{i=1}^{m}(\hat{y} - y)^2}$

"Least Squares"



Mathematically

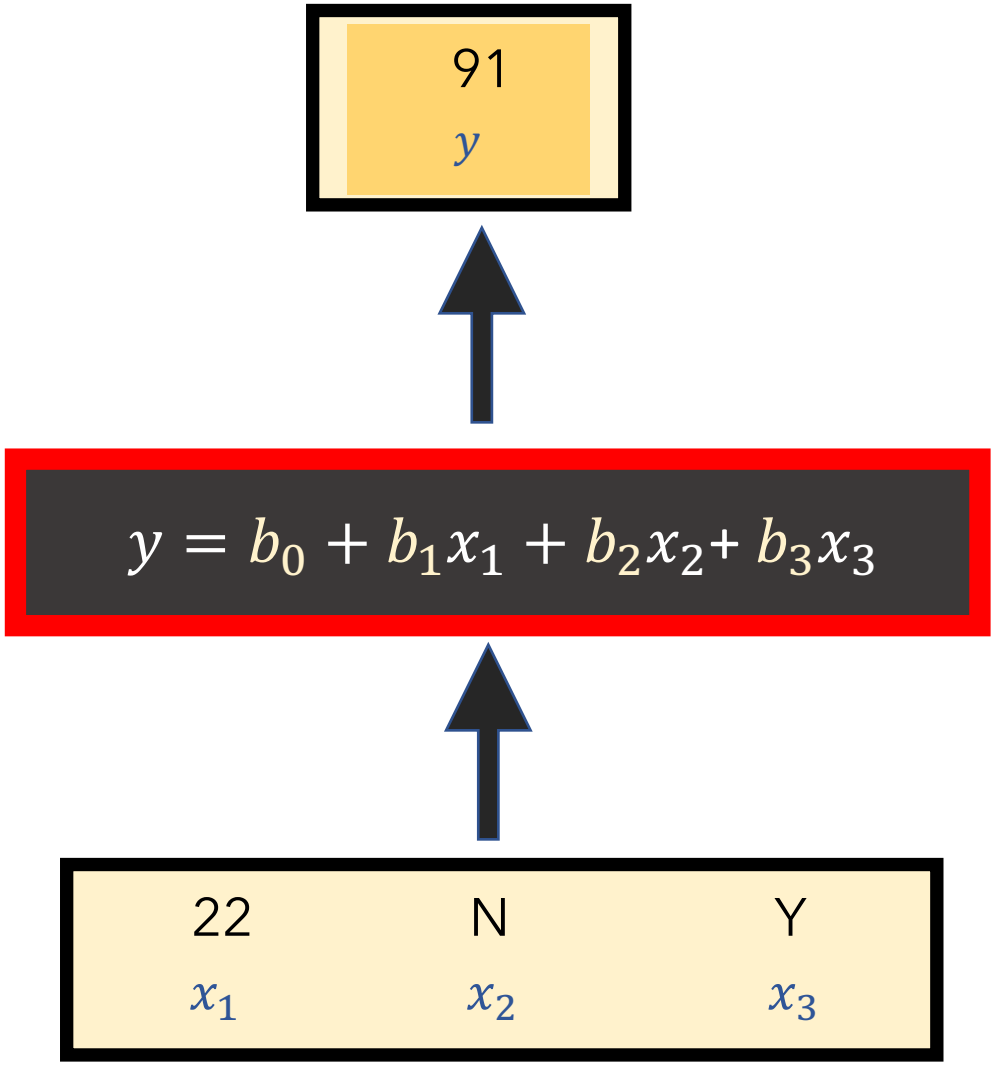
Linear Regression

Q1 When training our model, how do we measure its m predictions $\hat{y}$?

A1 Cost function $J(\boldsymbol{\theta}) = \underbrace{\frac{1}{2}\sum_{i=1}^{m}(\hat{y} - y)^2}$

"Least Squares"

- **Q2** How do we find the optimal $\boldsymbol{\theta}$ so that we yield the best predictions?



Mathematically

Linear Regression

Q1 When training our model, how do we measure its m predictions $\hat{y}$?

A1 Cost function $J(\boldsymbol{\theta}) = \underbrace{\frac{1}{2}\sum_{i=1}^{m}(\hat{y} - y)^2}$

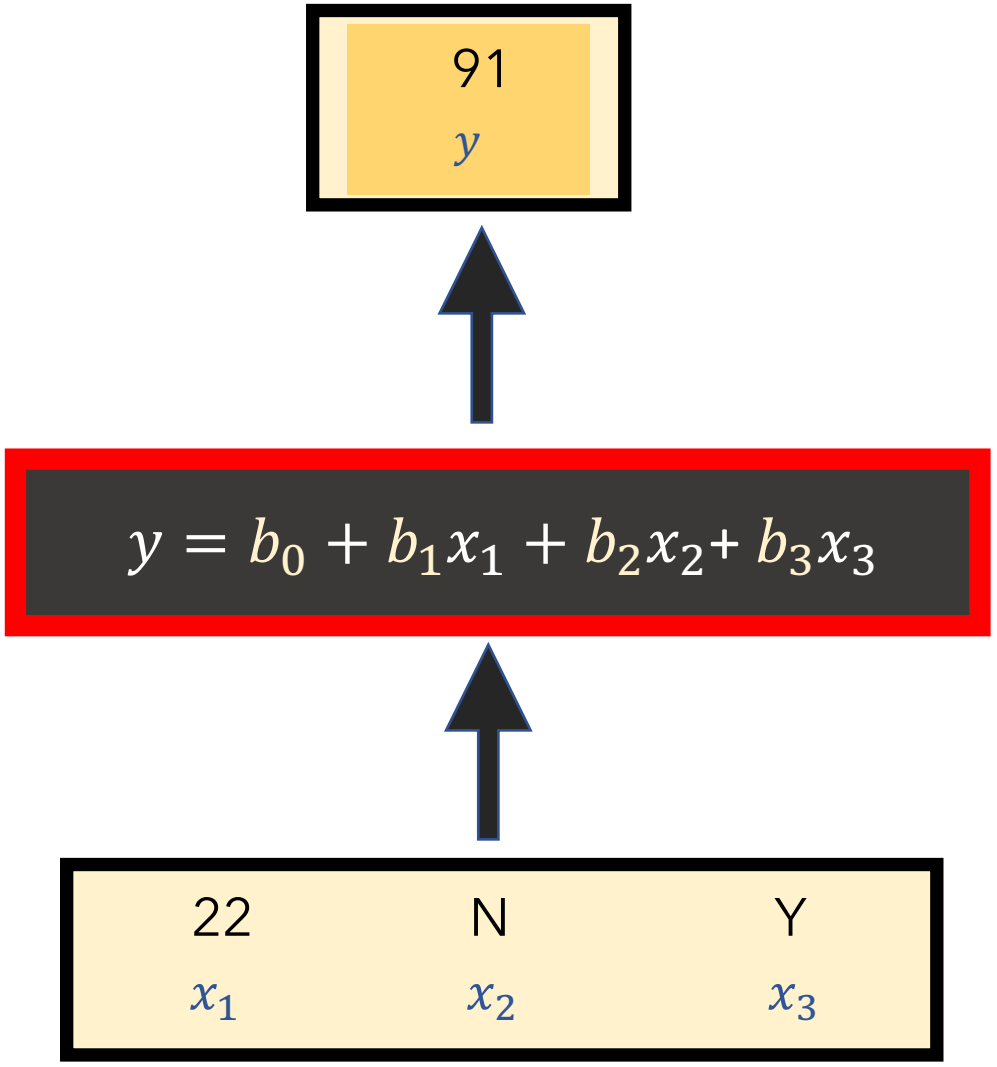
"Least Squares"

Q2 How do we find the optimal $\boldsymbol{\theta}$ so that we yield the best predictions?

A2 Two optimization algorithm options:

- Gradient Descent (iteratively search)
- Directly (closed-form solution)

$$\boldsymbol{\theta} = (X^T X)^{-1} X^T Y$$



Mathematically

Linear Regression

Fitted model example

The plane is chosen to minimize the sum of the squared vertical distances (per our loss function, least squares) between each observation (red dots) and the plane.

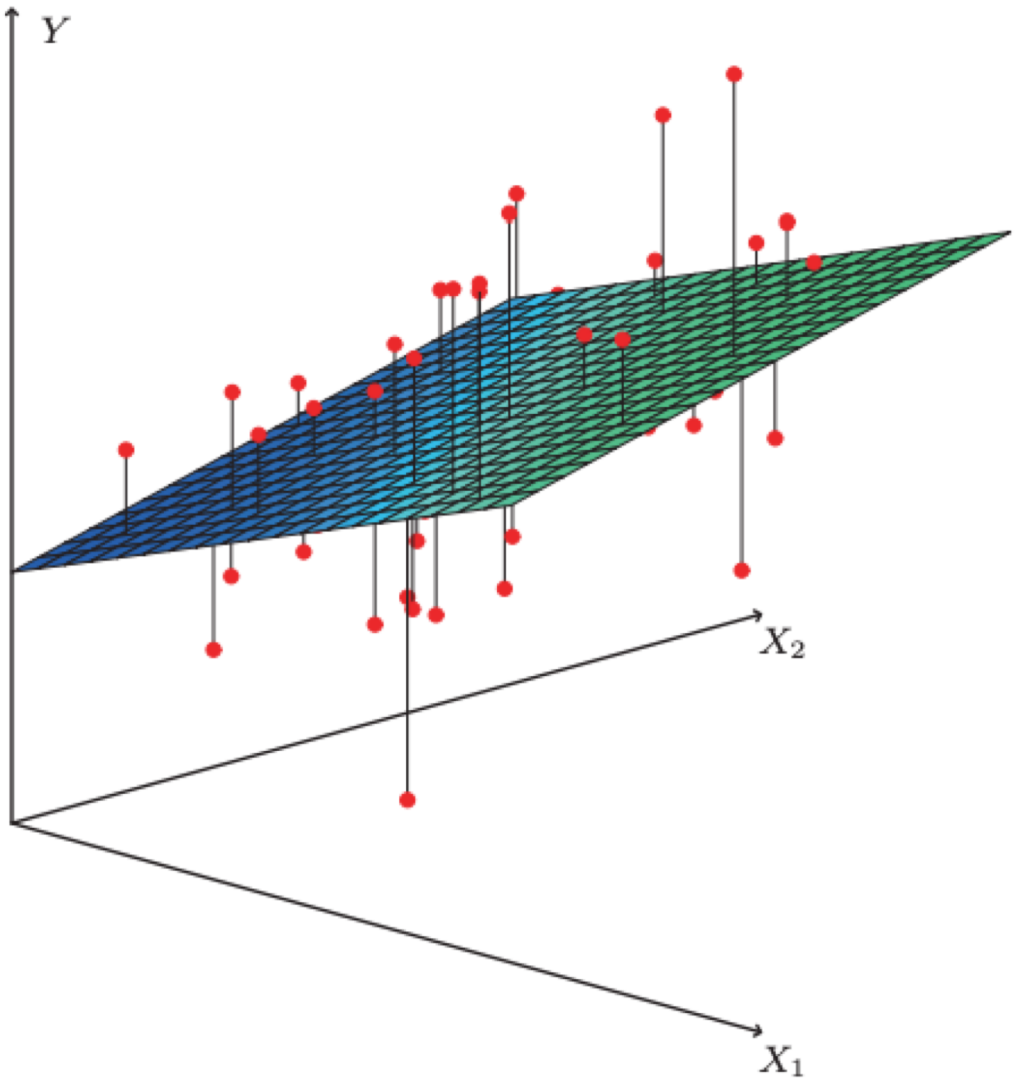


Photo from "*An Introduction to Statistical Learning*" (James, et al. 2017)

Linear Regression

PROS

- **Simple** and **fast** approach to model linear relationships
- Interpretable results via $\boldsymbol{\theta}$ ($\boldsymbol{\beta}$ coefficients)

CONS

- Can't model **non-linear** relationships
- Vulnerable to **outliers**
- Vulnerable to **collinearity**
- Assumes error terms are **uncorrelated***

* otherwise, we have false feedback during training

Supervised vs **Unsupervised**

Regression vs **Classification**

Linear Regression

Supervised

Regression

Parametric Models

- Parametric models assume our data **X** and **Y** can be represented by an underlying model $\boldsymbol{f}$ (i.e., $\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X}) + \varepsilon$) that has a particular form

  (e.g., a linear relationship, hence our using a linear model)

- Next, we aimed to fit the model $\boldsymbol{f}$ by estimating its parameters $\boldsymbol{\theta}$

  (we did so in a **supervised** manner)

Parametric Models

- Parametric models assume our data **X** and **Y** can be represented by an underlying model $\boldsymbol{f}$ (i.e., $\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X}) + \varepsilon$) that has a particular form
  (e.g., a linear relationship, hence our using a linear model)

- Next, we aimed to fit the model $\boldsymbol{f}$ by estimating its parameters $\boldsymbol{\theta}$
  (we did so in a **supervised** manner)

- Parametric models make the above assumptions. Namely, that there exists an underlying model $\boldsymbol{f}$ that has a fixed number of parameters.

	Supervised vs **Unsupervised**	**Regression** vs **Classification**	**Parametric** vs **Non-Parametric**
Linear Regression	**Supervised**	**Regression**	**Parametric**

Non-Parametric Models

Alternatively, what if we make no assumptions about the underlying model f ? Specifically, let's **not assume** f :

- has any particular distribution/shape

  (e.g., Gaussian, linear relationship, etc.)

- can be represented by a finite number of parameters.

Non-Parametric Models

Alternatively, what if we make no assumptions about the underlying model f ? Specifically, let's **not assume** f :

- has any particular distribution/shape

  (e.g., Gaussian, linear relationship, etc.)

- can be represented by a finite number of parameters.

This would constitute a non-parametric model.

Non-Parametric Models

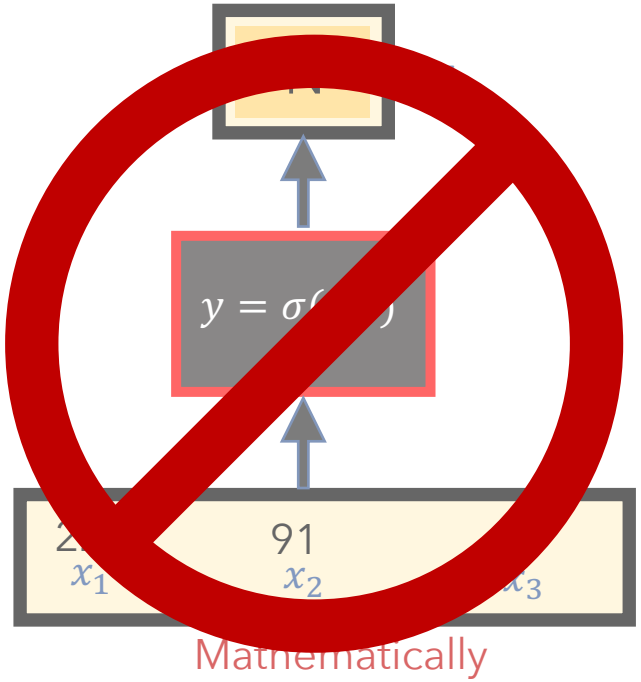
- Non-parametric models are **allowed to have parameters**; in fact, oftentimes the # of parameters grows as our amount of training data increases

- Since they make no strong assumptions about the form of the function/model, they are free to learn **any functional form** from the training data -- *infinitely complex*.

k-NN

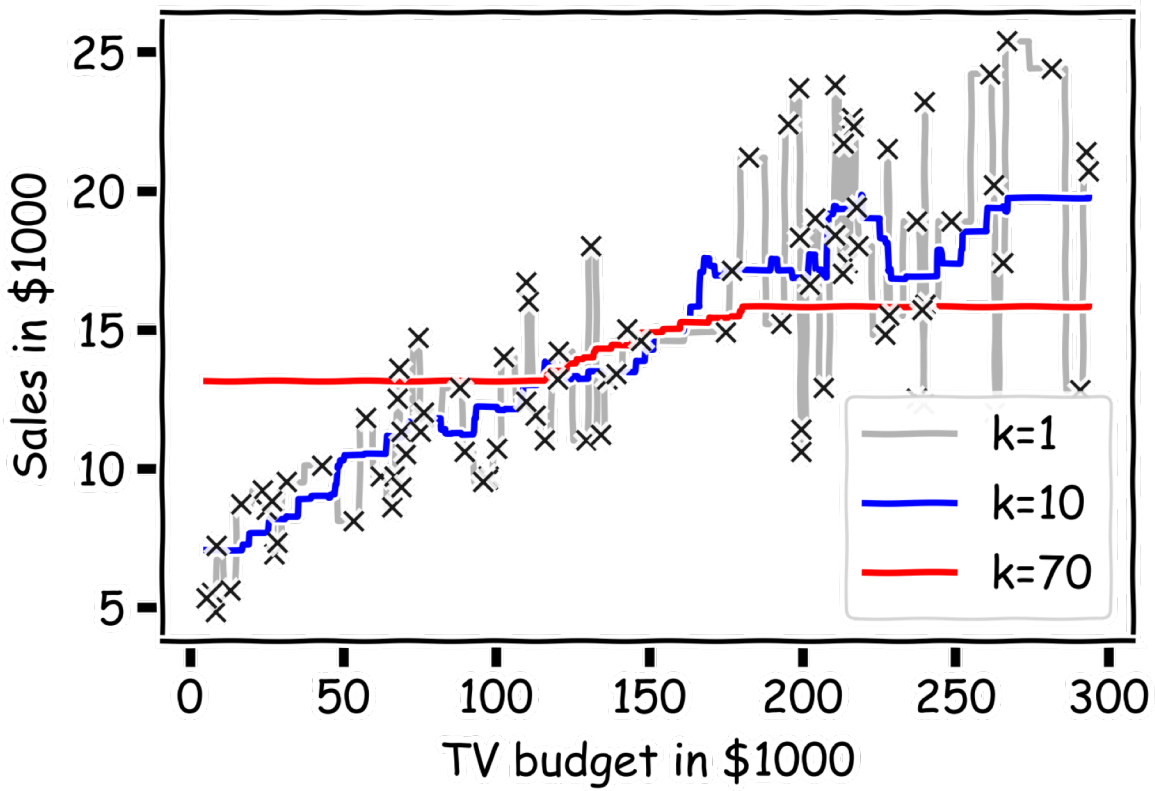
Refresher:

- k-NN doesn't train a model
- One merely specifies a $\boldsymbol{k}$ value
- At test time, a new piece of data $\boldsymbol{a}$:
  - must be compared to all other training data $\boldsymbol{b}$, to determine its k-nearest neighbors, per some distance metric $\boldsymbol{d(a, b)}$
  - is classified as being the majority class (if categorical) or average (if quantitative) of its k-neighbors



Conclusion:

- k-NN makes no assumptions about the data $\boldsymbol{X}$ or the form of $\boldsymbol{f}(\boldsymbol{X})$
- k-NN is a **non-parametric model**



k-NN

PROS

- **Intuitive** and **simple** approach
- Can model any type of data / places no assumptions on the data
- Fairly robust to missing data
- Good for highly sparse data
  (e.g., user data, where the columns are thousands of potential items of interest)

CONS

- Can be very **computationally expensive** if the data is large or high-dimensional
- Should carefully think about features, including scaling them
- Mixing quantitative and categorical data can be tricky
- Interpretation isn't meaningful
- Often, regression models are better, especially with little data

	Supervised vs **Unsupervised**	**Regression** vs **Classification**	**Parametric** vs **Non-Parametric**
Linear Regression	**Supervised**	**Regression**	**Parametric**
k-NN	**Supervised**	**either**	**Non-Parametric**

IMPORTANT

When **training** any supervised model, be careful of **overfitting** your model

A good model should generalize well to unseen (i.e., testing) data

---

Consider adding **regularization** term $R(\theta)$ to your cost function

Imposes a penalty based on your parameter values θ

L1 regularization: $R(\theta) = \sum_{i=1}^{n} |\theta_i|$ Prefers sparse weights (many 0's)

L2 regularization: $R(\theta) = \sum_{i=1}^{n} \theta_i^2$ Prefers many small-weight values

IMPORTANT

When **training** any supervised model, wisely use your training data

A good model should generalize well to unseen (i.e., testing) data

---

a. Shuffle your training data and optionally bootstrap samples

b. Perform cross-validation

Q1. Say our dataset has just 4 features (each row corresponds to a distinct day):

- Day of the week, where each value is one of {"M", "T", "W", "Th", "F", "Sa", "Su"}
- Average temperature outside today (range is from 21F to 102F)
- Average weather today, where each value is one of {"Sunny", "Cloudy", "Rainy", "Overcast", "Snowy"}
- # of people who paid to rent/ride a public city bike (i.e., Boston's BlueBike) for at least 10 minutes on the given day

We are interested in predicting the last feature by using the previous three. How would you go about modelling this? Data and modelling choices? Discuss w/ your group.

Q2. Say we want to estimate the quality of our Q1 model. In addition to using MSE, we want to use look at the R^2 value.

Imagine our code includes 2 variables:

- `target_y` which is a list of numbers that represent the gold truth numbers from the dataset
- `pred_y` which is a list of the numbers that represent our predictions
- Write 1 line of code to uses sklearn's built-in function to calculate the R^2 value

Break-out room time!

(Discussion, no coding)

Agenda

Review of Models

Review of Data

Debugging

Agenda

- Review of Models
- Review of Data
- Debugging

- Let's say our data has 1,000 rows

1,000

Your Data X

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68
45	N	N	61
21	N	Y	56
19	N	Y	36
44	Y	Y	53
36	Y	N	80
13	Y	Y	83
22	N	Y	68
28	Y	N	71
35	N	N	55
52	Y	Y	92
17	Y	Y	78

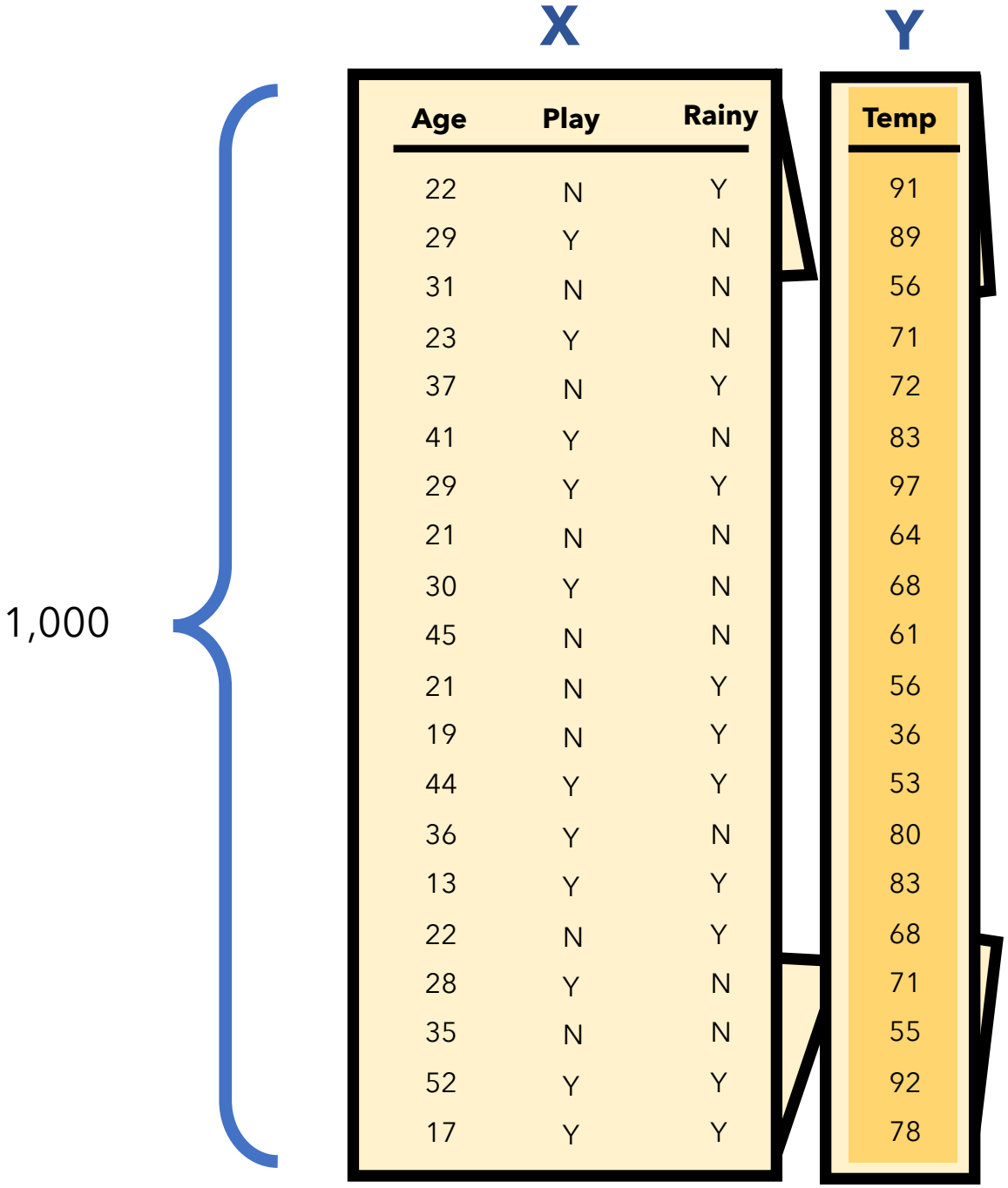
- Let's say our data has 1,000 rows
- And that we want to predict **Temp**

1,000

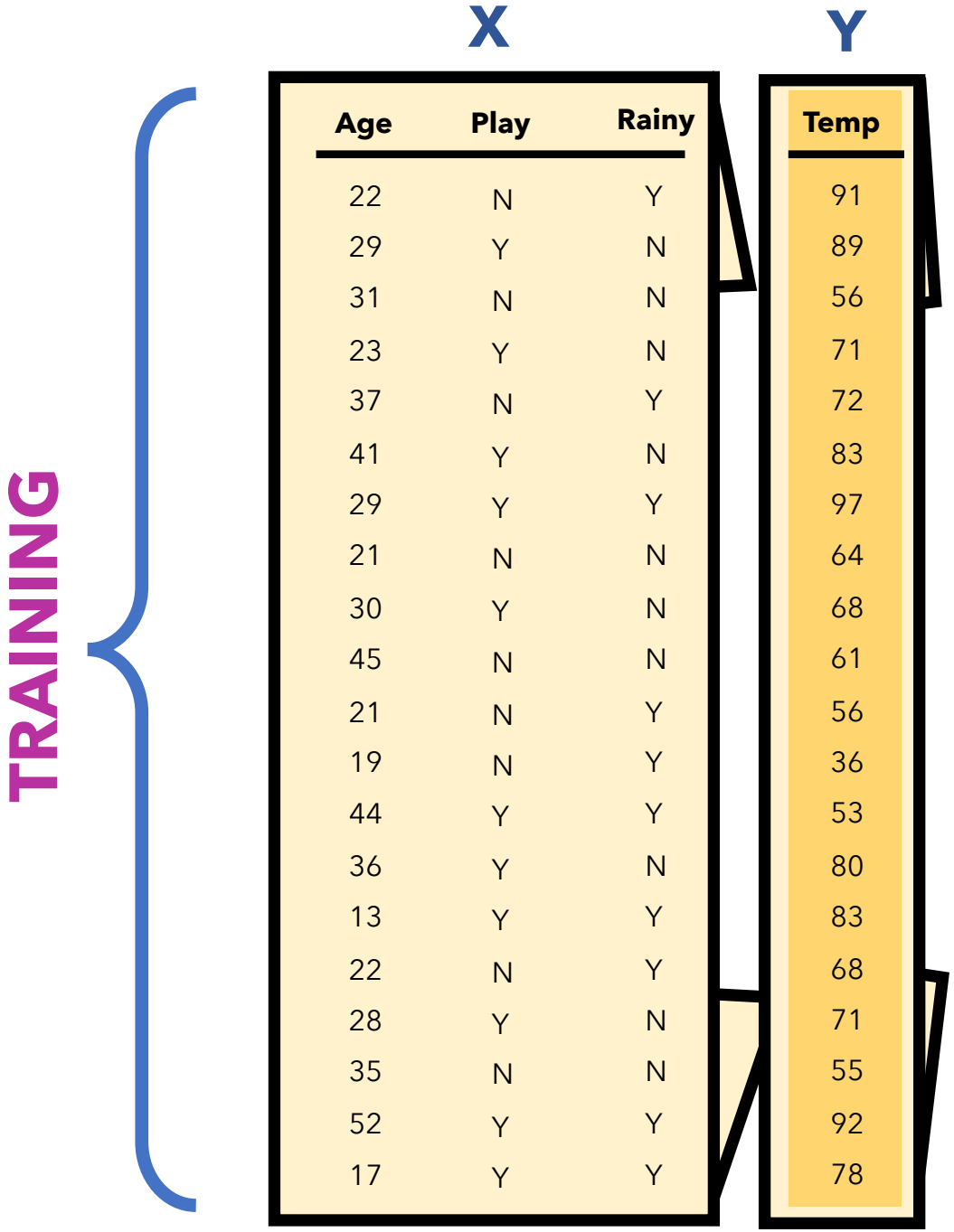
Your Data X

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68
45	N	N	61
21	N	Y	56
19	N	Y	36
44	Y	Y	53
36	Y	N	80
13	Y	Y	83
22	N	Y	68
28	Y	N	71
35	N	N	55
52	Y	Y	92
17	Y	Y	78

- Let's say our data has 1,000 rows
- And that we want to predict **Temp**



- Let's say our data has 1,000 rows
- And that we want to predict **Temp**
- **Which data should we use? All of it?**



- Let's say our data has 1,000 rows
- And that we want to predict **Temp**
- **Which data should we use? All of it?**

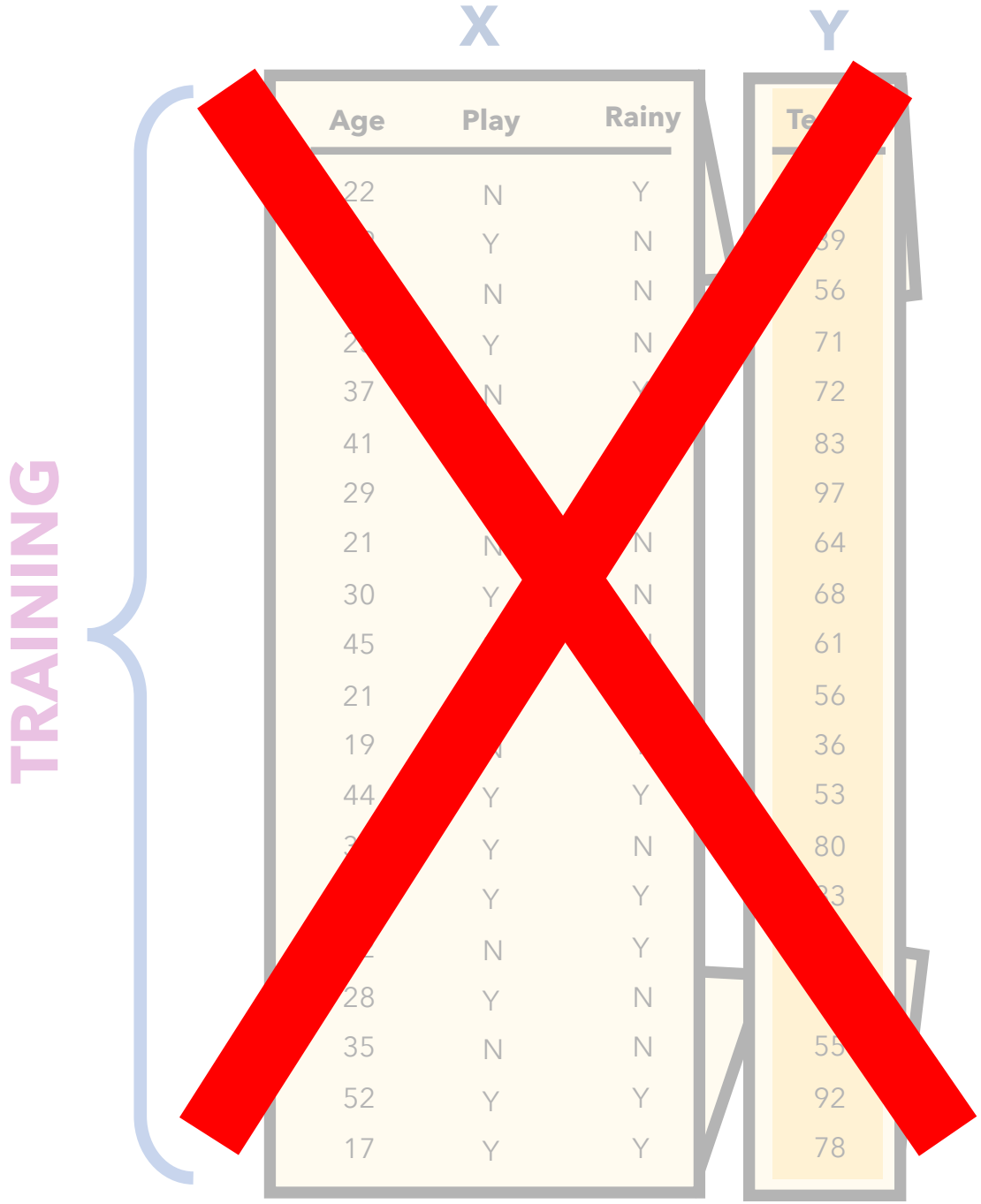
m_1 m_2 ••• m_N

- Could train n unique models on the **training set** (e.g., adjust the hyperparameters)
- Evaluate on **training set**

TRAINING

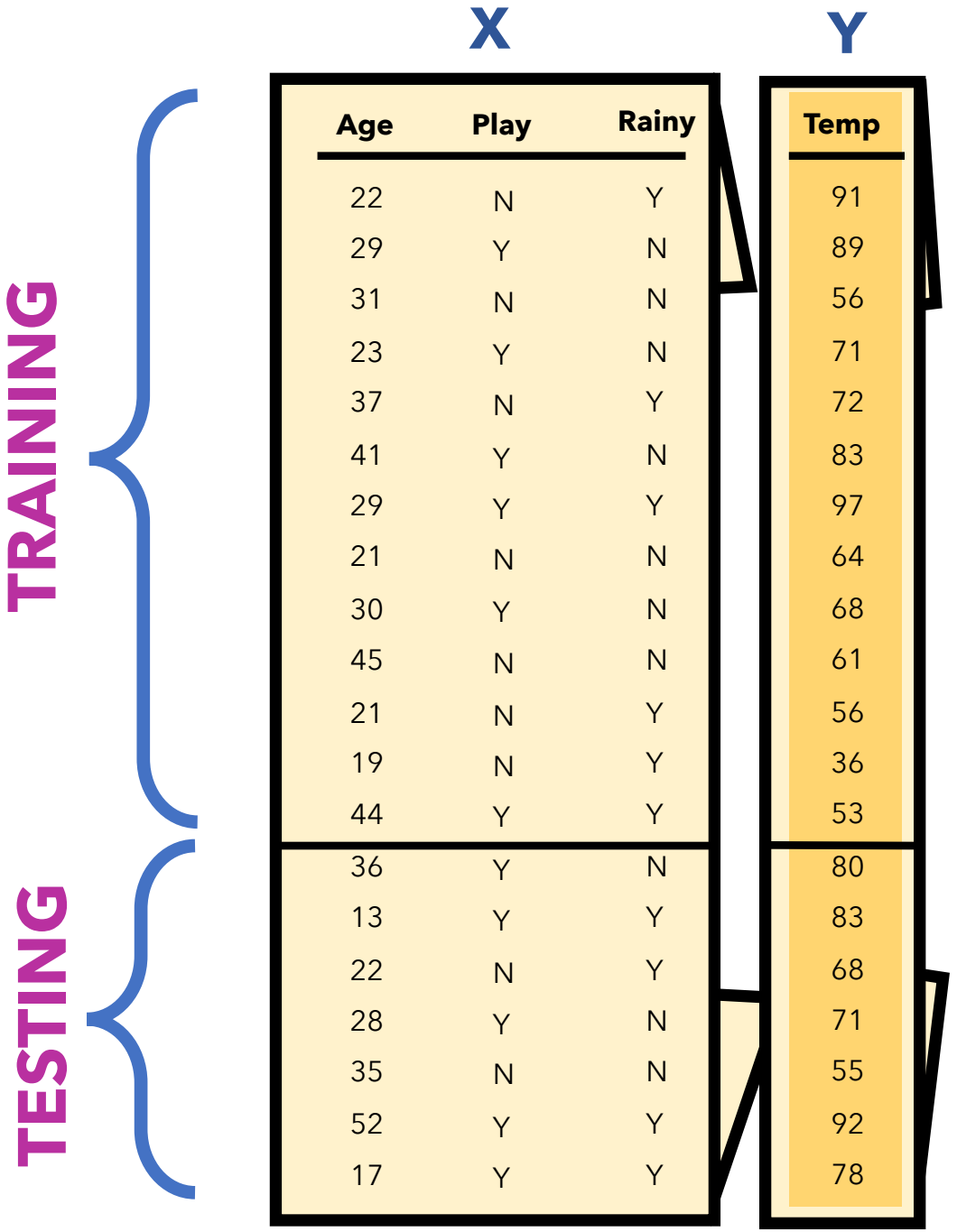
X			Y
Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68
45	N	N	61
21	N	Y	56
19	N	Y	36
44	Y	Y	53
36	Y	N	80
13	Y	Y	83
22	N	Y	68
28	Y	N	71
35	N	N	55
52	Y	Y	92
17	Y	Y	78

- Let's say our data has 1,000 rows
- And that we want to predict **Temp**
- **Which data should we use? All of it?**
  - Models always have a risk of just memorizing/overfitting on the training data
  - Many will likely yield 100% accuracy
  - We need some indication as to how well we'll do on **unseen, future data**



- Let's say our data has 1,000 rows
- And that we want to predict **Temp**
- **Which data should we use?**

  Split it in two?

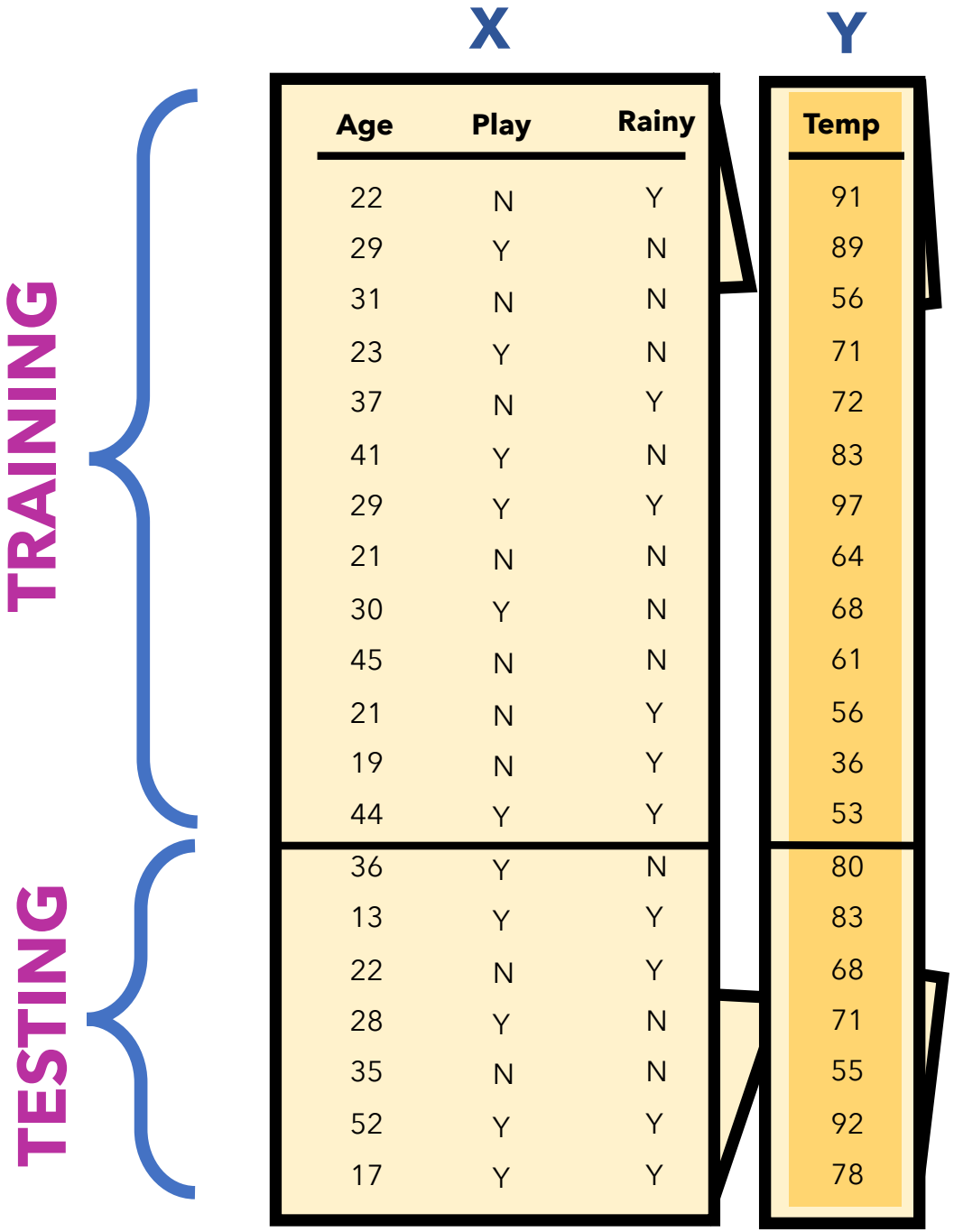


- Let's say our data has 1,000 rows
- And that we want to predict **Temp**
- **Which data should we use?**

  Split it in two?

m_1 m_2 ••• m_N

  - Could train models on the **training set**
  - Evaluate on the **testing set**
  - Select the model m_i that performs best on the **testing set**?



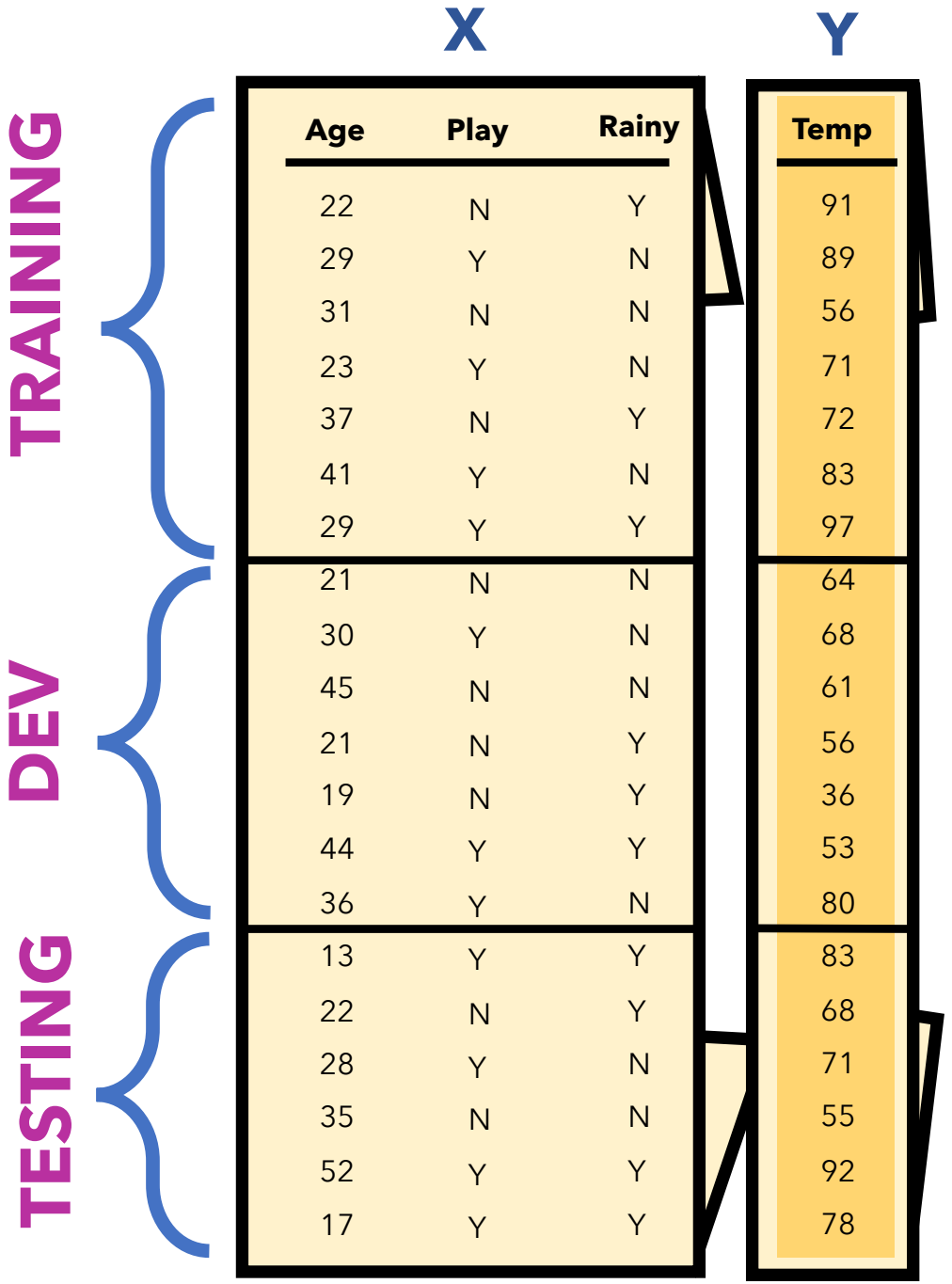
- Let's say our data has 1,000 rows
- And that we want to predict **Temp**
- **Which data should we use?**

  Split it in two?

  - Selecting our model based on its performance on the **testing set** could yield unrealistic expectations and it's like cheating, if such performance mattered (e.g., developing new models); it's like gaming the system

- Let's say our data has 1,000 rows
- And that we want to predict **Temp**
- **Which data should we use?**

  Split it into three?

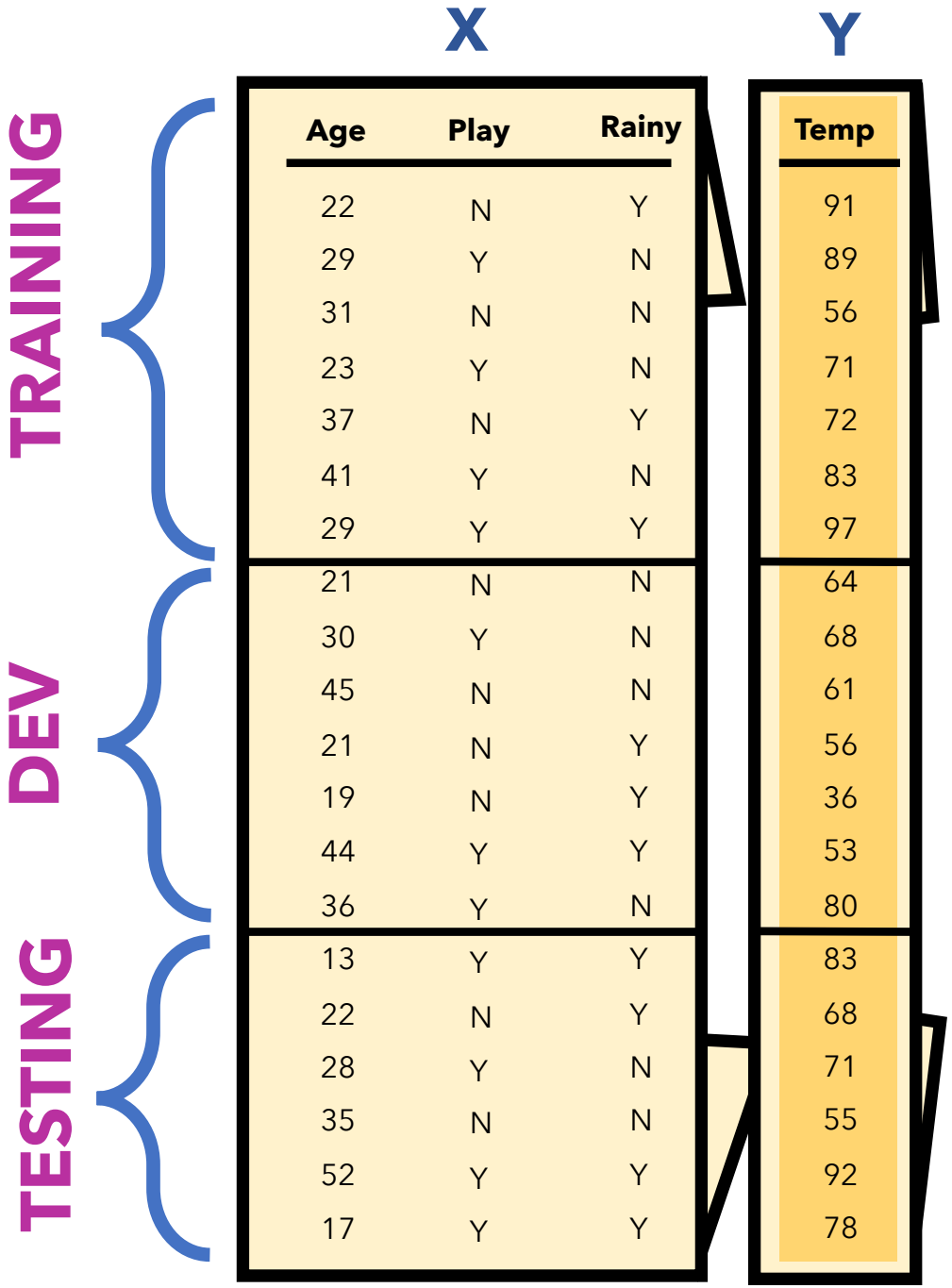


- Let's say our data has 1,000 rows
- And that we want to predict **Temp**
- **Which data should we use?**

  Split it into three?

m_1 m_2 ••• m_N

  - Could train models on the **training set**
  - Evaluate on the **dev set**
  - Select the model m_i that performs best on the **dev set**?

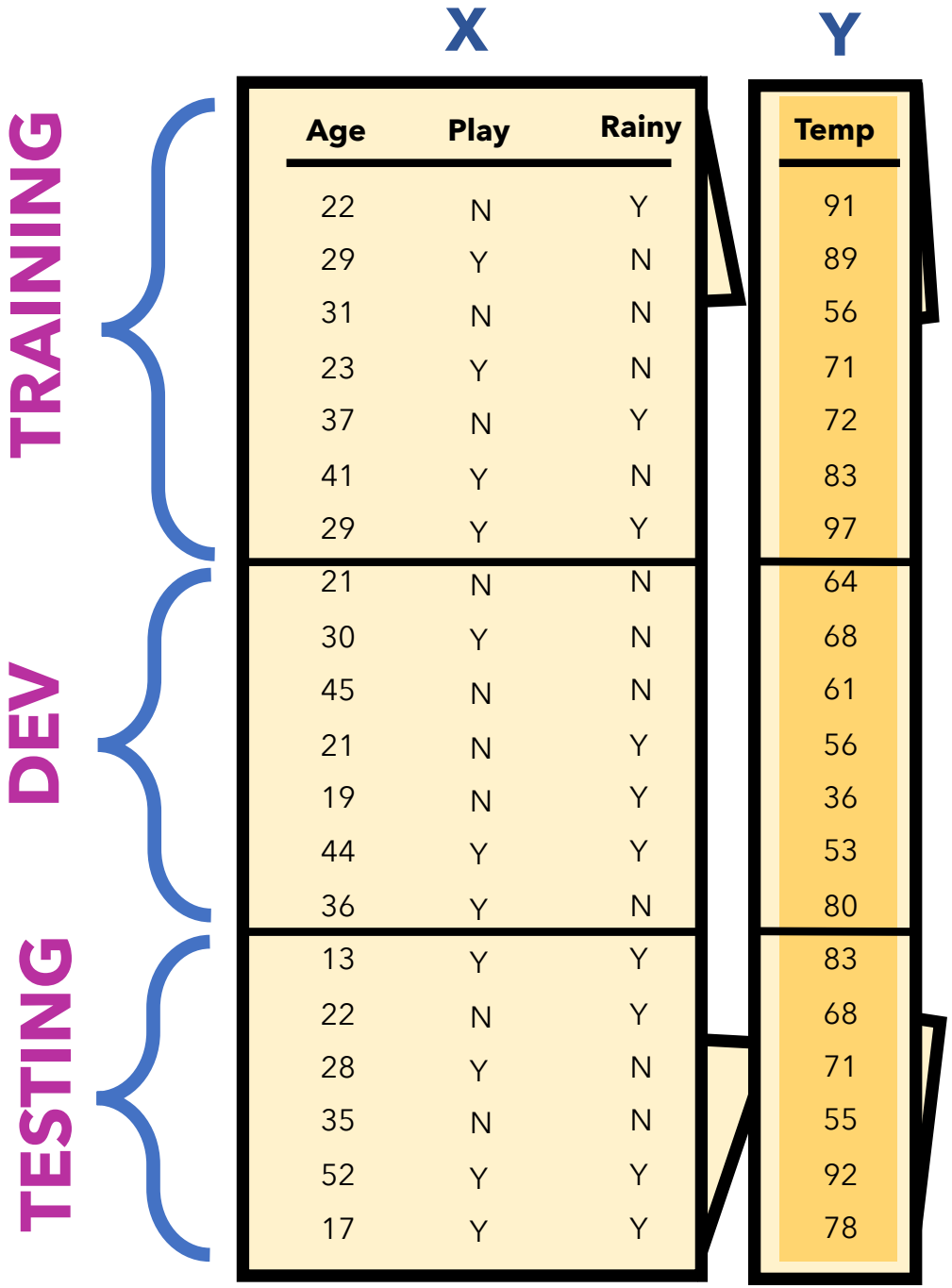


- Let's say our data has 1,000 rows
- And that we want to predict **Temp**
- **Which data should we use?**

  Split it into three?

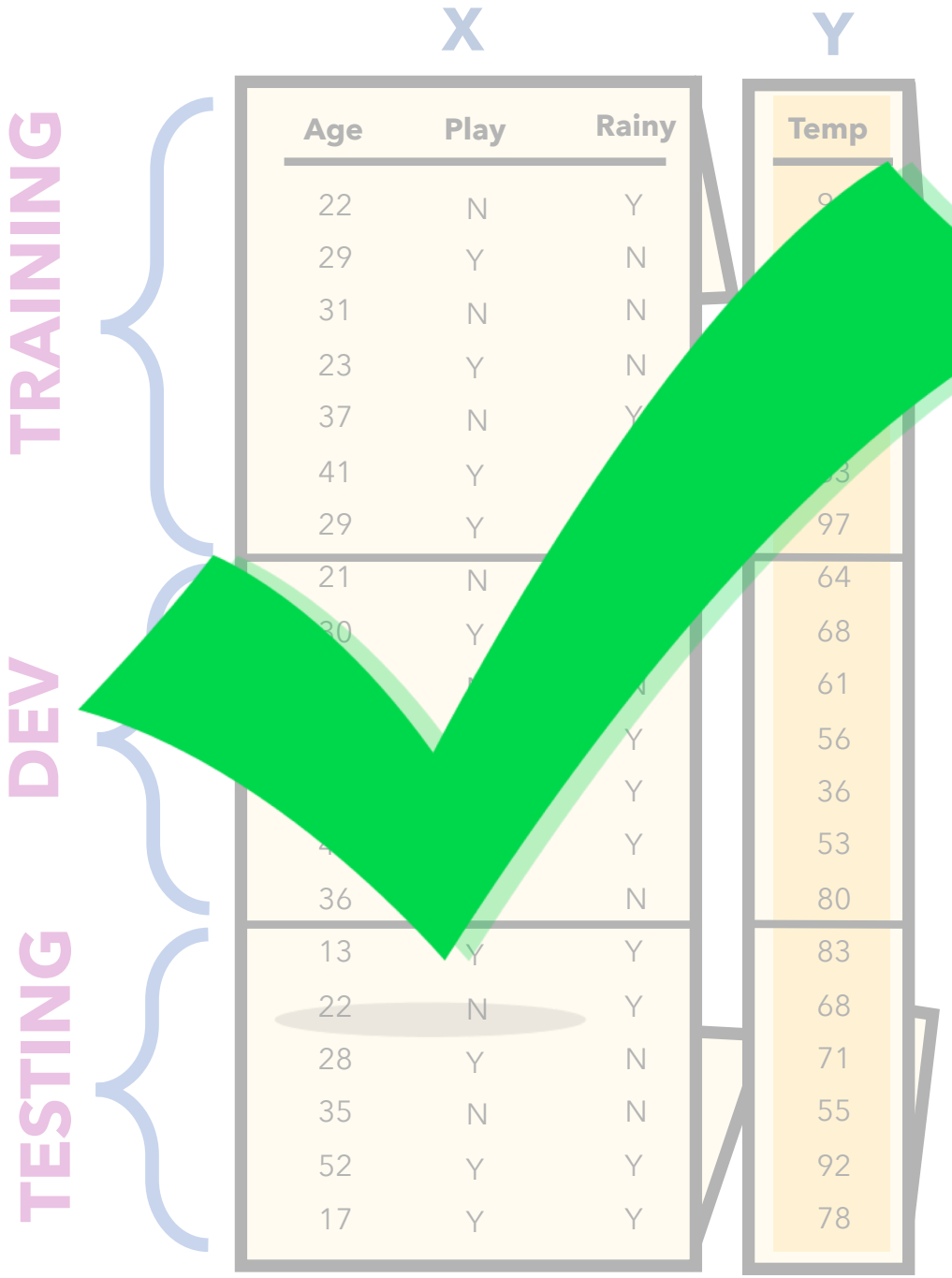
m_1 m_2 ••• m_N

  - Could train models on the **training set**
  - Evaluate on the **dev set**
  - Select the model m_i that performs best on the **dev set**?



- Let's say our data has 1,000 rows
- And that we want to predict **Temp**
- **Which data should we use?**

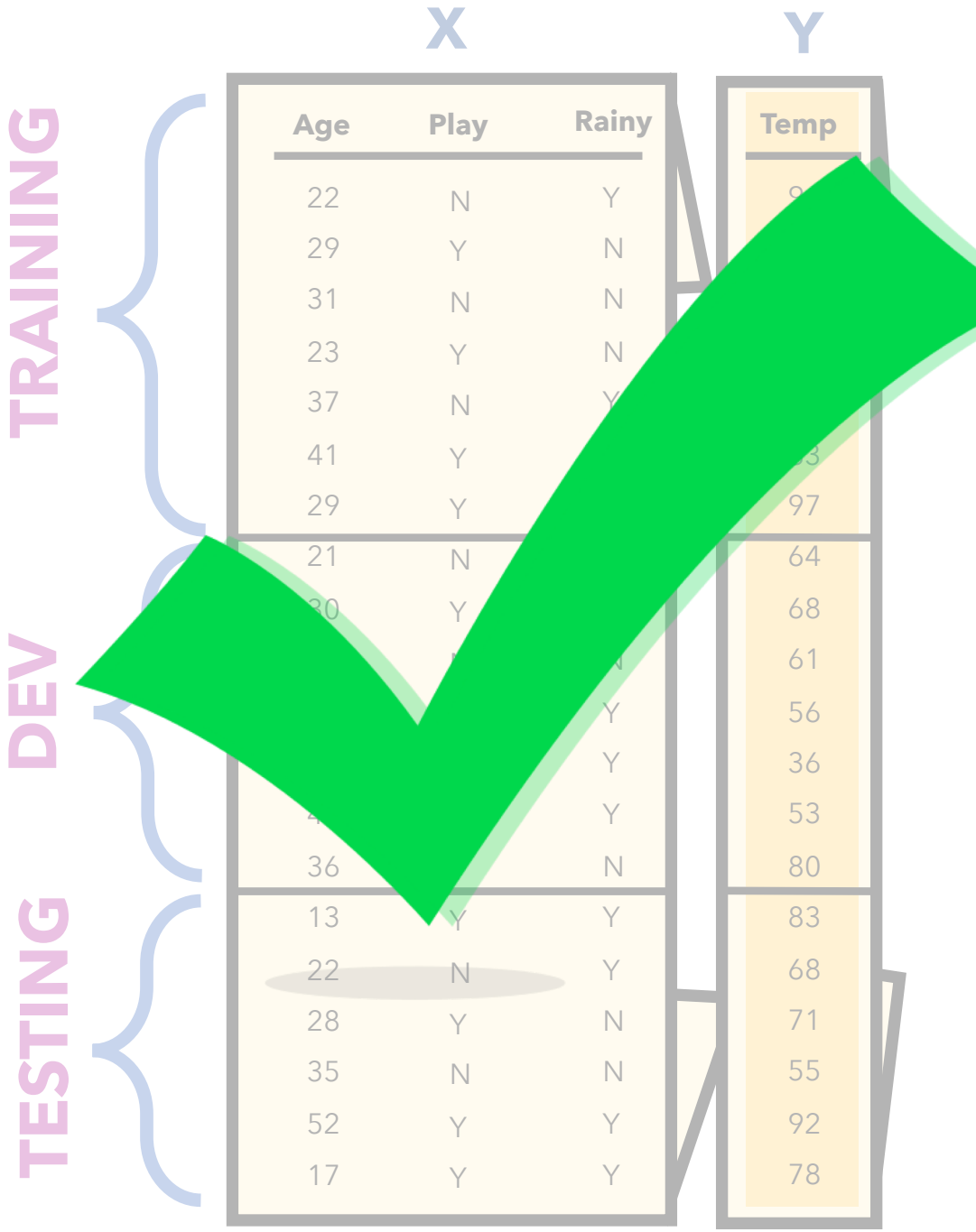
  Split it into three?
- Perfect! This allows us to tweak our models and optimize for the **dev** set, with no cheating, for we truly ignore the **test** set until we want a fair, final, one-time evaluation of our best model. **It's like blindfolding ourselves so we have a fair impression of the goodness of our model.**



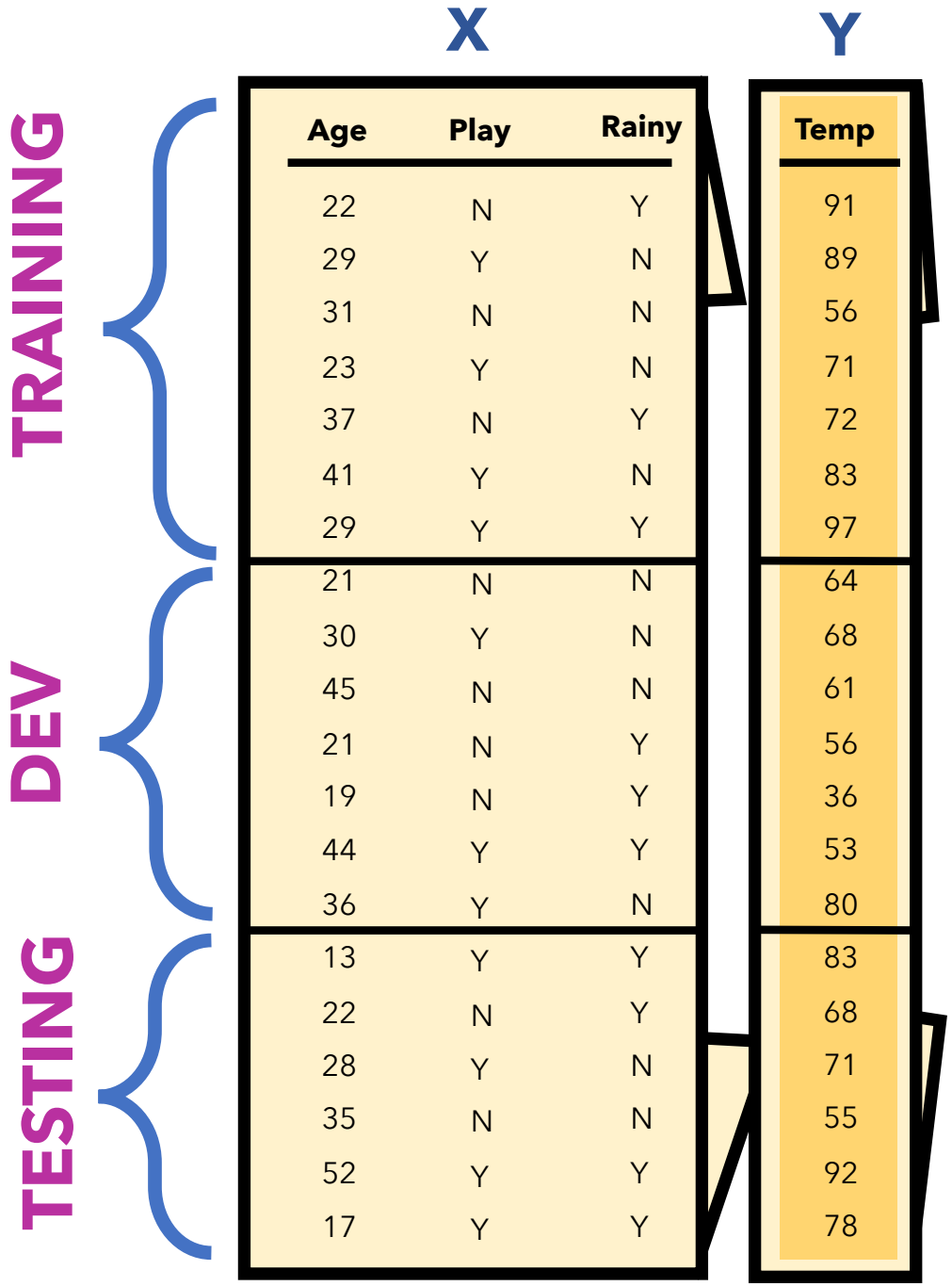
- Let's say our data has 1,000 rows
- And that we want to predict **Temp**
- **Which data should we use?**

  Split it into three?

- This allows us to fairly evaluate our model.
- Moreover, if we develop novel models, we can fairly compare against others' systems that run on the same data.



- The more data we have, the better
- We should always use these splits
- **Training** is typically ~70% of your entire data
- While this approach is sound, the **dev** set is often fairly small. Gives limited view into performance expectations. **Cross-validation** to the rescue!
- Popular datasets often come with these 3 splits already designated



ZOOM OUT

A machine learning model aims to use **data** to perform a particular **task** well. Often, the task is to predict or classify new data.

CHOMBOSAN / ALAMY STOCK PHOTO

Self-driving cars

Voice assistants

Face recognition

ZOOM OUT

If models for the following tasks only used a training set, no dev set or test, then all systems would be horrible!

CHOMBOSAN / ALAMY STOCK PHOTO

Self-driving cars

Woops, never seen this road before. CRASH!

Voice assistants

"Sorry, I've never heard words pronounced like this before."

Face recognition

Never seen your face before. Phone locked.

ANOTHER ANALOGY

When thinking of dataset splits, I like to think about a real student as our machine learning model.

ANOTHER ANALOGY

When thinking of dataset splits, I like to think about a real student as our machine learning model.

Say María is taking **organic chemistry**.

Her task is to learn the material, and her evaluation of such will be her final exam.

ANOTHER ANALOGY

María's **training** data is all of the course homework problems.

It prepares her. It's how she *learns*.

That data is her best indication of what's to come for the real-deal final exam (i.e., **testing** set).

ANOTHER ANALOGY

If she simply **memorized** (i.e., overfit to) her homework problems over the semester, she would likely be able to achieve 100% on those exact problems (the **training** set).

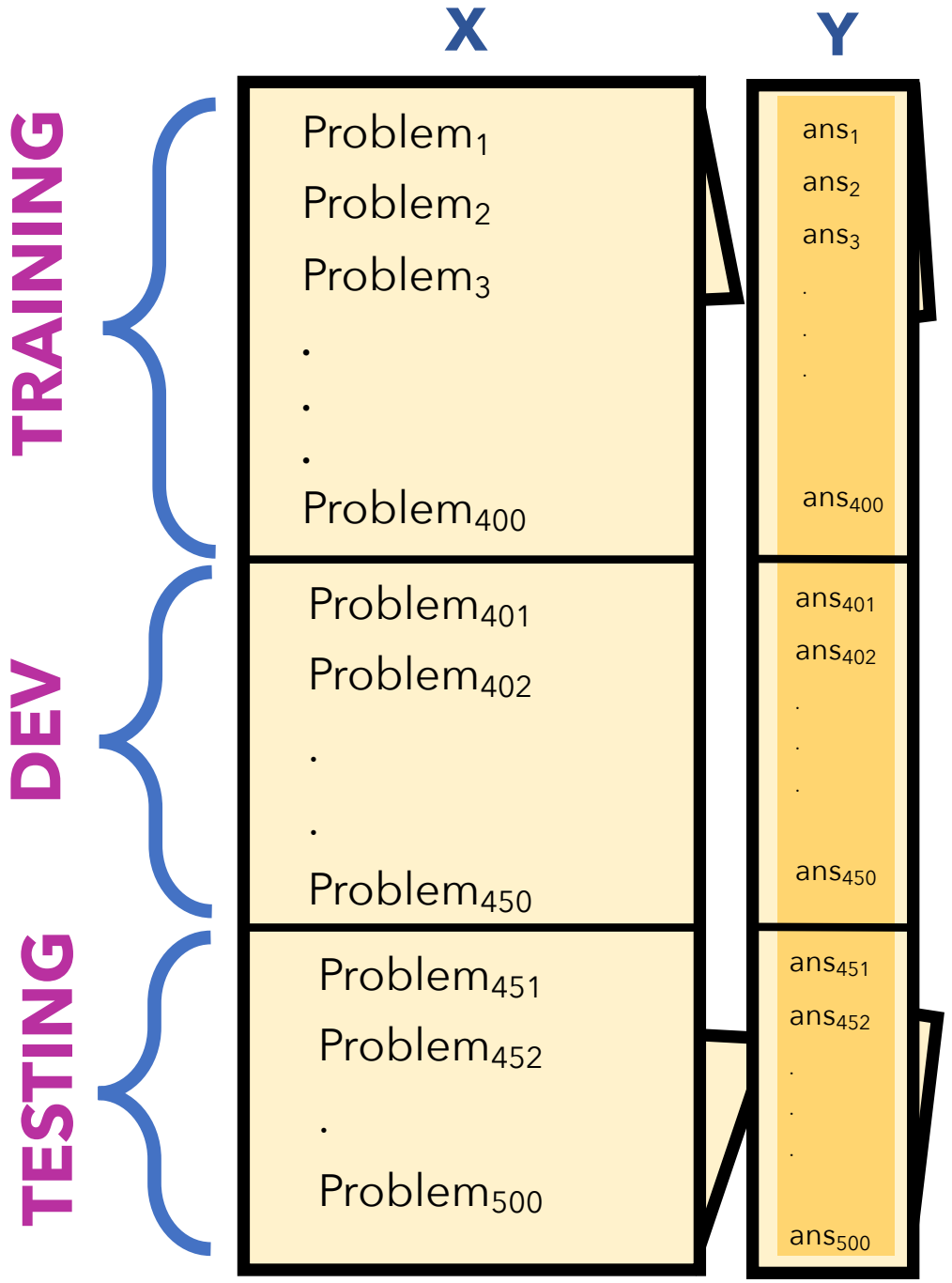
But, this gives no indication as to how she'll fare on the final exam (the **testing** set).

We always assume the **testing** set is from the same distribution that generated the **training** data. If it's *identical* to the training (homework), though, that would be bizarre, trivial, and lame.

ANOTHER ANALOGY

It would be way better if she learned the homework (**training**), then used the held-out problems (**dev** set) to fine-tune her learning.

This would help her learn best for the unseen problems that are to come! (**test** set)



Agenda

Review of Models

Review of Data

Debugging

Agenda

Review of Models

Review of Data

Debugging

Programming

We have recently used two highly popular, useful libraries, **statsmodels** and **sklearn**.

statsmodels is mostly focused on the *inference* task. It aims to make good estimates for $\boldsymbol{f}()$ (via solving for our β's), and it provides expansive details about its certainty. It provides lots of tools to discuss confidence, but isn't great at dealing with test sets.

sklearn is mostly focused on the *prediction* task. It aims to make a well-fit line to our input data $\boldsymbol{X}$, so as to make good $\boldsymbol{Y}$ predictions for some unseen inputs $\boldsymbol{X}$. It provides a shallower analysis of our variables. In other words, **sklearn** is great at test sets and dev sets, but it can't really reveal much about the uncertainty in the parameters or predictions.

REMINDER

Inference: estimates the function f, but the goal isn't to make predictions for Y; rather, it is more concerned with understanding the relationship between X and Y.

Prediction: estimates the function f with the goal of making accurate Y predictions for some unseen X.

Programming

- Programming takes practice. No way around it. Practice, practice, practice.
- We teach the **fundamentals** of data science and machine learning. Library packages, by themselves, are not fundamentals; they are accessible, high-level tools to accomplish certain tasks. They also change from time-to-time. Thus, it would be horrible if the course's main focus was on functions provided by **pandas**, **NumPy**, **scikits-learn**, etc.
- Via examples, we aim to show the benefits and power of them.
- They are incredibly powerful tools, and we make heavy usage of them, but we rely on you to explore how to make them work for you.

Debugging

- **Debugging** is a large topic**,** and yet it's only a small aspect of the larger field of **software testing**
- Here are some of my tips:

Debugging

- Pay close attention to the error message (read the output)
- When coding, start simple! Stop trying to do too much at once.
- **print**() things out. Often. Check the **type**, **size**, and **contents** of the data structure at hand.
- Be suspicious and doubtful of everything you type. The default is for things to not work. By default, your code will not work.

Debugging

- Pay attention to scope and leaky variables
- In Jupyter Notebooks, pay careful attention to the order in which you run cells
- Writing on paper to draft ideas and debug can be highly useful.
- Get a proper debugger (e.g., with Visual Studio code editor).

Debugging

- Everything can be distilled down to data structures and algorithms..
- Code can have:
  - Correct logic, but incorrect implementation
  - Incorrect logic, but correct implementation of your bad logic
  - Incorrect logic, and incorrect implementation
  - Correct logic, correct implementation.
- If there's an error, it's either in your **logic** or **implementation**, or both

Exercise time!