

Lecture #13: Classification & Logistic Reg. Basics

CS1090A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai, and Chris Gumb

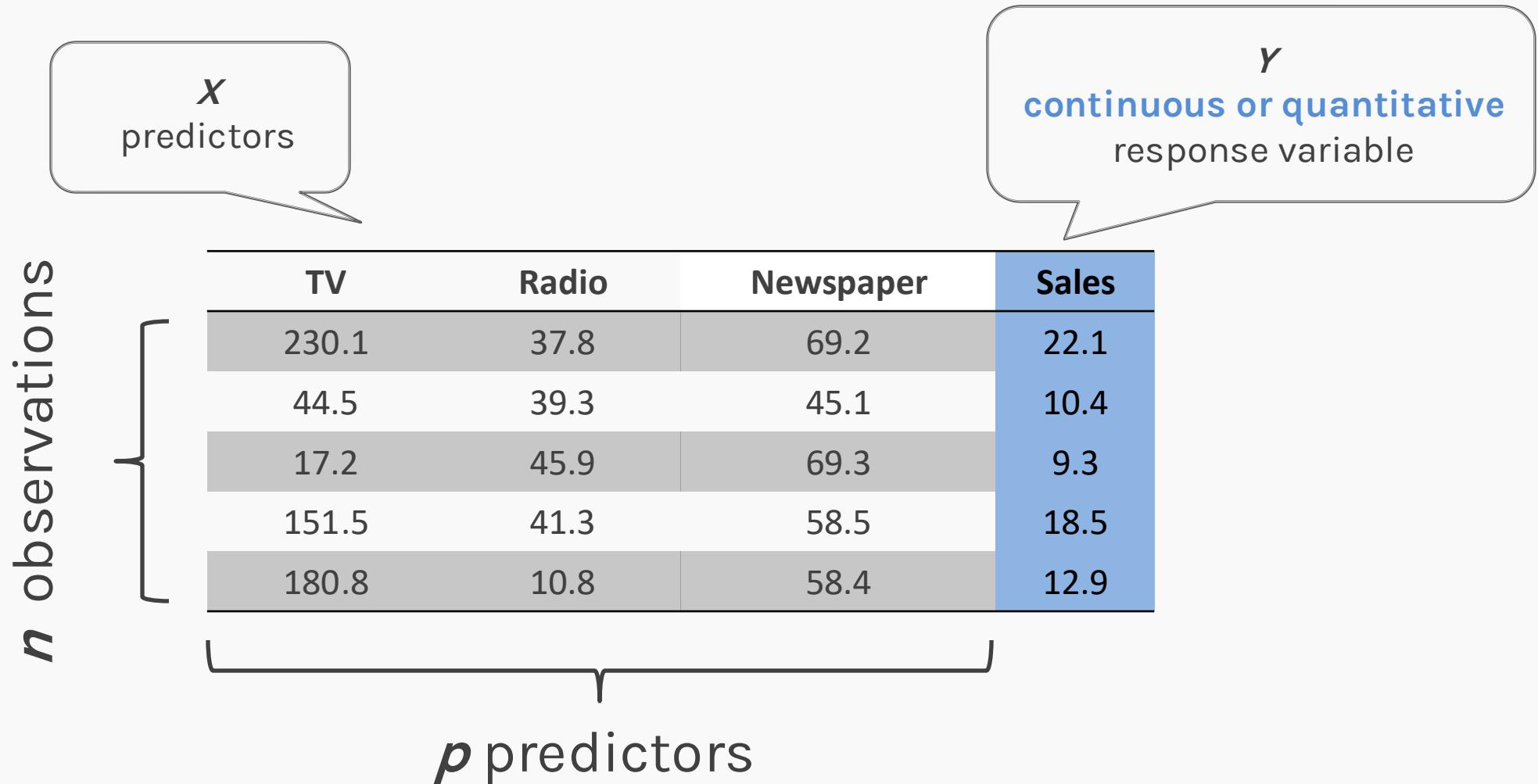


Outline

- **What is Classification?**
- Why not Linear Regression?
- Estimating the Simple Logistic Model
- Inference in Logistic Regression
- Multiple Logistic Regression
- Classification Decision Boundaries

What is Classification?

Advertising Data (from earlier lectures)



The diagram illustrates the structure of the advertising data. A speech bubble labeled X predictors points to the columns for TV, Radio, and Newspaper. Another speech bubble labeled Y continuous or quantitative response variable points to the Sales column. A large brace on the left indicates n observations, and a brace at the bottom indicates p predictors.

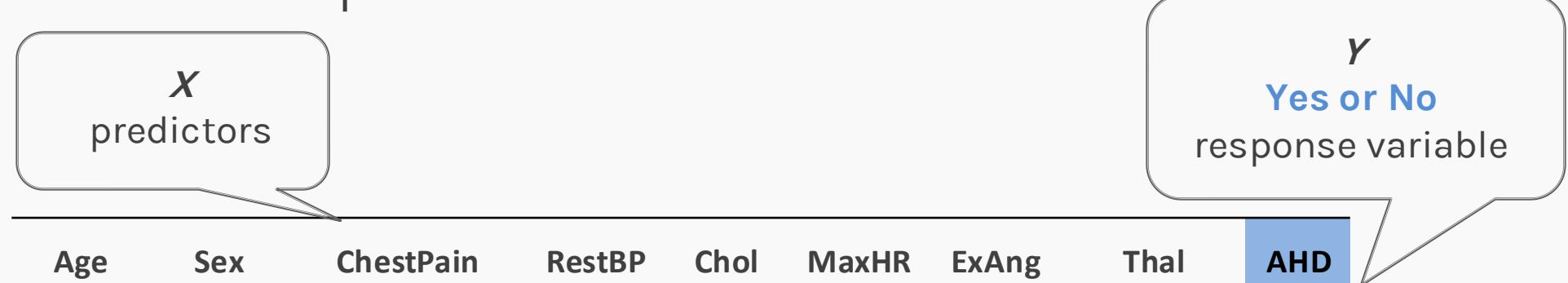
| | TV | Radio | Newspaper | Sales |
|--|-------|-------|-----------|-------|
| | 230.1 | 37.8 | 69.2 | 22.1 |
| | 44.5 | 39.3 | 45.1 | 10.4 |
| | 17.2 | 45.9 | 69.3 | 9.3 |
| | 151.5 | 41.3 | 58.5 | 18.5 |
| | 180.8 | 10.8 | 58.4 | 12.9 |

n observations

p predictors

What is Classification?

Consider another dataset that contains a **binary outcome** AHD for 303 patients who presented with chest pain.



| Age | Sex | ChestPain | RestBP | Chol | MaxHR | ExAng | Thal | AHD |
|-----|-----|--------------|--------|------|-------|-------|------------|-----|
| 63 | 1 | typical | 145 | 233 | 150 | 0 | fixed | No |
| 67 | 1 | asymptomatic | 160 | 286 | 108 | 1 | normal | Yes |
| 67 | 1 | asymptomatic | 120 | 229 | 129 | 1 | reversible | Yes |
| 37 | 1 | nonanginal | 130 | 250 | 187 | 0 | normal | No |

What is Classification?

Consider another dataset that contains a **binary outcome** AHD for 303 patients who presented with chest pain.

| X predictors | | | | | | | | AHD |
|--------------|-----|--------------|--------|------|-------|-------|------------|-----|
| Age | Sex | ChestPain | RestBP | Chol | MaxHR | ExAng | Thal | AHD |
| 63 | 1 | typical | 145 | 233 | 150 | 0 | fixed | No |
| 67 | 1 | asymptomatic | 160 | 286 | 108 | 1 | normal | Yes |
| 67 | 1 | asymptomatic | 120 | 229 | 129 | 1 | reversible | Yes |
| 37 | 1 | nonanginal | 130 | 250 | 187 | 0 | normal | No |

Yes indicates presence of heart disease

What is Classification?

Consider another dataset that contains a **binary outcome** AHD for 303 patients who presented with chest pain.

x
predictors

| Age | Sex | ChestPain | RestBP | Chol | MaxHR | ExAng | Thal | AHD |
|-----|-----|--------------|--------|------|-------|-------|------------|-----|
| 63 | 1 | typical | 145 | 233 | 150 | 0 | fixed | No |
| 67 | 1 | asymptomatic | 160 | 286 | 108 | 1 | normal | Yes |
| 67 | 1 | asymptomatic | 120 | 229 | 129 | 1 | reversible | Yes |
| 37 | 1 | nonanginal | 130 | 250 | 187 | 0 | normal | No |

What is Classification?

In summary,

Regression

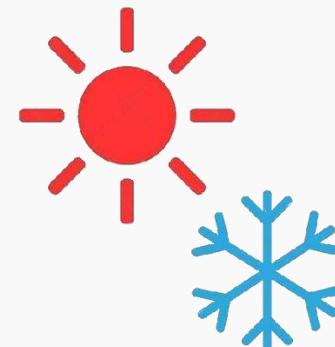
Performs well on tasks that require the prediction of a **quantitative** response variable.



What is the temperature going to be tomorrow?

Classification

Performs well on tasks that require the prediction of a **categorical or qualitative** response variable. It classifies an observation into a **category or class** labeled by Y.



Is it going to be hot or cold tomorrow?

Typical Classification Examples

Classification problems are ubiquitous in many domains, such as healthcare, finance, sports.

Some examples of classification problems are:

- To determine whether a startup is worth investing in
- To determine the disease type of patients based on various genomic markers
- To determine if a user is more likely to click on an advertisement.
- To determine if a given image is a real or a fake one

Outline

- What is Classification?
- Why not Linear Regression?
- Estimating the Simple Logistic Model
- Inference in Logistic Regression
- Multiple Logistic Regression
- Classification Decision Boundaries

Why not Linear Regression?

Assume you are given a dataset containing information of different students and your task is to predict whether their major is Computer Science, Statistics or otherwise.

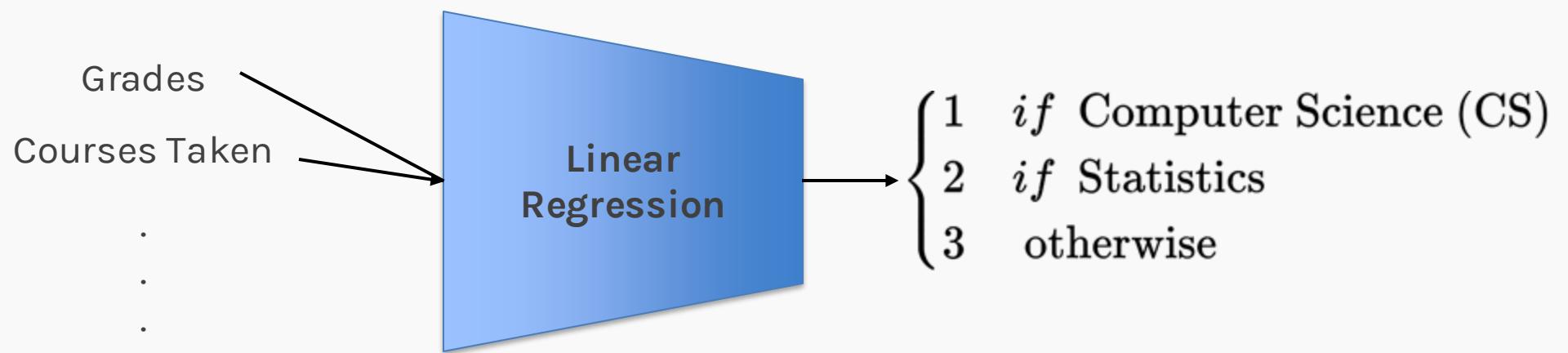
This categorical variable can't be used as is, but it could be encoded to be quantitative.

If y represents majors, then it could take on the values:

$$y = \begin{cases} 1 & \text{if Computer Science (CS)} \\ 2 & \text{if Statistics} \\ 3 & \text{otherwise} \end{cases}$$

Why not Linear Regression?

Now that we have encoded the values, a linear regression could be used to predict y from x .



But what is the problem here?

Why not Linear Regression?

This model would imply a specific ordering of the outcome.

For example, a change from $y=1$ to $y=2$ ([Computer Science](#) to [Statistics](#)) is the considered the same as a change from $y=2$ to $y=3$ ([Statistics](#) to [everyone else](#)). However, this change should not be interpreted as the same.

If a categorical response variable is [ordinal](#) (has a natural ordering, like Freshman, Sophomore, etc.), then a linear regression model would make some sense but is still not ideal.

Why not Linear Regression?

Additionally, if the ordering of the response variable is changed, the model estimates and predictions would be fundamentally different.

For example, a model trained with $y=1$ represents **Statistics** and $y=2$ represents **CS** is different from a model trained with the original ordering.

Why not Linear Regression?

- Consider a simpler problem where the response variable y has only two categories. Here, there is a natural ordering of the categories.

$$y = \begin{cases} 1, & \textit{has heart disease} \\ 0, & \textit{no heart disease} \end{cases}$$

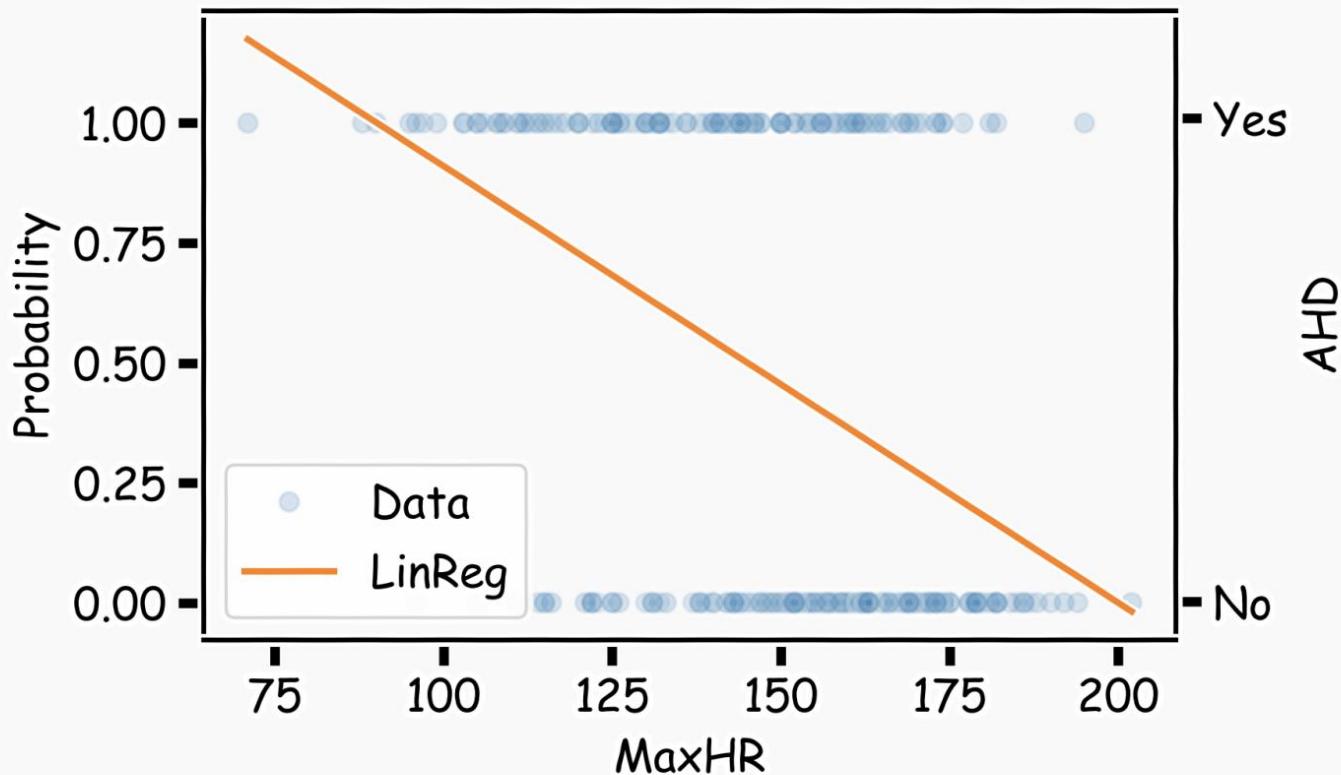
- Linear regression could be used to predict the **probability** $P(y = 1)$ directly from a set of predictors such as sex, cholesterol levels, etc.
- If $P(y = 1) \geq 0.5$, we could predict that the patient has heart disease and predict otherwise if $P(y = 1) < 0.5$.

Why not Linear Regression?



What could go wrong with this linear regression model?

Since this is modeling $P(y = 1)$, values for \hat{y} below 0 and above 1 would not make sense as a probability.

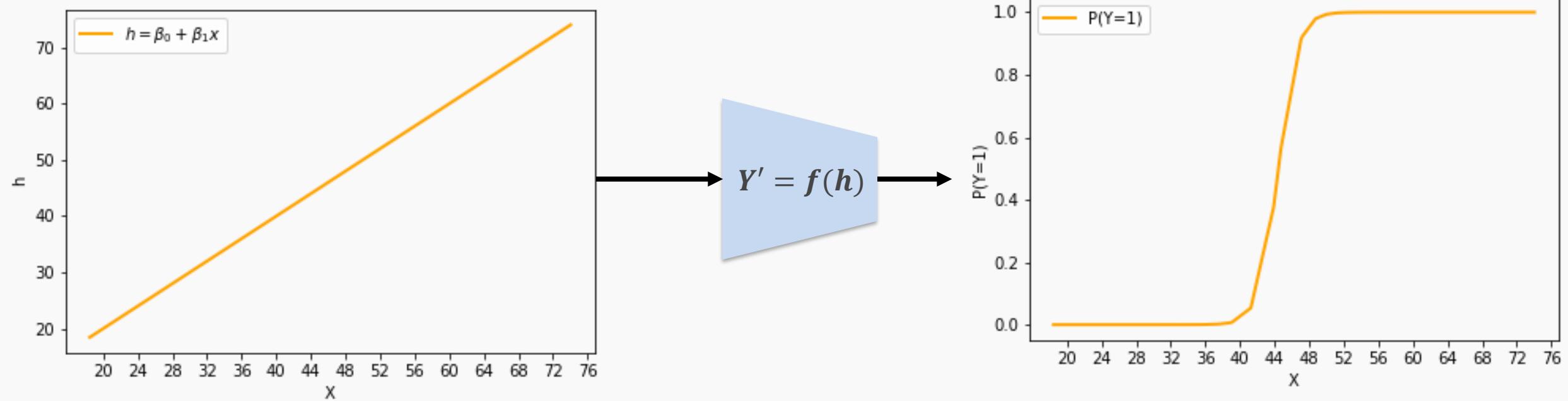


Outline

- What is Classification?
- Why not Linear Regression?
- **Estimating the Simple Logistic Model**
- Inference in Logistic Regression
- Multiple Logistic Regression
- Classification Decision Boundaries

What function should we use?

Now we know that linear regression yields values for probability that are larger than 1 or smaller than 0. So what can we do to fix this?



What function should we use?

We can use the **sigmoid function**:

$$h = \beta_0 + \beta_1 X \longrightarrow p = \frac{1}{1 + e^{-h}} \longrightarrow P(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

Logistic Regression

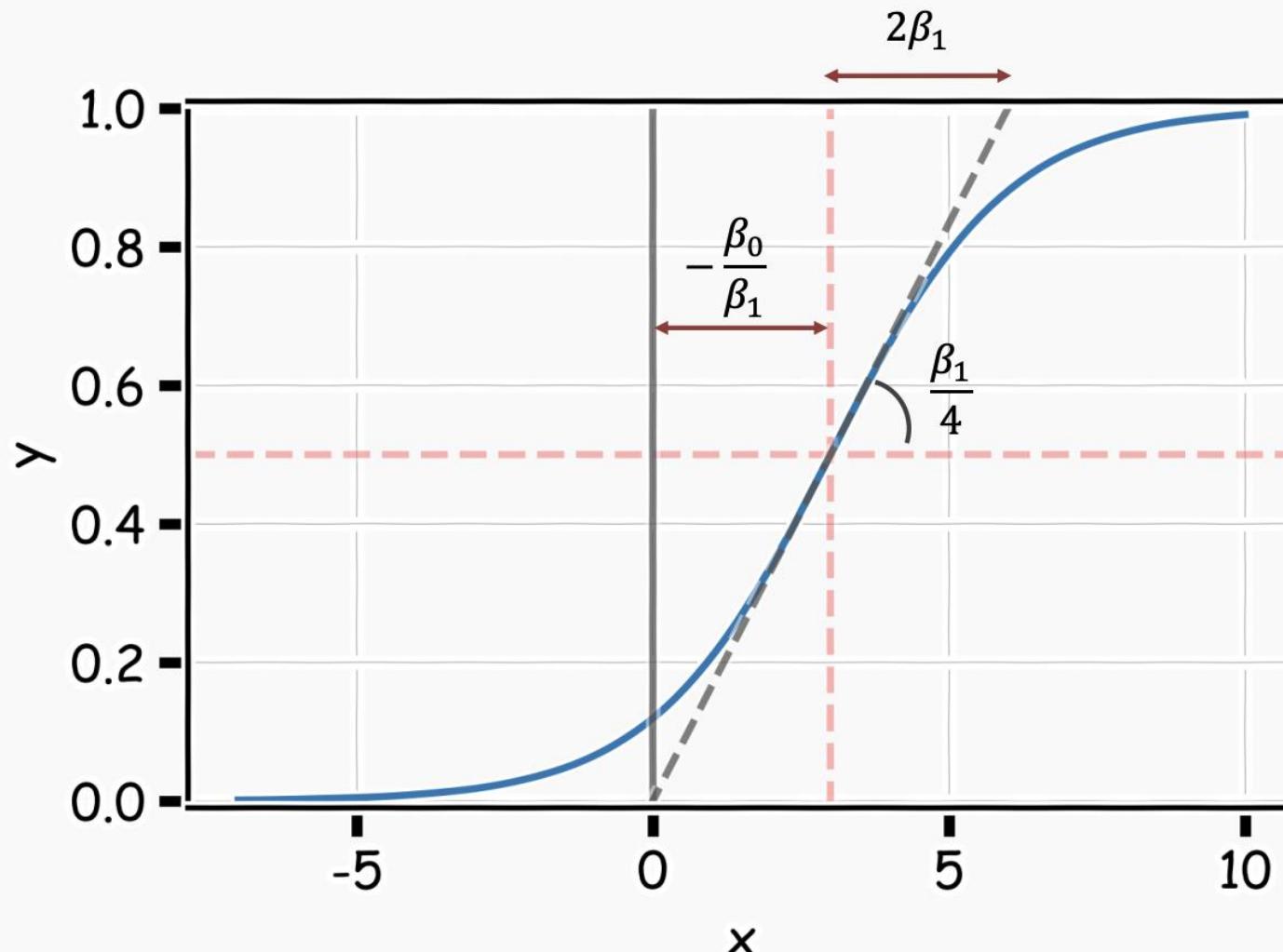
- Logistic Regression addresses the problem of estimating a probability, $P(y = 1)$, to be outside the range of [0,1].
- The logistic regression model uses a function, called the **logistic** function, to model $P(y = 1)$:

$$P(Y = 1) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

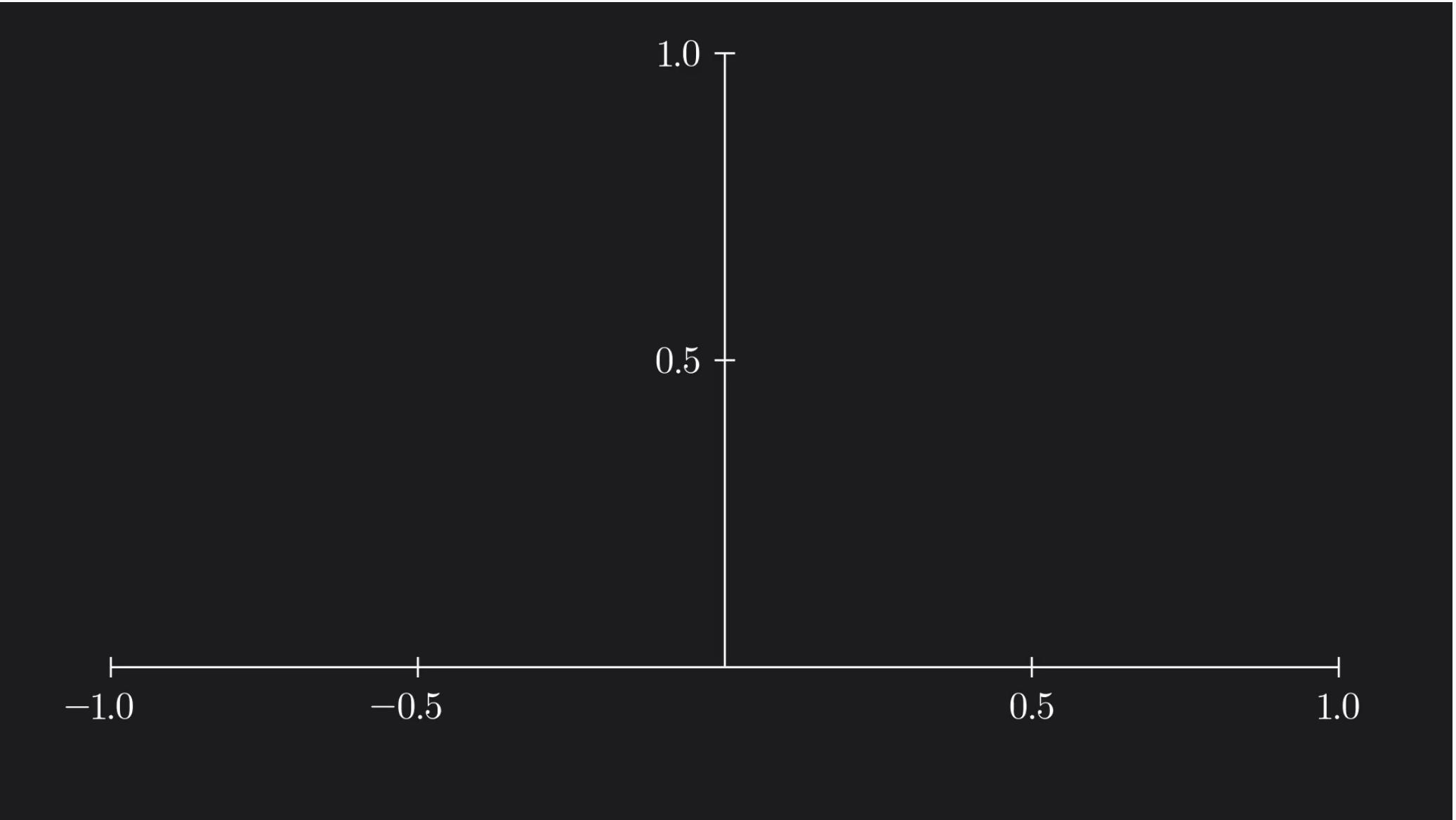
What are the parameters of this model?

Logistic Regression

The coefficients β_0 and β_1 now control the shape of this *S*-shaped curve.



Sigmoid Animation



Interpreting the Coefficients

With a little bit of algebraic work, the logistic model can be rewritten as:

$$\ln\left(\frac{P(Y = 1)}{1 - P(Y = 1)}\right) = \beta_0 + \beta_1 X$$


odds

Logistic regression is said to model the ***log-odds*** with a linear function of the predictors or features, X .

A one unit change in X is associated with a β_1 change in the log-odds of $P(Y = 1)$; or better yet, a one unit change in X is associated with an e^{β_1} change in the odds that $Y = 1$.

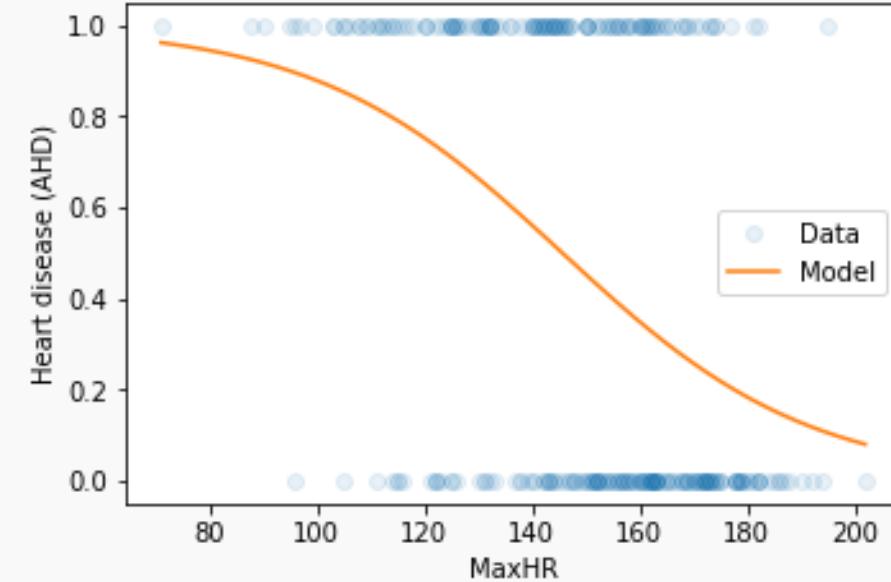
A First Logistic Regression Model in sklearn

Here is a logistic regression output to predict $Y = \text{AHD}$ from $X = \text{MaxHR}$:

```
logreg = LogisticRegression(penalty='none')
logreg.fit(df_heart[['MaxHR']], df_heart['AHD'])

print('Estimated beta1: \n', logreg.coef_)
print('Estimated beta0: \n', logreg.intercept_)
```

```
Estimated beta1:
 [[-0.04341112]]
Estimated beta0:
 [6.3249492]
```



What is the estimated model? What are the interpretations of the $\hat{\beta}$ s?

$$\ln\left(\frac{\hat{P}(Y = 1)}{1 - \hat{P}(Y = 1)}\right) = 6.325 - 0.0434(\text{MaxHR})$$

Estimating the Simple Logistic Model

Estimating parameter coefficients

Unlike in linear regression where there exists a closed-form solution to finding the estimates, $\hat{\beta}_j$'s, **that minimize the loss function**, logistic regression estimates cannot be calculated through simple matrix multiplication.

Questions:

- In linear regression what loss function was used to determine the parameter estimates?

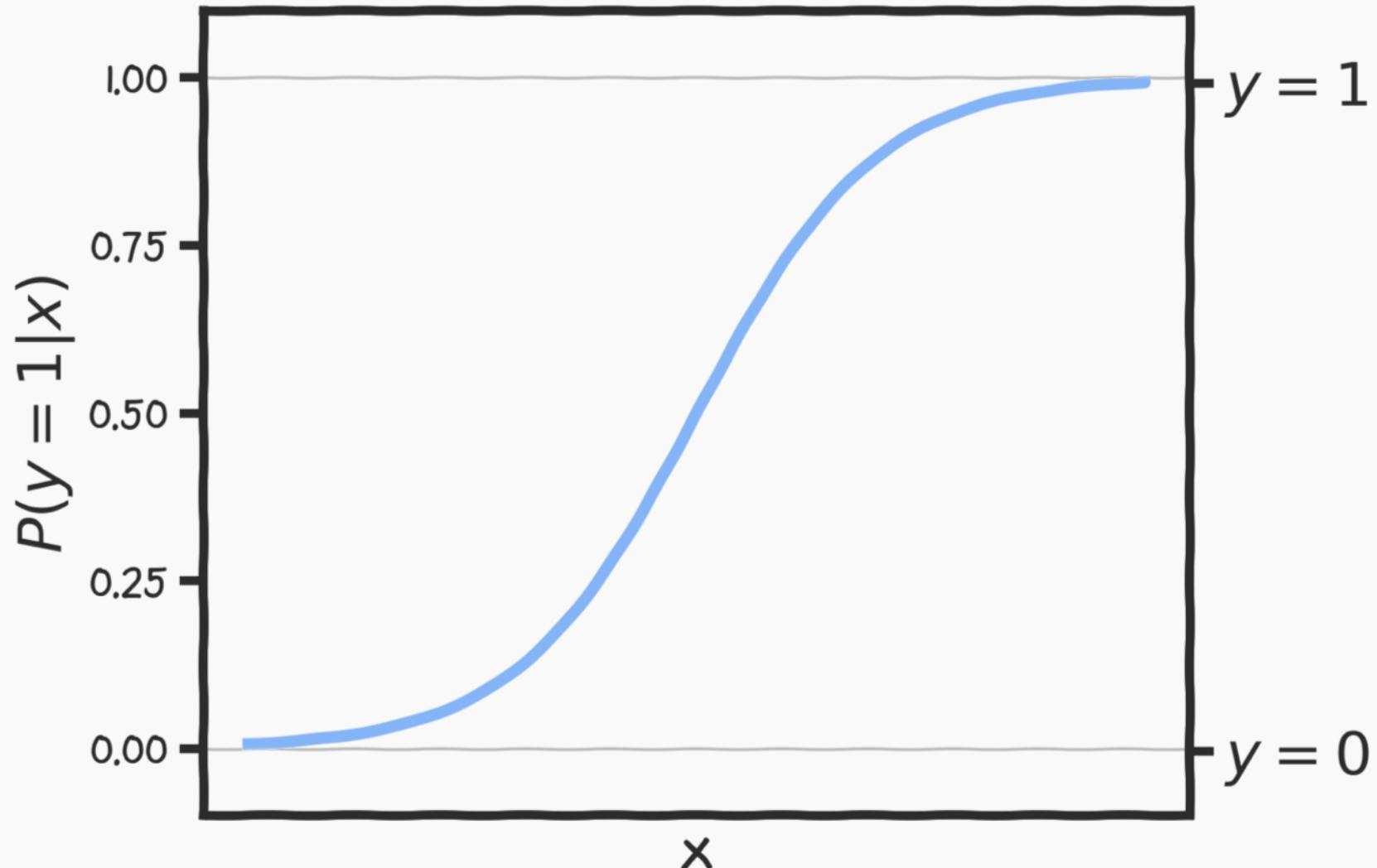
MSE

- What was the probabilistic perspective on linear regression?

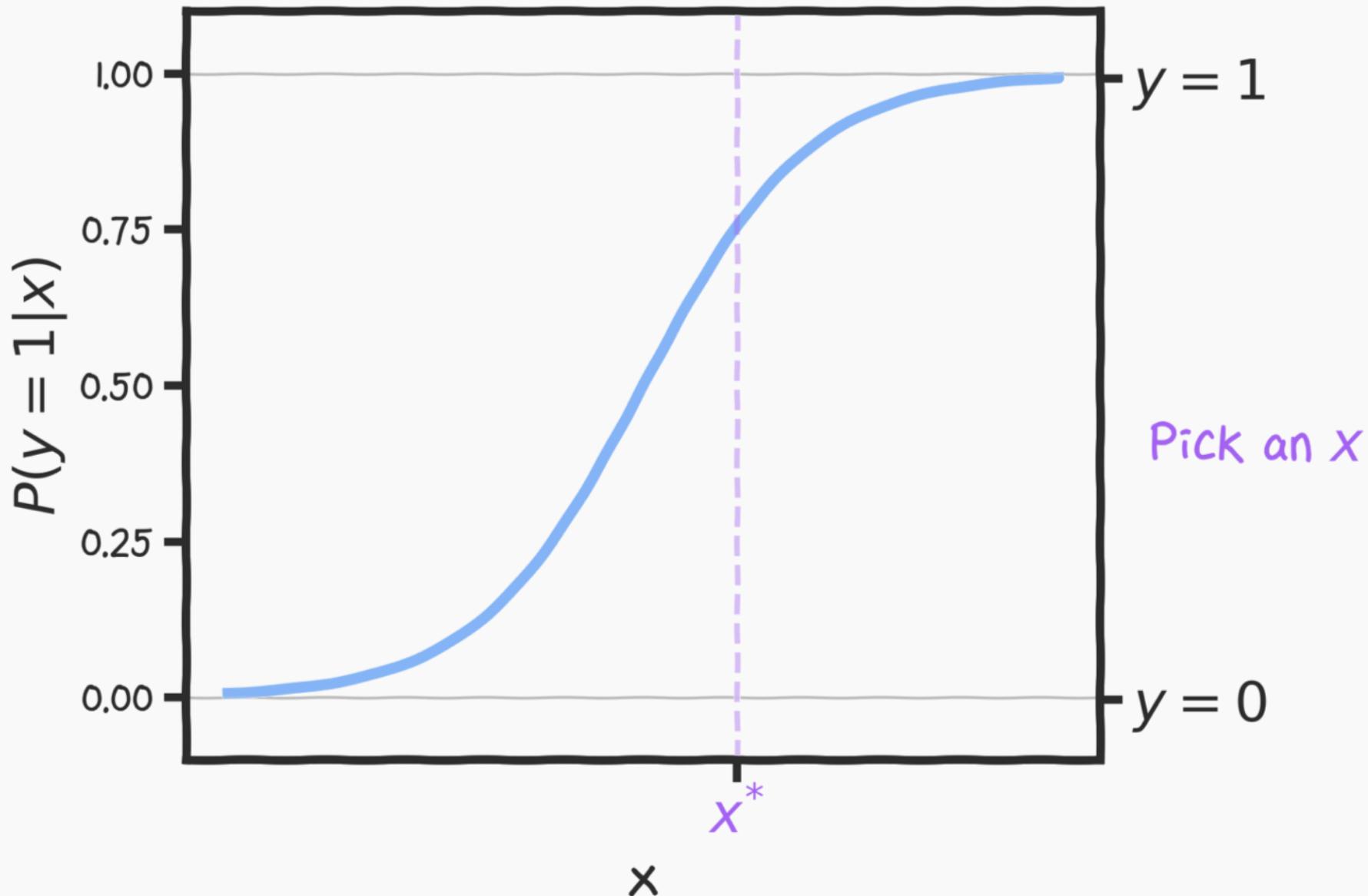
Normal Distribution

Logistic Regression also has a likelihood-based approach to estimating parameter coefficients.

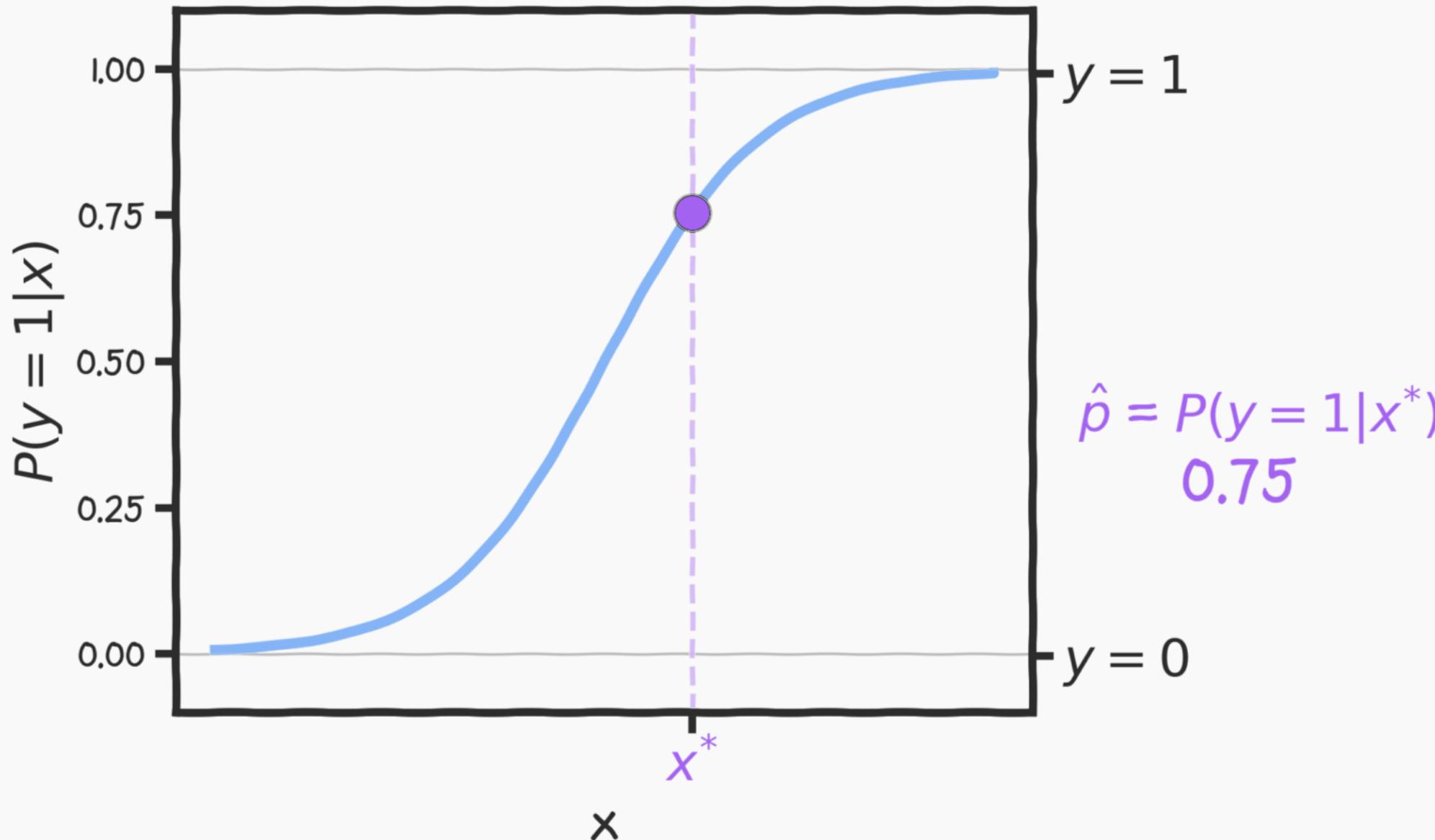
Logistic Regression



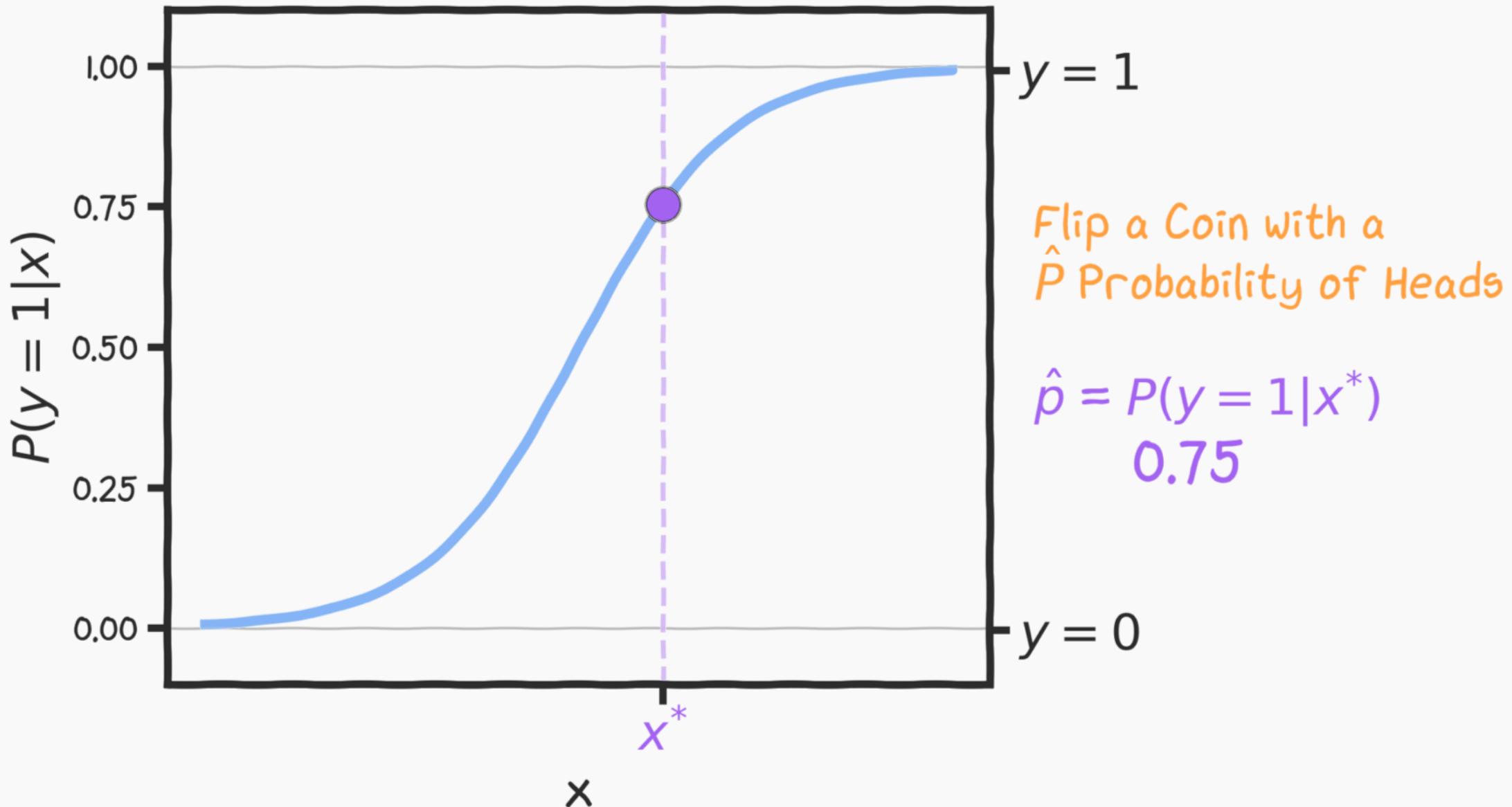
Logistic Regression



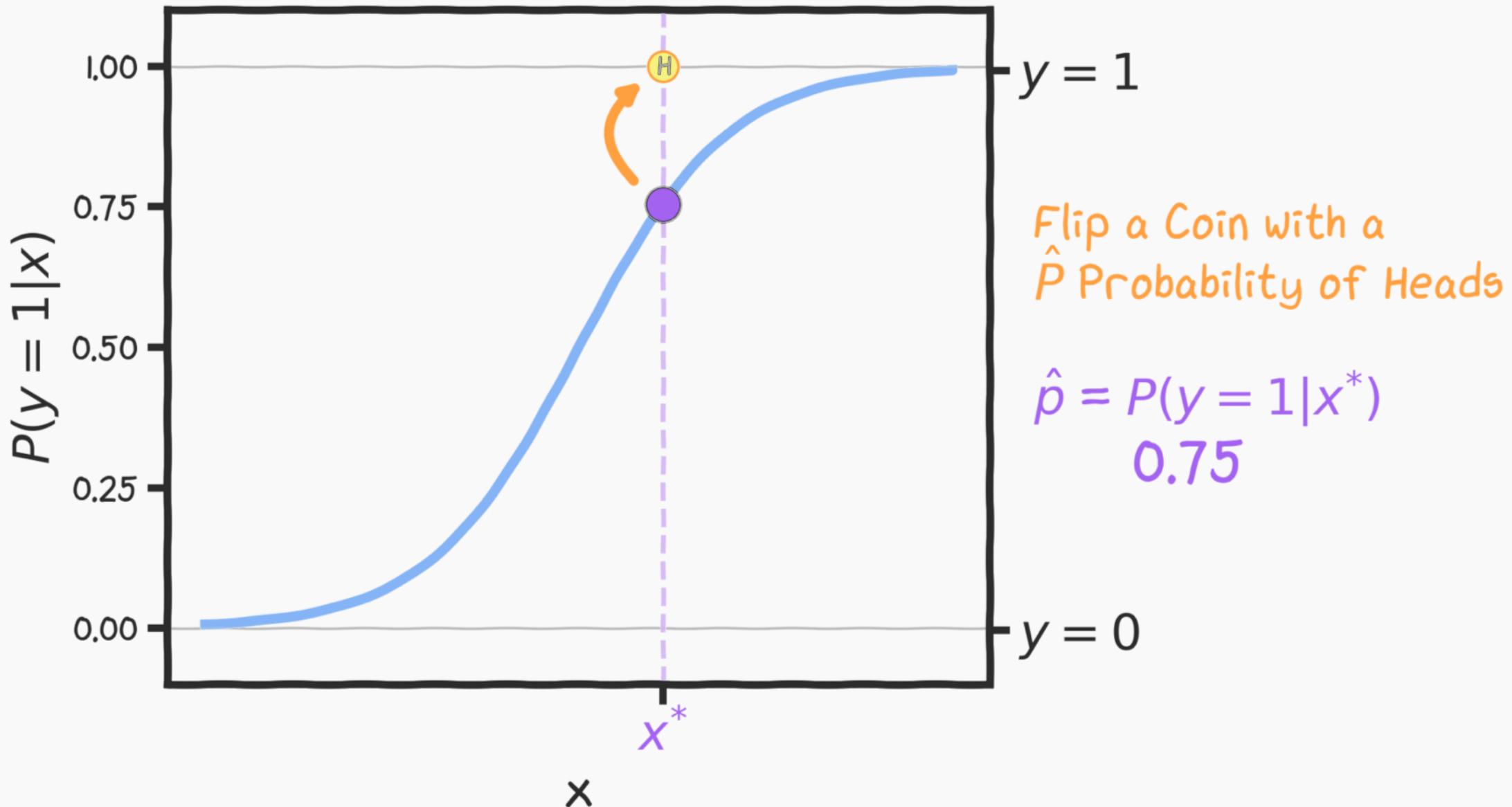
Logistic Regression



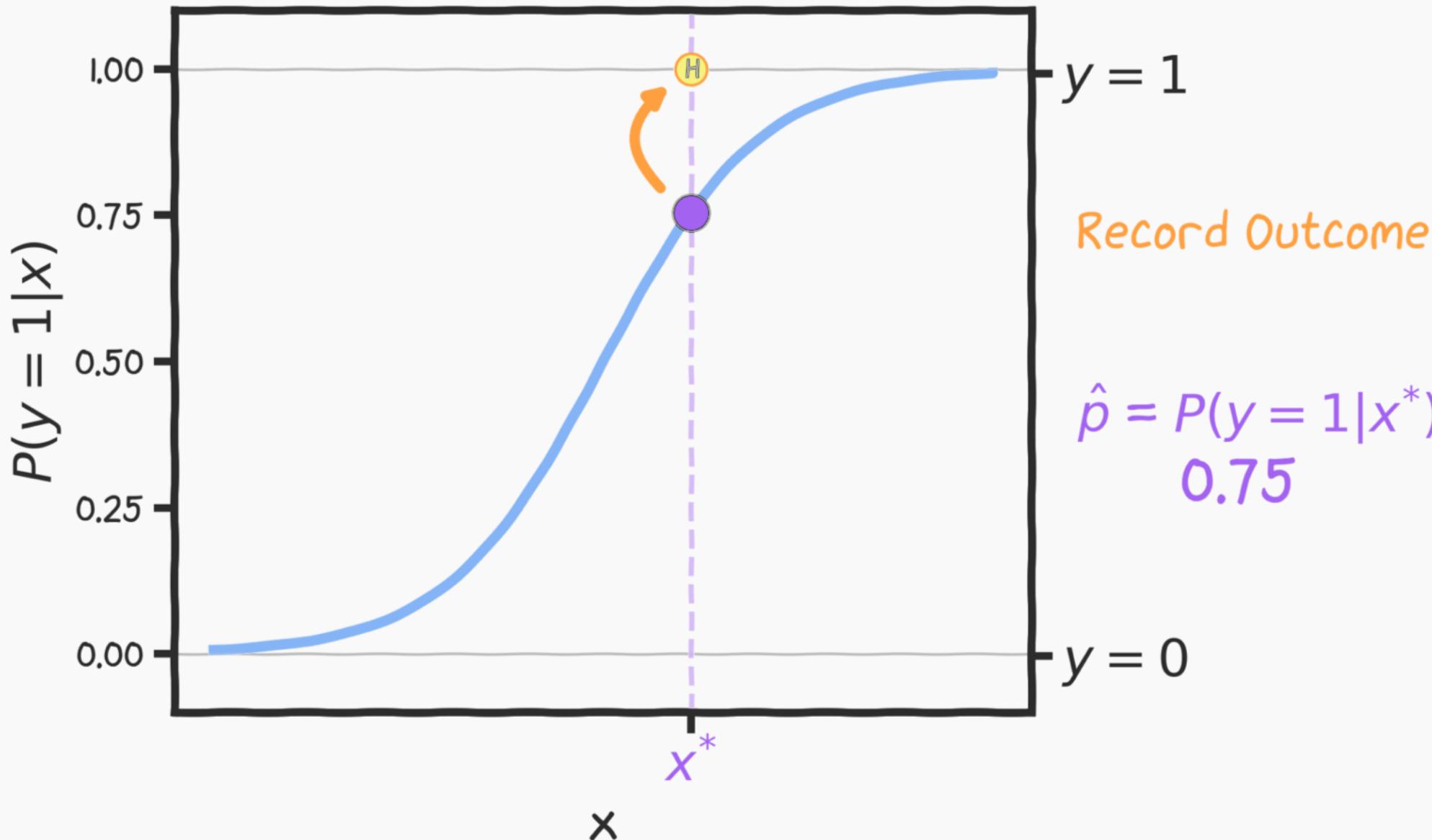
Logistic Regression



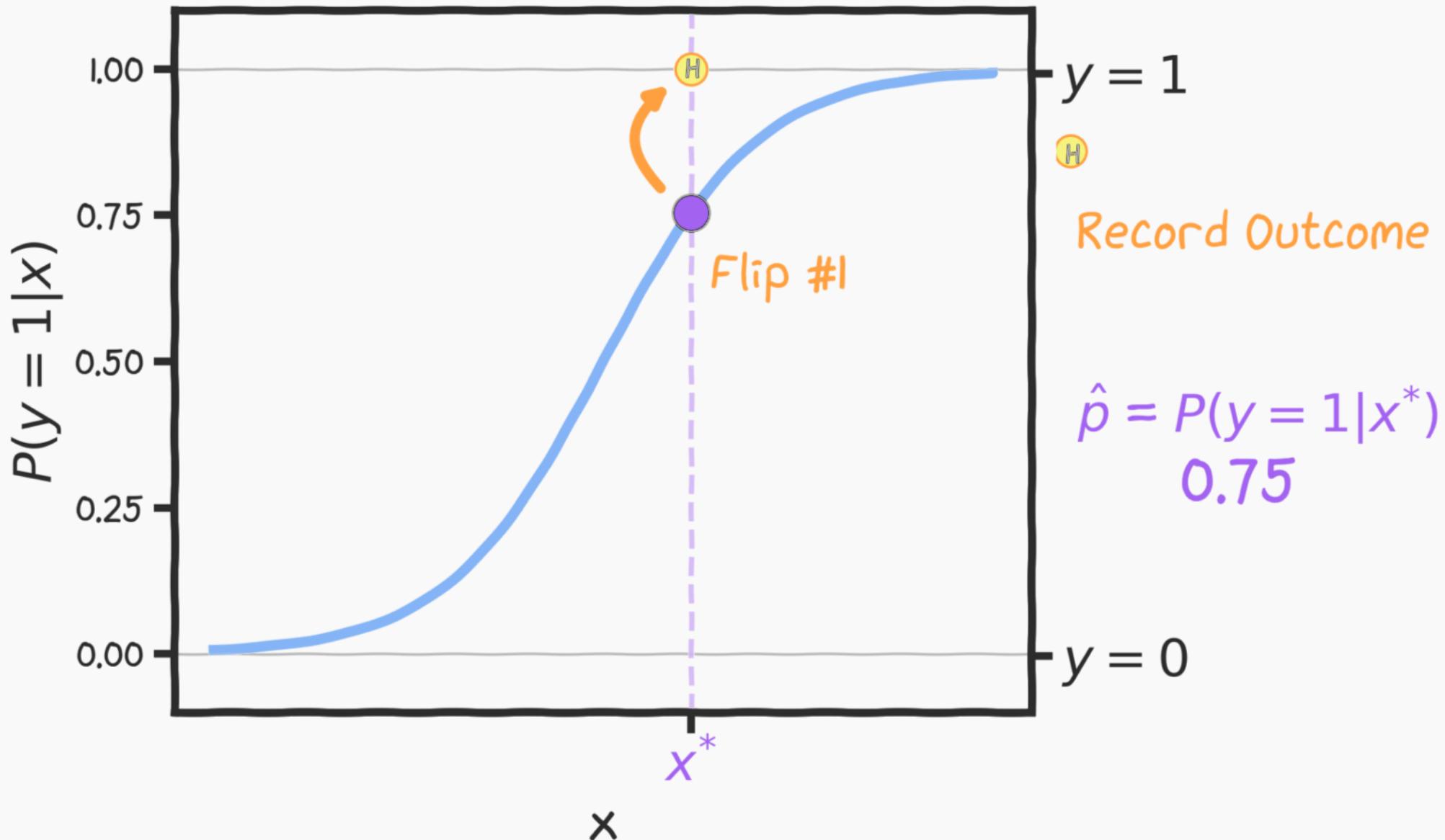
Logistic Regression



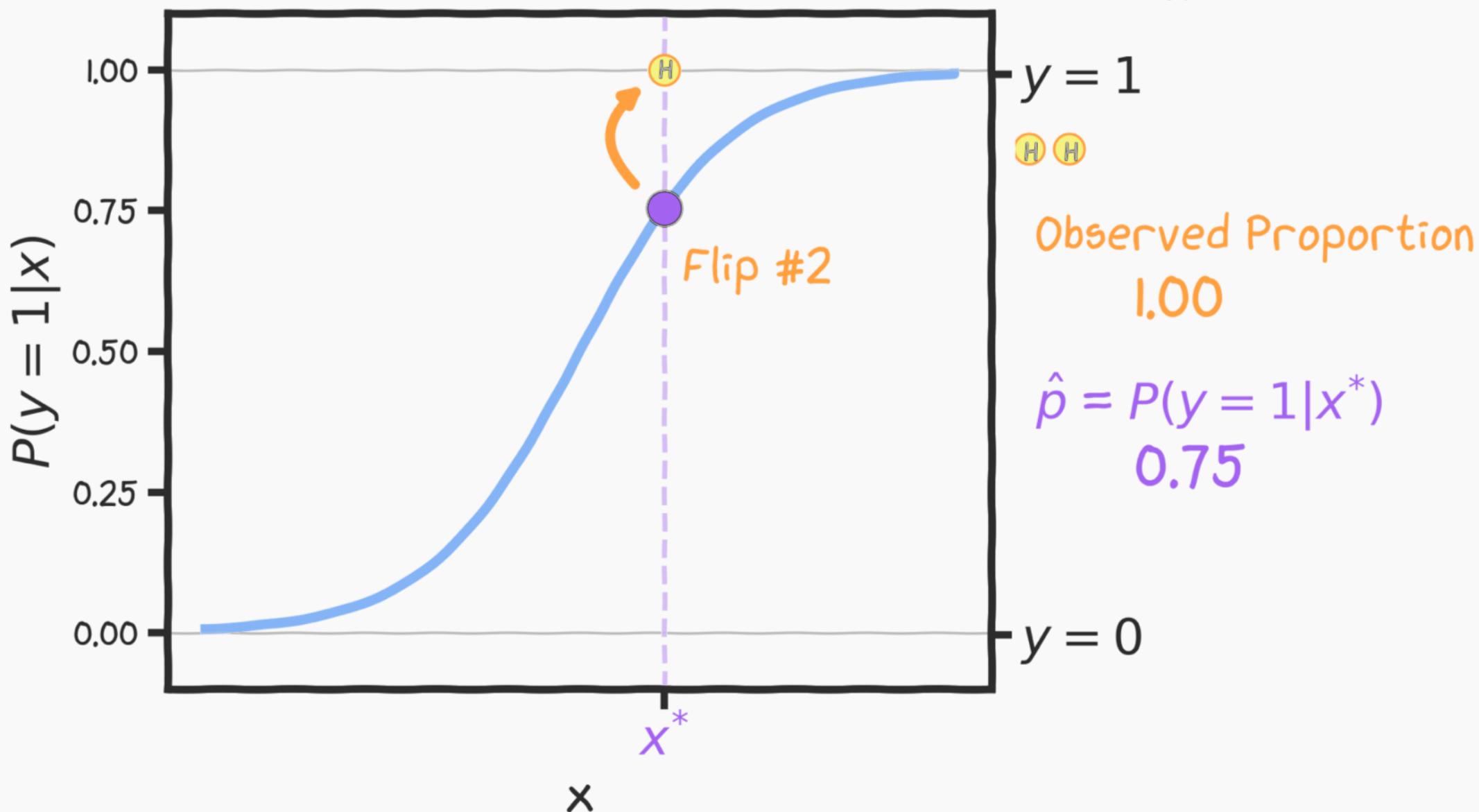
Logistic Regression



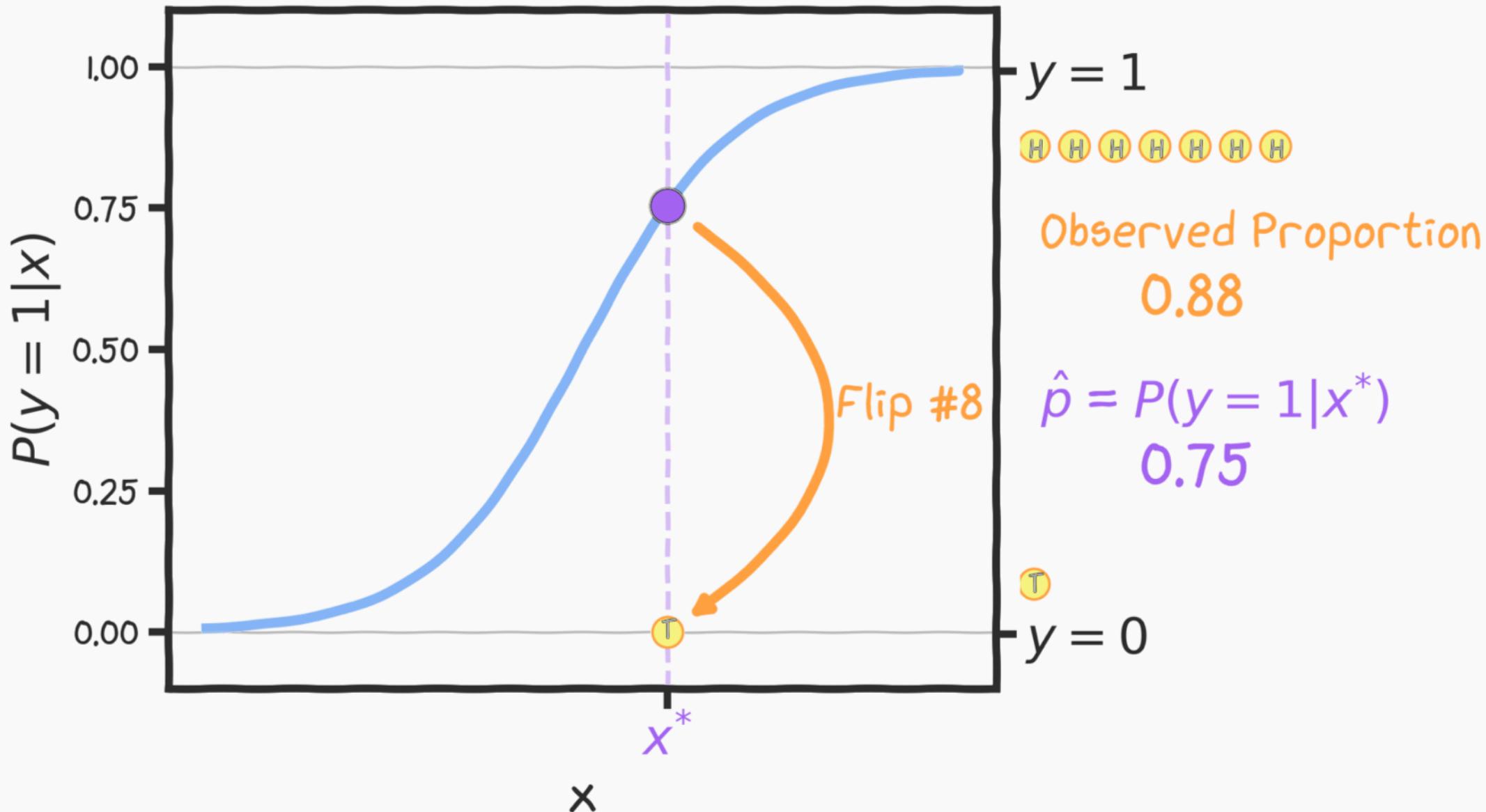
Logistic Regression



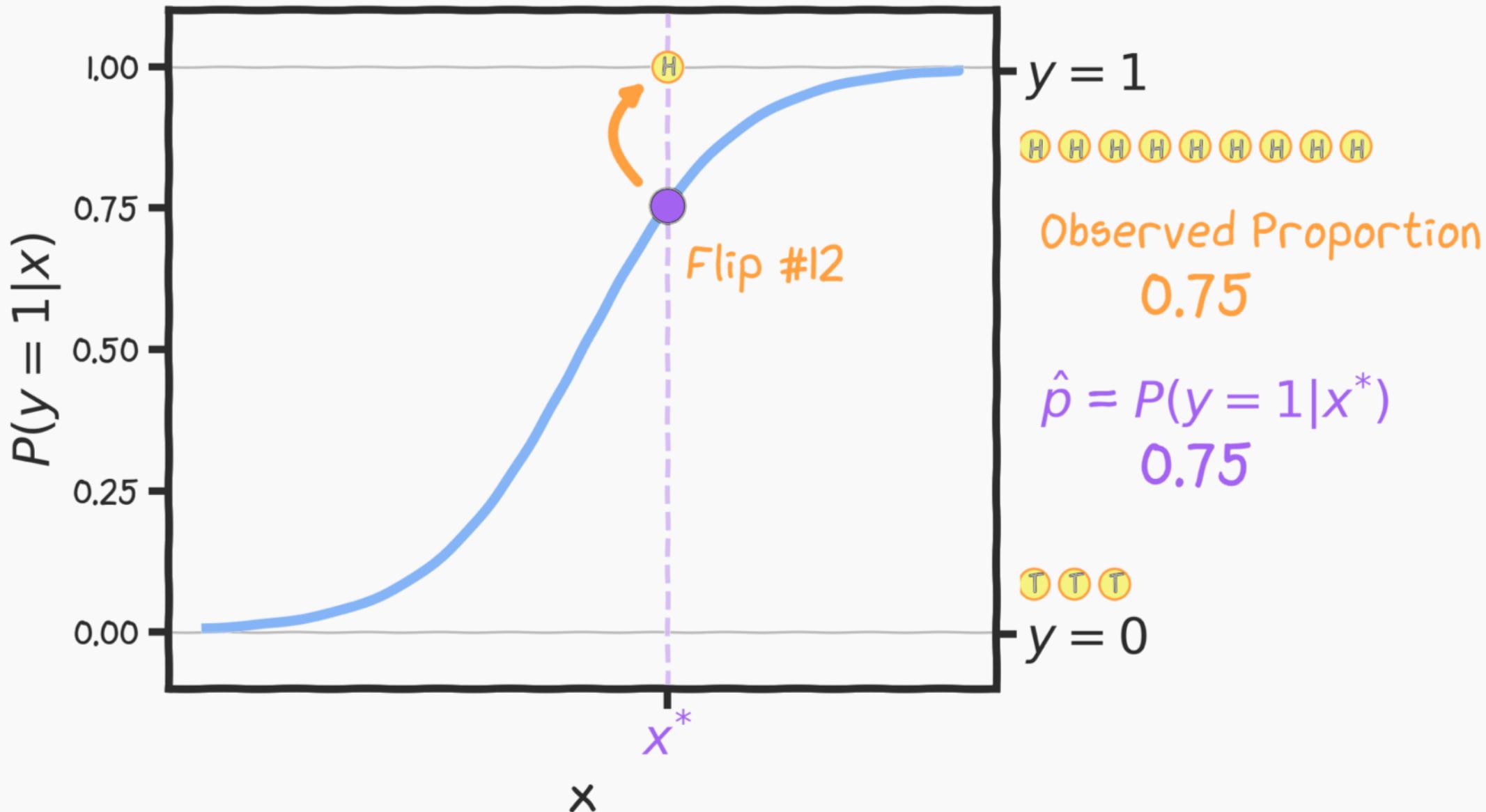
Logistic Regression



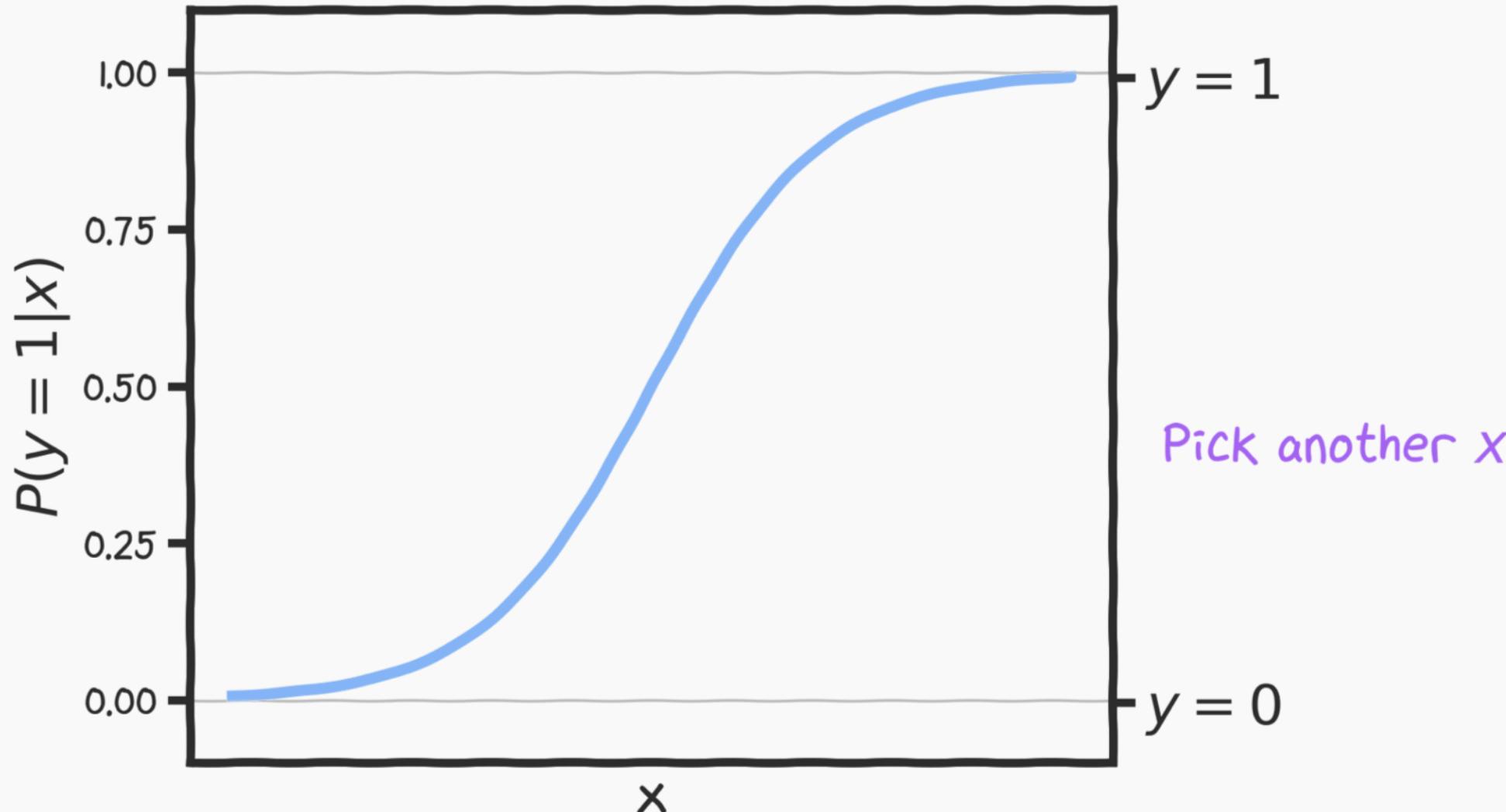
Logistic Regression



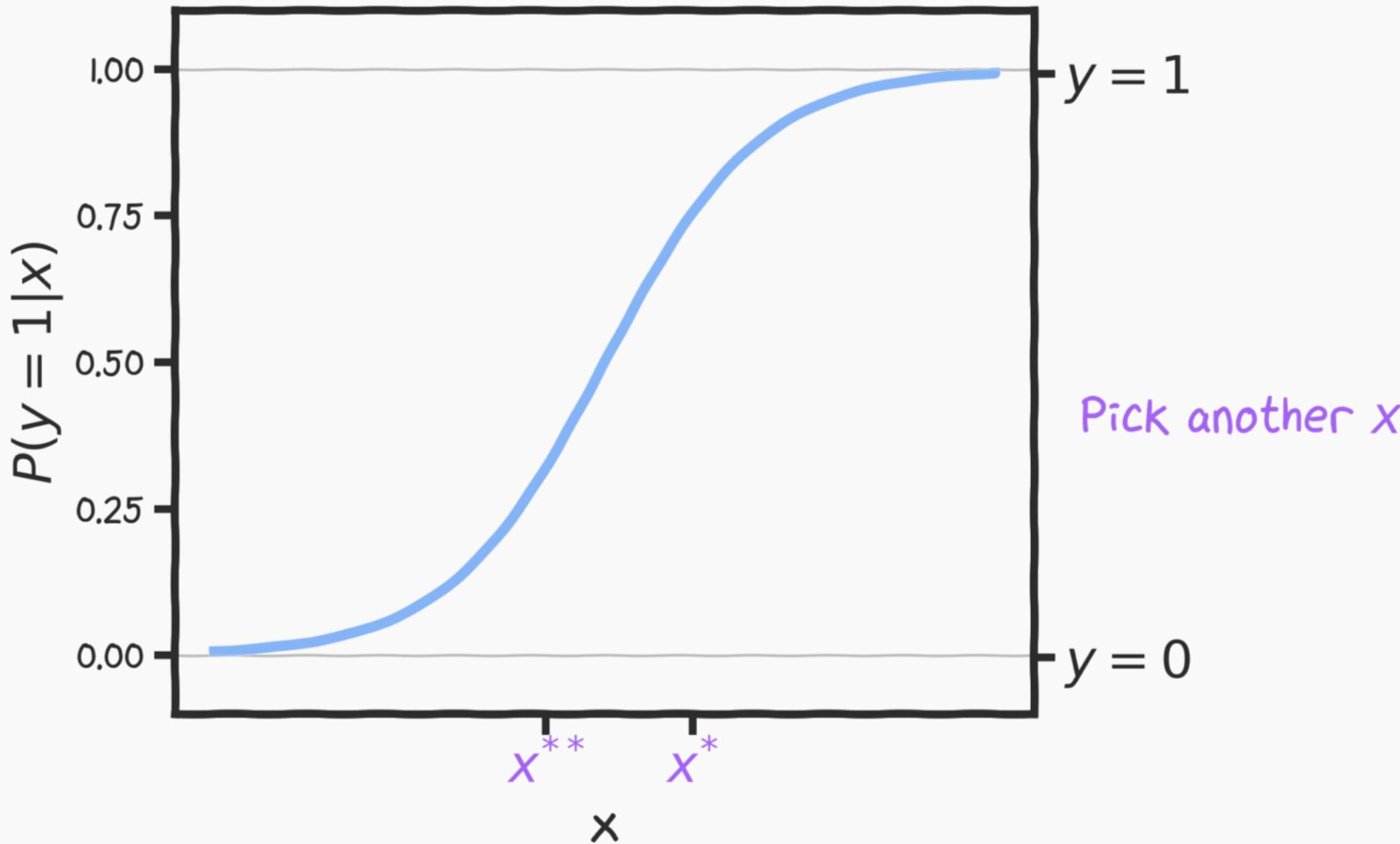
Logistic Regression



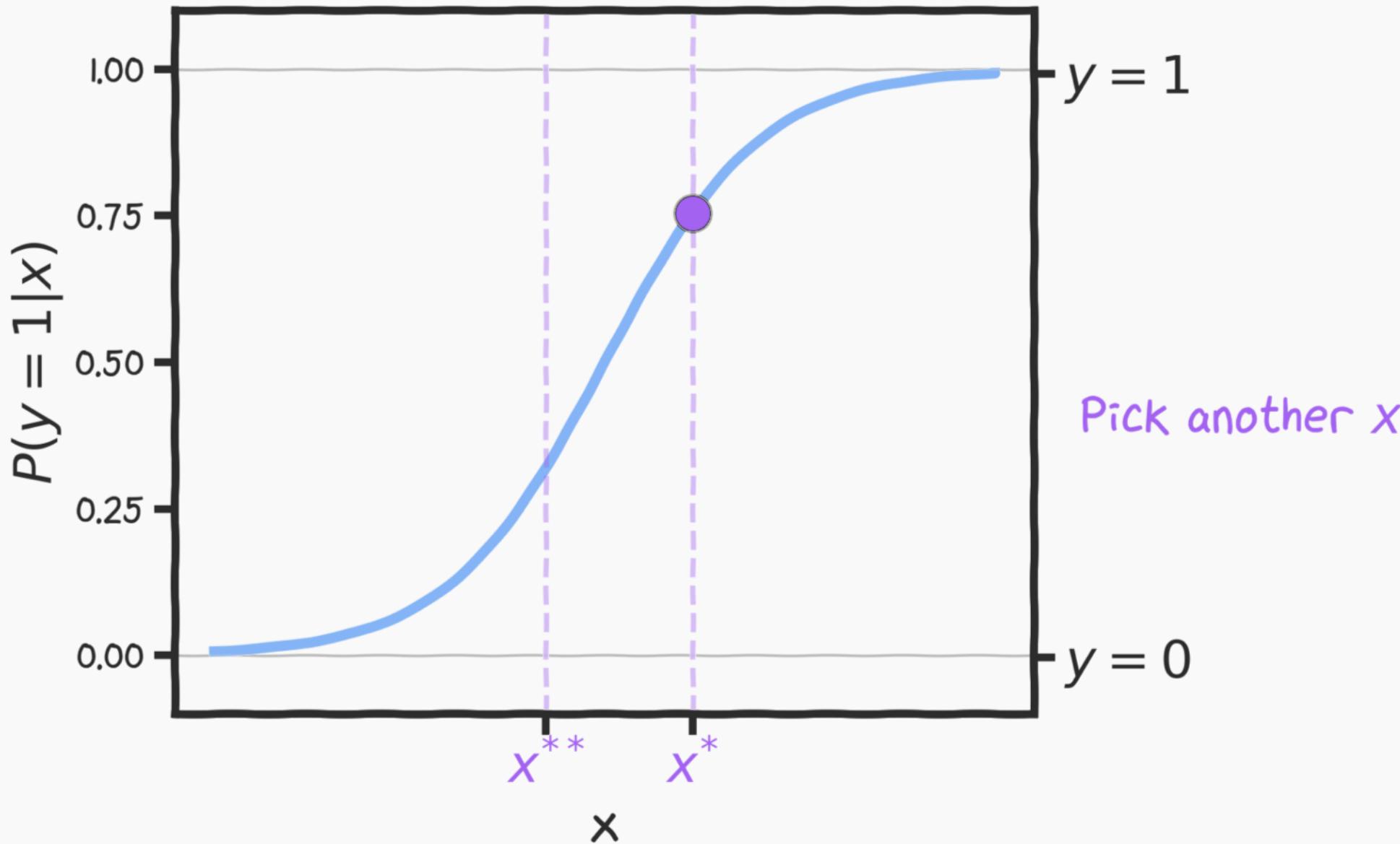
Logistic Regression



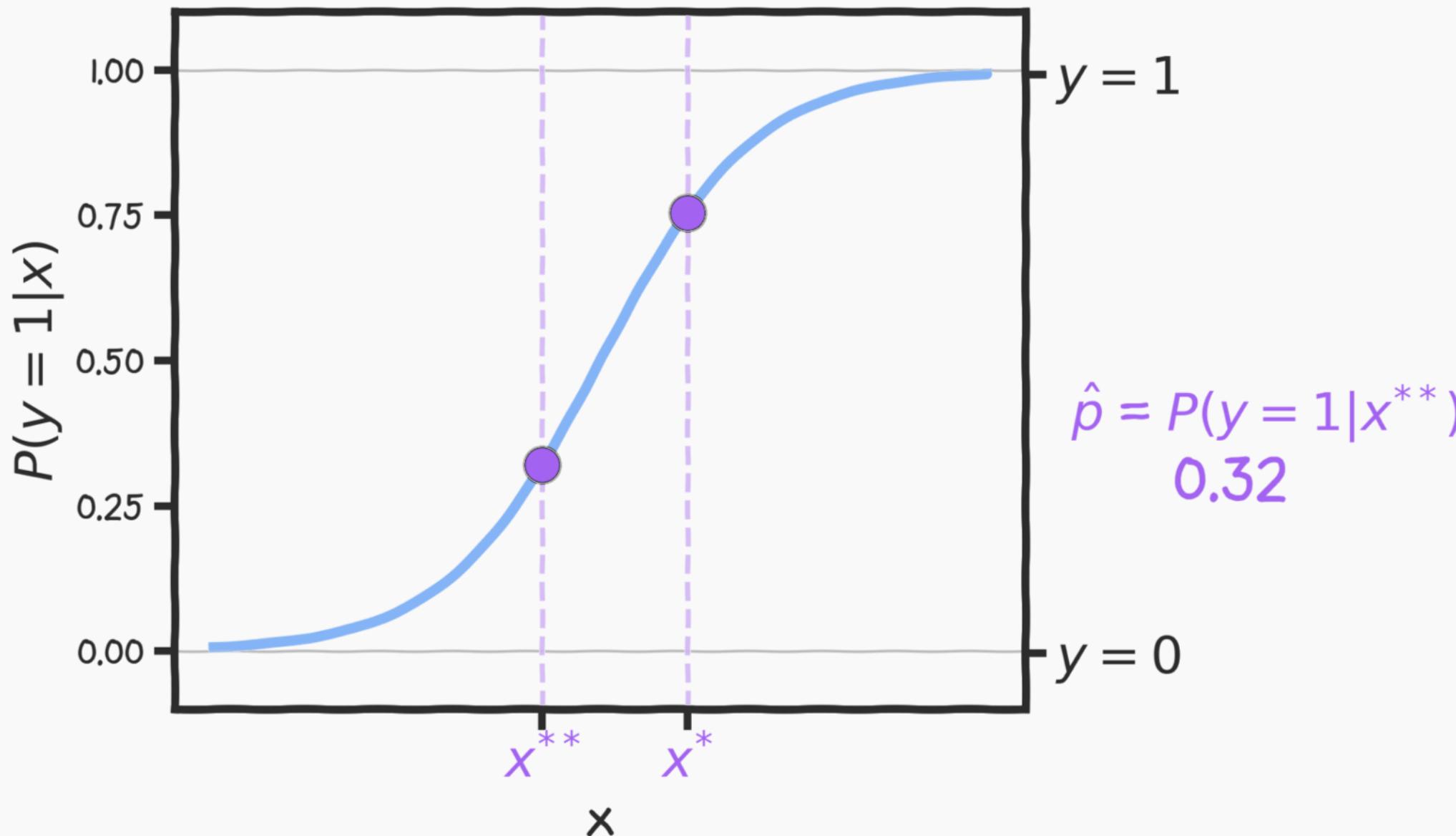
Logistic Regression



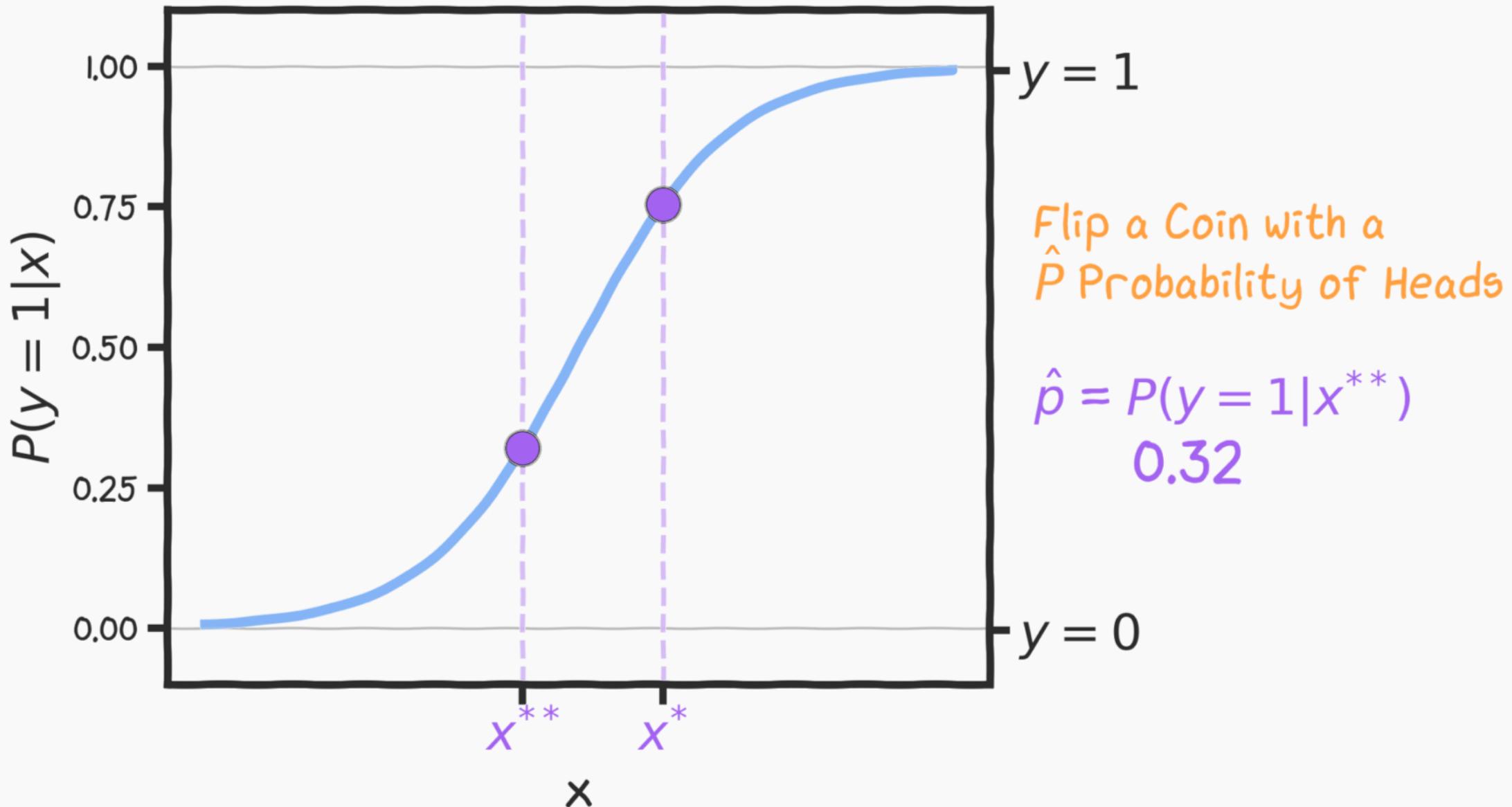
Logistic Regression



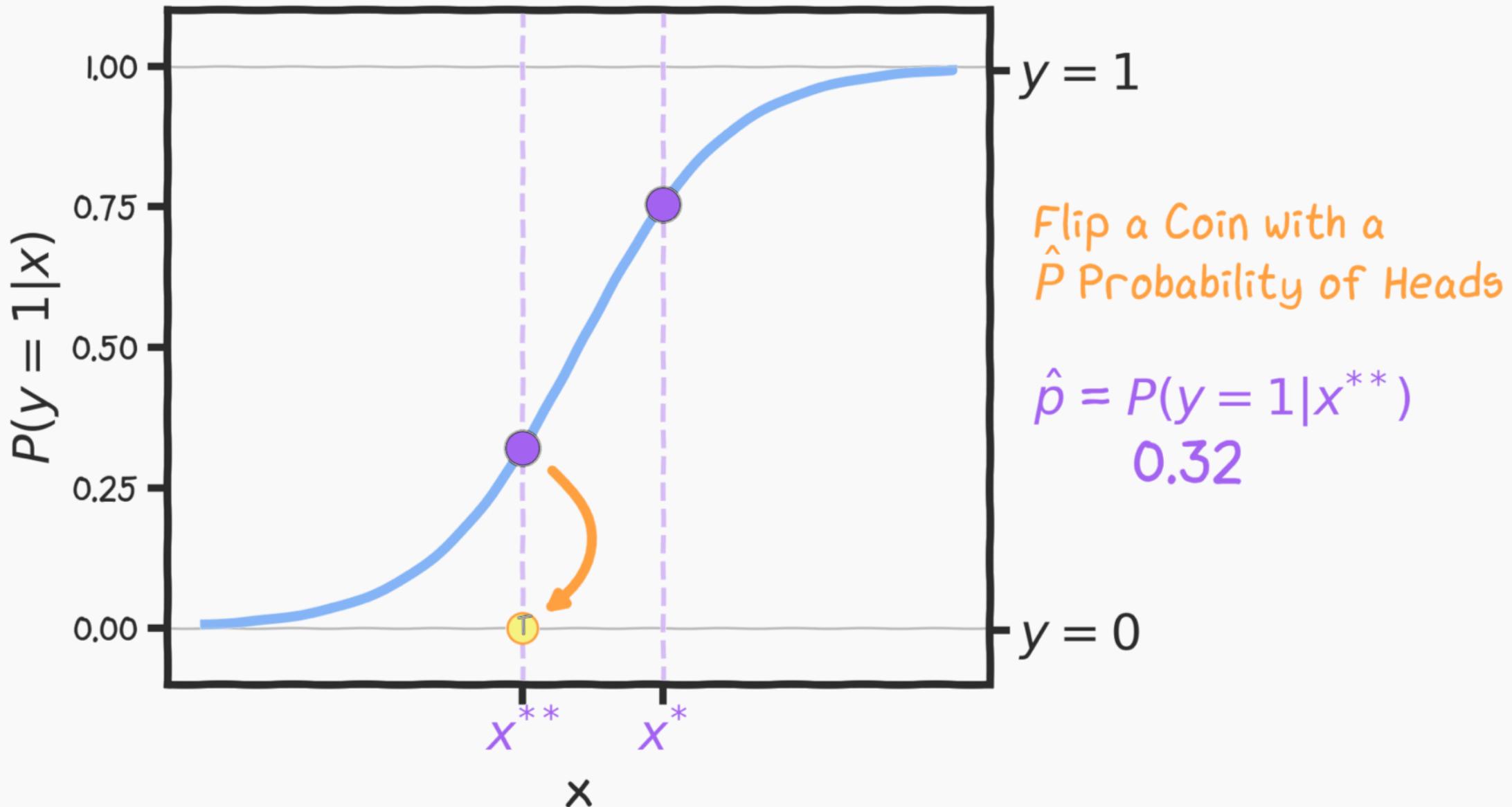
Logistic Regression



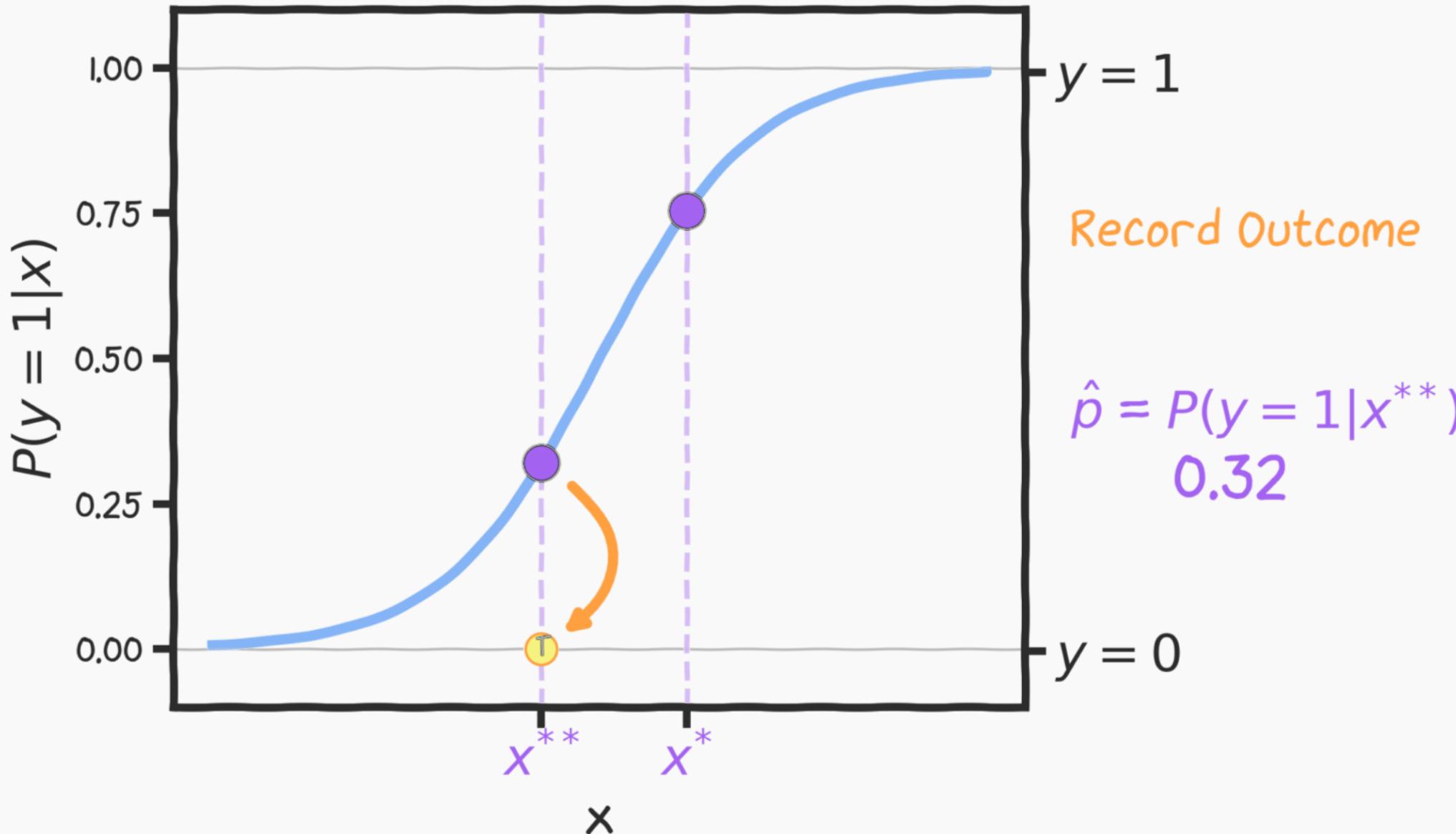
Logistic Regression



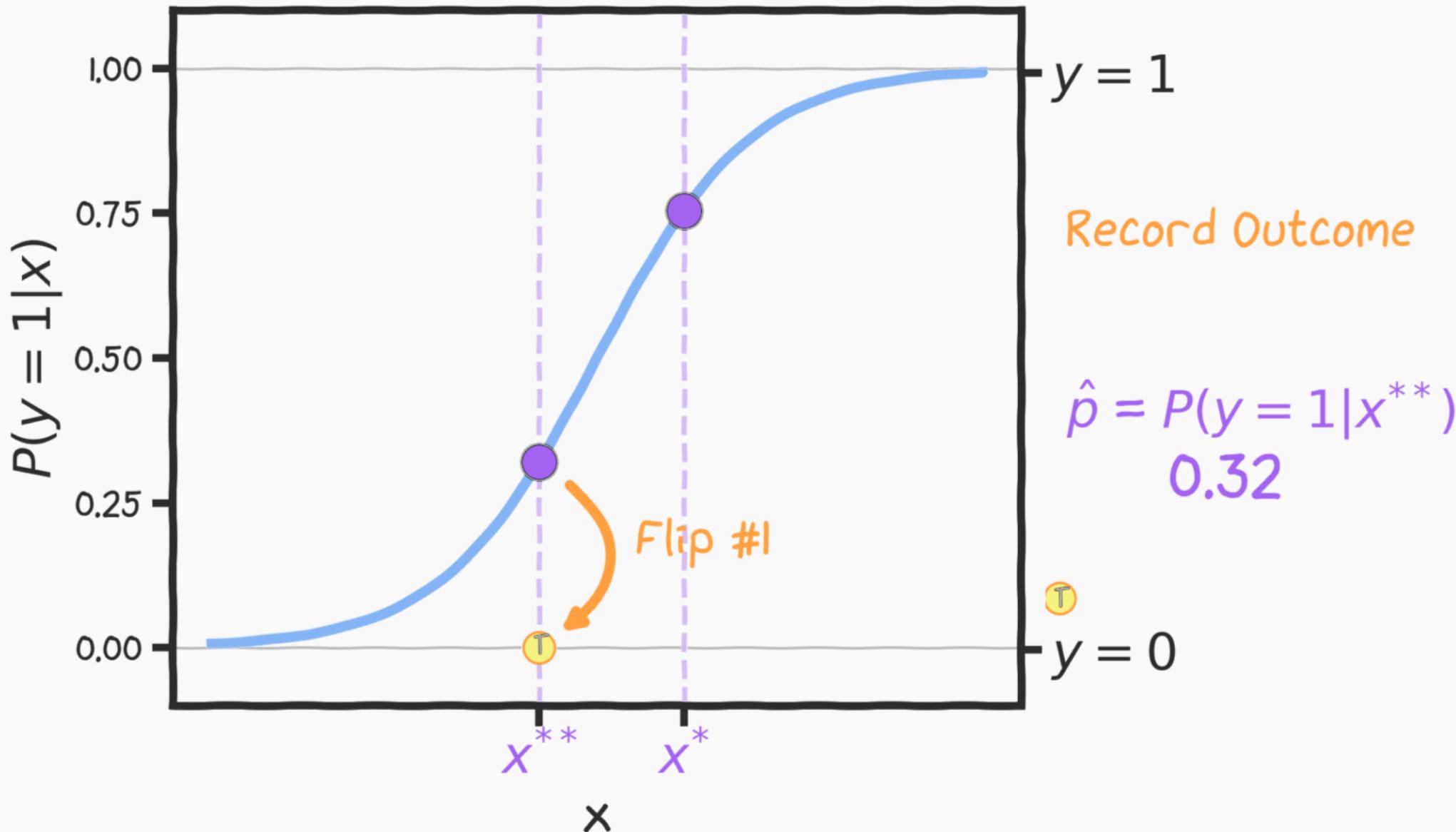
Logistic Regression



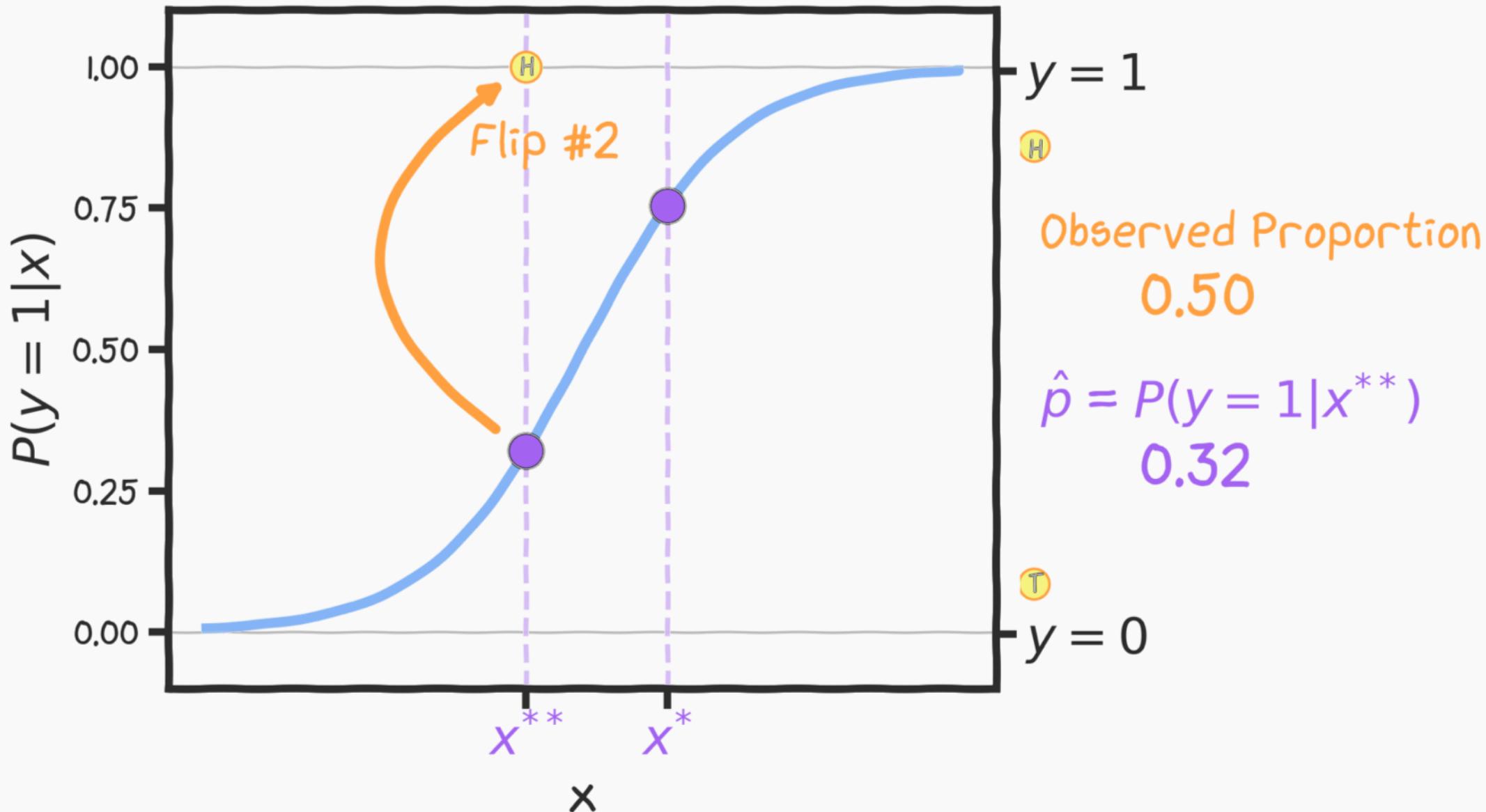
Logistic Regression



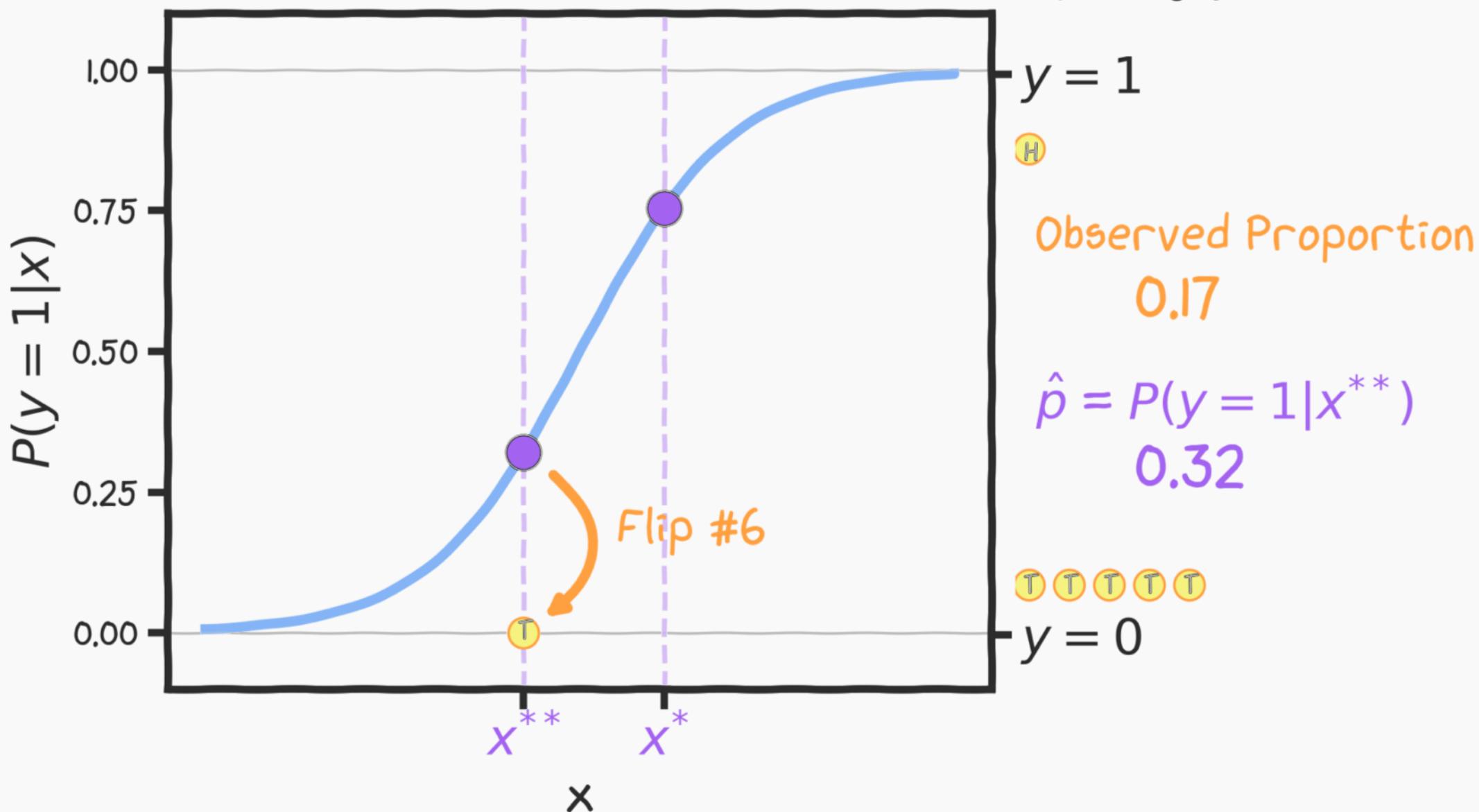
Logistic Regression



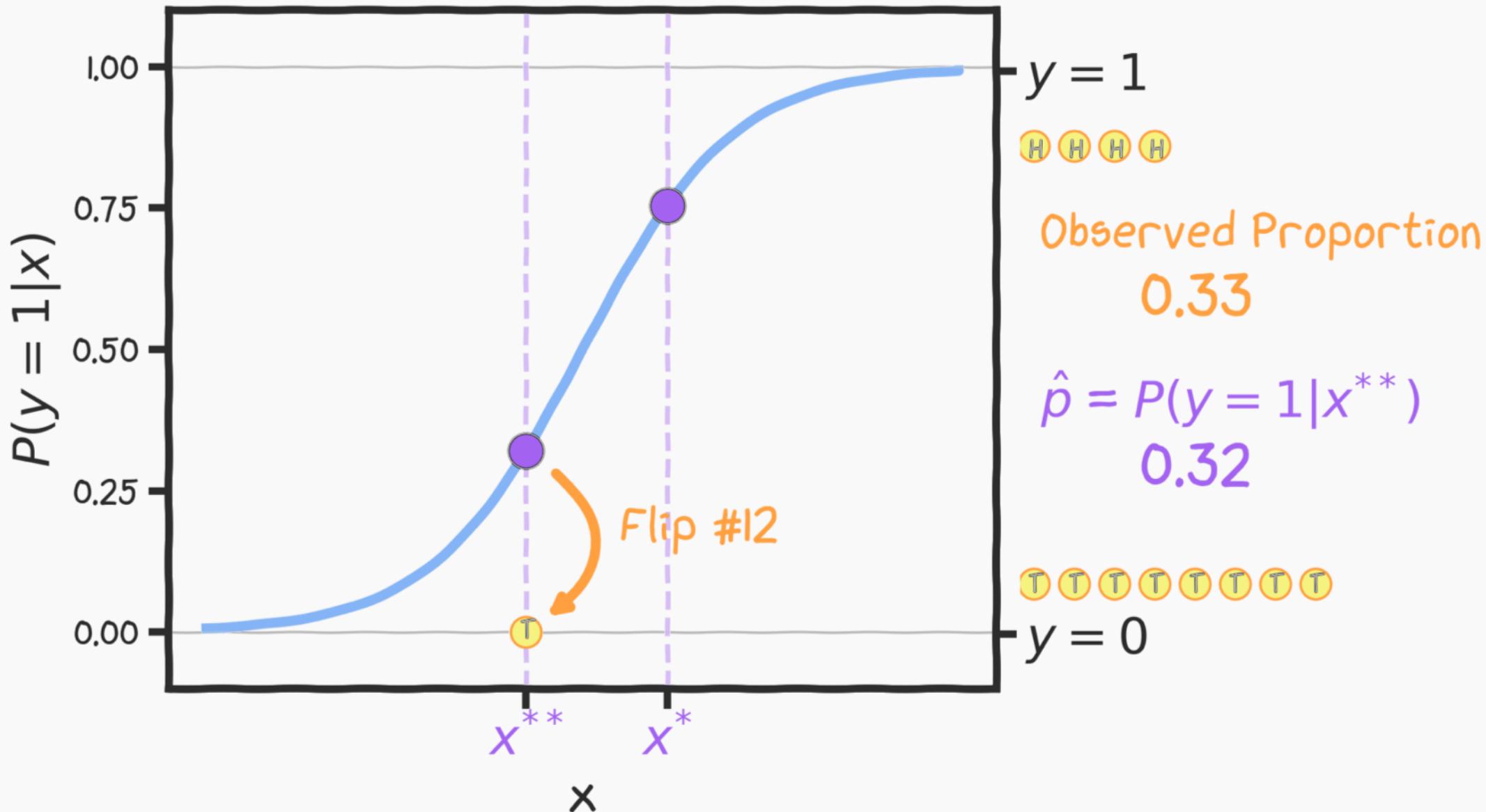
Logistic Regression



Logistic Regression

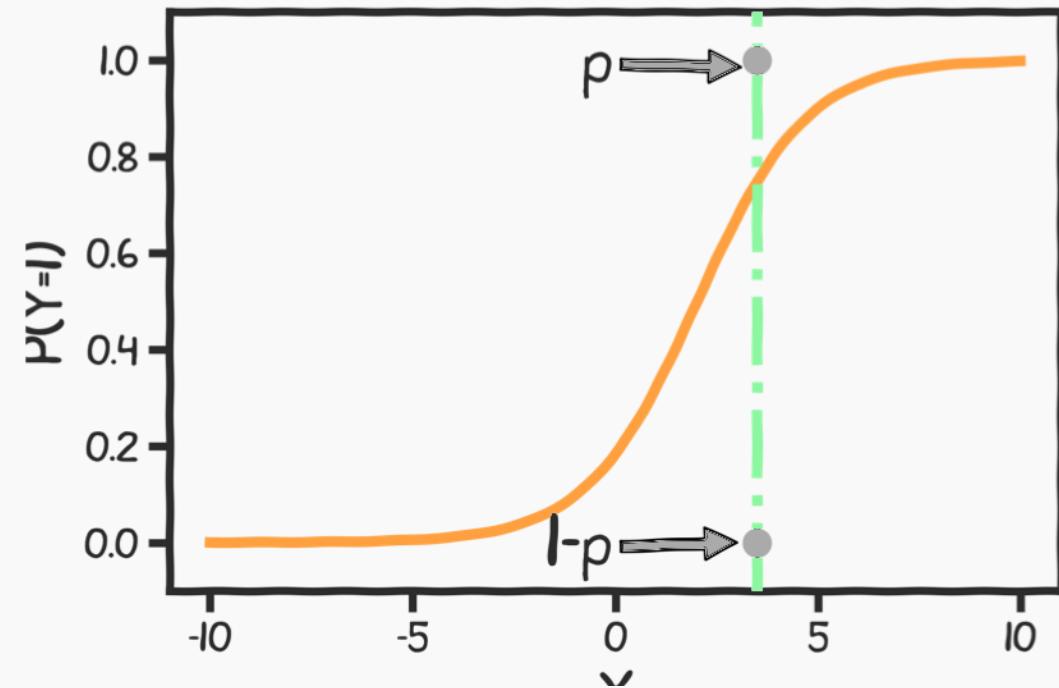


Logistic Regression



Estimating the Simple Logistic Model

For any X , the probability of getting heads or tails (1 or 0) is given by the logistic function shown in the plot as an orange line.



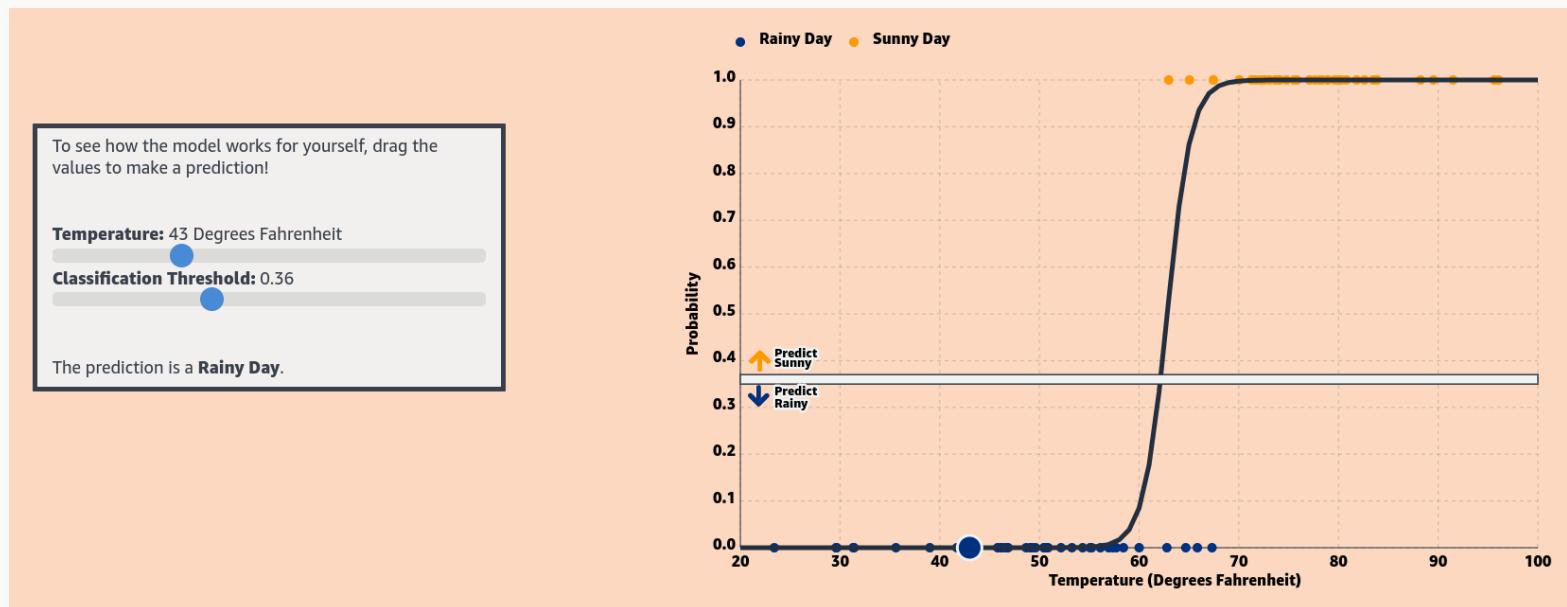
$$\left. \begin{array}{l} \text{Prob. } Y = 1: \quad P(Y = 1) = p \\ \text{Prob. } Y = 0: \quad P(Y = 0) = 1 - p \end{array} \right\} P(Y = y) = p^y(1 - p)^{(1-y)}$$

where $p = P(Y = 1|X = x)$ and therefore p depends on X .
Thus, not every p is the same for each individual measurement.

Side Note: MLU-Explain

MLU-Explain from Amazon hosts many interactive visual explanations of core machine learning concepts. Their page for logistic regression is a great resource!

<https://mlu-explain.github.io/logistic-regression/>



Estimating the Simple Logistic Model

The likelihood of a single observation for p given x and true label y is:

$$L(p_i|Y_i) = P(Y_i = y_i) = p_i^{y_i} (1 - p_i)^{1-y_i}$$

Assuming the observations are independent, the total probability (or likelihood $L(p_i|Y_i)$) of getting a certain outcome will be the product of all individual likelihoods.

$$L(p|Y) = \prod_i P(Y_i = y_i) = \prod_i p_i^{y_i} (1 - p_i)^{1-y_i}$$

Estimating the Simple Logistic Model

$$L(p|Y) = \prod_i P(Y_i = y_i) = \prod_i p_i^{y_i} (1 - p_i)^{1-y_i}$$

The **best model** is the model that gives us the **argmax $L(p|Y)$** .

We can adjust the β values such as we get the **argmax $L(p|Y)$** for our data .

Estimating the Simple Logistic Model

$$L(p|Y) = \prod_i P(Y_i = y_i) = \prod_i p_i^{y_i} (1 - p_i)^{1-y_i}$$

We usually do not like to deal with products, so we can show that finding the *argmax* $L(p|Y)$ is IDENTICAL to finding the $\text{argmin}(-\log(L(p|Y)))$:

First, we note that:

$$\underset{\beta}{\text{argmax}}(L(p|y)) = \underset{\beta}{\text{argmax}}(\log L(p|y))$$

Estimating the Simple Logistic Model

First, we note that:

$$\underset{\beta}{\operatorname{argmax}}(L(p|y)) = \underset{\beta}{\operatorname{argmax}}(\log L(p|y))$$

which we can expand as:

$$l(p|Y) = \log L(p|Y) = \log \prod_i p_i^{y_i} (1 - p_i)^{1-y_i} = \sum_i \log\{p_i^{y_i} (1 - p_i)^{1-y_i}\}$$

Estimating the Simple Logistic Model

$$l(p|Y) = \log L(p|Y) = \log \prod_i p_i^{y_i} (1 - p_i)^{1-y_i} = \sum_i \log\{p_i^{y_i} (1 - p_i)^{1-y_i}\}$$

We use the property that the log of a product is the sum of the logs

$$l(p|Y) = \log L(p|Y) = \sum_i \log p_i^{y_i} + \log(1 - p_i)^{1-y_i}$$

And that $\log a^b = b \log a$

$$l(p|Y) = \log L(p|Y) = \sum_i y_i \log p_i + (1 - y_i) \log(1 - p_i)$$

Estimating the Simple Logistic Model

Maximizing a function is equivalent to minimizing the negative of the function

$$\underset{\beta}{\operatorname{argmax}} L(p|y) = \underset{\beta}{\operatorname{argmax}} \log L(p|y) = \underset{\beta}{\operatorname{argmin}} (-\log L(p|y)) = \underset{\beta}{\operatorname{argmin}} (-l(p|Y))$$

Using the formula for the log-likelihood, we get

$$\beta^* = \underset{\beta}{\operatorname{argmin}} (-l(p|Y)) = - \sum_i y_i \log p_i + (1 - y_i) \log(1 - p_i)$$

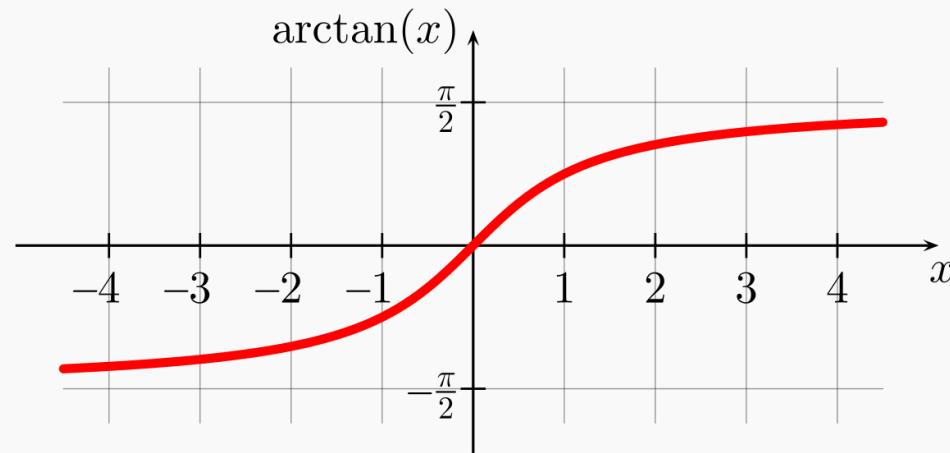
The final loss, called the **Binary Cross Entropy**, is then:

$$L_{BCE} = - \sum_i y_i \log p_i + (1 - y_i) \log(1 - p_i)$$

Food for thought:

Who came up with logistic regression? Why the logistic function again?

Why is the logistic function used as the S-shaped curve? What other functions could be used?



Stat 110 Idea: **any** CDF function (unbounded and continuous) could be used as the S-shaped curve. Econometricians love to use $\Phi(\beta_0 + \beta_1 X_1 + \dots)$.

Outline

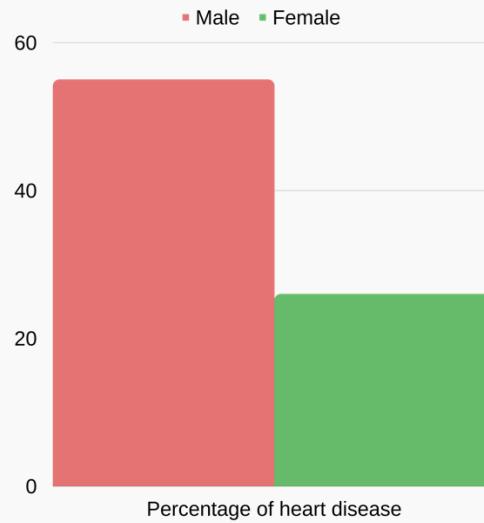
- What is Classification?
- Why not Linear Regression?
- Estimating the Simple Logistic Model
- **Inference in Logistic Regression**
- Multiple Logistic Regression
- Classification Decision Boundaries

Statistical Inference in Logistic Regression

Just like in linear regression, when the predictor, X , is binary, the interpretation of the model simplifies.

In this case, what are the interpretations of $\hat{\beta}_0$ and $\hat{\beta}_1$?

Consider the heart disease dataset represented here:



If we predict heart disease based on biological sex, how would you calculate and interpret the coefficient estimates $\hat{\beta}_0$ and $\hat{\beta}_1$?

A Second Logistic Regression Model in sklearn

Here is a logistic regression output to predict $Y = \text{AHD}$ from $X = \text{MaxHR}$:

```
logreg = LogisticRegression(penalty='none')
logreg.fit(df_heart[['Female']], df_heart['AHD'])

print('Estimated beta1: \n', logreg.coef_)
print('Estimated beta0: \n', logreg.intercept_)
```

```
Estimated beta1:
 [[-1.27219988]]
Estimated beta0:
 [0.21440982]
```

```
df_heart['Female'] = 1*(df_heart['Sex'] == 0)
pd.crosstab(df_heart['Female'], df_heart['AHD'])
```

| | AHD | No | Yes |
|--------|-----|-----|-----|
| Female | | | |
| 0 | 92 | 114 | |
| 1 | 72 | 25 | |

What is the estimated model? What are the interpretations of the $\hat{\beta}$ s?

$$\ln\left(\frac{\hat{P}(Y = 1)}{1 - \hat{P}(Y = 1)}\right) = 0.2144 - 1.272(\text{Female})$$

What is the estimated log-odd of AHD for Females and for Males? What about estimated probabilities? How does this agree with the table?

Statistical Inference in Logistic Regression

The **uncertainty of the estimates** $\hat{\beta}_0$ and $\hat{\beta}_1$ can be quantified and used to calculate both **confidence** intervals and **hypothesis** tests.

Of course, you can use **bootstrapping** (CI) and **permutation testing** (hypothesis testing) to perform these inferences.

Note:

The estimate for the standard errors of these estimates without bootstrap, is based on a quantity called **Fisher's Information** (beyond the scope of this class), which is related to the curvature of the log-likelihood function (the second derivative). Why does this make sense geometrically?

Due to the nature of the underlying Bernoulli distribution, if you estimate the underlying proportion p_i , you get the variance for free! Because of this, the inferences will be based on the normal approximation (and not t-distribution based).

Outline

- What is Classification?
- Why not Linear Regression?
- Estimating the Simple Logistic Model
- Inference in Logistic Regression
- **Multiple Logistic Regression**
- Classification Decision Boundaries

Multiple Logistic Regression

It is simple to illustrate examples in logistic regression when there is just one predictors variable.

But the approach ‘easily’ generalizes to the situation where there are **multiple predictors**.

A lot of the same details as linear regression apply to logistic regression. Interactions can be considered, multicollinearity is a concern and so is overfitting.

So how do we correct for such problems?

Regularization and checking though train and cross-validation!

We will get into the details of this, along with other extensions of logistic regression, in the next lecture.

Multiple Logistic Regression

Earlier we saw the general form of *simple* logistic regression, meaning when there is just one predictor used in the model:

$$\ln\left(\frac{P(Y = 1)}{1 - P(Y = 1)}\right) = \beta_0 + \beta_1 X$$

Multiple logistic regression is a generalization to multiple predictors. More specifically we can define a multiple logistic regression model to predict $P(Y = 1)$ as such:

$$\ln\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

Fitting Multiple Logistic Regression

The estimation procedure is identical to that as before for simple logistic regression:

- A likelihood approach is taken, and the negative log-likelihood is minimized across all parameters $\beta_0, \beta_1, \dots, \beta_p$ using an iterative method like Gradient Descent.

The actual fitting of a Multiple Logistic Regression is easy using software (of course there's a python package for that) as the iterative minimization of the -ve log-likelihood has already been hard coded.

In the `sklearn.linear_model` package, you just have to [create your multidimensional design matrix \$X\$](#) to be used as predictors in the `LogisticRegression` function.

Interpretation of Multiple Logistic Regression

Interpreting the coefficients in a multiple logistic regression is similar to that of linear regression.

Key: since there are other predictors in the model, the coefficient $e^{\hat{\beta}_j}$ is the multiplicative change in odds between the j^{th} predictor and the response. But do we have to say “Controlling for the other predictors in the model”?

We are trying to attribute the partial *effects* of each model controlling for the others (aka, controlling for possible *confounders*).

Multicollinearity plays a role just like in linear regression.

Outline

- What is Classification?
- Why not Linear Regression?
- Estimating the Simple Logistic Model
- Inference in Logistic Regression
- Multiple Logistic Regression
- **Classification Decision Boundaries**

Using Logistic Regression for Classification

How can we use logistic regression to perform classification?

That is, how can we predict when $Y = 1$ vs. when $Y = 0$?

We can classify all observations for which:

- Classify all observations with $\hat{P}(Y = 1) \geq 0.5$ to be in the group associated with $Y = 1$.
- Classify all observations with $\hat{P}(Y = 1) < 0.5$ to be in the group associated with $Y = 0$.

How would this extend if Y has 3+ classes?

Decision Boundaries for Classification

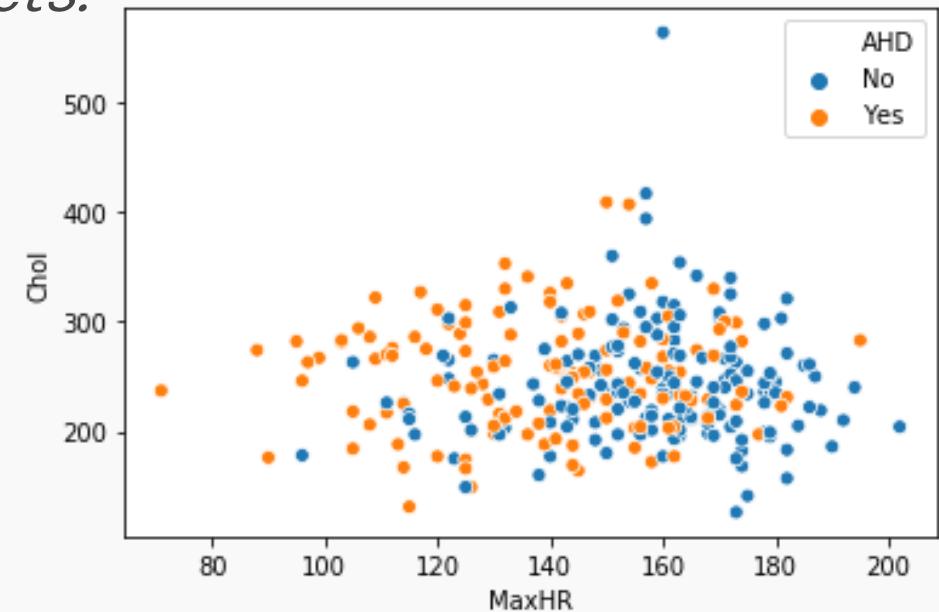
Recall that we could attempt to purely classify each observation based on whether the estimated $P(Y = 1)$ from the model is at least 0.5:

$$\hat{P}(Y = 1) \geq 0.5$$

This results in a **decision boundary**: a surface (line, curve, etc. in 2D) that separates the predicted classes into *sets*.

Here's a 2-D plot from our Heart Data Set:

How do you expect logistic regression to draw the decision boundary?



Decision Boundaries Example

Here is the output from a logistic regression model with 2 predictors:

What is the estimated model?

$$\ln \left(\frac{\hat{P}(Y=1)}{1-\hat{P}(Y=1)} \right) = 5.423 - 0.0439(\text{MaxHR}) + 0.0039(\text{Chol})$$

What are the interpretations of the $\hat{\beta}$ s?

What will the decision boundary look like? Key: if $\hat{P}(Y = 1) = 0.5$, then what are the estimated odds? What are the estimated log-odds?

In logistic regression, the decision boundary is defined when $X\beta = 0$.

```
data_x = df_heart[['MaxHR', 'Chol']]
data_y = df_heart['AHD']

logreg = LogisticRegression(penalty='none', fit_intercept=True)
logreg.fit(data_x, data_y);

print('Estimated beta1, beta2: \n', logreg.coef_)
print('Estimated beta0: \n', logreg.intercept_)

Estimated beta1, beta2:
 [[-0.04388093  0.00391746]]
Estimated beta0:
 [5.42271131]
```

2D Classification in Logistic Regression: Example #1

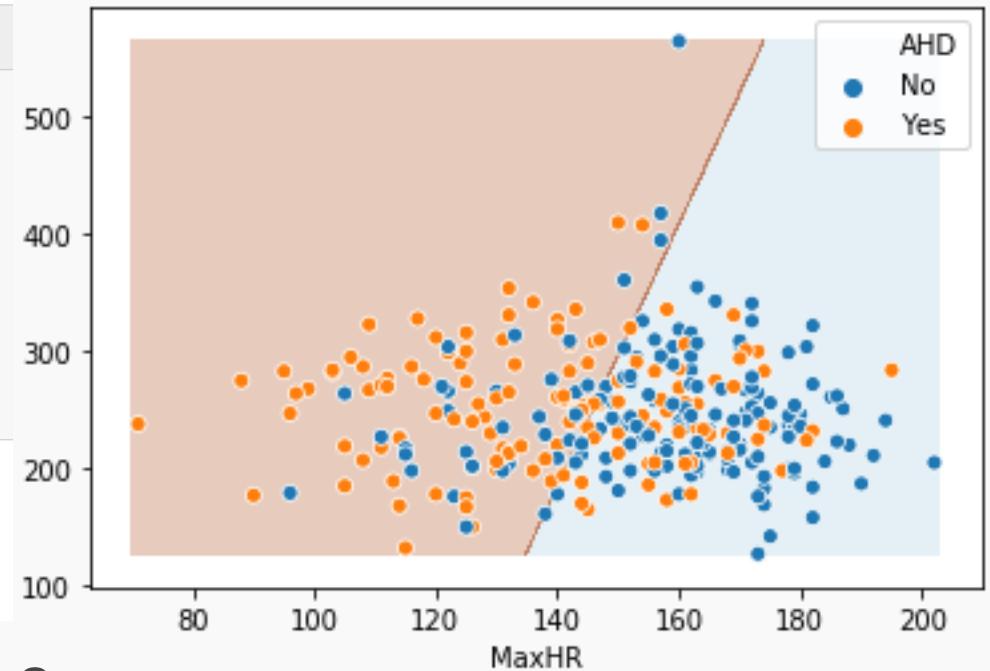
A logistic regression model was fit to predict $Y = \text{AHD}$ from $X_1 = \text{MaxHR}$ and $X_2 = \text{Chol}$. Results shown below:

```
data_x = df_heart[['MaxHR', 'Chol']]
data_y = df_heart['AHD']

logreg = LogisticRegression(penalty='none', fit_intercept=True)
logreg.fit(data_x, data_y);

print('Estimated beta1, beta2: \n', logreg.coef_)
print('Estimated beta0: \n', logreg.intercept_)

Estimated beta1, beta2:
 [[-0.04388093  0.00391746]]
Estimated beta0:
 [5.42271131]
```



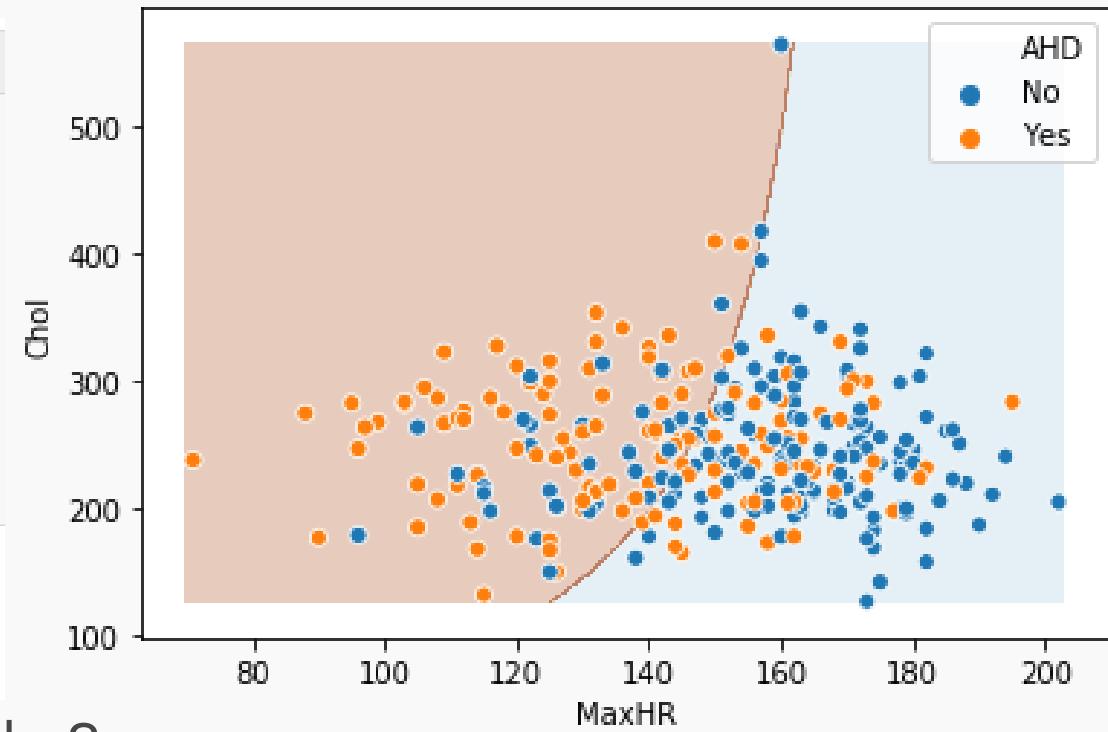
What will the decision boundary look like?

In logistic regression, decision boundaries are structured to be linear!

2D Classification in Logistic Regression: Example #2

A logistic regression model was fit to predict $Y = \text{AHD}$ from $X_1 = \text{MaxHR}$ and $X_2 = \text{Chol}$, and $X_3 = \text{their interaction}$. Results are shown below:

```
df_heart['Interaction'] = df_heart.MaxHR * df_heart.Chol  
  
data_x = df_heart[['MaxHR', 'Chol', 'Interaction']]  
data_y = df_heart['AHD']  
  
logreg = LogisticRegression(penalty='none', fit_intercept=True)  
logreg.fit(data_x, data_y);  
  
print('Estimated beta1, beta2, beta3: \n', logreg.coef_)  
print('Estimated beta0: \n', logreg.intercept_)  
  
Estimated beta1, beta2, beta3:  
 [[-0.00785835  0.02682656 -0.00015188]]  
Estimated beta0:  
 [5.70800455e-05]
```



What will the decision boundary look like?

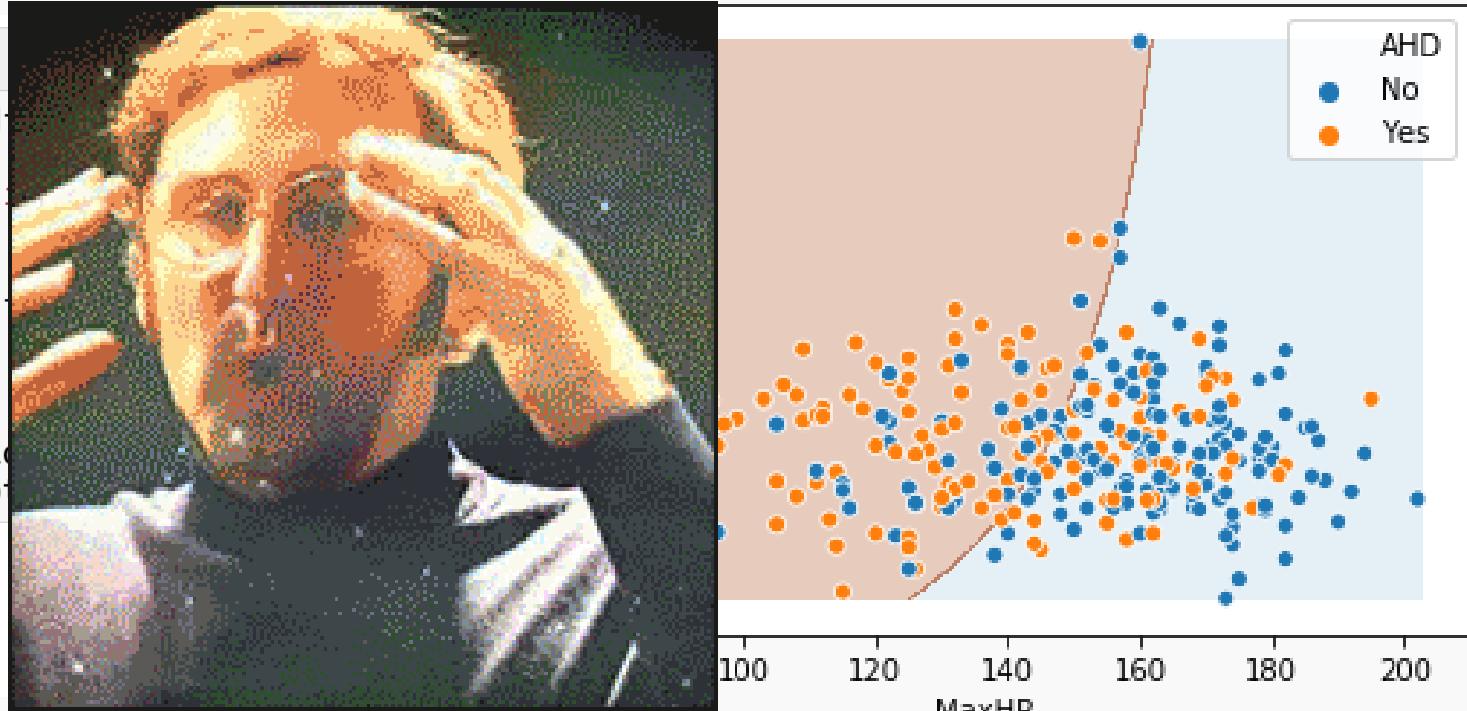
In logistic regression, decision boundaries are not always structured to be linear!

2D Classification in Logistic Regression: Example #2

A logistic regression model was fit to predict $Y = \text{AHD}$ from $X_1 = \text{MaxHR}$ and $X_2 = \text{Chol}$, and $X_3 = \text{their interaction}$. Results are shown below:

```
df_heart['Interaction'] = df_heart.MaxHR * df_heart.Chol  
  
data_x = df_heart[['MaxHR', 'Chol', 'Interaction']]  
data_y = df_heart['AHD']  
  
logreg = LogisticRegression(penalty='none', C=100)  
logreg.fit(data_x, data_y)  
  
print('Estimated beta1, beta2, beta3: \n', logreg.coef_)  
print('Estimated beta0: \n', logreg.intercept_)
```

Estimated beta1, beta2, beta3:
[[-0.00785835 0.02682656 -0.00015188]]
Estimated beta0:
[5.70800455e-05]



What will the decision boundary look like?

In logistic regression, decision boundaries are not always structured to be linear!

Polynomial Logistic Regression

We saw a 2-D plot last time which had two predictors, X_1, X_2 . A similar one is shown here but the decision boundary is again not linear.

We can extend multiple Logistic Regression as we did with polynomial regression:

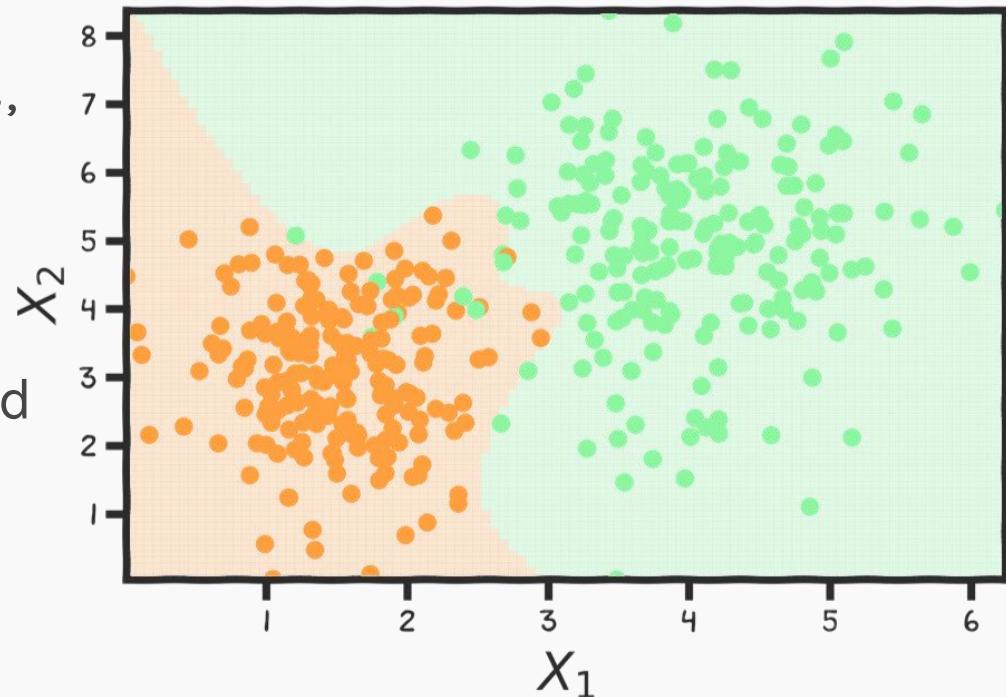
We transform the data by adding new predictors:

$$\tilde{x} = [1, \tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_M]$$

where $\tilde{x}_k = x^k$.

The polynomial Logistic Regression can be expressed as:

$$\log\left(\frac{P(Y = 1)}{1 - P(Y = 1)}\right) = \tilde{X}\beta$$



Geometry of Decision Boundaries (Logistic Regression)

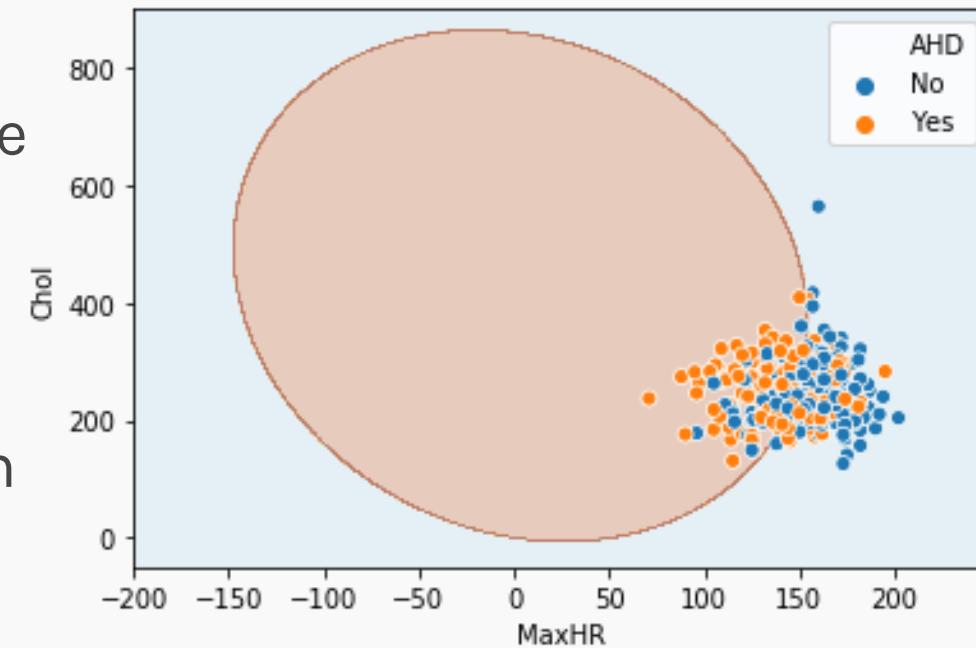
$$\log\left(\frac{P(Y = 1)}{1 - P(Y = 1)}\right) = \tilde{X}\beta$$

Thus we can define our logistic regression model to achieve a desired geometry.

For example, what set for $X = f(X_1, X_2)$ should we choose if we want a *circular* decision boundary?

$$X = \{X_1, X_1^2, X_2, X_2^2, X_1 X_2\}$$

What could be an alternative modeling approach
(think outside of logistic regression)?

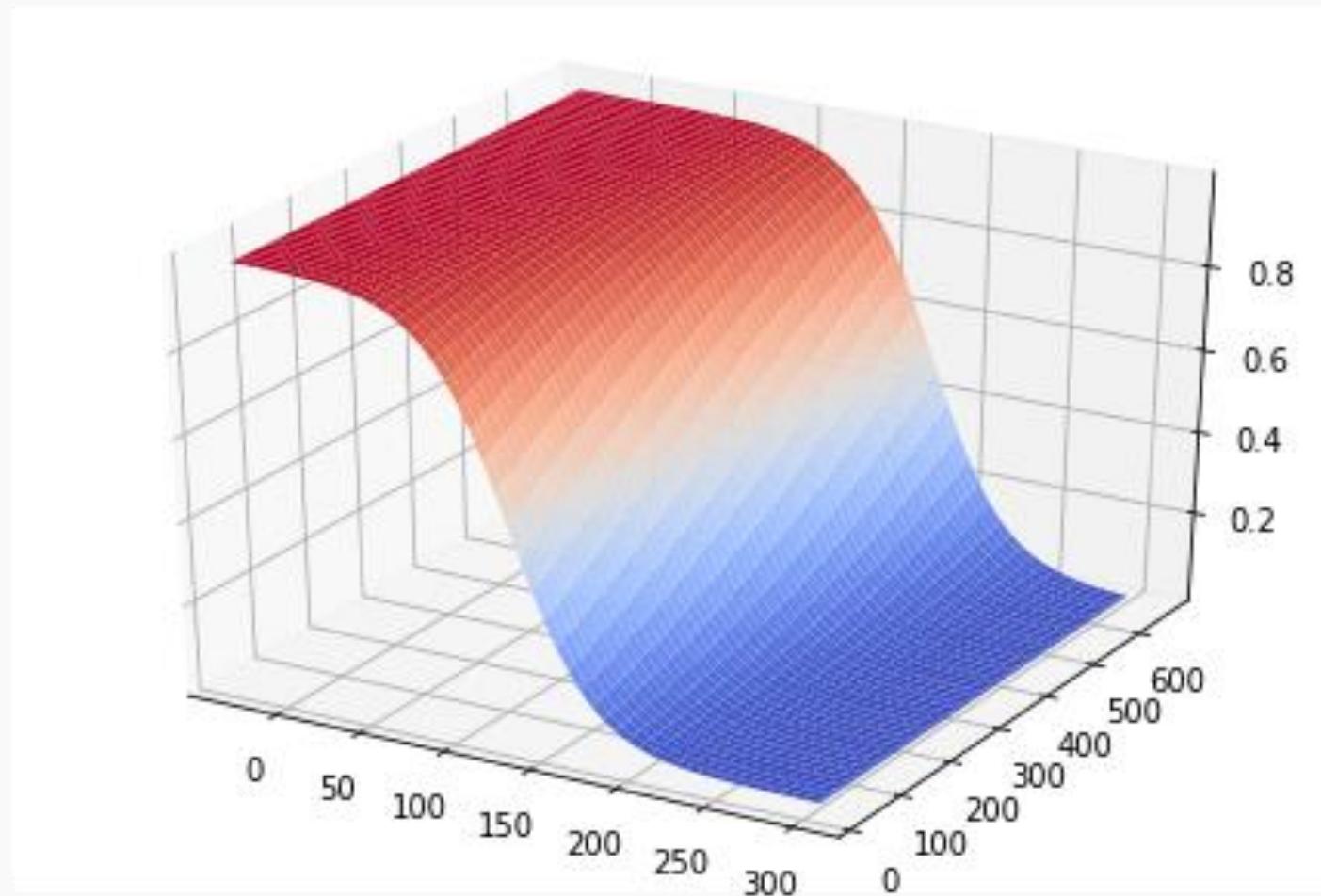


Lecture #14: Logistic Regression - Part II

CS1090A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai, and Chris Gumb



Visualizing Multiple Logistic Regression



Outline

- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- Multiclass Logistic Regression
 - Multinomial Logistic Regression
 - One-vs-Rest Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves

Interactions in Multiple Logistic Regression

Just like in linear regression, interaction terms can be considered in logistic regression. An **interaction terms** is incorporated into the model the same way, and the interpretation is very similar (on the log-odds scale of the response of course).

Write down the model for the Heart data for the 2 predictors plus the interactions term based on the output on the next slide.

Here, we are predicting AHD from Age, Female, and the interaction between the two.

Interpreting Multiple Logistic Regression: an Example

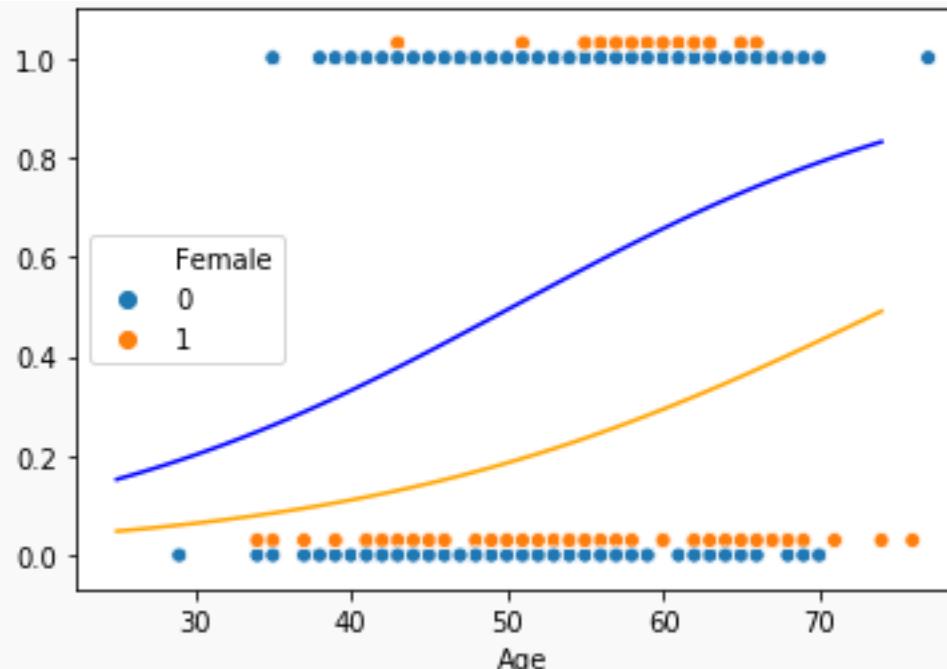
The results for the multiple logistic regression model are:

```
df_heart['Age_Female'] = df_heart['Age']*df_heart['Female']
X = df_heart[['Age', 'Female', 'Age_Female']]

logit = LogisticRegression(penalty='none', fit_intercept=True)
logit.fit(X, df_heart['AHD'])

print('Estimated betas (B0,B1,B2,B3):', logit.intercept_, logit.coef_)

Estimated betas (B0,B1,B2,B3): [-3.40463831] [[ 0.06754528 -1.08200061 -0.00742792]]
```



Some questions

```
Estimated betas (B0,B1,B2,B3): [-3.40463831] [[ 0.06754528 -1.08200061 -0.00742792]]
```

1. Write down the complete model. Break this down into the model to predict log-odds of heart disease (HD) based on Age for males and the same model for females.

Some questions

Estimated betas (B0,B1,B2,B3): [-3.40463831] [[0.06754528 -1.08200061 -0.00742792]]

2. Interpret the results of this model. What does the coefficient for the interaction term represent?

Outline

- Interpreting interactions in logistic regression
- **Regularization in Logistic Regression**
- Multiclass Logistic Regression
 - Multinomial Logistic Regression
 - One-vs-Rest Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves

Review: Regularization in Linear Regression

What was the loss function in linear regression (not regularized)?

We saw in linear regression that maximizing the log-likelihood is equivalent to minimizing the sum of squares error:

$$\operatorname{argmin}_{\beta} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \operatorname{argmin}_{\beta} \sum_{i=1}^n (y_i - (\beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi}))^2$$

Review: Regularization in Linear Regression

A regularization approach was to add a penalty to this loss.

For Ridge Regression this becomes:

$$\operatorname{argmin}_{\beta} \sum_{i=1}^n (y_i - (\beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi}))^2 + \lambda \sum_{j=1}^J \beta_j^2$$

This penalty *shrinks* the estimates towards zero.

Note: This it is analogue of using a Normal prior centered at zero in the Bayesian paradigm.

Recall: Loss function in Logistic Regression

A similar approach can be used in logistic regression. Here, maximizing the log-likelihood is equivalent to minimizing the following loss function (**binary cross-entropy**):

$$\operatorname{argmin}_{\beta_0, \beta_1, \dots, \beta_p} \left[- \sum_{i=1}^n (y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i)) \right]$$

$$\text{where } p_i = \frac{1}{1 - e^{-(\beta_0 + \beta_1 x_{1,i} + \dots + \beta_p x_{p,i})}}$$

Why is this a good loss function to minimize? Where does this come from?

The log-likelihood for independent $Y_i \sim \text{Bern}(p_i)$.

Regularization in Logistic Regression

A penalty factor can then be added to this loss function and results in a new loss function that penalizes large values of the parameters:

$$\operatorname{argmin}_{\beta} \left[- \sum y_i \log p_i + (1 - y_i) \log(1 - p_i) + \lambda \sum_{j=1}^J \beta_j^2 \right]$$

The result is just like in linear regression: **shrink** the parameter estimates towards zero.

Note: the `sklearn` package uses a different tuning parameter: instead of λ they use a constant that is $C = \frac{1}{\lambda}$.

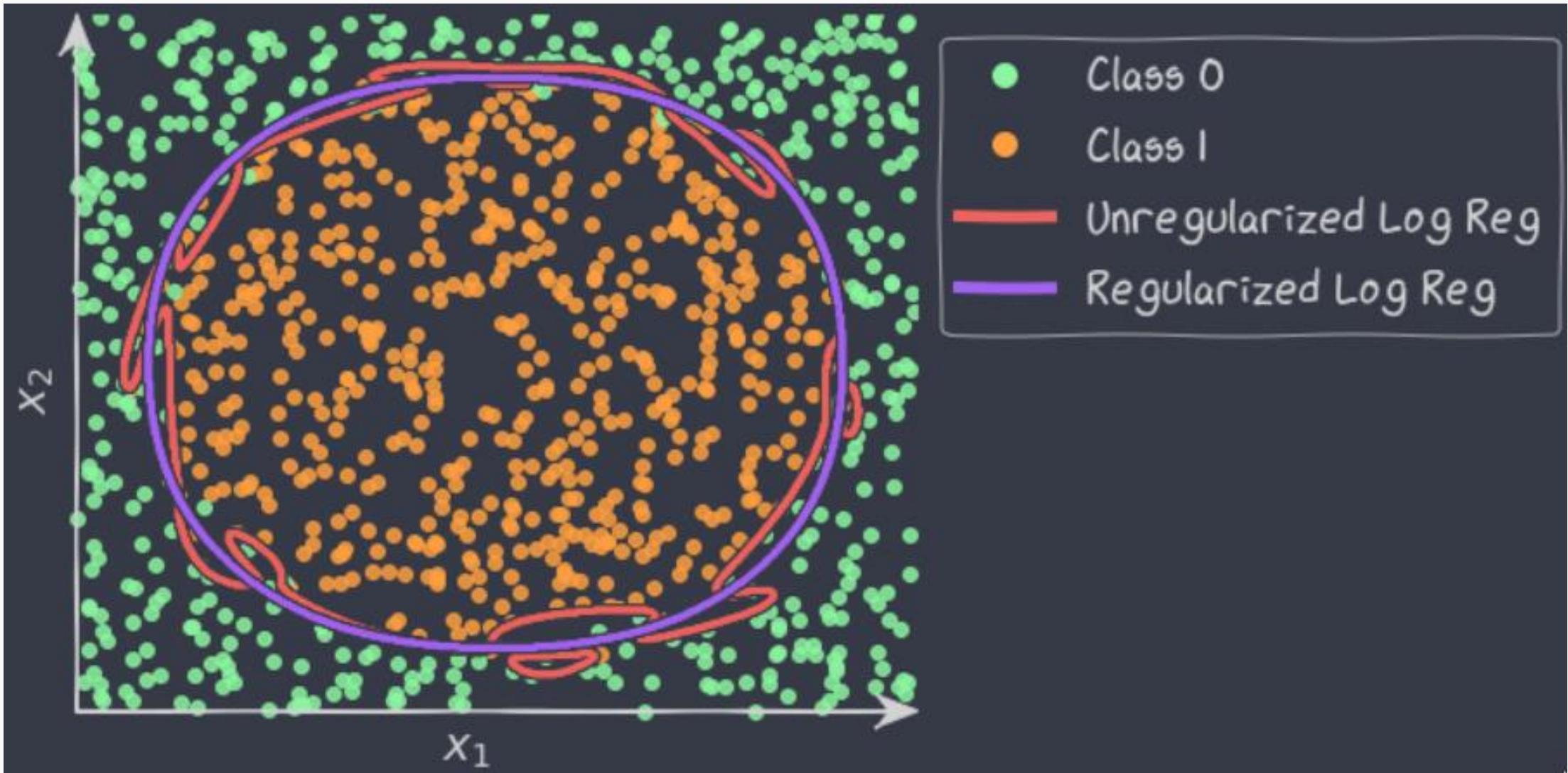
Regularization in Logistic Regression: tuning lambda

Just like in linear regression, the regularization parameter must be chosen.

How should we go about doing this?

Cross Validation! Through building multiple training and validation set , we can select the best regularization parameter.

Regularized Decision Boundaries



Outline

- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- **Multiclass Logistic Regression**
 - **Multinomial Logistic Regression**
 - One-vs-Rest Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves

Logistic Regression for predicting 3+ classes

There are several extensions to standard logistic regression when the response variable Y has more than 2 categories. The two most common are:

Nominal is used when the categories have no inherent order (like eye color: blue, green, brown, hazel, etc).

Ordinal logistic regression is used when the categories have a specific hierarchy (like class year: Freshman, Sophomore, Junior, Senior; or a 7-point rating scale from strongly disagree to strongly agree).

Multinomial Logistic Regression

There are two common approaches to estimating a nominal (not-ordinal) categorical variable that has more than 2 classes.

The first approach sets one of the categories in the response variable as the *reference* group, and then fits separate logistic regression models to predict the other cases based off of the reference group. For example, we could attempt to predict a student's concentration:

$$y = \begin{cases} 1 & \text{if Computer Science (CS)} \\ 2 & \text{if Statistics} \\ 3 & \text{otherwise} \end{cases}$$

from predictors X_1 number of psets per week, X_2 how much time playing video games per week, etc.

Multinomial Logistic Regression (cont.)

We could select the $y=3$ case as the reference group (other concentration), and then fit two **separate models**:

- 1) A model to predict $y=1$ (CS) from $y=3$ (others)
- 2) A model to predict $y=2$ (Stat) from $y=3$ (others).

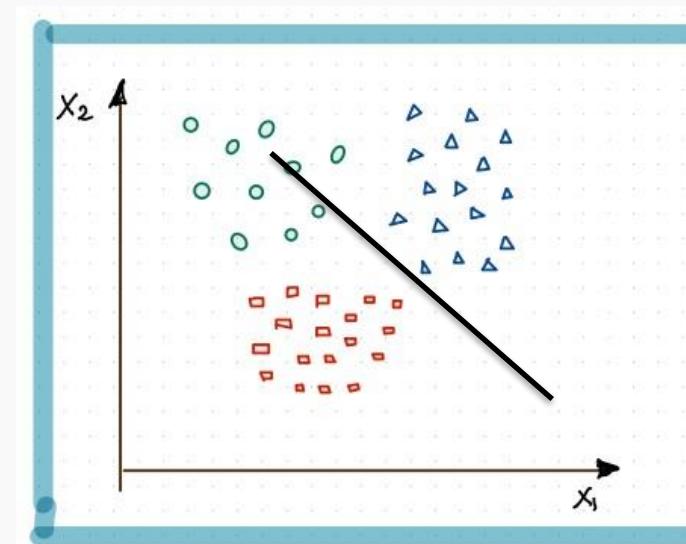
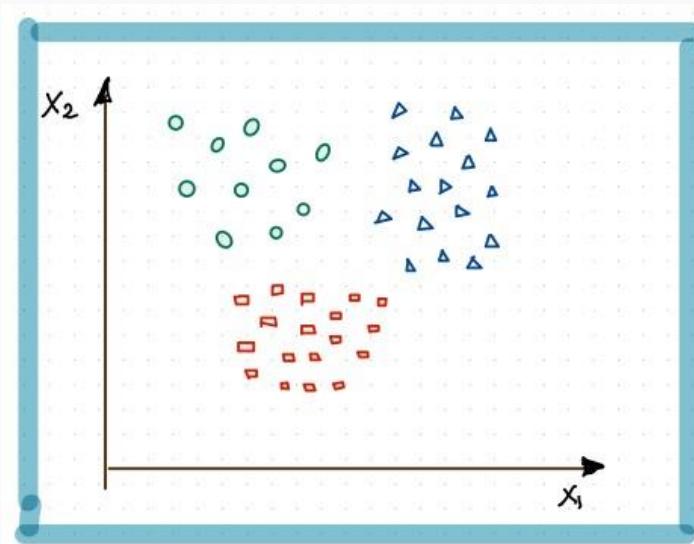
Ignoring interactions, how many parameters would need to be estimated?

How could these models be used to estimate the probability of an individual falling in each concentration?

Multinomial Logistic

Classifying three classes:

Red, Blue and Green can be turn into two binary Logistic Regressions



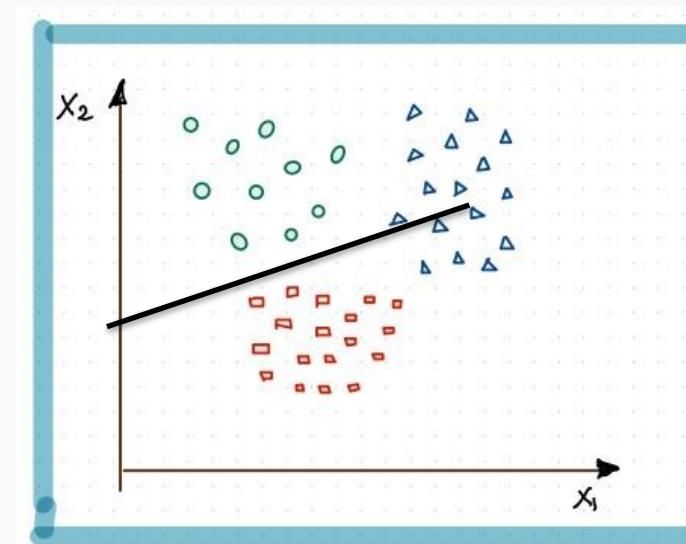
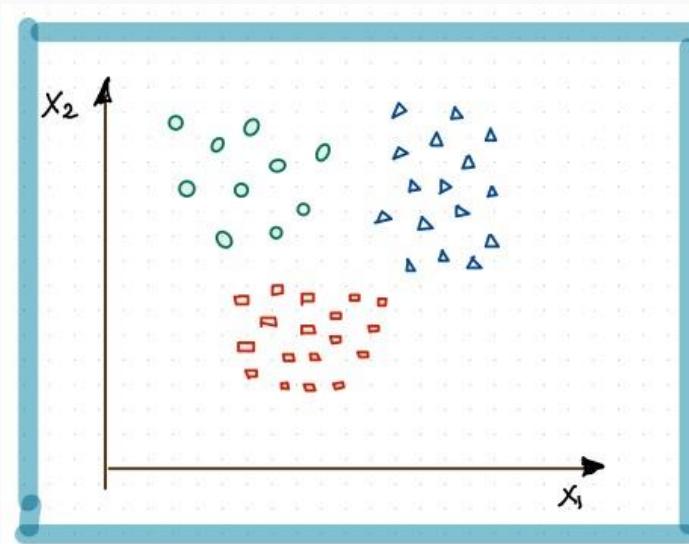
Blue vs Red

$$\ln \left(\frac{P(b)}{P(r)} \right) = \beta_b X$$

Multinomial Logistic

Classifying three classes,

Red, Blue and Green can be turn into two binary Logistic Regressions



Green vs Red

$$\ln \left(\frac{P(g)}{P(r)} \right) = \beta_g X$$

Multinomial Logistic Regression: the model

To predict K classes ($K > 2$) from a set of predictors X , a multinomial logistic regression can be fit:

$$\ln \left(\frac{P(Y = 1)}{P(Y = K)} \right) = \beta_{0,1} + \beta_{1,1}X_1 + \beta_{2,1}X_2 + \cdots + \beta_{p,1}X_p$$

$$\ln \left(\frac{P(Y = 2)}{P(Y = K)} \right) = \beta_{0,2} + \beta_{1,2}X_1 + \beta_{2,2}X_2 + \cdots + \beta_{p,2}X_p$$

⋮

$$\ln \left(\frac{P(Y = K - 1)}{P(Y = K)} \right) = \beta_{0,K-1} + \beta_{1,K-1}X_1 + \beta_{2,K-1}X_2 + \cdots + \beta_{p,K-1}X_p$$

Each separate model can be fit as independent standard logistic regression models!

Multinomial Logistic Regression in sklearn

```
mlogit = LogisticRegression(penalty="none", multi_class = 'multinomial')
mlogit.fit (nfl22[["Down", "ToGo"]], nfl22["Play_Type"])
```

The coefficients

```
print('Estimated beta1: \n', mlogit.coef_)
print('Estimated beta0: \n', mlogit.intercept_)
```

Estimated beta1:

```
[[ 1.71107026  0.09577593]
 [-0.40924154  0.03968915]
 [-1.30182872 -0.13546508]]
```

Estimated beta0:

```
[-6.29293303  1.88149967  4.41143335]
```

But wait, I thought you said we only fit $K-1$ logistic regression models?!? Why are there K intercepts and K sets of coefficients????

```
pd.crosstab(nfl22[["PlayType"]],
            nfl22[["Play_Type"]])
```

| Play_Type | 0 | 1 | 2 |
|------------|------|-------|-------|
| PlayType | | | |
| CLOCK STOP | 48 | 0 | 0 |
| FIELD GOAL | 854 | 0 | 0 |
| FUMBLES | 92 | 0 | 0 |
| PASS | 0 | 15397 | 0 |
| PUNT | 1874 | 0 | 0 |
| QB KNEEL | 353 | 0 | 0 |
| RUSH | 0 | 0 | 10794 |
| SACK | 0 | 1106 | 0 |
| SCRAMBLE | 0 | 784 | 0 |

What is sklearn doing?

The $K-1$ models in multinomial regression lead to the following probability predictions:

$$\ln \left(\frac{P(Y = k)}{P(Y = K)} \right) = \beta_{0,k} + \beta_{1,k}X_1 + \beta_{2,k}X_k + \cdots + \beta_{p,k}X_p$$

Note:
the different
denominators

$$P(Y = k) = P(Y = K)e^{\beta_{0,k} + \beta_{1,k}X_1 + \beta_{2,k}X_k + \cdots + \beta_{p,k}X_p}$$

This give us $K-1$ equations to estimate K probabilities for everyone.

But probabilities add up to 1 ☺, so we are all set.

sklearn then converts the above probabilities back into new betas
(just like logistic regression, but the betas won't match):

$$\ln \left(\frac{P(Y = k)}{P(Y \neq k)} \right) = \beta'_{0,k} + \beta'_{1,k}X_1 + \beta'_{2,k}X_k + \cdots + \beta'_{p,k}X_p$$

Outline

- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- Multiclass Logistic Regression
 - Multinomial Logistic Regression
 - **One-vs-Rest Logistic Regression**
- Bayes Theorem and Misclassification Rates
- ROC Curves

One vs. Rest (OvR) Logistic Regression

An alternative multiclass logistic regression model in sklearn is called the 'One vs. Rest' (OvR) approach, which is our second method.

If there are 3 classes, then 3 separate logistic regressions are fit, where the probability of each category is predicted over the rest of the categories combined. So for the concentration example, 3 models would be fit:

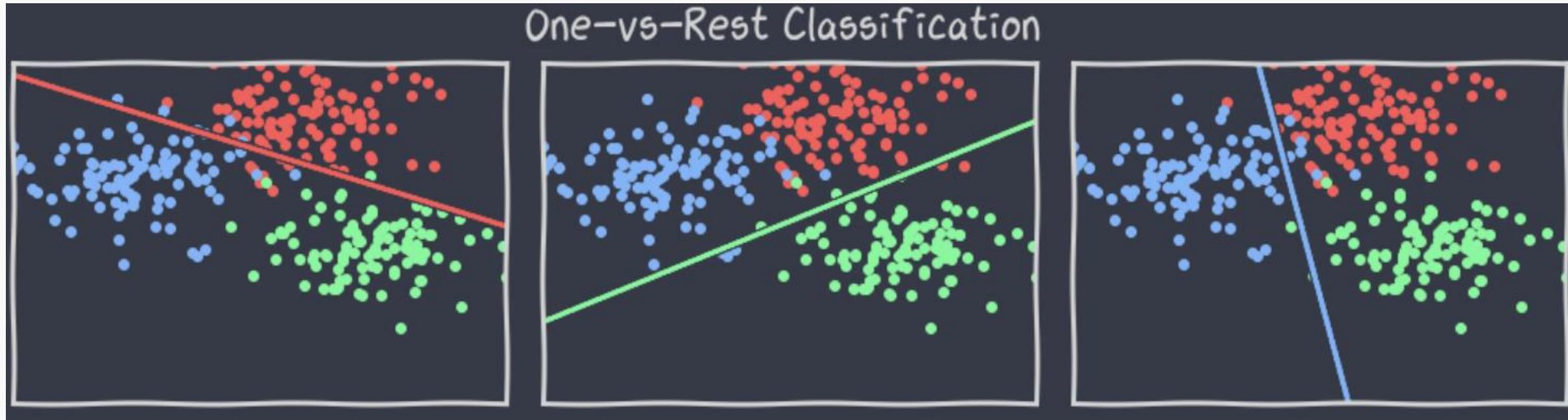
- a first model would be fit to predict CS from (Stat and Others) combined.
- a second model would be fit to predict Stat from (CS and Others) combined.
- a third model would be fit to predict Others from (CS and Stat) combined.

A picture is worth 1000 words.

One vs. Rest (ovr)

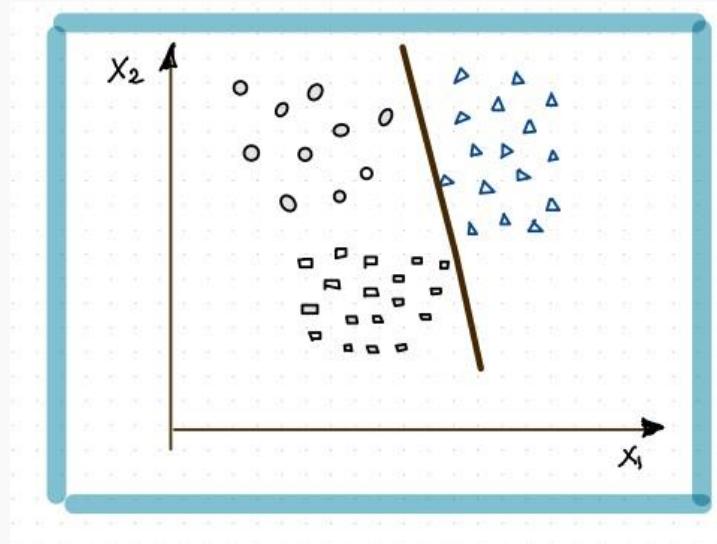
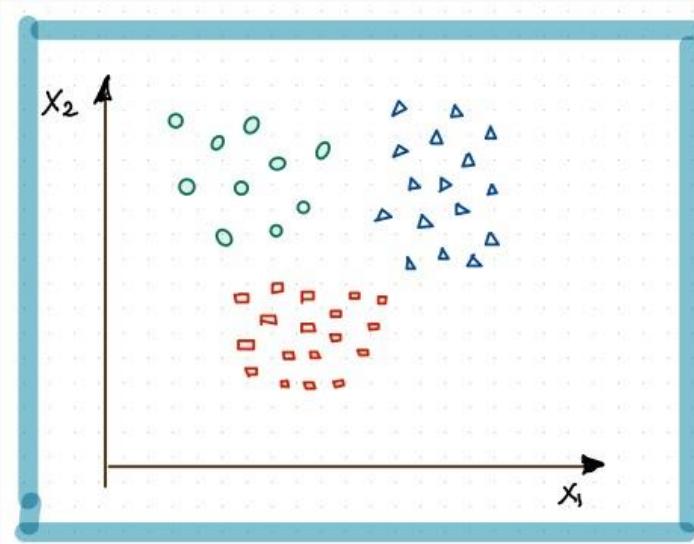
Classifying three classes,

Red, Blue and Green can be turn into three binary Logistic Regressions



One vs. Rest (ovr)

Classifying three classes,
Red, Blue and Green can be turn into three binary Logistic
Regressions



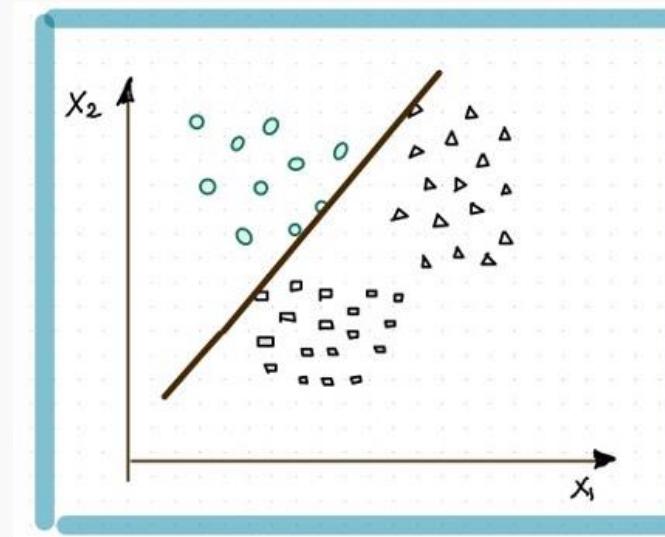
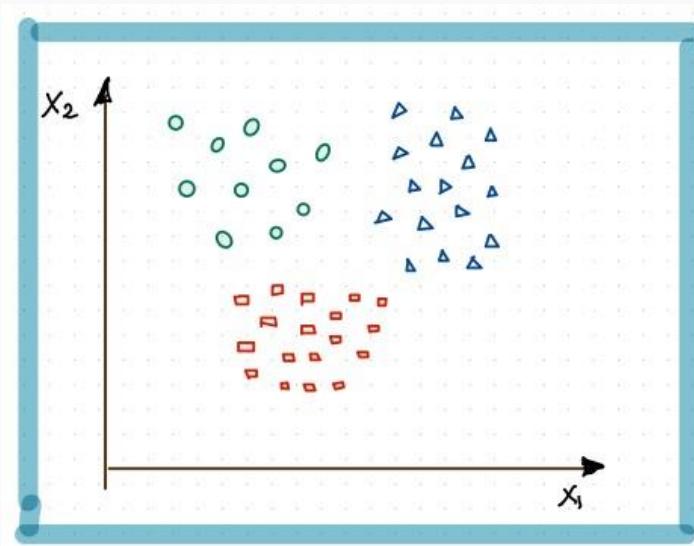
Blue vs others

$$\ln \left(\frac{P(b)}{1 - P(b)} \right) = \beta_b X$$

One vs. Rest (ovr)

Classifying three classes,

Red, Blue and Green can be turn into three binary Logistic Regressions



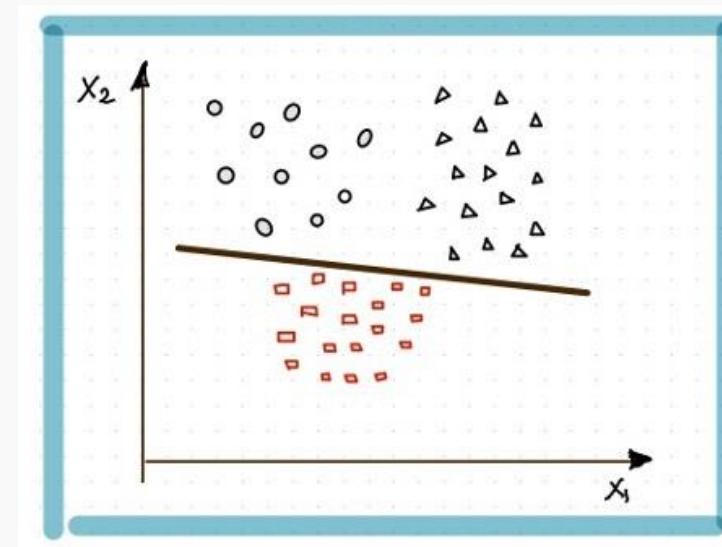
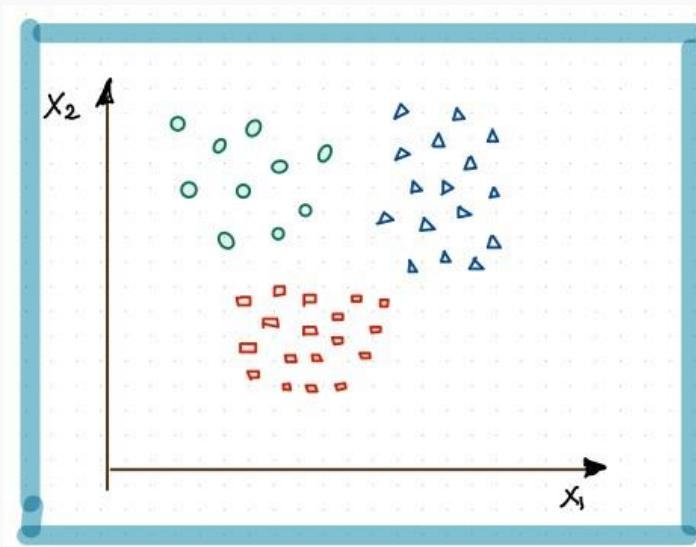
Green vs others

$$\ln \left(\frac{P(g)}{1 - P(g)} \right) = \beta_g X$$

One vs. Rest (ovr)

Classifying three classes,

Red, Blue and Green can be turn into three binary Logistic Regressions

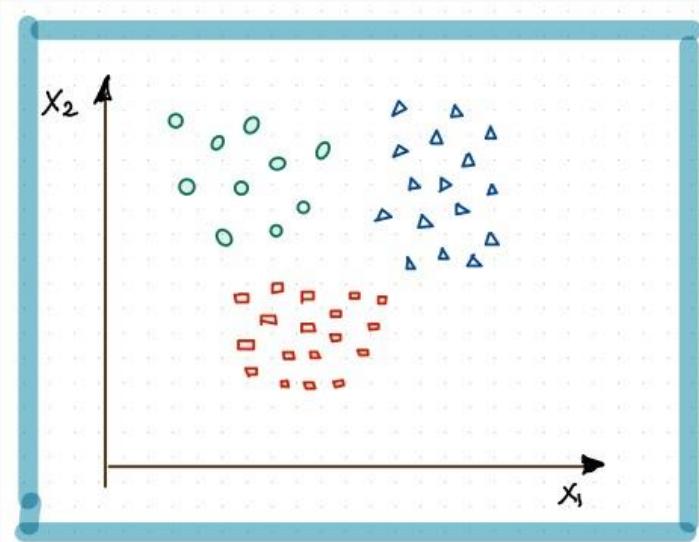


Red vs others

$$\ln\left(\frac{P(r)}{1 - P(r)}\right) = \beta_r X$$

One vs. Rest (ovr)

Classifying three classes,
Red, **Blue** and **Green** can be turn into three binary Logistic
Regressions



Green vs others:

$$\ln\left(\frac{P(g)}{1 - P(g)}\right) = \beta_g X$$

Blue vs others:

$$\ln\left(\frac{P(b)}{1 - P(b)}\right) = \beta_b X$$

Red vs others:

$$\ln\left(\frac{P(r)}{1 - P(r)}\right) = \beta_r X$$

sklearn **normalizes** the output of each of the three models when predicting probabilities:

$$\tilde{P}(b) = \frac{P(b)}{P(g)+P(b)+P(r)}$$

$$\tilde{P}(g) = \frac{P(g)}{P(g)+P(b)+P(r)}$$

$$\tilde{P}(r) = \frac{P(r)}{P(g)+P(b)+P(r)}$$

Estimation and Regularization in multiclass settings

There is no difference in the approach to estimating the coefficients in the multiclass setting: we maximize the log-likelihood (or minimize negative log-likelihood).

This combined negative log-likelihood of all K classes is sometimes called the **cross-entropy or multinomial logistic loss**:

$$\ell = \frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K \mathbb{1}(y_i = k) \ln(\hat{P}(y_i = k)) + \mathbb{1}(y_i \neq k) \ln(1 - \hat{P}(y_i = k))$$

And regularization can be done like always: add on a penalty term to this loss function based on L1 (sum of the absolute values) or L2 (sum of squares) norms.

Outline

- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- Multiclass Logistic Regression
 - Multinomial Logistic Regression
 - One-vs-Rest Logistic Regression
- **Bayes Theorem and Misclassification Rates**
- ROC Curves

Probability Review: Bayes' Theorem

What is conditional probability?

$$P(B|A) = \frac{P(A \text{ and } B)}{P(A)}$$

And using the fact that $P(A \text{ and } B) = P(A|B)P(B)$ we get the simplest form of Bayes' Theorem:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

Another version of Bayes' Theorem is found by substituting in the Law of Total Probability (LOTP) into the denominator:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A|B)P(B) + P(A|B^C)P(B^C)}$$

Diagnostic Testing

In the diagnostic testing paradigm, one cares about whether the results of a test (like a classification test) matches truth (the true class that observation belongs to). The simplest version of this is trying to detect disease ($D+$ vs. $D-$) based on a diagnostic test ($T+$ vs. $T-$).

Medical examples of this include various screening tests: breast cancer screening through (i) self-examination and (ii) mammography, prostate cancer screening through (iii) PSA tests, and Colo-rectal cancer through (iv) colonoscopies.

These tests are a little controversial because of poor predictive probability of the tests.

Diagnostic Testing (cont.)

Bayes' theorem can be rewritten for diagnostic tests:

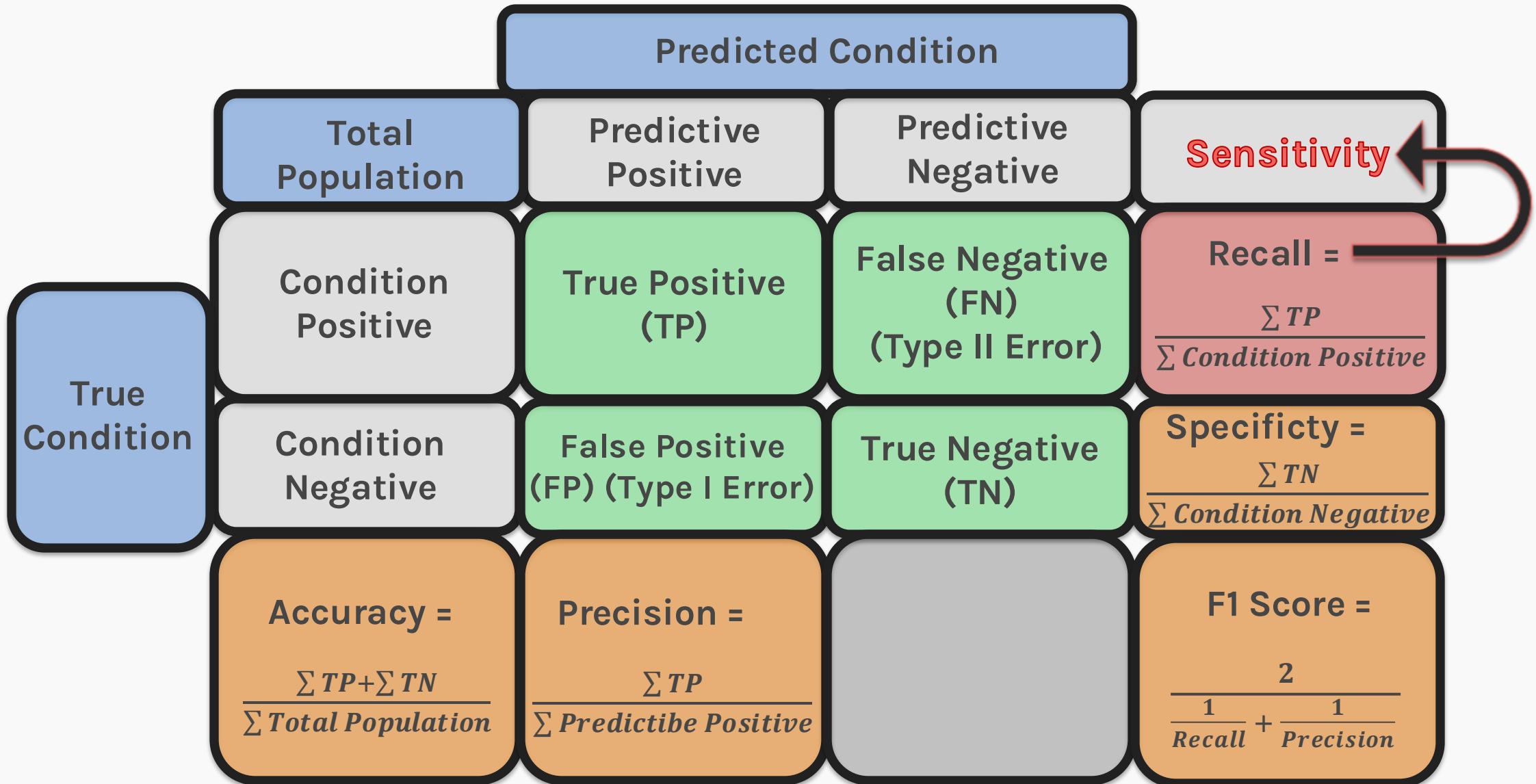
$$P(D+|T+) = \frac{P(T+|D+)P(D+)}{P(T+|D+)P(D+) + P(T+|D-)P(D-)}$$

These probability quantities can then be defined as:

- **Sensitivity:** $P(T+|D+)$
- **Specificity:** $P(T-|D-)$
- **Prevalence:** $P(D+)$
- **Positive Predictive Value:** $P(D+|T+)$
- **Negative Predictive Value:** $P(D-|T-)$

1 - Specificity

How do positive and negative predictive values relate? Be careful...



Diagnostic Testing

We mentioned that these tests are a little controversial because of their poor predictive probability. When will these tests have poor positive predictive probability?

When the disease is not very prevalent, then the number of 'false positives' will overwhelm the number of true positive. For example, PSA screening for prostate cancer has sensitivity of about 90% and specificity of about 97% for some age groups (men in their fifties), but prevalence is about 0.1%.

What is positive predictive probability for this diagnostic test?

Why do we care?



Error in Classification

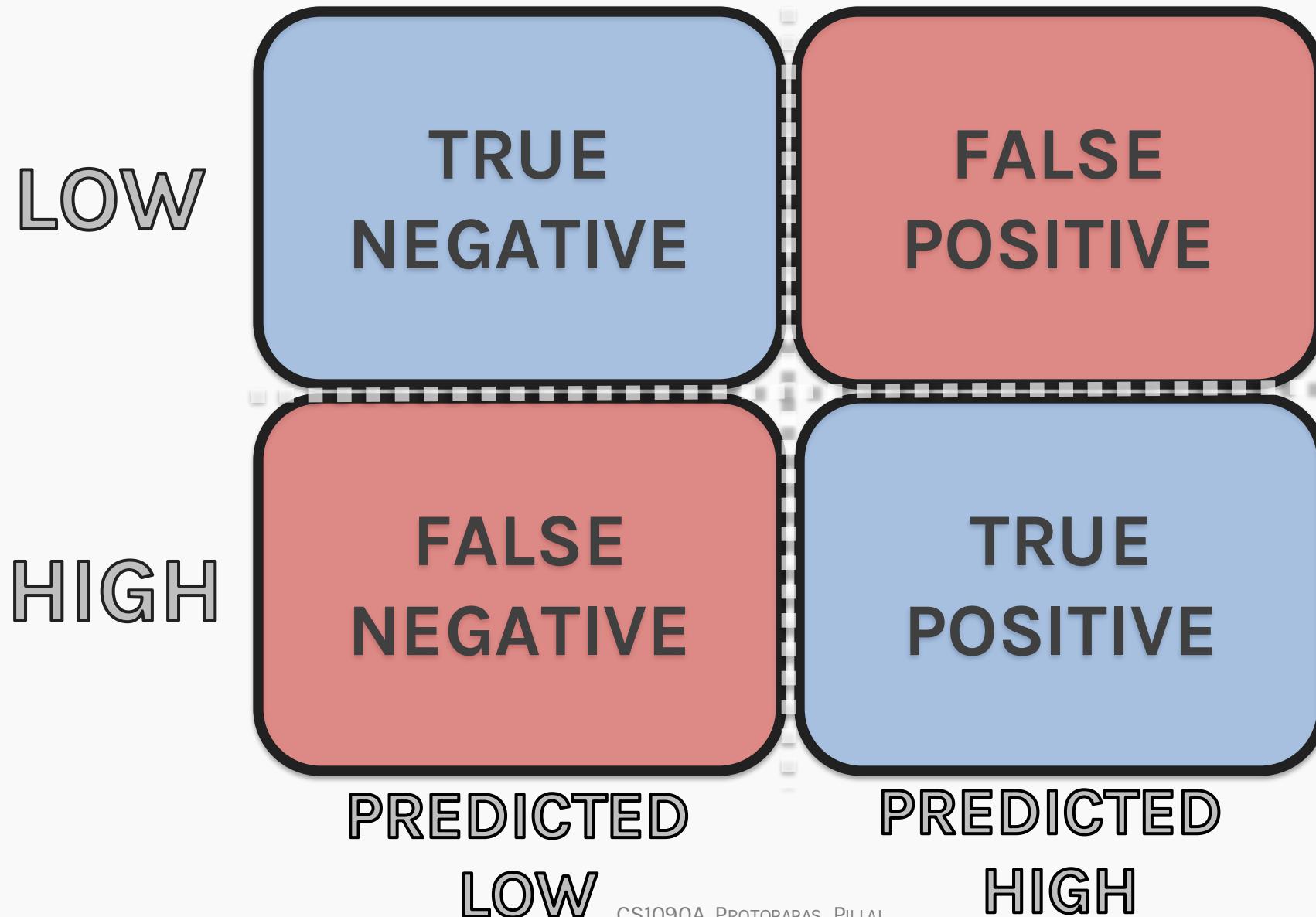
There are 2 major types of error in classification problems based on a binary outcome. They are:

False positives: incorrectly predicting $\hat{Y} = 1$ when it truly is in $Y = 0$.

False negatives: incorrectly predicting $\hat{Y} = 0$ when it truly is in $Y = 1$.

The results of a classification algorithm are often summarized in two ways: (1) a **confusion matrix**, sometimes called a **contingency table**, or a 2×2 table (more generally $(k \times k)$ table) and (2) a receiver operating characteristics (ROC) curve.

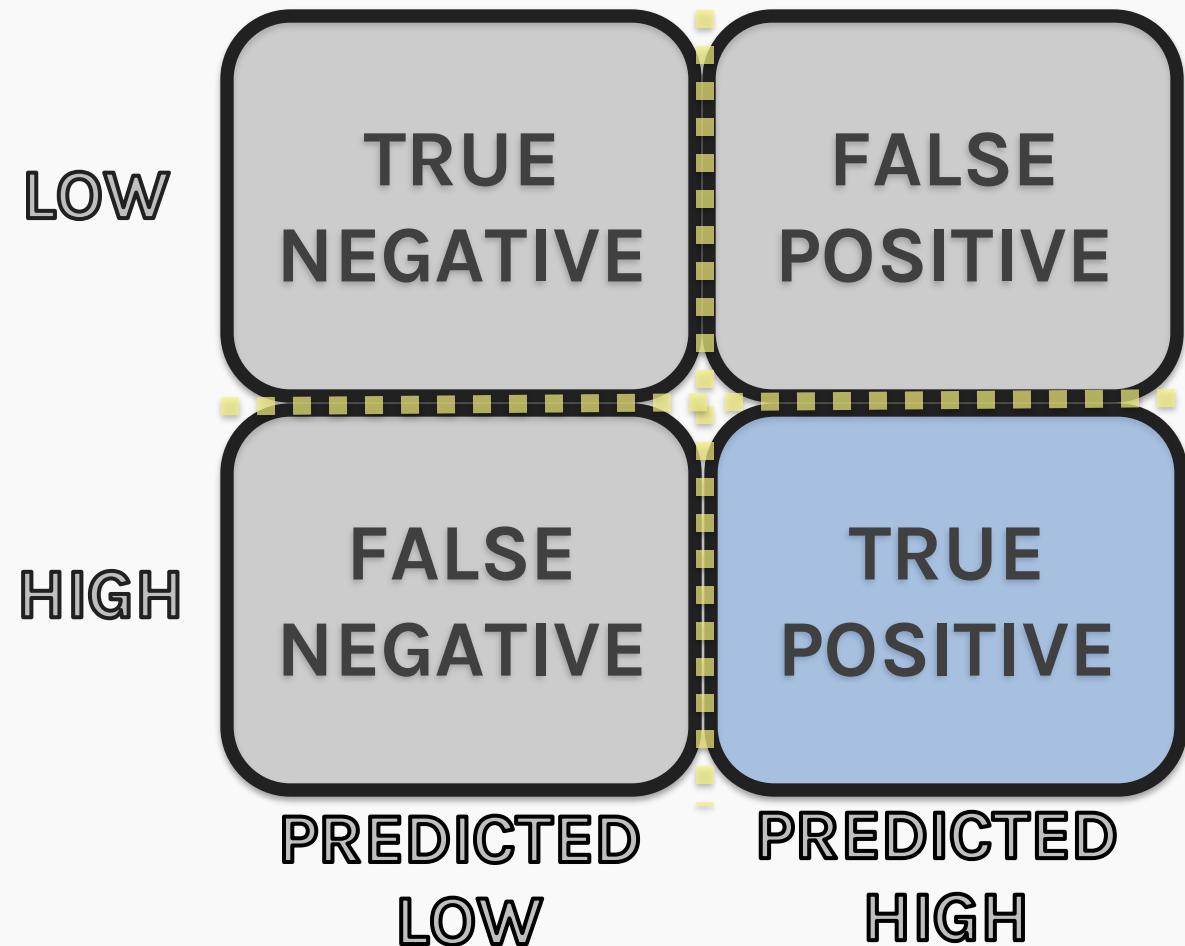
The 'Confusion' Matrix



The 'Confusion' Matrix

TRUE POSITIVE (TP)

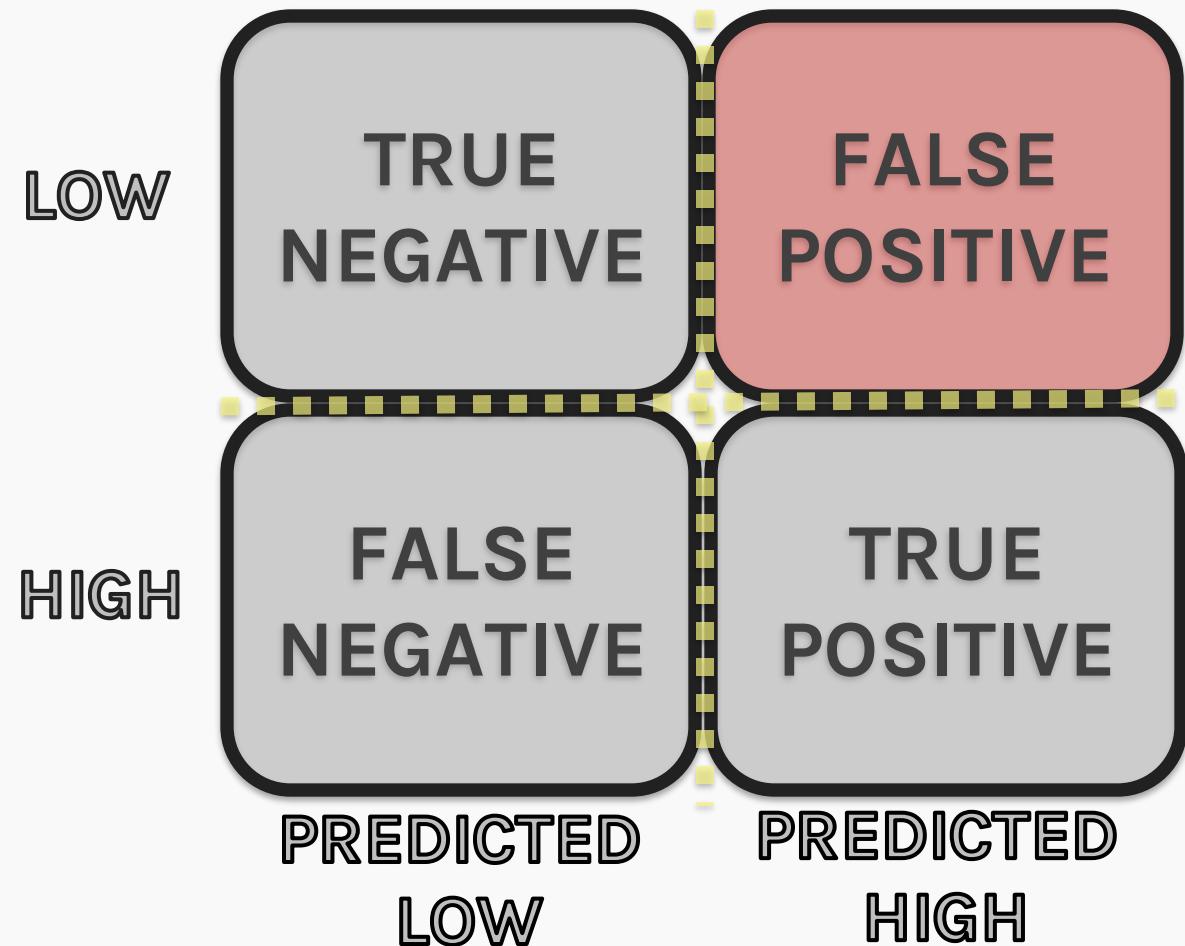
- Samples that are positive that the classifier predicts as positive are called True Positives.
- Example: a positive Covid test result would be a TRUE POSITIVE if you actually have Covid.



The 'Confusion' Matrix

FALSE POSITIVE (FP)

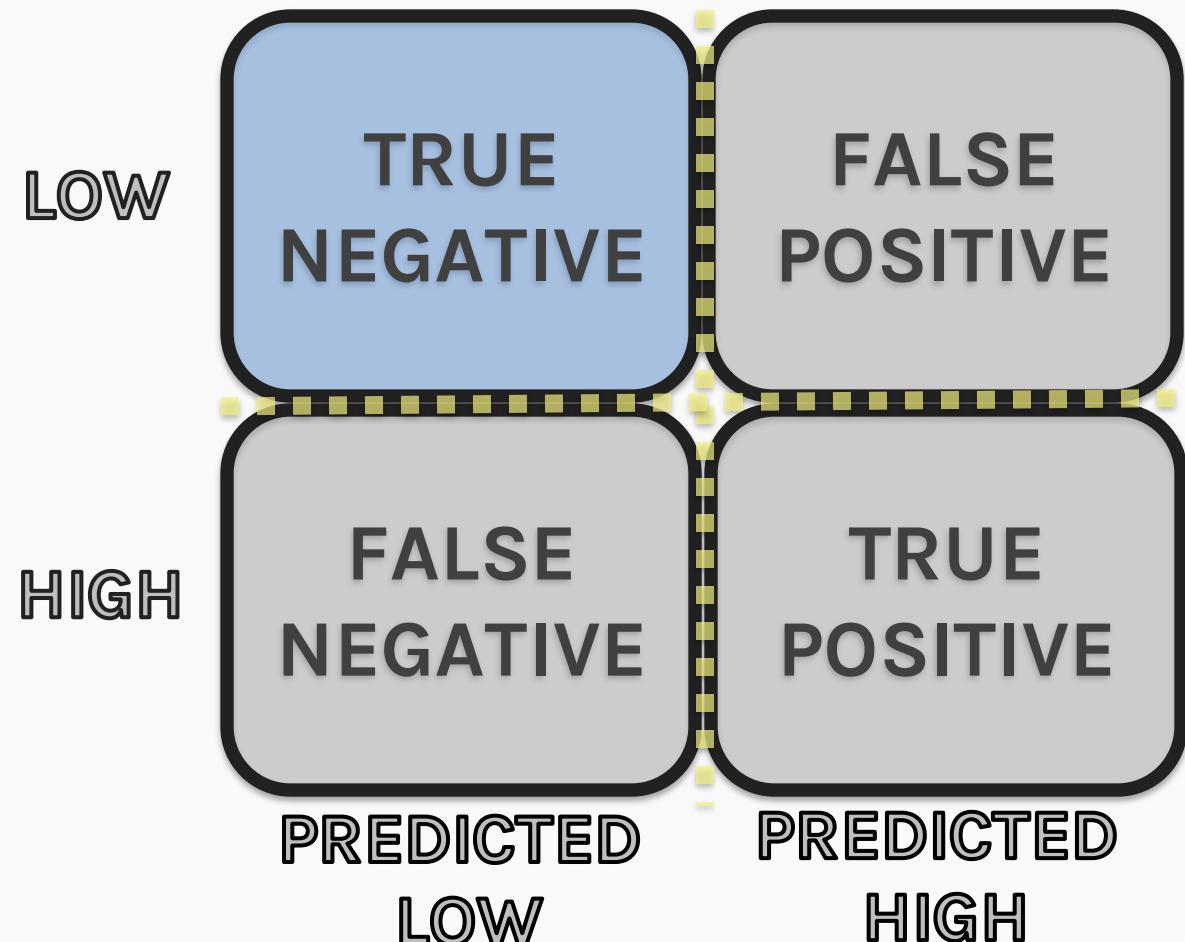
- Samples that are negative that the classifier predicts as positive are called False Positives.
- Example: a positive Covid test result would be a FALSE POSITIVE if you actually don't have Covid.



The 'Confusion' Matrix

TRUE NEGATIVE (TN)

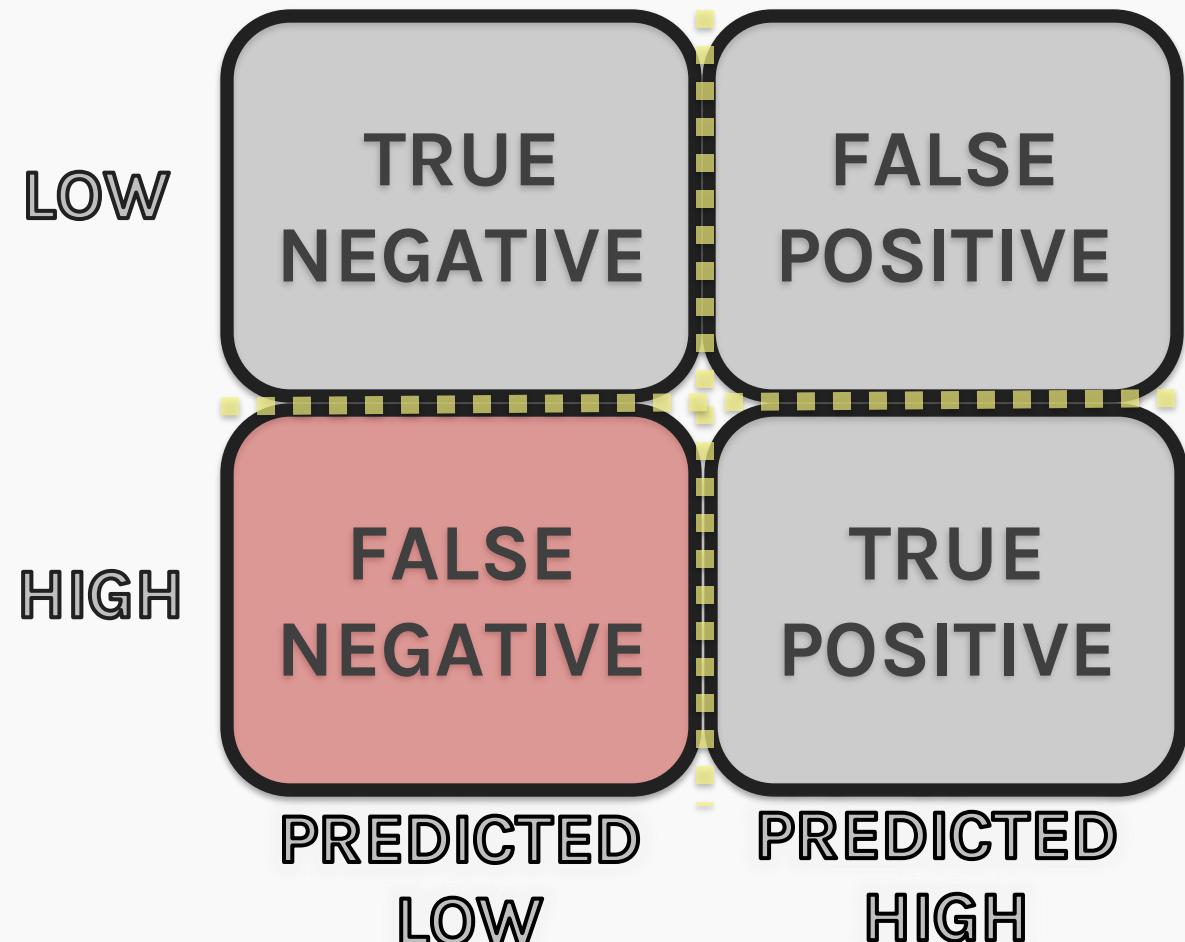
- Samples that are negative that the classifier predicts as negative are called True Negatives.
- Example: a negative Covid test result would be a TRUE NEGATIVE if you actually don't have Covid.



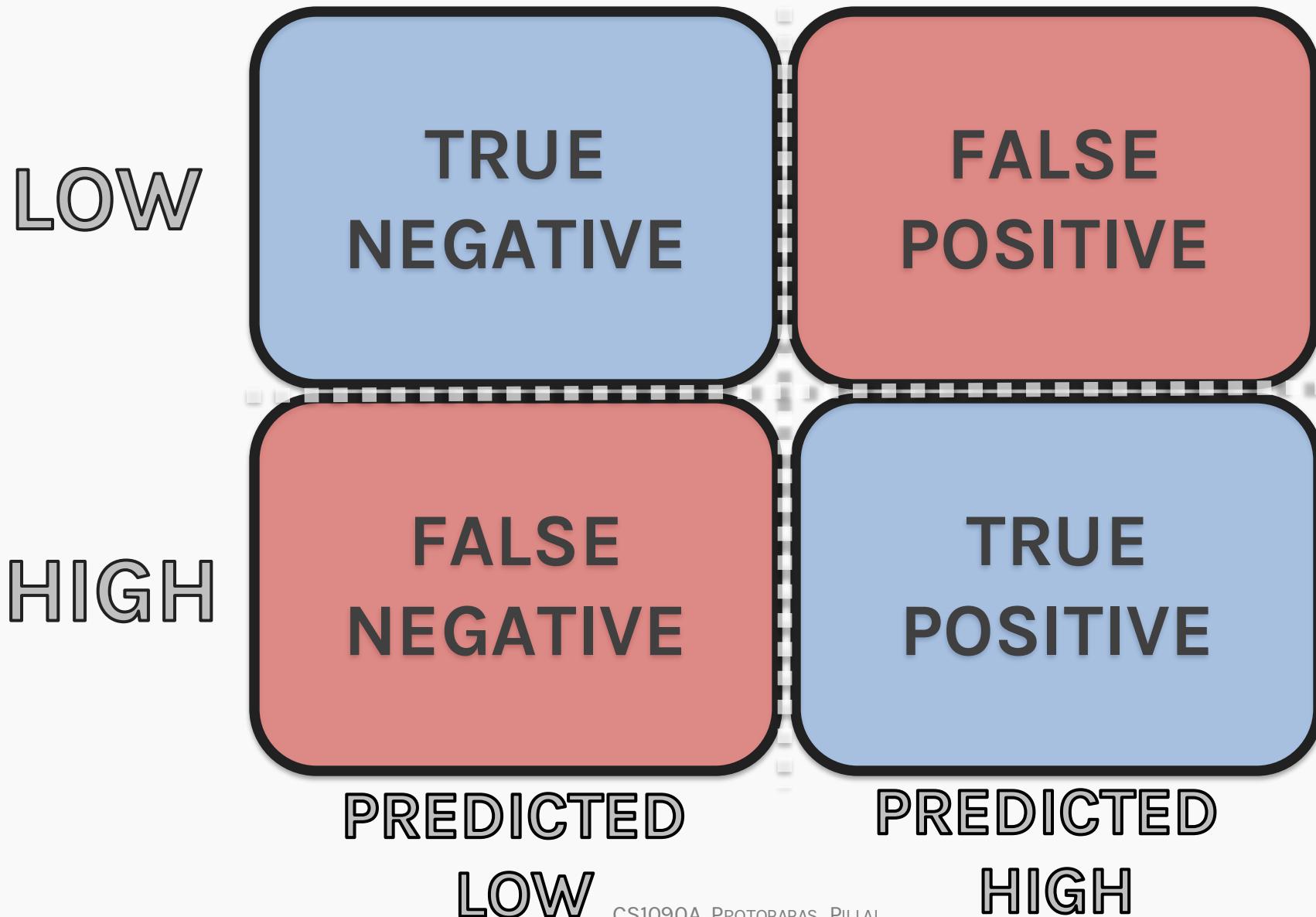
The 'Confusion' Matrix

FALSE NEGATIVE (FN)

- Samples that are negative that the classifier predicts as positive are called False Negatives.
- Example: a negative Covid test result would be a FALSE NEGATIVE if you actually have Covid.



The 'Confusion' Matrix



Confusion matrix

When a classification algorithm (like logistic regression) is used, the results can be summarized in a ($k \times k$) table as such:

| | Predicted no AHD ($\hat{Y} = 0$) | Predicted AHD ($\hat{Y} = 1$) |
|-----------------------------|---------------------------------------|------------------------------------|
| Truly no AHD ($Y = 0$) | 110 | 54 |
| Truly AHD ($Y = 1$) | 53 | 86 |

The table above was a classification based on a logistic regression model to predict AHD based on “3” predictors: X_1 = Age, X_2 = Sex, and X_3 = interaction between Age and Sex.

Bayes' Classifier Choice

A classifier's error rates can be tuned to modify this table. How?

The choice of the Bayes' classifier level will modify the characteristics of this table.

If we thought it was more important to predict AHD patients correctly (fewer false negatives), what could we do for our Bayes' classifier level?

We could classify instead based on:

$$\hat{P}(Y = 1) > \pi$$

and we could choose π to be some level other than 0.50.

Let's see what the table looks like if π were 0.40 or 0.60 instead. What should happen to the False Positive and False Negative frequencies?

Other Confusion tables

Based on $\pi = 0.4$:

| | Predicted no AHD ($\hat{Y} = 0$) | Predicted AHD ($\hat{Y} = 1$) |
|-----------------------------|---------------------------------------|------------------------------------|
| Truly no AHD ($Y = 0$) | 93 | 71 |
| Truly AHD ($Y = 1$) | 38 | 101 |

What has improved? What has worsened?

Based on $\pi = 0.6$:

| | Predicted no AHD ($\hat{Y} = 0$) | Predicted AHD ($\hat{Y} = 1$) |
|-----------------------------|---------------------------------------|------------------------------------|
| Truly no AHD ($Y = 0$) | 138 | 26 |
| Truly AHD ($Y = 1$) | 74 | 65 |

Which should we choose? Why?

Outline

- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- Multiclass Logistic Regression
 - Multinomial Logistic Regression
 - One-vs-Rest Logistic Regression
- Bayes Theorem and Misclassification Rates
- **ROC Curves**

ROC Curves

The Radio Operator Characteristics (ROC) curve illustrates the trade-off for all possible thresholds chosen for the two types of error (or correct classification).

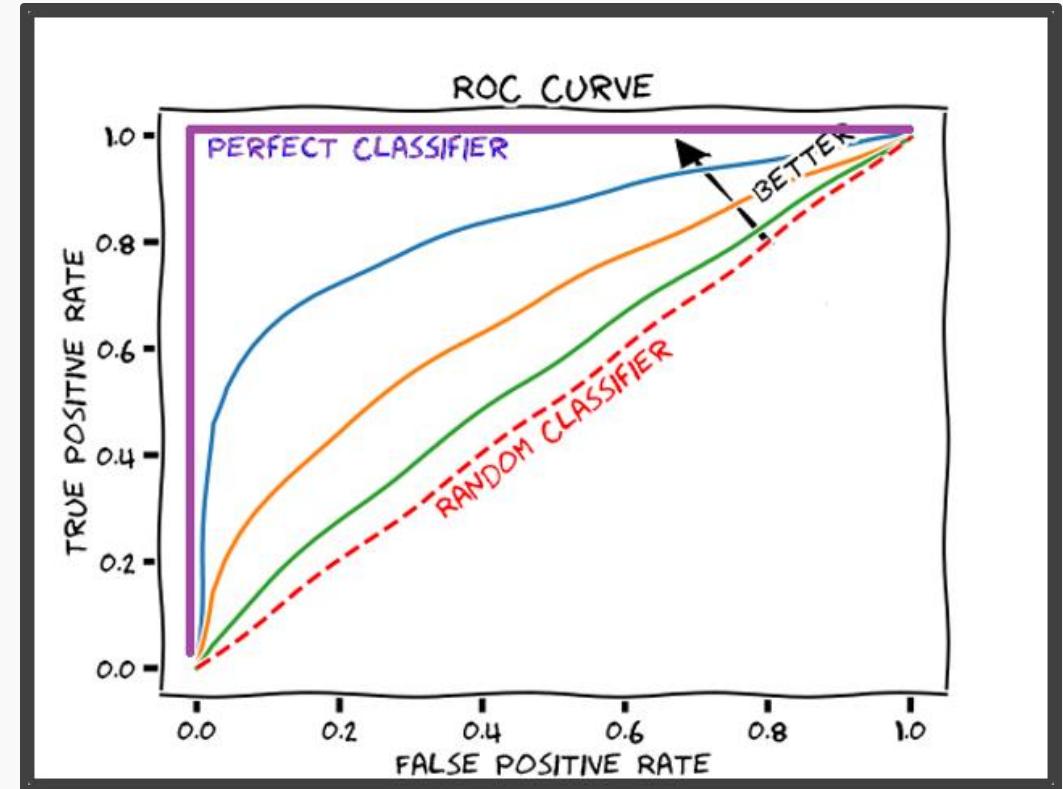
The vertical axis displays the true positive predictive value and the horizontal axis depicts the true negative predictive value.

What is the shape of an ideal ROC curve?

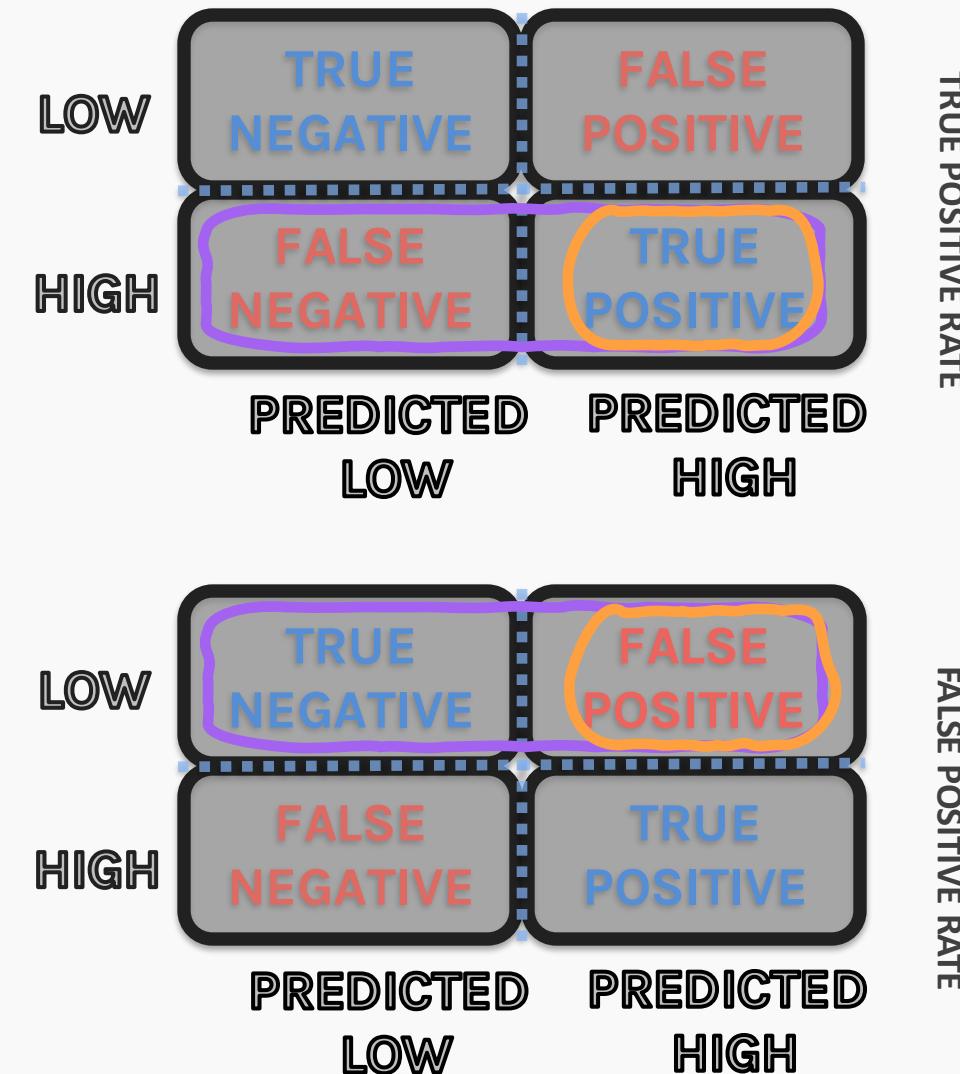
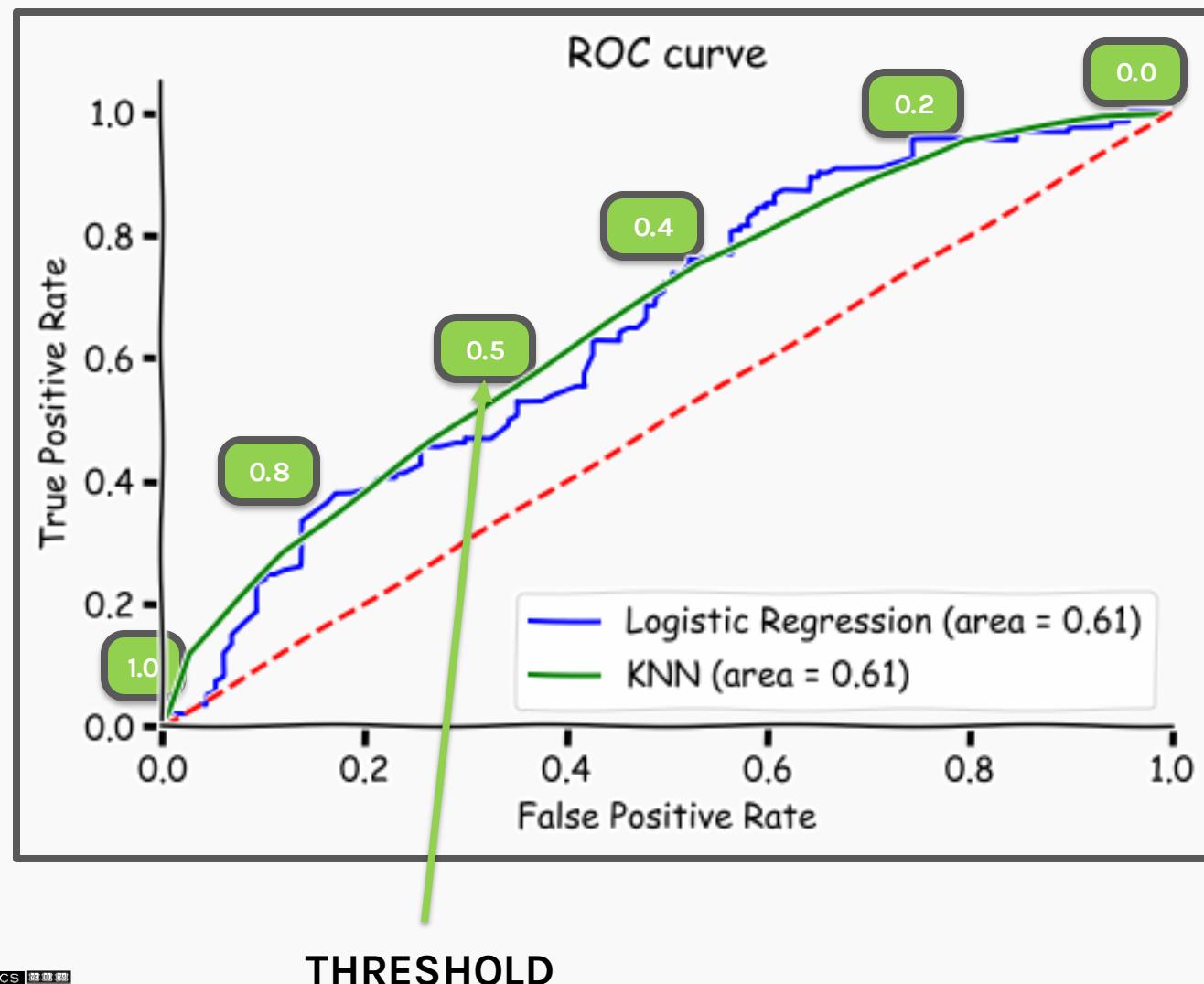
See next slide for an example.

Receiver Operating Characteristic curve (ROC)

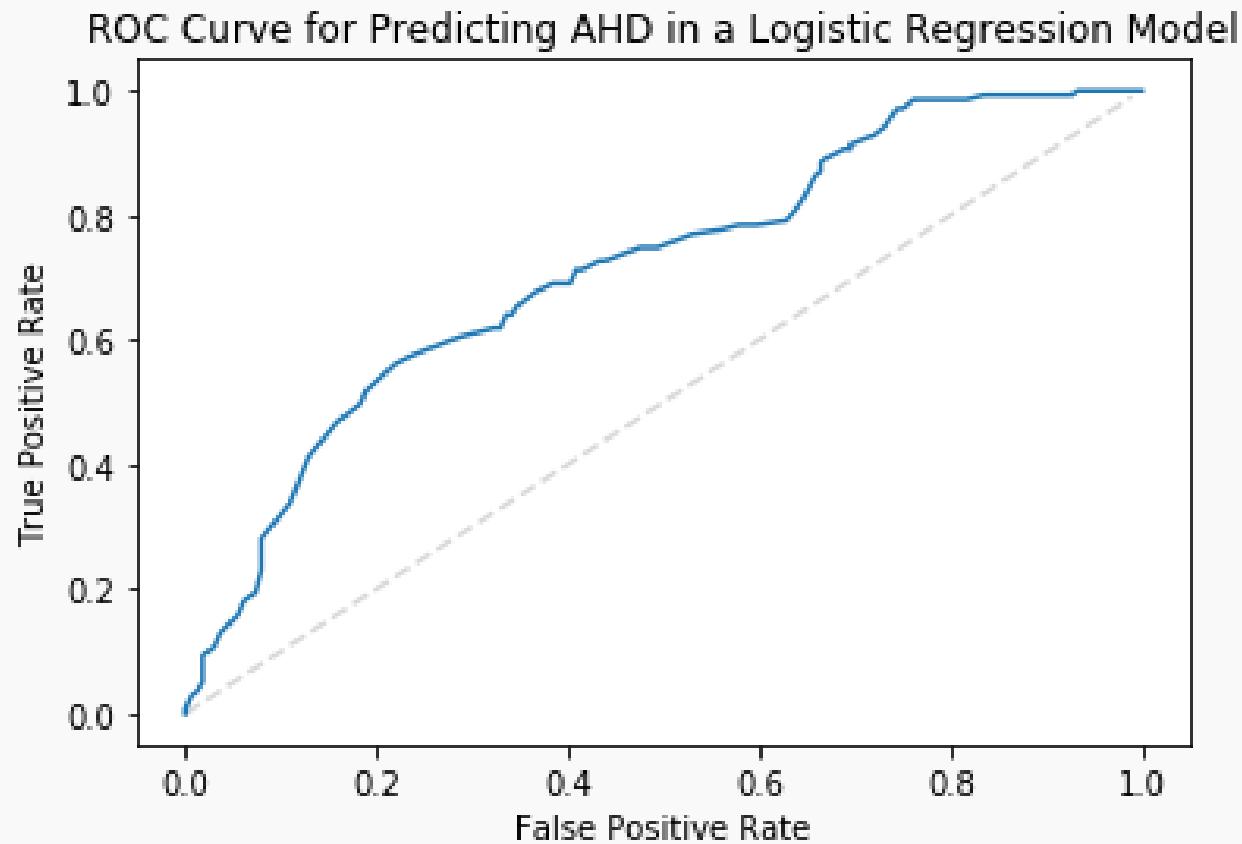
- The ROC curve was first developed by radar engineers during World War II for detecting enemy objects in battlefields.
- The ROC curve is created by plotting the **true positive rate (TPR)** against the **false positive rate (FPR)** at various threshold settings.



ROC curve for various thresholds



ROC Curve Example



AUC for measuring classifier performance

The overall performance of a classifier, calculated over all possible thresholds, is given by the **area under the ROC curve** (AUC).

An ideal ROC curve will hug the top left corner, so the larger the AUC the better the classifier.

What is the worst-case scenario for AUC? What is the best case?
What is AUC if we independently just flip a [biased] coin to perform classification?

AUC can be used to compare various approaches to classification:
Logistic regression, k -NN, Decision Trees (to come), etc.

Lecture 15: Causal Inference

CS1090A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai, and Chris Gumb



Lecture Outline

- Introduction
- Simpson's Paradox
- Causal Structure
- Correlation and causation
- Causal Effects
- Randomized Control
- Adjusting for Confounders

Association vs. Causation

In many of our methods (regression, for example) we often want to measure the association between two variables: the response, Y , and the predictor, X . For example, this association is modeled by a β coefficient in regression, or amount of increase in R^2 in a regression tree associated with a predictor, etc...

If β is *significantly different* from zero (or amount of R^2 is greater than by chance alone), then there is evidence that the response is associated with the predictor.

How can we determine if β is *significantly different* from zero in a model?

Association vs. Causation (cont.)

But what can we say about a **causal association**? That is, can we manipulate X in order to influence Y ?

Not necessarily. Why not?

There is potential for **confounding factors** to be the driving force for the observed association.

Controlling for confounding

How can we fix this issue of confounding variables?

There are 2 main approaches:

1. Model all possible confounders by including them into the model (multiple regression, for example). Or use sophisticated '**causal methods**' to account for the confounders.
2. A **randomized experiment** can be performed where the scientist manipulates the levels of the predictor (now called the **treatment**) to see how this leads to changes in the response.

What are the advantages and disadvantages of each approach?

Controlling for confounding: advantages/disadvantages

1. Modeling the confounders:

- **Advantages**: cheap
- **Disadvantages**: not all confounders may be measured.

2. Performing an experiment:

- **Advantages**: confounders will be **balanced**, on average, across treatment groups
- **Disadvantages**: expensive, can be an artificial environment

What is Causal Inference?

Options

- A. A special type of boosting method?
- B. The causal way of doing modeling?
- C. Finding the relationships between predictors and response variables
- D. Inferring the effects of any treatment/policy/intervention/etc

What is Causal Inference?

Options

- A. A special type of boosting method?
- B. The casual way of doing modeling?
- C. Finding the relationships between predictors and response variables
- D. Inferring the effects of any treatment/policy/intervention/etc

Which of these are examples of causal inference?

Options

- A. Effect of treatment on a disease
- B. Effect of social media on mental health
- C. Effect of climate change policy on emissions
- D. Effect of going to labs and lectures on final grade
- E. All of the above

Which of these are examples of causal inference?

Options

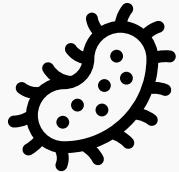
- A. Effect of treatment on a disease
- B. Effect of social media on mental health
- C. Effect of climate change policy on emissions
- D. Effect of going to labs and lectures on final grade
- E. All of the above

Lecture Outline

- Introduction
- **Simpson's Paradox**
- Causal Structure
- Correlation and causation
- Causal Effects
- Randomized Control
- Adjusting for Confounders

Simpson's paradox

Example: PyCA-109a



(Python Coding Anxiety)

Treatment T: A (Raderzole) and B (Gumboxin)

Condition Severity C: mild (0) or severe (1)

Outcome Y: alive (0) or dead (1)



Simpson's paradox

Overall Mortality Rate Table:

| Treatment | Total |
|-----------------|-----------------|
| A (Raderzole) | 19% 190/1000 |
| B (Gumboxin) | 20% 100/500 |

$$\mathbb{E}[Y|T]$$

Which treatment is best?

Options

- A. Raderzole because a smaller percentage of the people died
- B. Gumboxin because fewer people died
- C. This is an imbalanced dataset, I need more info
- D. What about the severity conditions of the people who took Raderzole or Gumboxin?

Which treatment is best?

Options

- A. Raderzole because a smaller percentage of the people died
- B. Gumboxin because fewer people died
- C. This is an imbalanced dataset, I need more info
- D. What about the severity conditions of the people who took Raderzole or Gumboxin?

Simpson's paradox

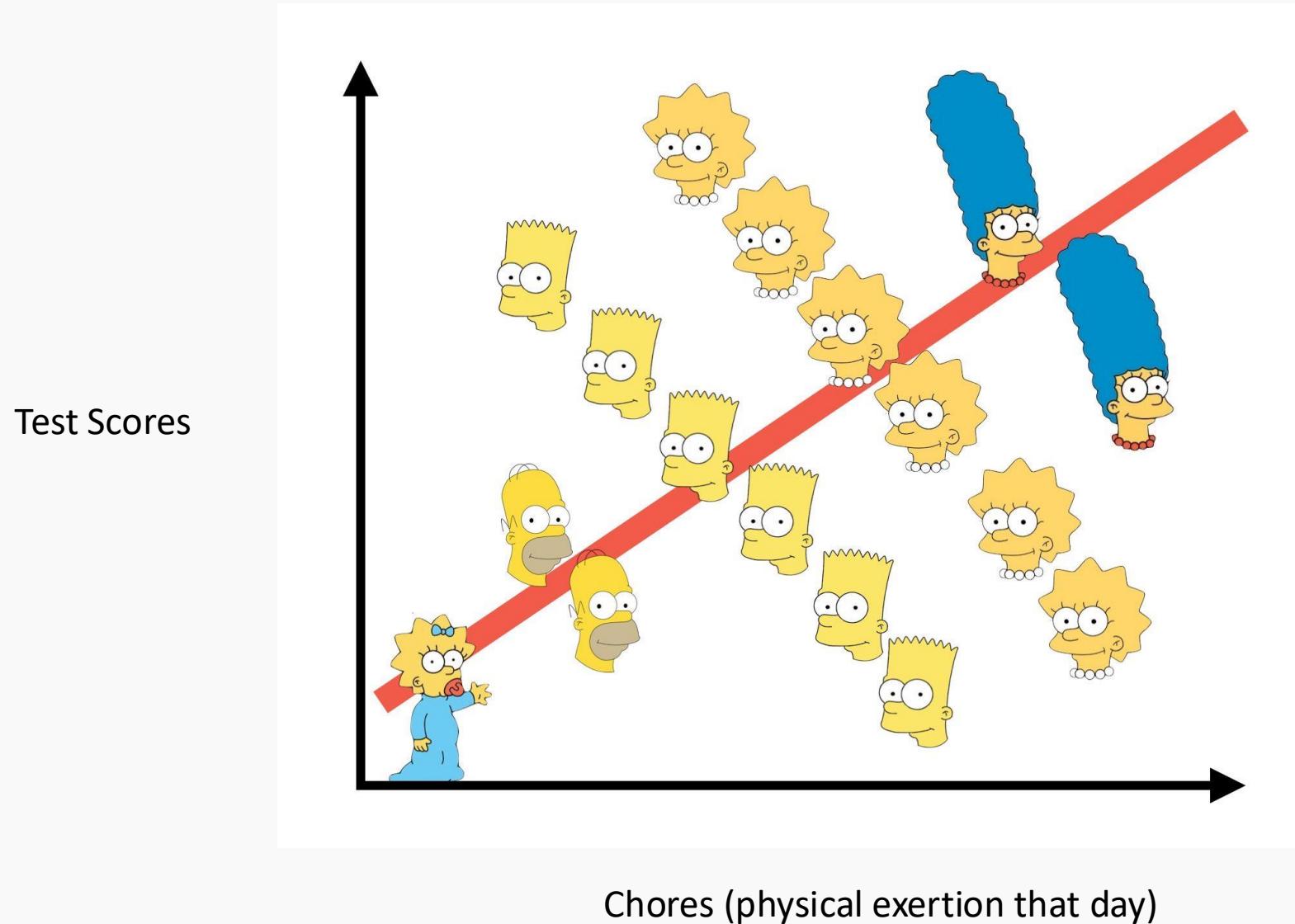


Mortality Rate Table

| | | Condition Severity | | |
|------------------|-----------|--------------------|--------|----------|
| Treatment | | Mild | Severe | Total |
| Treatment | Condition | | | |
| | | 17% | 40% | 19% |
| A (Raderzole) | Mild | 150/900 | 40/100 | 190/1000 |
| B (Gumboxin) | Severe | 50/350 | 50/150 | 100/500 |
| | Total | 20% | 33% | 20% |

$$\mathbb{E}[Y|T, C = 0] \quad \mathbb{E}[Y|T, C = 1] \quad \mathbb{E}[Y|T]$$

Simpson's paradox



Which treatment should we choose?

Options

- A. Raderzole
- B. Gumboxin
- C. It may still depend on the conditions

Which treatment should we choose?

Options

- A. Raderzole
- B. Gumboxin
- C. It may still depend on the conditions

Simpson's paradox

Why might it still depend on the condition?

| | | Condition Severity | | |
|------------------|----------------------|----------------------|------------------------|--|
| Treatment | Mild | Severe | Total | |
| A (Raderzole) | 17% 150/900 | 40% 40/100 | 19% 190/1000 | |
| B (Gumboxin) | 14% 50/350 | 33% 50/150 | 20% 100/500 | |

$$\mathbb{E}[Y|T, C = 0] \quad \mathbb{E}[Y|T, C = 1] \quad \mathbb{E}[Y|T]$$

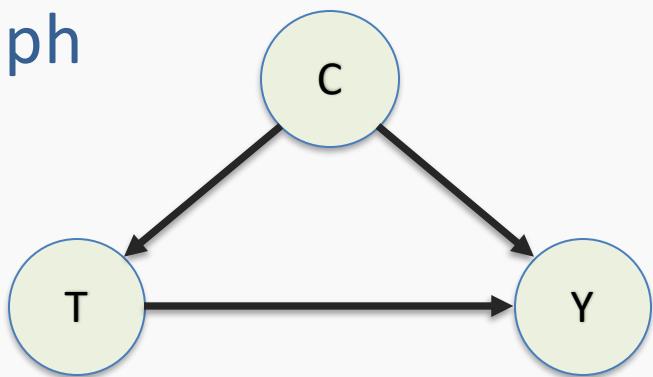
Lecture Outline

- Introduction
- Simpson's Paradox
- **Causal Structure**
- Correlation and causation
- Causal Effects
- Randomized Control
- Adjusting for Confounders

Causal Structure

| | Condition Severity | | |
|------------------|--------------------|---------------|-----------------|
| Treatment | Mild | Severe | Total |
| A (Raderzole) | 17% 150/900 | 40% 40/100 | 19% 190/1000 |
| B (Gumboxin) | 14% 50/350 | 33% 50/150 | 20% 100/500 |

Causal Graph

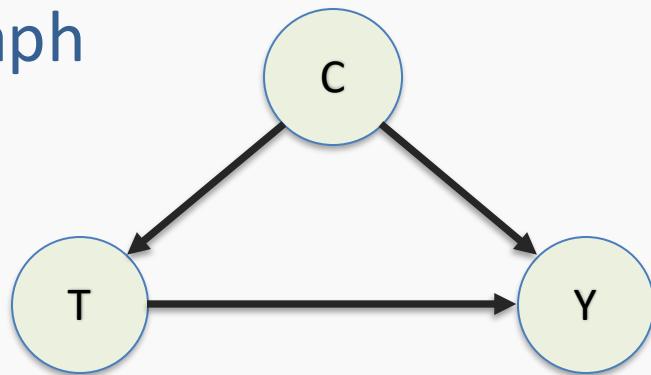


- Condition Severity is a cause of the treatment (leads to a diff treatment).
- Condition Severity and Treatment are both causes of the outcome

Causal Structure

| | Condition Severity | | |
|------------------|--------------------|---------------|-----------------|
| Treatment | Mild | Severe | Total |
| A (Raderzole) | 17% 150/900 | 40% 40/100 | 19% 190/1000 |
| B (Gumboxin) | 14% 50/350 | 33% 50/150 | 20% 100/500 |

Causal Graph



- Condition Severity is a cause of the treatment



Mild conditions



Visit doctor PPPP

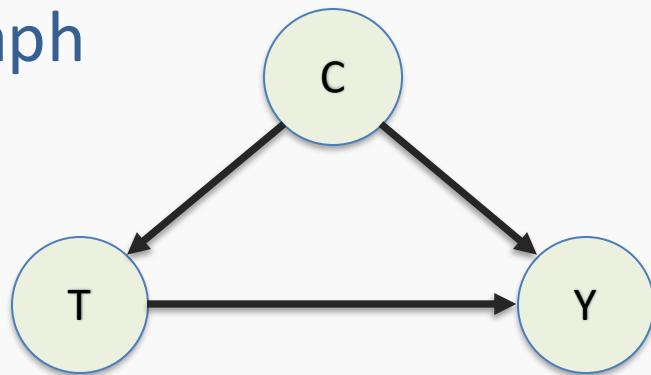


Treatment A (Raderzole) –
doctor wants to save
treatment B (Gumboxin) for
more severe conditions

Causal Structure

| | Condition Severity | | |
|------------------|--------------------|---------------|-----------------|
| Treatment | Mild | Severe | Total |
| A (Raderzole) | 17% 150/900 | 40% 40/100 | 19% 190/1000 |
| B (Gumboxin) | 14% 50/350 | 33% 50/150 | 20% 100/500 |

Causal Graph



- Condition Severity is a cause of the treatment



Mild conditions



Visit doctor PPPP

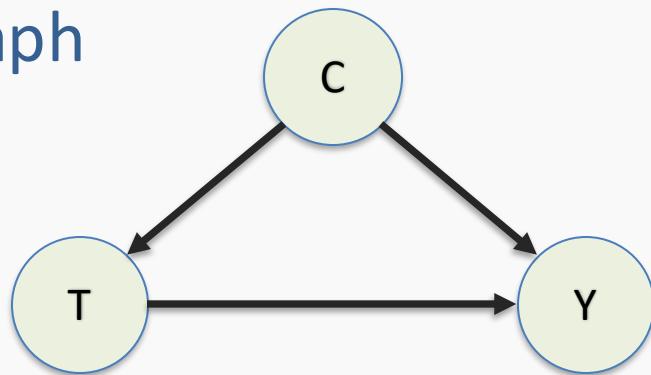


Treatment A (Raderzole) –
doctor wants to save
treatment B (Gumboxin) for
more severe conditions

Causal Structure

| | Condition Severity | | |
|------------------|--------------------|---------------|-----------------|
| Treatment | Mild | Severe | Total |
| A (Raderzole) | 17% 150/900 | 40% 40/100 | 19% 190/1000 |
| B (Gumboxin) | 14% 50/350 | 33% 50/150 | 20% 100/500 |

Causal Graph



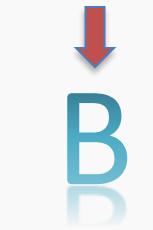
- Condition Severity is a cause of the treatment



Severe conditions



Visit doctor PPPP

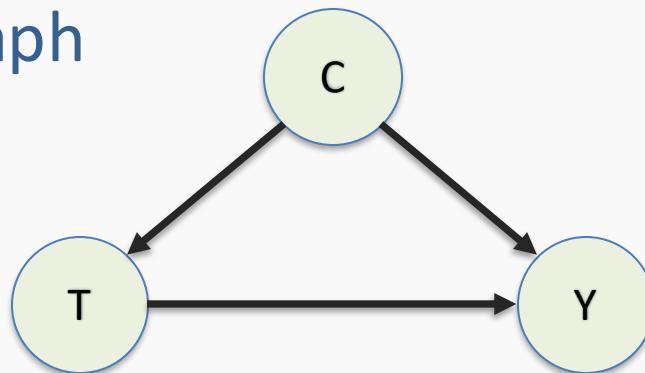


Treatment B (Gumboxin) –
doctor knows Gumboxin is
more effective

Causal Structure

| | Condition Severity | | |
|------------------|----------------------|------------------------------|------------------------|
| Treatment | Mild | Severe | Total |
| A (Raderzole) | 17% 150/900 | 40% 40/ 100 | 19% 190/1000 |
| B (Gumboxin) | 14% 50/350 | 33% 50/ 150 | 20% 100/500 |

Causal Graph



- Condition Severity is a cause of the treatment



Severe conditions



Visit doctor PPPP



Treatment B (Gumboxin) –
doctor knows Gumboxin is
more effective

Which treatment should we choose?

Options

- A. Treatment A (Raderzole)
- B. Treatment B (Gumboxin)

Which treatment should we choose?

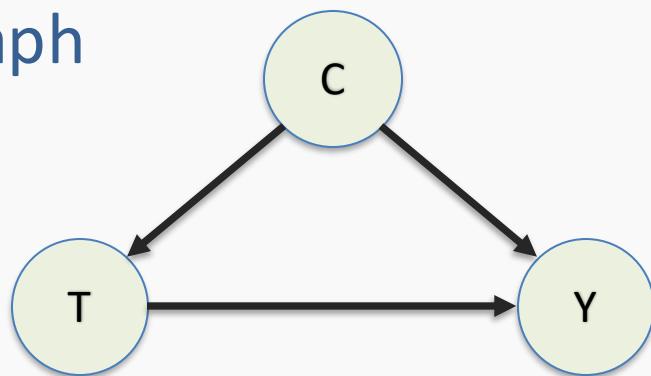
Options

- A. Treatment A (Raderzole)
- B. Treatment B (Gumboxin) -- because the subgroups give lower mortality rate

Causal Structure

| | Condition Severity | | |
|------------------|--------------------|---------------|-----------------|
| Treatment | Mild | Severe | Total |
| A (Raderzole) | 17% 150/900 | 40% 40/100 | 19% 190/1000 |
| B (Gumboxin) | 14% 50/350 | 33% 50/150 | 20% 100/500 |

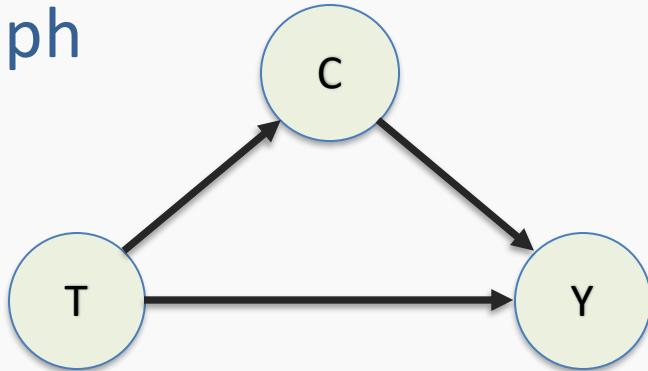
Causal Graph



Causal Structure

| | Condition Severity | | |
|------------------|--------------------|---------------|-----------------|
| Treatment | Mild | Severe | Total |
| A (Raderzole) | 17% 150/900 | 40% 40/100 | 19% 190/1000 |
| B (Gumboxin) | 14% 50/350 | 33% 50/150 | 20% 100/500 |

Causal Graph



- Treatment is a cause of the Condition Severity



Visit doctor PPPP



B



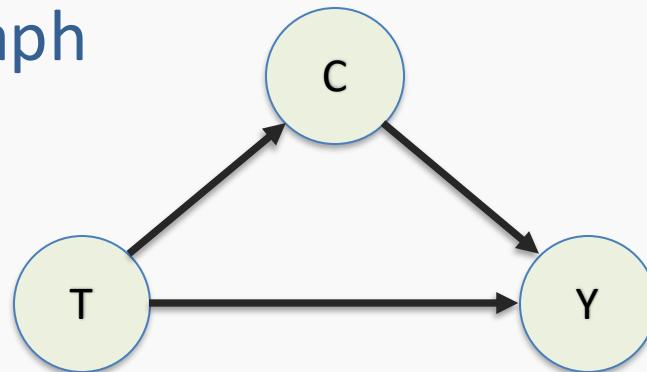
Treatment B (Gumboxin).
Takes long to get and while
waiting conditions become
severe

Conditions become severe

Causal Structure

| | Condition | | |
|------------------|----------------|---------------|-----------------|
| Treatment | Mild | Severe | Total |
| A (Raderzole) | 17% 150/900 | 40% 40/100 | 19% 190/1000 |
| B (Gumboxin) | 14% 50/350 | 33% 50/150 | 20% 100/500 |

Causal Graph



- Treatment is a cause of the Condition Severity



Visit doctor

A

Treatment A (Raderzole). No wait time



Conditions never becomes severe, remains mild

Treatment causes people to have mild or severe conditions

Which treatment should we choose?

Options

- A. Treatment A (Raderzole)
- B. Treatment B (Gumboxin)

Which treatment should we choose?

Options

- A. Treatment A (Raderzole) because B (Gumboxin) makes you have severe conditions
- B. Treatment B (Gumboxin)

Lecture Outline

- Introduction
- Simpson's Paradox
- Causal Structure
- **Correlation and causation**
- Causal Effects
- Randomized Control
- Adjusting for Confounders

Correlation does not imply causation

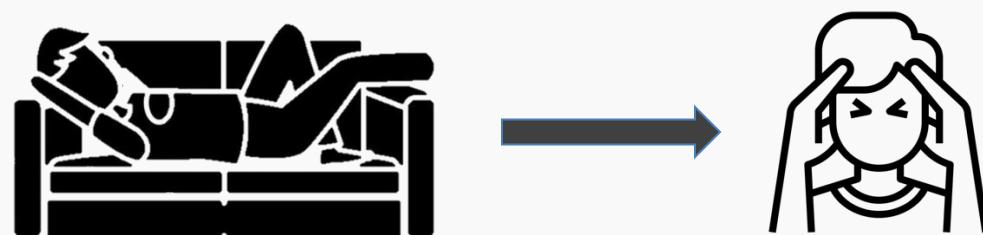
Example:

People who sleep fully dressed wake up with headaches

Correlation does not imply causation

Example:

Sleeping fully dressed correlates with waking up with headaches

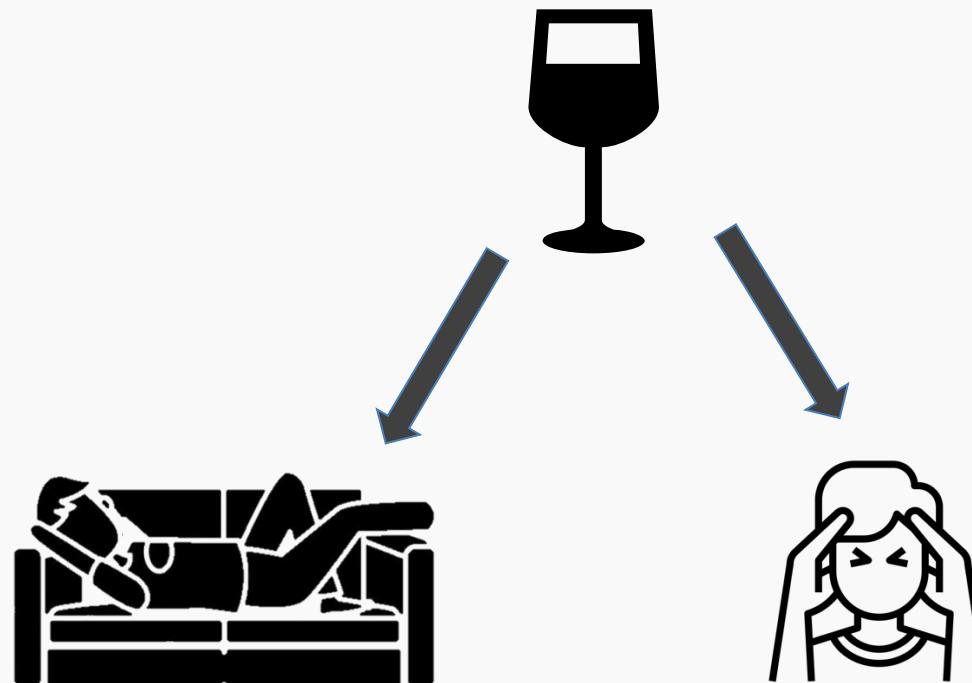


Correlation does not imply causation

Example:

Sleeping fully dressed correlates with waking up with headaches

Common cause: drinking the night before



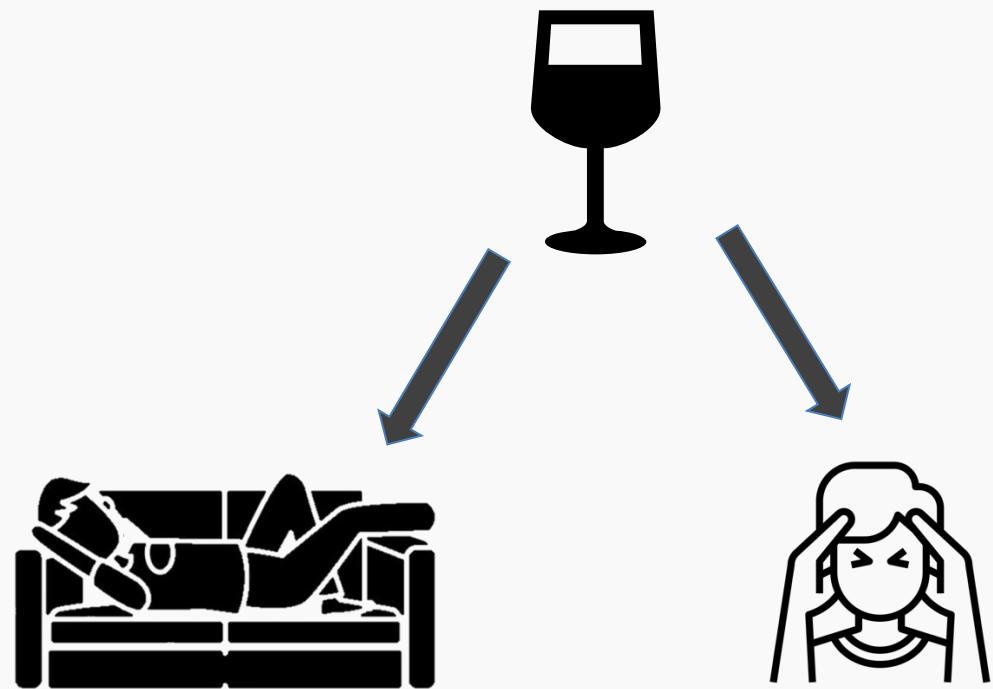
Example:

Sleeping fully dressed correlates with waking up with headaches

Common cause: drinking the night before

Two type of people in this world:

1. Dressed up sleepers
2. Non-dressed up sleepers



Sleeping fully dressed correlates with waking up with headaches

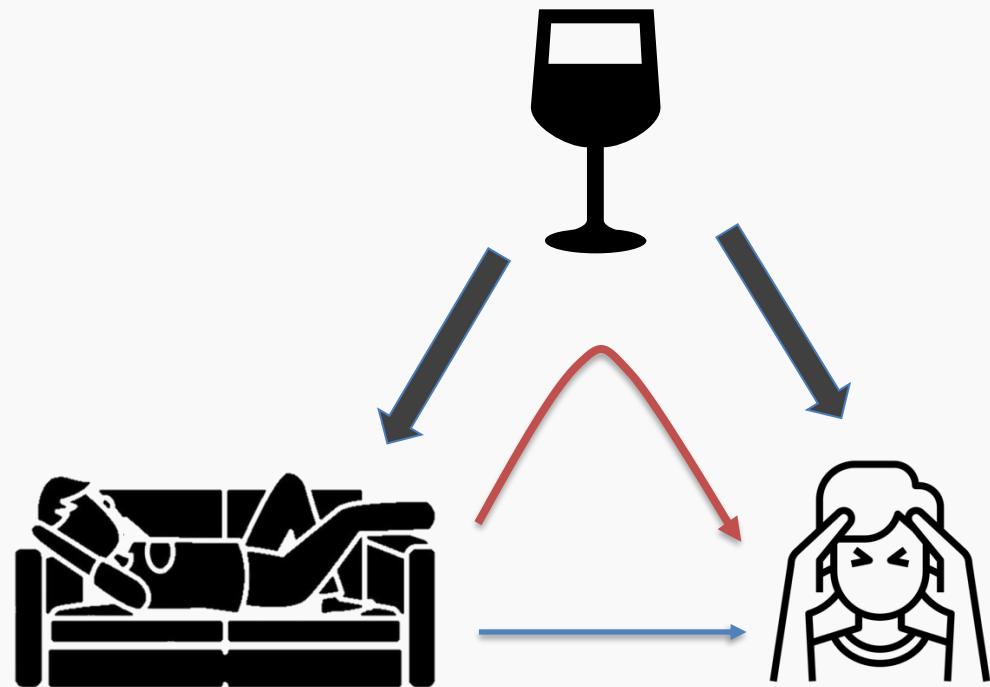
Common cause: drinking the night before

Two type of people in this world:

1. Dressed up sleepers
2. Non-dressed up sleepers

Confounding Association ->

Causal Association ->



What is association and what is correlation?

Options

- A. They are the same describing relationship of two variables
- B. Association is a statistical dependance where correlation is ML dependance
- C. Association is a statistical dependance and correlation a linear statistical dependence
- D. Association is a statistical dependance and correlation a linear statistical dependence but we used them interchangeably

What is association and what is correlation?

Options

- A. They are the same describing relationship of two variables
- B. Association is a statistical dependance where correlation is ML dependance
- C. Association is a statistical dependance and correlation a linear statistical dependence (aka, direction)
- D. Association is a statistical dependance and correlation a linear statistical dependence but we used them interchangeably

Sleeping fully dressed correlates with waking up with headaches

Common cause: drinking the night before

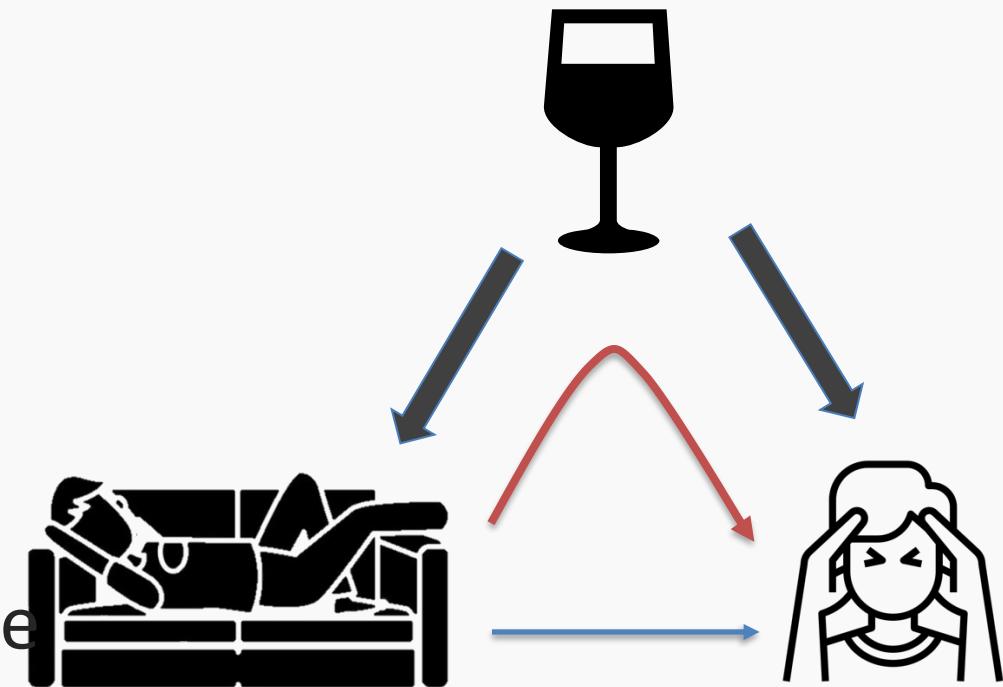
Two type of people in this world:

1. Dressed up sleepers
2. Non-dressed up sleepers

Confounding Association ->

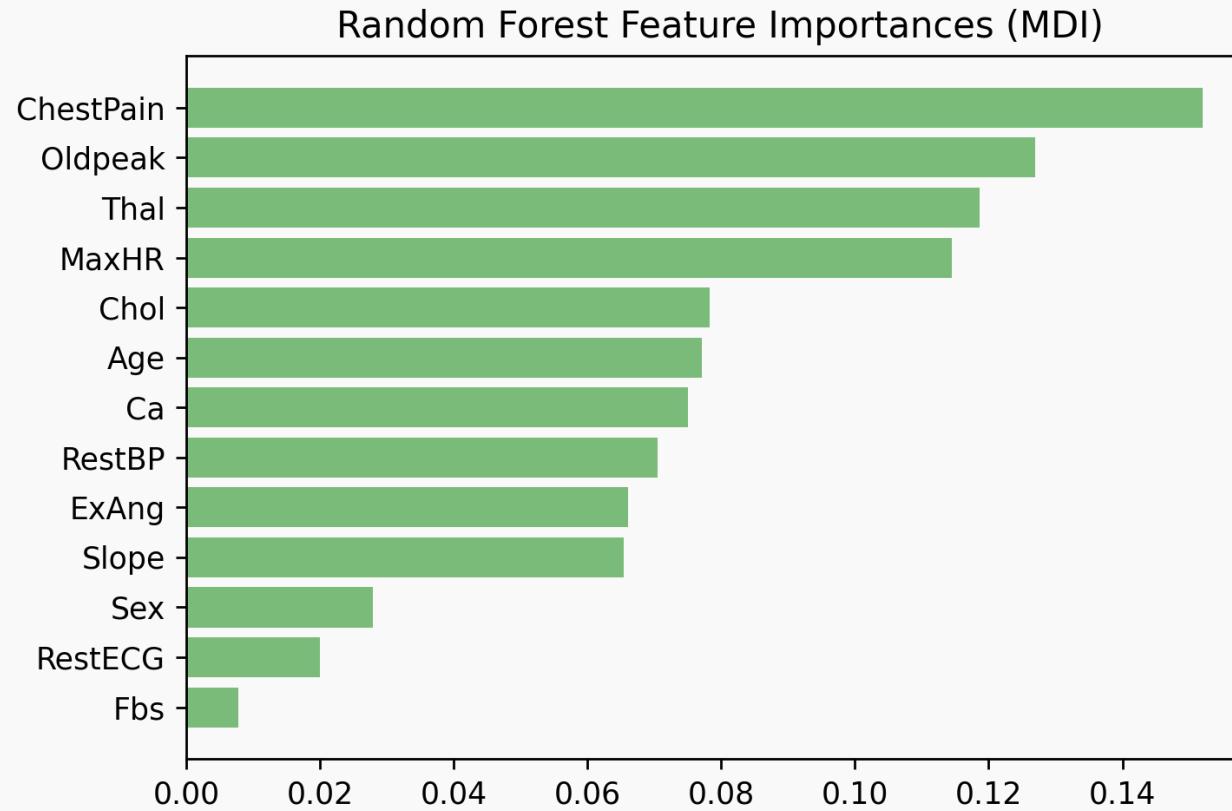
Causal Association ->

If we measure correlation, we see a mixture
of causal and confounding



Correlation = causation is something we do all the time

Sometimes because we have in our minds, **cognitive bias**



Correlation = causation is something we do all the time

Sometimes because it is convenient:

If correlation does not imply causation, then what does imply causation

Options

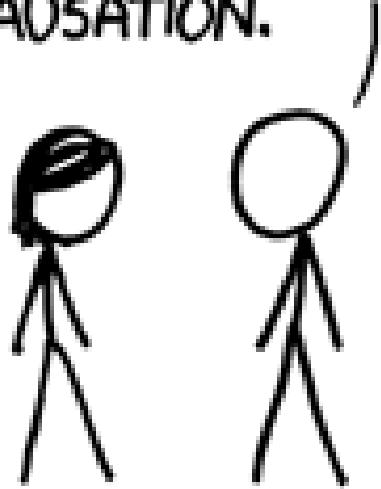
- A. Anything with correlation above 0.9 is a causation
- B. Any non-linear correlation
- C. To know if something causes something else

If correlation does not imply causation, then what does imply causation

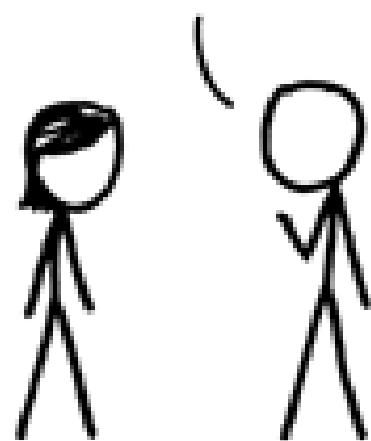
Options

- A. Anything with correlation above 0.9 is a causation
- B. Any non-linear correlation
- C. To know if something causes something else. We need to figure out, how to identify and measure it

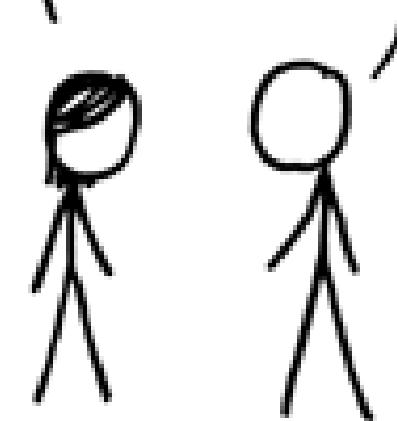
I USED TO THINK
CORRELATION IMPLIED
CAUSATION.



THEN I TOOK A
STATISTICS CLASS.
NOW I DON'T.



SOUNDS LIKE THE
CLASS HELPED.
WELL, MAYBE.

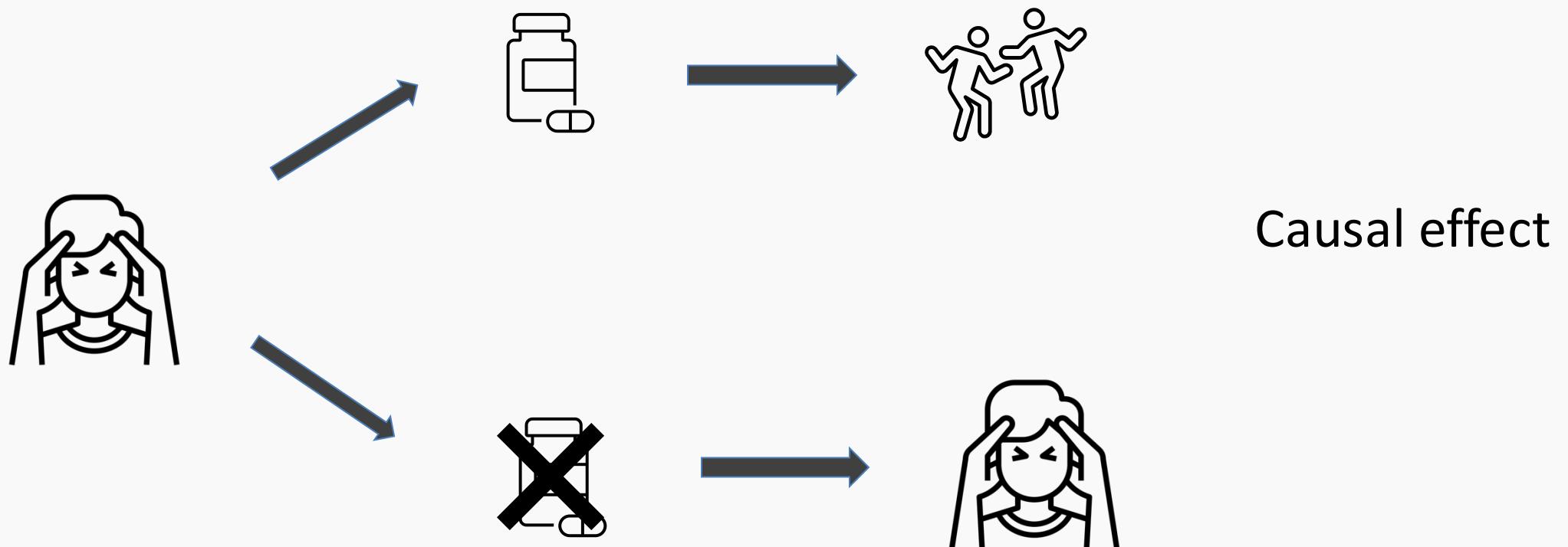


Lecture Outline

- Introduction
- Simpson's Paradox
- Causal Structure
- Correlation and causation
- **Causal Effects (formalism)**
- Randomized Control

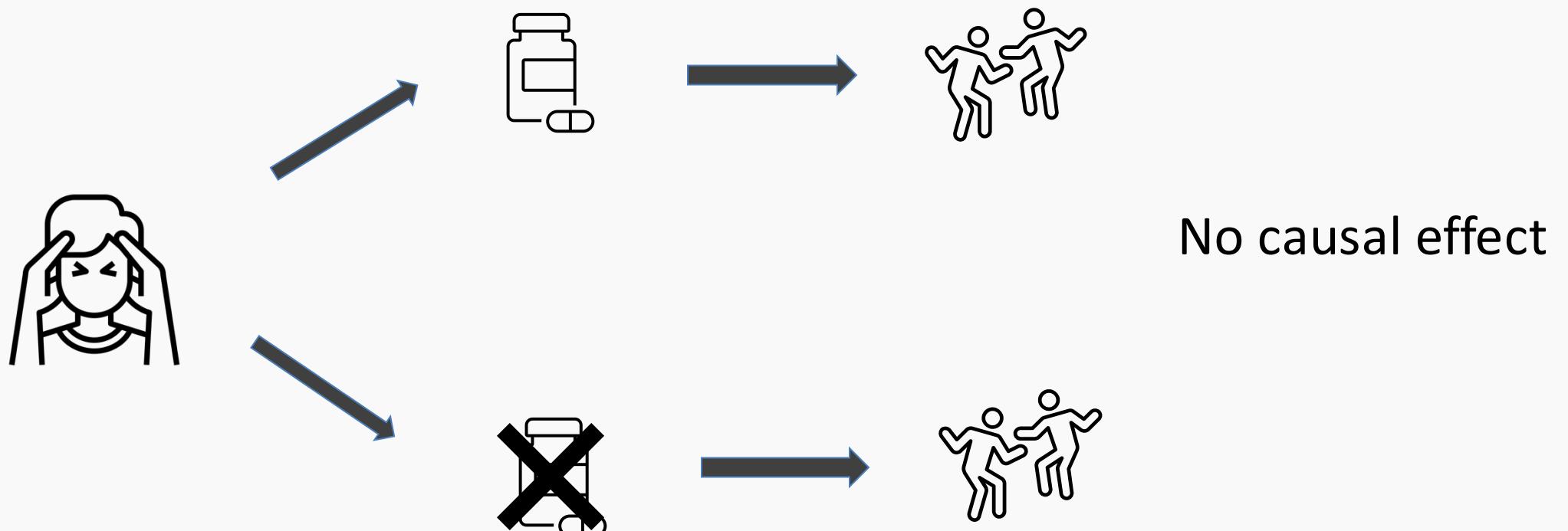
Potential outcomes

Inferring the effect of treatment on some outcome

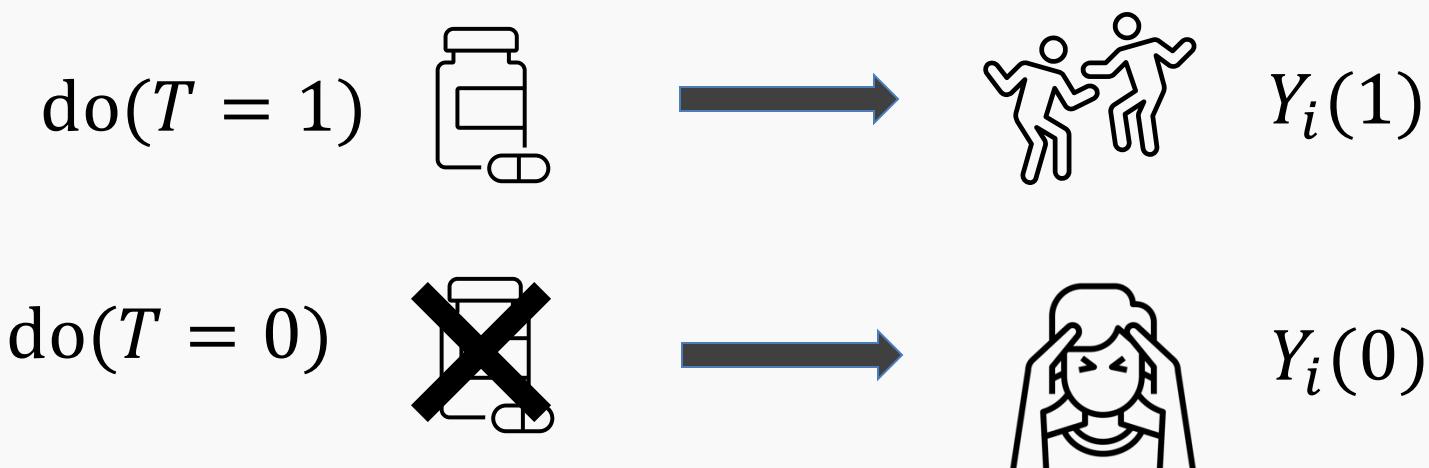


Potential outcomes

Inferring the effect of treatment on some outcome



Some notation

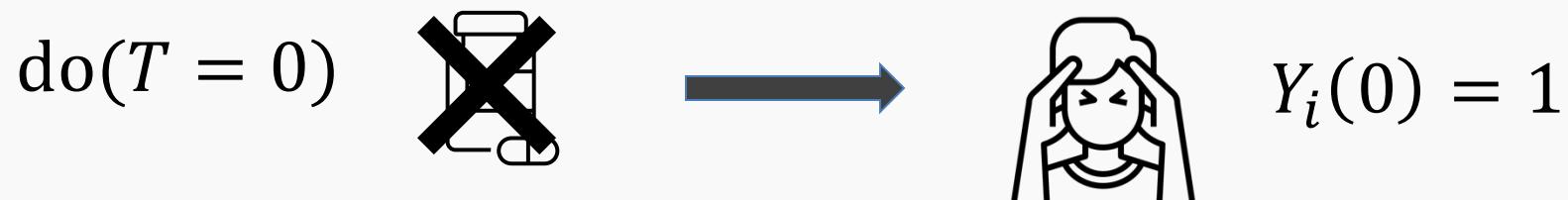
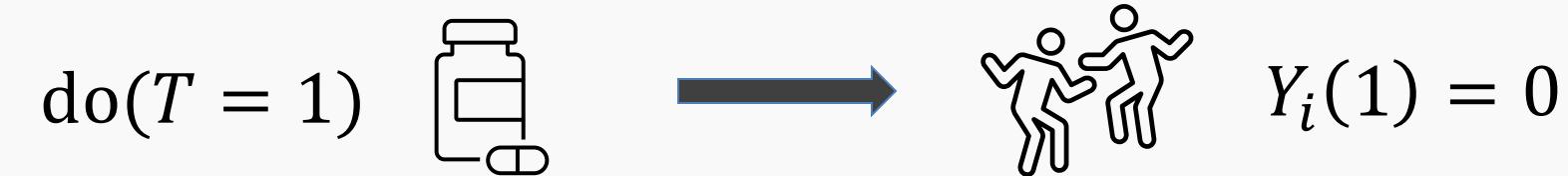


Causal effect

$$Y_i(1) - Y_i(0)$$

T : treatment
 Y : observed outcome
 i : denotes individual/observation
 $Y_i(T)$: potential outcome under treatment T

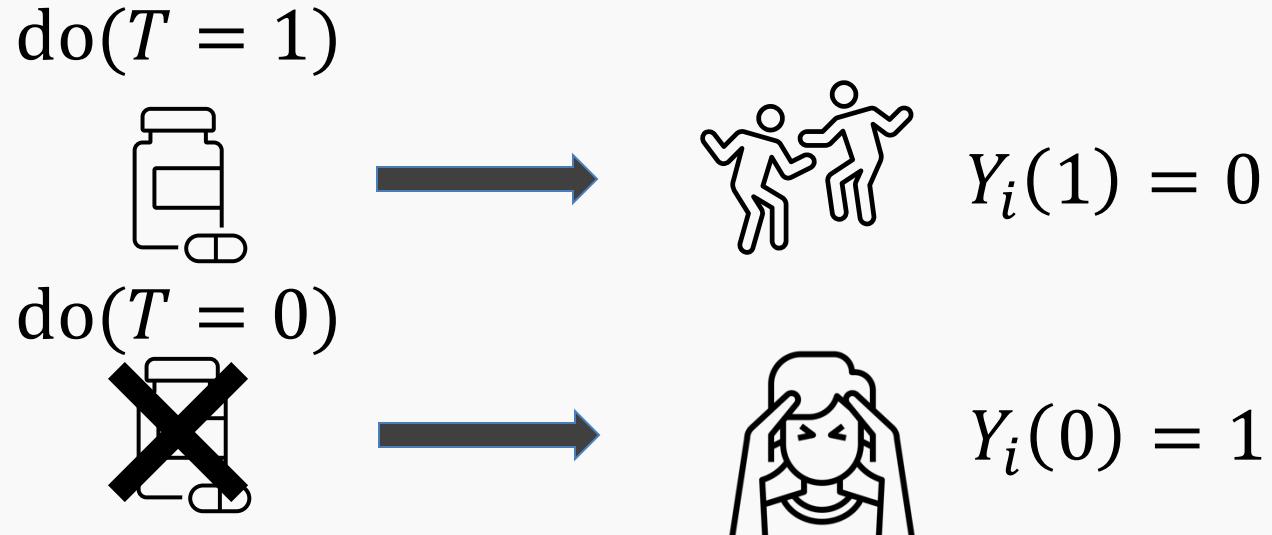
Fundamental problem of causal inference



Causal effect

$$Y_i(1) - Y_i(0) = -1$$

Fundamental problem of causal inference

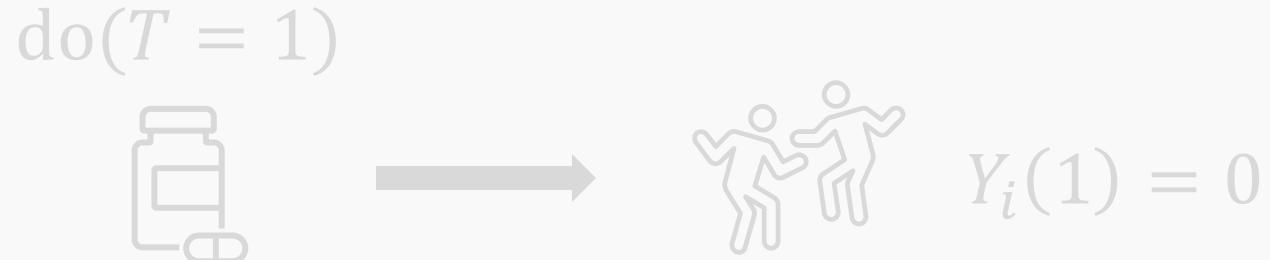


Causal effect

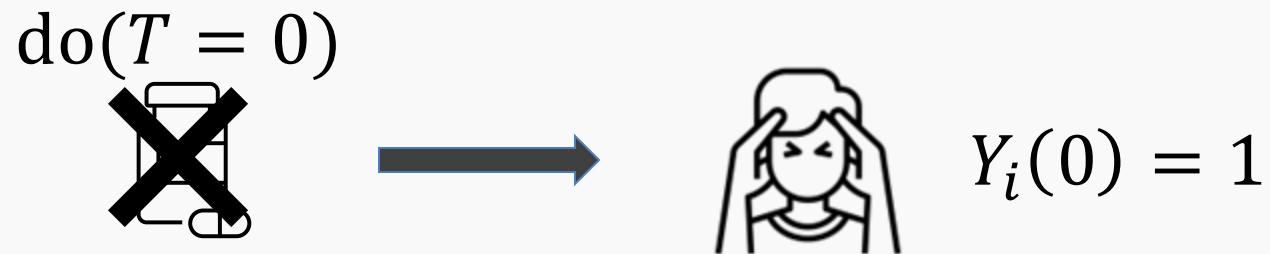
$$Y_i(1) - Y_i(0) = -1$$

Fundamental problem of causal inference

Counterfactual



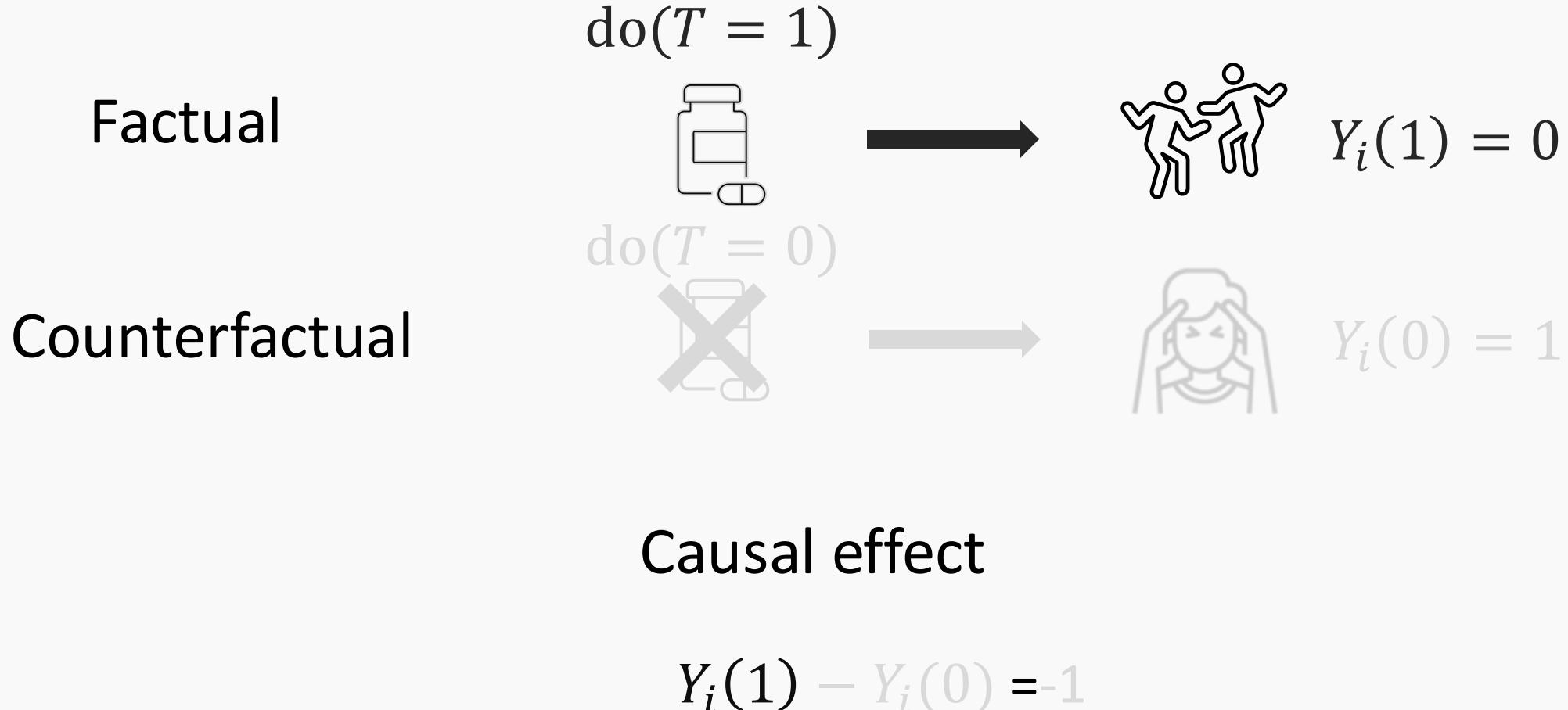
Factual



Causal effect

$$Y_i(1) - Y_i(0) = -1$$

Fundamental problem of causal inference



How do we avoid the fundamental problem of causal inference

Options

- A. We will do what we usually do: ignore all problems unless they affect my life
- B. Guess the counterfactual effect
- C. Infer the counterfactual effect using some imputation method
- D. Take the averages

How do we avoid the fundamental problem of causal inference

Options

- A. We will do what we usually do: ignore all problems unless they affect my life
- B. Guess the counterfactual effect
- C. Infer the counterfactual effect using some imputation method
- D. Take the averages

Average Treatment Effect

Individual treatment effect (ITE):

$$Y_i(1) - Y_i(0)$$

Average treatment effect (ATE):

$$\begin{aligned}\mathbb{E}[Y_i(1) - Y_i(0)] &= \mathbb{E}[Y(1)] - \mathbb{E}[Y(0)] \\ &\neq \mathbb{E}[Y|T = 1] - \mathbb{E}[Y|T = 0]\end{aligned}$$

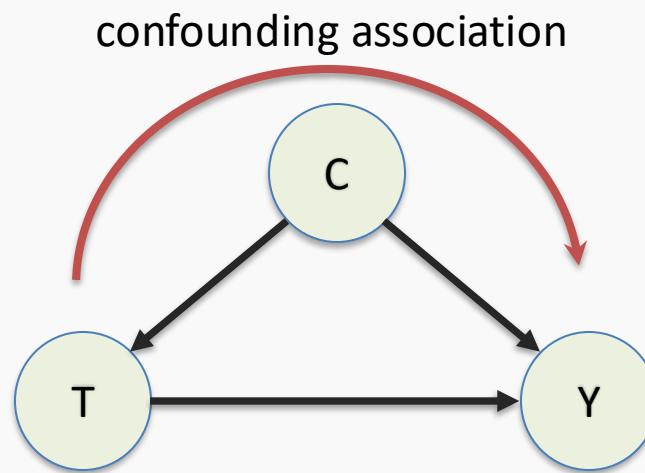
Average Treatment Effect

Individual treatment effect (ITE):

$$Y_i(1) - Y_i(0)$$

Average treatment effect (ATE):

$$\begin{aligned}\mathbb{E}[Y_i(1) - Y_i(0)] &= \mathbb{E}[Y(1)] - \mathbb{E}[Y(0)] \\ &\neq \mathbb{E}[Y|T = 1] - \mathbb{E}[Y|T = 0]\end{aligned}$$



Average Treatment Effect

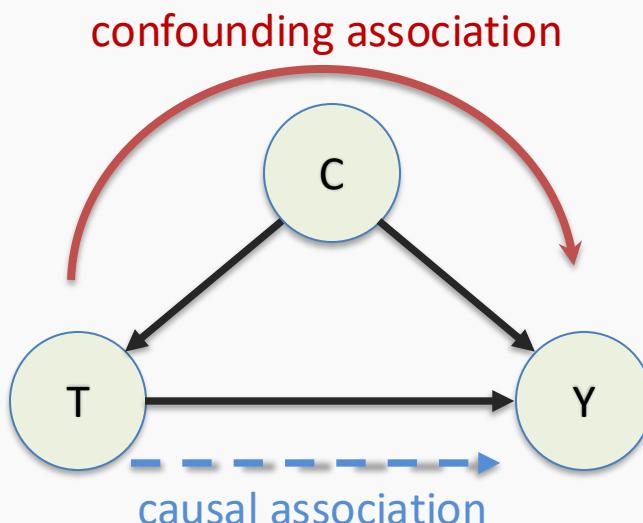
Individual treatment effect (ITE):

$$Y_i(1) - Y_i(0)$$

Average treatment effect (ATE):

$$\begin{aligned}\mathbb{E}[Y_i(1) - Y_i(0)] &= \mathbb{E}[Y(1)] - \mathbb{E}[Y(0)] \\ &\neq \mathbb{E}[Y|T = 1] - \mathbb{E}[Y|T = 0]\end{aligned}$$

Mix of
confounding
and causal



Necessary Assumptions

- In order to measure the causal effect of treatment (T) on the outcome (Y), we need to make a major assumption:
- **Stable Unit Treatment Values Assumption (SUTVA)**: the response/outcome of a particular subject depends only on the treatment to which they were assigned, not the treatments of others around them.
- How can this be violated?
 - If there are any **spillover effects** between subjects (sometimes called **interference**): if one subject's treatment effects another subject's response (or treatment). Think: individuals within a household.
 - If the treatment is not consistent (sometime called **hidden variations of treatments**): if a subject's treatment *level* may be different the second time around. Think: a bad batch of pills.
Note: this is different than heterogeneous treatment *effects*.

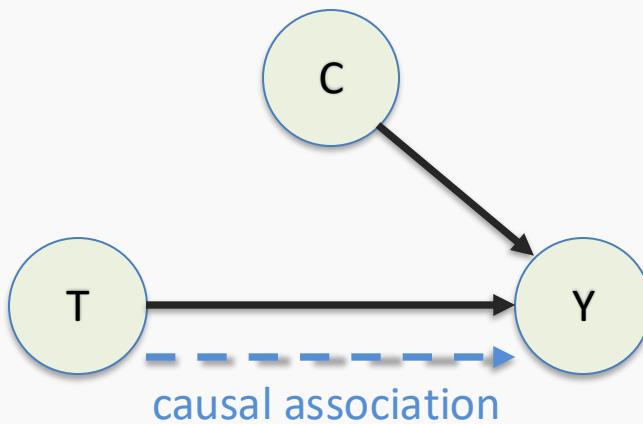
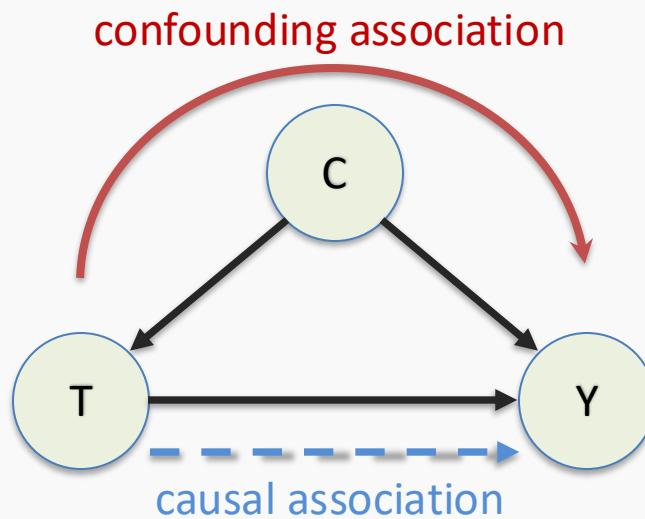
Lecture Outline

- Introduction
- Simpson's Paradox
- Causal Structure
- Correlation and causation
- Causal Effects
- **Randomized Control**
- Adjusting for Confounders

Randomized control trials (RCT)

Average treatment effect (ATE) with confounding:

$$\mathbb{E}[Y(1)] - \mathbb{E}[Y(0)] \neq \mathbb{E}[Y|T = 1] - \mathbb{E}[Y|T = 0]$$



$$\mathbb{E}[Y(1)] - \mathbb{E}[Y(0)] = \mathbb{E}[Y|T = 1] - \mathbb{E}[Y|T = 0]$$

Randomized control trials (RCTs):

Randomize subjects into treatment group or control group

1. Treatment does not have any causal parents
2. The groups now are equal

Can we always do RCTs?

Options

- A. Yes, it is easy, just randomize the treatments
- B. No, because sometimes we can not control the experiment
- C. No, because sometimes it is not ethical
- D. No, because sometimes it is not feasible
- E. No, because sometimes it is not possible
- F. RCTs are too expensive

Can we always do RCTs?

Options

- A. Yes, it is easy, just randomize the treatments
- B. No, because sometimes we can not control the experiment
- C. No, because sometimes it is not ethical
- D. No, because sometimes it is not feasible
- E. No, because sometimes it is not possible
- F. RCTs are too expensive

Lecture Outline

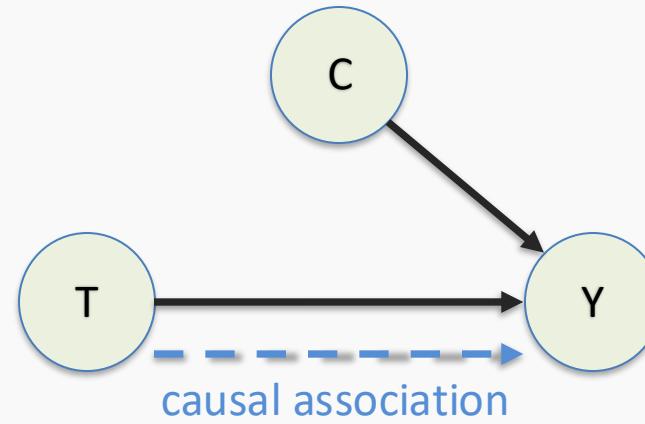
- Introduction
- Simpson's Paradox
- Causal Structure
- Correlation and causation
- Causal Effects
- Randomized Control
- **Adjusting for Confounders**

Observational Studies

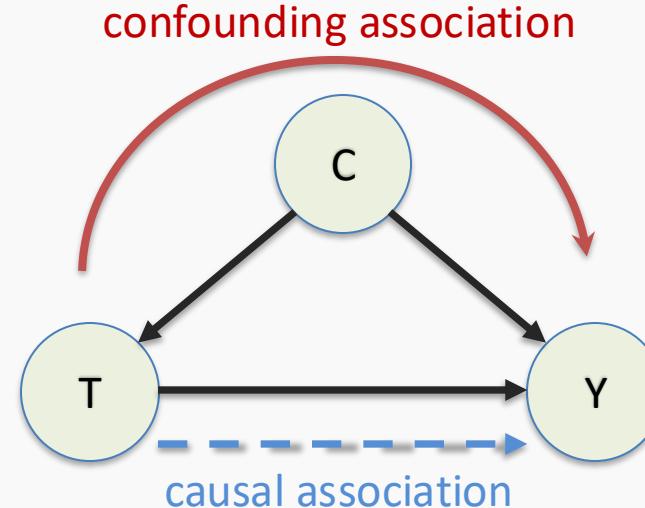
Can't always perform RCT

- Can't randomize
- Ethical
- Not feasible
- Not possible
- RCTs are expensive and difficult to set

Ideal



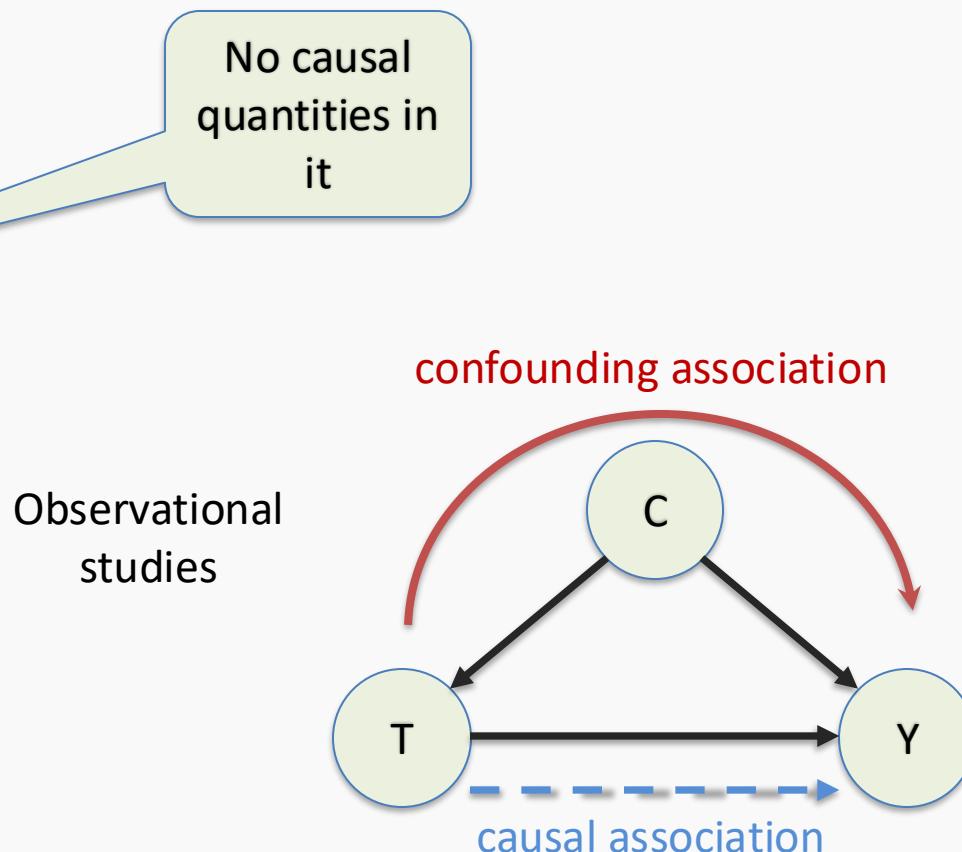
Observational studies



Measuring a Causal Effect in Observational Studies

Adjust for confounders. Or adjust for a variable W . If W is the right variable: in this case $W = C$ (condition severity).

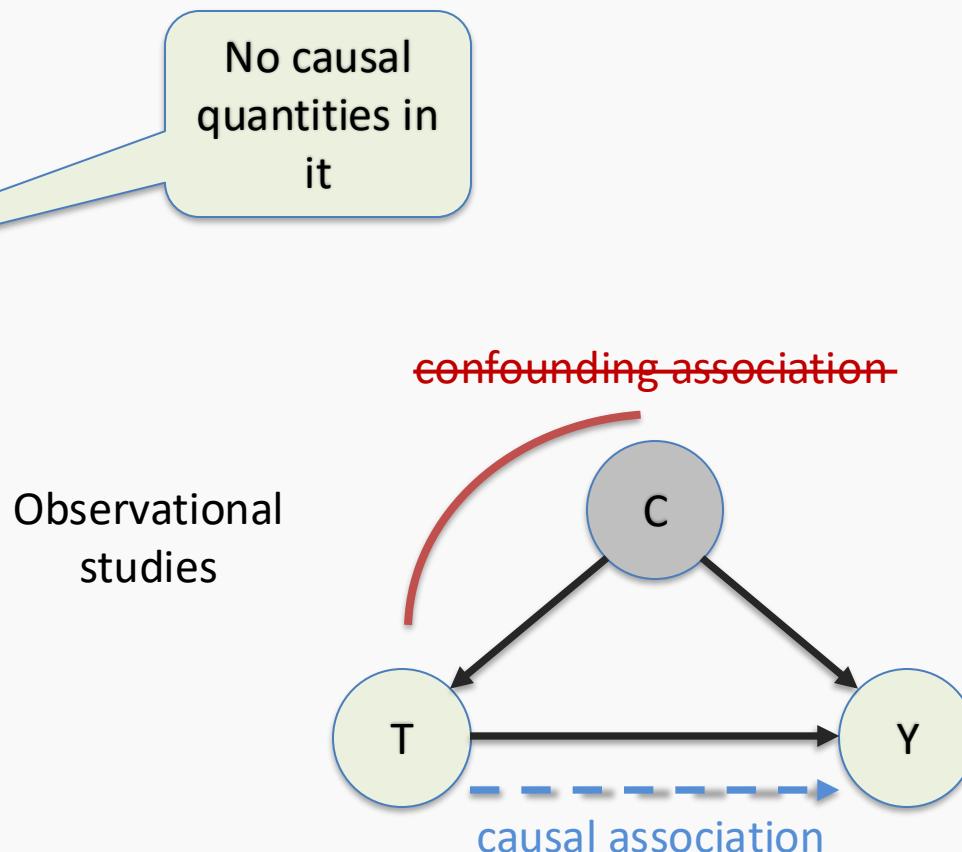
$$\mathbb{E}[Y(t)|W = w] = \mathbb{E}[Y|t, w]$$



Measuring a Causal Effect in Observational Studies

Adjust for confounders. Or adjust for a variable W . If W is the right variable: in this case $W = C$ (condition severity).

$$\mathbb{E}[Y(t)|W = w] = \mathbb{E}[Y|t, w]$$



Measuring a Causal Effect in Observational Studies

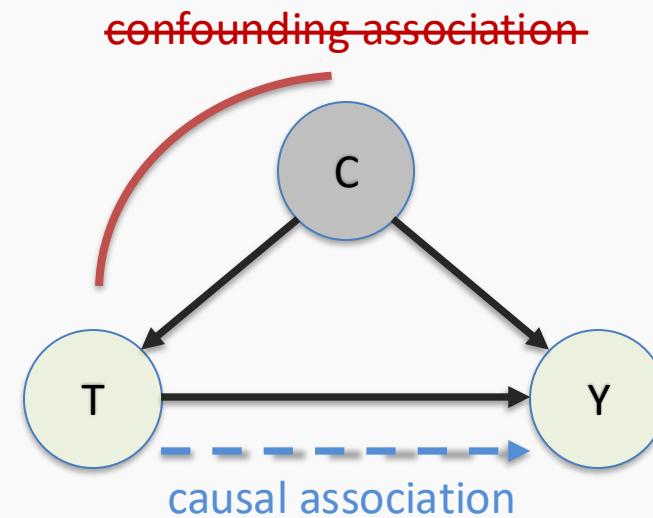
Adjust for confounders. Or adjust for a variable W . If W is the right variable: in this case $W = C$ (condition severity).

$$\mathbb{E}[Y(t)|W = w] = \mathbb{E}[Y|t, w]$$

$$\mathbb{E}[Y(t)] = \mathbb{E}_w[Y|t, W]$$

Marginalize over W

Observational studies

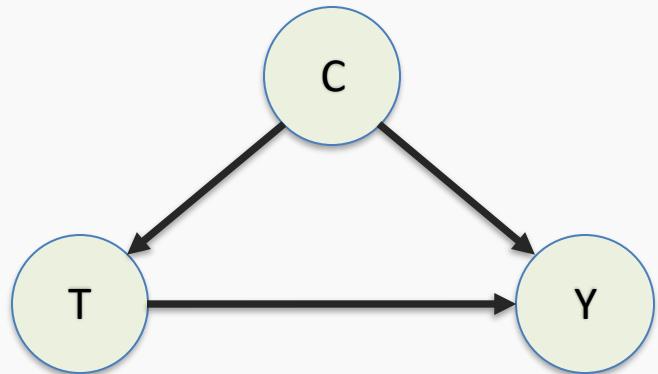


Example of Adjustment on PyCA-109a

$$\mathbb{E}[Y(t)] = \mathbb{E}_C[Y|t, C]$$

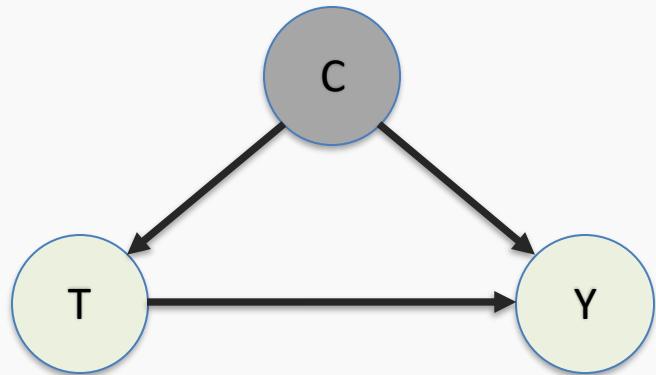
| | | Condition | | |
|-----------|----------------------|----------------------|------------------------|--|
| Treatment | Mild | Severe | Total | |
| A | 17% 150/900 | 40% 40/100 | 19% 190/1000 | |
| B | 14% 50/350 | 33% 50/150 | 20% 100/500 | |

$$\mathbb{E}[Y|T, C = 0] \quad \mathbb{E}[Y|T, C = 1] \quad \mathbb{E}[Y|T]$$



Example of Adjustment on PyCA-109a

$$\mathbb{E}[Y(t)] = \mathbb{E}_C[Y|t, C]$$

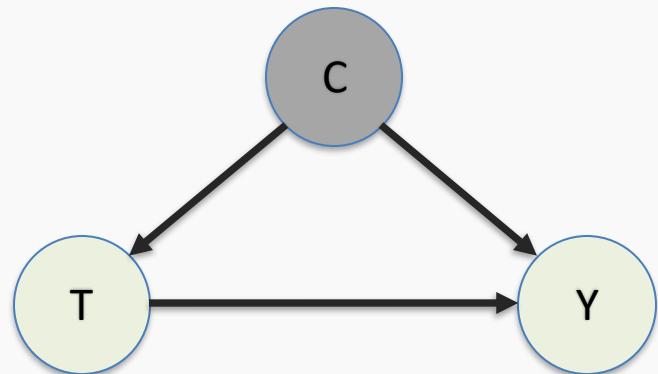


| | Condition | | |
|-----------|----------------------|----------------------|------------------------|
| Treatment | Mild | Severe | Total |
| A | 17% 150/900 | 40% 40/100 | 19% 190/1000 |
| B | 14% 50/350 | 33% 50/150 | 20% 100/500 |

$$\mathbb{E}[Y|T, C = 0] \quad \mathbb{E}[Y|T, C = 1] \quad \mathbb{E}[Y|T]$$

Example of Adjustment on PyCA-109a

$$\mathbb{E}[Y(t)] = \mathbb{E}_C[Y|t, C] = \sum_c \mathbb{E}_C[Y|t, c] P(c)$$



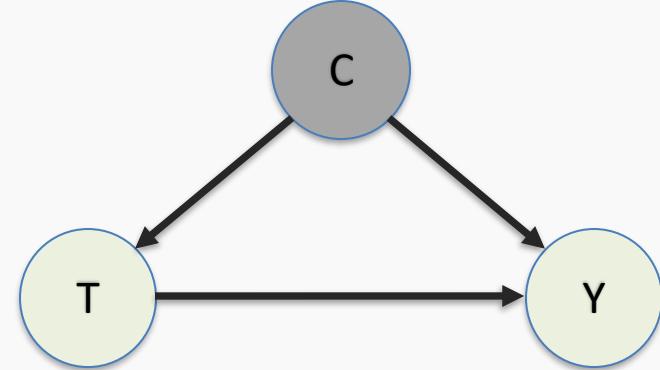
| | Condition | | |
|-----------|----------------------|----------------------|------------------------|
| Treatment | Mild | Severe | Total |
| A | 17% 150/900 | 40% 40/100 | 19% 190/1000 |
| B | 14% 50/350 | 33% 50/150 | 20% 100/500 |

$$\mathbb{E}[Y|T, C = 0] \quad \mathbb{E}[Y|T, C = 1] \quad \mathbb{E}[Y|T]$$

Example of Adjustment on PyCA-109a

$$\mathbb{E}[Y(t)] = \mathbb{E}_C[Y|t, C] = \sum_c \mathbb{E}_C[Y|t, c] P(c)$$

$$\mathbb{E}[Y|T = A, C = 0] = 0.17$$



| | | Condition | | | |
|-----------|----------------|---------------|-----------------|--------|--|
| Treatment | Mild | Severe | Total | Causal | |
| A | 17% 150/900 | 40% 40/100 | 19% 190/1000 | | |
| B | 14% 50/350 | 33% 50/150 | 20% 100/500 | | |

$$\mathbb{E}[Y|T, C = 0]$$

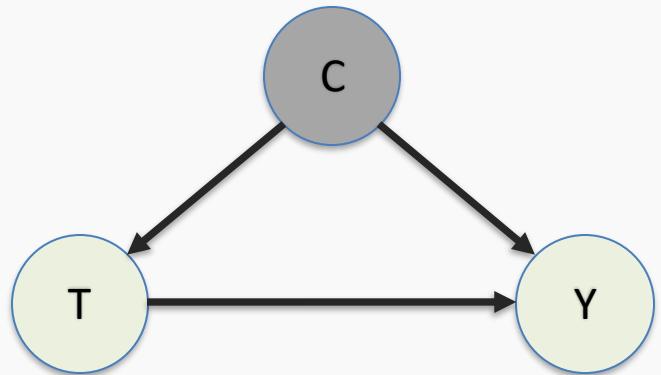
$$\mathbb{E}[Y|T, C = 1]$$

$$\mathbb{E}[Y|T]$$

$$\mathbb{E}[Y|do(T)]$$

Example of Adjustment on PyCA-109a

$$\mathbb{E}[Y(t)] = \mathbb{E}_C[Y|t, C] = \sum_c \mathbb{E}_C[Y|t, c] P(c)$$



| | Condition | | | |
|-----------|----------------|---------------|-----------------|--------|
| Treatment | Mild | Severe | Total | Causal |
| A | 17% 150/900 | 40% 40/100 | 19% 190/1000 | 22.5% |
| B | 14% 50/350 | 33% 50/150 | 20% 100/500 | 17.2% |

$$\mathbb{E}[Y|T, C = 0]$$

$$\mathbb{E}[Y|T, C = 1]$$

$$\mathbb{E}[Y|T]$$

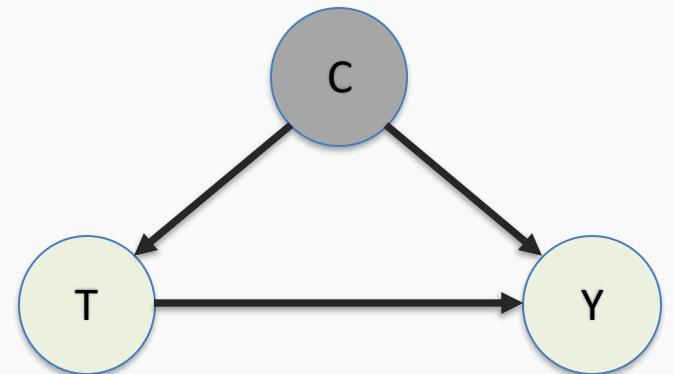
$$\mathbb{E}[Y|do(T)]$$

$$\frac{1250}{1500} (0.17) + \frac{250}{1500} (0.4) = 0.208$$

$$\frac{1250}{1500} (0.14) + \frac{250}{1500} (0.33) = 0.172$$

Example of Adjustment on PyCA-109a

$$\mathbb{E}[Y(t)] = \mathbb{E}_C[Y|t, C] = \sum_c P(c) \mathbb{E}_C[Y|t, c]$$



| | Condition Severity | | | |
|-----------|--------------------|---------------|-----------------|------------------------------|
| Treatment | Mild | Severe | Total | Causal, adjusted for C |
| A | 17% 150/900 | 40% 40/100 | 19% 190/1000 | 20.83% |
| B | 14% 50/350 | 33% 50/150 | 20% 100/500 | 17.17% |

$$\mathbb{E}[Y|T, C = 0]$$

$$\mathbb{E}[Y|T, C = 1]$$

$$\mathbb{E}[Y|T]$$

$$\mathbb{E}_C[Y|T]$$

P(c) rebalances the confounder across treatments.

$$\frac{1250}{1500} (0.17) + \frac{250}{1500} (0.4) = 0.2083$$

$$\frac{1250}{1500} (0.14) + \frac{250}{1500} (0.33) = 0.1717$$

Example of Adjustment on PyCA-109a

Average treatment effect (ATE):

$$\mathbb{E}[Y_i(1) - Y_i(0)] = \mathbb{E}[Y(1)] - \mathbb{E}[Y(0)]$$

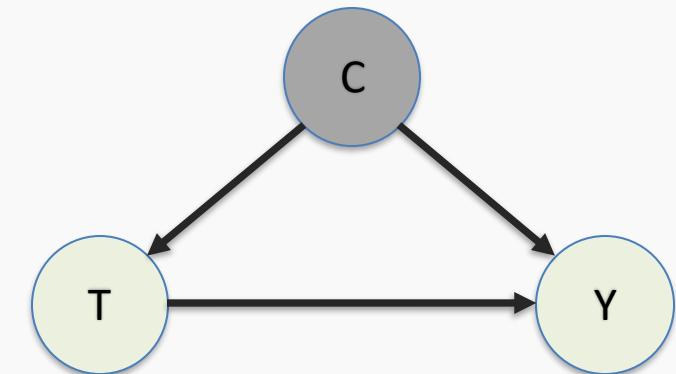
| | Condition Severity | | | |
|-----------|--------------------|---------------|-----------------|------------------------------|
| Treatment | Mild | Severe | Total | Causal, adjusted for C |
| A | 17% 150/900 | 40% 40/100 | 19% 190/1000 | 20.83% |
| B | 14% 50/350 | 33% 50/150 | 20% 100/500 | 17.17% |

$$\mathbb{E}[Y|T, C = 0]$$

$$\mathbb{E}[Y|T, C = 1]$$

$$\mathbb{E}[Y|T]$$

$$\mathbb{E}_C[Y|T]$$



P(c) rebalances the confounder across treatments.

$$\frac{1250}{1500} (0.17) + \frac{250}{1500} (0.4) = 0.2083$$

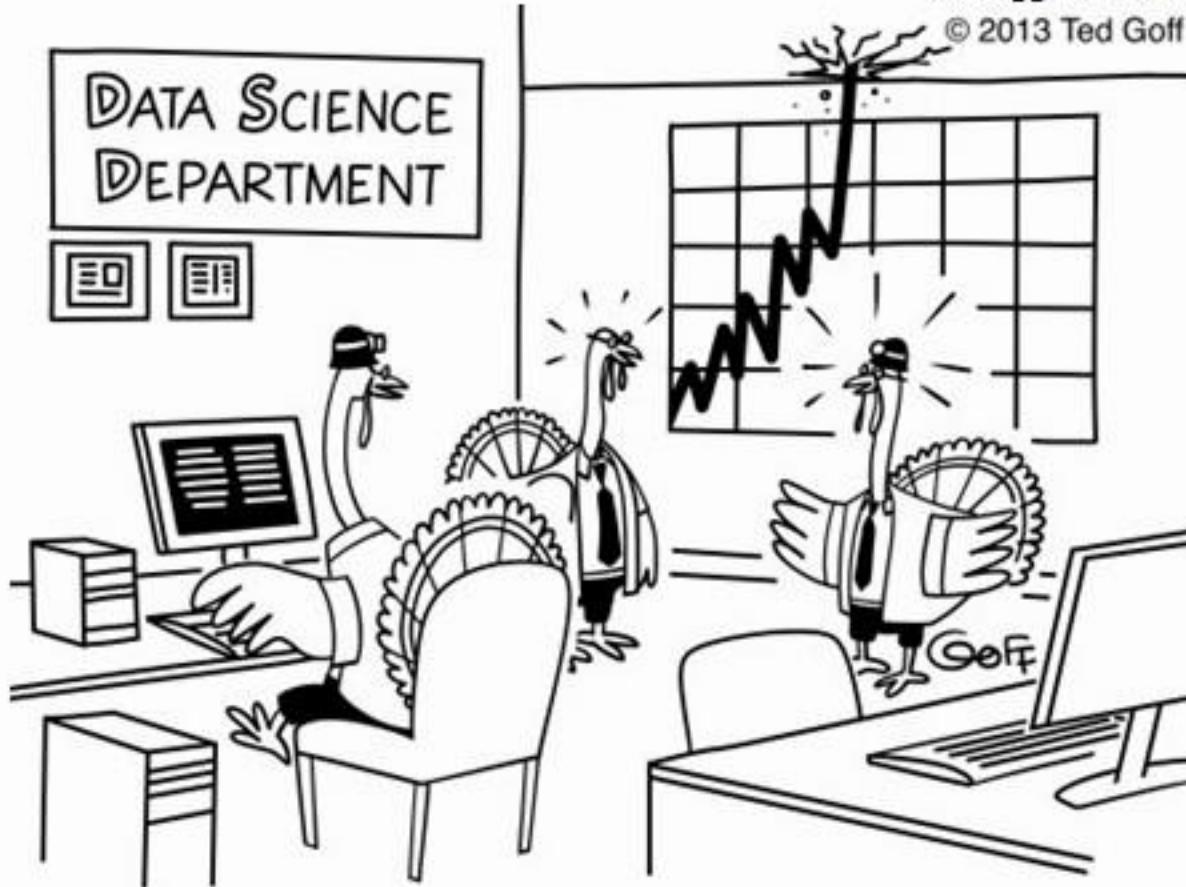
$$\frac{1250}{1500} (0.14) + \frac{250}{1500} (0.33) = 0.1717$$

AND THIS IS JUST THE BEGINNING!!!
(and overly simplified)

- Synthetic control
- Regression discontinuity
- Instrumental variables
- Propensity scores
- Etc.

Assessing Causal Relationships – Next Time

- There are 2 main approaches to assessing causal associations between variables (we have hinted at them already)
 1. Using careful study design: we could perform **experiments** (called **A-B Testing** in the world of Data Science).
 2. Using careful analysis of observational data: we perform **observational causal inference** and take extra care in creating the counterfactual.
- **Key issue:** we may never be able to recreate the correct counterfactual when using observational data.
- **Note:** causal inference of experimental data is often very simple – a simple *t*-test or randomization test. Causal inference of observational data is very difficult – we will learn a few approaches next time.



**"I don't like the look of this.
Searches for gravy and turkey stuffing
are going through the roof!"**

Lecture 16: Causal Inference II

aka STAT109A, AC209A, CSCIE-109A

CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai, and Chris Gumb



Outline

AB Testing (Randomized, Control Trials)

Adjusting for Confounders

Propensity Scores

- Estimation
- Adjusting, Weighting, and Matching

Covariate Balancing

Experiments and A/B Testing

- In the world of Data Science, performing experiments to determine causation, like a randomized, controlled trial (RCT), is called **A/B testing**.
- Control group: group of users exposed to the “normal” or “baseline” version of the system
- Treatment group: group of users exposed to the experimental version of the system

Experiments and A/B Testing

A/B testing is often used in the tech industry

- to determine which form of website design (the treatment) leads to more ad clicks, purchases, etc... (the response).
- or to determine the effect of a new app rollout (treatment) on revenue or usage (the response).

What is an A/B Test?

Options

- A. The [Presidential Fitness Test](#) of how many sit-ups you can do in 60 seconds.
- B. Any comparison of CS 109A and CS 109B.
- C. Any comparison of 2 groups or treatments.
- D. An example of a Randomized, Controlled Trial to determine a causal relationship between treatment and a response/outcome.

Today's thought example: the golden AI goose

CS 109 is thinking about implementing an AI teaching assistant to aid in Python coding for the course.

- This will be an interactive widget, and appear as a golden goose inside the Jupyter notebook:



How should we go about implementing an experiment (A/B test) in order to determine its effectiveness? How should we measure effectiveness?

Completely Randomized Design

There are many ways to design an experiment, depending on the number of treatment types, number of treatment groups, how the treatment effect may vary across subgroups, etc...

The simplest type of experiment is called a Completely Randomized Design (CRD). If two treatments, call them treatment A and treatment B , are to be compared across n subjects, then $n/2$ subject are randomly assigned to each group.

- If $n = 100$, this is equivalent to putting all 100 names in a hat, and pulling 50 names out and assigning them to treatment A .

Assigning subject to treatments

In order to **balance confounders**, the subjects must be properly randomly assigned to the treatment groups, and sufficient enough sample sizes need to be used.

For a CRD with 2 treatment arms, how can this randomization be performed via a computer?

You can just sample $n/2$ numbers from the values $1, 2, \dots, n$ without replacement and assign those individuals (in a list) to treatment group A , and the rest to treatments group B . This is equivalent to sorting the list of numbers, with the first half going to treatment A and the rest going to treatment B .

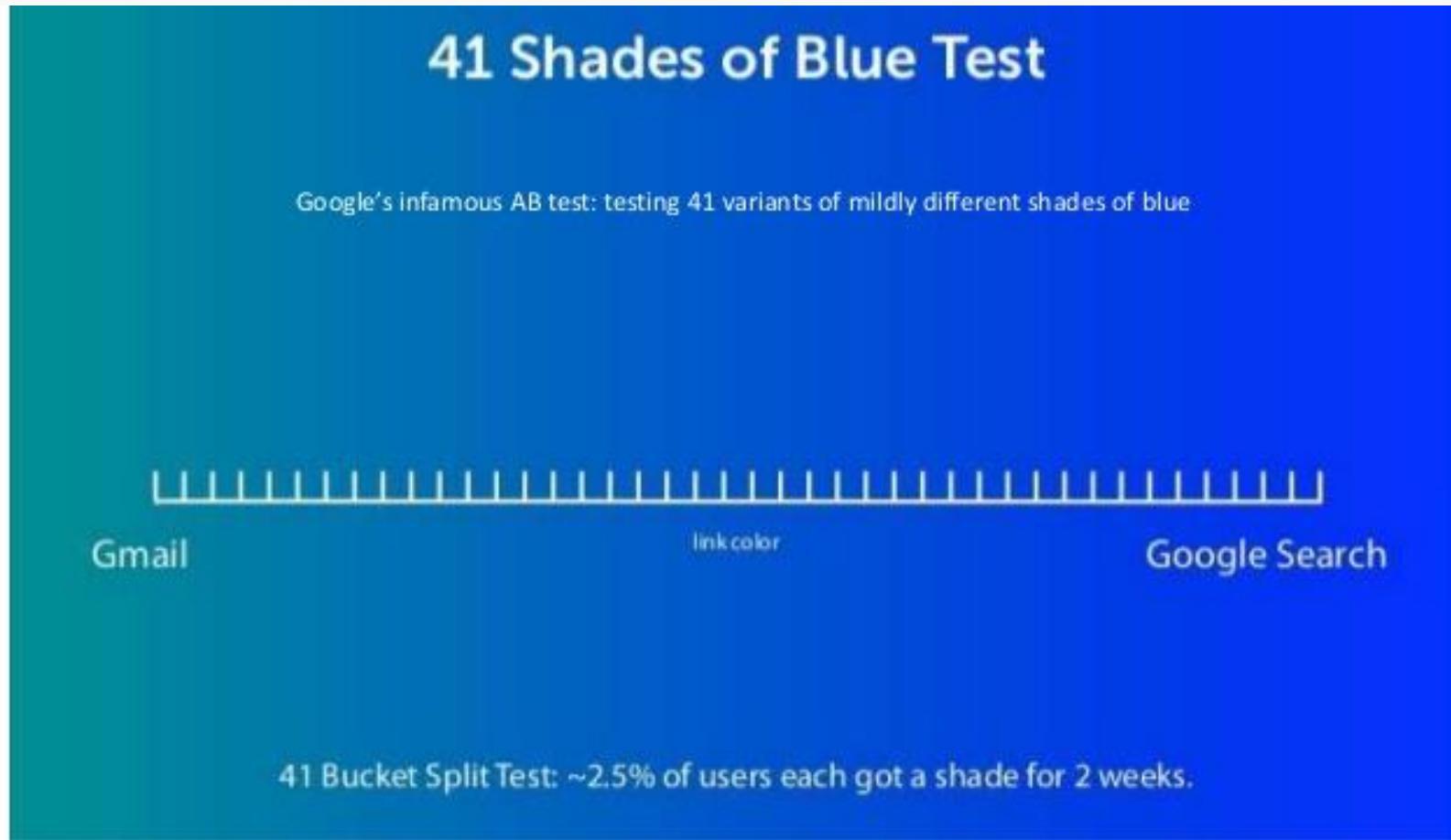
This is just like a 50-50 test-train split!

Beyond just A vs. B

How can an AB test be expanded to include more than two options?
What if there are more than just one type of treatment?

1. The **multivariate experimental design** generalizes this approach. If there are two treatment types (font color, and website layout), then both treatments' effects can (and should) be tested simultaneously. Why?
2. In a **full factorial experimental design**, each and every combination of treatments are considered different treatment groups. Experiments online are cheap. Full factorial designs are often possible and feasible.

Example #1: An infamous A/B Test, [41 Shades of Blue](#)



How should the study proceed? How should the data be analyzed?

Example #2: The 2008 Obama Campaign

In 2008, the Obama campaign raised much of its money via online donations through its website.

They wanted to optimize the launch page that visitors saw when they came to the campaign website. They were attempting to maximize the number of visitors that would sign up for their emailing list.

There were 2 treatments they attempted to vary:

- the image or video the user saw.
- the words on the click-through button.

Media choices (6 of them):

- 3 images and 3 videos were possibly shown

Click- through button

- one of 4 choices:

SIGN UP NOW

JOIN US NOW

SIGN UP

LEARN MORE



How to design the experiment?

How should this experiment unfold?

1. What was the response variable?
2. What were the treatments? What were the treatment groups?
3. How many observations (sample size) needed to be selected in order to determine which treatment group is most effective?
4. What analysis should be performed?

Obama 2008: the specifics

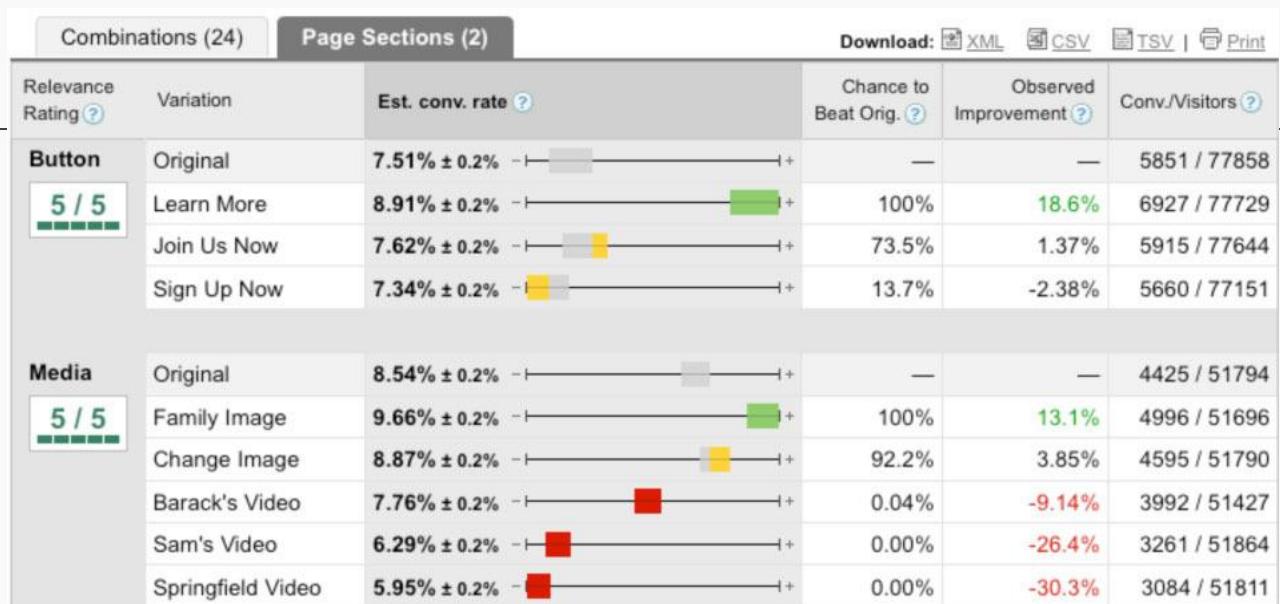
1. What was the response variable?
 - sign-up rate
2. What were the treatments? What were the treatment groups?
 - 2 treatments (media type and button). 24 treatment groups
3. How many observations (sample size) needed to be selected in order to determine which treatment group is most effective?
 - The campaign decided to run the experiment on 310,382 visitors!!!
4. What analysis should be performed?
 - Classically, a χ^2 test for independence could be performed.
Or better yet, a randomization test ☺

The data were overwhelming...

The Results

The results are shown to the right (note: they are from a 3rd party site that runs AB tests for website design: Optimizely).

<https://blog.optimizely.com/2010/11/29/how-obama-raised-60-million-by-running-a-simple-experiment/>



The results (cont.)

The winning variation had a sign-up rate of 11.6%. The original page had a sign-up rate of 8.26%. That's an improvement of 40.6% in sign-up rate...[leading to an] additional 2,880,000 email addresses translated into 288,000 more volunteers...and an additional \$60 million in donations.

See any issue in this conclusion?

But more importantly, they did learn one lesson: those intimately involved in designing websites (or medical treatments) are too biased to properly make conclusions as to what works best. They like the videos, and they performed the worst.

And the winner was:



Example #3: The app update roll-out problem

A company is interested in updating their app/program, so they start a ‘pilot program’ to test the waters to see how this update will affect some important measure (like revenue or usage). How should they do this?

They select a sample of users and ask them to voluntarily update the app on their phones in order to estimate the affect of this update.

Any issues with this design?

Volunteers will always be the most excited, dedicated users: a biased sample from all of their users.

We can potentially check for this bias via a χ^2 test for goodness-of-fit.

Challenges in A/B Testing

1. Understanding how short-term metrics (measured during A/B tests) lead to long-term improvements in user experience and/or revenue.

Short-term vs. Long-term Metrics:

- An increase in ad clicks suggests an increase in revenue.
- Showing lots of ads (often) hurts the user experience and decreases retention (i.e., long-term ad-click revenue)

2. Selecting the correct metric.
3. Making sure that your claims are exactly tested.
4. Fixing bugs in the experimental infrastructure.
5. Using sound statistical methods.

Potential Problems in A/B Testing

1. We need enough to reach statistical significance. Insufficient sample sizes can lead to inconclusive or misleading results.
2. Users exposed to both variants over time might change their behavior, distorting test outcomes.
3. Testing different versions can create inconsistent user experiences, potentially frustrating or confusing users.
4. Designing, running, and analyzing A/B tests can consume significant resources in terms of time, tools, and manpower.

Outline

AB Testing (Randomized, Control Trials)

Adjusting for Confounders

Propensity Scores

- Estimation
- Adjusting, Weighting, and Matching

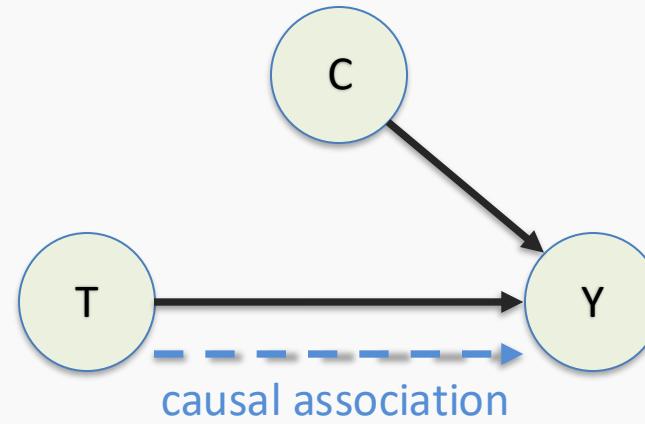
Covariate Balancing

Observational Studies

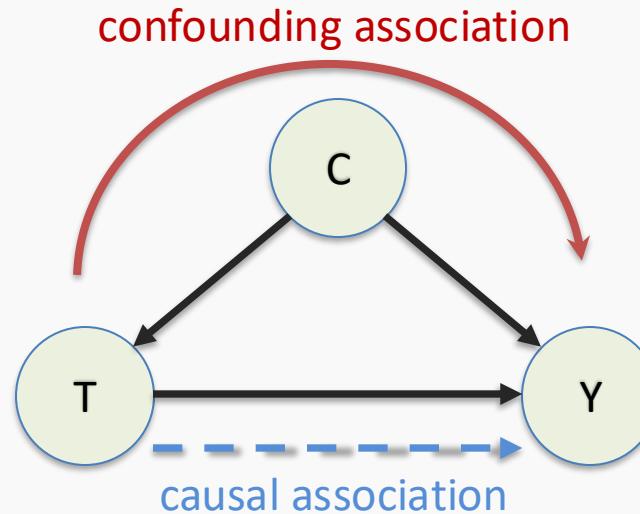
Can't always perform RCT

- Can't randomize
- Ethical problem
- Not feasible
- Not possible
- RCTs are expensive and difficult to set

Ideal



Observational studies



Recall the PyCA-109a example

Treatment T: A (Raderzole) and B (Gumboxin)

Condition Severity C: mild (0) or severe (1)

Outcome Y: alive (0) or dead (1)

| | | Condition Severity | | |
|------------------|----------------|--------------------|-----------------|--|
| Treatment | Mild | Severe | Total | |
| A (Raderzole) | 17% 150/900 | 40% 40/100 | 19% 190/1000 | |
| B (Gumboxin) | 14% 50/350 | 33% 50/150 | 20% 100/500 | |

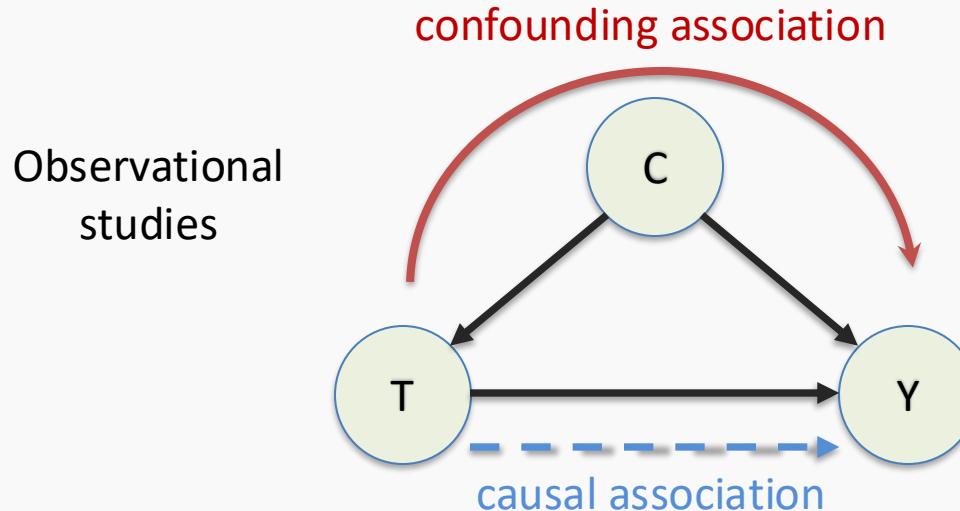
$$\mathbb{E}[Y|T, C = 0] \quad \mathbb{E}[Y|T, C = 1] \quad \mathbb{E}[Y|T]$$

Measuring a Causal Effect in Observational Studies

Adjust for confounders C (condition severity).

Consider the quantity

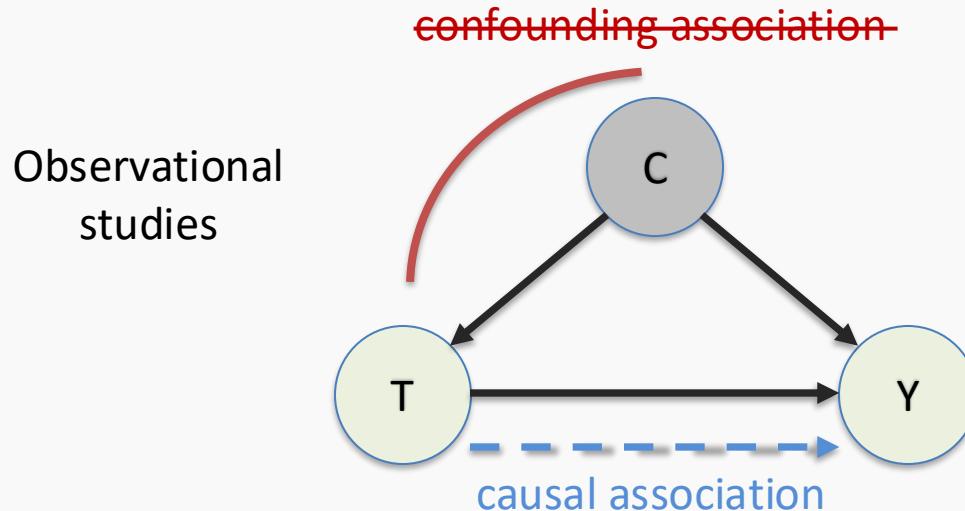
$$\mathbb{E}[Y(t)|C = c]$$



Measuring a Causal Effect in Observational Studies

Adjust for confounders C (condition severity).

$$\mathbb{E}[Y(t)|C = c] = \mathbb{E}[Y|t, c]$$



Measuring a Causal Effect in Observational Studies

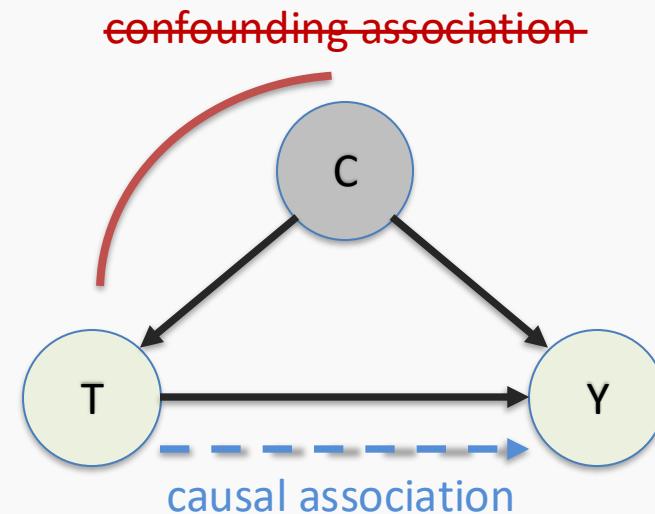
Adjust for confounders C (condition severity).

$$\mathbb{E}[Y(t)|C = c] = \mathbb{E}[Y|t, c]$$

$$\mathbb{E}[Y(t)] = \mathbb{E}_C[Y|t, C]$$

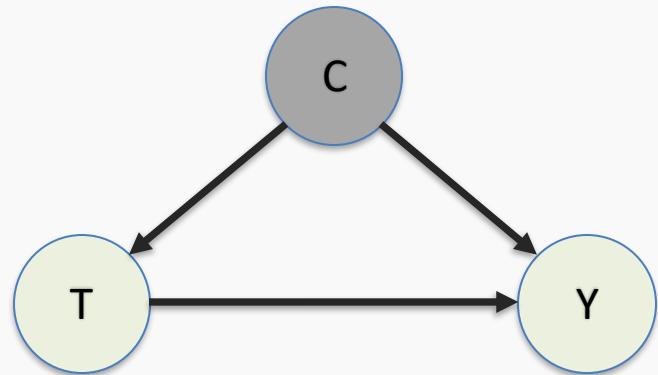
Observational
studies

Marginalize over
confounder C



Example of Adjustment on PyCA-109a

$$\mathbb{E}[Y(t)] = \mathbb{E}_C[Y|t, C]$$

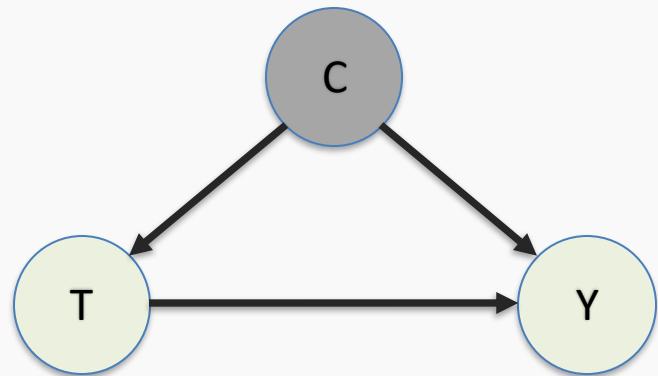


| | Condition | | |
|-----------|----------------------|----------------------|------------------------|
| Treatment | Mild | Severe | Total |
| A | 17% 150/900 | 40% 40/100 | 19% 190/1000 |
| B | 14% 50/350 | 33% 50/150 | 20% 100/500 |

$$\mathbb{E}[Y|T, C = 0] \quad \mathbb{E}[Y|T, C = 1] \quad \mathbb{E}[Y|T]$$

Example of Adjustment on PyCA-109a

$$\mathbb{E}[Y(t)] = \mathbb{E}_C[Y|t, C] = \sum_c \mathbb{E}_C[Y|t, c] P(c)$$



| | Condition | | |
|-----------|----------------------|----------------------|------------------------|
| Treatment | Mild | Severe | Total |
| A | 17% 150/900 | 40% 40/100 | 19% 190/1000 |
| B | 14% 50/350 | 33% 50/150 | 20% 100/500 |

$$\mathbb{E}[Y|T, C = 0] \quad \mathbb{E}[Y|T, C = 1] \quad \mathbb{E}[Y|T]$$

Example of Adjustment on PyCA-109a

$$\mathbb{E}[Y(t)] = \mathbb{E}_C[Y|t, C] = \sum_c \mathbb{E}_C[Y|t, c] P(c)$$

$$\mathbb{E}[Y|T = A, C = 0] = 0.17$$

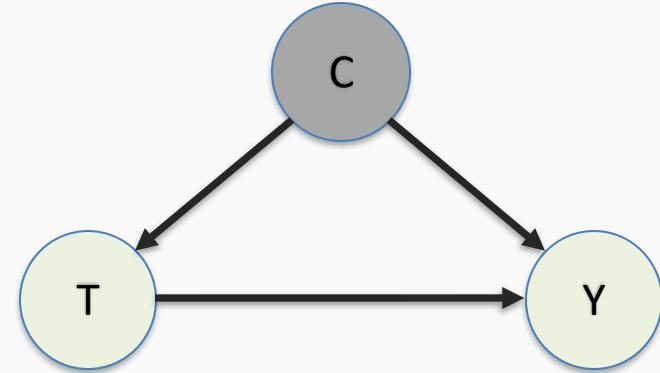
| | Condition | | | |
|-----------|----------------|---------------|-----------------|--------|
| Treatment | Mild | Severe | Total | Causal |
| A | 17% 150/900 | 40% 40/100 | 19% 190/1000 | |
| B | 14% 50/350 | 33% 50/150 | 20% 100/500 | |

$$\mathbb{E}[Y|T, C = 0]$$

$$\mathbb{E}[Y|T, C = 1]$$

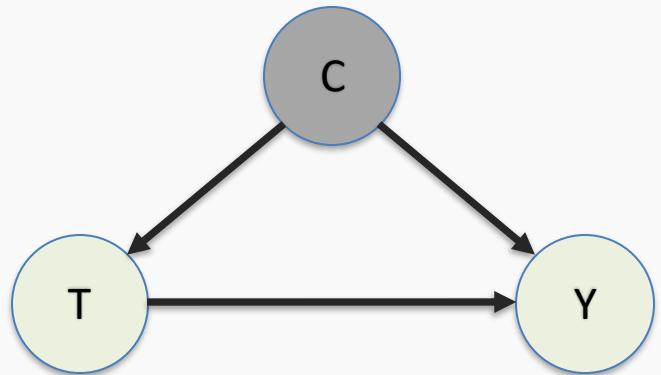
$$\mathbb{E}[Y|T]$$

$$\mathbb{E}[Y|do(T)]$$



Example of Adjustment on PyCA-109a

$$\mathbb{E}[Y(t)] = \mathbb{E}_C[Y|t, C] = \sum_c \mathbb{E}_C[Y|t, c] P(c)$$



| | Condition | | | |
|-----------|----------------|---------------|-----------------|--------|
| Treatment | Mild | Severe | Total | Causal |
| A | 17% 150/900 | 40% 40/100 | 19% 190/1000 | 22.5% |
| B | 14% 50/350 | 33% 50/150 | 20% 100/500 | 17.2% |

$$\mathbb{E}[Y|T, C = 0]$$

$$\mathbb{E}[Y|T, C = 1]$$

$$\mathbb{E}[Y|T]$$

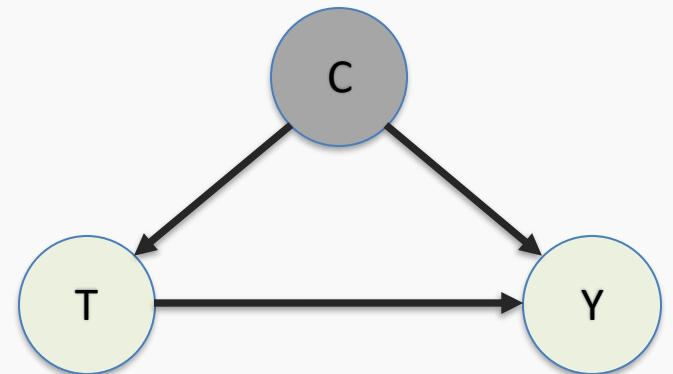
$$\mathbb{E}[Y|do(T)]$$

$$\frac{1250}{1500} (0.17) + \frac{250}{1500} (0.4) = 0.208$$

$$\frac{1250}{1500} (0.14) + \frac{250}{1500} (0.33) = 0.172$$

Example of Adjustment on PyCA-109a

$$\mathbb{E}[Y(t)] = \mathbb{E}_C[Y|t, C] = \sum_c P(c) \mathbb{E}_C[Y|t, c]$$



| | Condition Severity | | | |
|-----------|--------------------|---------------|-----------------|------------------------------|
| Treatment | Mild | Severe | Total | Causal, adjusted for C |
| A | 17% 150/900 | 40% 40/100 | 19% 190/1000 | 20.83% |
| B | 14% 50/350 | 33% 50/150 | 20% 100/500 | 17.17% |

$$\mathbb{E}[Y|T, C = 0]$$

$$\mathbb{E}[Y|T, C = 1]$$

$$\mathbb{E}[Y|T]$$

$$\mathbb{E}_C[Y|T]$$

P(c) rebalances the confounder across treatments.

$$\frac{1250}{1500} (0.17) + \frac{250}{1500} (0.4) = 0.2083$$

$$\frac{1250}{1500} (0.14) + \frac{250}{1500} (0.33) = 0.1717$$

Example of Adjustment on PyCA-109a

Average treatment effect (ATE):

$$\mathbb{E}[Y_i(1) - Y_i(0)] = \mathbb{E}[Y(1)] - \mathbb{E}[Y(0)]$$

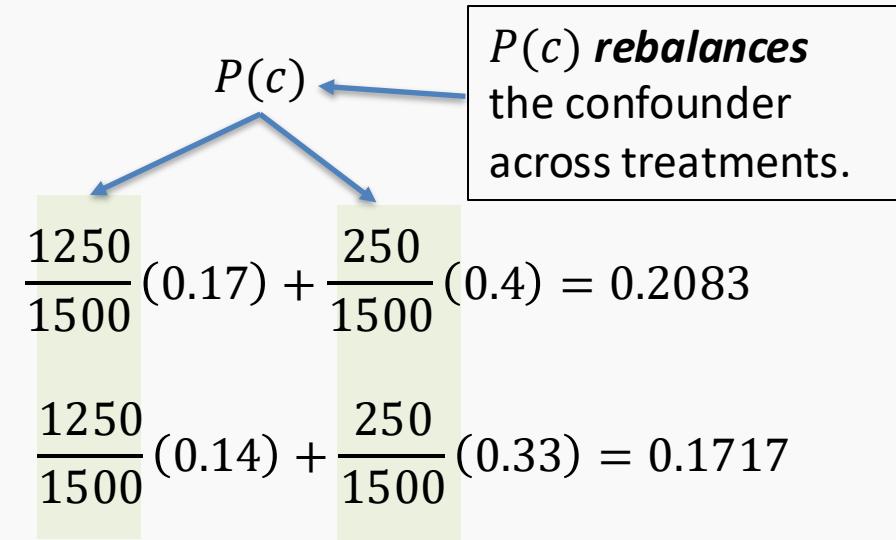
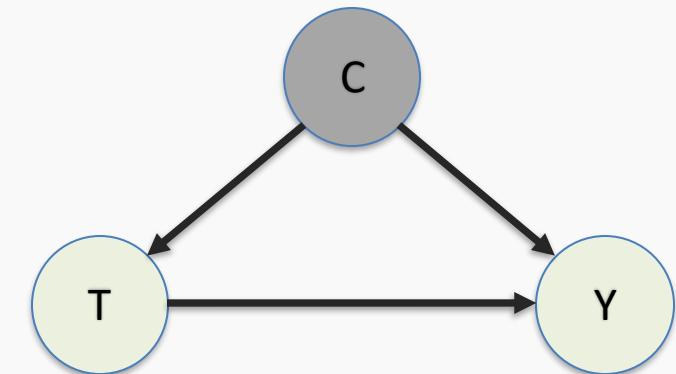
| | Condition Severity | | | |
|-----------|--------------------|---------------|-----------------|------------------------------|
| Treatment | Mild | Severe | Total | Causal, adjusted for C |
| A | 17% 150/900 | 40% 40/100 | 19% 190/1000 | 20.83% |
| B | 14% 50/350 | 33% 50/150 | 20% 100/500 | 17.17% |

$$\mathbb{E}[Y|T, C = 0]$$

$$\mathbb{E}[Y|T, C = 1]$$

$$\mathbb{E}[Y|T]$$

$$\mathbb{E}_C[Y|T]$$



AND THIS IS JUST THE BEGINNING!!!
(and overly simplified)

- Synthetic control
- Regression discontinuity
- Instrumental variables
- Propensity scores
- Etc.

Today's thought example: the golden AI goose

How should we go about implementing an observational study in order to determine the golden AI goose's effectiveness?



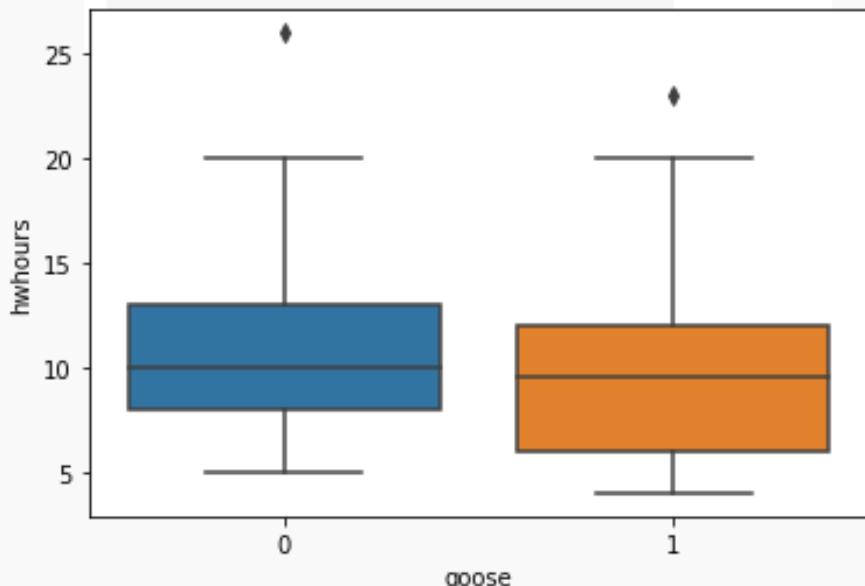
What other variables (predictors/confounders) should we measure?

- Class year
- Concentration
- GPA
- Etc.

Example: the golden AI goose (synthetic data)

```
ai = pd.read_csv('data/AIgoose.csv')  
ai.head()
```

| | hwhours | goose | conc | year | gpa |
|---|---------|-------|-------|------|------|
| 0 | 20 | 0 | other | soph | 3.68 |
| 1 | 12 | 0 | cs | jr | 3.92 |
| 2 | 10 | 0 | cs | sr | 3.20 |
| 3 | 13 | 0 | stat | soph | 4.00 |

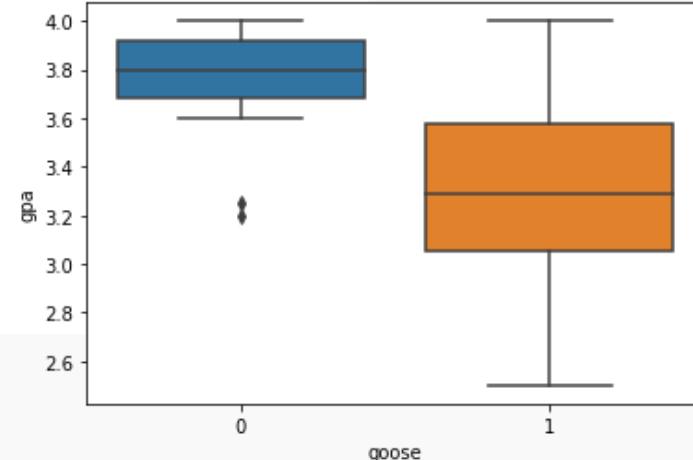


```
pd.crosstab(ai["goose"],ai["conc"])
```

| | conc | cs | other | stat |
|--------------|------|----|-------|------|
| goose | | | | |
| 0 | 5 | 4 | 4 | |
| 1 | 7 | 12 | 3 | |

```
pd.crosstab(ai["goose"],ai["year"])
```

| | year | jr | soph | sr |
|--------------|------|----|------|----|
| goose | | | | |
| 0 | 7 | 3 | 3 | |
| 1 | 10 | 5 | 7 | |



Outline

AB Testing (Randomized, Control Trials)

Adjusting for Confounders

Propensity Scores

- Estimation
- Adjusting, Weighting, and Matching

Covariate Balancing

Propensity Scores

To adjust for the confounding with treatment assignment that may exist in an observational study, the **propensity score** can be estimated.

Propensity score: the **probability** that a subject/observation (ex: user, person, classroom, etc.) is assigned to a particular treatment, T_i , given their set of observed predictors, X_i .

$$e(X_i) = P(T_i = 1|X_i)$$

Why Does Propensity Score Matter

- Dimension Reduction:

When the dimension of covariate X is very large, estimating causal effects can be complex. Propensity score simplifies this by reducing dimension.

- Correct Selection Bias:

By using propensity score, we can upweight the units that are underrepresented in our data.

- Balance Covariates:

Conditional on propensity score $e(X)$, the distribution of covariate X is the same for treated and untreated.

Propensity Scores: estimation

The propensity score is the probability of receiving a treatment given X (the predictors). How can we estimate/model this probability?

We can use **any** reasonable classification model to estimate the probability of **treatment** using the rest of the set of covariates.

Standard approach: use logistic regression.

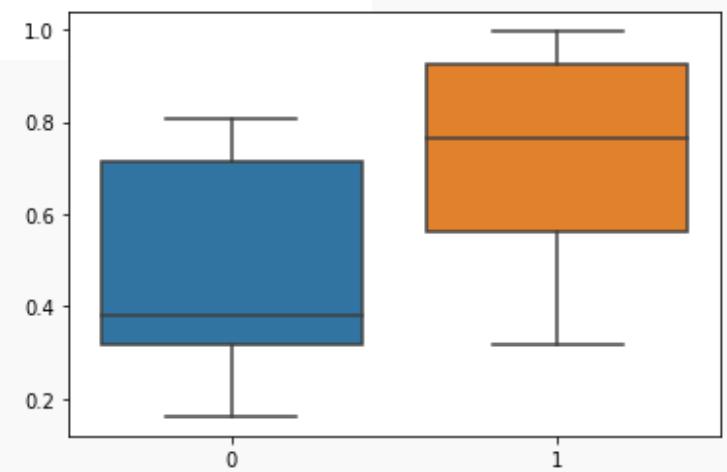
Key confusion: this is a model for treatment, T_i , not the response, Y_i .

Propensity Scores: estimation

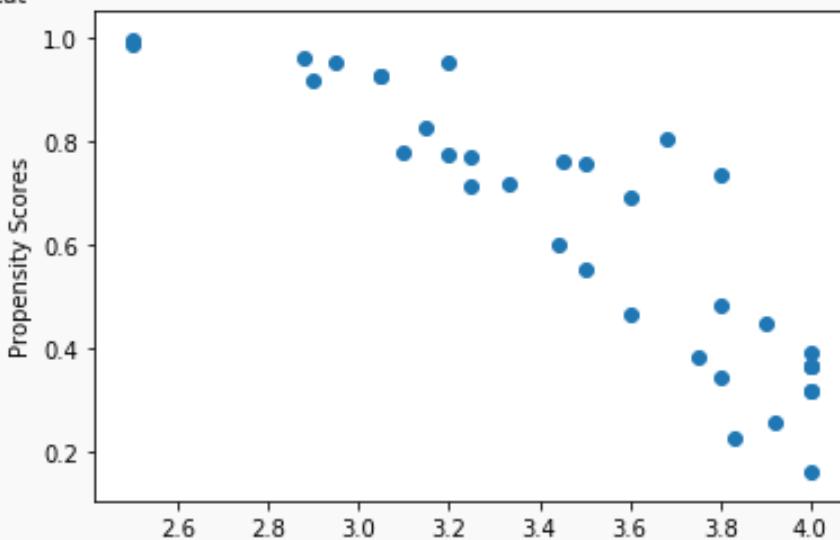
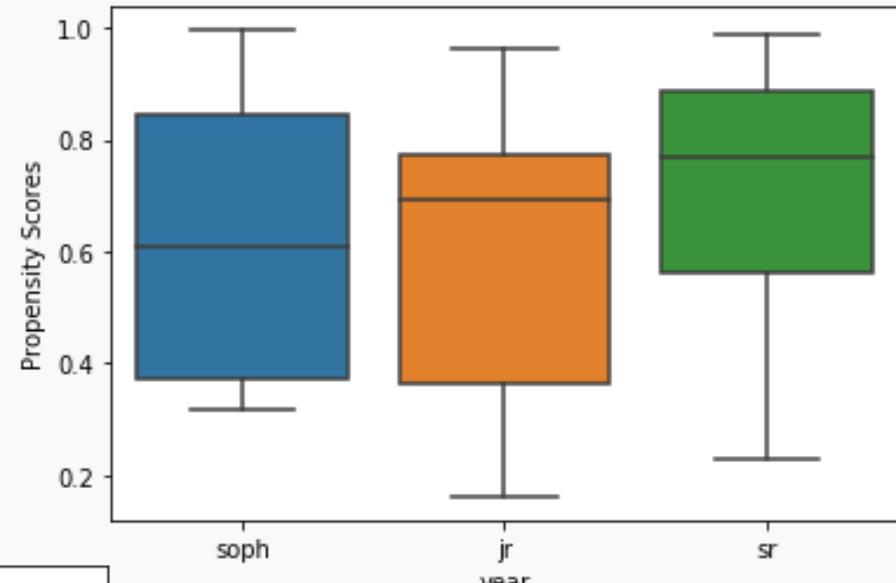
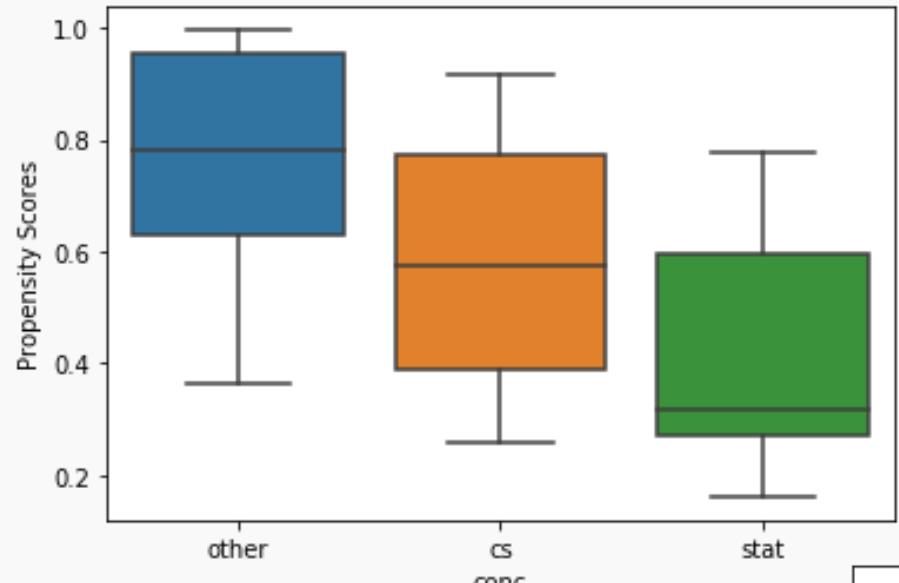
```
X = pd.concat([ai[['gpa']],
               pd.get_dummies(ai[['year', 'conc']], drop_first=True)],
              axis=1)
print(X.columns)
logit = linear_model.LogisticRegression(fit_intercept=True, penalty='none')
logit.fit(X, ai['goose'])
print(logit.coef_)
propensity_scores = logit.predict_proba(X)[:, 1]

sns.boxplot(y=logit.predict_proba(X)[:, 1], x=ai["goose"]);

Index(['gpa', 'year_soph', 'year_sr', 'conc_other', 'conc_stat'], dtype='object')
[-3.4082797  0.8882769 -0.16033108  0.7814993 -0.30950449]
```



Propensity Scores: evidence of confounding?



Using Propensity Scores for balancing confounders

Great, we now have propensity scores! How should we use them?

There are generally 3 approaches to incorporating propensity scores:

1. Covariate Adjustment:
2. Weighting:
3. Matching:

Using Propensity Scores: covariate adjustment

The simplest way to use propensity scores is to incorporate them into the main response model as **an additional (and only) predictor**:

This can result in a poorly adjusted model as the **estimand** is poorly defined.

Using Propensity Scores: inverse probability weighting

The second simplest way to use propensity scores is to incorporate them as sampling weights into the main response model:

This can result in a poorly estimated causal effect as the sampling weights can be huge (if $\hat{e}(X_i)$ is close to 0 or 1).

Using Propensity Scores: matching

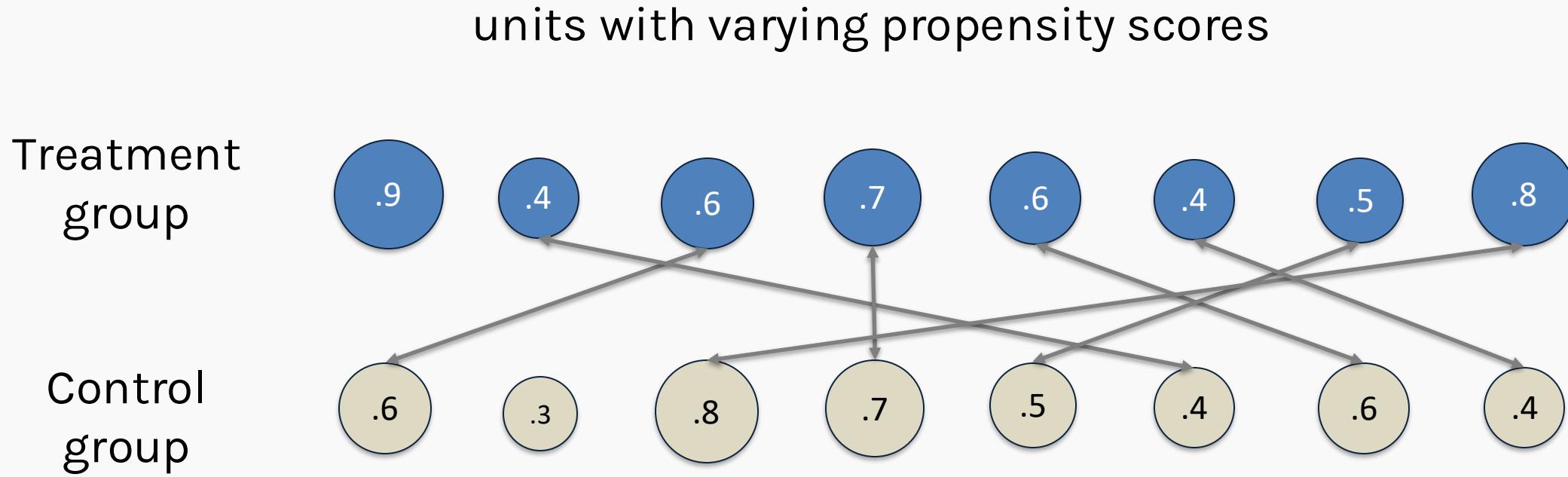
The best way to use propensity scores is to match on them: for every treated observation, find the most similar control observation. This creates a synthetic counterfactual for every observed treated observation.

A few key considerations:

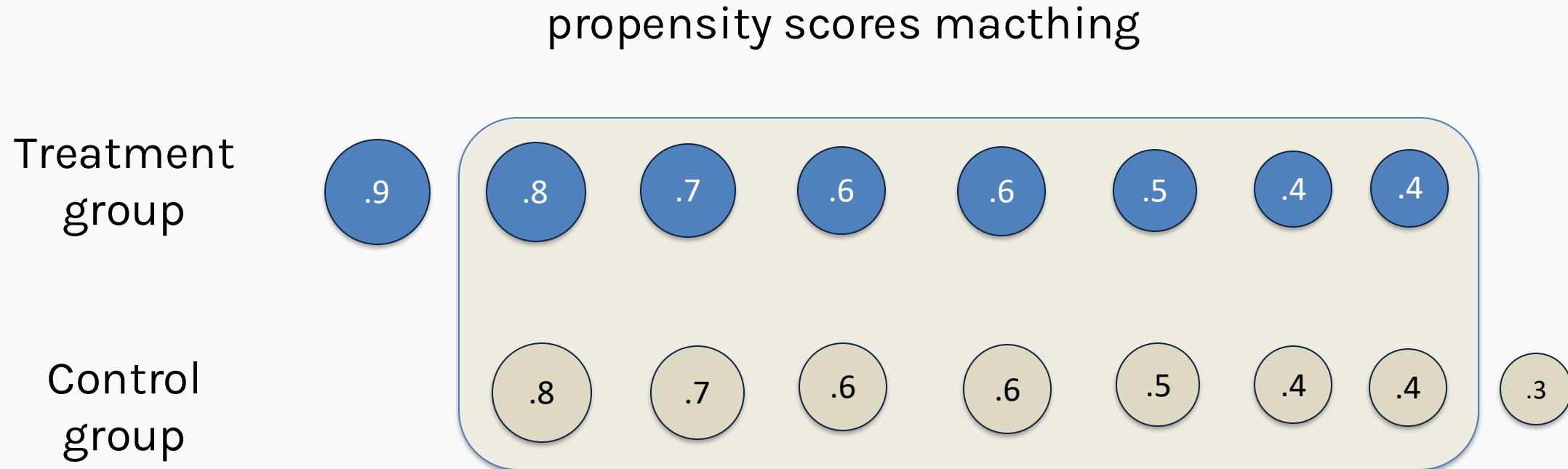
- Should we match with replacement?
- Should we strictly perform 1-to-1 matching?
- Should we throw away the extremes?

There are many ways to perform matching, and there is no one best approach or one measure to determine which matching algorithm is the best.

Propensity Score Matching: toy example



Propensity Score Matching: toy example



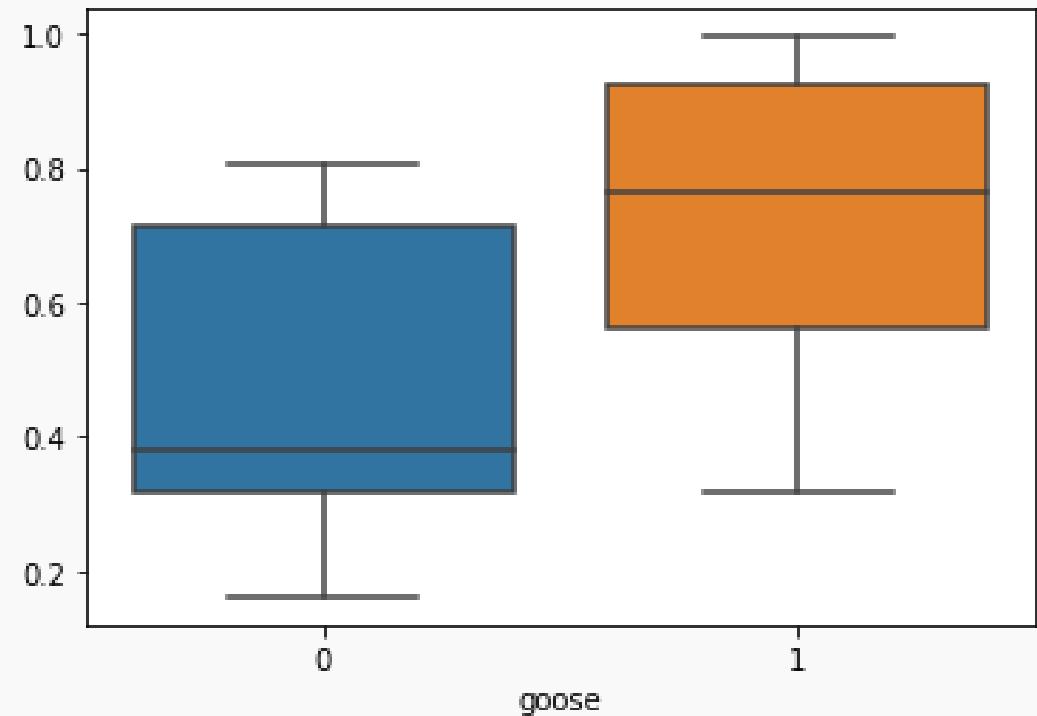
Using Propensity Scores: throwing away extremes

A thought question:

Should we even consider individuals that have probability of being treated close to 1 (or close to 0) in an observational study?

Consider this plot
of propensity scores:

What are the matches for those students that did not use the golden goose ($\text{goose} = 0$) that have $\hat{e}(X_i) < 0.3$? What about the students for which $\text{goose} = 1$ and $\hat{e}(X_i) > 0.8$?



Propensity Score Matching in Python

There is a Python package to aid in propensity score matching:

PsmPy: <https://pypi.org/project/psmepy/>

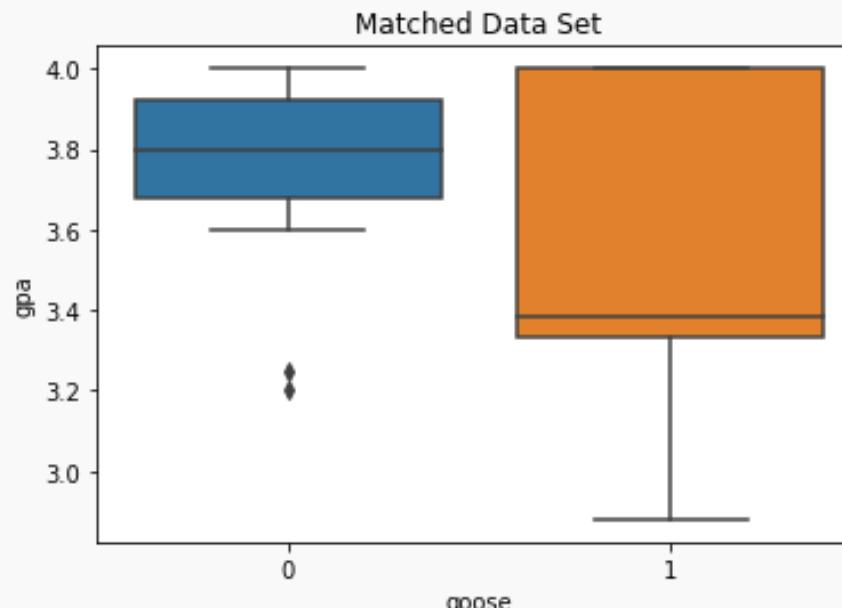
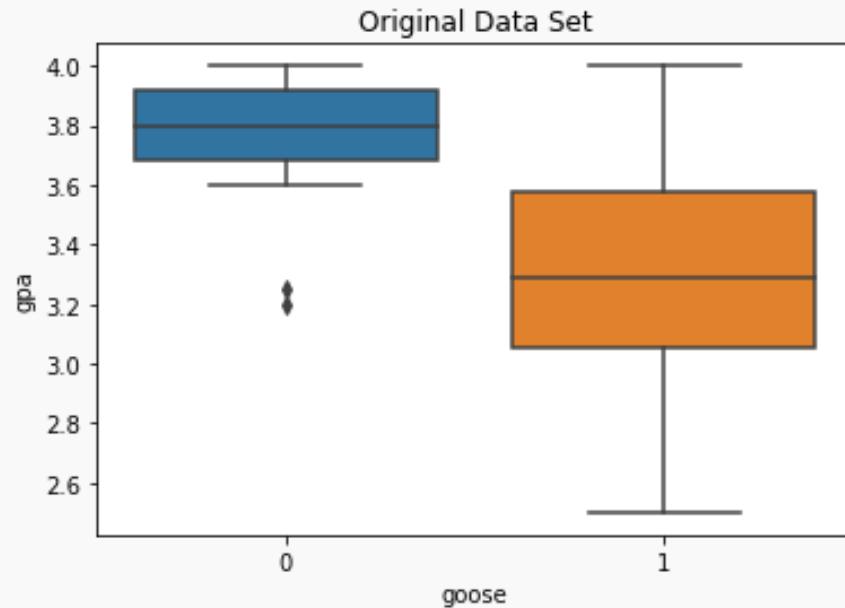
```
from psmepy import PsmPy
from psmepy.functions import cohenD
from psmepy.plotting import *

ai['id']=ai.index
X = pd.concat([ai[['gpa','goose','id']],
               pd.get_dummies(ai[['year','conc']]),drop_first=True]),
               axis=1)

psm = PsmPy(X, treatment='goose', idx='id')
```

```
psm.logistic_ps(balance = False)
psm.knn_matched(matcher='propensity_score', replacement=True,
                 caliper=None, drop_unmatched=True)
```

Propensity Score Matching: does it work?



```
pd.crosstab(ai['conc'],ai['goose'])
```

| goose | 0 | 1 |
|-------|---|---|
| conc | | |

| cs | 5 | 7 |
|----|---|---|
|----|---|---|

| other | 4 | 12 |
|-------|---|----|
|-------|---|----|

| stat | 4 | 3 |
|------|---|---|
|------|---|---|

```
pd.crosstab(df_matched['conc_other'], df_matched['goose'])
```

| goose | 0 | 1 |
|-------|---|---|
|-------|---|---|

| conc_other | | |
|------------|--|--|
|------------|--|--|

| 0 | 9 | 8 |
|---|---|---|
|---|---|---|

| 1 | 4 | 4 |
|---|---|---|
|---|---|---|

```
pd.crosstab(df_matched['conc_stat'], df_matched['goose'])
```

| goose | 0 | 1 |
|-------|---|---|
|-------|---|---|

| conc_stat | | |
|-----------|--|--|
|-----------|--|--|

| 0 | 9 | 9 |
|---|---|---|
|---|---|---|

| 1 | 4 | 3 |
|---|---|---|
|---|---|---|

Propensity Score Matching: the estimated causal effect?

```
X = pd.concat([ai[['gpa']],
               pd.get_dummies(ai[['year', 'conc']], drop_first=True),
               axis=1)
lm = linear_model.LinearRegression(fit_intercept=True)
lm.fit(ai[['goose']], ai['hwhours'])
lm.coef_[0]

-1.6153846153846156
```

```
lm.fit(pd.concat([ai[['goose']], X], axis=1), ai['hwhours'])
lm.coef_[0]

-2.983590934130868
```

```
lm.fit(pd.concat([ai[['goose']], pd.Series(propensity_scores)], axis=1), ai['hwhours'])
lm.coef_[0]

-3.4315624566363354
```

```
lm.fit(ai[['goose']], ai['hwhours'], sample_weight=propensity_scores)
lm.coef_[0]

-3.076729084815634
```

```
lm.fit(df_matched[['goose']], df_matched['hwhours'])
lm.coef_[0]
-2.7690438663340706
```

Outline

AB Testing (Randomized, Control Trials)

Adjusting for confounders

Propensity Scores

- Estimation
- Adjusting, Weighting, and Matching

Covariate Balancing

Direct Covariate Balancing

What was the purpose of using propensity scores?

- To adjust for covariate imbalance among predictors.

Propensity scores are great, but not the only approach to achieve this goal.

We can directly match the covariates in order to balance them!
This done through a linear programming algorithm.

Word of warning

Remember the main issue with observational studies

- Covariates, both measured and unmeasured, are likely to be imbalanced across treatment groups.

Using adjustments, models, propensity scores, and covariate balancing does a great job of fixing this imbalance for the measured confounders, but **does NOT guarantee** that all unmeasured confounders are balanced.

Note: it is likely to improve the unmeasured confounders balance.
Why?

- Unmeasured confounders are likely correlated to those that are measured.

Decision Trees - Classification

CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai and Chris Gumb



Alice Cheng
Zhangjiajie National Park in China

Outline

- Motivation
- Decision Trees - Classification
 - Intuition
 - Predictions
 - Splitting Criteria
 - Stopping Conditions

Outline

- Motivation
- Decision Trees - Classification
 - Intuition
 - Predictions
 - Splitting Criteria
 - Stopping Conditions

When you're trying to be one with
nature, but someone mentions
food

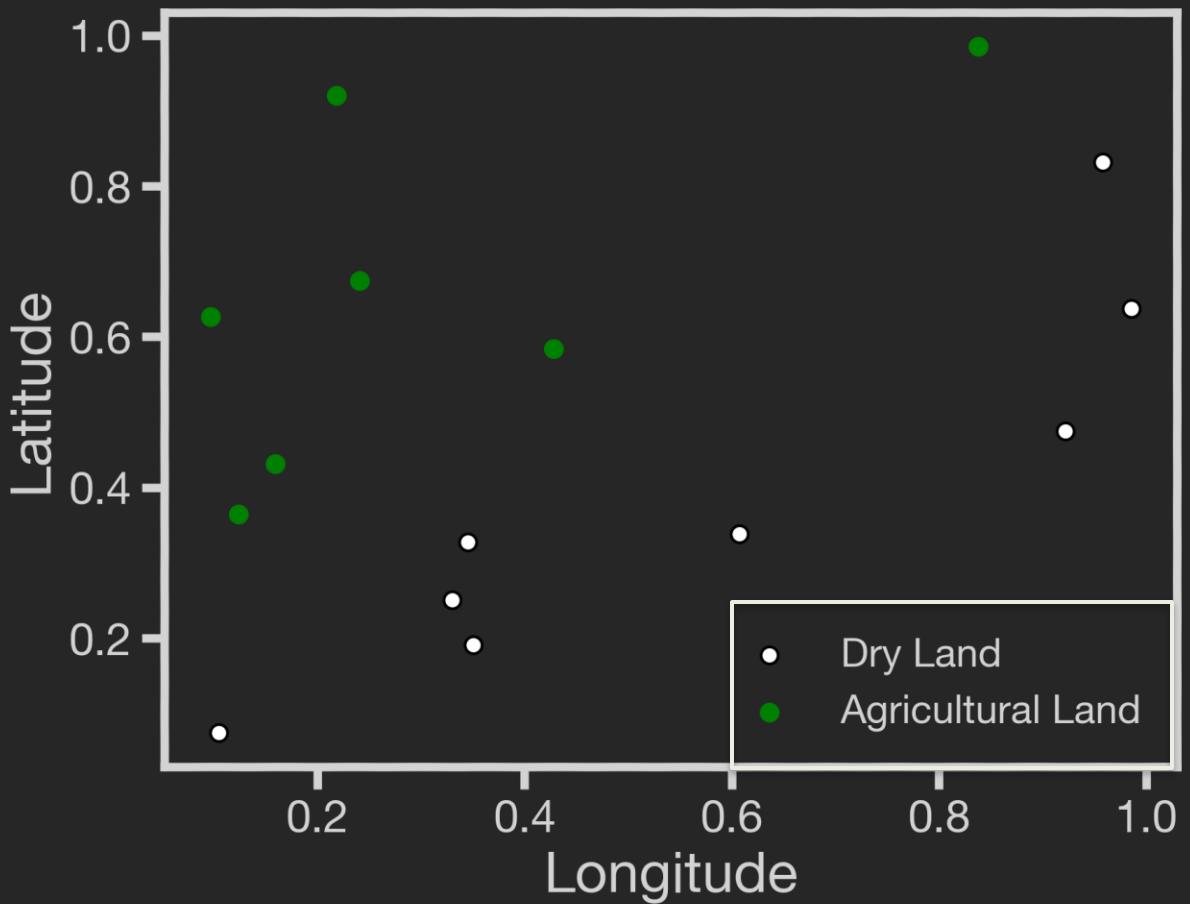


Motivation

The simplest and most straightforward classification model we know is **logistic regression**.

Logistic regression works best when:

- a) The classes are well-separated in the feature space, and
- b) The classification boundary has a nice geometry



Motivation

Classification Boundaries or **Decision Boundaries** are defined where the probabilities of being in class 1 and class 0 are equal, i.e.

$$P(Y = 1) = P(Y = 0)$$

Motivation

Classification Boundaries or **Decision Boundaries** are defined where the probabilities of being in class 1 and class 0 are equal, i.e.

$$P(Y = 1) = P(Y = 0) = 1 - P(Y = 1)$$

$$\Rightarrow P(Y = 1) = 0.5$$

which is equivalent to when the log-odds are zero:

$$X\beta = 0$$

HOW?

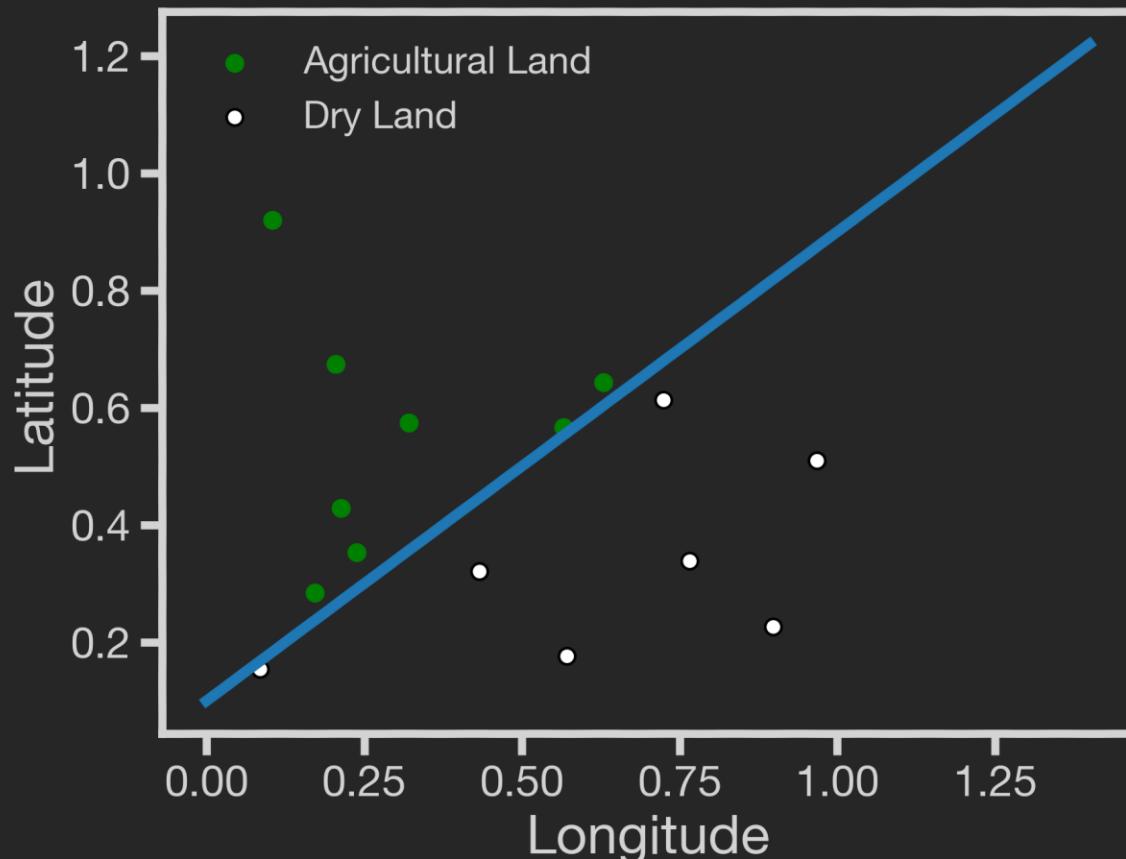
$$\log \frac{P(Y=1)}{P(Y=0)} = X\beta$$
$$\log \frac{P(Y=1)}{P(Y=0)} = \log 1 = 0 \rightarrow X\beta = 0$$

This equation defines a line or a hyperplane.



Motivation

For example, the equation that defines the decision boundary shown with a blue line in the figure is:



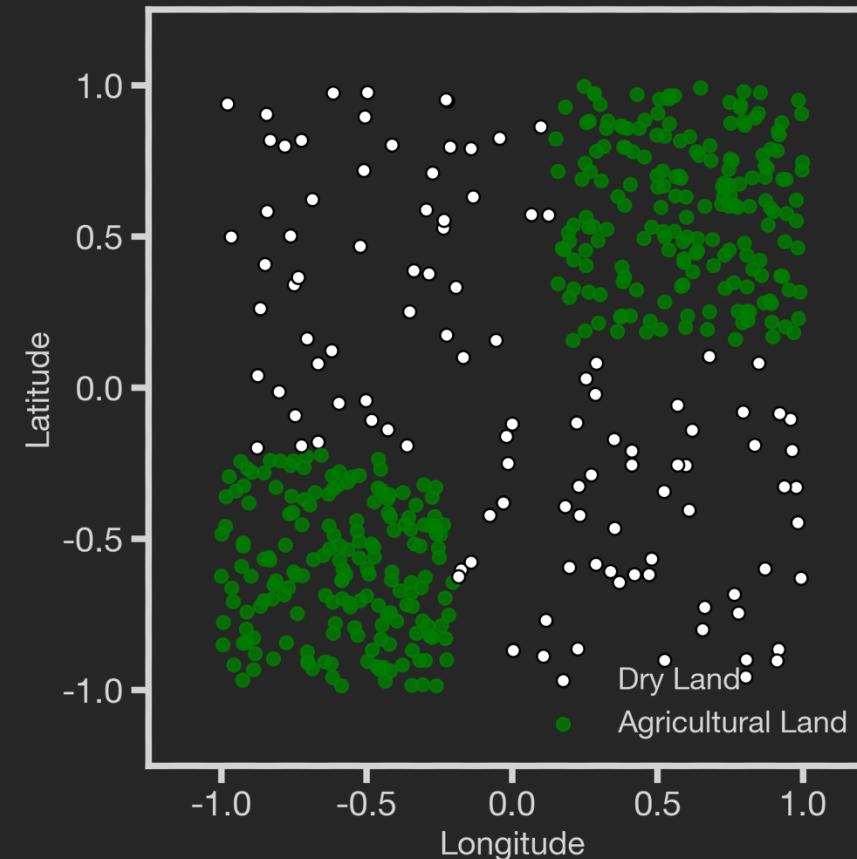
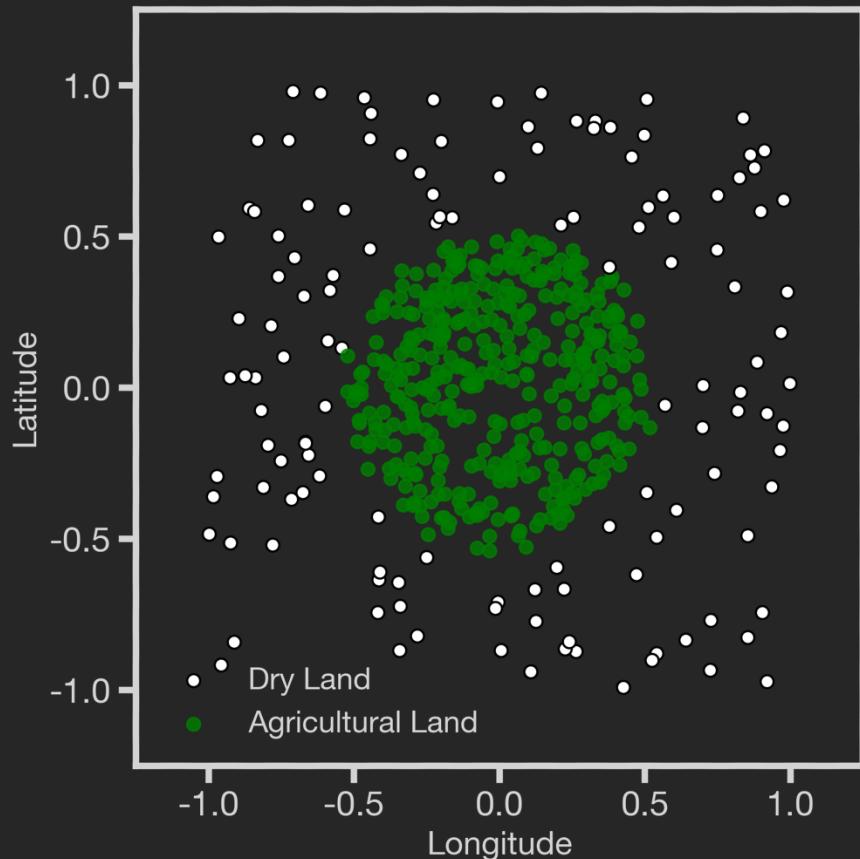
$$\text{Latitude} = 0.8 \text{ Longitude} + 0.1$$

or

$$-0.8 \text{ Longitude} + \text{Latitude} - 0.1 = 0$$

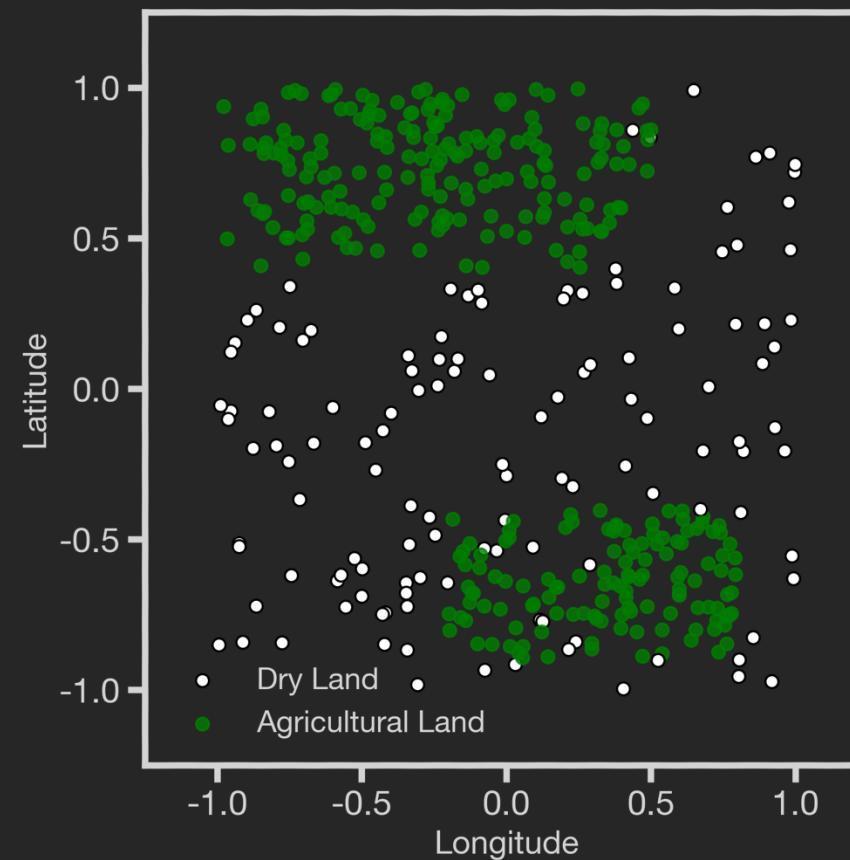
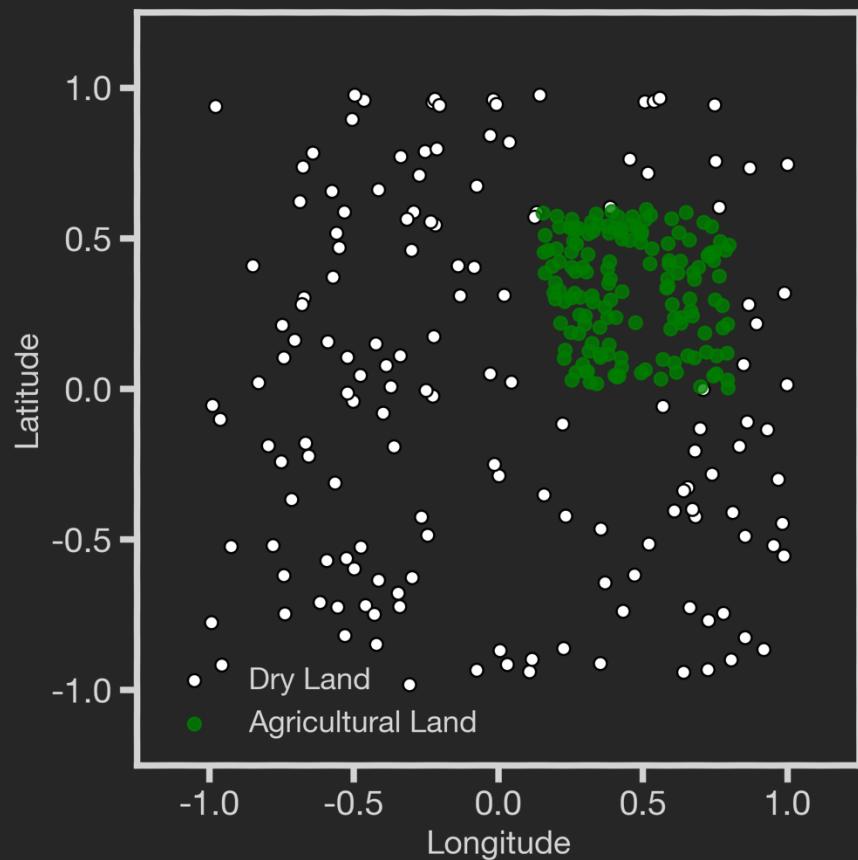
Motivation

This works well when the classification boundary has a nice simple geometry, but what about these?



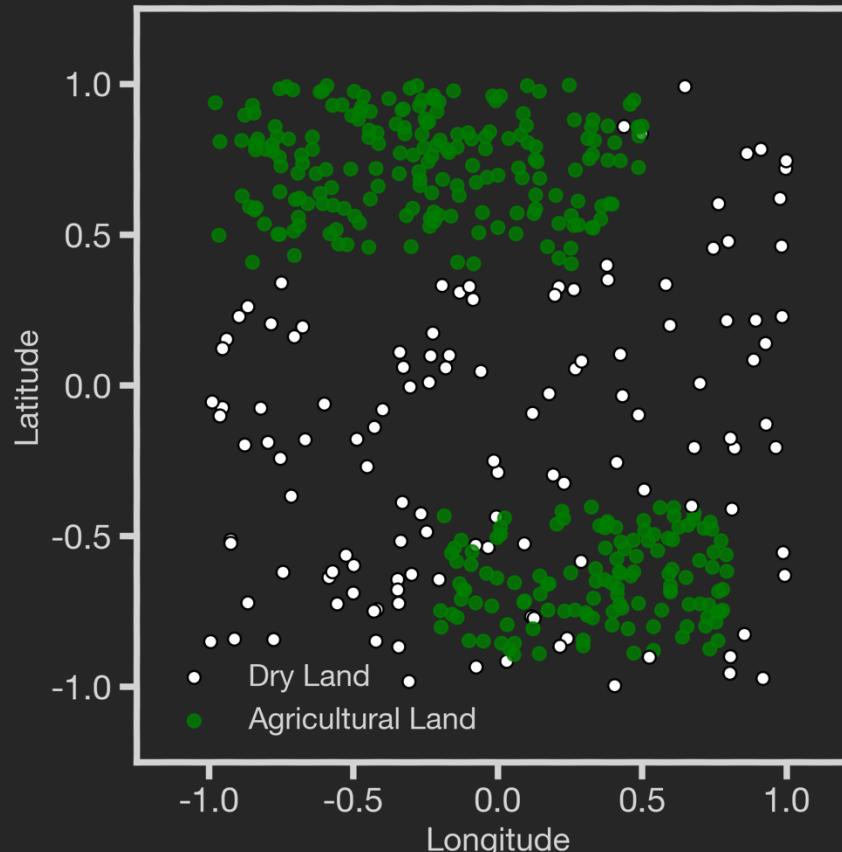
Motivation

Or these?



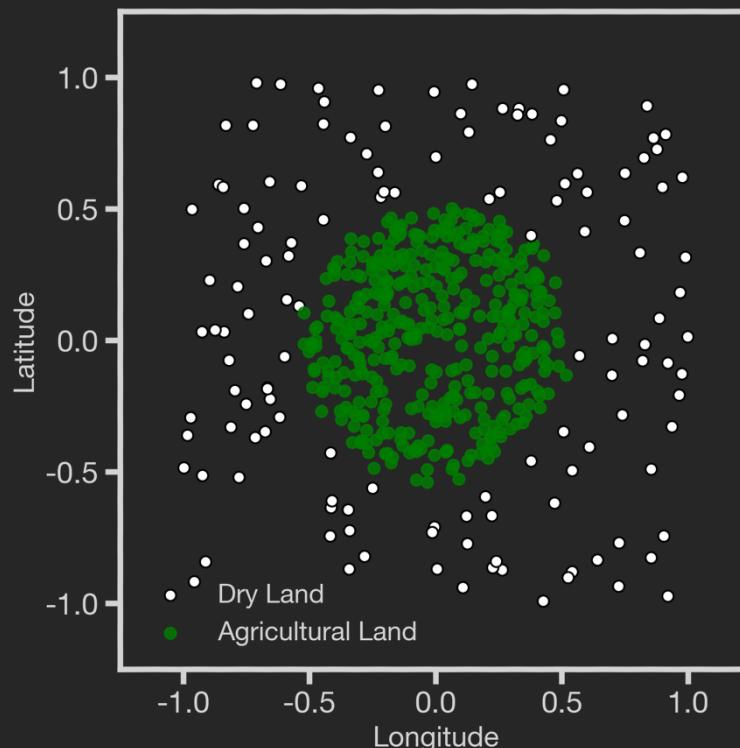
Motivation

In such datasets the classes are still well-separated in the feature space, but *the decision boundaries cannot easily be described by a single equation.*



Motivation

Logistic regression models with linear boundaries are intuitive to interpret by examining the impact of each predictor on the log-odds of a positive classification. It is less straightforward to interpret nonlinear decision boundaries and how they interact with individual predictor values.



$$x_1^2 + x_2^2 - 0.25 = 0$$

The modeling wish list

We would like to have models that

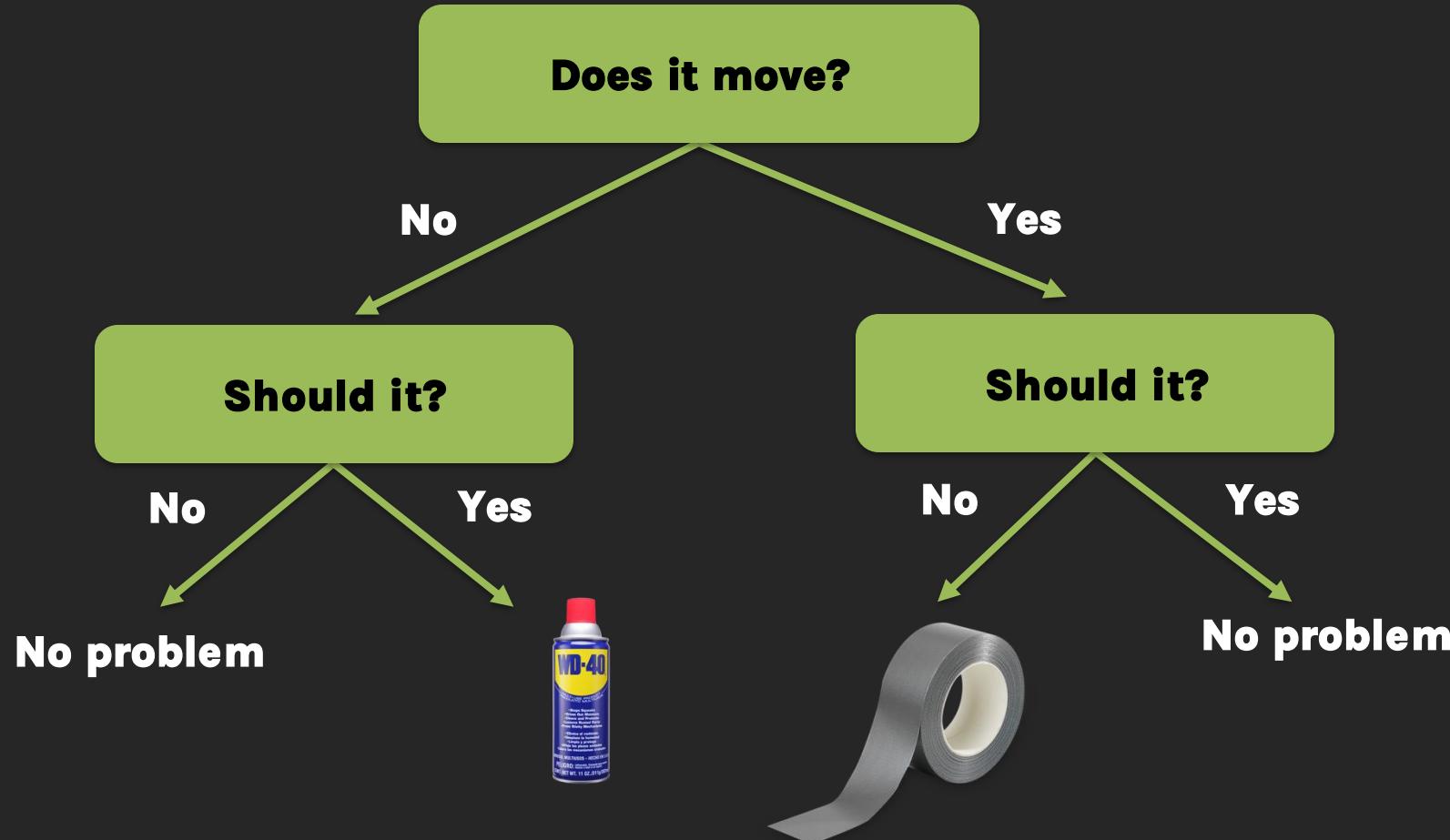
1. Allow for complex decision boundaries
2. Are also easy to interpret
3. Are straightforward and efficient to compute

Interpretable Models

People in every walk of life have long been using interpretable models for differentiating between classes of objects and phenomena.

For example, the Engineering Flowchart:

Engineering Flowchart



Interpretable Models

It turns out that the **simple flow chart** in our example can be formulated as a mathematical model for classification, and this model has the properties we desire:

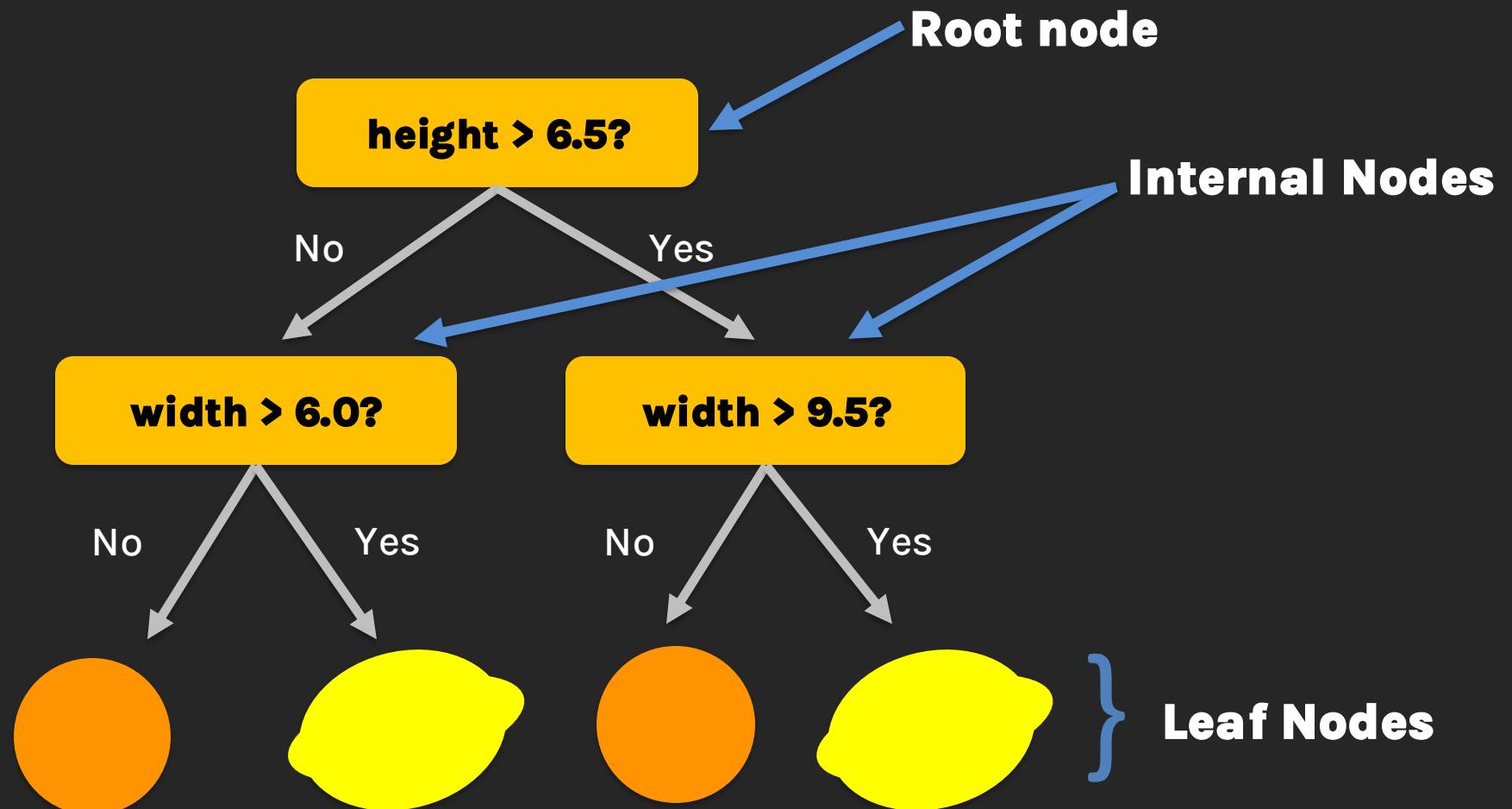
- The model is **interpretable** by humans.
- It has **sufficiently complex** decision boundaries.
- The decision boundaries are **locally linear**, which means each component of the decision boundary is simple to describe mathematically.

Outline

- Motivation
- Decision Trees – Classification
 - Intuition
 - Predictions
 - Splitting Criteria
 - Stopping Conditions

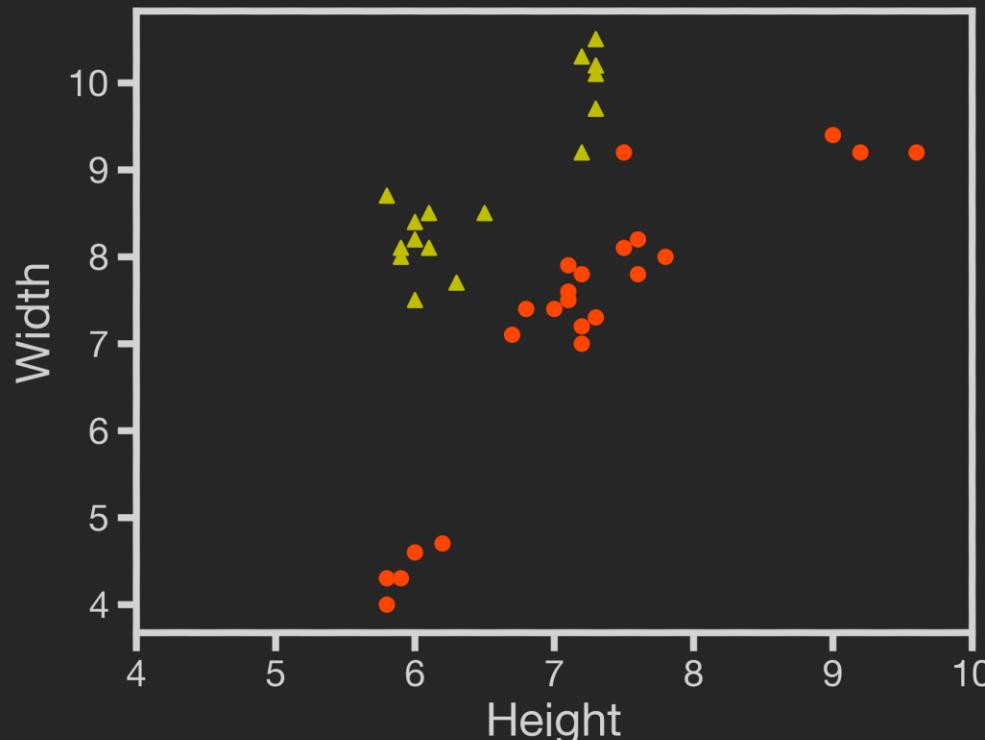
Example

A decision tree to classify fruits into lemons and oranges:



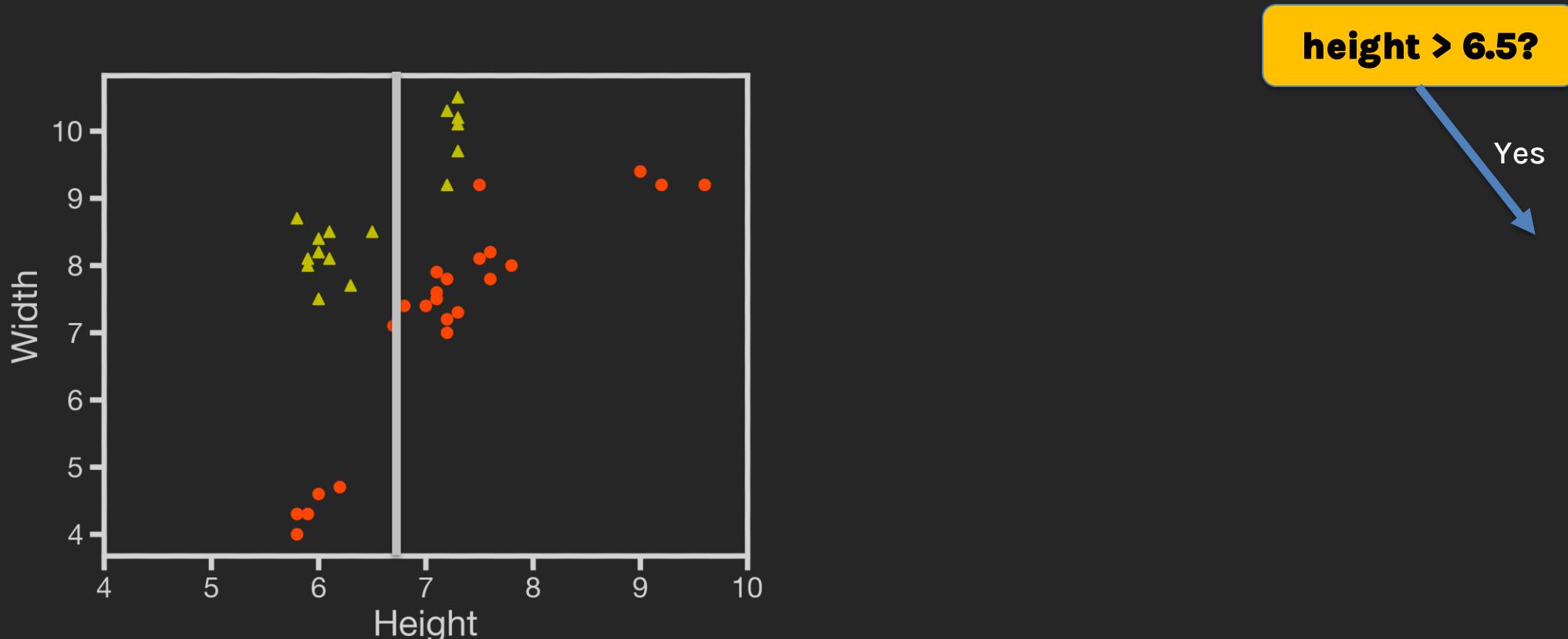
The Geometry of Flow Charts

Every flow chart tree corresponds to a partition of the feature space by **lines parallel to the axis or (hyper) planes**. Conversely, every such partition can be written as a flow chart tree.



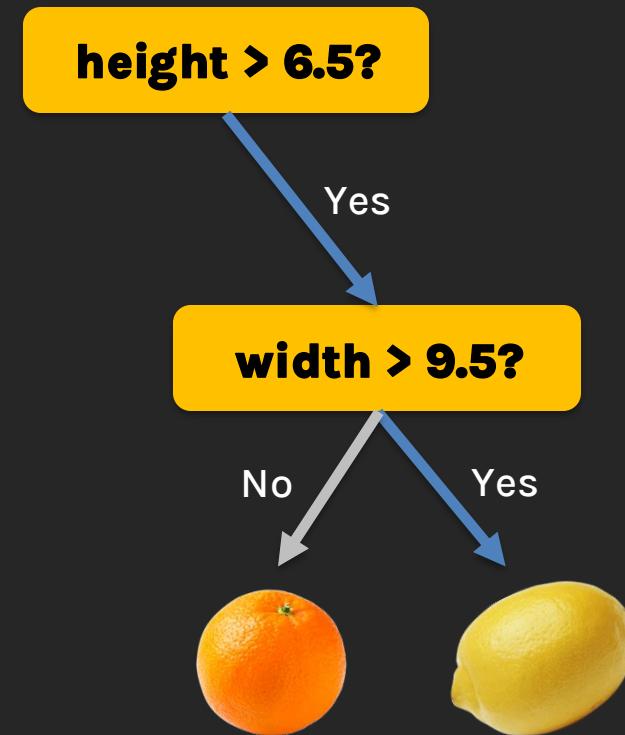
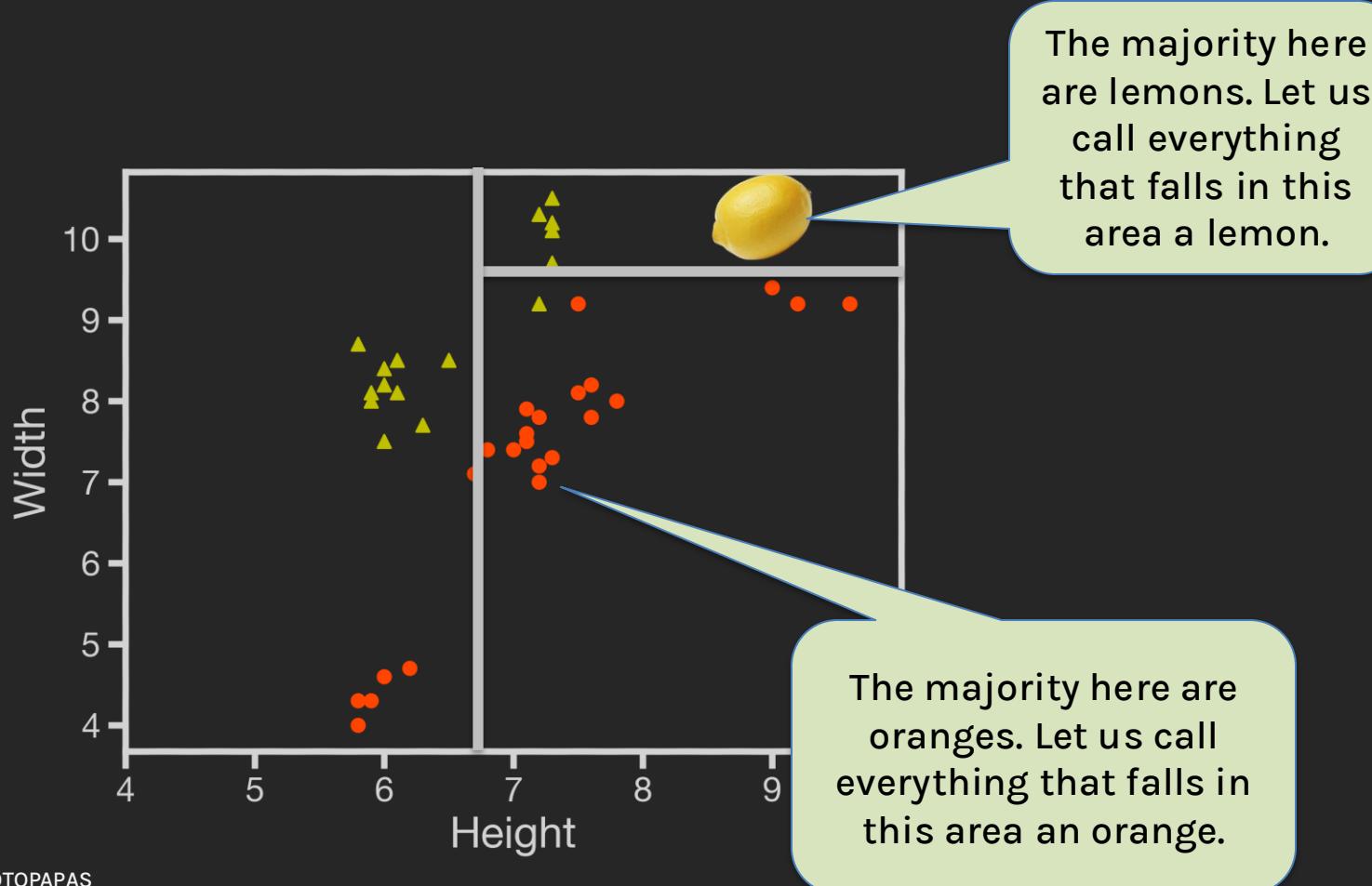
The Geometry of Flow Charts

Every flow chart node corresponds to a partition of the feature space by lines parallel to the axis or (hyper) planes. As a result, any partition created in this way can be represented as a flowchart tree.

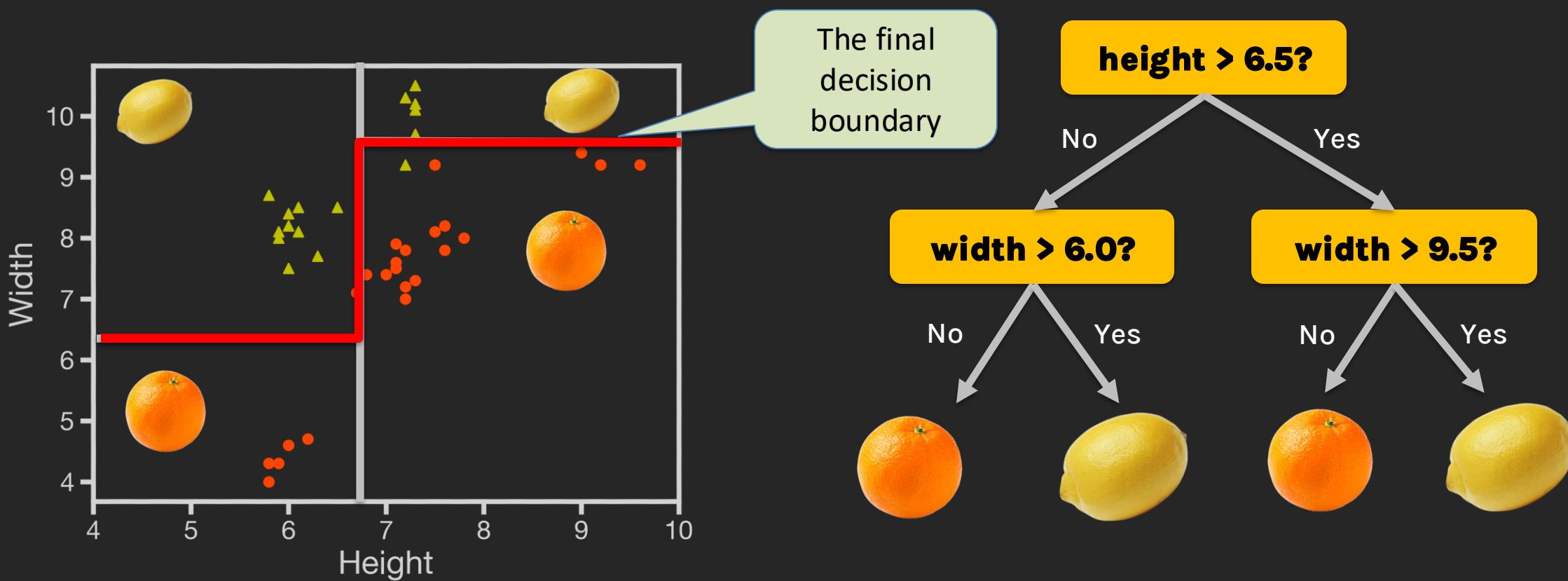


The Geometry of Flow Charts

Every flow chart node corresponds to a partition of the feature space by lines parallel to the axis or (hyper) planes. As a result, any partition created in this way can be represented as a flowchart tree.



The Geometry of Flow Charts

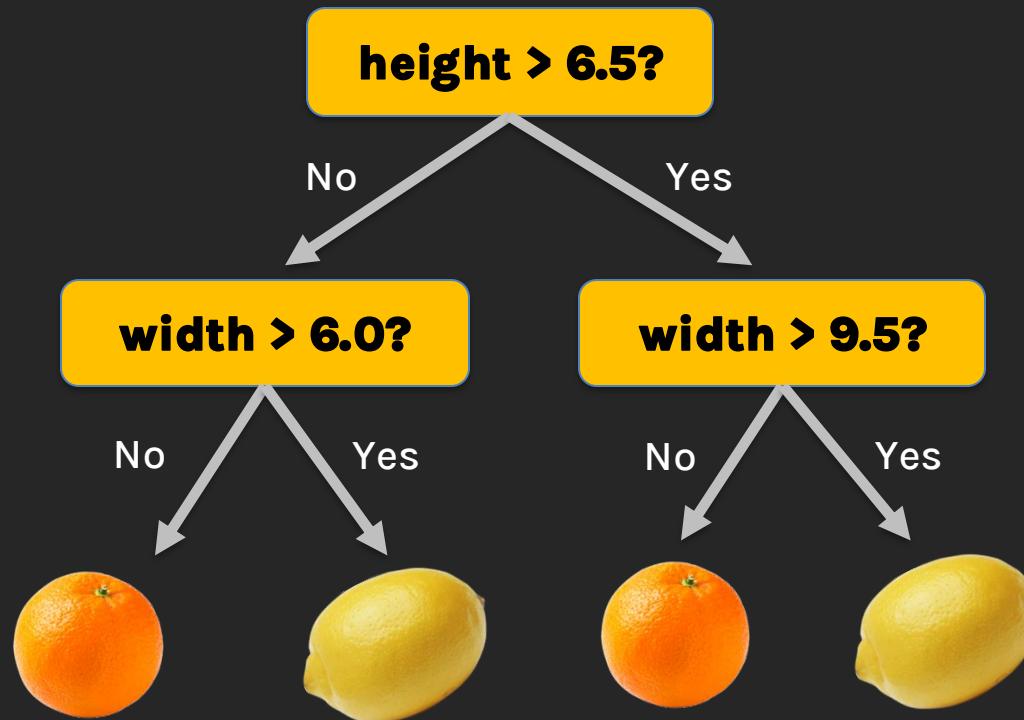


Outline

- Motivation
- Decision Trees – Classification
 - Intuition
 - Predictions
 - Splitting Criteria
 - Stopping Conditions

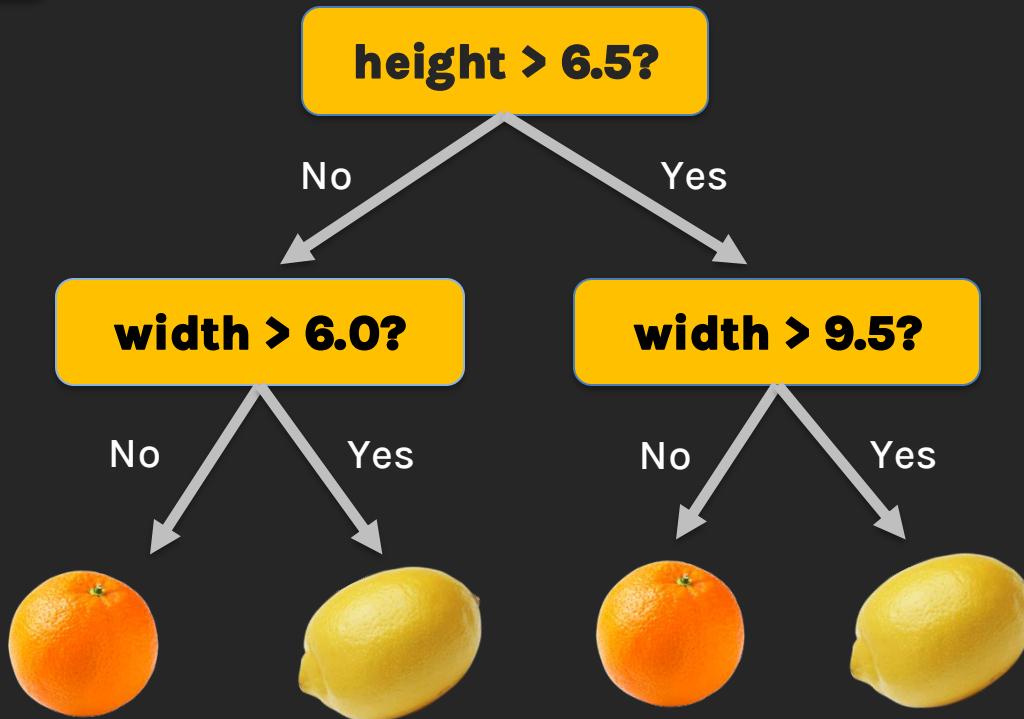
Predictions

Let's use the decision tree from before to make predictions.



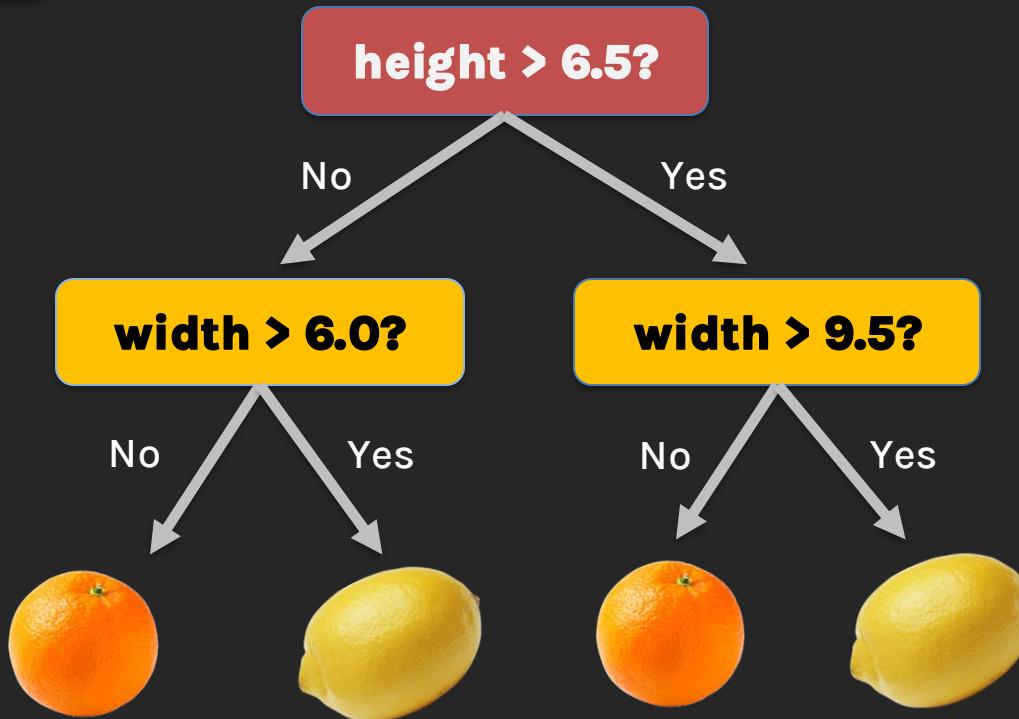
Predictions

? height = 5.9
 width = 5.8



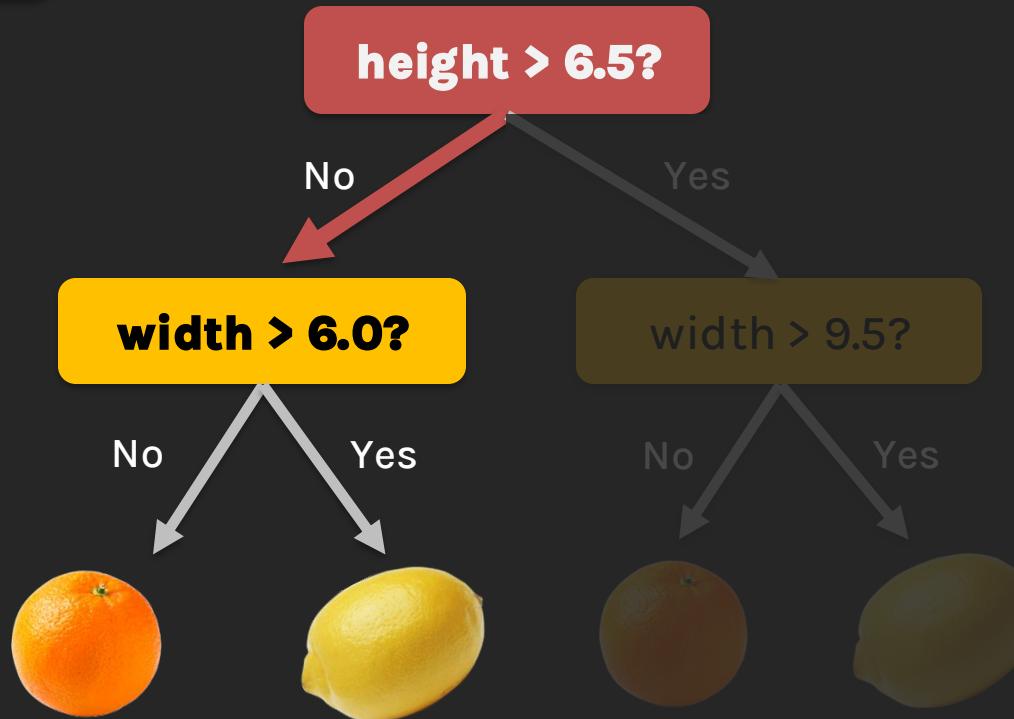
Predictions

? height = 5.9
 width = 5.8



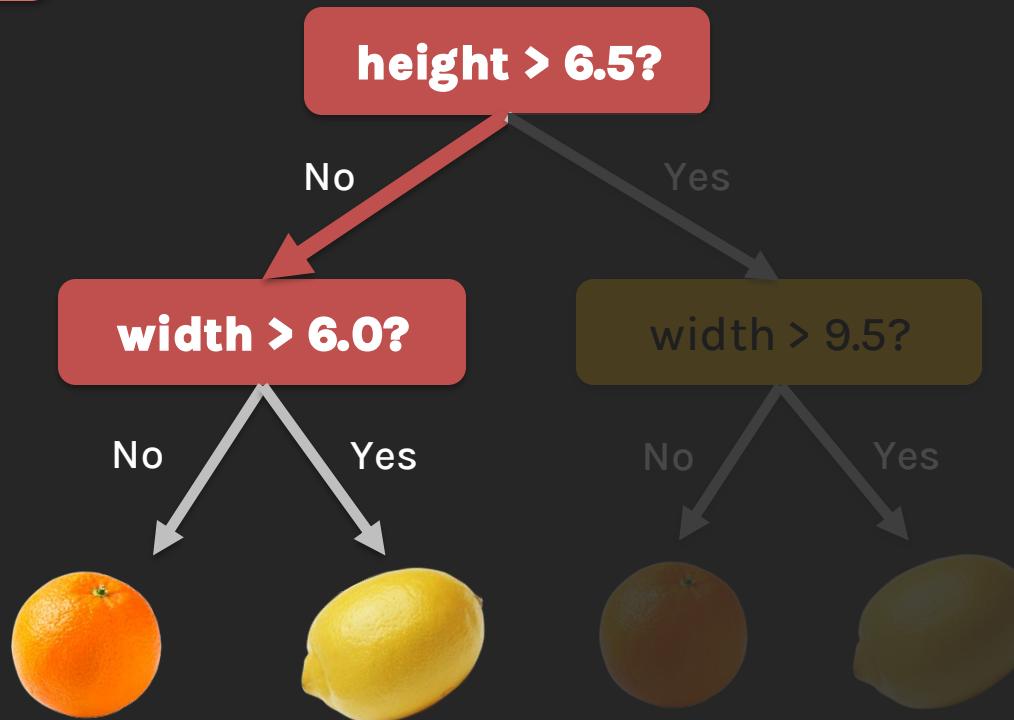
Predictions

? height = 5.9
 width = 5.8



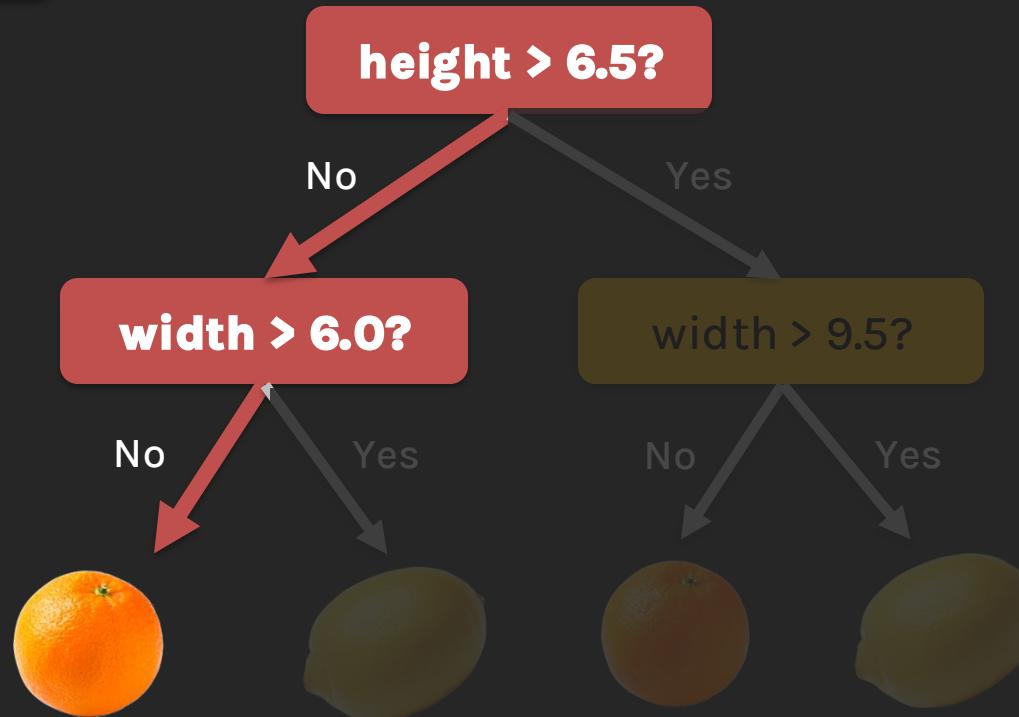
Predictions

? height = 5.9
 width = 5.8



Predictions

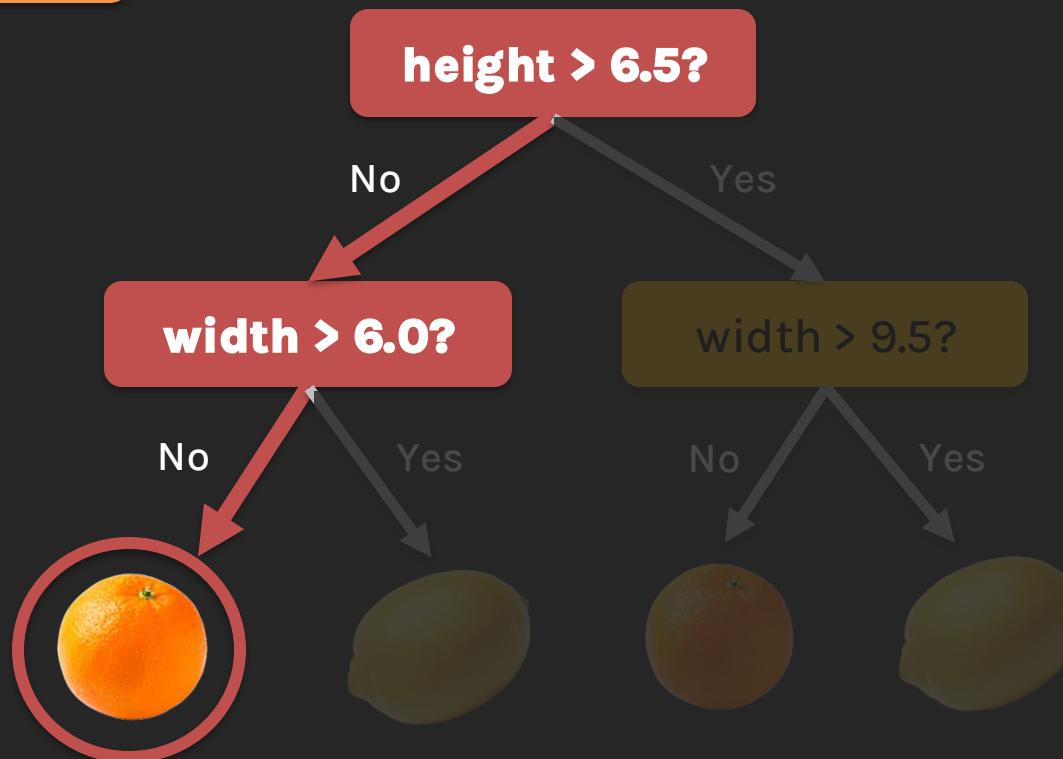
? height = 5.9
 width = 5.8



Predictions



**height = 5.9
width = 5.8**

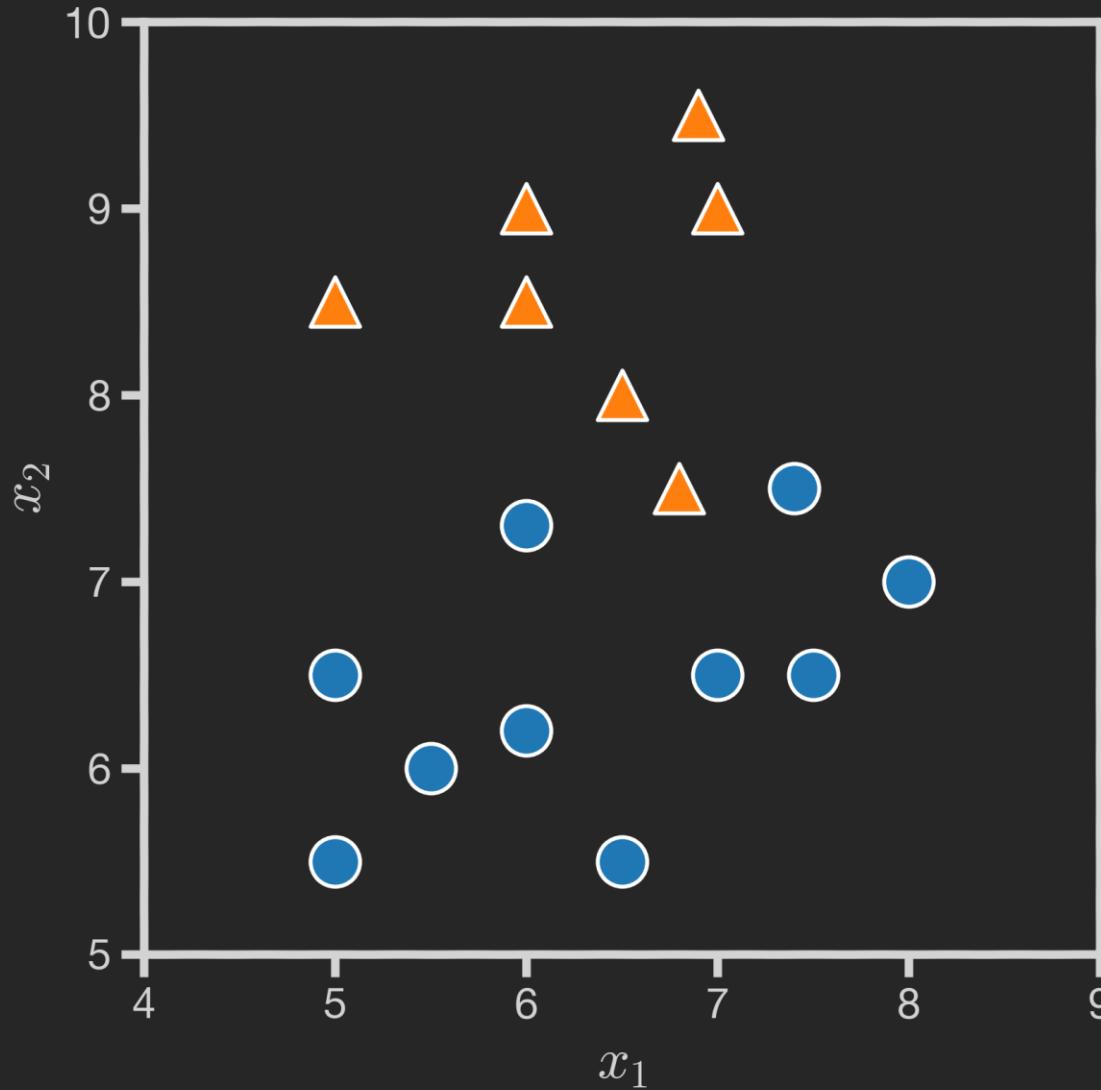


THIS METHOD IS ALSO CALLED TRAVERSING THE TREE

Outline

- Motivation
- Decision Trees – Classification
 - Intuition
 - Predictions
 - Splitting Criteria
 - Stopping Conditions

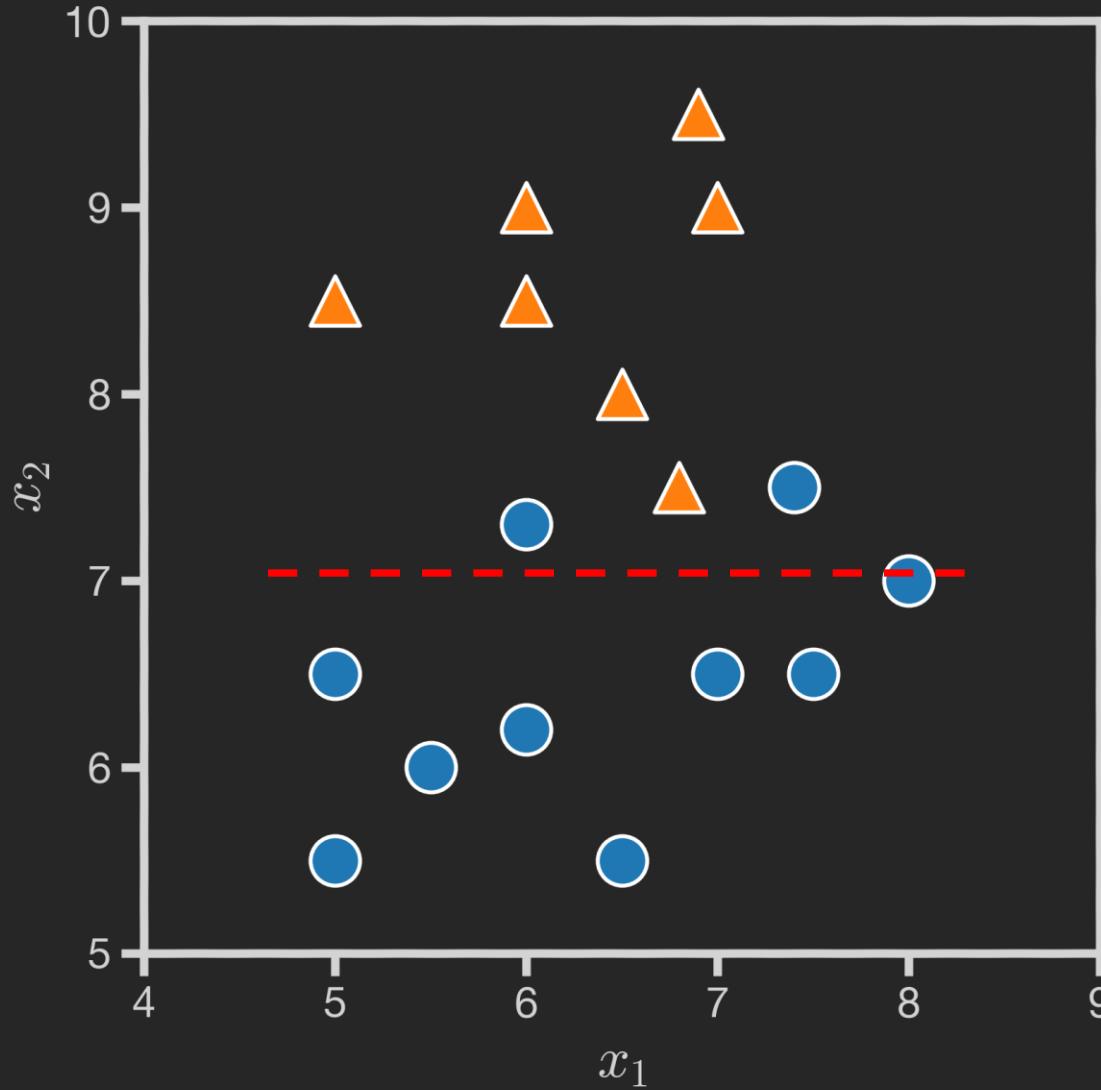
Splitting Criteria



Which gives a better split?

- Should it be split along x_1 or x_2 ?
- Where should we split?

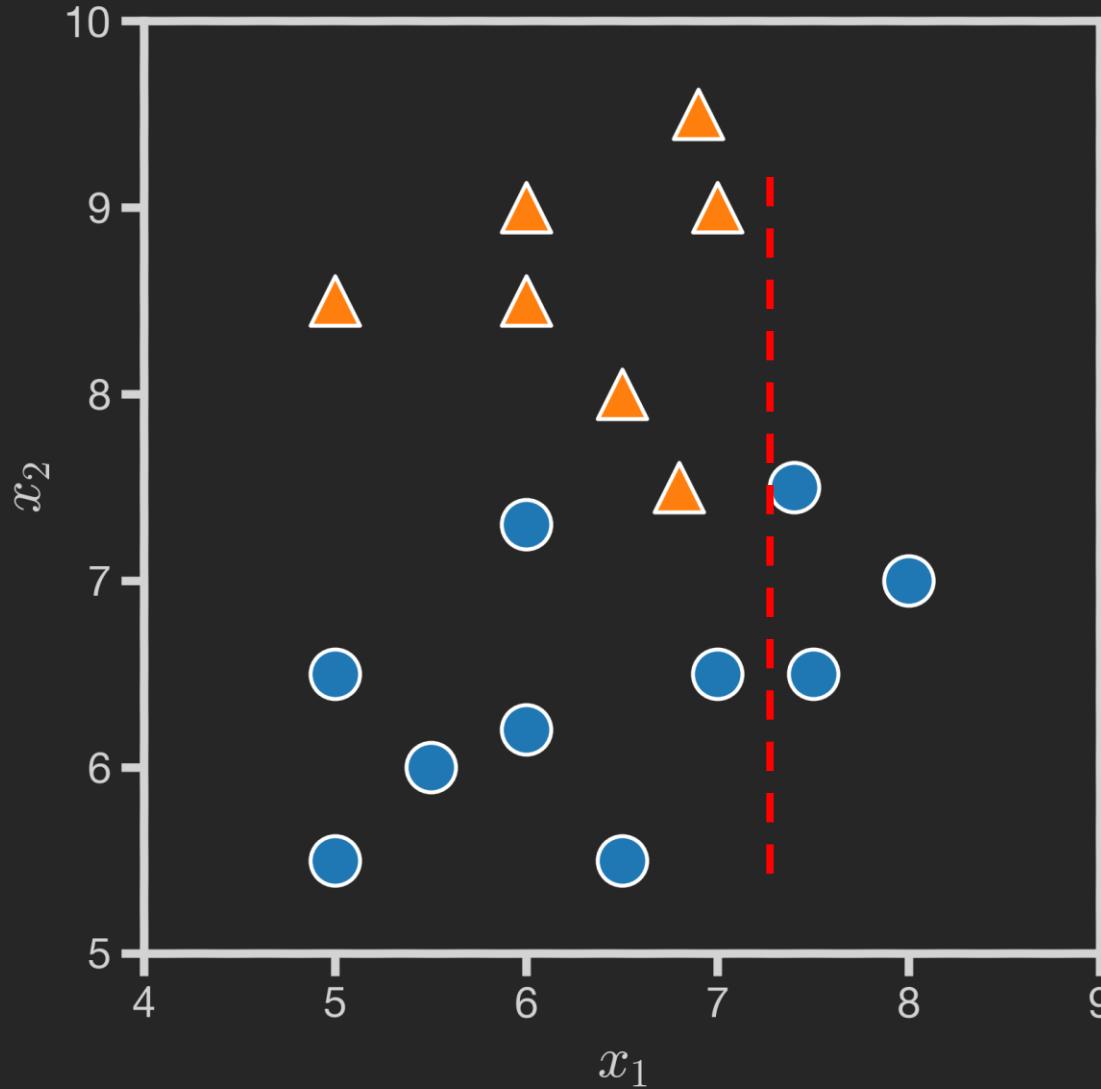
Splitting Criteria



Which gives a better split?

- Should it be split along x_1 or x_2 ?
- Where should we split?

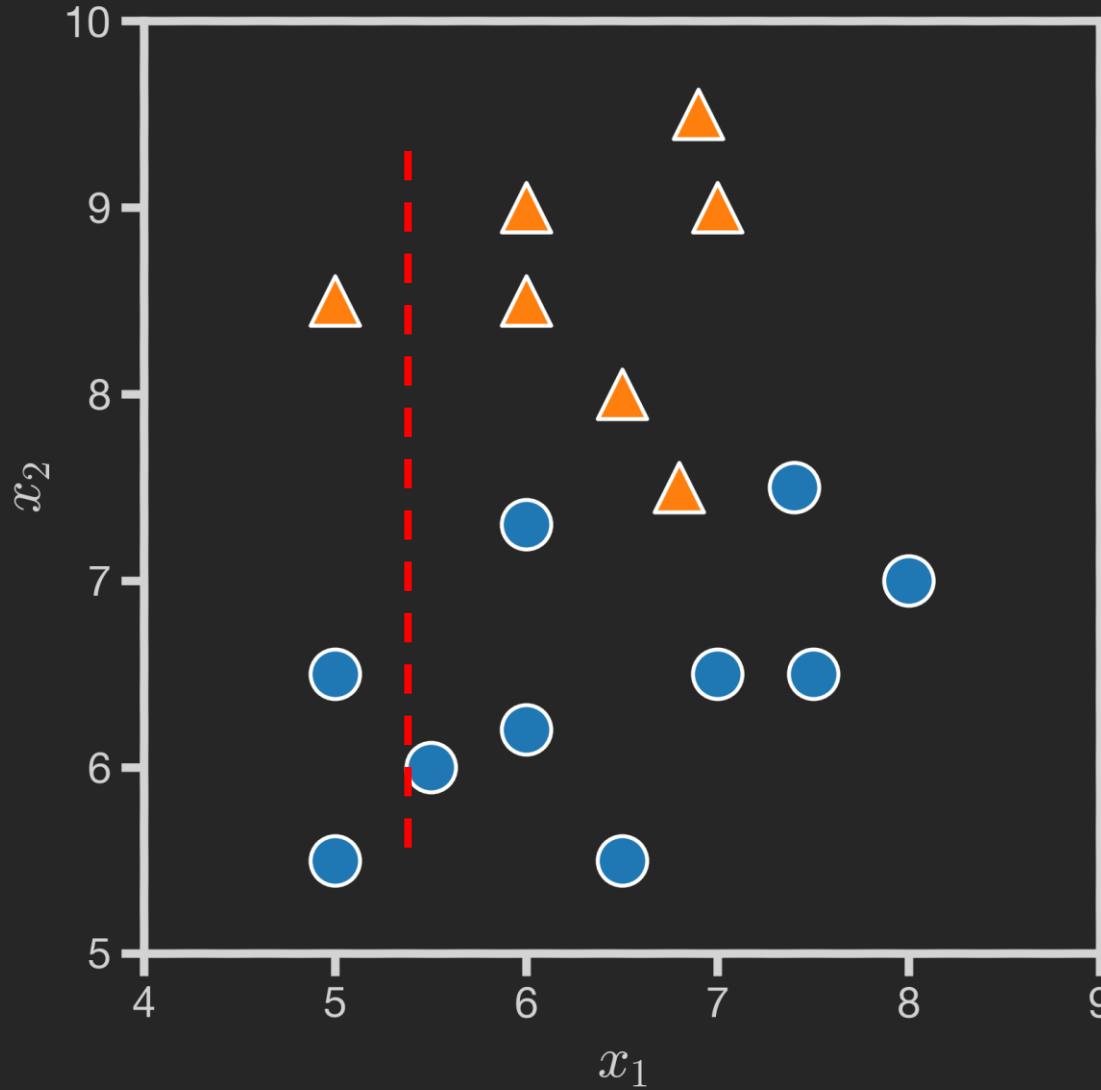
Splitting Criteria



Which gives a better split?

- Should it be split along x_1 or x_2 ?
- Where should we split?

Splitting Criteria



Which gives a better split?

- Should it be split along x_1 or x_2 ?
- Where should we split?

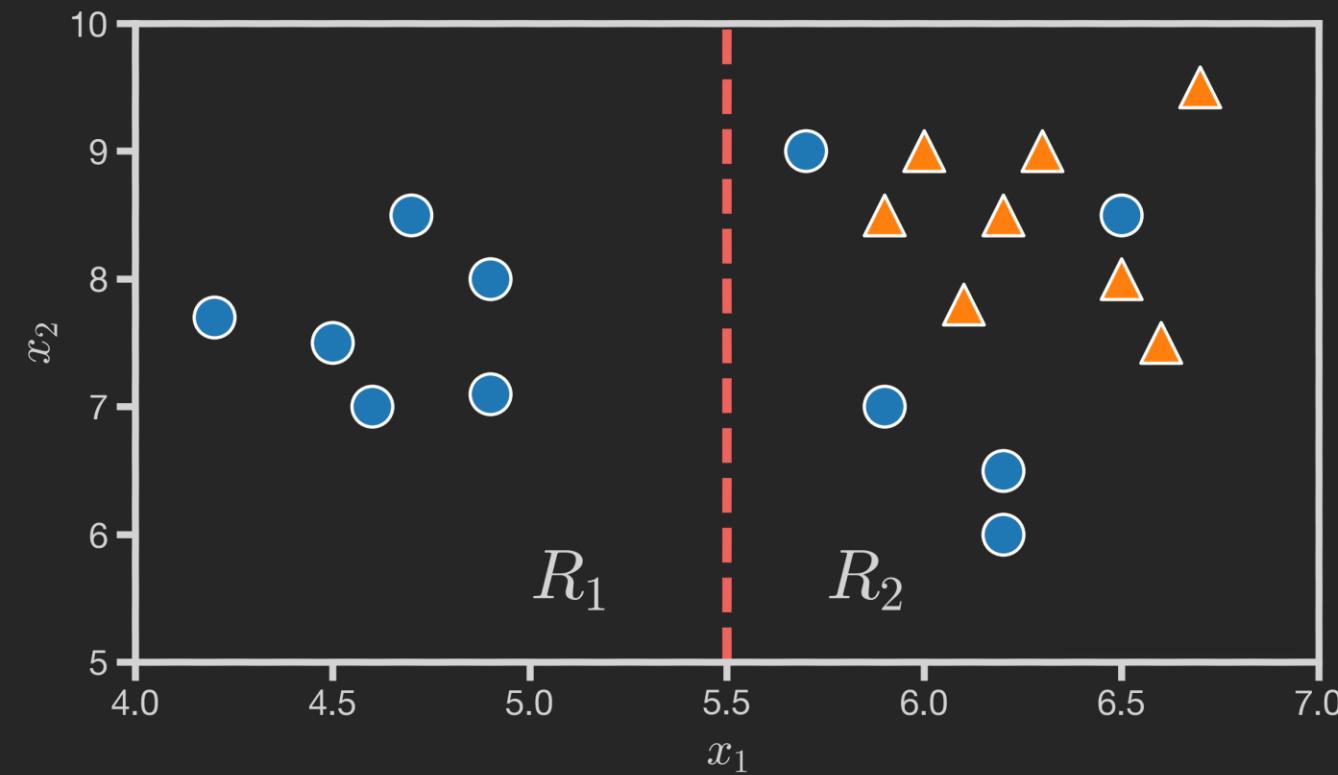
Optimality of Splitting

While there is no ‘correct’ way to define an optimal split, there are some commonsense guidelines for every splitting criterion:

- The regions in the feature space should grow progressively **purer** with the number of splits. That is, we should see that each region ‘specializes’ towards a single class.
- We should end up with **no empty regions** – every region should contain training points.

Note: The splitting criterion of a split could take a **differentiable or non-differentiable form**.

Classification Error



Consider region R_2 :

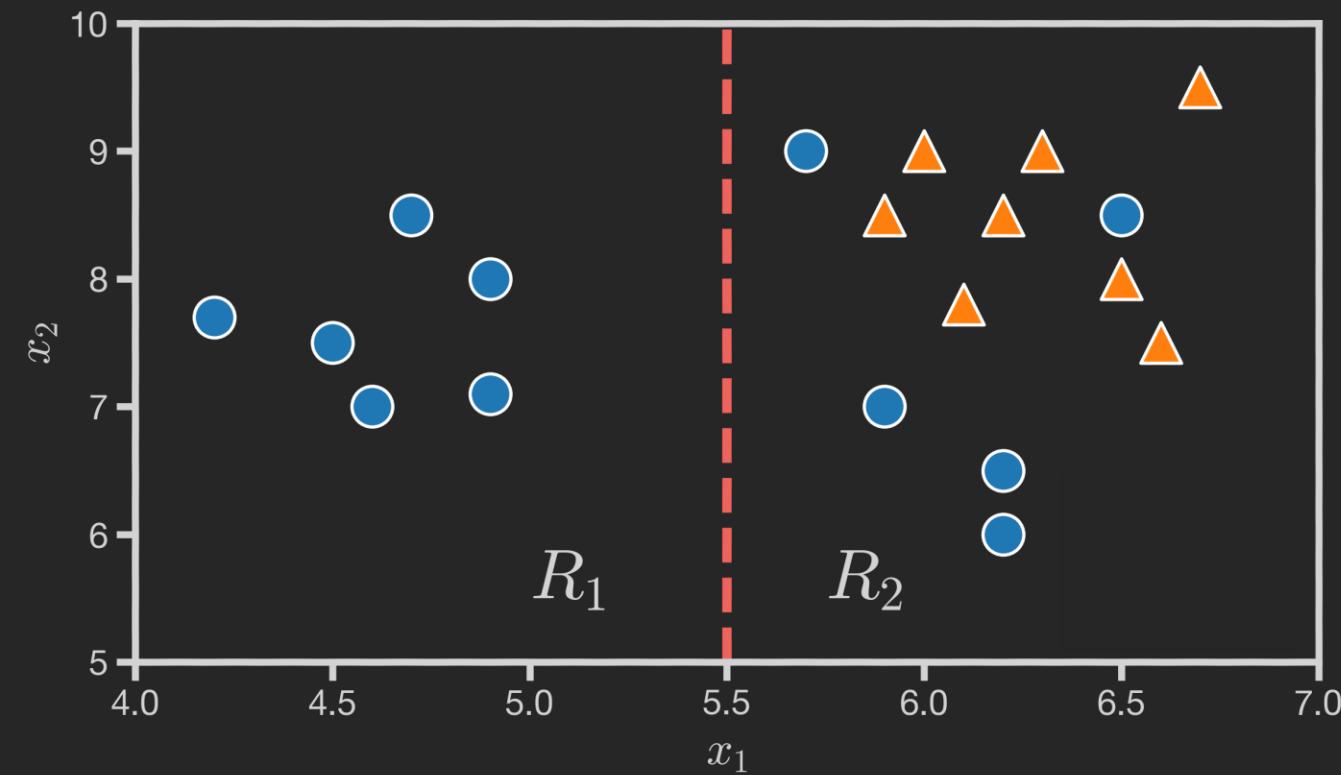
$$\begin{aligned}\text{Classification error} &= \frac{\text{error}}{\text{total}} \\ &= \frac{\text{Number of minority class data points}}{\text{Total number of data points}} \\ &= \frac{\text{Number of } \bullet}{\text{Total number of data points}} \\ &= 1 - \frac{\text{Number of majority data points}}{\text{Total number of data points}}\end{aligned}$$

Classification Error

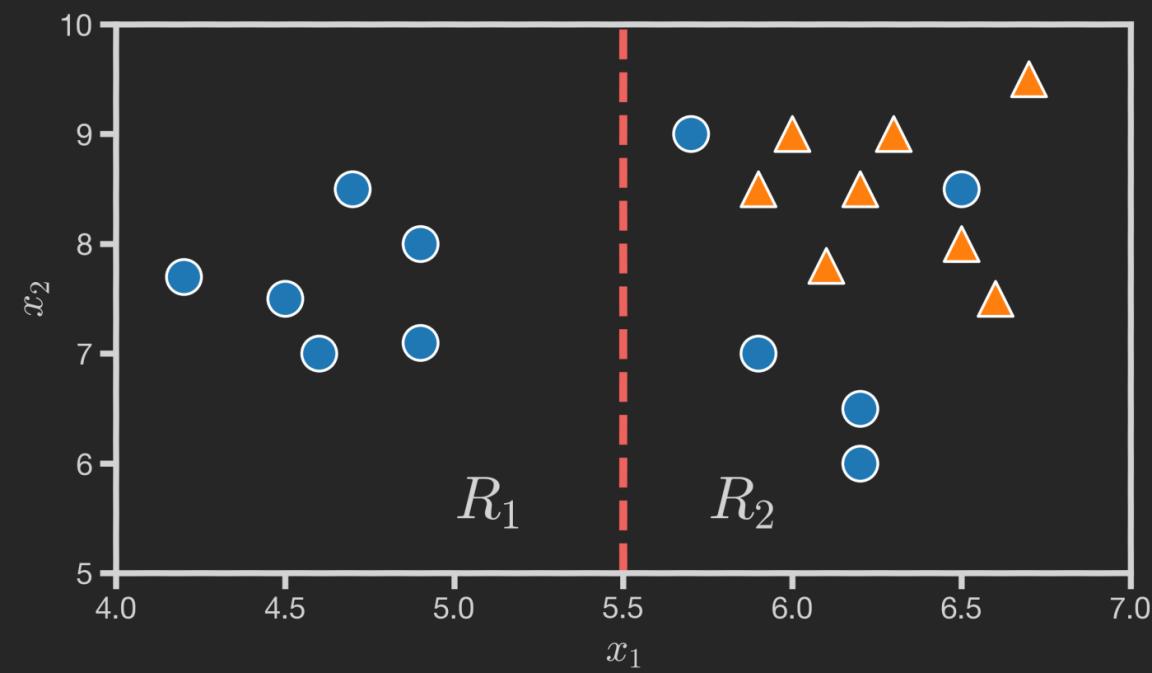
Classification error in region R_2 :

$$\begin{aligned} &= 1 - \frac{\text{Number of majority data points}}{\text{Total number of data points}} \\ &= 1 - \Psi(\Delta | R_2) \\ &= 1 - \max_k (\Psi(k | R_r)) \end{aligned}$$

$\Psi(k | R_r)$ is the proportion of training points in R_2 that are labeled class k.



Classification Error



$Error = 1 - \max_k(\Psi(k|R_r))$

| | | |
|-------|---|---|
| | | |
| R_1 | 6 | 0 |
| R_2 | 5 | 8 |

$1 - \max\left(\frac{6}{6}, \frac{0}{6}\right) = 0$

$1 - \max\left(\frac{5}{13}, \frac{8}{13}\right) = \frac{5}{13} = 0.38$

Classification Error

In general:

Assume we have **P predictors (or features)** and **K classes**. Suppose we select the p^{th} predictor and split a region along the **threshold t_p** .

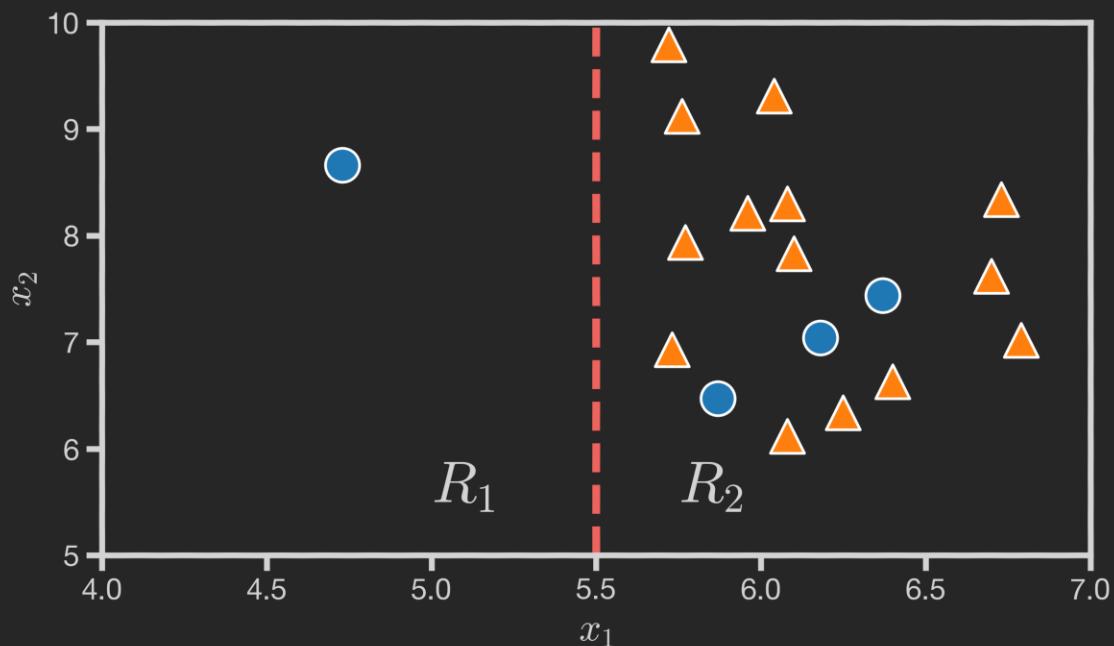
We can assess the quality of this split by calculating the **classification error** made by each newly created region:

$$\text{Error } (R_r | p, t_p) = 1 - \max_k(\Psi(k|R_r))$$

where $\Psi(k|R_r)$ is the proportion of training points in R_r that are labeled class k .

Classification Error

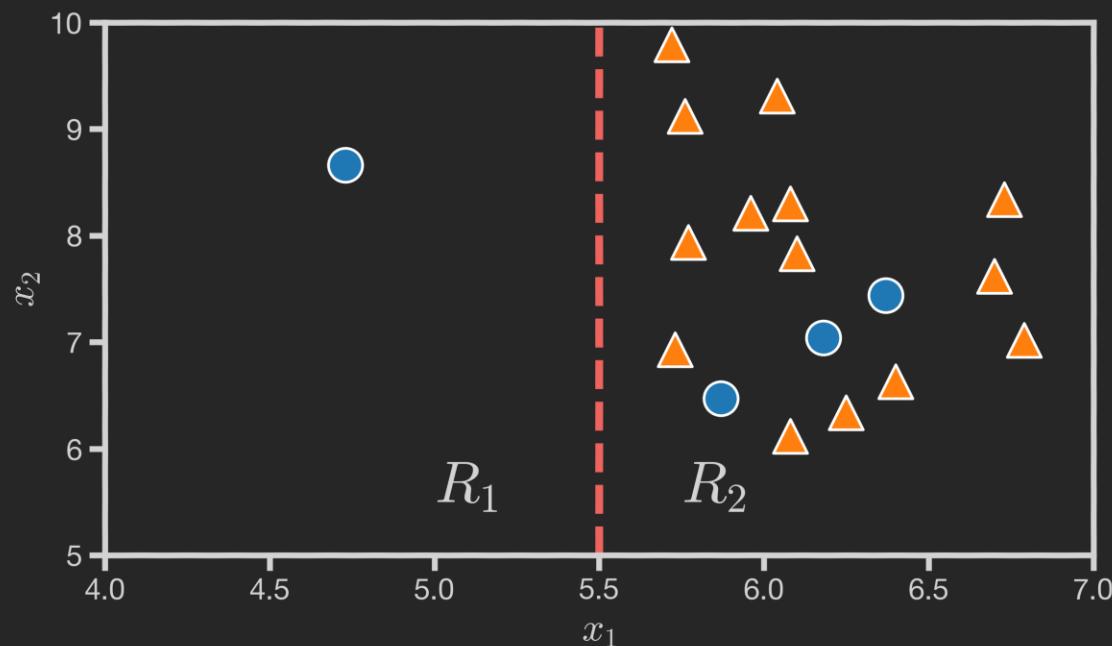
Now calculate the error for the following split:



| | | | |
|-------|---|----|--|
| | | | $Error = 1 - \max_k(\Psi(k R_r))$ |
| R_1 | 1 | 0 | $1 - \max\left(\frac{1}{1}, \frac{0}{1}\right) = 0$ |
| R_2 | 3 | 14 | $1 - \max\left(\frac{3}{17}, \frac{14}{17}\right) = \frac{3}{17} = 0.18$ |

R_1 has a smaller error than R_2 .
Does that mean this is a good split?

Classification Error



We need to take the **weighted average** over both regions so the number of points in each region is taken into consideration:

$$\min_{p, t_p} \left[\frac{N_1}{N} \text{Error}(R_1 | p, t_p) + \frac{N_2}{N} \text{Error}(R_2 | p, t_p) \right]$$

where N_r is the number of training points inside region R_r .

Gini Index

Assume we have **P predictors** and **K classes**. Suppose we select the p^{th} predictor and split a region along the **threshold t_p** .

We can assess the quality of this split by measuring the **Gini Index** made by each newly created region by calculating:

1 - sum of the squares of the proportions of points from the k -th class in the region r (PSI).

$$Gini(R_r | p, t_p) = 1 - \sum_k \Psi(k|R_r)^2$$

Question: What is the effect of squaring the proportions of each class?



Which of the following statements best describes the effect of squaring the proportions of each class when calculating the Gini Index?

Options:

- A) Squaring the proportions magnifies the influence of majority classes in a region, thereby overstating their significance.
- B) The squaring process accentuates the contrast between pure regions and mixed ones, enabling the index to better differentiate between quality splits.
- C) The act of squaring diminishes the index's sensitivity to minor fluctuations in class distributions, making it more stable.
- D) Squaring the proportions directly correlates the Gini Index with the count of predictors P , biasing the selection of split predictors.

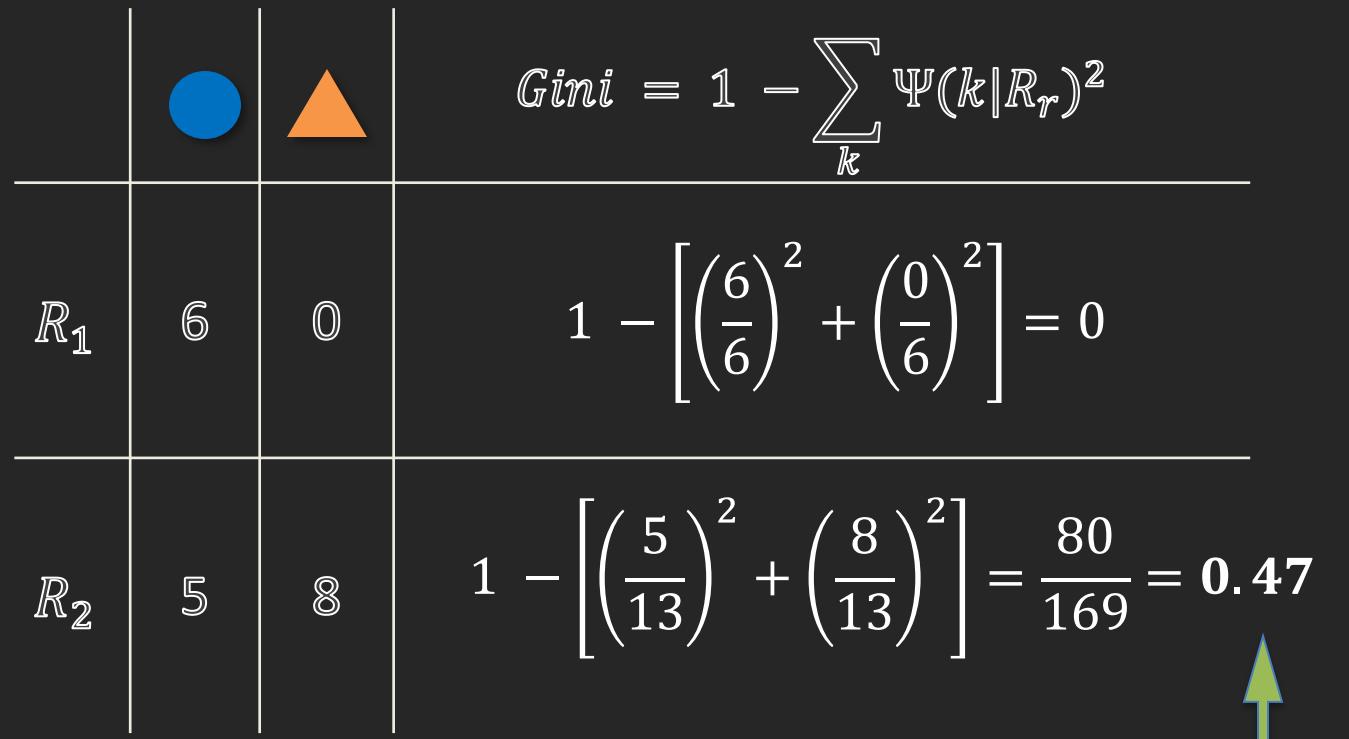
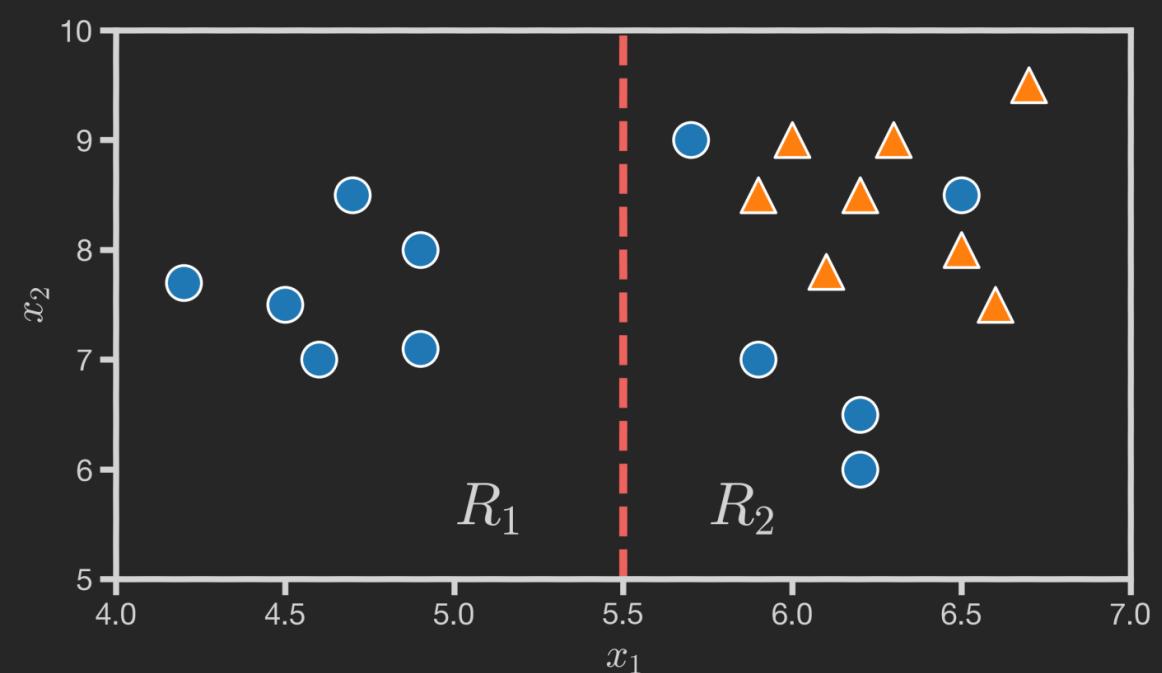


Which of the following statements best describes the effect of squaring the proportions of each class when calculating the Gini Index?

Options:

- A) Squaring the proportions magnifies the influence of majority classes in a region, thereby overstating their significance.
- B) The squaring process accentuates the contrast between pure regions and mixed ones, enabling the index to better differentiate between quality splits.
- C) The act of squaring diminishes the index's sensitivity to minor fluctuations in class distributions, making it more stable.
- D) Squaring the proportions directly correlates the Gini Index with the count of predictors P , biasing the selection of split predictors.

Gini Index



Compared to 0.38 when we used **classification error**

Gini Index

We can now try to find the predictor p and the threshold t_p that minimize the **weighted average Gini Index** over the two regions:

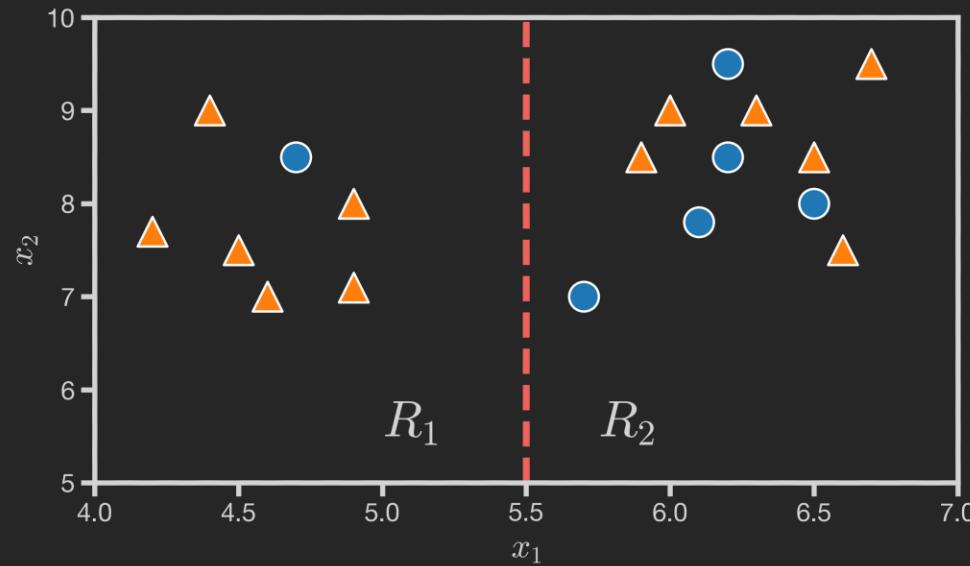
$$\min_{p,t_p} \left[\frac{N_1}{N} Gini(R_1|p, t_p) + \frac{N_2}{N} Gini(R_2|p, t_p) \right]$$

where $N_{1,2}$ is the number of training points inside region $R_{1,2}$.

Information Theory

The last metric for evaluating the quality of a split is motivated by metrics of uncertainty in information theory.

Question: In the below plot, which region is ‘purer’?



While both regions are impure, R_1 clearly sends a **stronger ‘signal’** for class 2 than R_2 .

Information Theory

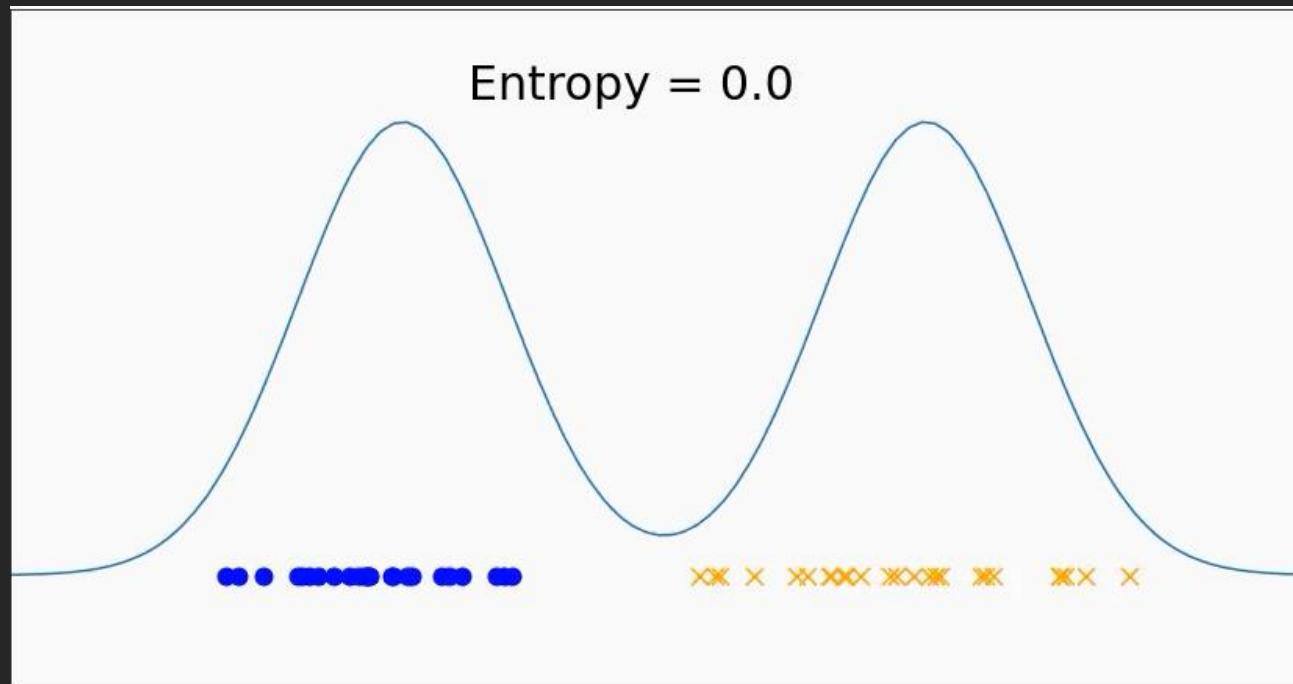
One way to quantify the strength of a signal in a particular region is to analyze the **distribution of classes** within the **region**. We compute the **entropy** of this distribution.

For a random variable with a discrete distribution, the entropy is computed by:

$$H(x) = - \sum_{x \in X} \Psi(x) \log_2 \Psi(x)$$

Information Theory

$$H(x) = - \sum_{x \in X} \Psi(x) \log_2 \Psi(x)$$



Entropy

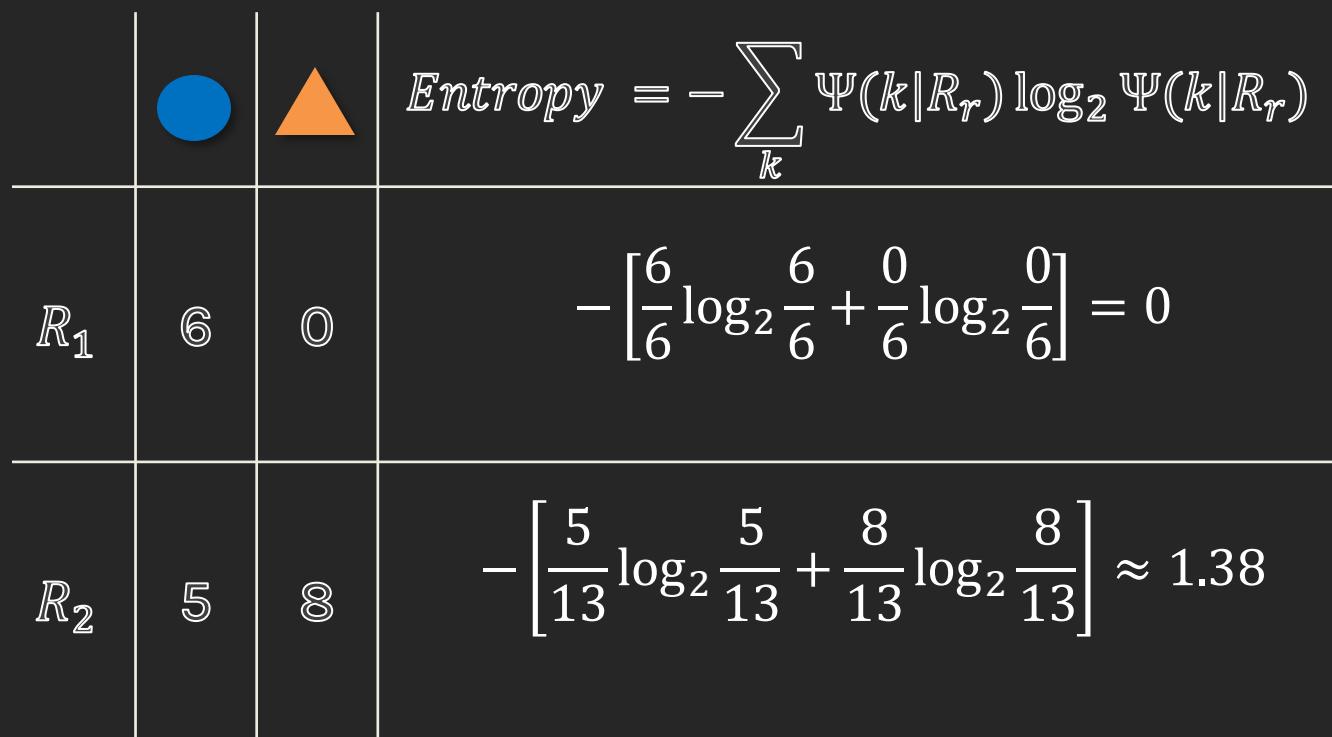
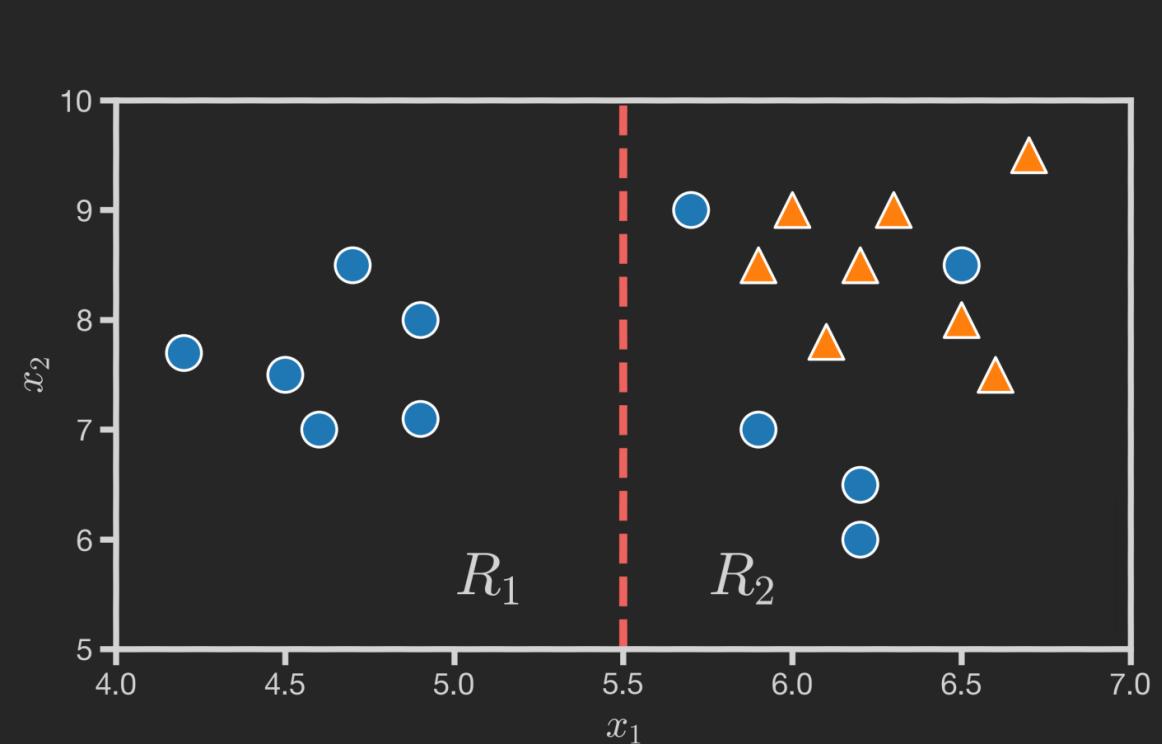
Assume we have **P predictors** and **K classes**. Suppose we select the p^{th} predictor and split a region along the **threshold t_p** .

We can assess the quality of this split by measuring the **entropy** of the class distribution in each newly created region by calculating:

$$\text{Entropy}(R_r | p, t_p) = - \sum_k \Psi(k|R_r) \log_2 \Psi(k|R_r)$$

Note: We are actually computing the conditional entropy of the distribution of training points amongst the K classes given that the point is in region r .

Entropy



Entropy

We can now try to find the predictor p and the threshold t_p that minimize the **weighted average entropy** over the two regions:

$$\min_{p,t_p} \left[\frac{N_1}{N} \text{Entropy}(R_1|p, t_p) + \frac{N_2}{N} \text{Entropy}(R_2|p, t_p) \right]$$

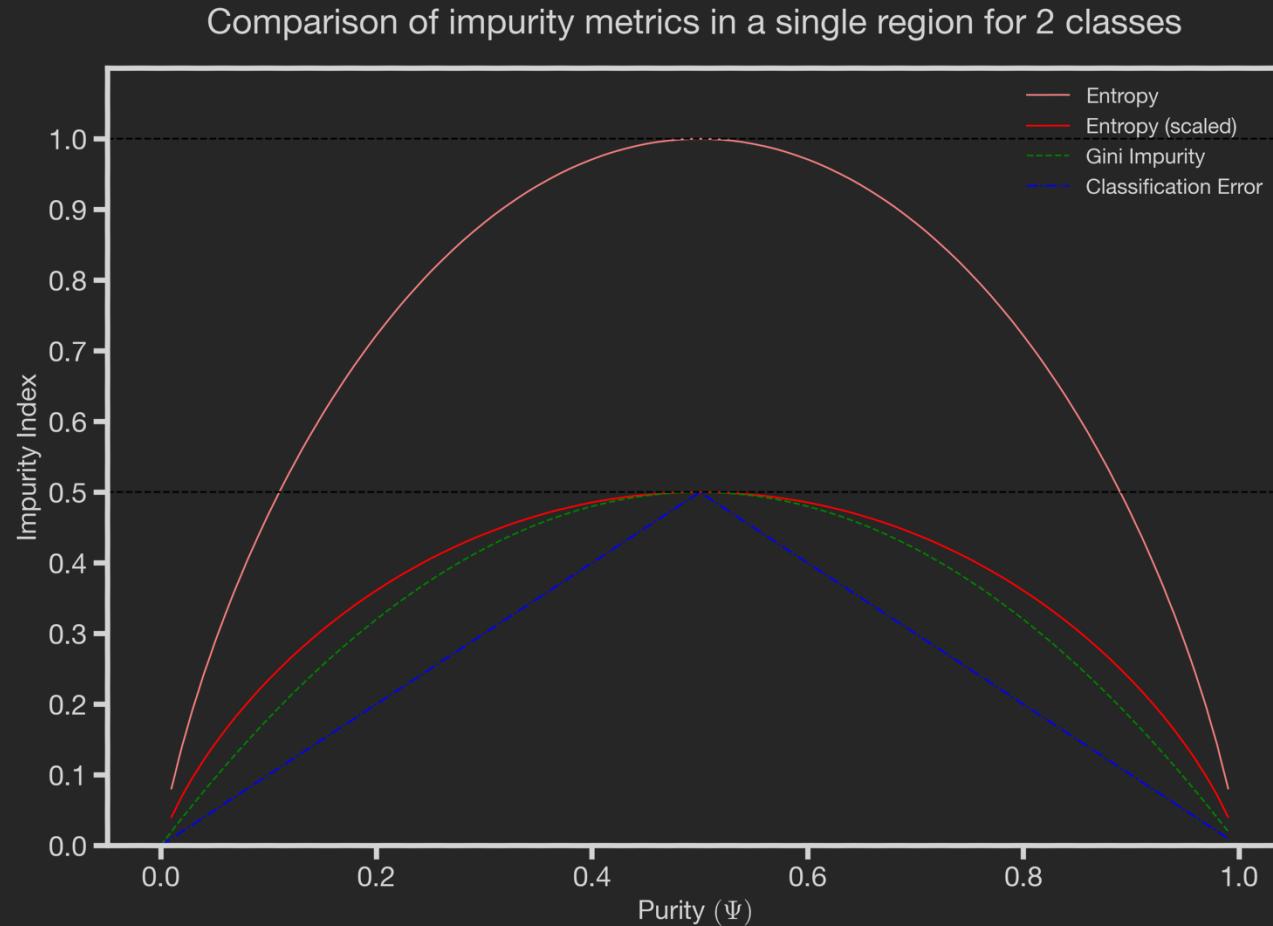
where $N_{1,2}$ is the number of training points inside region $R_{1,2}$.

Entropy

| Error Measurement | Value |
|----------------------|-------|
| Classification Error | 0.38 |
| Gini Index | 0.47 |
| Entropy | 1.38 |

Comparison of Criteria

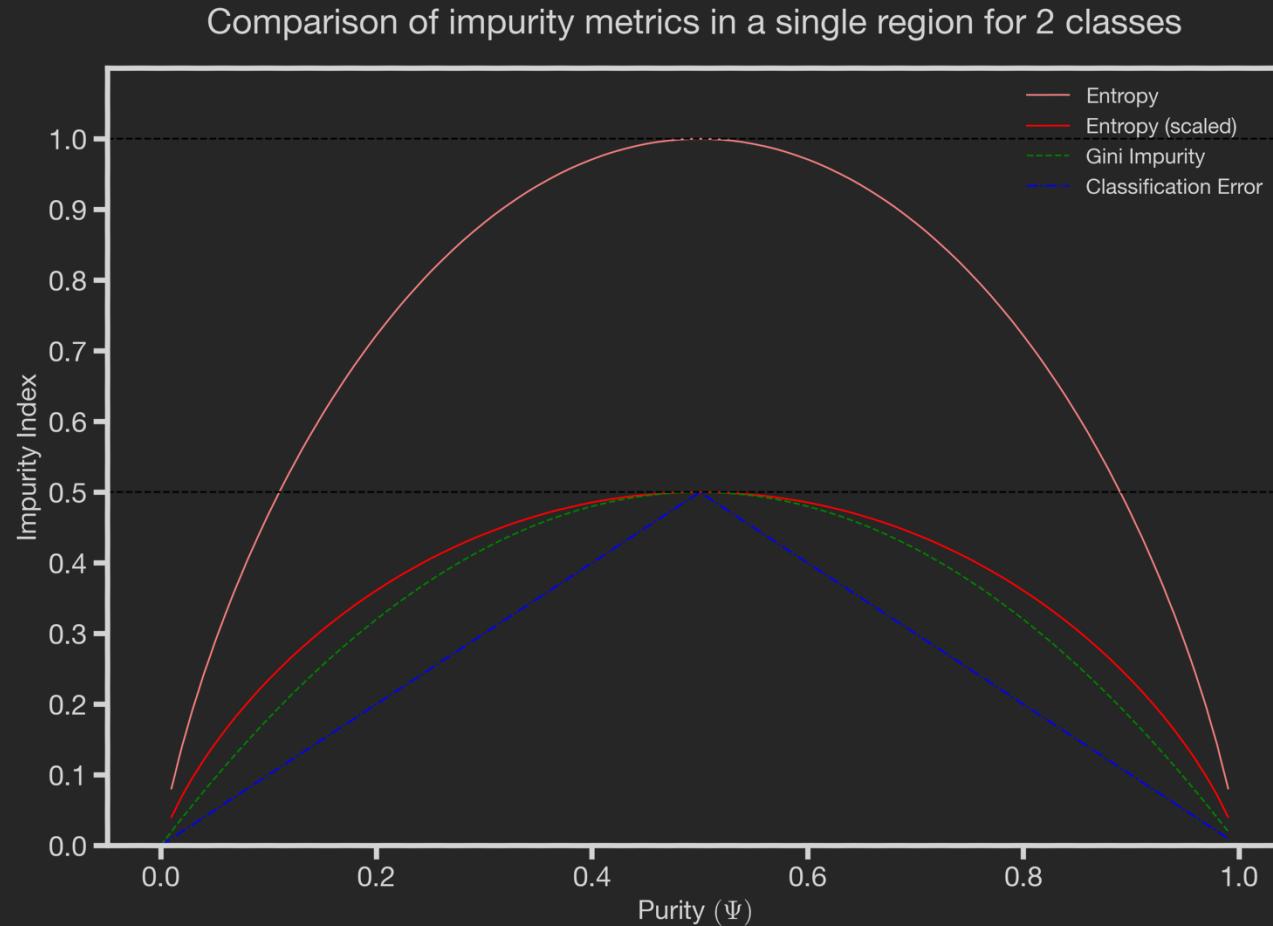
Question: Which of the three criteria fits our guideline the best?



Note: Sklearn uses the unscaled entropy as a splitting criterion

Comparison of Criteria

Question: Which of the three criteria fits our guideline the best?



Entropy **penalizes
impurity the most**



Learning the Model

Learning the ‘optimal’ decision tree for any given set of data is NP complete (intractable) for numerous simple definitions of ‘optimal’. Instead, we will use a **greedy algorithm**.

1. Start with an empty decision tree (undivided feature space)
2. Choose the ‘optimal’ predictor on which to split and choose the ‘optimal’ threshold value for splitting.
3. Recurse on each new node until the **stopping condition** is met.
4. For the case of classification, predict each region to have a class label based on the largest class of the training points in that region (majority class).

Learning the Model

Recurse on each new node until the **stopping condition** is met.

Please download and install the Slido app on all computers you use



What are two key advantages of decision tree models over logistic regression, especially in situations where classes are well-separated but boundaries are complex?

- ① Start presenting to display the poll results on this slide.

Please download and install the Slido app on all computers you use



How are predictions made using a decision tree model?

- ① Start presenting to display the poll results on this slide.

Please download and install the Slido app on all computers you use



Why is it desirable to have progressively purer regions as we move down the tree in a decision tree model?

- ① Start presenting to display the poll results on this slide.

Summary

What are two key advantages of decision tree models over logistic regression, especially in situations where classes are well-separated but boundaries are complex?

Decision trees can handle complex decision boundaries that linear models like logistic regression struggle with. They are also inherently interpretable, making it easy to understand the reasoning behind predictions.

How do predictions are made using a decision tree model?

To make a prediction, we traverse the tree by following the path determined by the feature values of the input data point. We reach a leaf node, which provides the predicted class label.

Why is it desirable to have progressively purer regions as we move down the tree?

Purity refers to the homogeneity of class labels within a region of the feature space. Purer regions are dominated by a single class. Increasing purity as we descend the tree means each region becomes more specialized towards a specific class, leading to better classification.

Summary

How does the Gini Index is used to assess the quality of a split in a decision tree. What is the impact of squaring the proportions of classes in its calculation?

The Gini Index measures impurity within a region. A lower Gini Index indicates greater purity. Squaring the class proportions amplifies the contribution of dominant classes, making the index more sensitive to imbalances and emphasizing splits that create purer regions.

What is the primary motivation behind using entropy as a splitting criterion in decision trees? How does it relate to the concept of information gain?

Entropy quantifies the uncertainty or randomness in a region's class distribution. By minimizing entropy, we aim to create splits that maximize information gain, meaning the resulting regions have lower uncertainty about class labels.

Decision Trees – Stopping Conditions

CS109A Introduction to Data Science

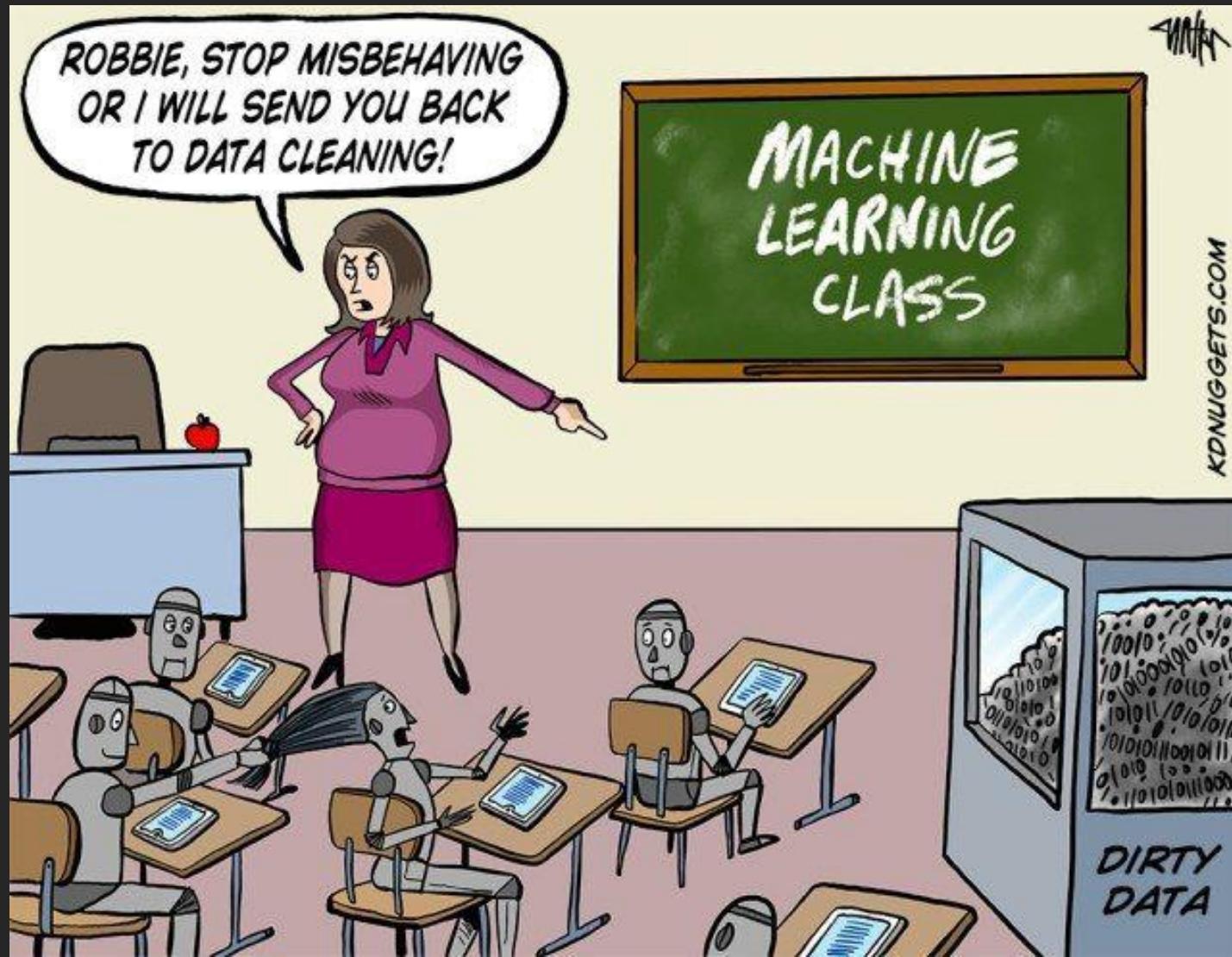
Pavlos Protopapas, Natesh Pillai, and Chris Gumb



Robert Wood
Troost Lake in midtown Kansas City

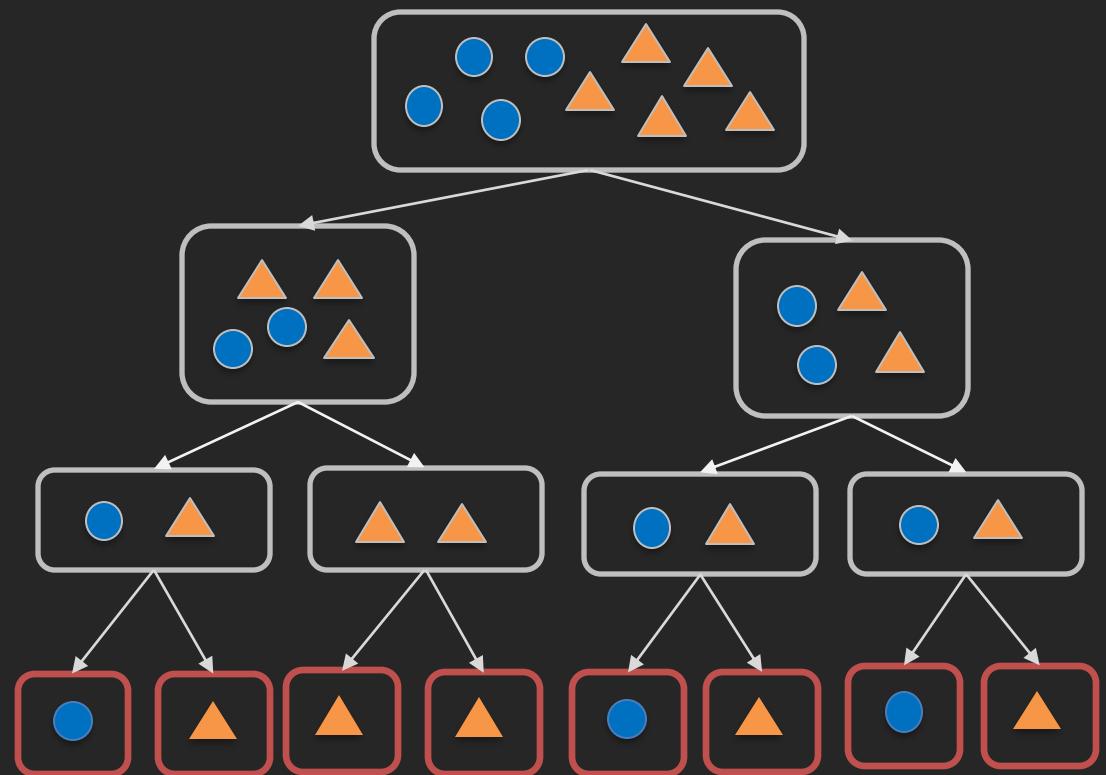
Outline

- Motivation
- Decision Trees – Classification
 - Intuition
 - Predictions
 - Splitting Criteria
 - Stopping Conditions



Stopping Conditions

Question: If we don't terminate the decision tree algorithm manually, what will the leaf nodes of the decision tree look like?



The tree will continue to grow until each region contains **exactly one training point** and the model attains 100% **training accuracy**.

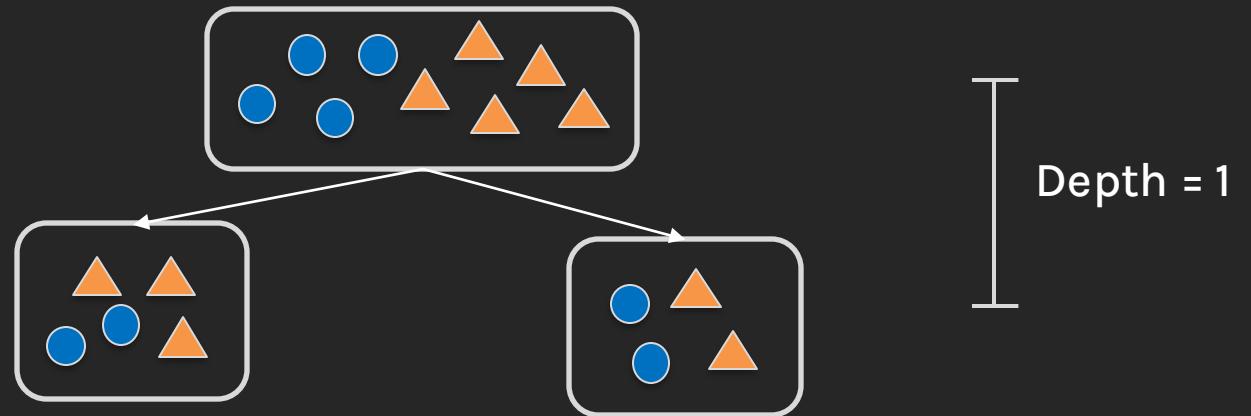
Stopping Conditions

Question: How can we prevent this from happening?

Stopping Conditions

The most common stopping condition is to limit the maximum depth (*max_depth*) of the tree.

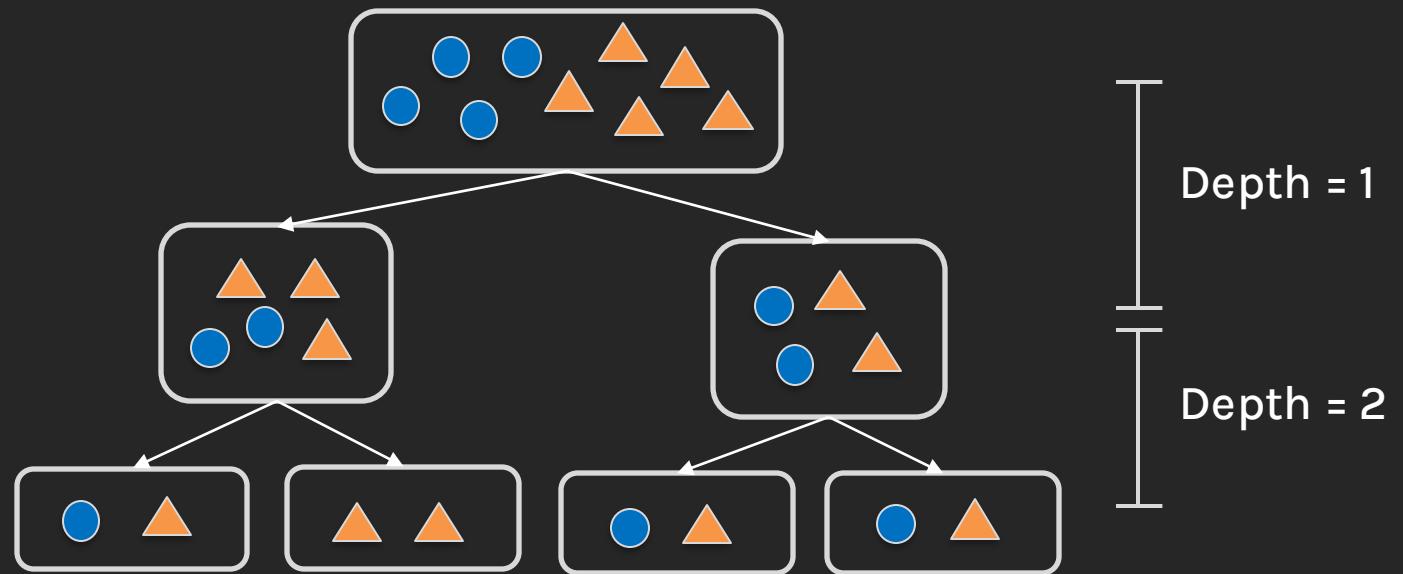
max_depth = 1



Stopping Conditions

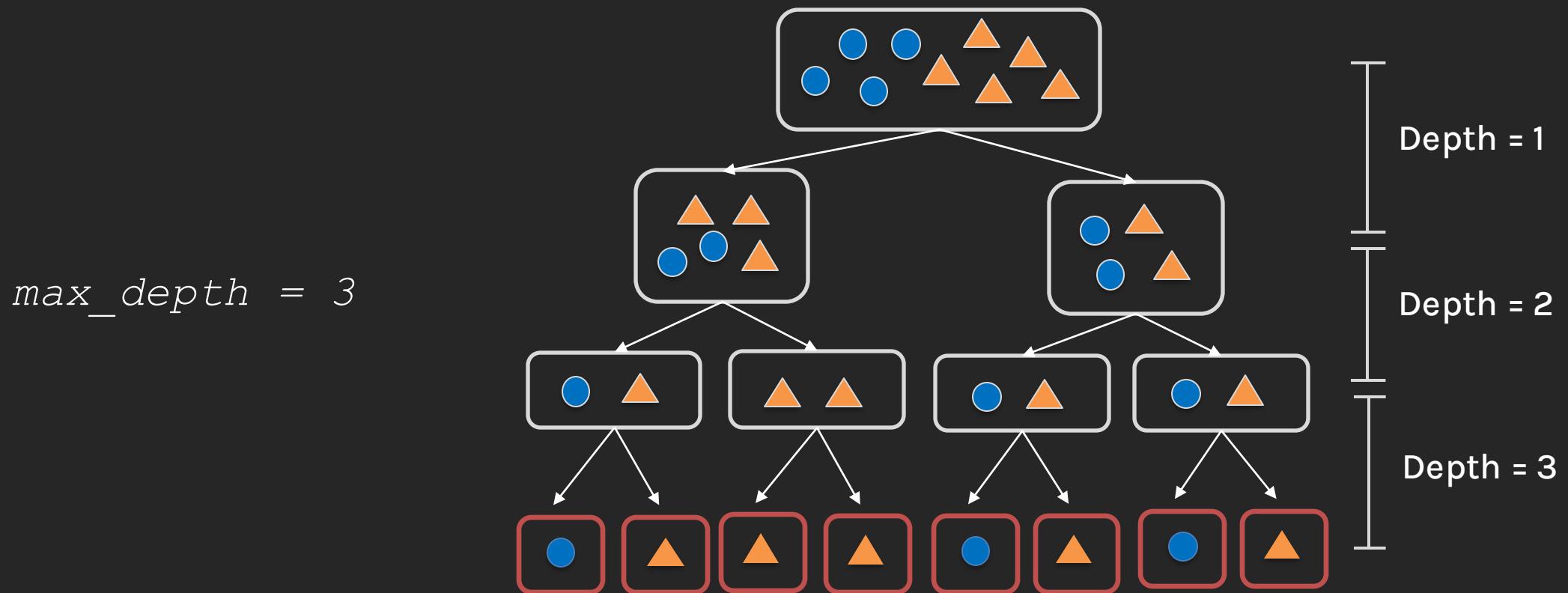
The most common stopping condition is to limit the maximum depth (*max_depth*) of the tree.

max_depth = 2



Stopping Conditions

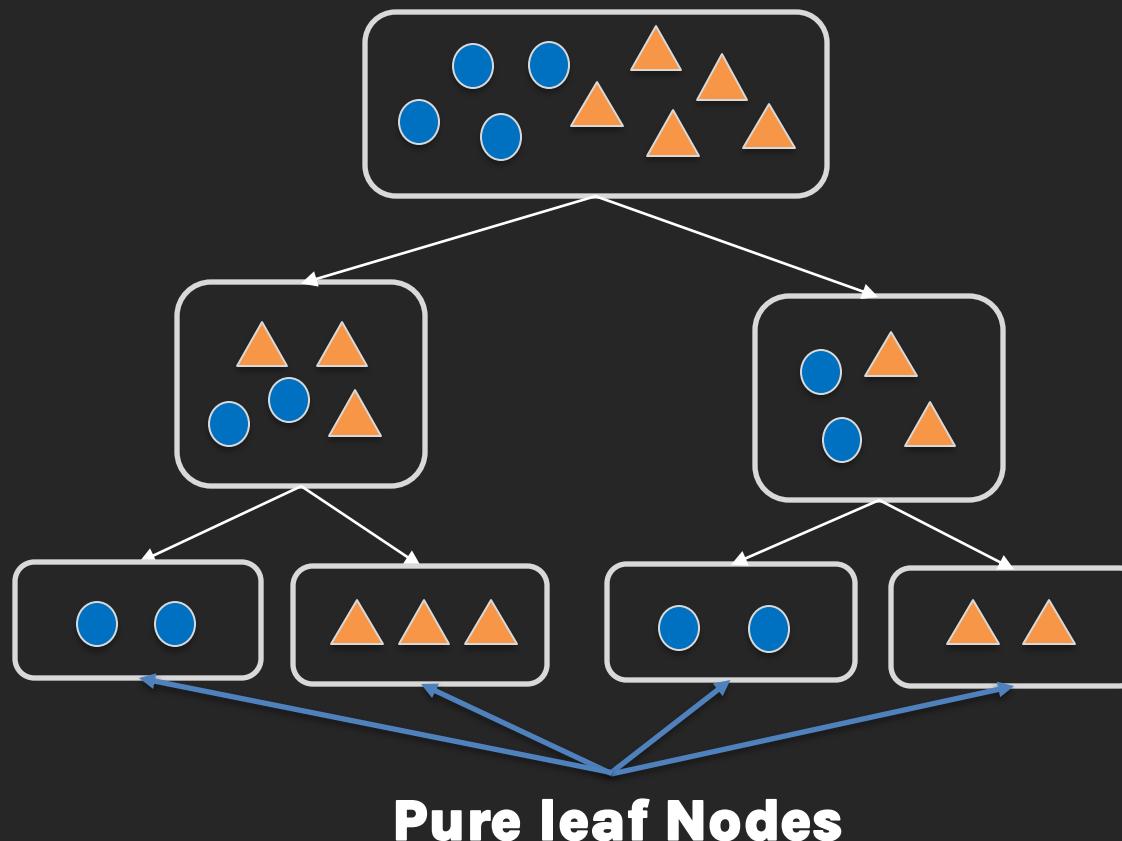
The most common stopping condition is to limit the **maximum depth** (*max_depth*) of the tree.



Stopping Conditions

Other common simple stopping conditions are:

- Don't split a region if all instances in the region belong to the same class.

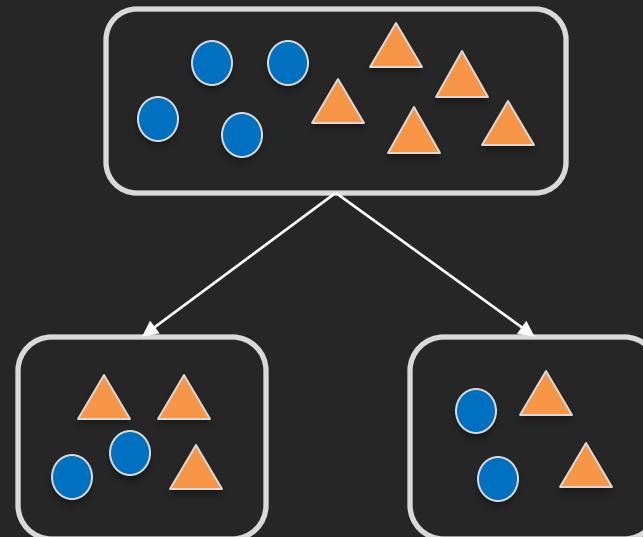


Stopping Conditions

Other common simple stopping conditions are:

- Don't split a region if the number of instances in any of the sub-regions will fall below pre-defined threshold ($min_samples_leaf$).

$$min_samples_leaf = 4$$



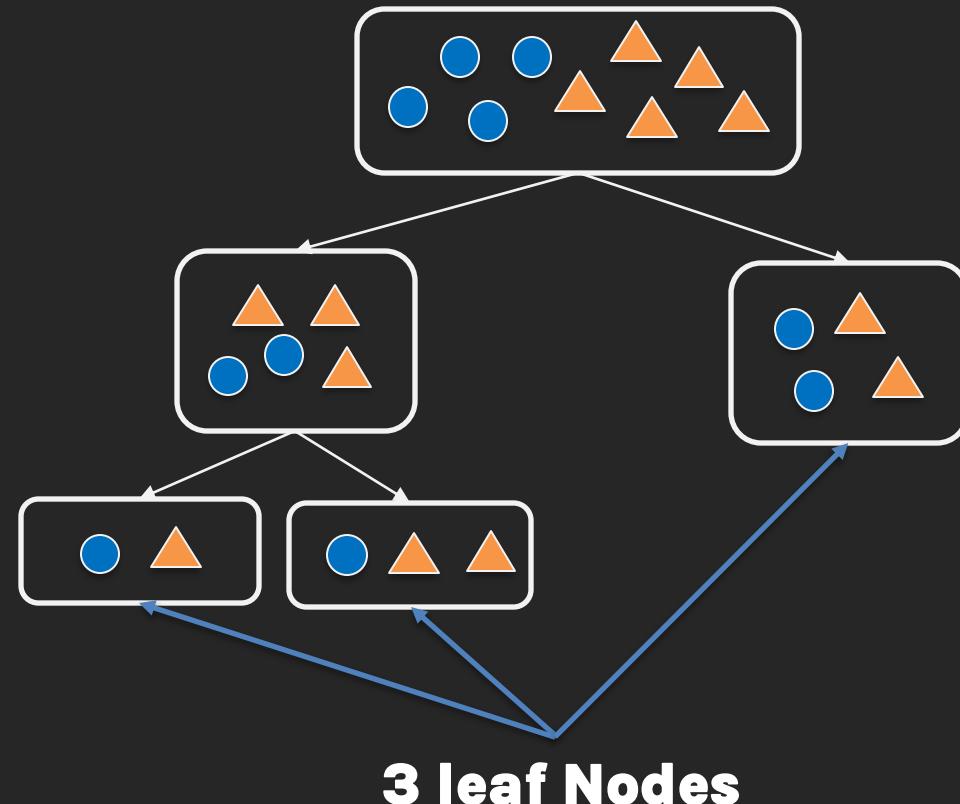
Stopping Conditions

Other common simple stopping conditions are:

- Don't split a region if the total number of leaves in the tree will exceed a pre-defined threshold (`max_leaf_nodes`).

`max_leaf_nodes = 3`

Do you see any issue with this?



Stopping Conditions

Normally, Sklearn grows trees in what is called ‘**level-order**’-fashion until a stopping condition such as `max_depth` is met.

However, if a value for `max_leaf_nodes` is specified, Sklearn will instead grow the tree in a ‘**best-first**’ fashion.

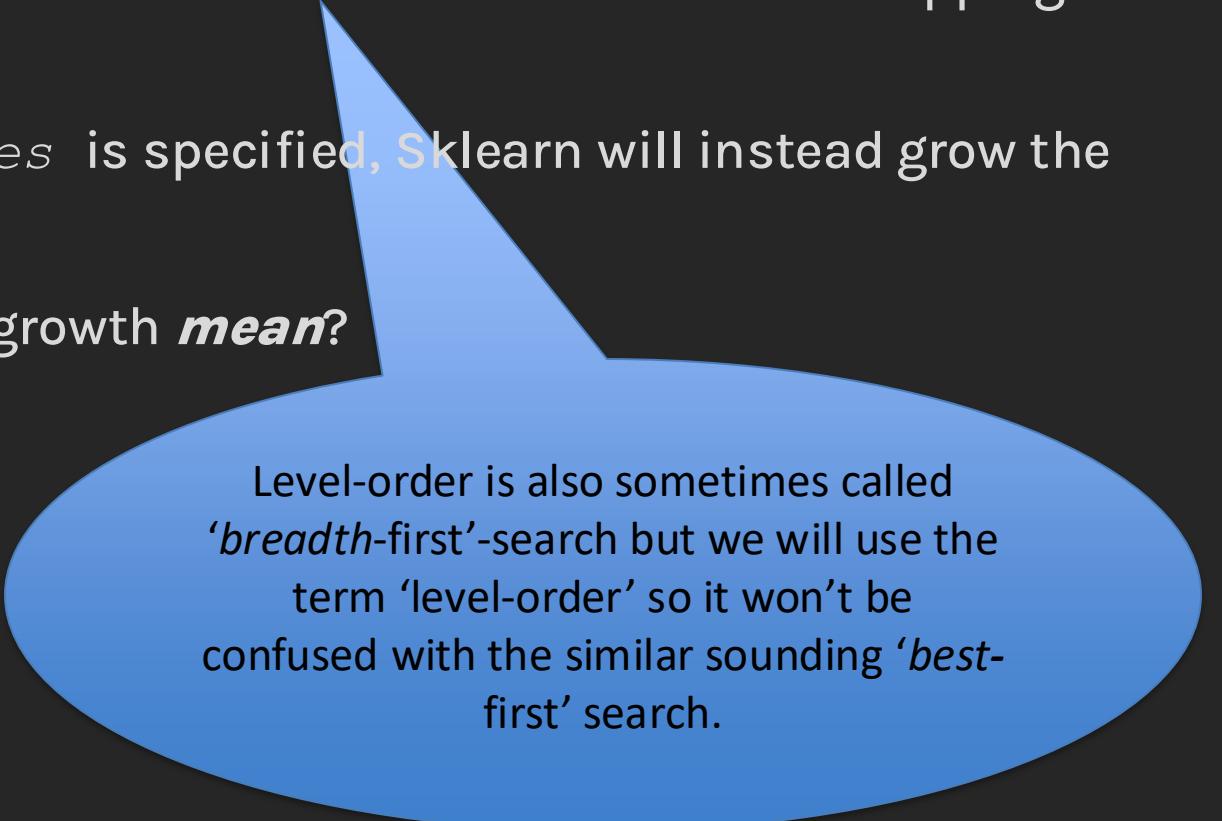
But what do **level-order** and **best-first** growth **mean**?

Stopping Conditions

Normally, Sklearn grows trees in what is called ‘**level-order**’-fashion until a stopping condition such as `max_depth` is met.

However, if a value for `max_leaf_nodes` is specified, Sklearn will instead grow the tree in a ‘**best-first**’ fashion.

But what do **level-order** and **best-first** growth **mean**?



Level-order is also sometimes called ‘*breadth-first*’-search but we will use the term ‘**level-order**’ so it won’t be confused with the similar sounding ‘**best-first**’ search.

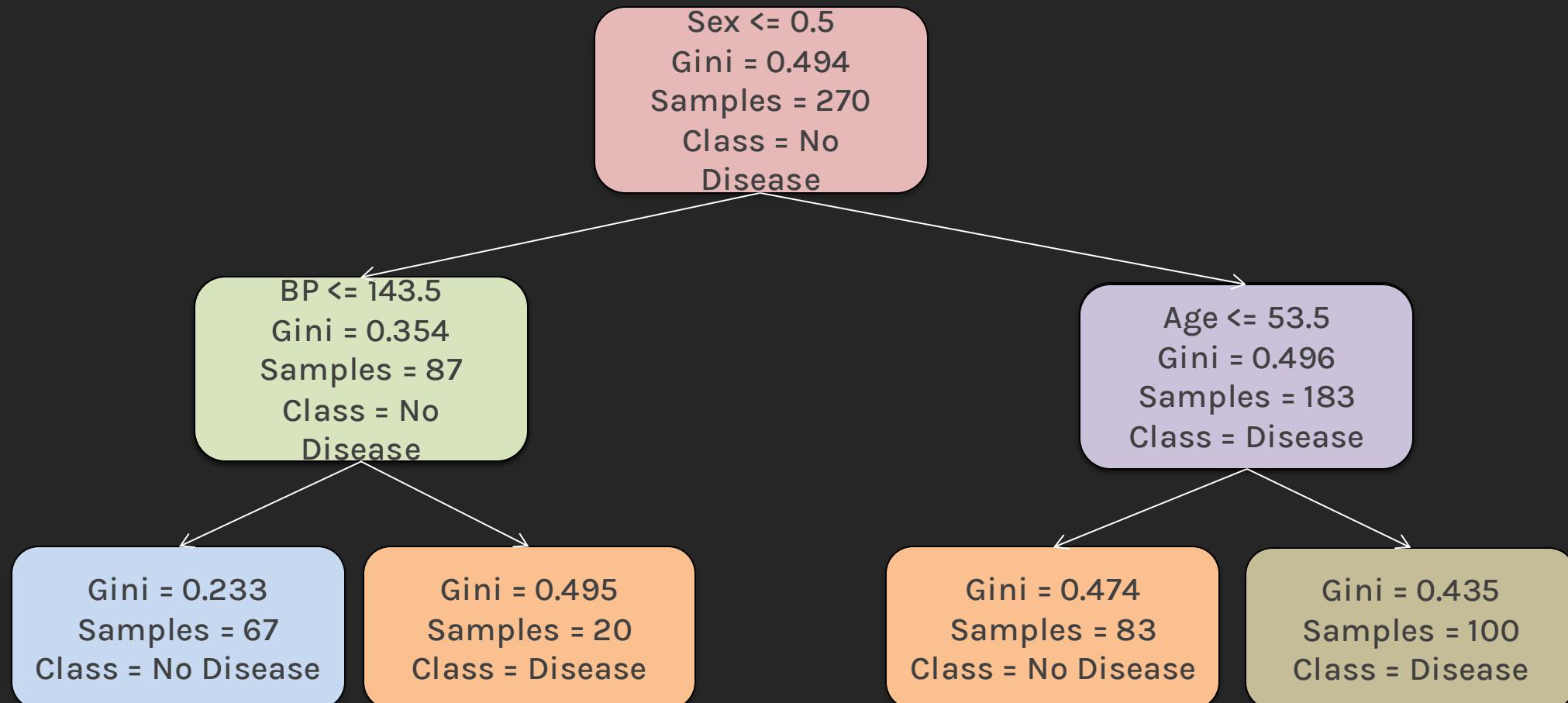
Example 1: Level-Order

Consider the following decision tree with `max_depth=2` that predicts if a person has heart disease based on age, sex, BP and cholesterol:

Gini = 0.494
Samples = 270
Class = No
Disease

Example 1: Level-Order

Consider the following decision tree with `max_depth=2` that predicts if a person has heart disease based on age, sex, BP and cholesterol:



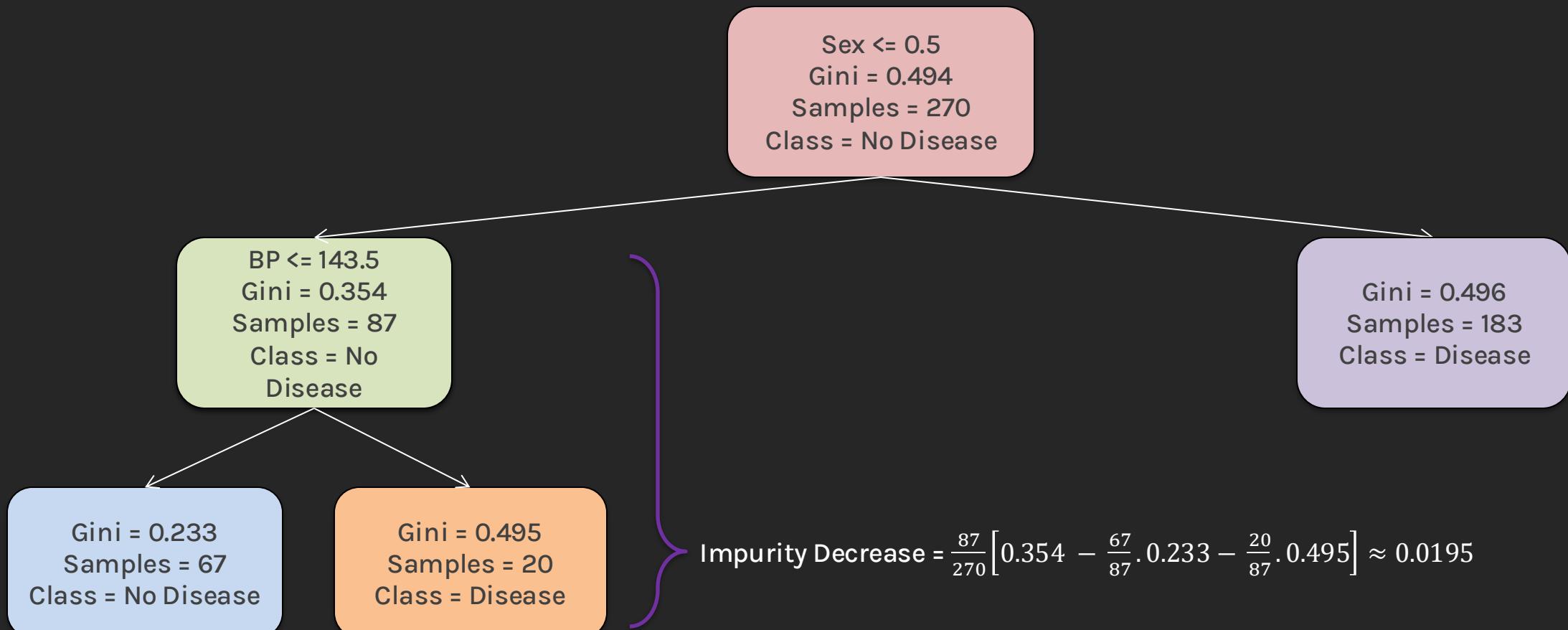
Example 1: Best-first growth

Sklearn determines the best split based on **impurity decrease**. The resulting tree will be the same when fully grown, just the order in which it is built is different.

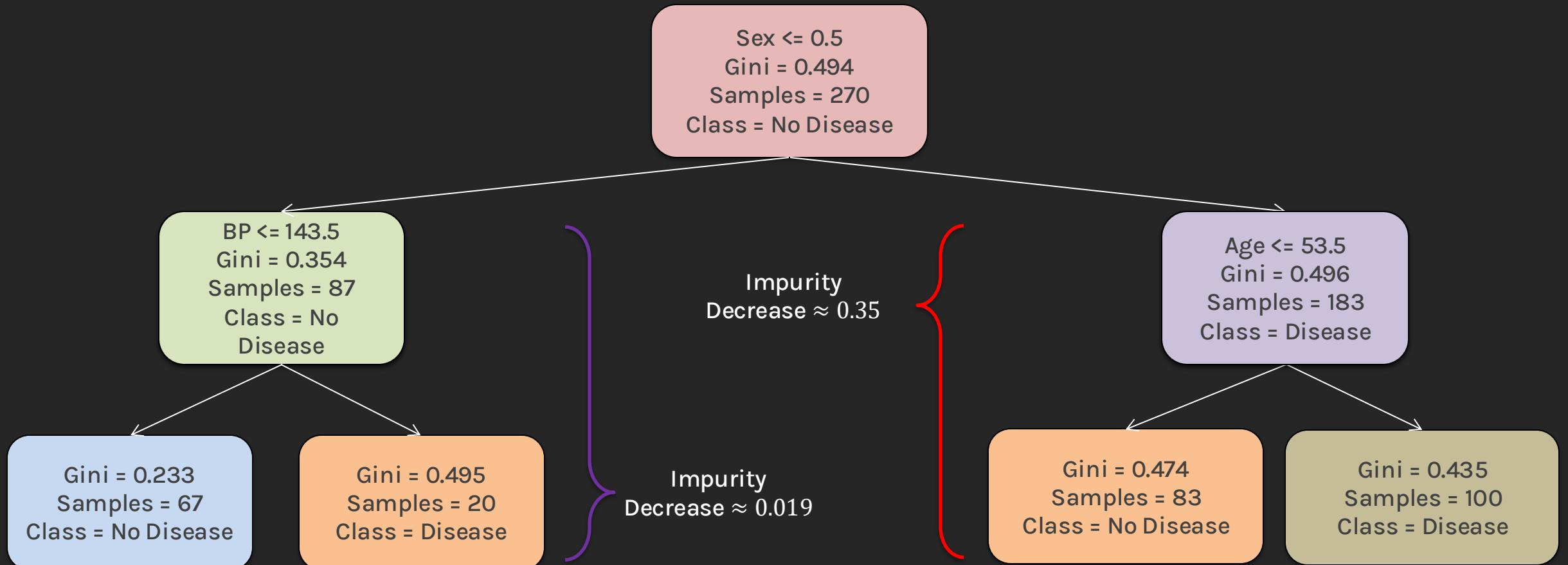
Gini = 0.494
Samples = 270
Class = No Disease

Example 1: Best-first growth

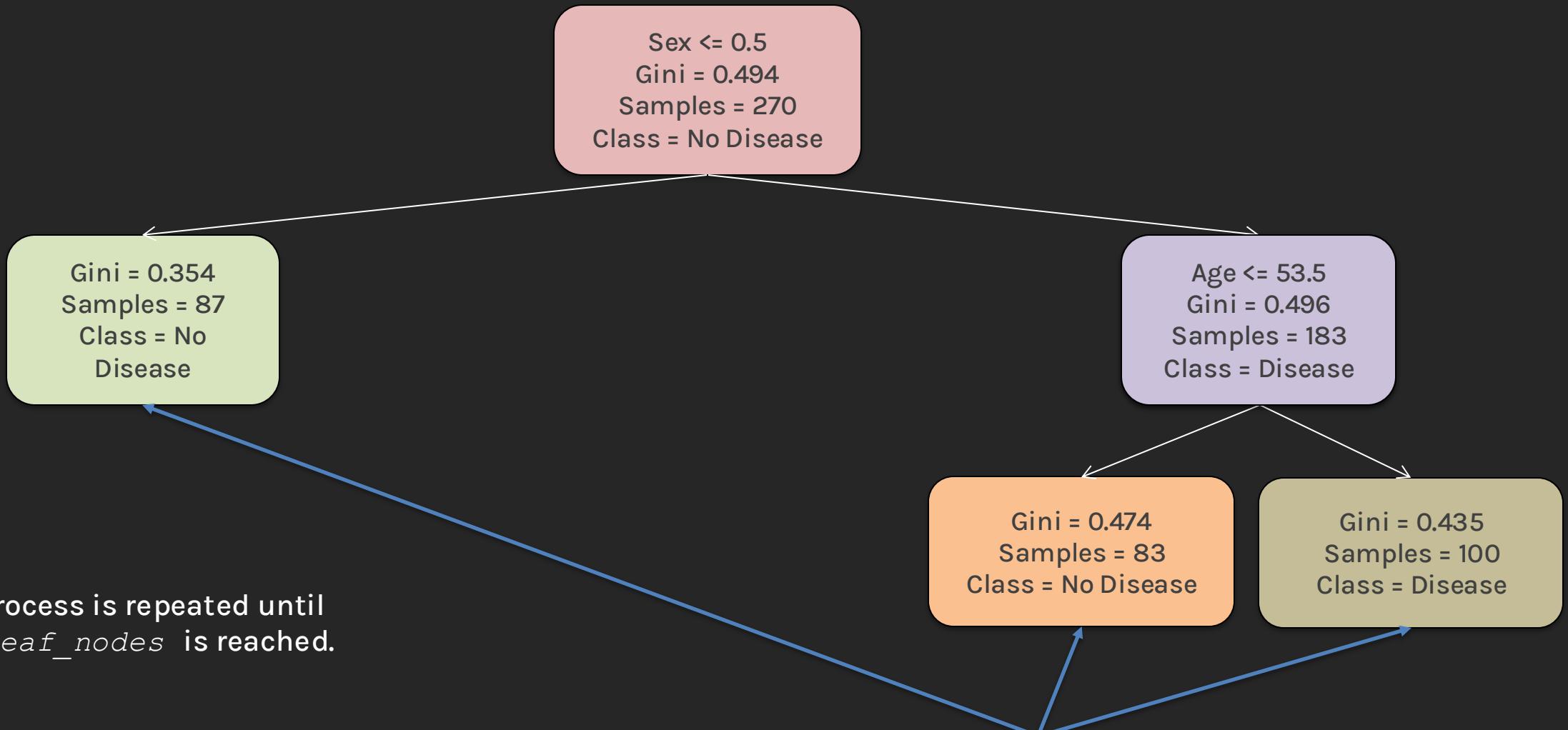
Sklearn determines the best split based on **impurity decrease**. The resulting tree will be the same when fully grown, just the order in which it is built is different.



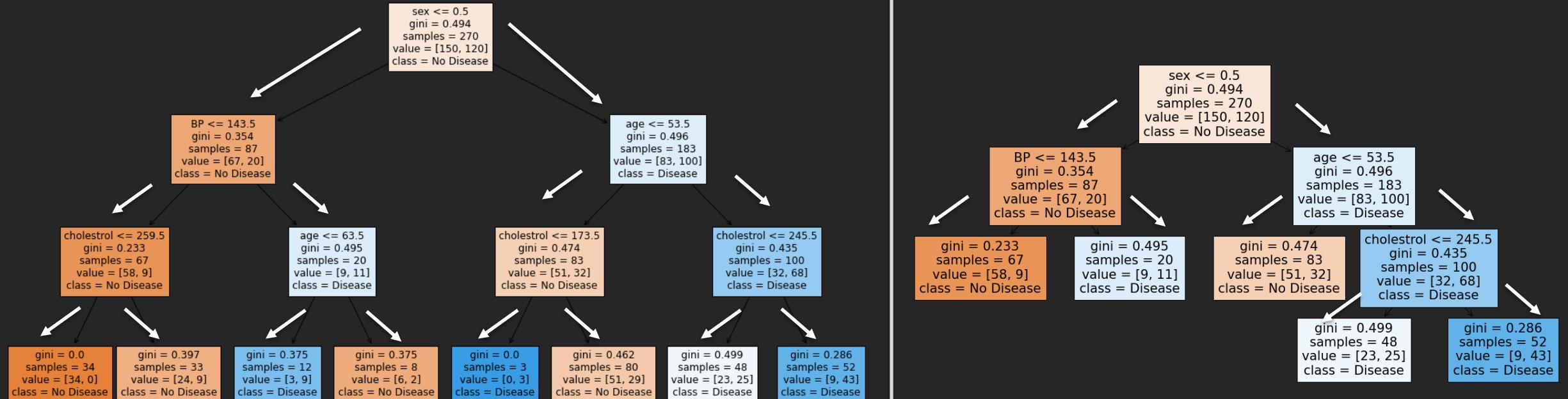
Example 1: Best-first growth



Example 1: Best-first growth



Example 2: Level-order vs Best-first growth



`max_depth = 3`

`max_leaf_nodes = 5`

Stopping Conditions

A more restrictive stopping condition is:

Compute the *gain* in purity of splitting a region R into R_1 and R_2 :

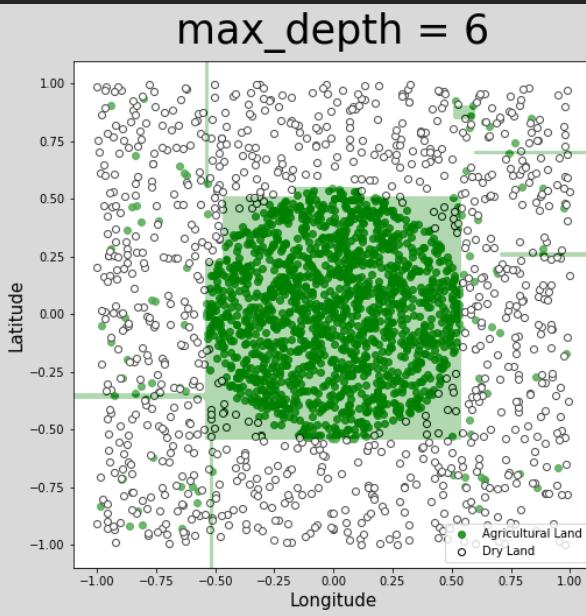
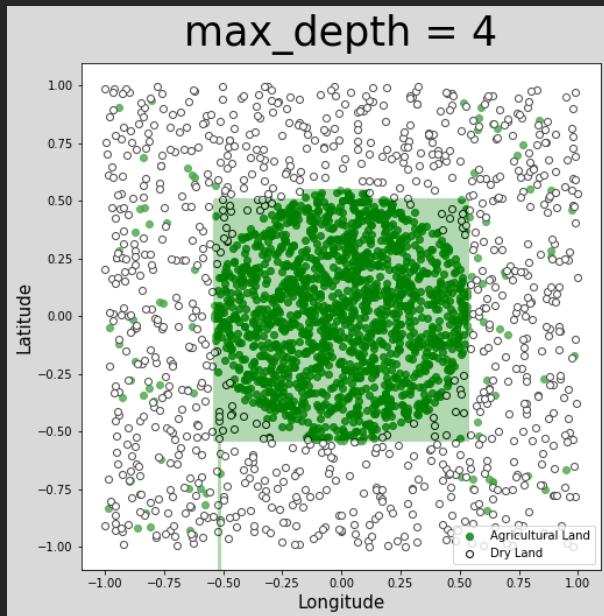
$$Gain(R) = \Delta(R) = m(R) - \frac{N_1}{N}m(R_1) - \frac{N_2}{N}m(R_2)$$

\uparrow
Classification Error/Gini Index/Entropy

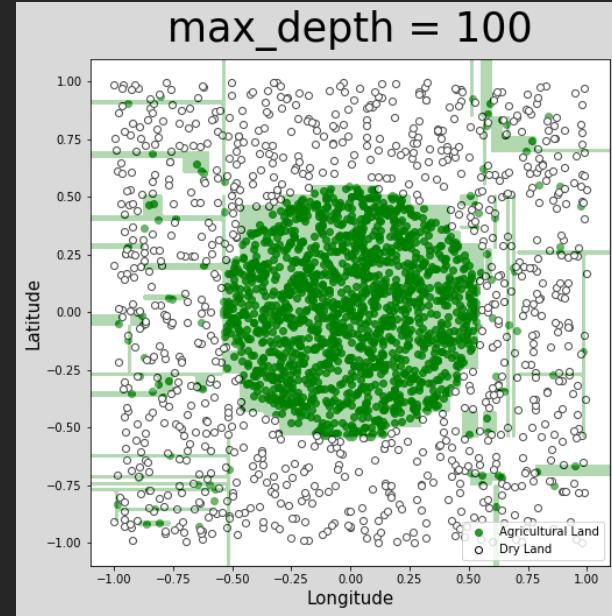
Don't split if the gain is less than some pre-defined threshold (`min_impurity_decrease`).

How do we decide what is the appropriate stopping condition or stopping method?

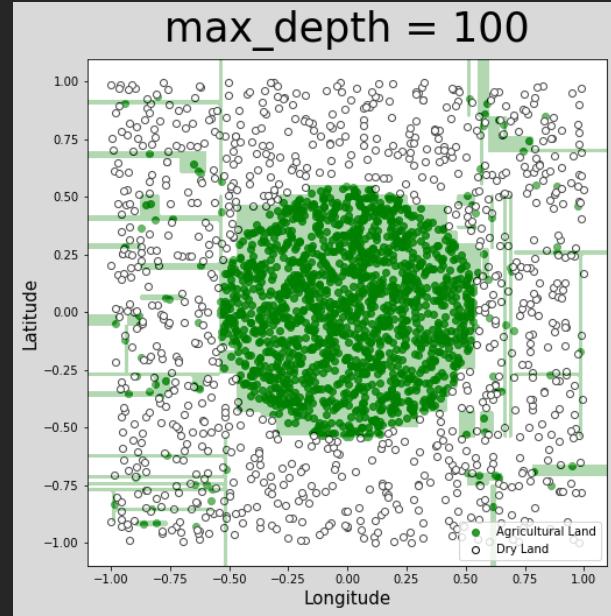
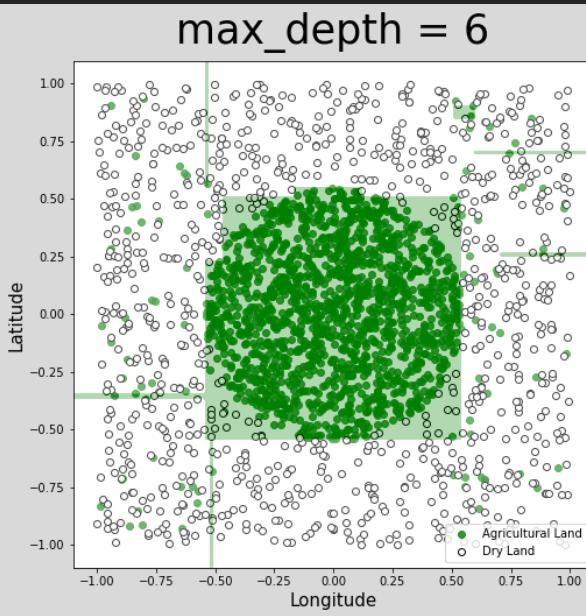
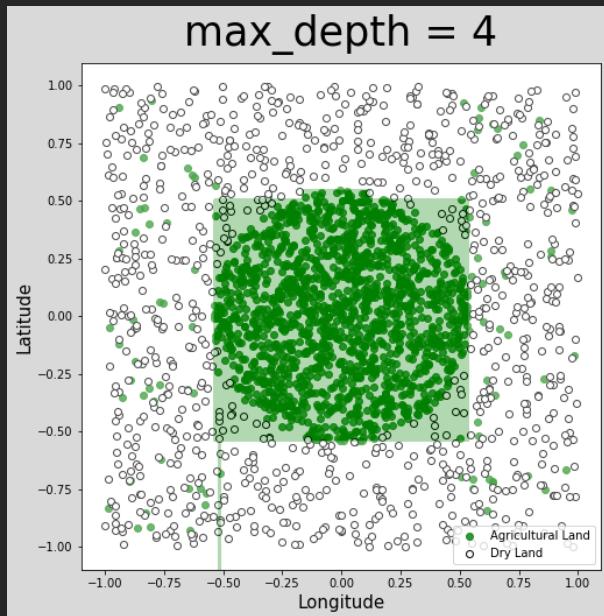
Variance vs Bias



• • • •



Variance vs Bias

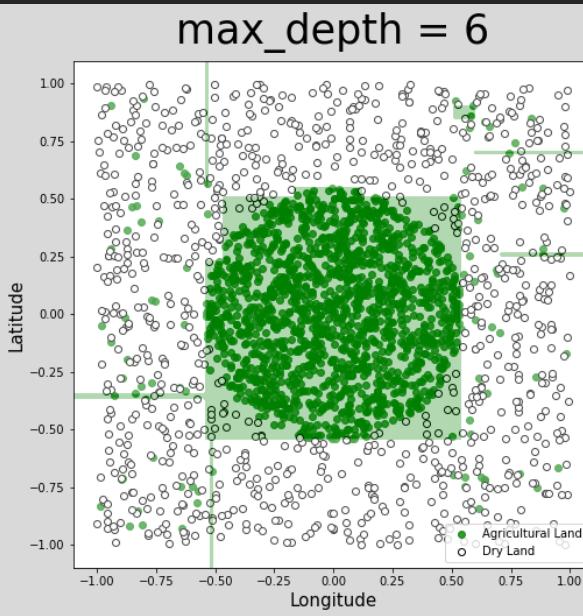
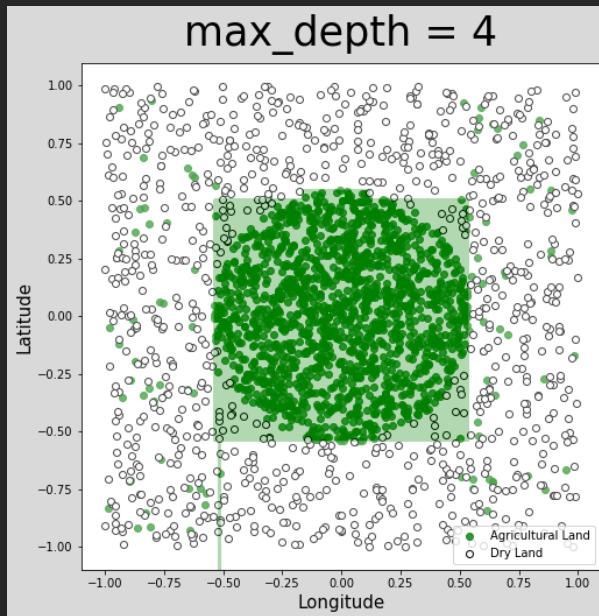


• • • •

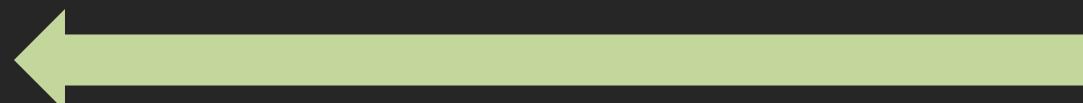
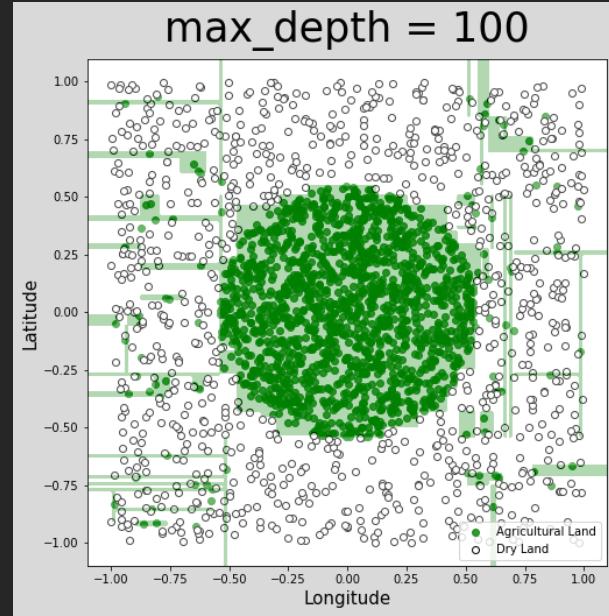


Bias decreases (can overfit)

Variance vs Bias

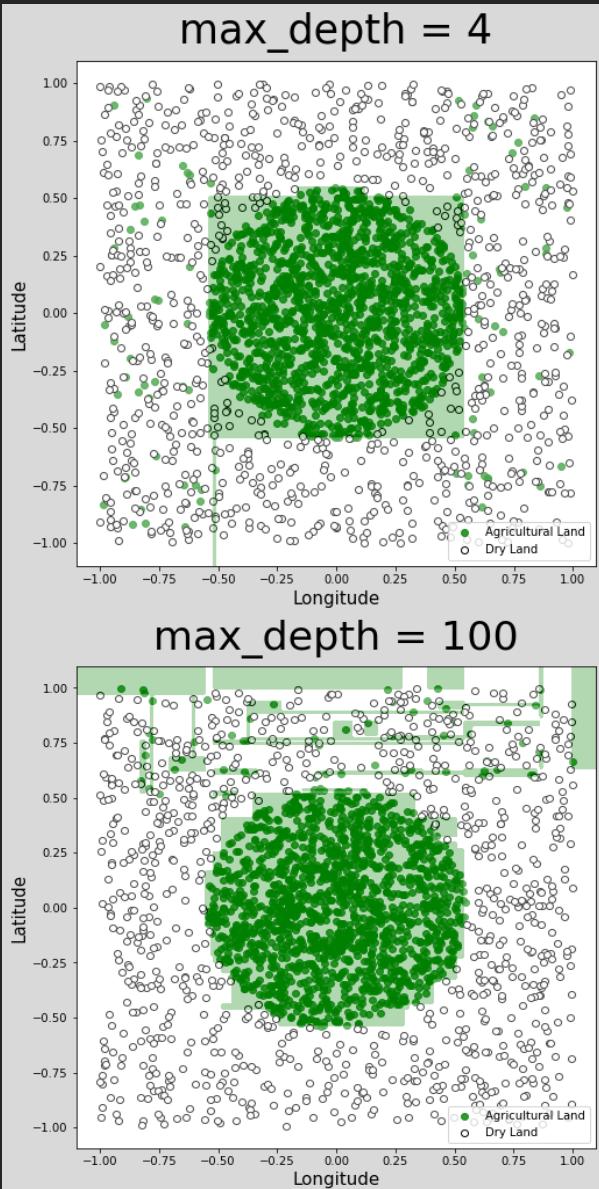


• • • •



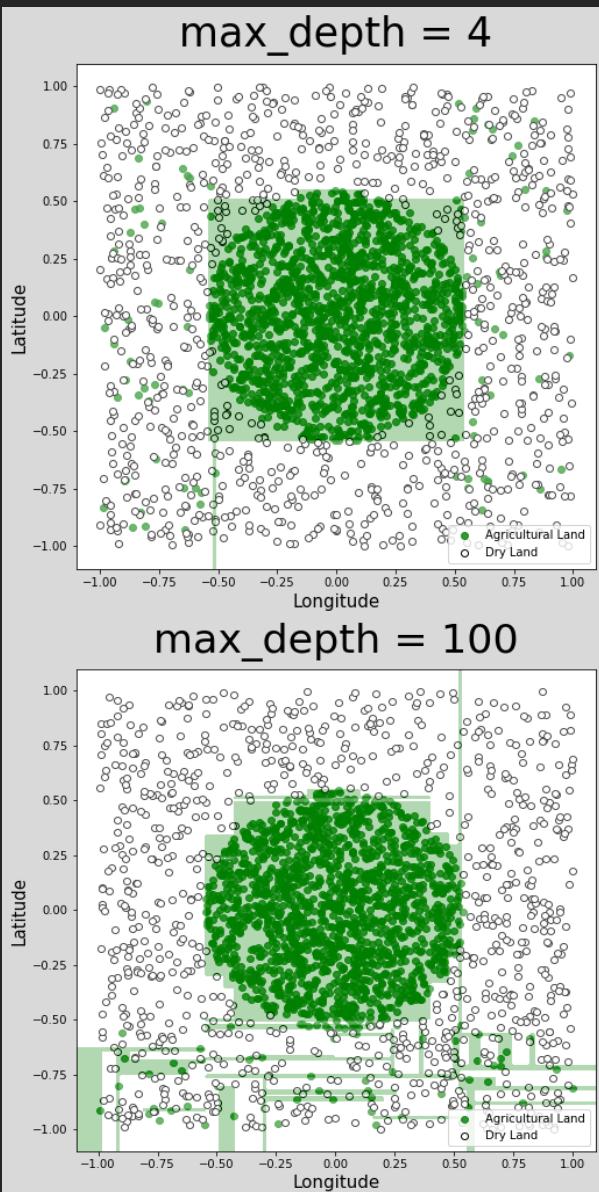
Complex trees are also harder to interpret and more computationally expensive to train.

Variance vs Bias



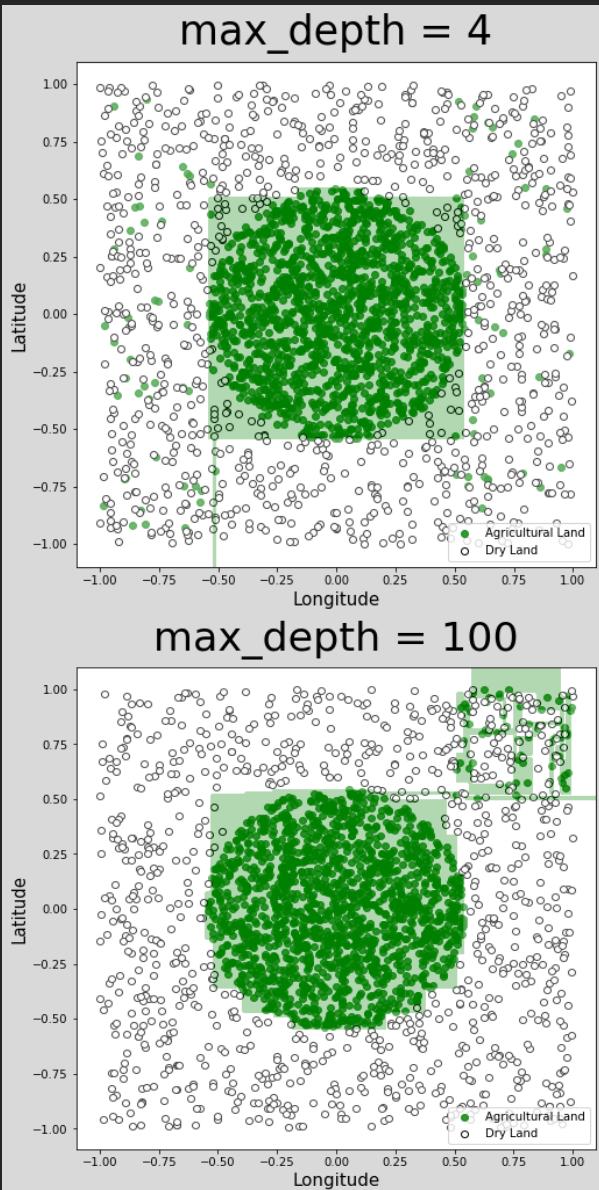
- **High Bias:** Trees of low depth are not a good fit for the training data - it's unable to capture the nonlinear boundary separating the two classes.
- **Low Variance:** Trees of low depth are robust to slight perturbations in the training data - the square carved out by the model is stable if you move the boundary points a bit.
- **Low Bias:** With a high depth, we can obtain a model that correctly classifies all points on the boundary (by zig-zagging around each point).

Variance vs Bias



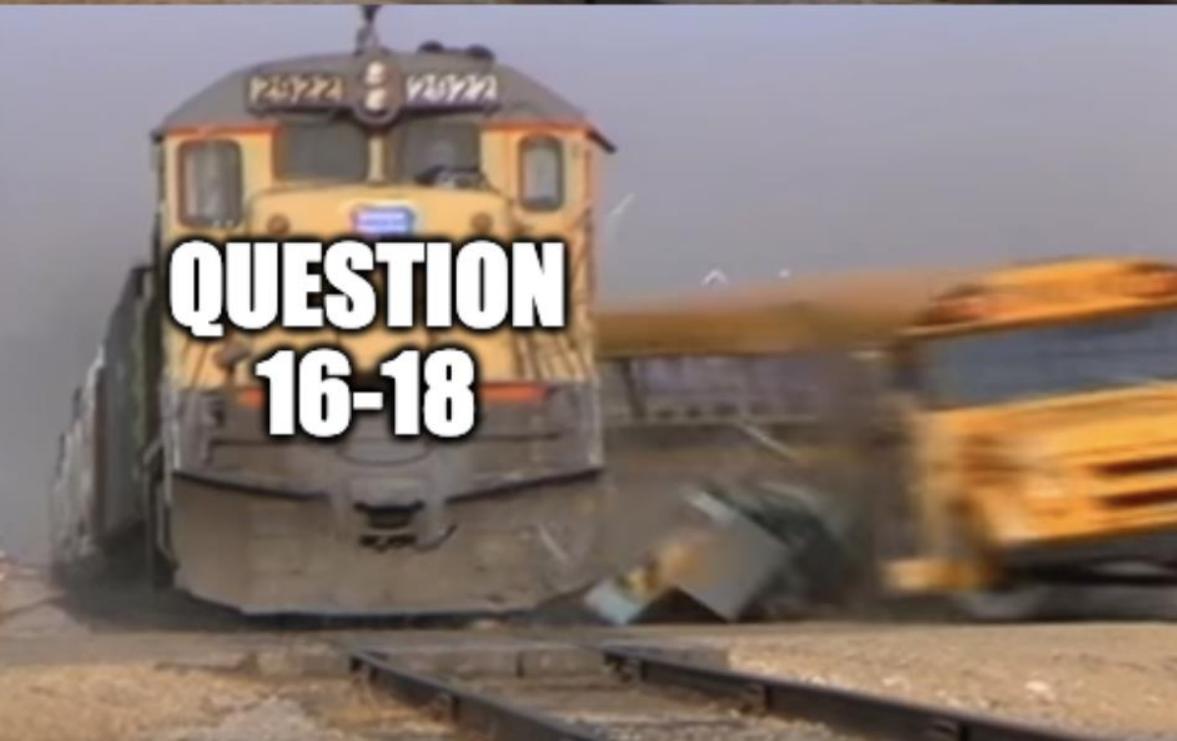
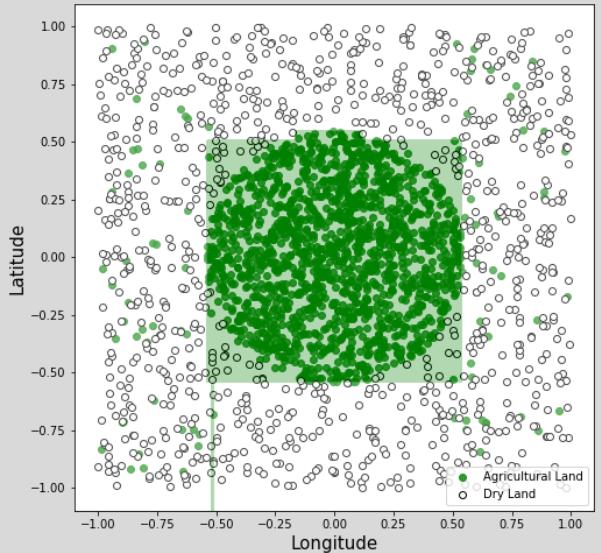
- **High Bias:** Trees of low depth are not a good fit for the training data - it's unable to capture the nonlinear boundary separating the two classes.
- **Low Variance:** Trees of low depth are robust to slight perturbations in the training data - the square carved out by the model is stable if you move the boundary points a bit.
- **Low Bias:** With a high depth, we can obtain a model that correctly classifies all points on the boundary (by zig-zagging around each point).

Variance vs Bias

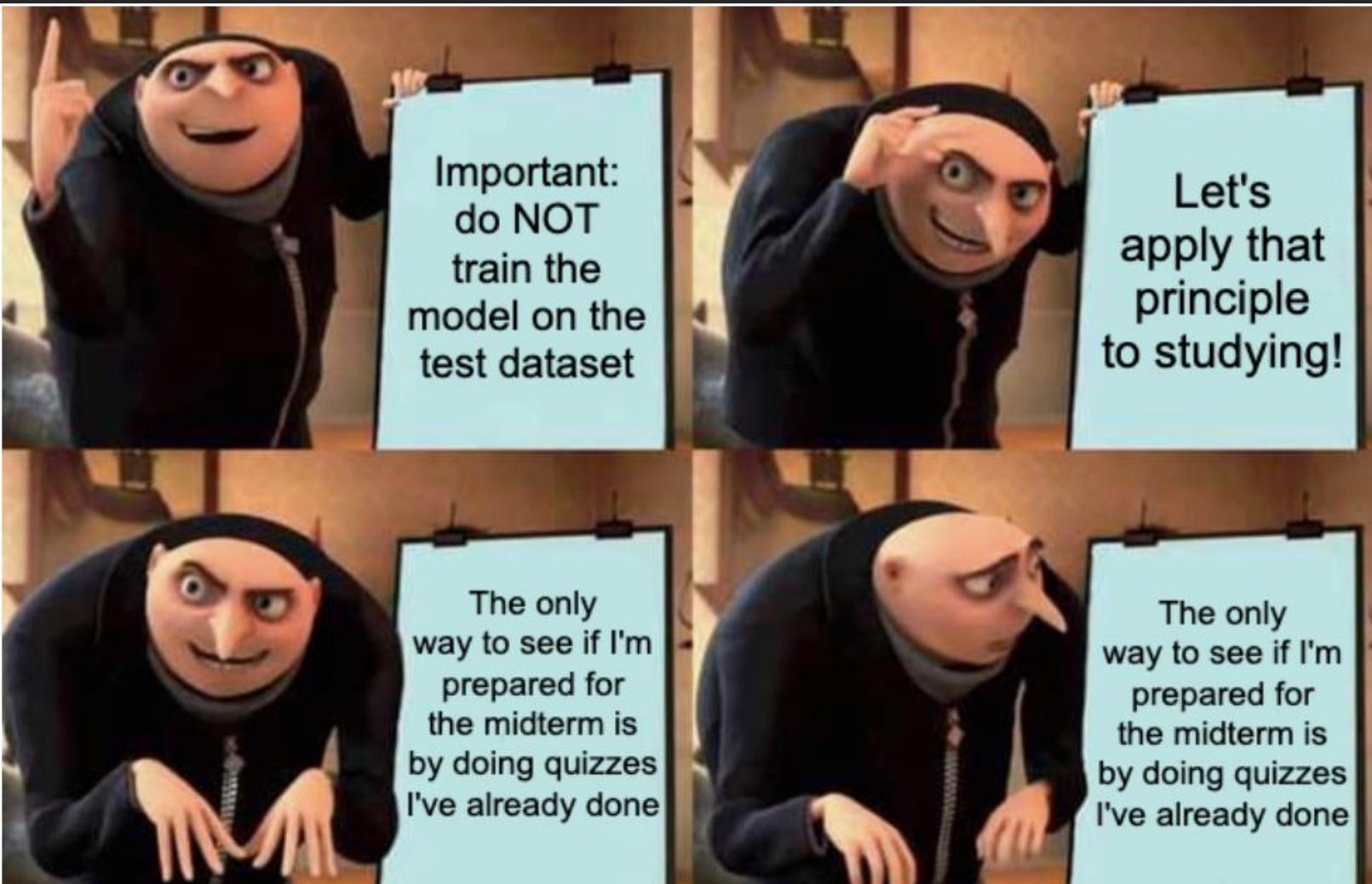
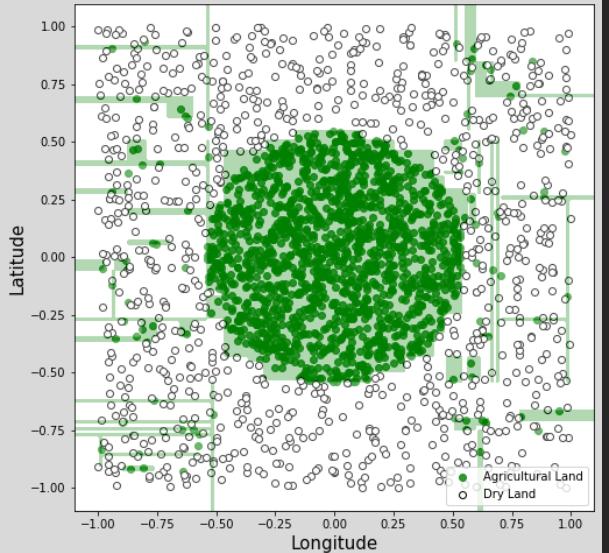


- **High Bias:** Trees of low depth are not a good fit for the training data - it's unable to capture the nonlinear boundary separating the two classes.
- **Low Variance:** Trees of low depth are robust to slight perturbations in the training data - the square carved out by the model is stable if you move the boundary points a bit.
- **Low Bias:** With a high depth, we can obtain a model that correctly classifies all points on the boundary (by zig-zagging around each point).
- **High Variance:** Trees of high depth are sensitive to perturbations in the training data, especially to changes in the boundary points.

max_depth = 4



max_depth = 100



Stopping Conditions

`max_depth`

`min_samples_leaf`

`max_leaf_nodes`

`min_impurity_decrease`

How can we determine the appropriate hyperparameters?

cross-validation

Please download and install the Slido app on all computers you use



Explain the concept of overfitting in decision trees. How can stopping conditions help prevent overfitting?

- ① Start presenting to display the poll results on this slide.

Please download and install the Slido app on all computers you use



Which of the following are common stopping conditions in decision tree learning, and how do they help limit tree growth?

- ① Start presenting to display the poll results on this slide.

Please download and install the Slido app on all computers you use



What is the difference between level-order and best-first growth in decision trees, and when might you prefer one method over the other?

- ① Start presenting to display the poll results on this slide.

Summary

Explain the concept of overfitting in decision trees. How can stopping conditions help prevent overfitting?

Overfitting occurs when the decision tree becomes excessively complex, memorizing the training data but failing to generalize to new data. Stopping conditions limit tree growth, preventing it from becoming too specific to the training data.

Describe two common stopping conditions used in decision tree learning and explain how they limit tree growth.

Two examples are `max_depth`, which limits the maximum depth of the tree, and `min_samples_leaf`, which sets a minimum number of samples required to form a leaf node. Both prevent the tree from growing too deep and overfitting.

Differentiate between level-order and best-first growth in decision trees. When would you prefer one method over the other?

Level-order growth builds the tree level by level, while best-first prioritizes splits with the highest impurity decrease. Best-first is advantageous when seeking the most informative splits early on, potentially leading to smaller trees with good performance.

Summary

Explain the trade-off between bias and variance in decision trees. How does tree depth impact bias and variance?

Shallow trees have high bias (simplistic model) and low variance (robust to data fluctuations). Deep trees have low bias (complex model) but high variance (sensitive to data changes). The optimal depth balances this trade-off.

What is the role of cross-validation in determining the optimal hyperparameters for a decision tree model?

Cross-validation helps estimate a model's performance on unseen data. By evaluating different hyperparameter combinations (e.g., tree depth, stopping criteria) using cross-validation, we can select those that lead to the best generalization performance.



Thank you



Decision Trees - Pruning



CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai and Chris Gumb

Seufert, Eric
St. John, USVI

**SO MY DAUGHTER SAYS
YOU WORK IN DATA SCIENCE**

**YES, TODAY I BUILT
A MODEL WITH R² OF 1**

**YOU HAVE EXACTLY 10
SECONDS TO GET OUT OF MY HOUSE**

Outline

- Decision Trees - Regression
- Numerical vs Categorical Attributes
- Pruning

Alternatives to Using Stopping Conditions

What is the major issue with pre-specifying a stopping condition?

- You may stop too early or stop too late.

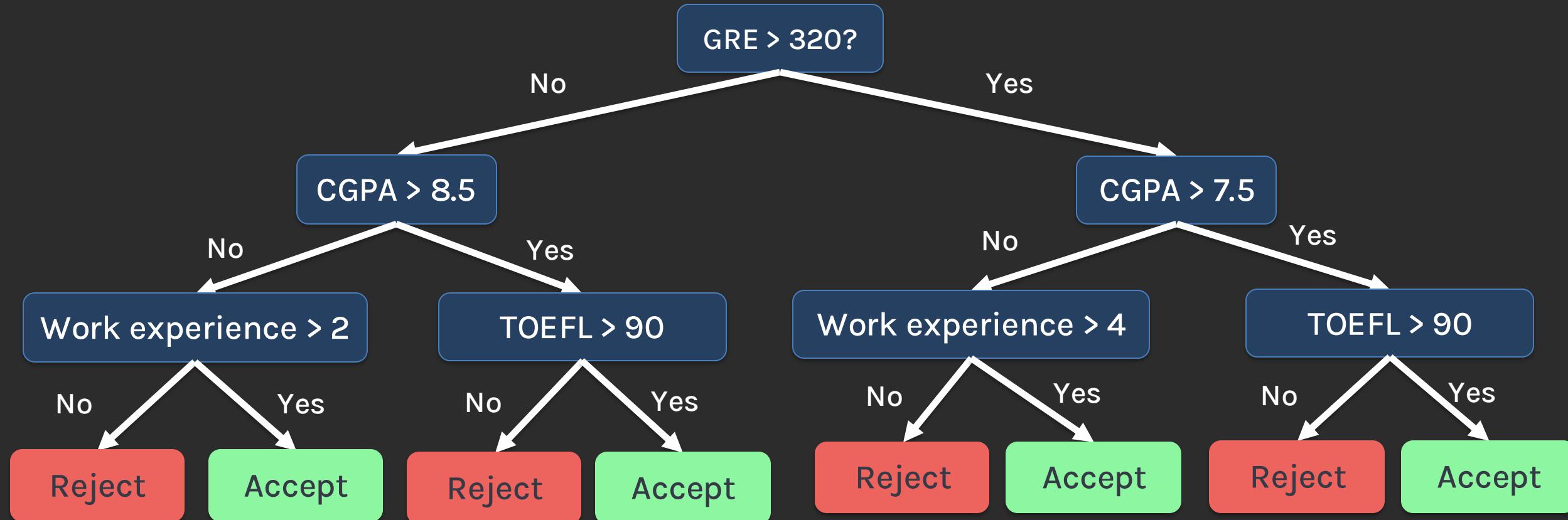
How can we fix this issue?

- Choose several stopping criteria (e.g., set minimal Gain(R) at various levels) and cross-validate to decide which one is the best.

What is an alternative approach to this method?

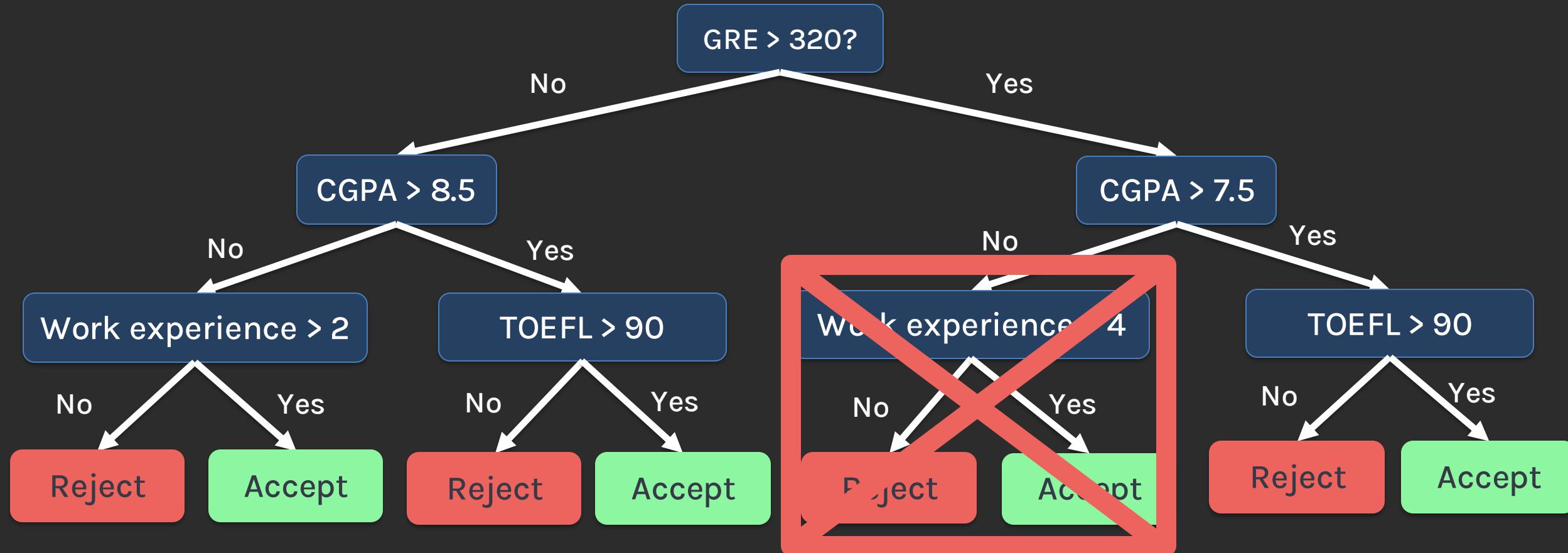
- Don't stop. Instead, prune the tree!

Example: Evaluating Applications to a GRADUATE Program



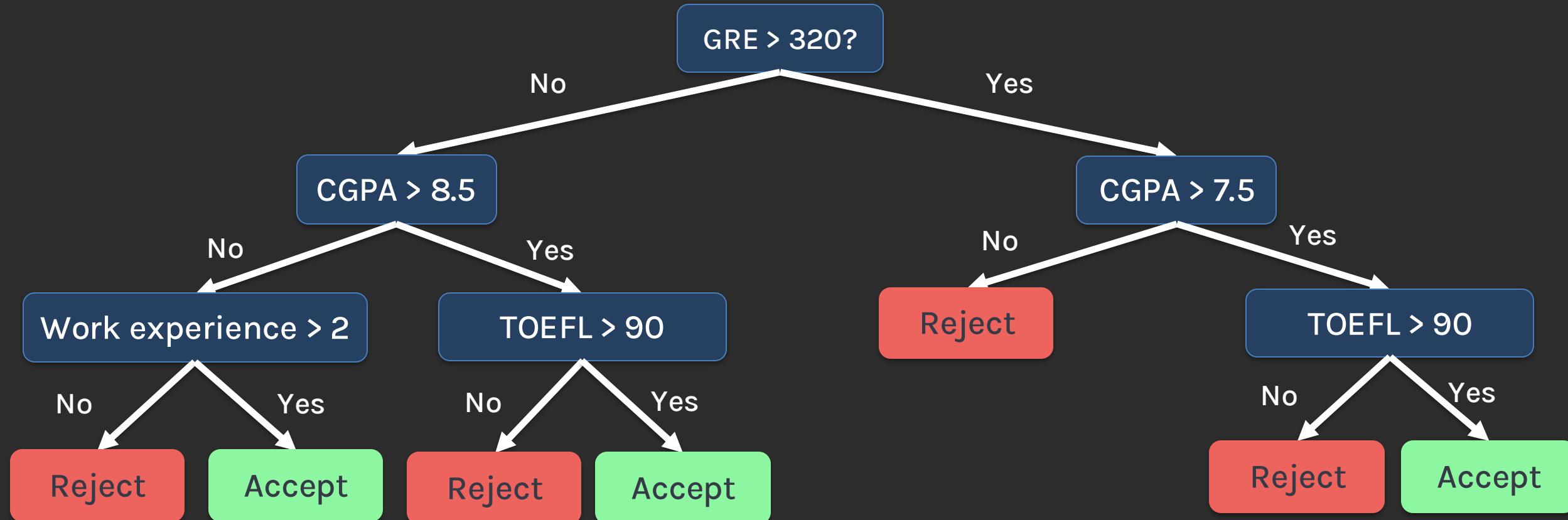
** Above representation is only for pedagogical purposes.

Example: Evaluating Applications to a GRADUATE Program



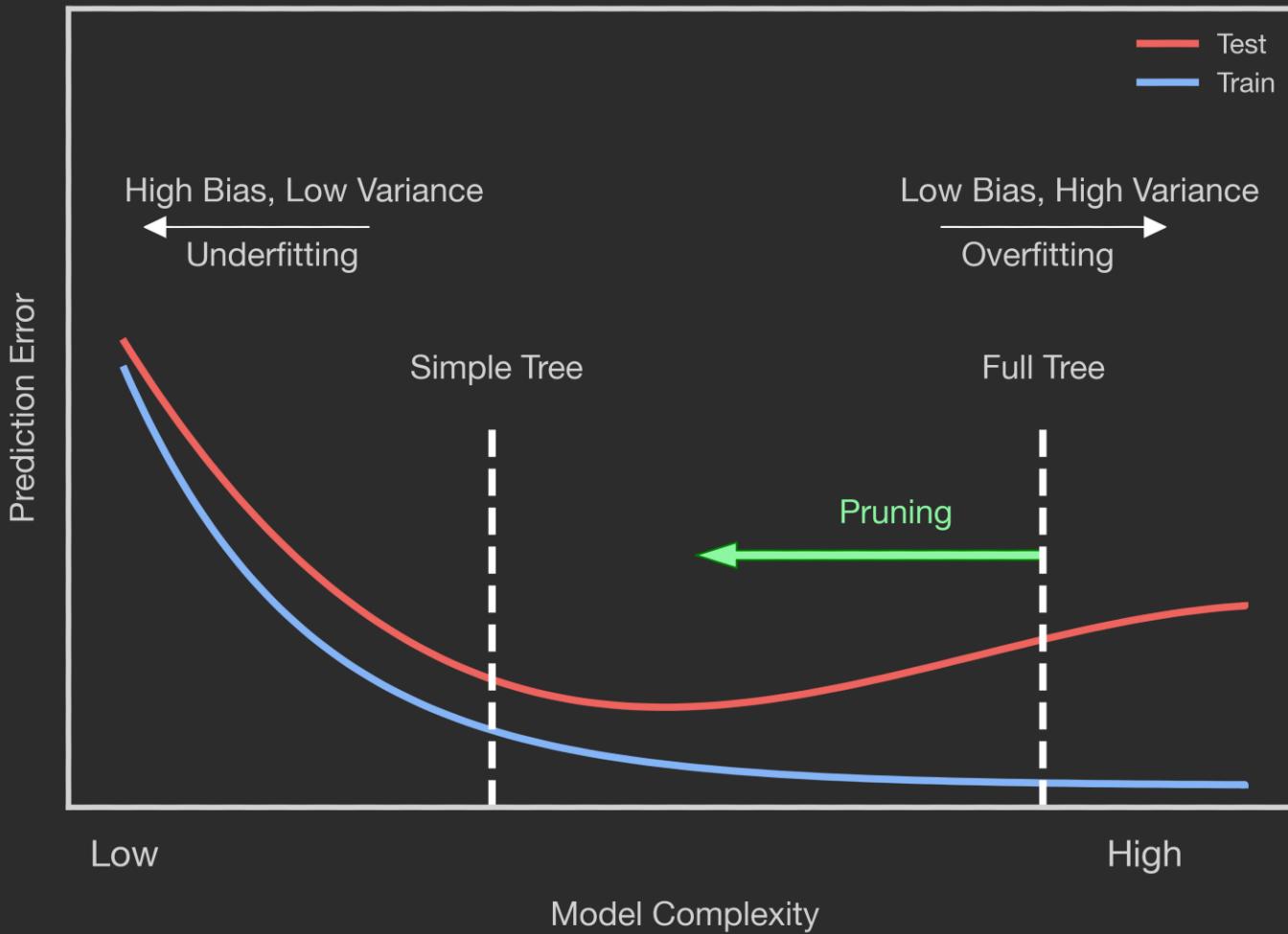
** Above representation is only for pedagogical purposes.

Example: Evaluating Applications to a GRADUATE Program



** Above representation is only for pedagogical purposes.

Motivation for Pruning



Rather than preventing a complex tree from growing, we can obtain a simpler tree by ‘pruning’ a complex one.

Pruning

There are many methods of pruning. A common one is the **cost complexity pruning**:

$$C(T) = \text{Error}(T) + \alpha |T|$$

↑
Decision tree

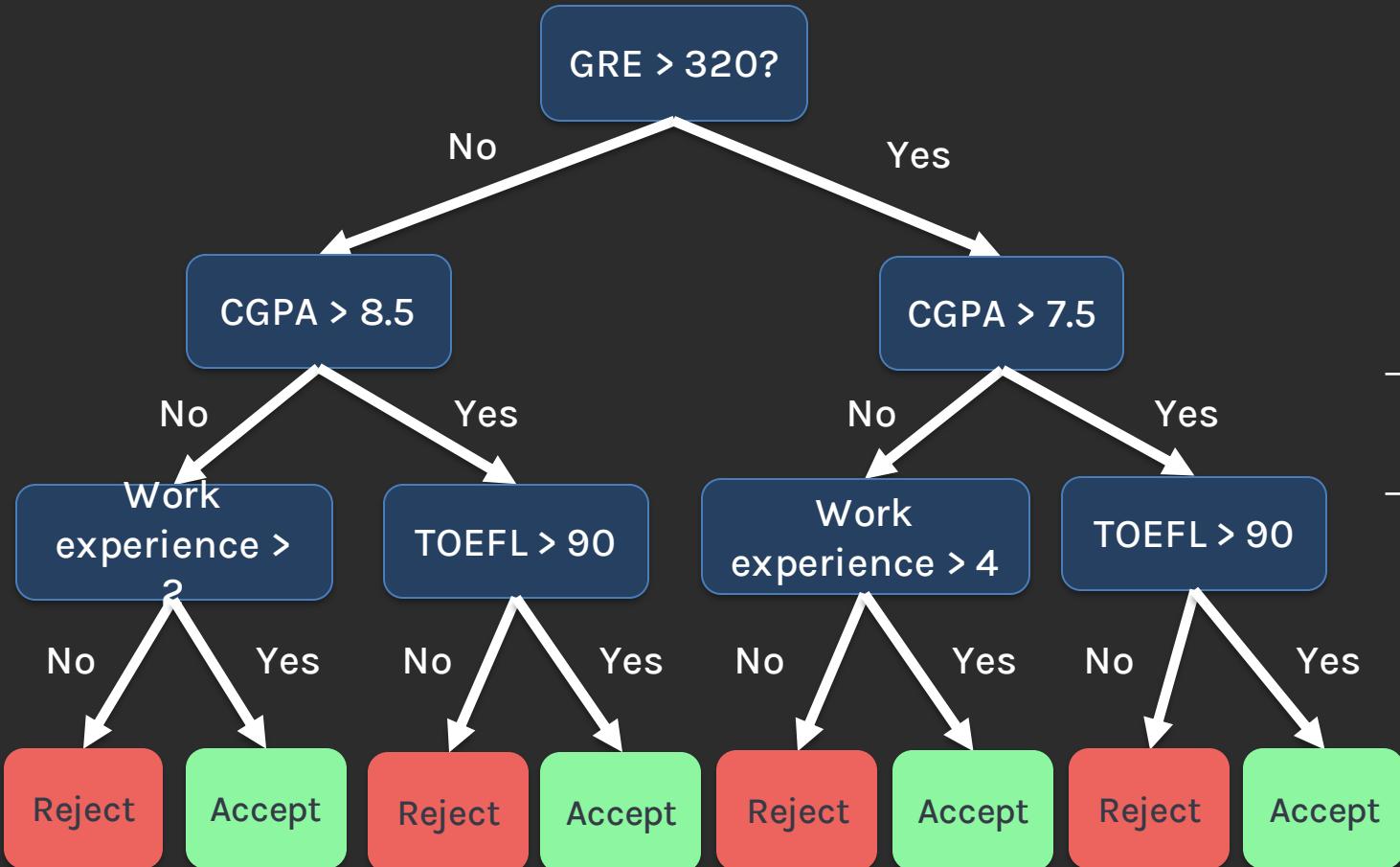
Classification Error
Complexity Parameter
Number of leaves in the tree

Regularization term

The diagram illustrates the cost function $C(T) = \text{Error}(T) + \alpha |T|$. It features three colored arrows pointing to different parts of the equation: an orange arrow to the $\text{Error}(T)$ term labeled "Classification Error", a purple arrow to the $\alpha |T|$ term labeled "Complexity Parameter", and a green arrow to the $|T|$ term labeled "Number of leaves in the tree". A blue bracket is placed over the $\alpha |T|$ term, which is also labeled "Regularization term" to its right.

In other words, we add a ‘regularization’ term!

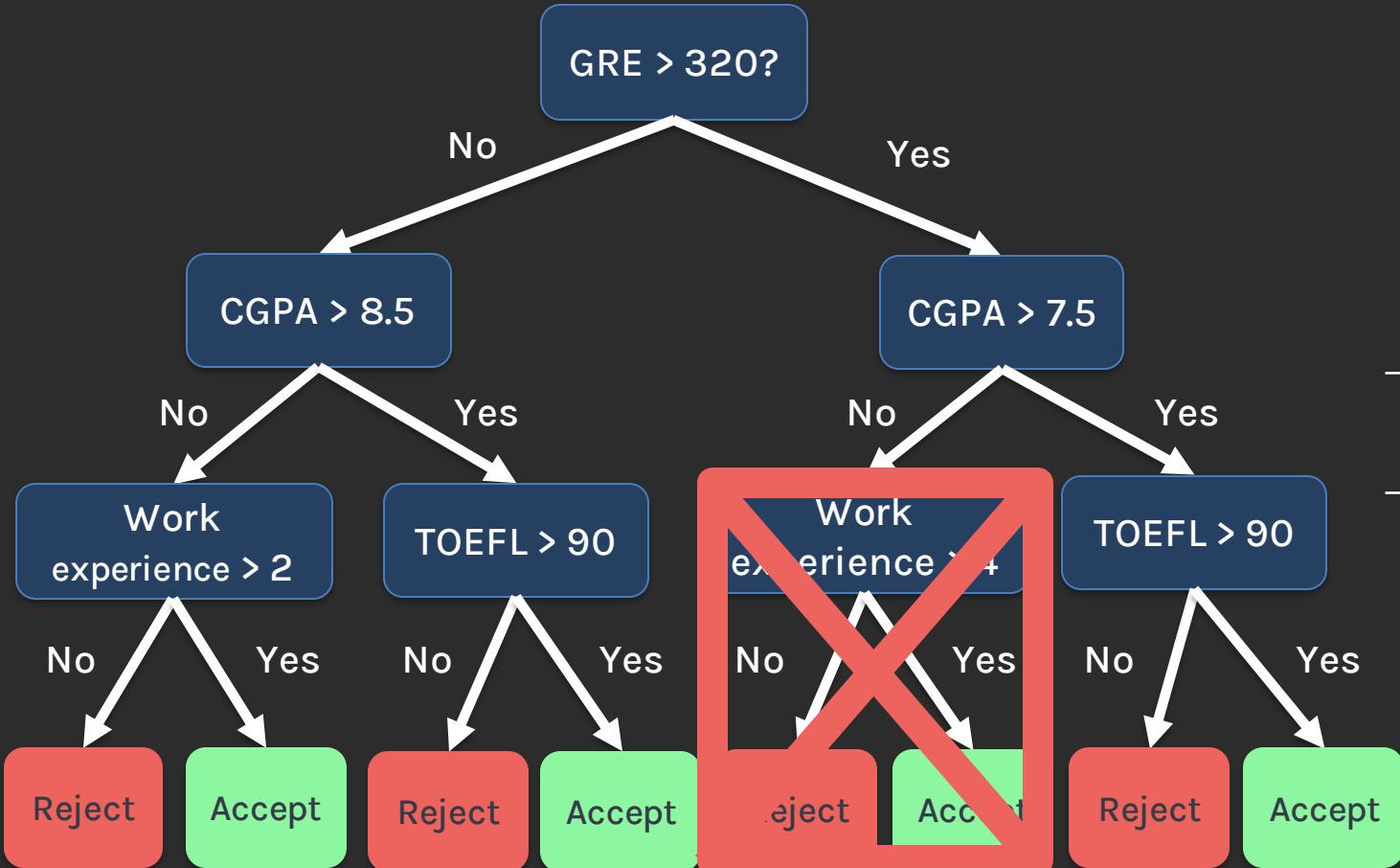
Cost-Complexity Pruning: Example



$$\alpha = 0.2$$

| Tree | Error(T) | T | Error(T) + $\alpha T $ |
|------|----------|---|-------------------------|
| T | 0.32 | 8 | $0.32 + 0.2 * 8 = 1.92$ |

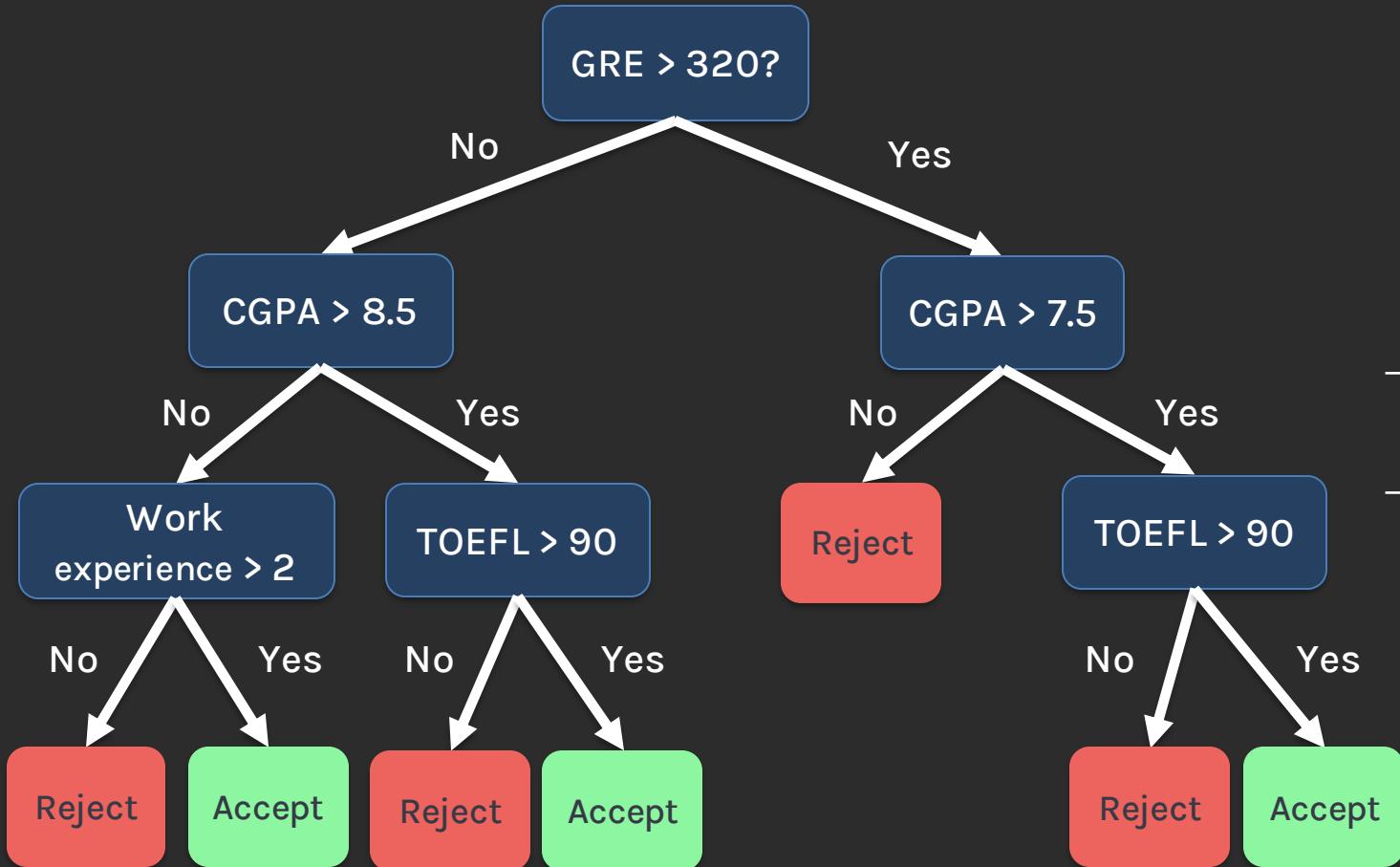
Cost-Complexity Pruning: Example



$$\alpha = 0.2$$

| Tree | Error(T) | T | Error(T) + $\alpha T $ |
|------|----------|---|-------------------------|
| T | 0.32 | 8 | 1.92 |

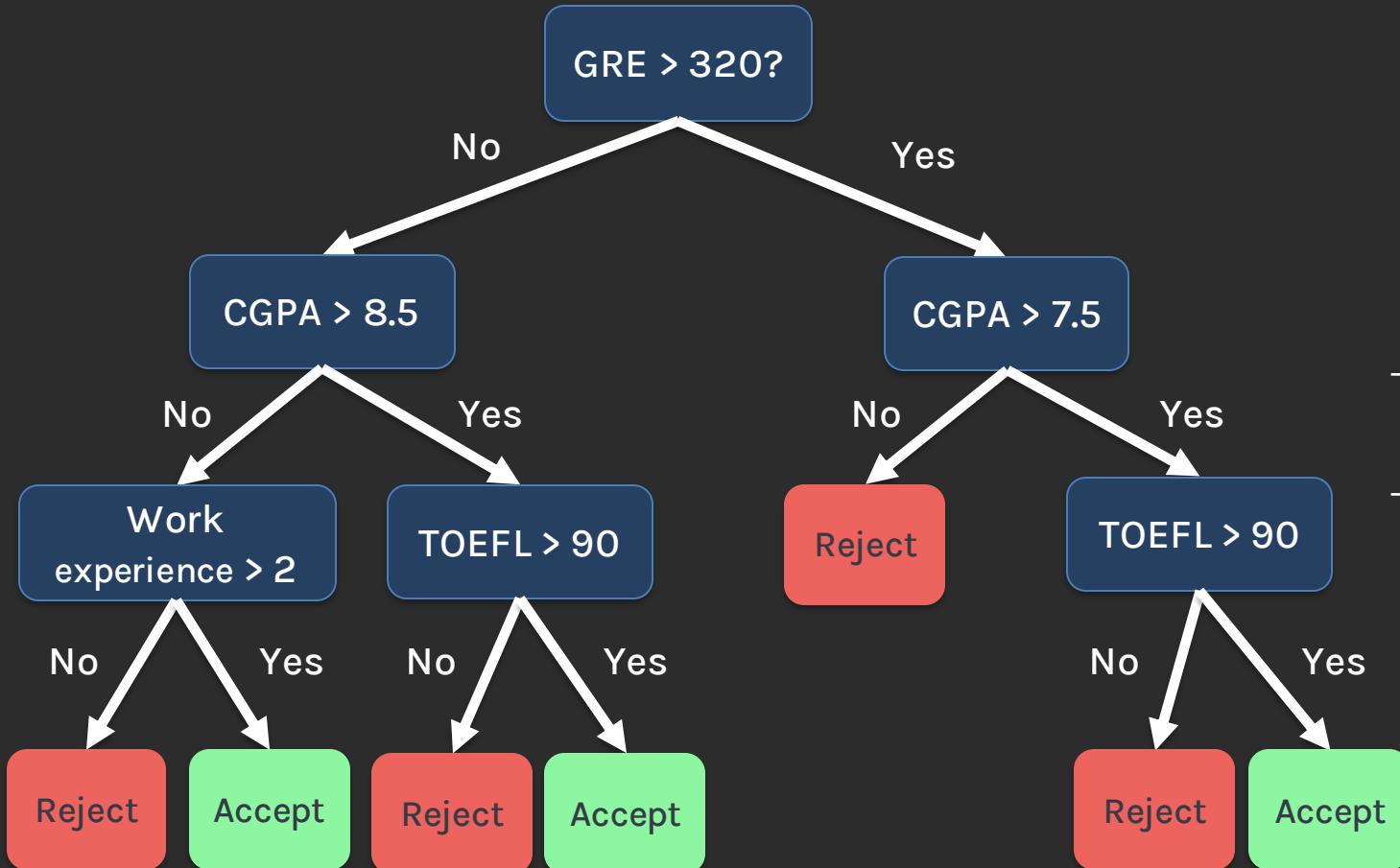
Cost-Complexity Pruning: Example



$$\alpha = 0.2$$

| Tree | Error(T) | T | Error(T) + $\alpha T $ |
|-------------|----------|---|-----------------------------|
| T | 0.32 | 8 | 1.92 |
| T_{small} | 0.33 | 7 | $0.33 + 0.2 \cdot 7 = 1.73$ |

Cost-Complexity Pruning: Example



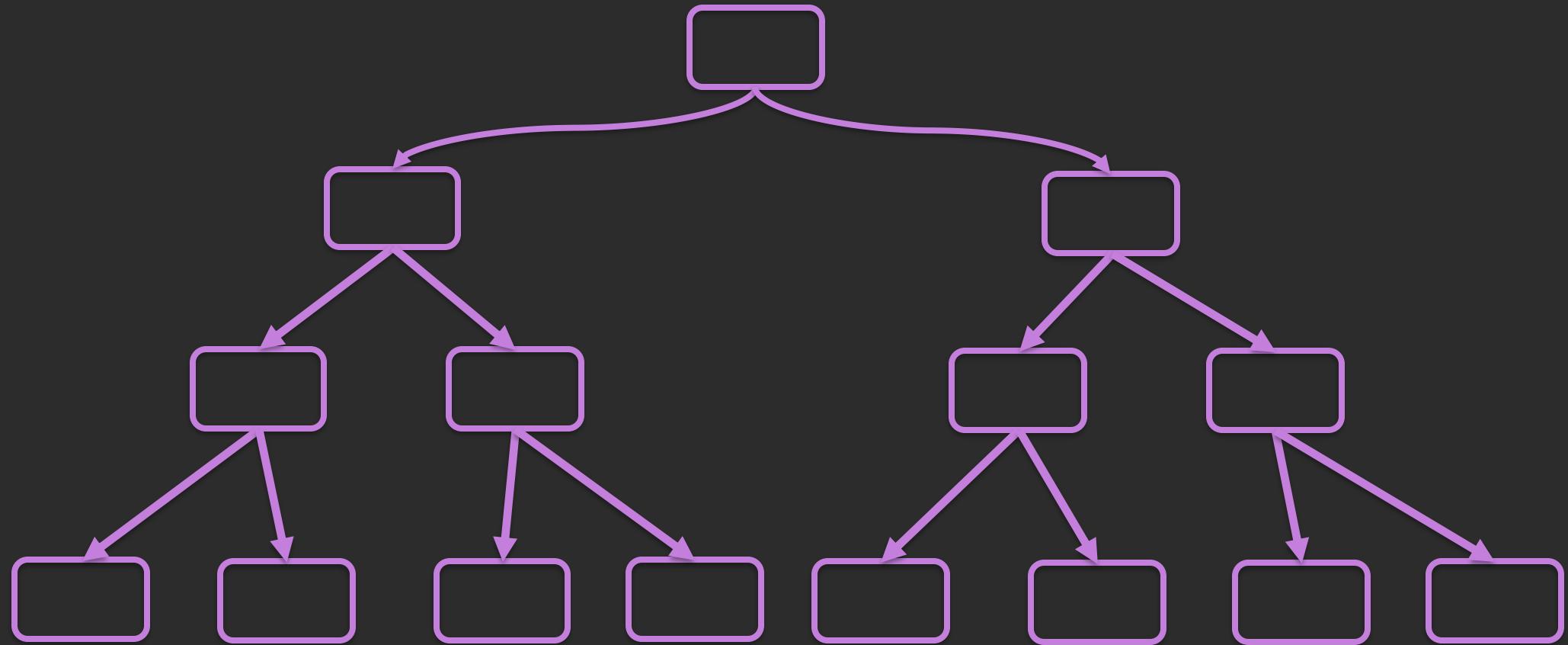
$$\alpha = 0.2$$

| Tree | Error(T) | T | Error(T) + $\alpha T $ |
|-------------|----------|---|-------------------------|
| T | 0.32 | 8 | 1.92 |
| T_{small} | 0.33 | 7 | 1.73 |

The smaller tree has a larger error $\text{Error}(T)$ but smaller complexity score $C(T)$.

Pruning

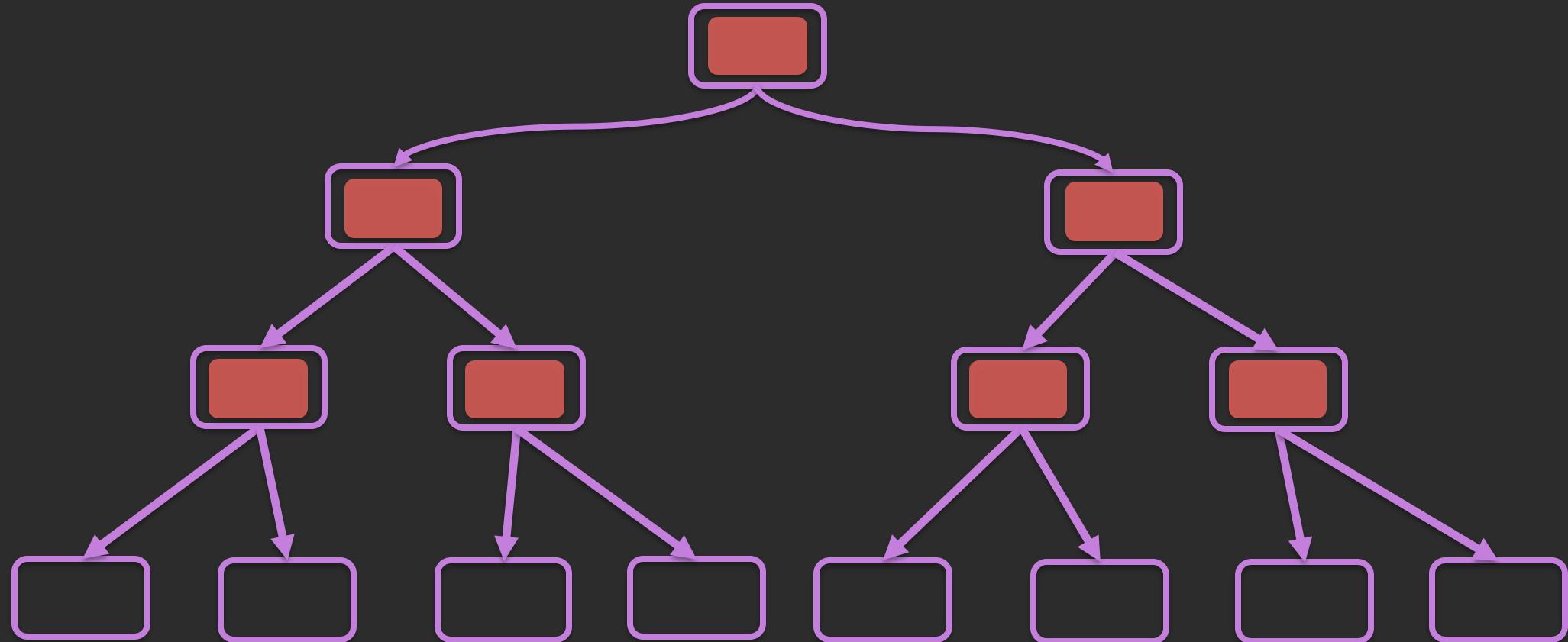
Suppose we have **a full tree T_0** as shown below.



How many possible ways are there to prune this tree?

Pruning

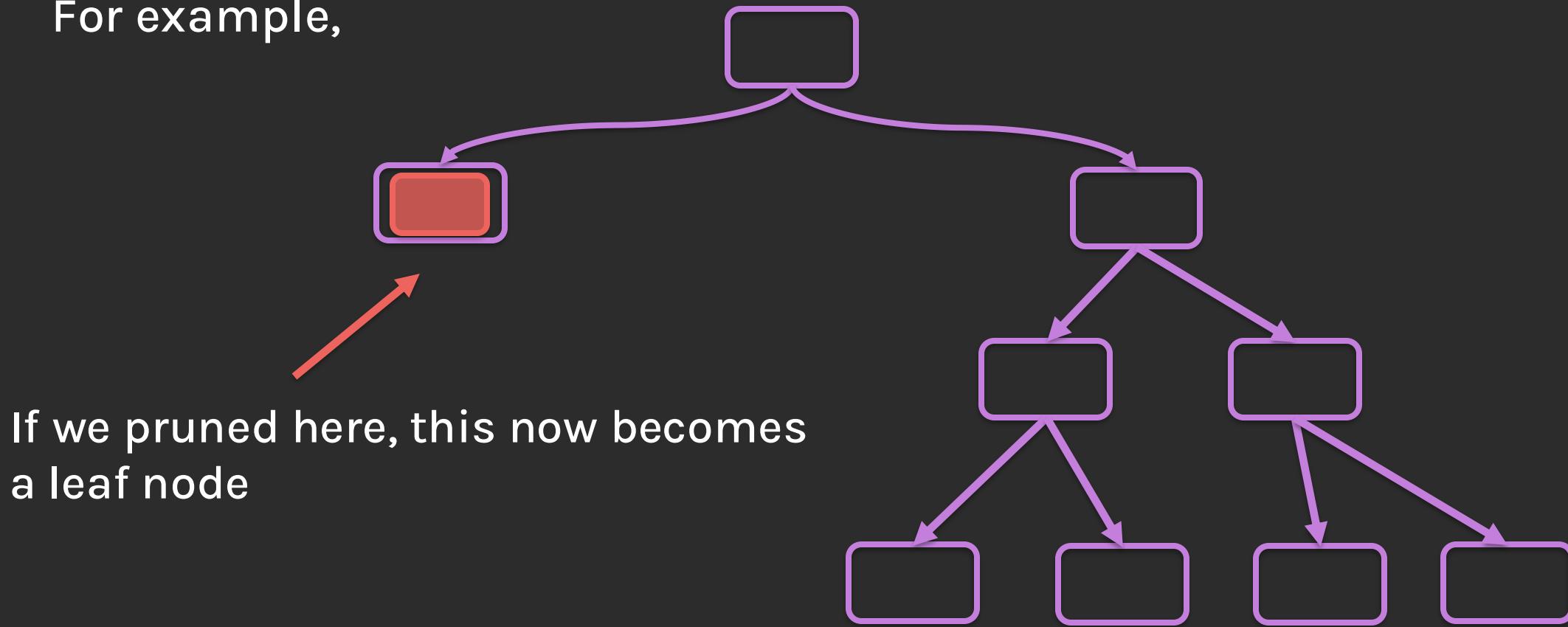
There are **7** possible pruning locations, which are shown with red rectangles.



Pruning

For each of those pruning locations, we will get 7 possible pruned trees, T^* .

For example,



Question: How do we choose the best pruned tree?

Pruning

We will choose the one that maximizes the difference of **cost complexity score** between a full tree and a pruned tree.

Quick recap: Cost complexity score is

$$C(T) = \text{Error}(T) + \alpha |T|$$

Classification Error

Complexity Parameter

Number of leaves in the tree

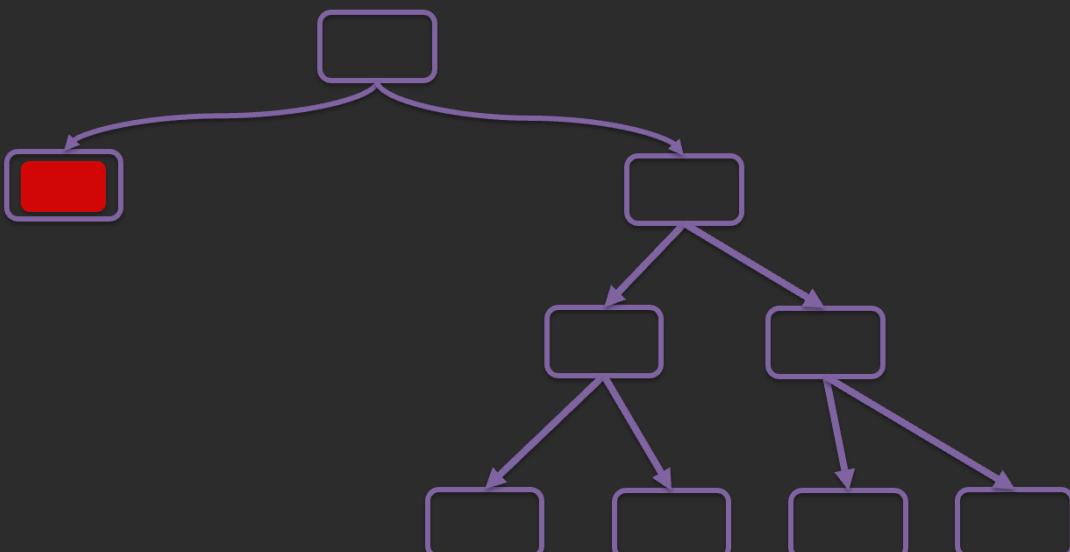
Decision tree

Pruning

Our goal is to maximize $C(T) - C(T^*)$

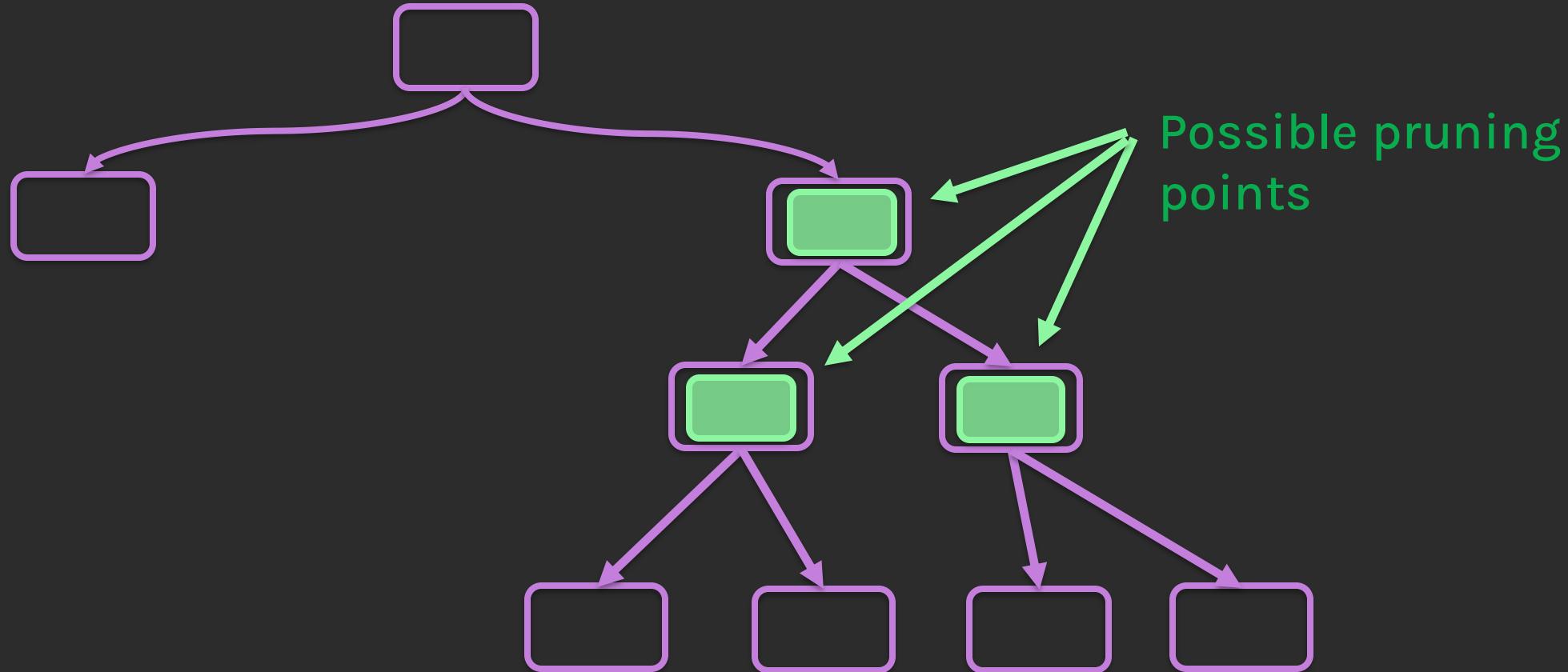
1. $\underset{T^*}{\operatorname{argmax}} [C(T) - C(T^*)]$
2. $\underset{T^*}{\operatorname{argmax}} [E(T) - E(T^*) + \alpha|T| - \alpha|T^*|]$
3. $\underset{T^*}{\operatorname{argmax}} \left[\frac{|E(T) - E(T^*)|}{\alpha|T^*| - \alpha|T|} - 1 \right]$ (divide by $\alpha|T^*| - \alpha|T|$)
4. $\underset{T^*}{\operatorname{argmax}} \left[\frac{|E(T) - E(T^*)|}{\alpha|T^*| - \alpha|T|} \right]$
5. $\underset{T^*}{\operatorname{argmin}} \left[\frac{|E(T) - E(T^*)|}{\alpha|T| - \alpha|T^*|} \right]$

This will result in a subtree, $T^{(1)}$



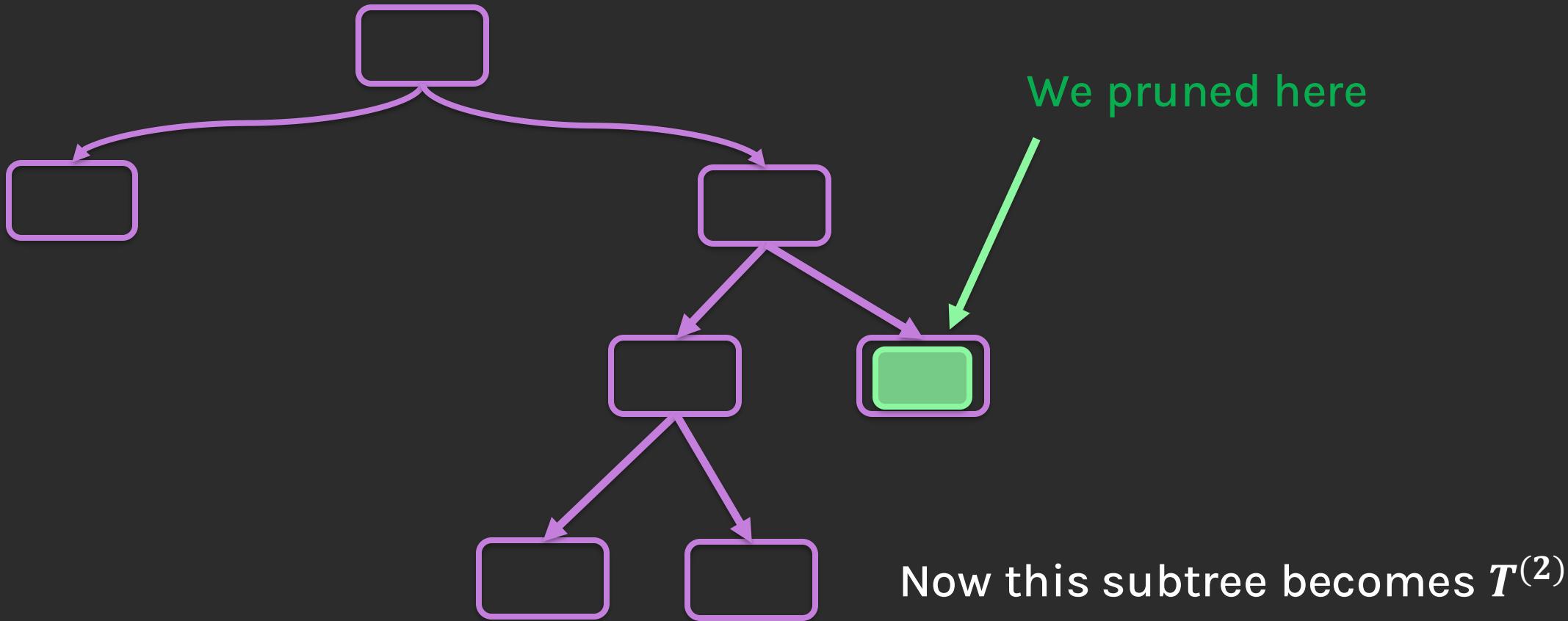
Pruning

Now, we again consider all possible pruning from $T^{(1)}$.



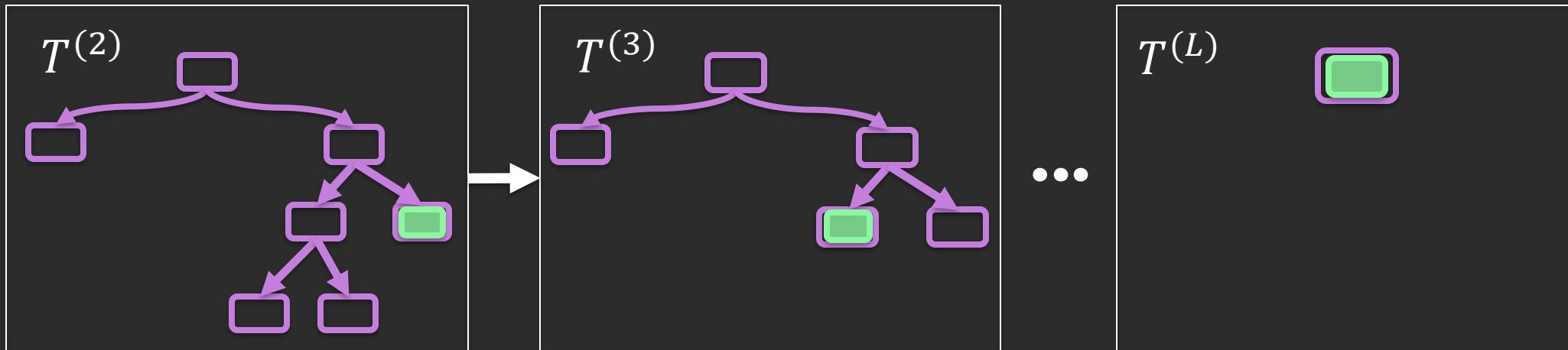
Pruning

We again minimize the ratio of the difference of the errors **over the difference in the complexity**



Pruning

We iterate this pruning process to obtain $T^{(2)}, T^{(3)}, \dots, T^{(L)}$ where $T^{(L)}$ is the tree containing just the root of $T^{(0)}$.



We select the optimal tree $T^{(i)}$ by cross validation.

This is the T^* given α . Finally, we choose **the optimal α** by cross validation!

Summary

What is the main drawback of pre-defining stopping criteria for decision tree growth?

Pre-defined stopping criteria may lead to trees that are either too simple (underfitting) or too complex (overfitting) since it's challenging to determine the optimal stopping point beforehand.

Explain the alternative approach to using stopping conditions for decision tree construction.

Instead of stopping conditions, an alternative approach is to grow a large tree first, then prune it back to find a balanced structure, offering flexibility for optimal complexity.

What is the core concept behind pruning in decision trees?

Pruning simplifies a decision tree by removing branches with minimal impact on overall accuracy, enhancing the tree's ability to generalize.

Summary

Describe the formula used for calculating the cost complexity of a decision tree.

Cost complexity ($C(T)$) is calculated as: $C(T) = \text{Error}(T) + \alpha|T|$, where $\text{Error}(T)$ is the classification error, α is the complexity parameter, and $|T|$ is the number of leaves.

In cost-complexity pruning, how does the complexity parameter (α) influence the trade-off between tree size and error?

The complexity parameter (α) adjusts the penalty for tree size, with higher values favoring smaller trees with potentially higher error and lower values favoring more complex trees with potentially lower error.

How do you determine the best pruned tree from a set of candidate trees generated by pruning?

The best pruned tree is selected by maximizing the difference in cost complexity scores, choosing the tree with the most significant reduction in complexity without a major increase in error.

Thank you





Regression Using Trees

CS109A Introduction to Data Science

Pavlos Protopapas, Natesh Pillai and Chris Gumb

Outline

- Decision Trees - Regression
- Numerical vs Categorical Attributes
- Pruning

Outline

- Decision Trees – Regression
- Numerical vs Categorical Attributes
- Pruning



Regression Trees

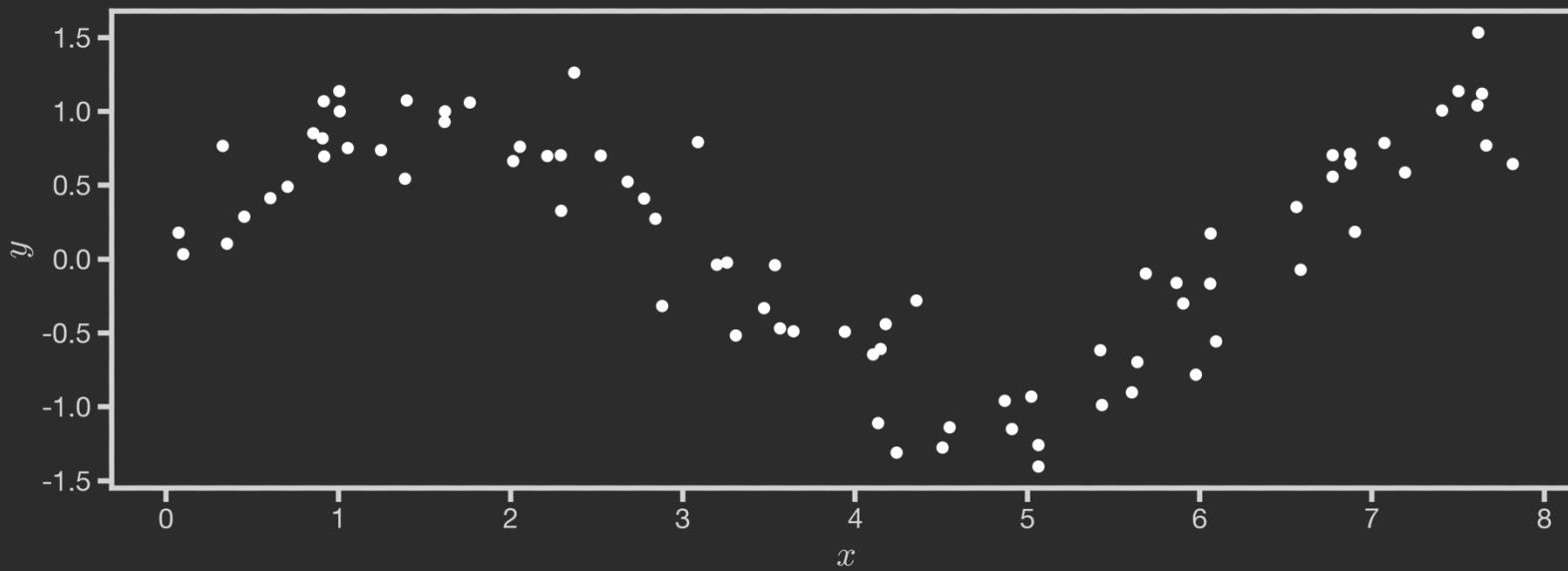
How can this decision tree approach apply to a *regression problem* (quantitative outcome)?

Questions to consider:

- How would you determine any **splitting criteria**?
- What would be a reasonable **objective function**?
- How would you perform **prediction at each leaf**?

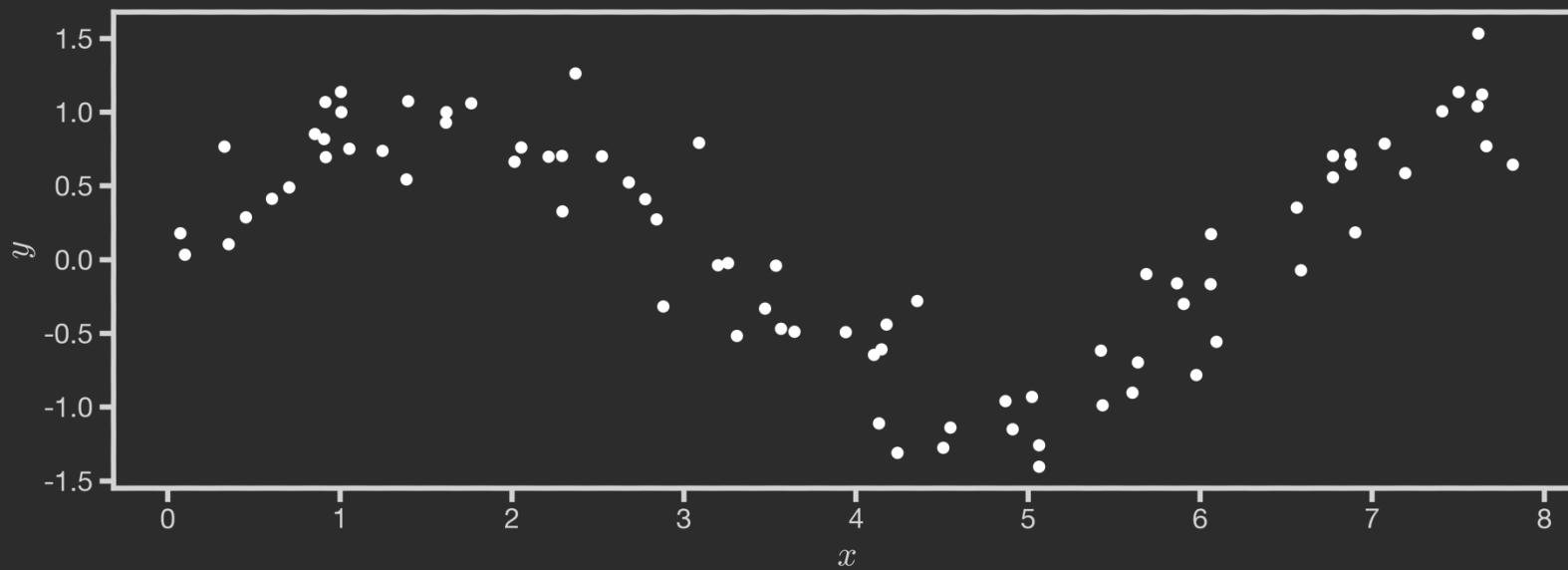
Splitting Criteria

The plot visualizes data points of $\{x, y\}$. By observing patterns, we can identify how and where to make optimal splits. Ideally, each split should lead to groups of data points with homogeneous values, allowing us to better predict based on values within that group.

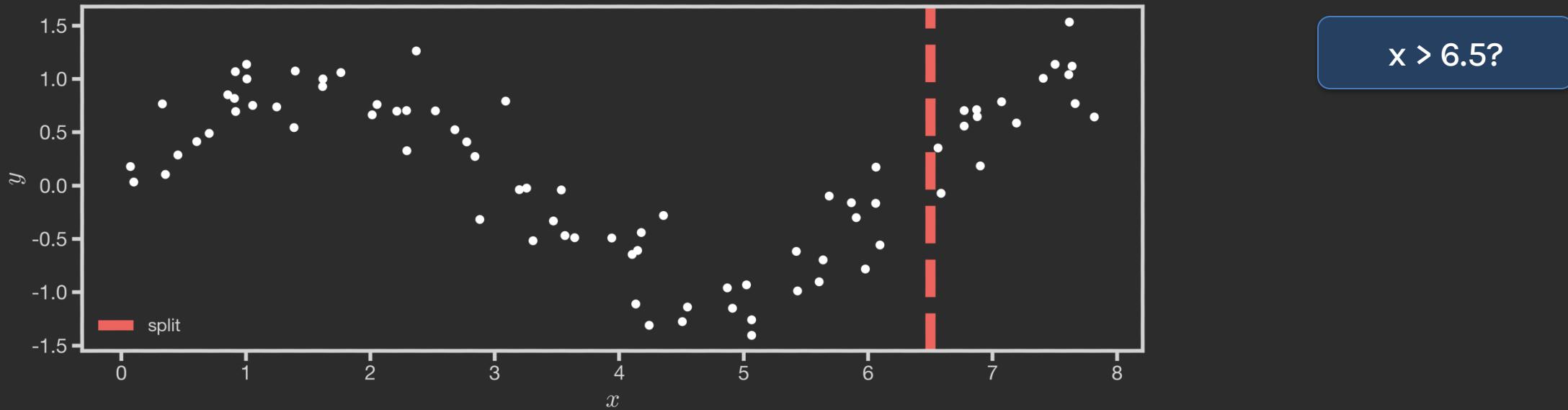


Splitting Criteria

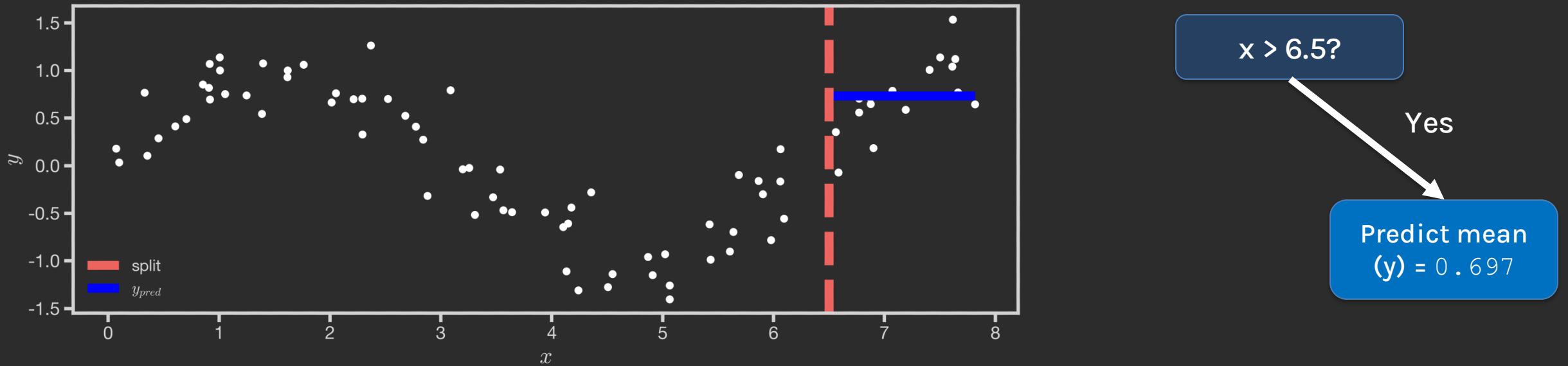
Idea: Divide points into groups of homogeneous values. Predict for each group.



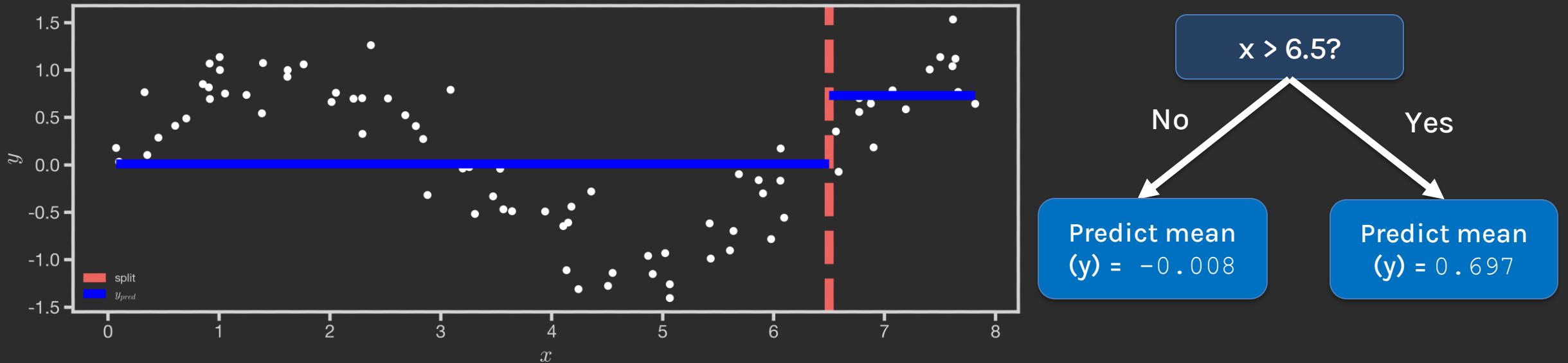
Splitting Criteria



Splitting Criteria

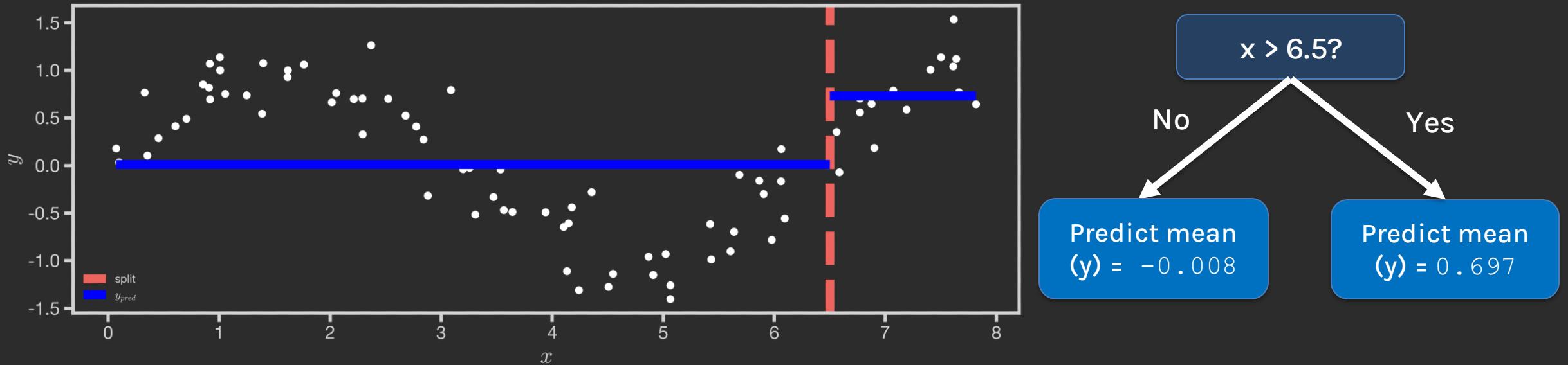


Splitting Criteria



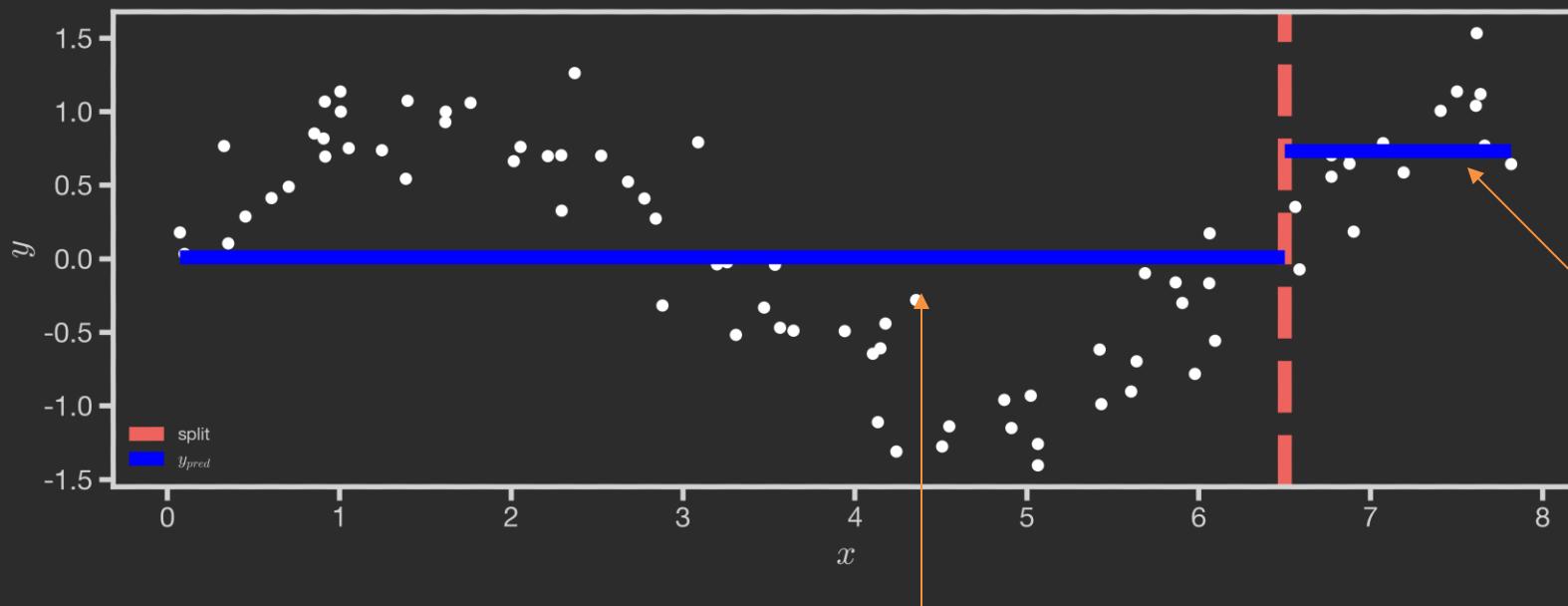
Splitting Criteria

Question: How did we choose the splitting criteria for this regression tree?



Splitting Criteria

We can assess the quality of this split by calculating the mean squared error for each newly created region:

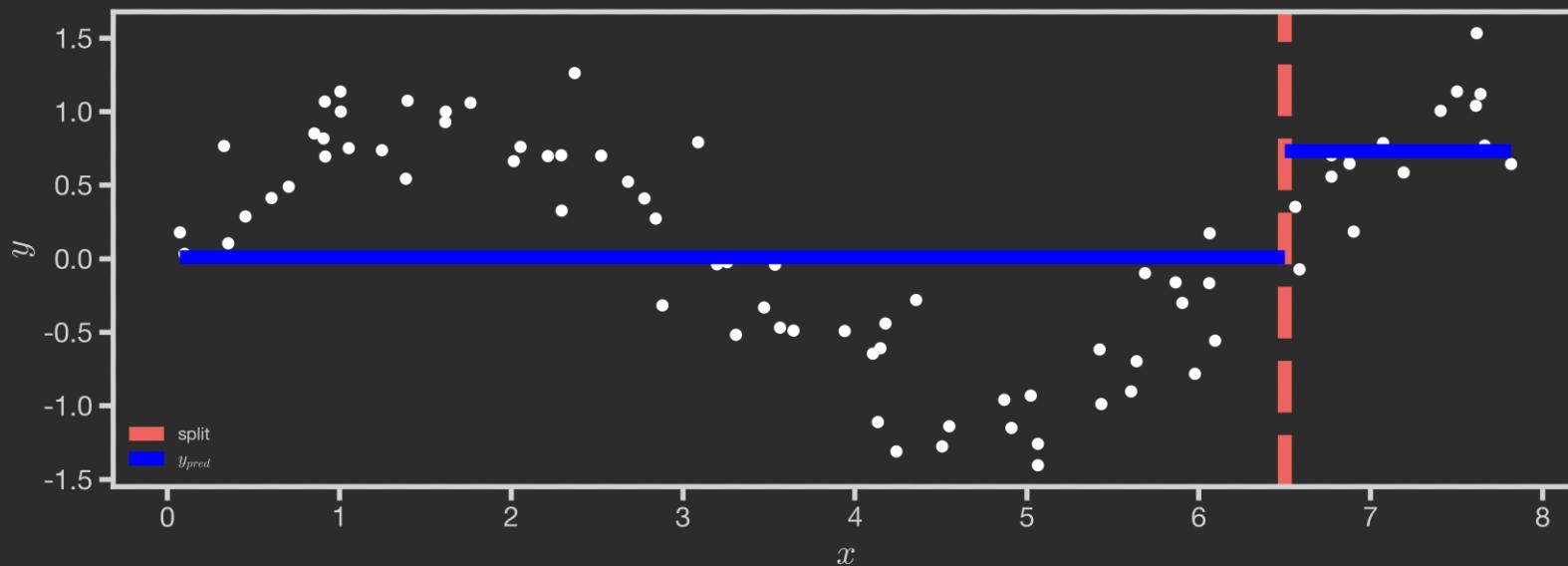


$$MSE(R_1) = \frac{1}{n_1} \sum_{i \in R_1} (y_i - \bar{y}_{R_1})^2$$

$$MSE(R_2) = \frac{1}{n_2} \sum_{i \in R_2} (y_i - \bar{y}_{R_2})^2$$

Splitting Criteria

We can assess the quality of this split by calculating the mean squared error for each newly created region:



This is the prediction

$$MSE(R_r) = \frac{1}{n_r} \sum_{i \in R_r} (y_i - \bar{y}_{R_r})^2$$

Note: This is the “same” as the variance within region R_r !

Splitting Criteria

When calculating the MSE, we need to consider the number of points in each region. So, we take the **weighted** average over both regions.

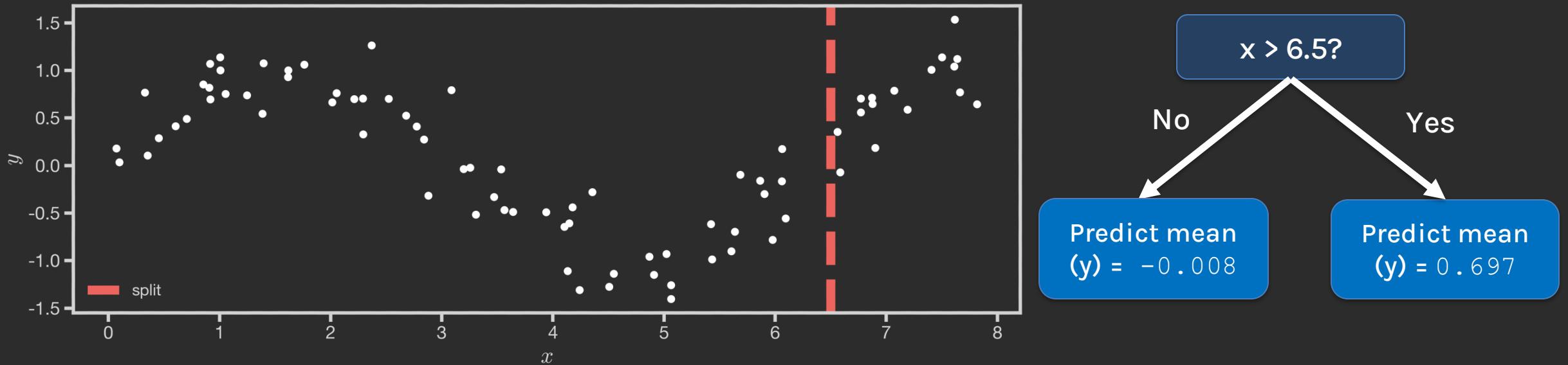
$$\min_{p, t_p} \left[\frac{N_1}{N} MSE(R_1) + \frac{N_2}{N} MSE(R_2) \right]$$



Reminder: p denotes the predictor, and t_p denotes the threshold.

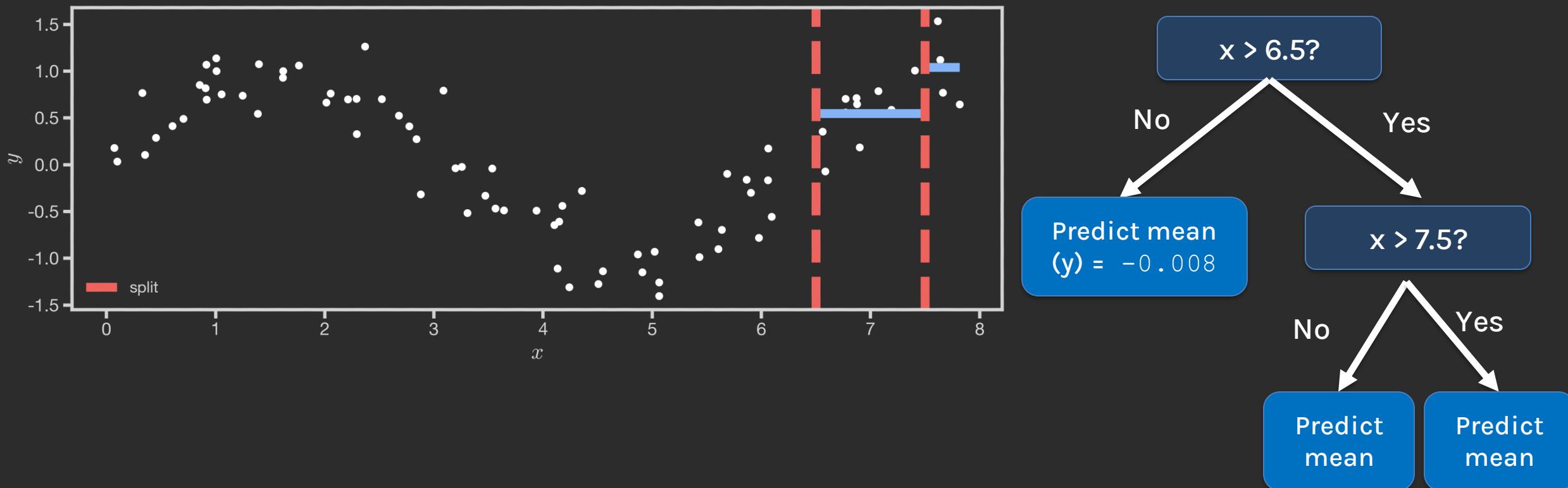
Splitting Criteria

Question: How did we choose the splitting criteria for this regression tree?

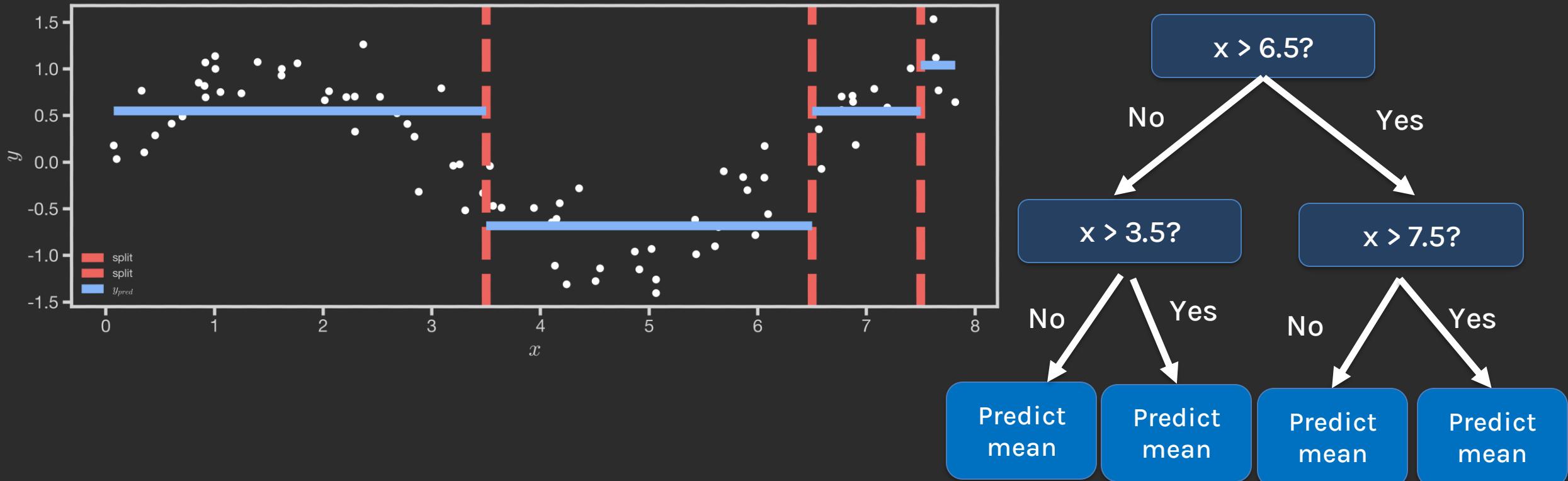


Splitting Criteria

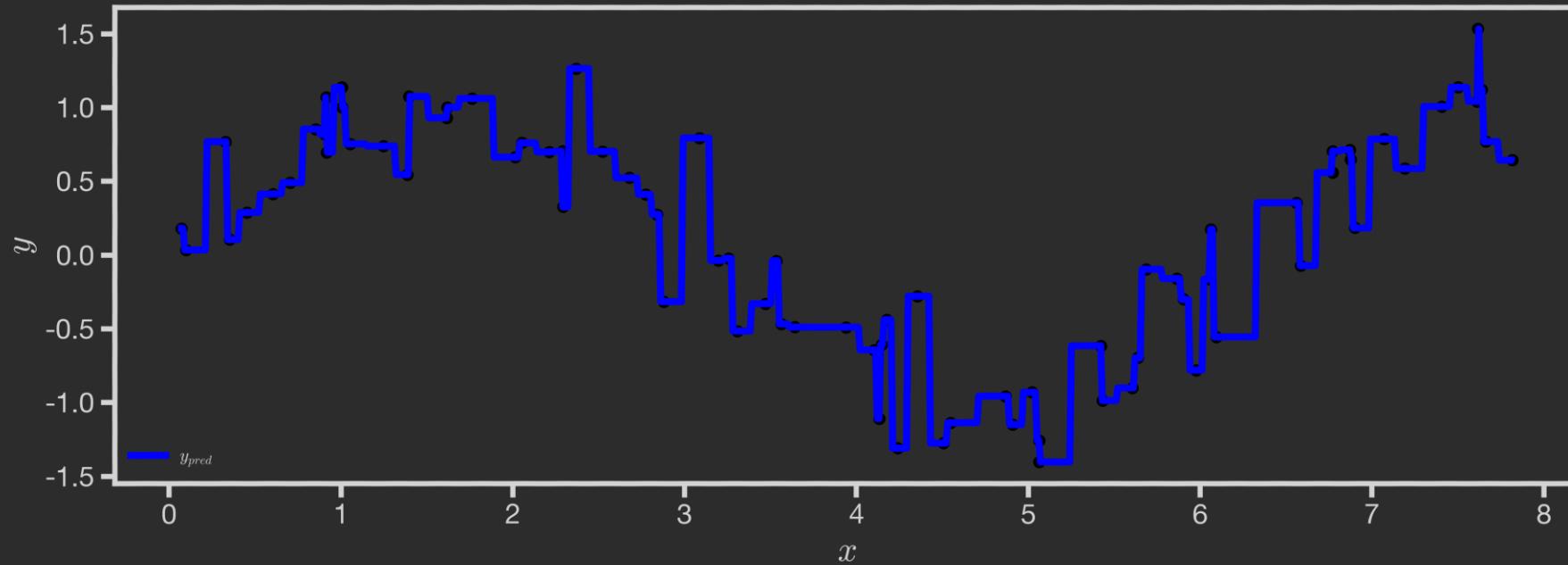
We continue splitting in the same way.



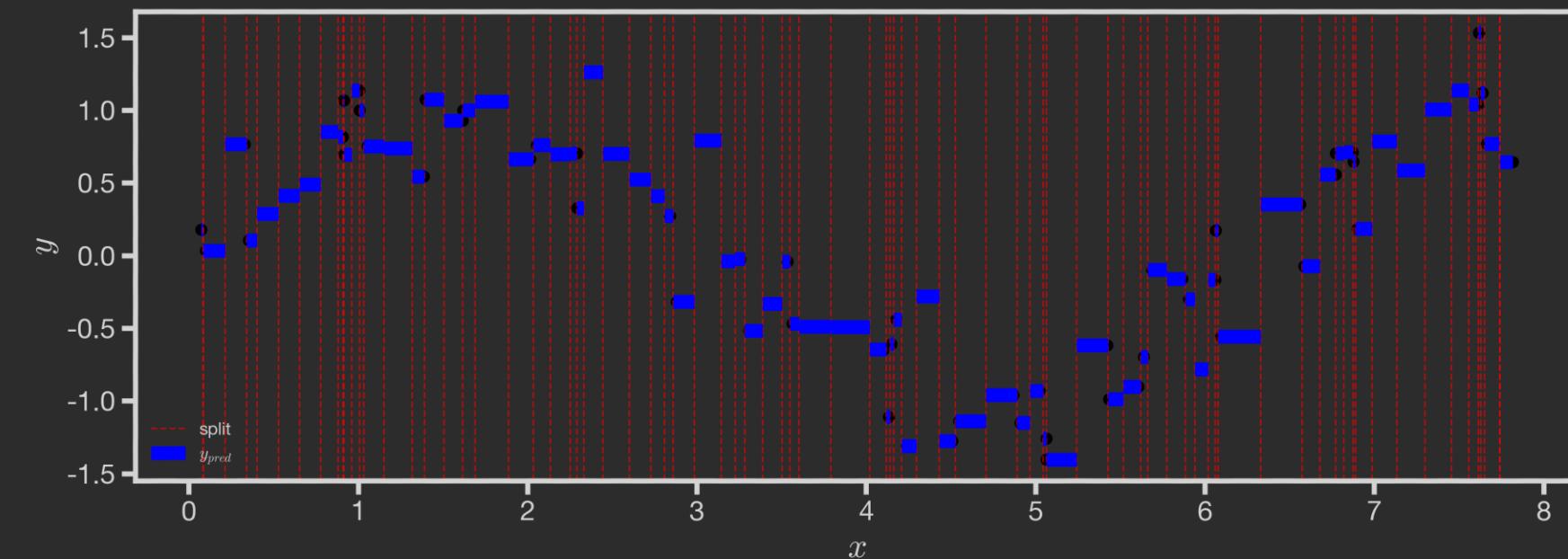
Regression Tree (max_depth = 2)



Regression Tree (max_depth = 5)



Regression Tree (max_depth = 10)



Stopping Conditions

Most of the stopping conditions we saw for classification trees, such as **maximum depth** or **minimum number** of points in a region, can also be applied to regression.

Instead of purity gain, we can compute **accuracy gain** (in this case **MSE reduction**) for splitting a region R and stop the tree when the gain is less than some pre-defined threshold.

$$Gain(R) = \Delta(R) = MSE(R) - \left(\underbrace{\frac{N_1}{N} MSE(R_1) + \frac{N_2}{N} MSE(R_2)}_{MSE \text{ after split}} \right)$$

$\underbrace{\phantom{\frac{N_1}{N} MSE(R_1) + \frac{N_2}{N} MSE(R_2)}}_{MSE \text{ without split}}$

Regression Trees Prediction

For any data point x_i

1. Traverse the tree until we reach a leaf node.
2. Predict \hat{y}_i to be the **averaged value** of the response variable y 's in the leaf (this is from the **training set**).

Outline

- Decision Trees - Regression
- Numerical vs Categorical Attributes
- Pruning

Numerical vs Categorical Attributes

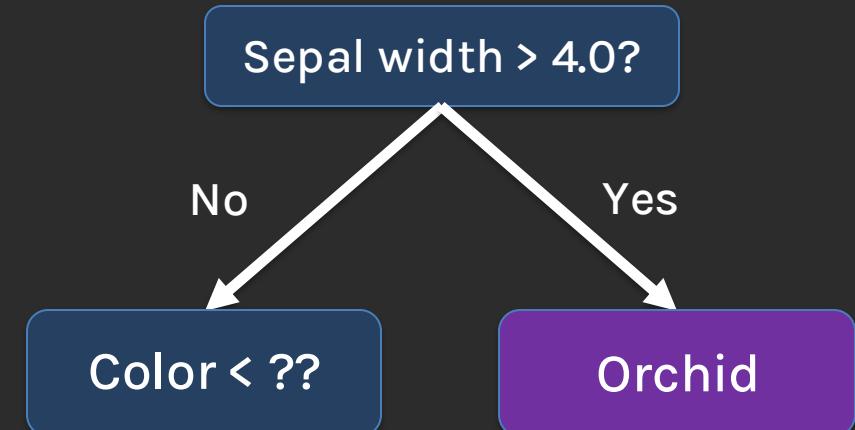
Consider the following data:

| Sepal width | Color | Flower |
|-------------|--------|-----------|
| 3.0 mm | Yellow | Sunflower |
| 3.5 mm | Red | Rose |
| 3.7 mm | Purple | Tulip |
| 4.5 mm | Purple | Orchid |

Question: How do we construct a decision tree for this data?

Numerical vs Categorical Attributes

| Sepal width | Color | Flower |
|-------------|--------|-----------|
| 3.0 mm | Yellow | Sunflower |
| 3.5 mm | Red | Rose |
| 3.7 mm | Purple | Tulip |
| 4.5 mm | Purple | Orchid |



Note that the ‘compare and branch’ method by which we defined classification trees works well for numerical features.

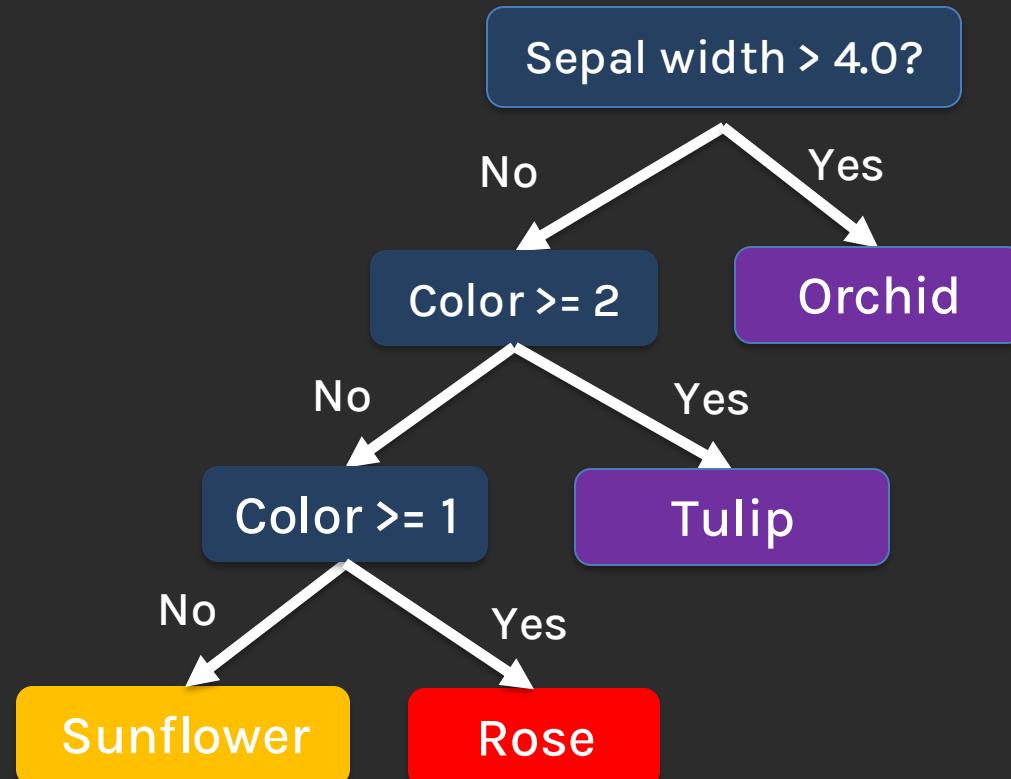
If a feature is categorical (with more than two possible values), a comparison like ***feature < threshold*** does not make sense.

Numerical vs Categorical Attributes

A simple solution is to encode the values of a categorical feature using numbers and treat this feature like a numerical variable.

If we encode **Yellow** = 0, **Red** = 1, **Purple** = 2, our decision tree can be:

| Sepal width | Color | Flower |
|-------------|-------|-----------|
| 3.0 mm | 0 | Sunflower |
| 3.5 mm | 1 | Rose |
| 3.7 mm | 2 | Tulip |
| 4.5 mm | 2 | Orchid |



Numerical vs Categorical Attributes

In the example, we encoded:

$$\text{Yellow} = 0, \text{Red} = 1, \text{Purple} = 2$$

Then the possible non-trivial splits on **color** are:

$$\{\{\text{Yellow}\}, \{\text{Red}, \text{Purple}\}\} \text{ and } \{\{\text{Yellow}, \text{Red}\}, \{\text{Purple}\}\}$$

Color>0

Color>1

But if we encode the categories numerically as:

$$\text{Yellow} = 2, \text{Red} = 0, \text{Purple} = 1$$

The possible splits are:

$$\{\{\text{Red}\}, \{\text{Yellow}, \text{Purple}\}\} \text{ and } \{\{\text{Red}, \text{Purple}\}, \{\text{Yellow}\}\}$$

Color>0

Color>1

Numerical vs Categorical Attributes

$\{\text{Yellow}\}, \{\text{Red, Purple}\}$ and $\{\text{Yellow, Red}\}, \{\text{Purple}\}$

$\{\text{Red}\}, \{\text{Yellow, Purple}\}$ and $\{\text{Red, Purple}\}, \{\text{Yellow}\}$

Depending on the encoding, the splits we optimize over can be different!

Numerical vs Categorical Attributes

In the example, we used **ordinal encoding**. If your categorical data is not ordinal, this is not good. You'll end up with splits that do not make sense. How do we encode this?

One-hot-encoding or dummy encoding!

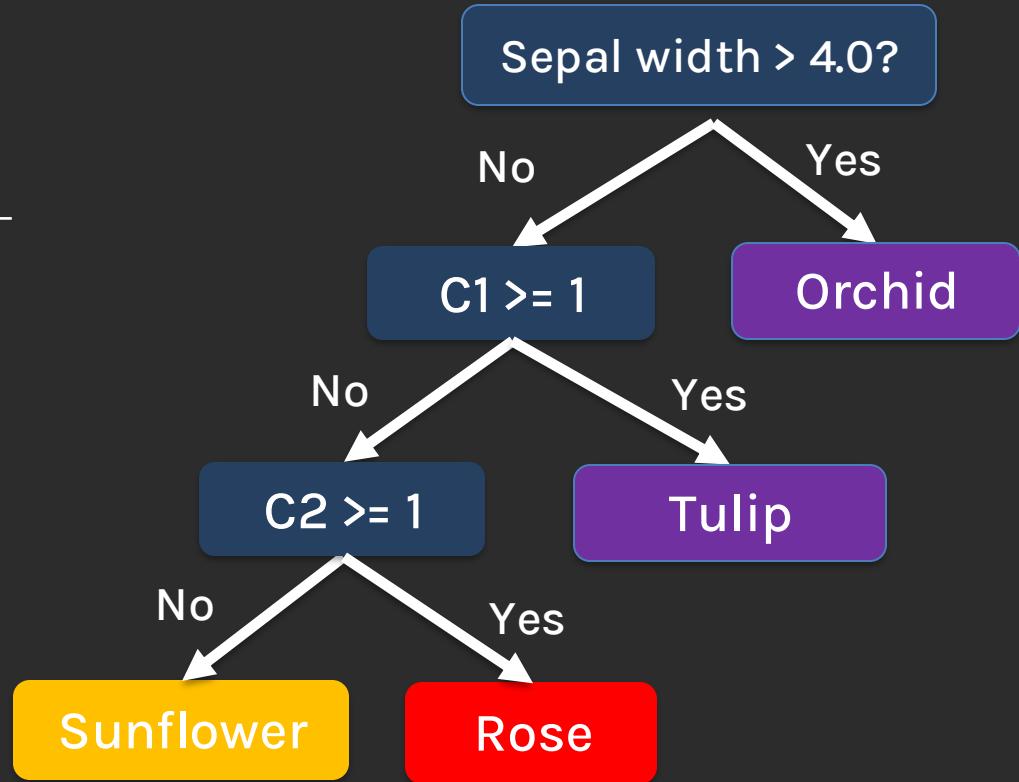
Numerical vs Categorical Attributes

| Sepal width | Color | Flower | OHE | Sepal width | C1 | C2 | C3 | Flower |
|-------------|--------|-----------|-----|-------------|----|----|----|-----------|
| 3.0 mm | Yellow | Sunflower | | 3.0 mm | 0 | 0 | 1 | Sunflower |
| 3.5 mm | Red | Rose | | 3.5 mm | 0 | 1 | 0 | Rose |
| 3.7 mm | Purple | Tulip | | 3.7 mm | 1 | 0 | 0 | Tulip |
| 4.5 mm | Purple | Orchid | | 4.5 mm | 1 | 0 | 0 | Orchid |

Numerical vs Categorical Attributes

| Sepal width | C1 | C2 | C3 | Flower |
|-------------|----|----|----|-----------|
| 3.0 mm | 0 | 0 | 1 | Sunflower |
| 3.5 mm | 0 | 1 | 0 | Rose |
| 3.7 mm | 1 | 0 | 0 | Tulip |
| 4.5 mm | 1 | 0 | 0 | Orchid |

We do not need this!



Categorical Predictors

As it stands, sklearn decision trees do not handle categorical data.

From sklearn [documentation](#):

scikit-learn uses an optimized version of the CART algorithm; however, scikit-learn implementation does not support categorical variables for now.

In practice, the **effects** of our **choice** of naive **encoding** of categorical variables are often **negligible**. Models resulting from **different choices** of encoding will perform comparably.

Summary

How does the prediction process differ between classification trees and regression trees?

Classification trees predict a class label for each leaf node, while regression trees predict the average value of the response variable in the leaf.

Explain how mean squared error (MSE) is used as a splitting criterion in regression trees.

MSE is calculated for each potential split, representing the average squared difference between predicted and actual values in each region. The split with the lowest weighted MSE is chosen.

Describe the concept of “accuracy gain” in the context of regression tree pruning.

Accuracy gain is the reduction in MSE from a split. If the gain is below a threshold, the split is skipped, effectively pruning the tree.

Summary

Why is a simple numerical encoding of categorical features often problematic for decision trees?

Numerical encoding can impose an artificial order on categories, leading to illogical splits, especially for non-ordinal data.

Compare and contrast ordinal encoding and one-hot encoding for categorical features in decision tree models.

Ordinal encoding assigns integers based on order, which can be useful for ordinal data. One-hot encoding creates binary features for each category, avoiding implied order but increasing dimensionality.

Explain why a comparison like “feature < threshold” is not applicable to categorical features.

Categorical features lack natural numerical ordering, so threshold comparisons don’t make sense for them.



Bagging: Introduction to Bagging

CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai and Chris Gumb

Steven Liu
Indonesia

Outline

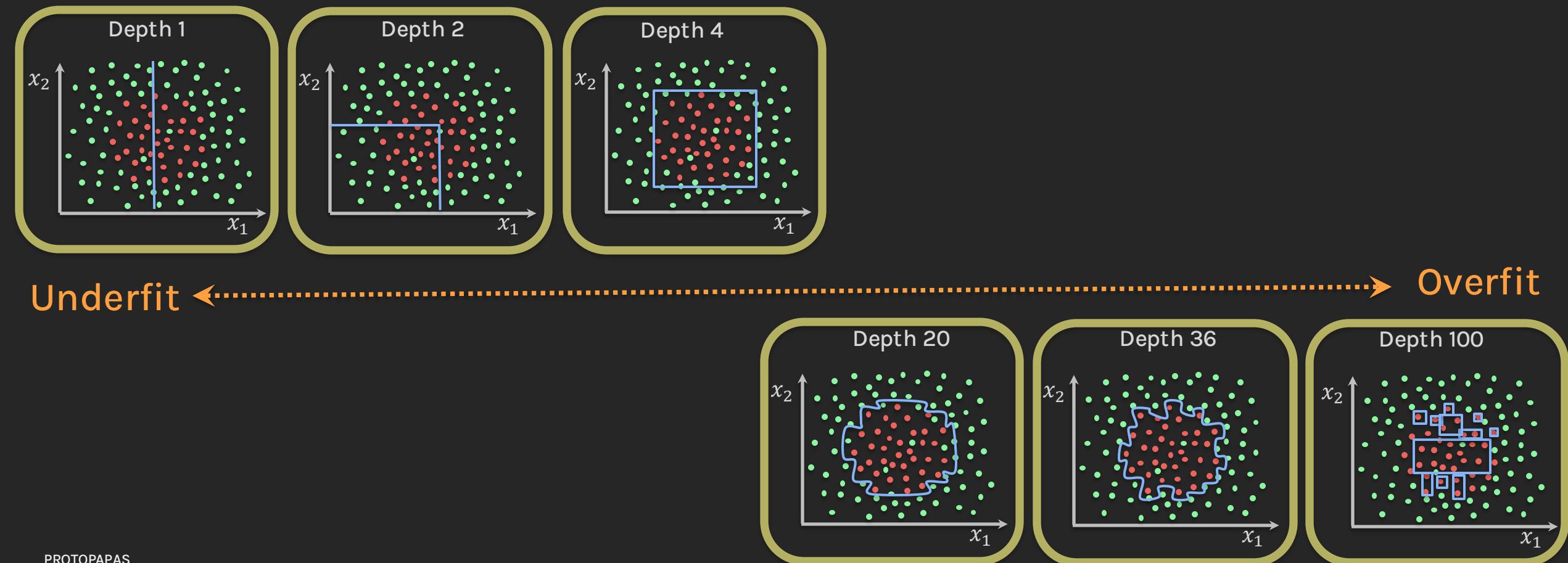
- Motivation
- Bagging
- Out-of-bag Error

Outline

- Motivation
- Bagging
- Out-of-bag Error

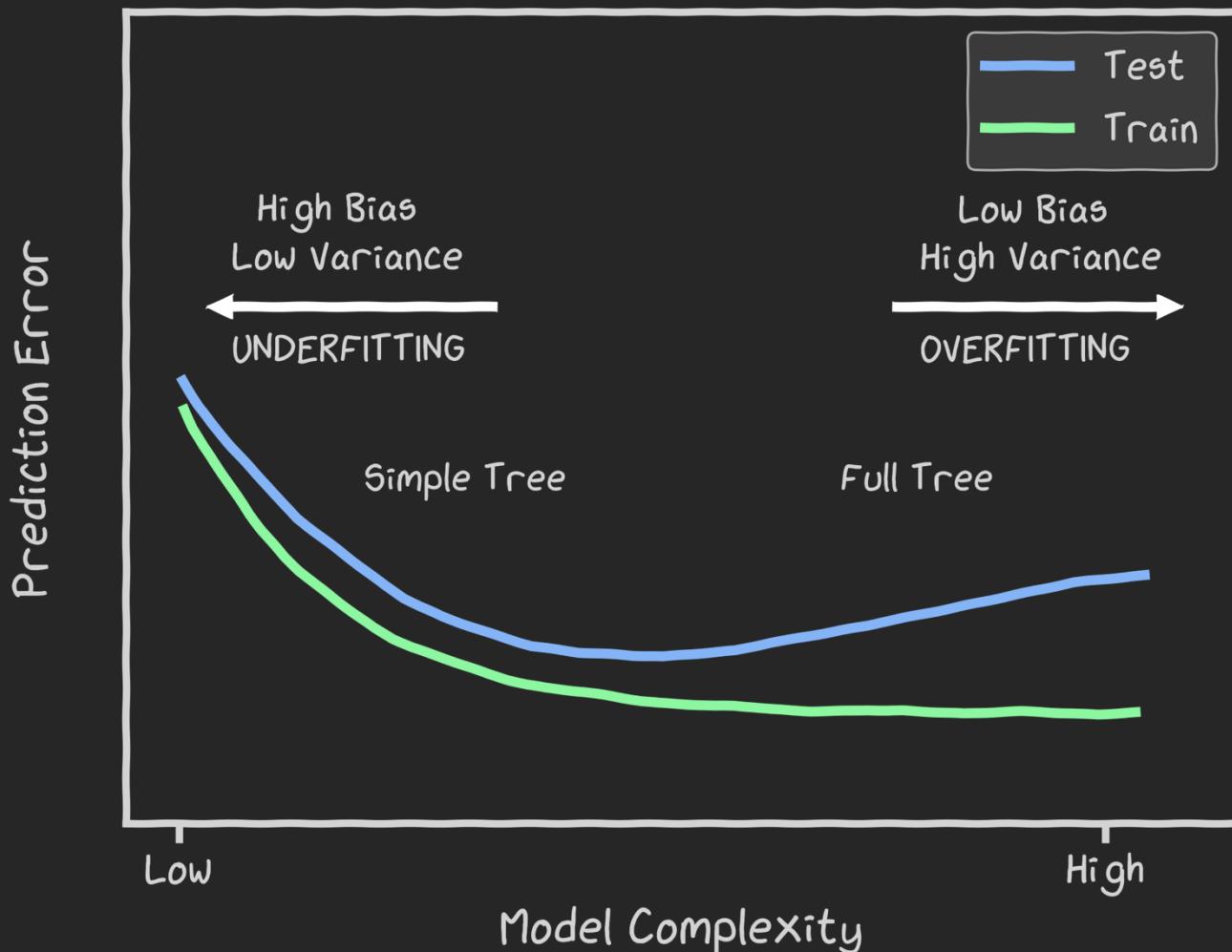
Underfitting and Overfitting

When a tree is too **shallow**, it cannot divide the input data into enough regions, so the model **underfits**. When the tree is too **deep**, it cuts the input space into too many regions and fits to the noise of the data, so it **overfits**.



Overfitting

Avoid overfitting by pruning or limiting the depth of the tree and using CV.



Limitations of Decision Tree Models

Using a greedy algorithm, decision trees models are highly interpretable and fast to train.

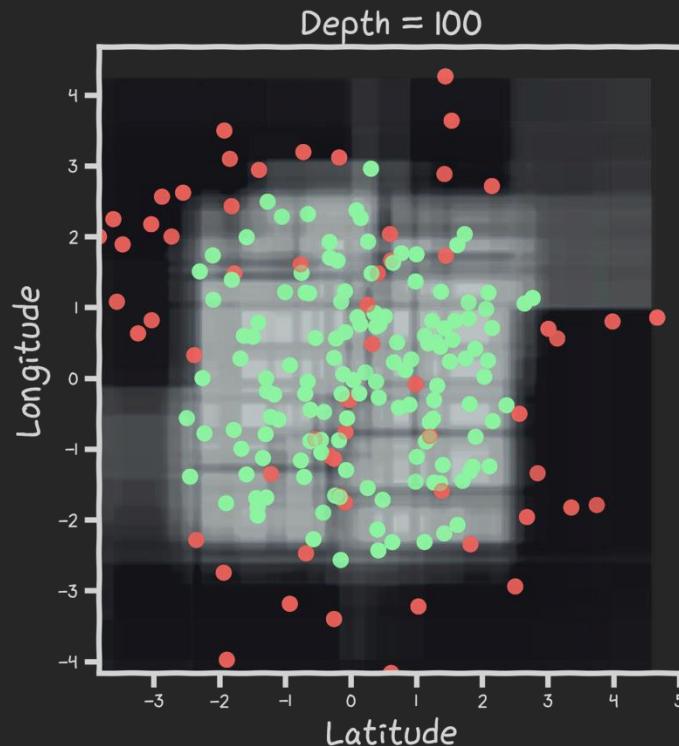
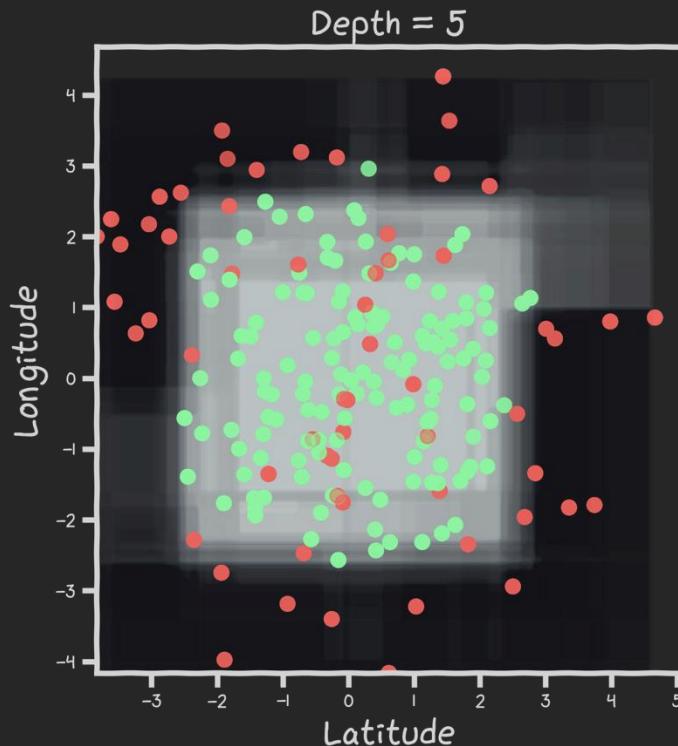
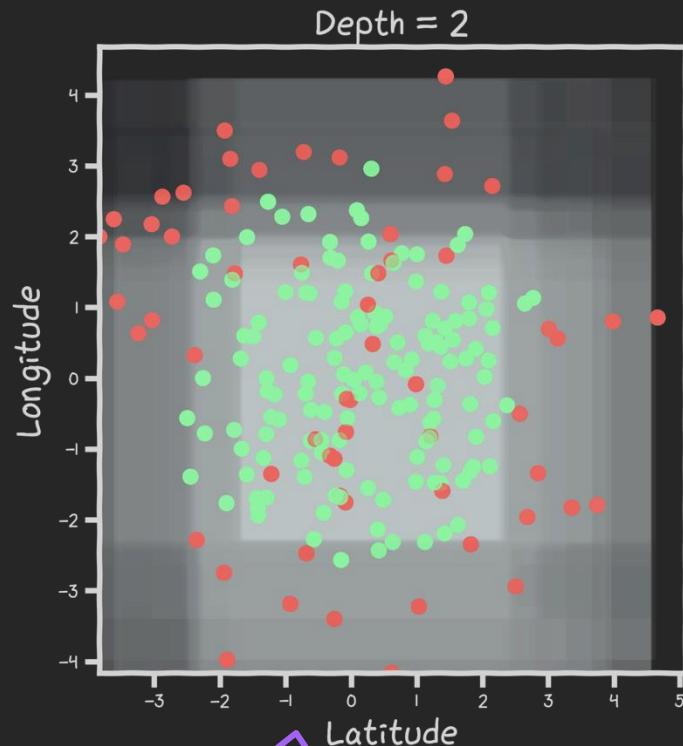
However, to **capture a complex decision boundary** (or approximate a complex function for regression), we must do axis-aligned splits. As a result, we need to use a large tree or deep tree.

Deep trees have high variance and are prone to overfitting.

For these reasons, in practice, decision tree models **underperform** in comparison to other classification or regression methods.

Motivation for Bagging

Decision tree models often underperform when compared to other classification or regression methods in situations of complex decision boundaries.



In certain cases it underfits.

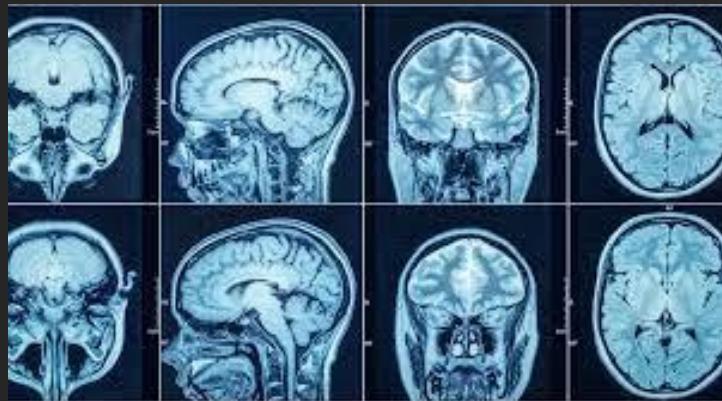
... and in certain cases it overfits!

Outline

- Motivation
- Bagging
- Out-of-bag Error

Ensemble Learning - Intuition

The intuition of ensemble learning is to build a single model by training and aggregating multiple models.



Consider the case of MRI scans of patients and the goal is to find out if they have a brain tumor or not.

Ensemble learning is like consulting multiple doctors instead of just one.

Ensemble Learning - Intuition



Instead of relying on the prediction from **one specialist** we consult **multiple doctors**.

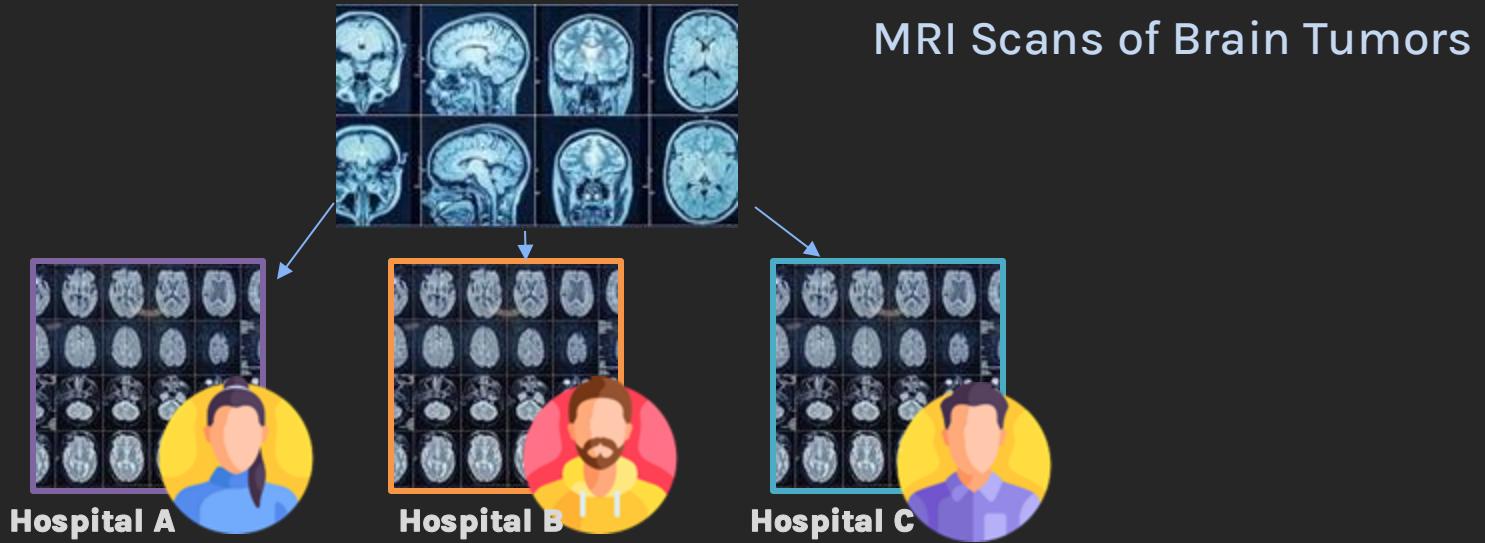
We individually ask each of these doctors to predict whether the scan indicates the presence of brain tumor.

Each 'specialist' or model views the MRI scans and makes a prediction. Their collective decisions are then **aggregated** to form a **final verdict**. This approach helps in mitigating individual model errors, increasing the accuracy of the prediction.

Ensemble Learning - Intuition

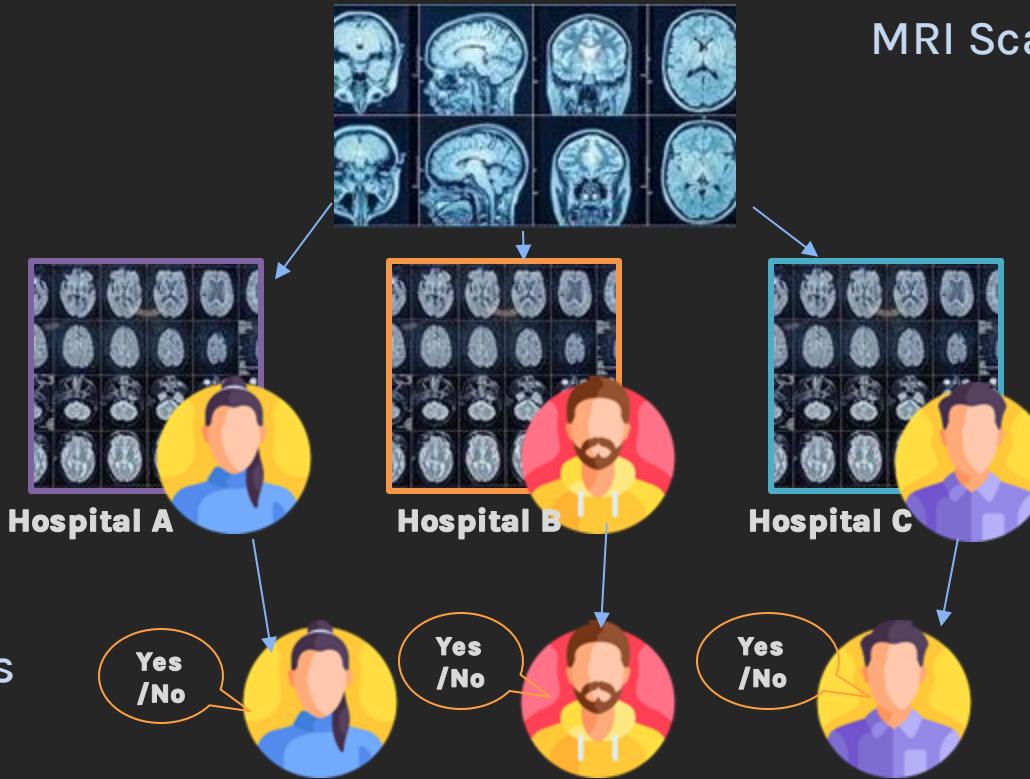
Doctors who are

- trained at different hospitals and
- trained on the examples available to them



Ensemble Learning - Intuition

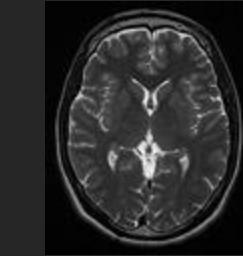
MRI Scans of Brain Tumors



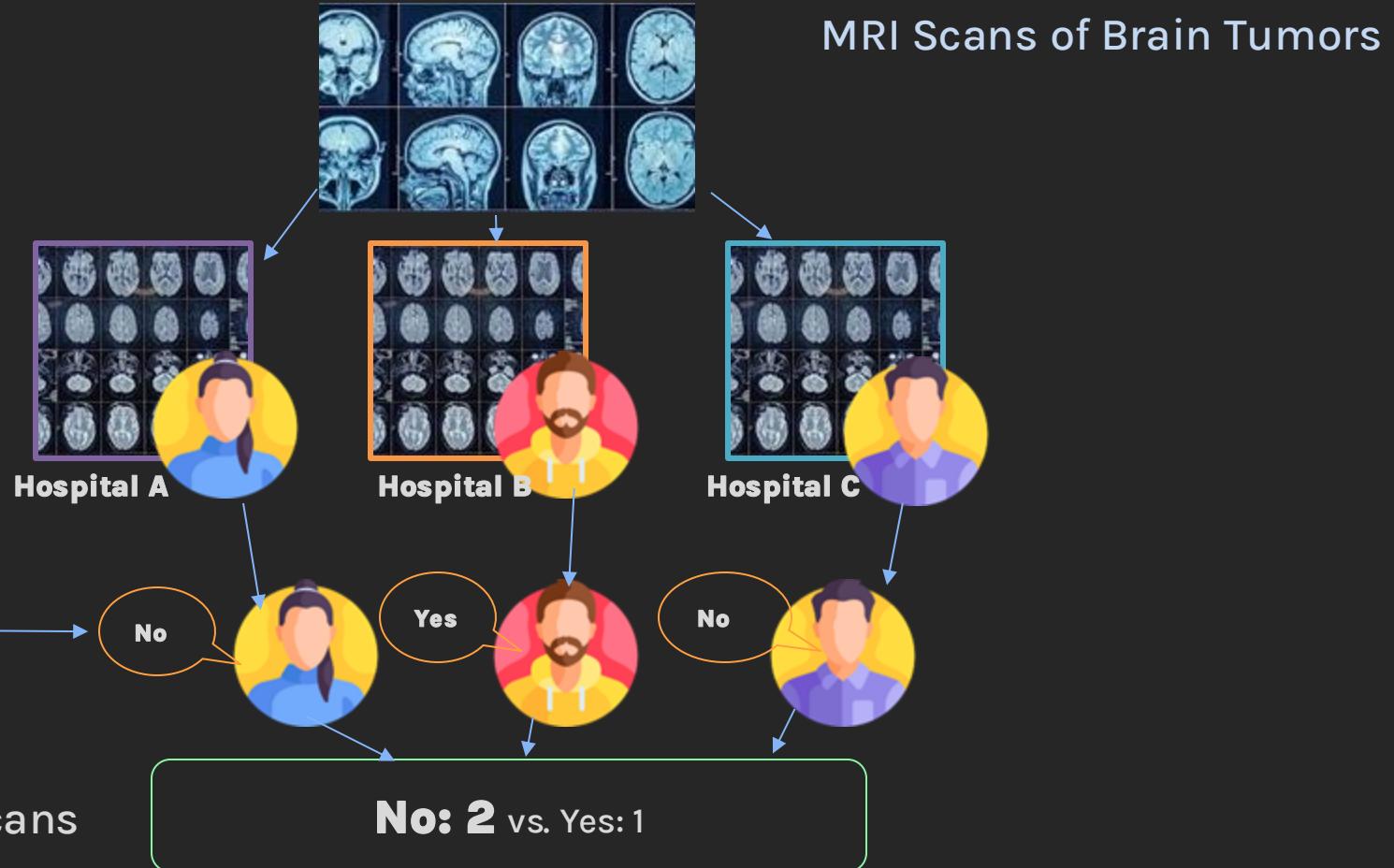
Every doctor is “trained” to identify tumors

Ensemble Learning - Intuition

Hoping to increase the likelihood of an accurate diagnosis, Mr. X sends his MRI scans to the 3 doctors.



Mr. X's MRI Scans



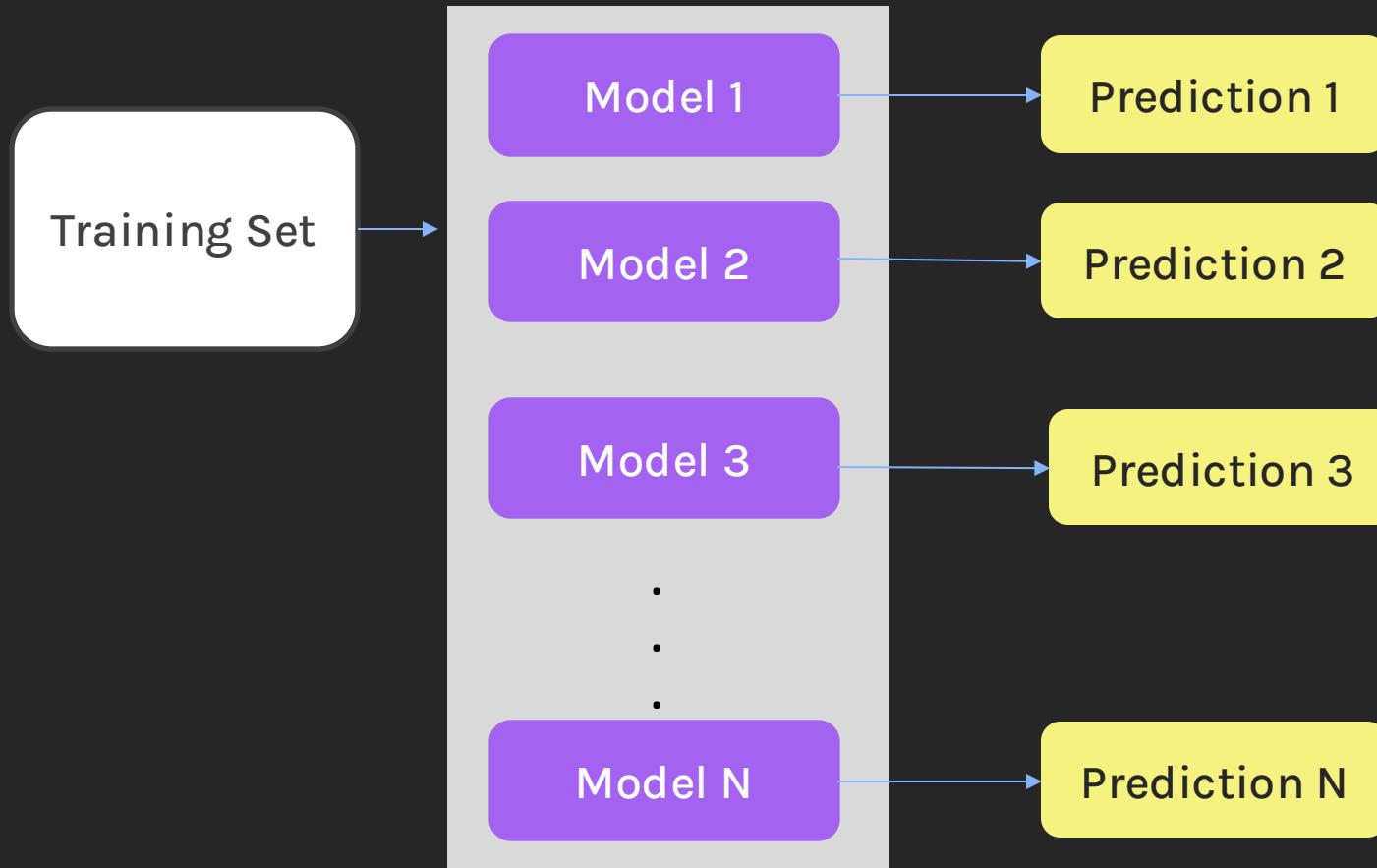
Ensemble Learning

Ensemble learning is a machine learning technique that combines several base models to produce one single optimal predictive model.

Training Set

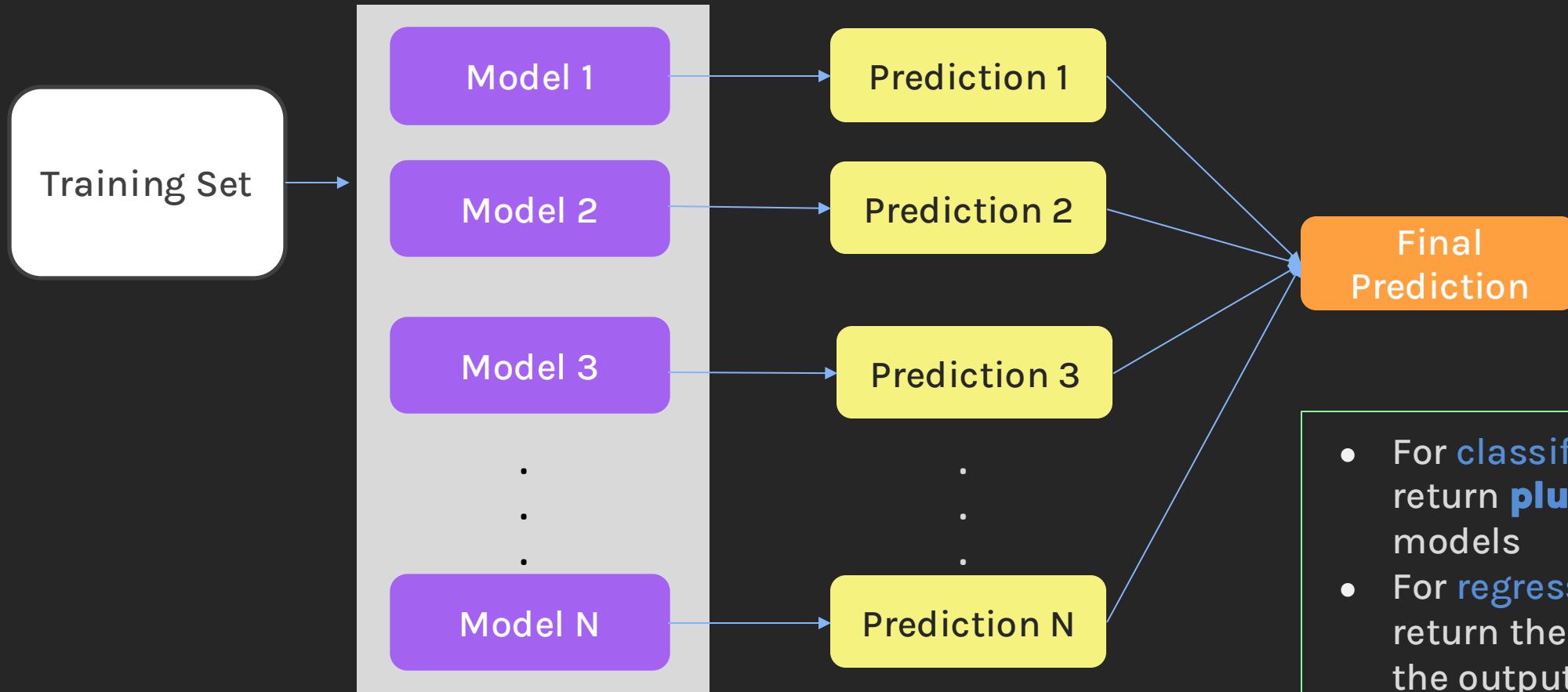
Ensemble Learning

Ensemble learning is a machine learning technique that combines several base models to produce one single optimal predictive model.



Ensemble Learning

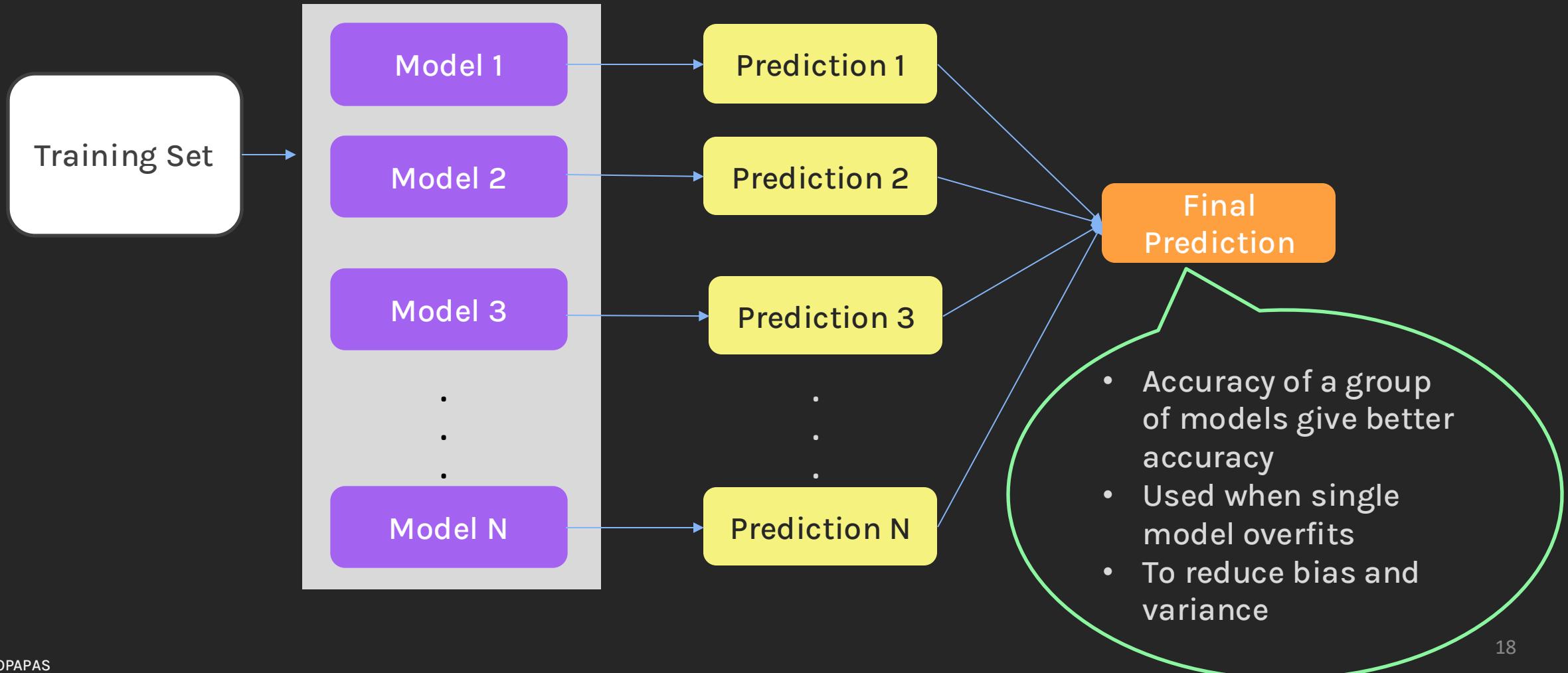
Ensemble learning is a machine learning technique that combines several base models to produce one single optimal predictive model.



- For **classification**, we return **plurality** of the models
- For **regression**, we return the **average** of the outputs from the models

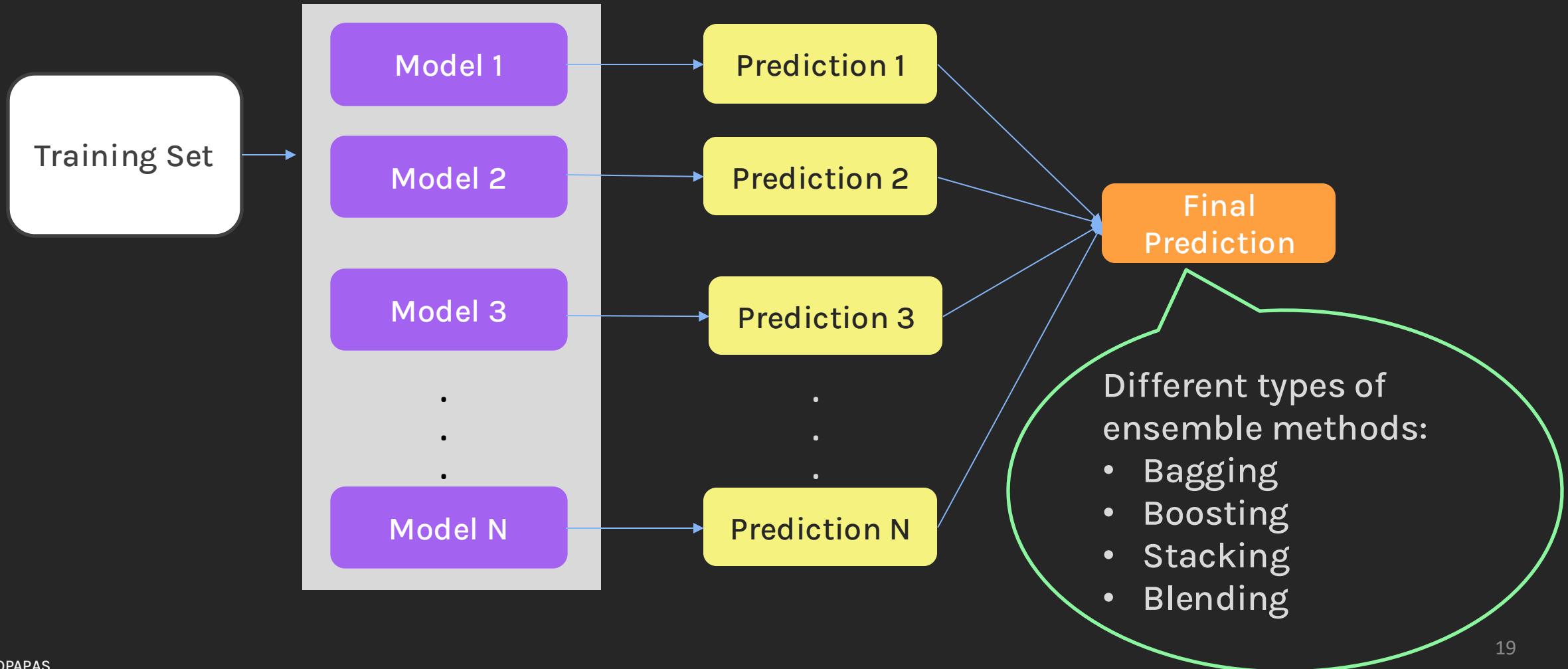
Ensemble Learning

Ensemble learning is a machine learning technique that combines several base models to produce one single optimal predictive model.



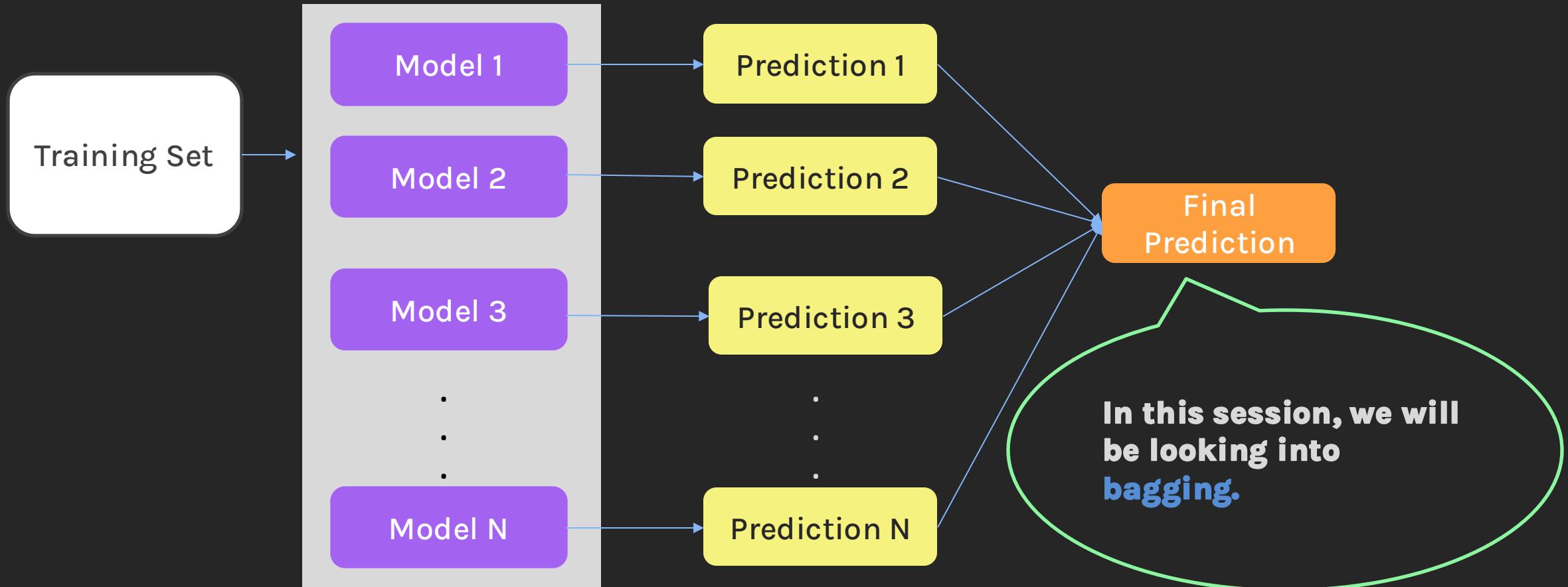
Ensemble Learning

Ensemble learning is a machine learning technique that combines several base models to produce one single optimal predictive model.



Ensemble Learning

Ensemble learning is a machine learning technique that combines several base models to produce one single optimal predictive model.



Ensemble Learning - Intuition

MRI Scans of
Brain Tumor
Patients



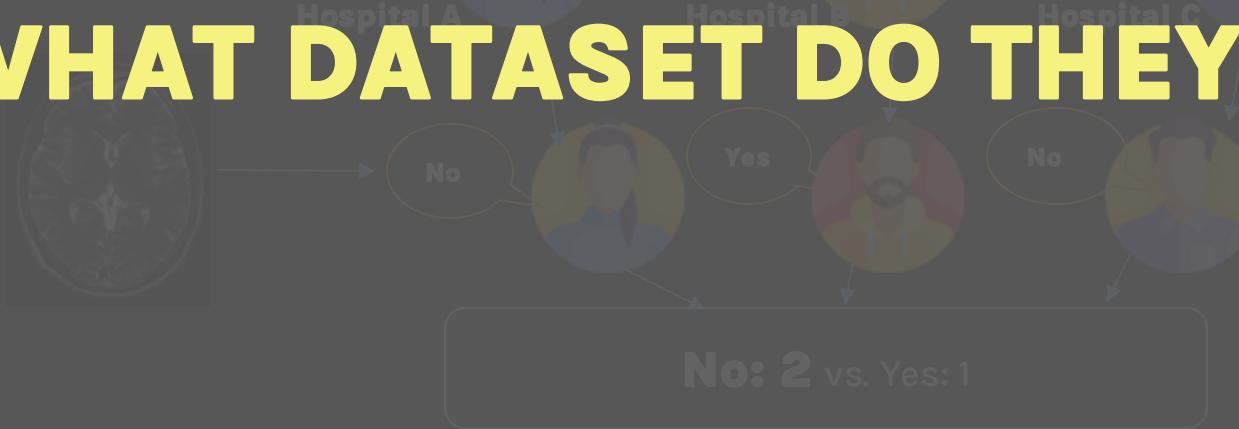
This is the intuition behind **ensemble method**, a method of building a single model by training and aggregating multiple models.

Doctors trained on samples of Brain tumor MRI Scans in the educational institutions they studied in.

Mr. X sends his MRI Scan forward to get the diagnosis from 3 doctors just to make sure that the results are more confident.

BUT WAIT, HOW DO THESE DOCTORS OR MODELS LEARN?

WHAT DATASET DO THEY SEE?



Bootstrap - Motivation

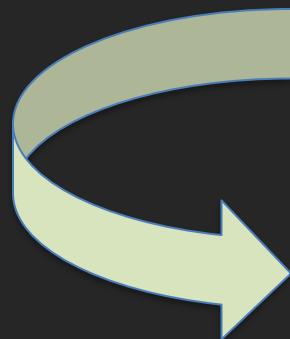
In practice, we do not have different datasets or different doctors.

We want multiple models (doctors) to train on different datasets.

However, we have only one dataset.

How can we generate more datasets?

To address the data scarcity for model fitting, we generate new datasets using ...



Bootstrapping

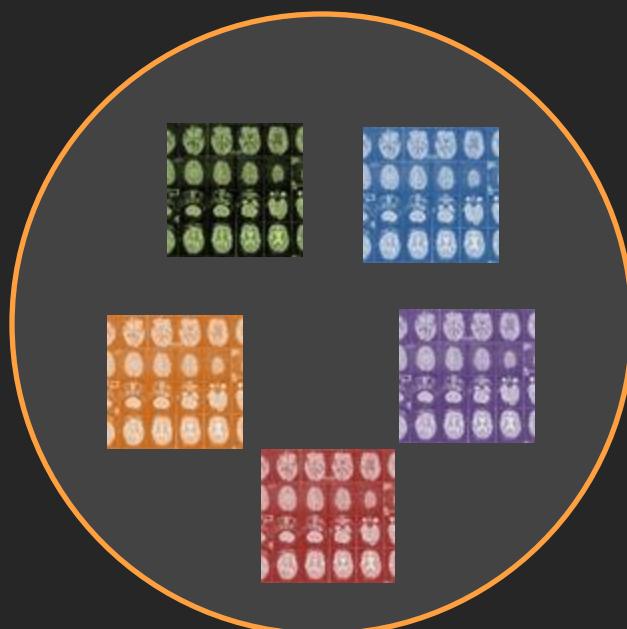
Bootstrapped datasets!

**But what is
Bootstrapping
?**

Bootstrapping

Bootstrapping is the process of **sampling with replacement** from a dataset and performing calculations on such multiple bootstrapped datasets to get an overall aggregate inference.

Dataset consisting of the MRI scans of 5 patients

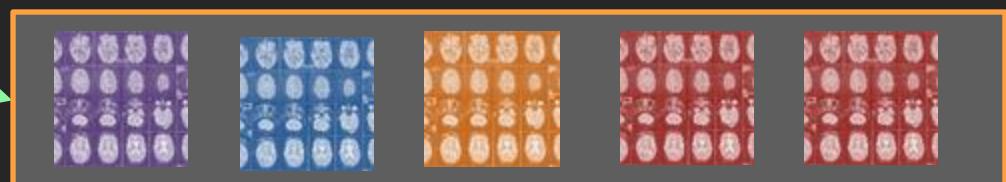
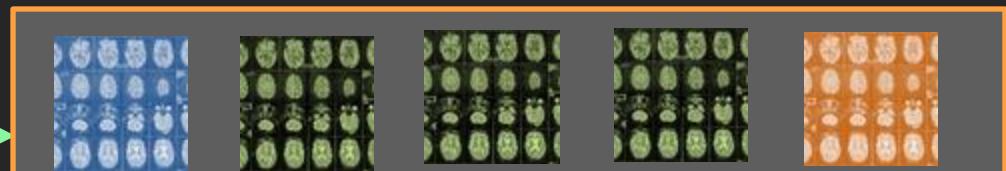
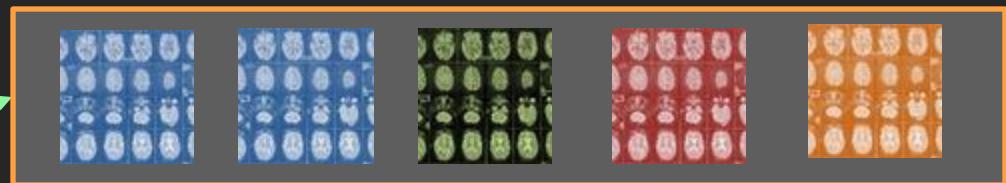


Possible Bootstrapped datasets

Randomly sample with replacement

Randomly sample with replacement

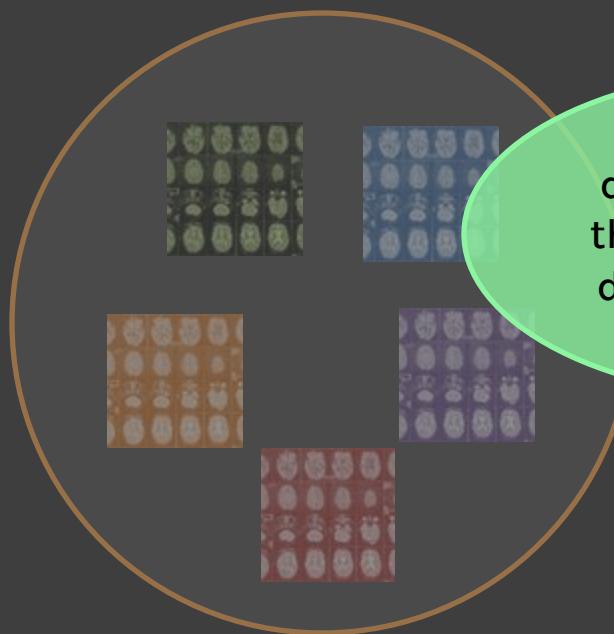
Randomly sample with replacement



Bootstrapping [TWO KEY IDEAS WE WANT TO EMPHASIZE)

Bootstrapping is the process of **sampling with replacement** from a dataset and performing calculations on such multiple bootstrapped datasets to obtain overall aggregate inference.

Dataset consisting of the MRI scans of 5 patients



The bootstrapped dataset consists of the same amount of data as the original dataset.

Randomly sample with replacement

Possible Bootstrapped datasets



Randomly choosing data and allowing for duplication is called
Sampling with replacement!

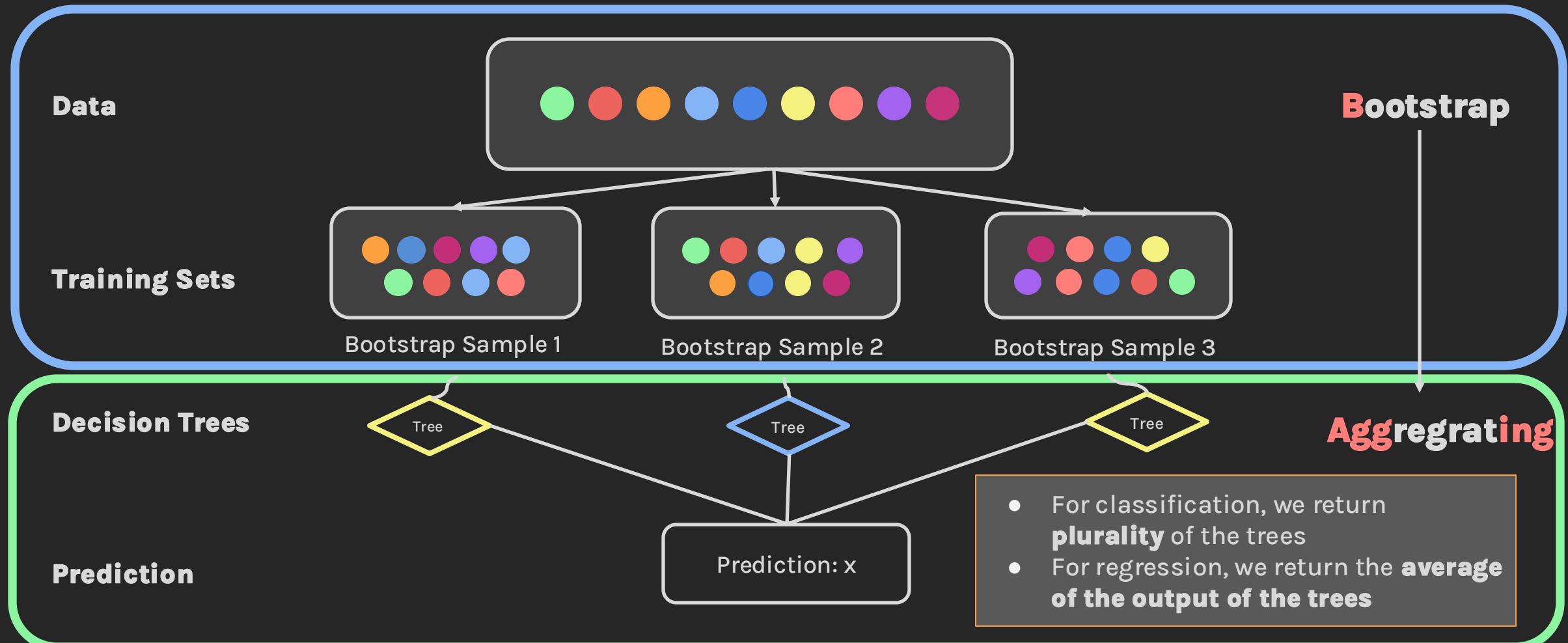
**Combining Ensemble and
Bootstrapping together ...**

Bootstrap + **aggregating**

Bagging

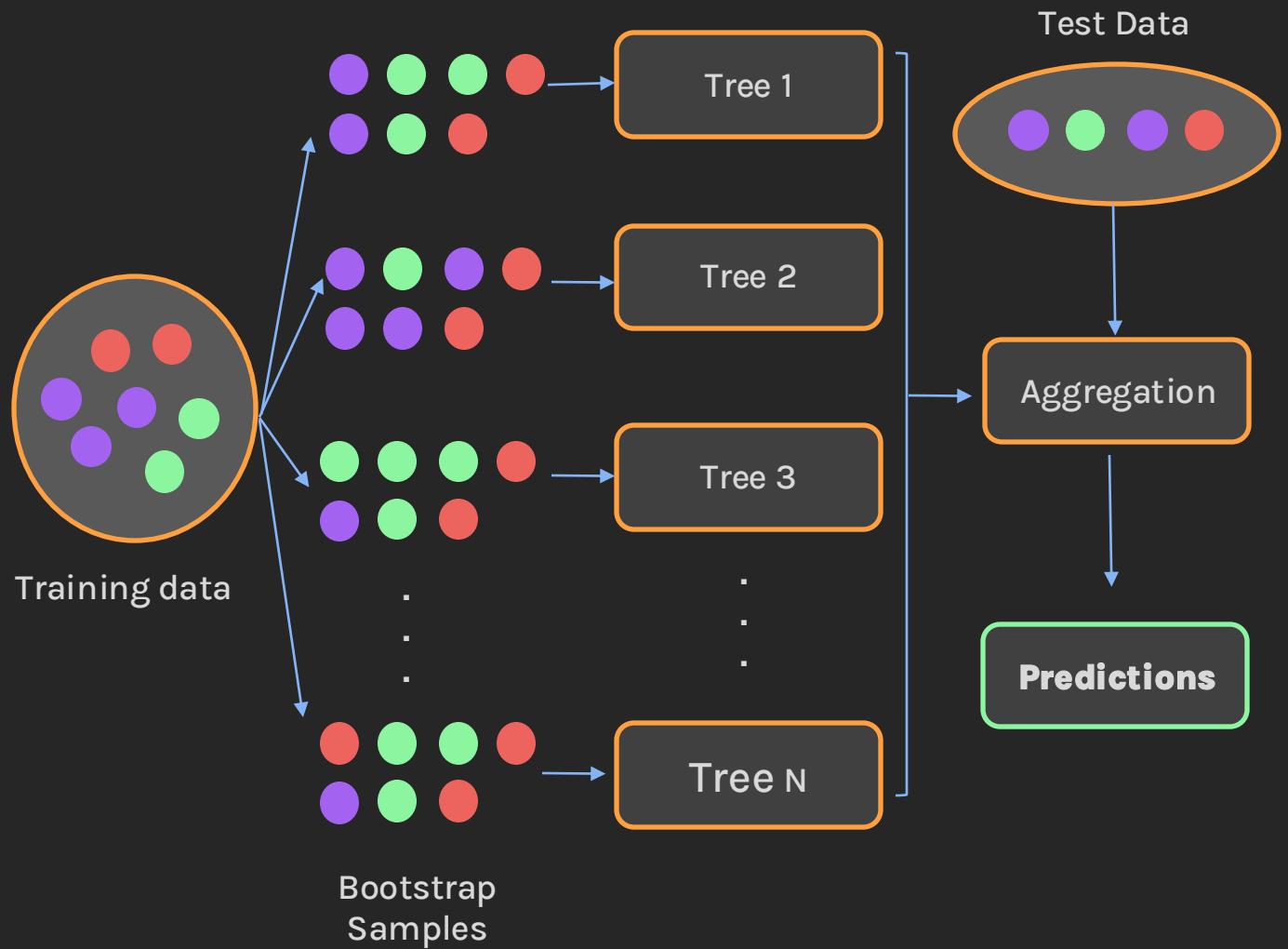
We get Bagging!

This method is called **Bagging** (Breiman, 1996), short for, of course, **Bootstrap Aggregating**.

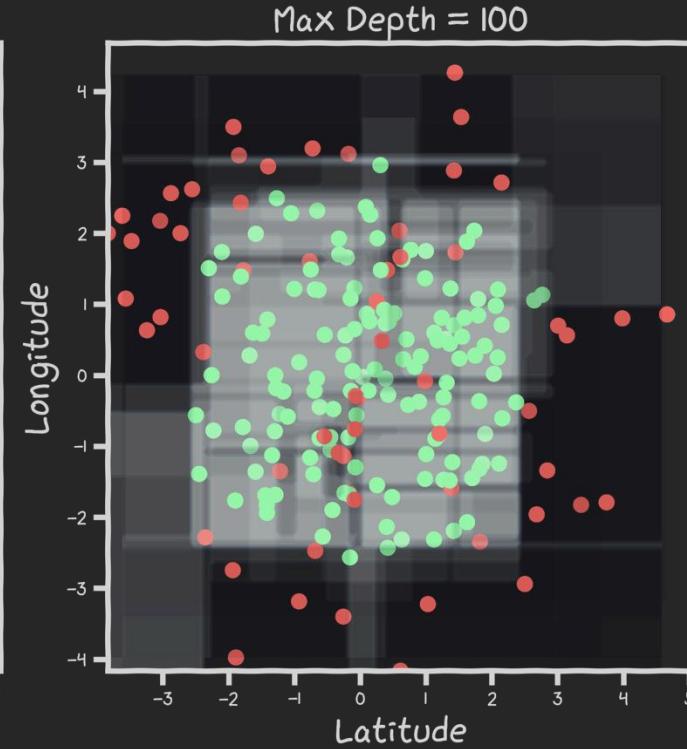
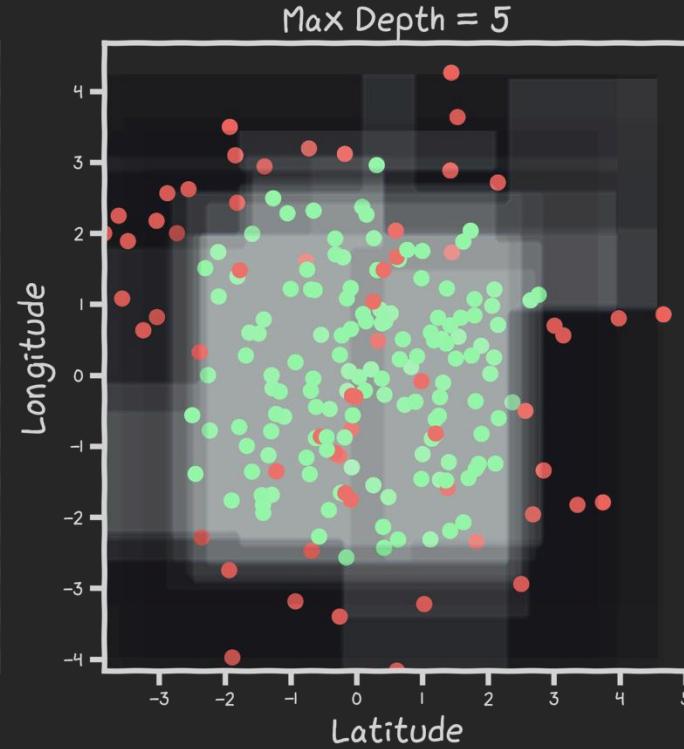
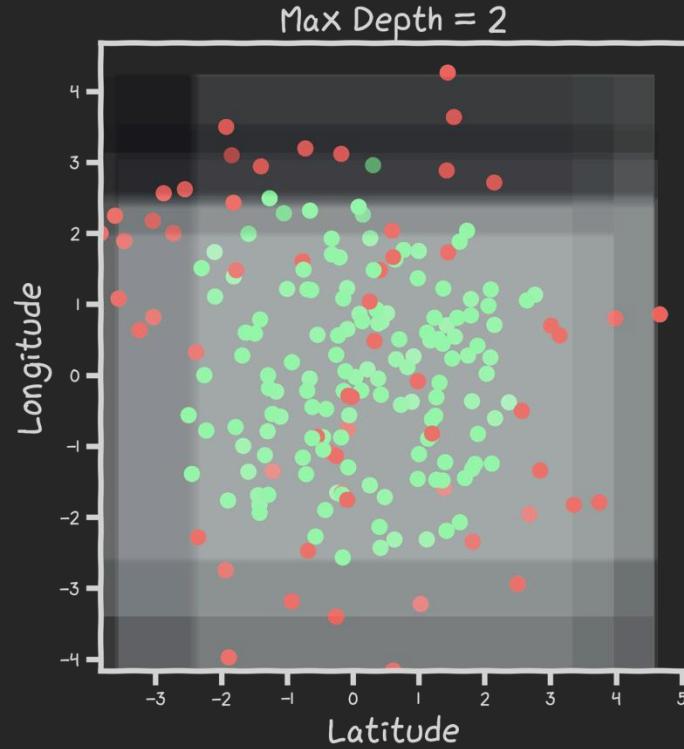


Bagging - Bootstrap + Aggregating

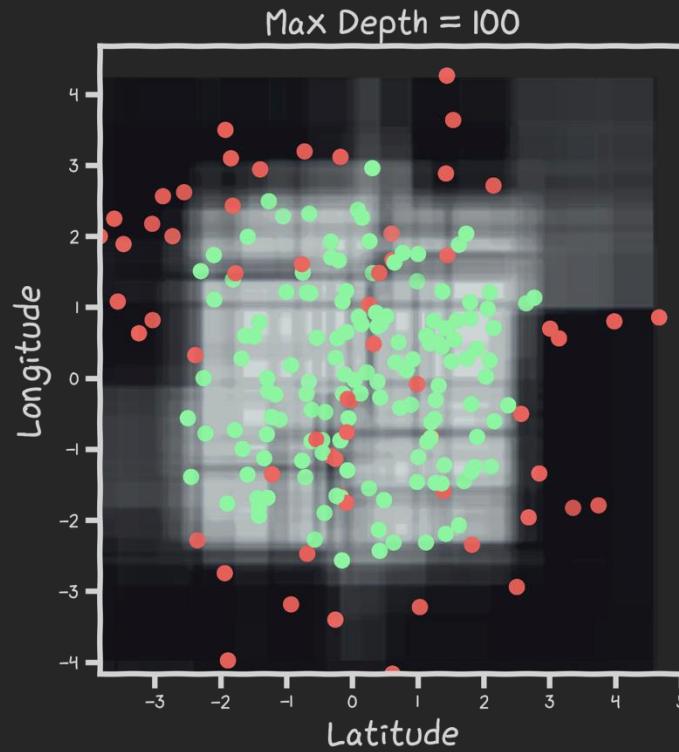
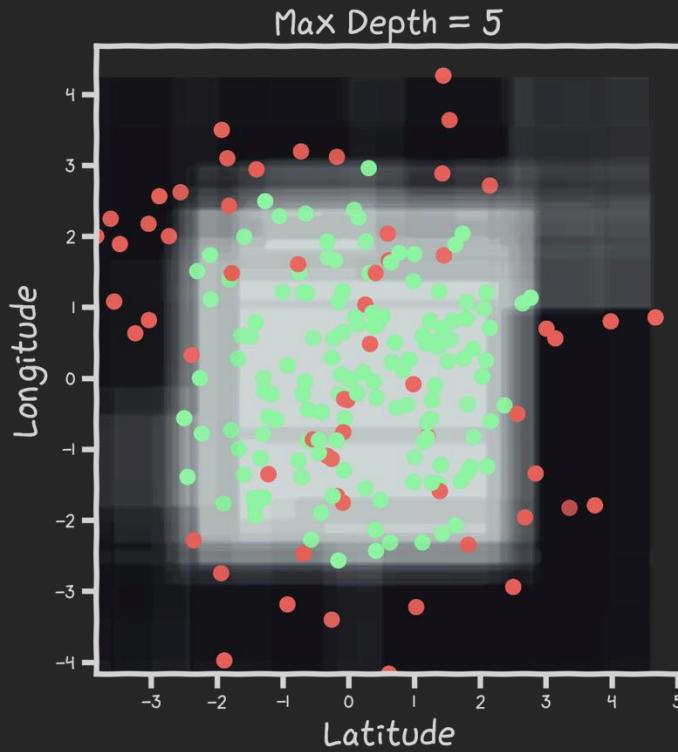
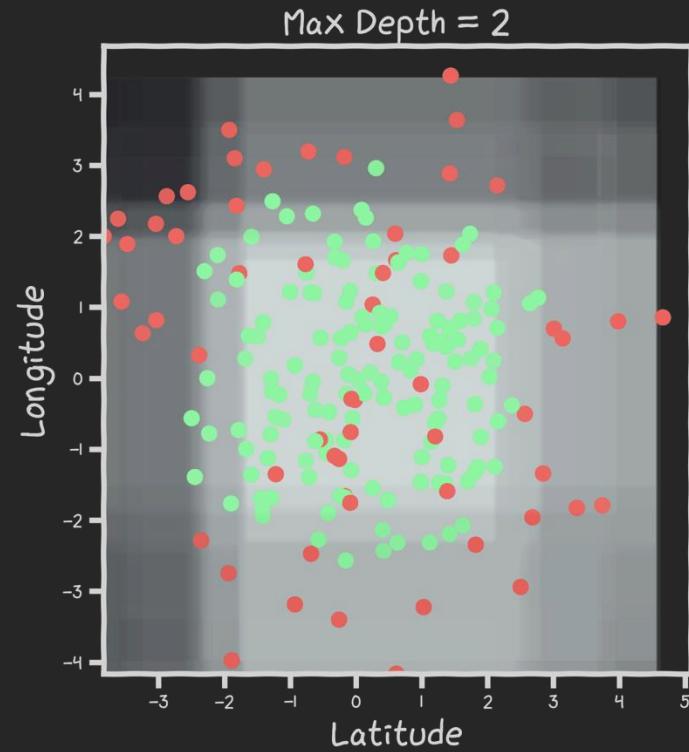
1. **Bootstrap:** we generate multiple samples of training data, via bootstrapping. We train a deep decision tree on each sample of data.
2. **Aggregating:** for a given input, we output the averaged outputs of all the models for that input.



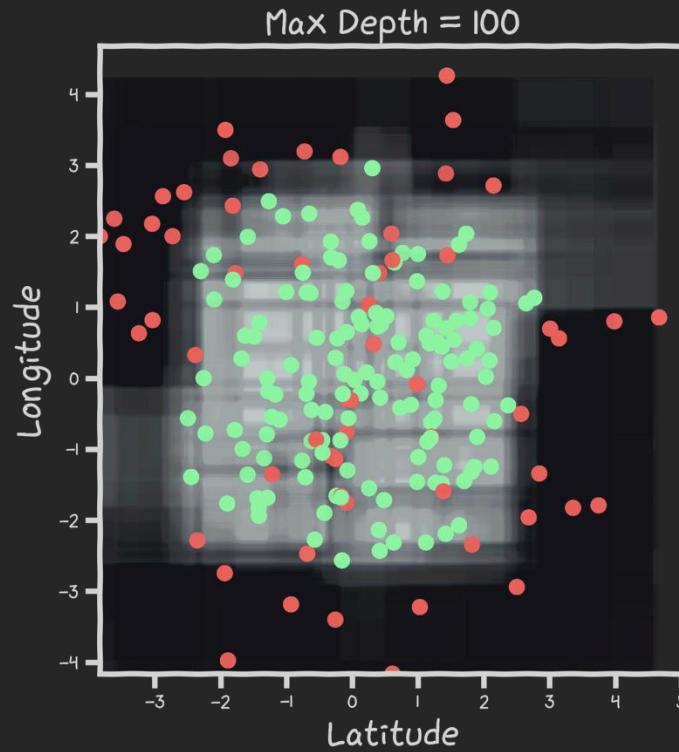
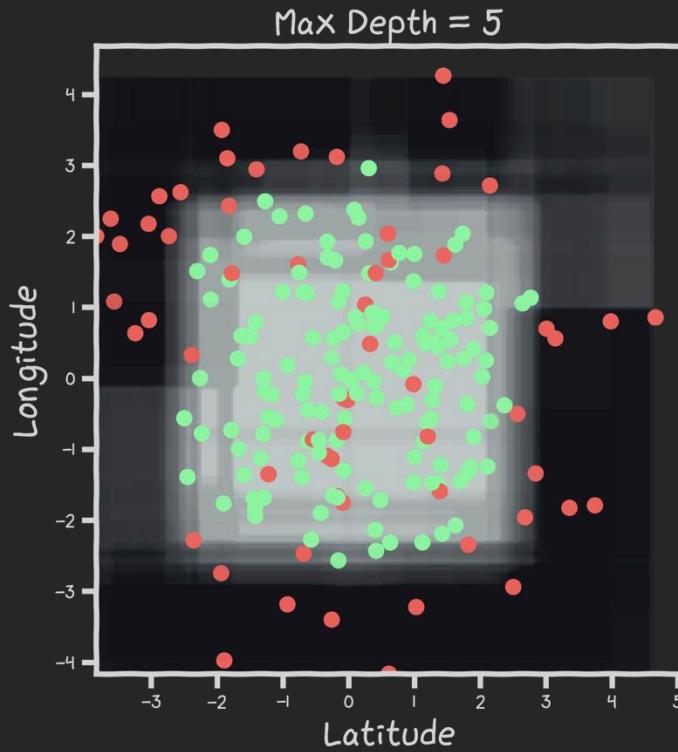
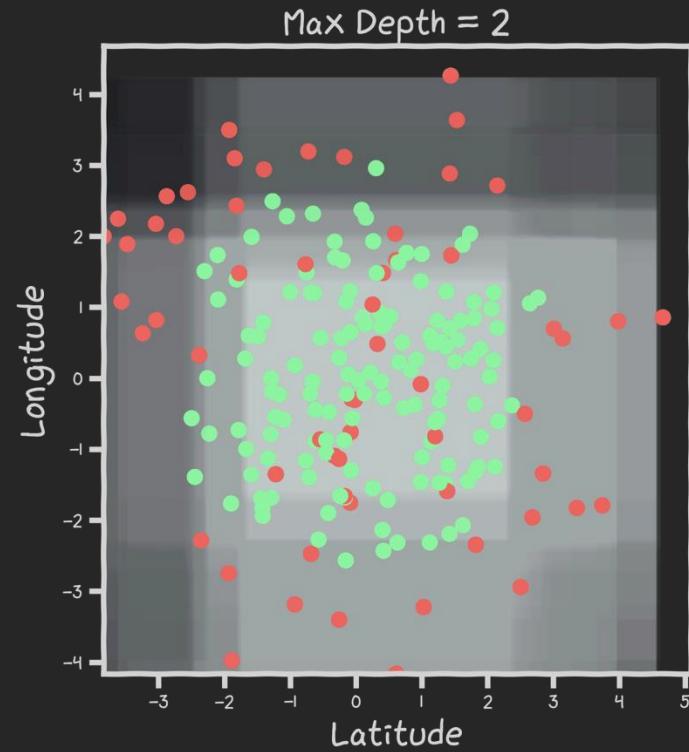
Examples of tree of various depth for 10 bootstrapped samples



Examples of tree of various depth for 100 bootstrapped samples



Examples of tree of various depth for 150 bootstrapped samples

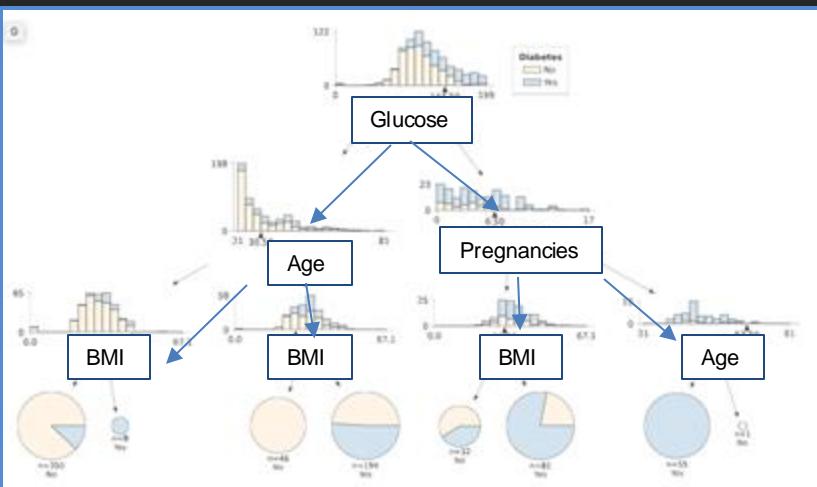
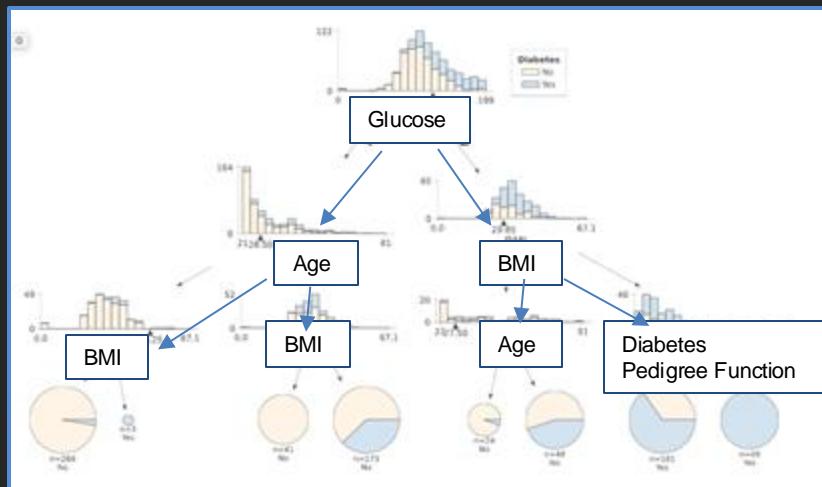
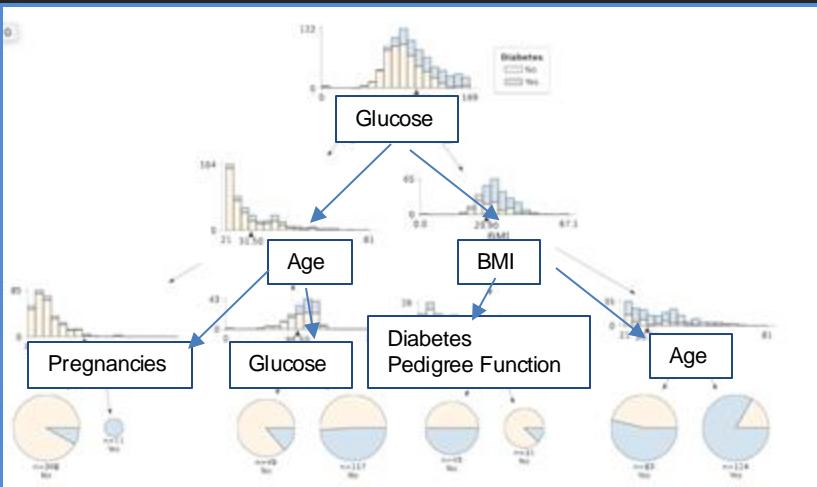
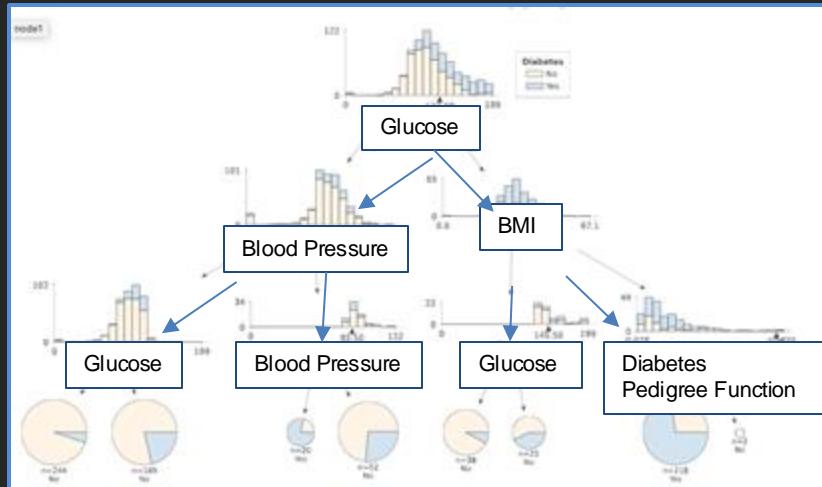


Advantages of Bagging

Bagging enjoys the benefits of:

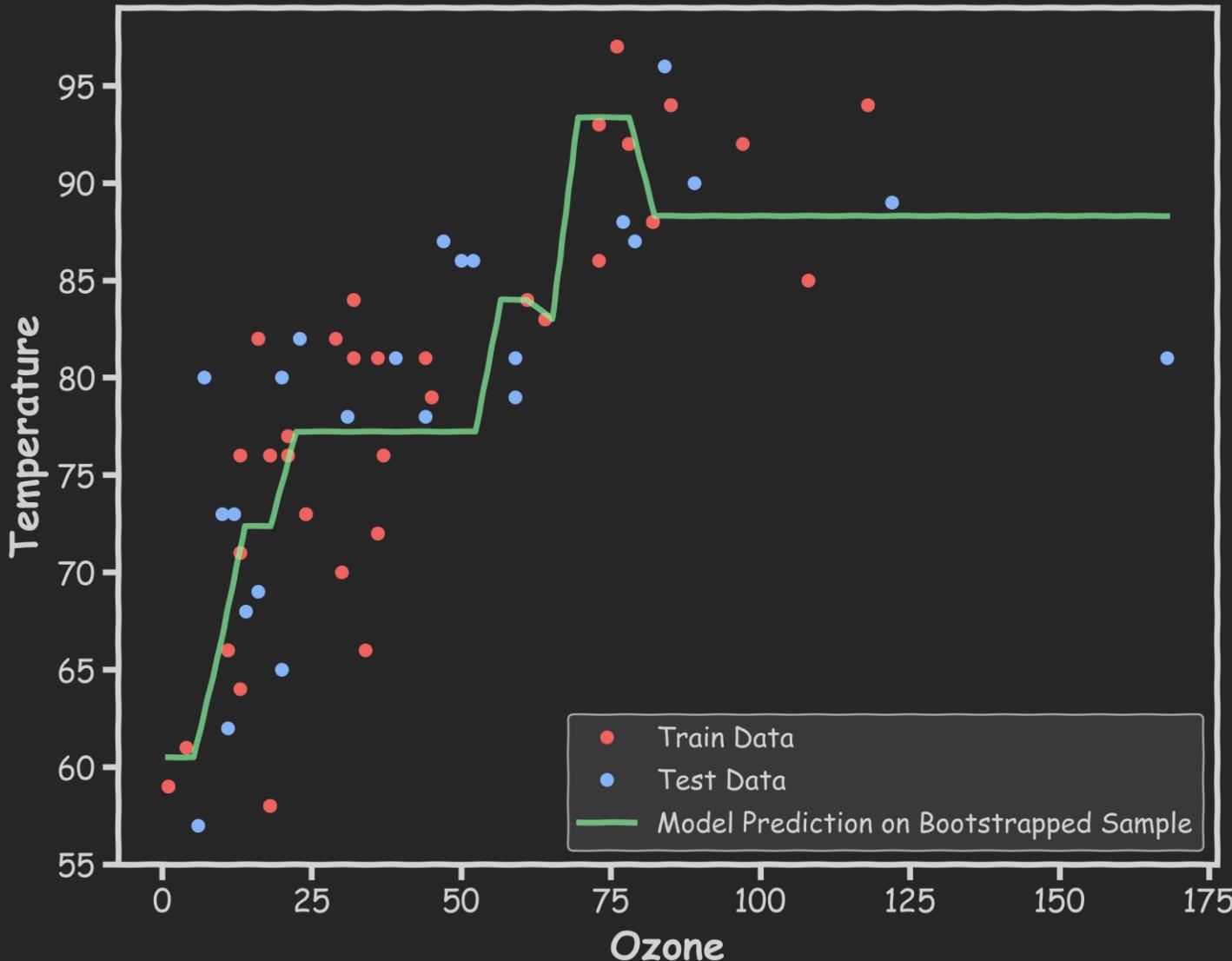
1. **High expressiveness** - by using deeper trees each model is able to approximate complex functions and decision boundaries.
2. **Low variance** - averaging the prediction of all the models reduces the variance in the final prediction, assuming that we choose a sufficiently large number of trees.

Classification in Bagging



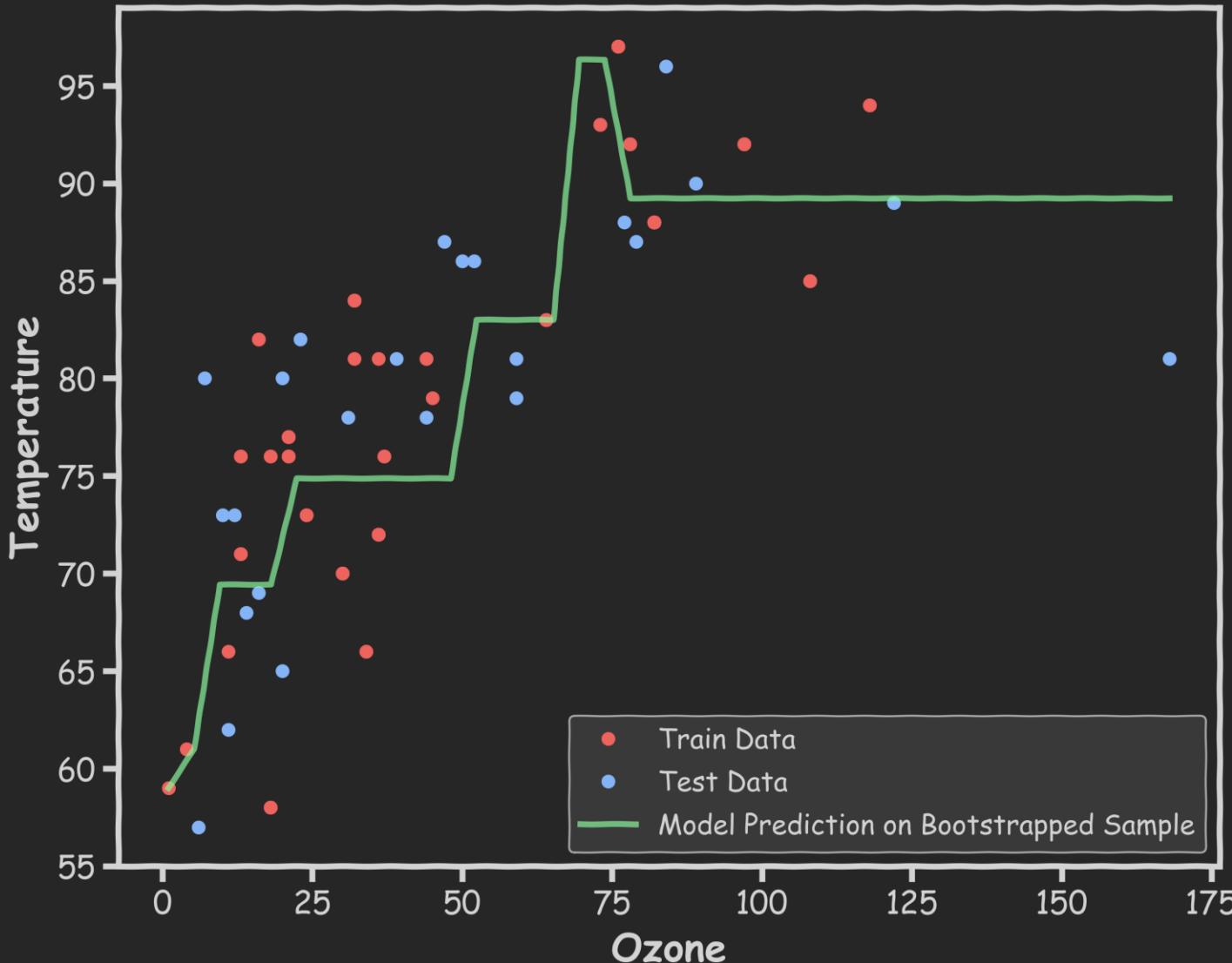
For each bootstrap, we build a decision tree. The results is a combination (majority) of the predictions from all trees.

Prediction from Decision Tree #1 with Bootstrapped Sample #1



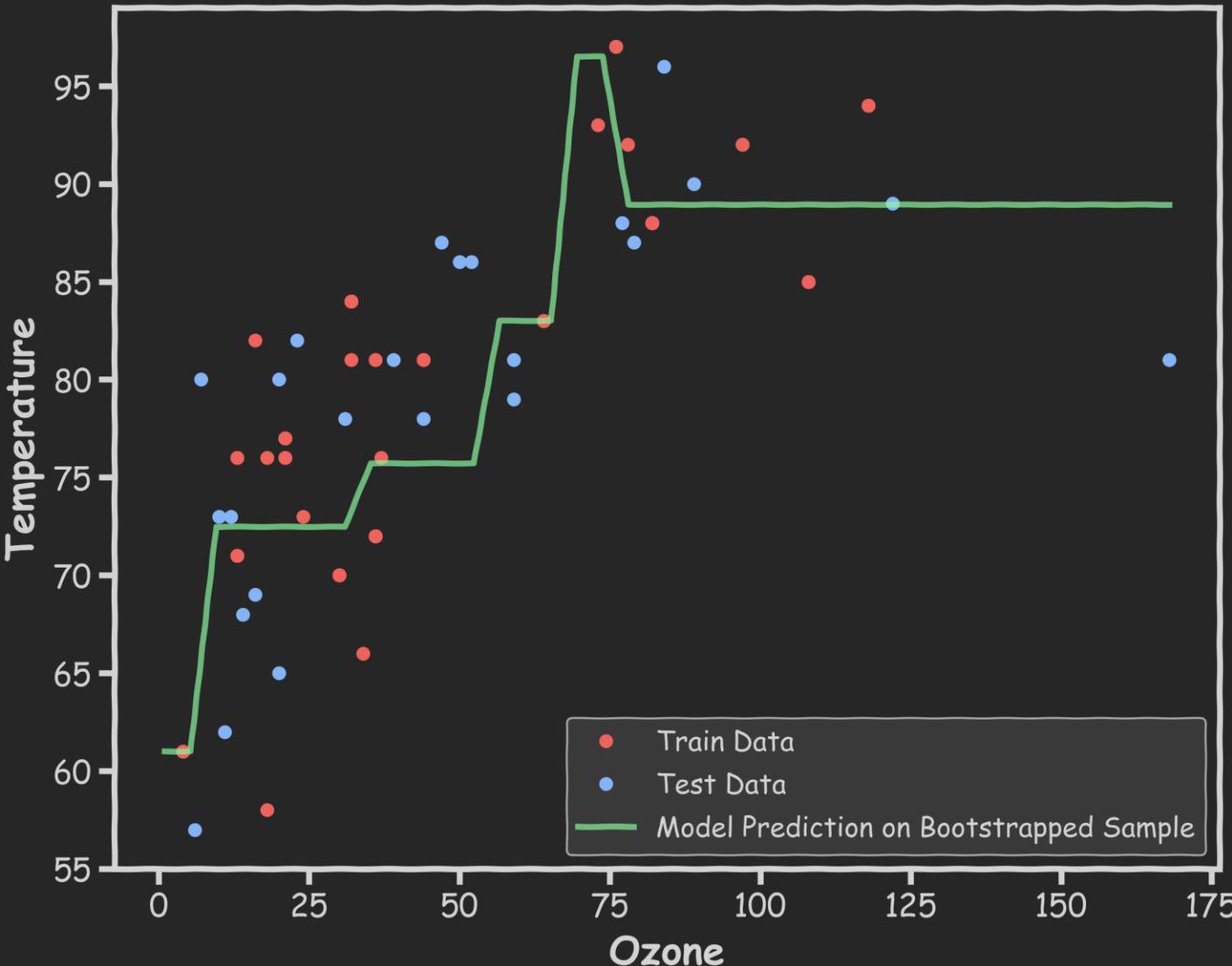
Model prediction on the same test set when fit on one version of the bootstrapped train data

Prediction from Decision Tree #2 with Bootstrapped Sample #2



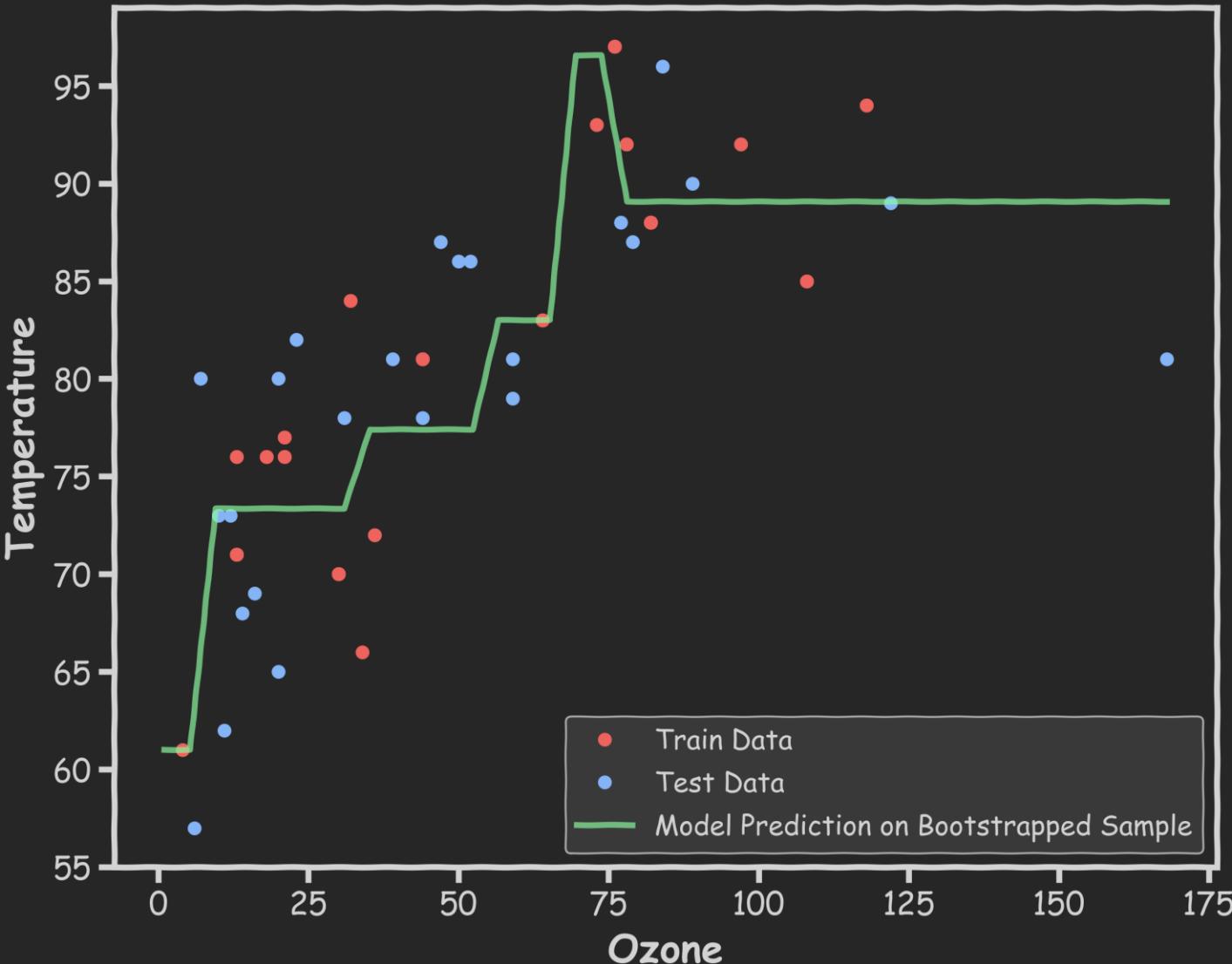
Model prediction on the same test set when fit on one version of the bootstrapped train data

Prediction from Decision Tree #3 with Bootstrapped Sample #3



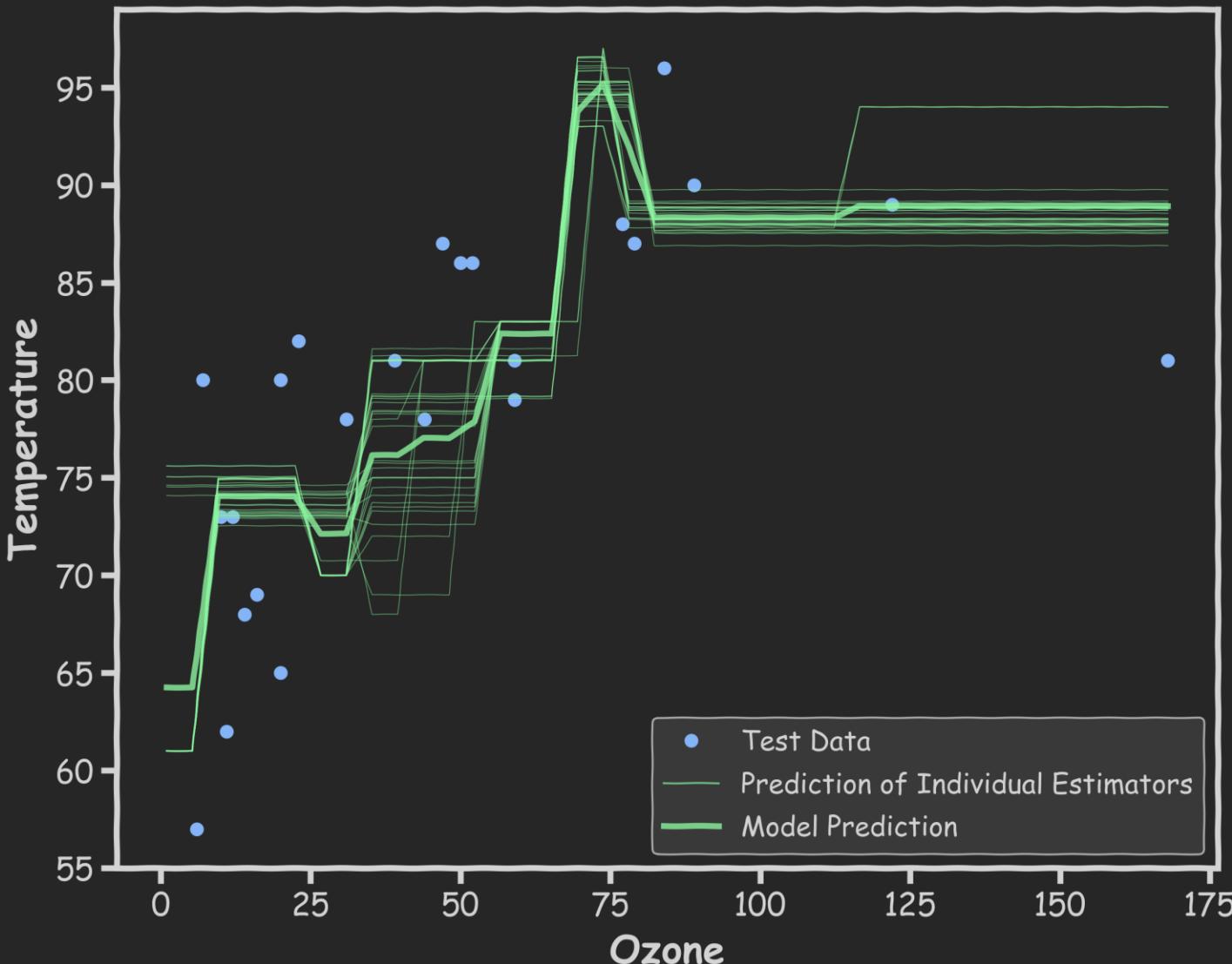
Model prediction on the same test set when fit on one version of the bootstrapped train data

Prediction from Decision Tree #4 with Bootstrapped Sample #4



Model prediction on the same test set when fit on one version of the bootstrapped train data

Prediction from all Decision Trees



The prediction by the bagging regressor model is the **average** of all the individual predictions of the trees.

Drawbacks of Bagging

Interpretability -

A **major drawback** of bagging (and other *ensemble methods* that we will study) is that the averaged model is no longer easily interpretable - i.e. one can no longer trace the ‘logic’ of an output through a series of decisions based on predictor values!

Underfitting and overfitting -

Drawbacks of Bagging

Interpretability -

A **major drawback** of bagging (and other *ensemble methods* that we will study) is that the averaged model is no longer easily interpretable - i.e. one can no longer trace the ‘logic’ of an output through a series of decisions based on predictor values!

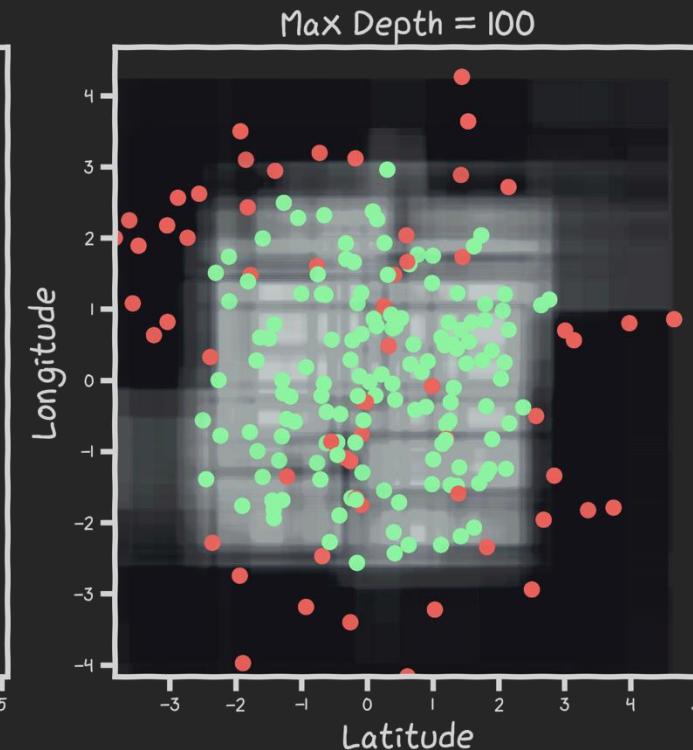
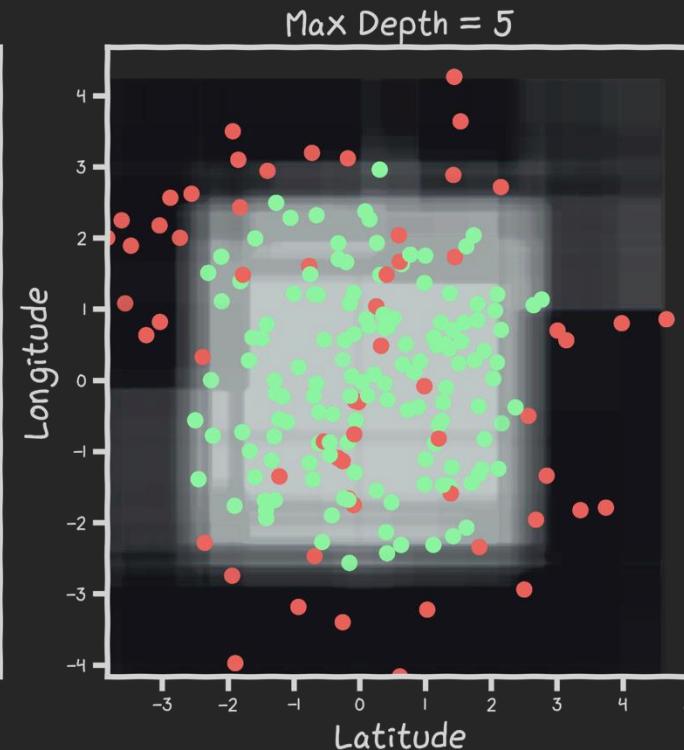
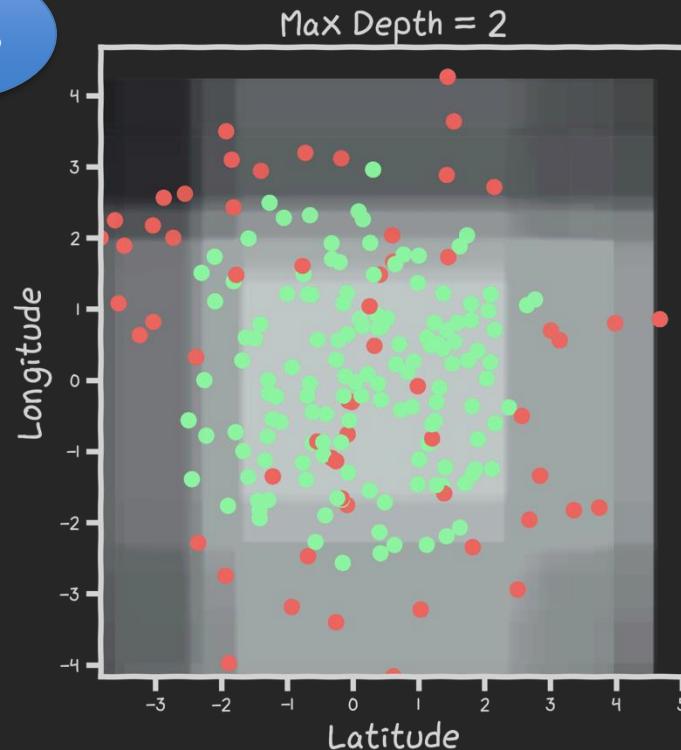
This interpretability challenge will be addressed in our upcoming lecture on random forests, where we will introduce methods like **MDI** (Mean Decrease Impurity) and **permutation importance** to provide insights into model decisions.

Underfitting & Overfitting

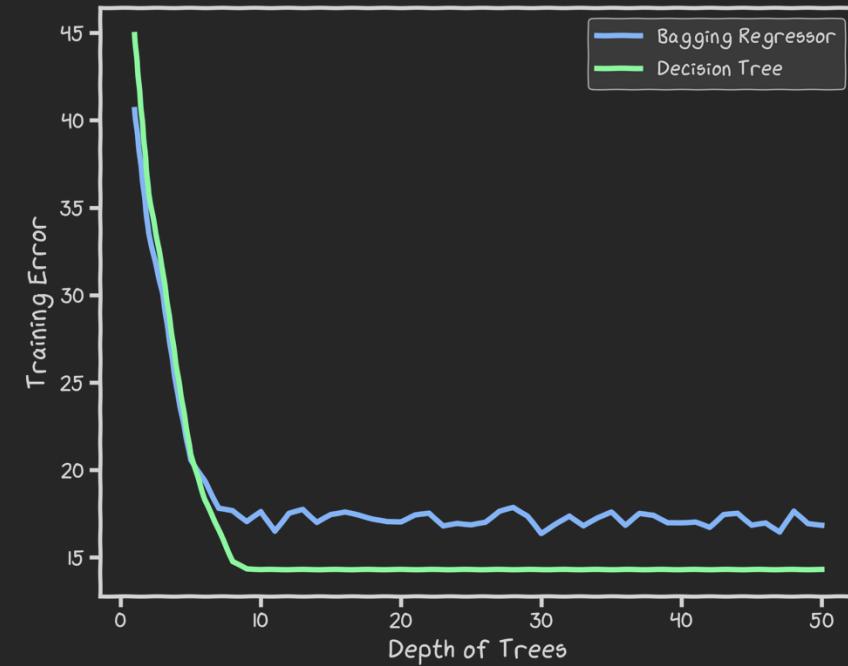
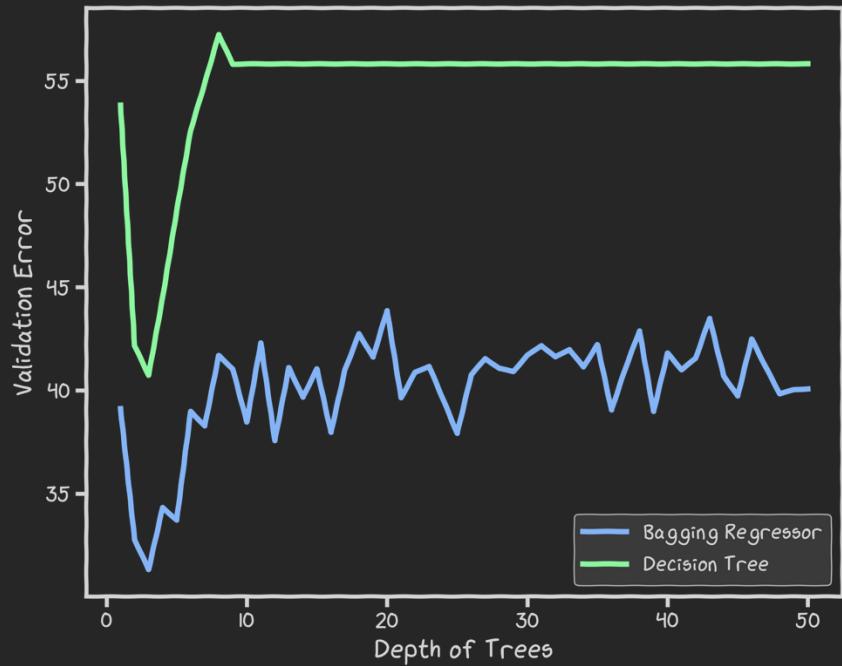
Here we fit 100 trees using bootstrapped samples. Even with multiple estimators, the shallow tree will not be able to capture the real pattern.

Number of Bootstraps = 100

Trees



Underfitting and Overfitting in Decision Tree vs. Bagging



- The graphs shows that the **more depth we add to Decision Trees, the faster the model overfits**
- However, for Bagging, there is still some form of **better model performance maintained after increase of depth of trees** although eventually it does **lead to no improvement of performance after a certain point as well**

Cases of underfitting and overfitting in Bagging

Cross validation is cool but computationally expensive and requires a larger dataset.

Before we conclude the bagging session, we will present another method of measuring the performance of ensemble methods.

Out-of-bag error

Summary

What are the limitations of single decision tree models, and how does bagging address these limitations?

Single decision tree models, particularly deep ones, are prone to overfitting and may have high variance. Bagging addresses these limitations by creating an ensemble of multiple decision trees trained on different bootstrap samples of the data. By averaging the predictions of these trees, bagging reduces variance and mitigates overfitting.

Explain the concept of bootstrapping and its significance in bagging.

Bootstrapping is a resampling technique where multiple datasets are created by randomly sampling with replacement from the original data. In bagging, bootstrapping is used to generate diverse training sets for each decision tree in the ensemble.

How does bagging leverage ensemble learning to improve prediction accuracy?

Bagging utilizes ensemble learning by combining the predictions of multiple decision trees trained on different bootstrap samples. This approach helps to mitigate individual tree errors and biases, as the final prediction is based on the consensus or average of multiple models. This aggregation process improves overall prediction accuracy.

Summary

Describe the aggregation process in bagging for both classification and regression tasks.

For classification tasks, the aggregation process in bagging involves majority voting. Each tree in the ensemble makes a prediction, and the class with the most votes across all trees is selected as the final prediction. In regression tasks, the aggregation process typically involves averaging the predictions of all trees to obtain the final prediction.

What are the advantages of bagging over single decision tree models?

Bagging offers several advantages over single decision tree models: increased prediction accuracy by reducing variance and overfitting, improved generalization ability, and robustness to noisy data.

Explain the main drawback of bagging in terms of model interpretability.

A major drawback of bagging is the loss of interpretability compared to single decision trees. While decision trees are inherently interpretable, the aggregation process in bagging creates a complex model where the reasoning behind predictions is not easily traceable. This makes it challenging to understand the underlying relationships between variables and predictions.

Summary

Can bagging lead to underfitting? If so, explain why and provide an example.

Yes, bagging can lead to underfitting if the individual decision trees are too shallow. If the trees are not complex enough to capture the underlying patterns in the data, the bagged model may not be able to adequately represent the true relationship between variables, resulting in underfitting. For example, if we are trying to model a complex non-linear relationship with very shallow decision trees, even with many trees, bagging might still underfit the data.

Describe the relationship between the depth of individual trees in a bagged model and the model's tendency to overfit or underfit.

The depth of individual trees in a bagged model plays a significant role in its tendency to overfit or underfit. Deeper trees have higher complexity and can capture more intricate patterns in the data, but they are also more susceptible to overfitting. Shallow trees, on the other hand, have lower complexity and are less prone to overfitting, but they may underfit the data if the true relationship is complex. The optimal tree depth for a bagged model depends on the specific dataset and the balance between bias and variance.



Bagging OOB Error



CS109A Introduction to Data Science
Pavlos Protopapas, Kevin Rader and Chris Gumb

Wenjun Liu
Yosemite

Outline

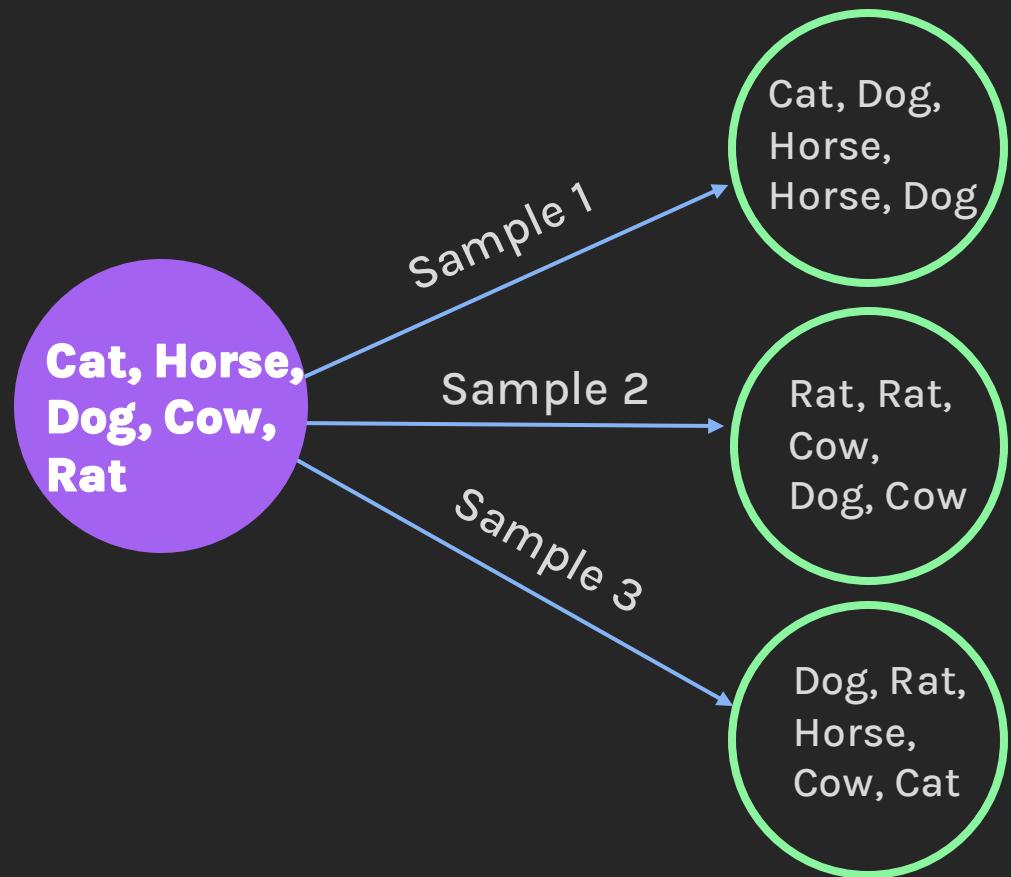
- Motivation
- Bagging
- **Out-of-bag Error**

What is OOB?

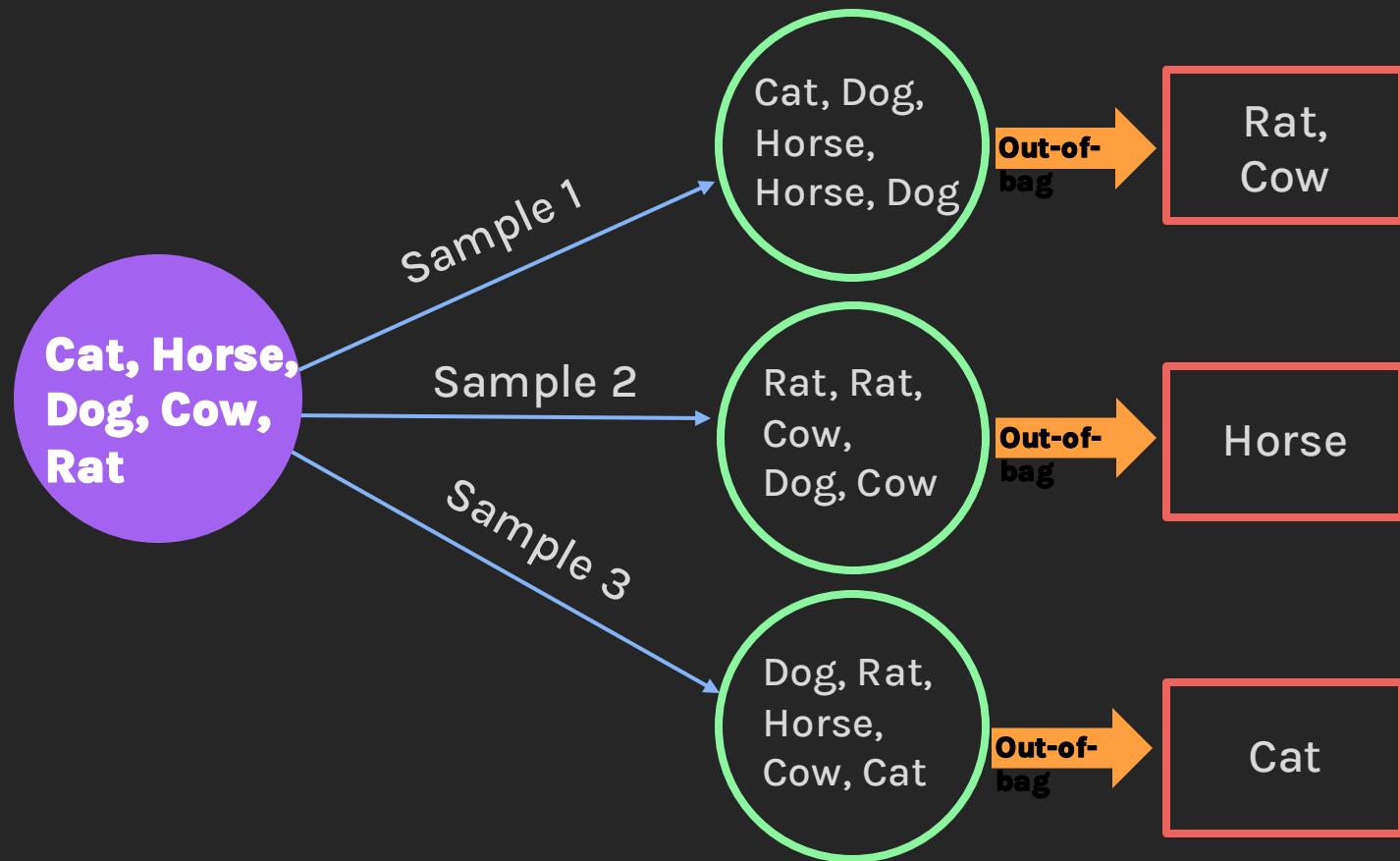


**Cat, Horse,
Dog, Cow,
Rat**

What is OOB?

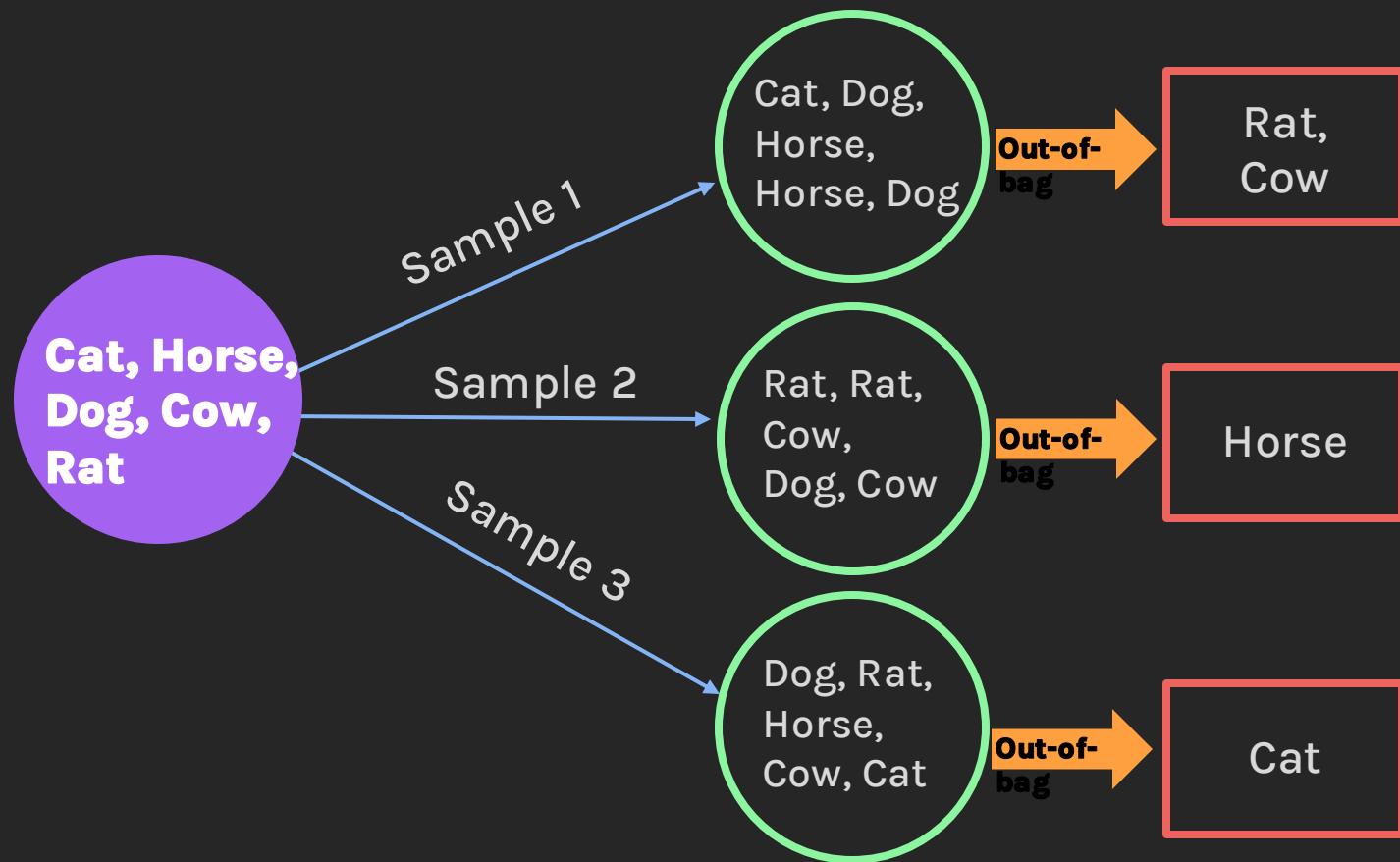


What is OOB?



Out-of-bag estimate is a method of determining the prediction error whilst being trained.

What is OOB?



Out-of-bag estimate is a method of determining the prediction error whilst being trained.

Why?

- To measure generalizability.
- To replace the need for a separate measurement of performance for a validation-set performance.

Let us explore this in more details with another example

Out-of-bag Error (OOB)

Original Data

| X | Y |
|-------|-------|
| x_1 | y_1 |
| x_2 | y_2 |
| x_3 | y_3 |
| x_4 | y_4 |
| x_5 | y_5 |
| • | • |
| • | • |
| • | • |
| x_n | y_n |

Response/Target

Predictor/Feature
PROTOPAPAS

Out-of-bag Error (OOB)

Original Data

| X | Y |
|-------|-------|
| X_1 | y_1 |
| X_2 | y_2 |
| X_3 | y_3 |
| X_4 | y_4 |
| X_5 | y_5 |
| • | • |
| • | • |
| • | • |
| X_n | y_n |

Bootstrap Sample

1

| X | Y |
|----------|----------|
| X_4 | y_4 |
| X_9 | y_9 |
| X_{11} | y_{11} |
| X_{21} | y_{21} |
| X_{35} | y_{35} |
| • | • |
| • | • |
| • | • |
| X_k | y_k |



Response/Target

Predictor/Feature

PROTOPAPAS

Out-of-bag Error (OOB)

Original Data

| X | Y |
|-------|-------|
| X_1 | y_1 |
| X_2 | y_2 |
| X_3 | y_3 |
| X_4 | y_4 |
| X_5 | y_5 |
| ... | ... |
| ... | ... |
| ... | ... |
| X_n | y_n |

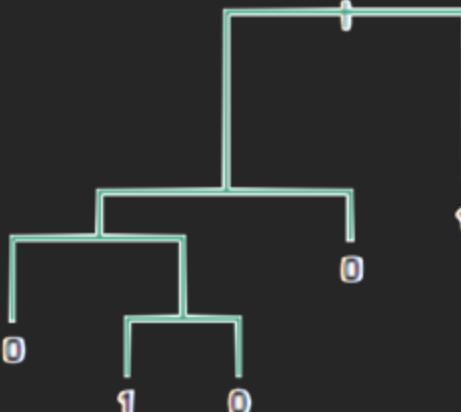
Bootstrap Sample

1

| X | Y |
|----------|----------|
| X_4 | y_4 |
| X_9 | y_9 |
| X_{11} | y_{11} |
| X_{21} | y_{21} |
| X_{35} | y_{35} |
| ... | ... |
| ... | ... |
| ... | ... |
| X_k | y_k |

Decision Tree

1



Response/Target

Out-of-bag Error (OOB)

Original Data

| X | Y |
|-------|-------|
| X_1 | y_1 |
| X_2 | y_2 |
| X_3 | y_3 |
| X_4 | y_4 |
| X_5 | y_5 |
| ... | ... |
| X_n | y_n |

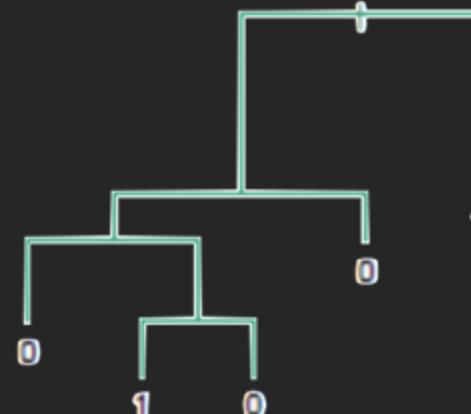
Bootstrap Sample

1

| X | Y |
|----------|----------|
| X_1 | y_1 |
| X_3 | y_3 |
| X_5 | y_5 |
| X_{21} | y_{21} |
| X_{35} | y_{35} |
| ... | ... |
| X_k | y_k |

Decision Tree

1



Used and unused data

| X | Y |
|-------|-------|
| X_1 | y_1 |
| X_2 | y_2 |
| X_3 | y_3 |
| X_4 | y_4 |
| X_5 | y_5 |
| ... | ... |
| X_n | y_n |

Response/Target

Out-of-bag Error (OOB)

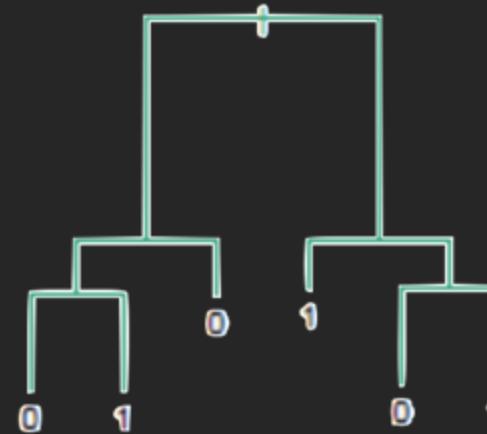
Original Data

| X | Y |
|-------|-------|
| X_1 | y_1 |
| X_2 | y_2 |
| X_3 | y_3 |
| X_4 | y_4 |
| X_5 | y_5 |
| • | • |
| • | • |
| • | • |
| X_n | y_n |

Bootstrap Sample
2

| X | Y |
|----------|----------|
| X_5 | y_5 |
| X_7 | y_7 |
| X_{13} | y_{13} |
| X_{27} | y_{27} |
| X_{32} | y_{32} |
| • | • |
| • | • |
| • | • |
| X_k | y_k |

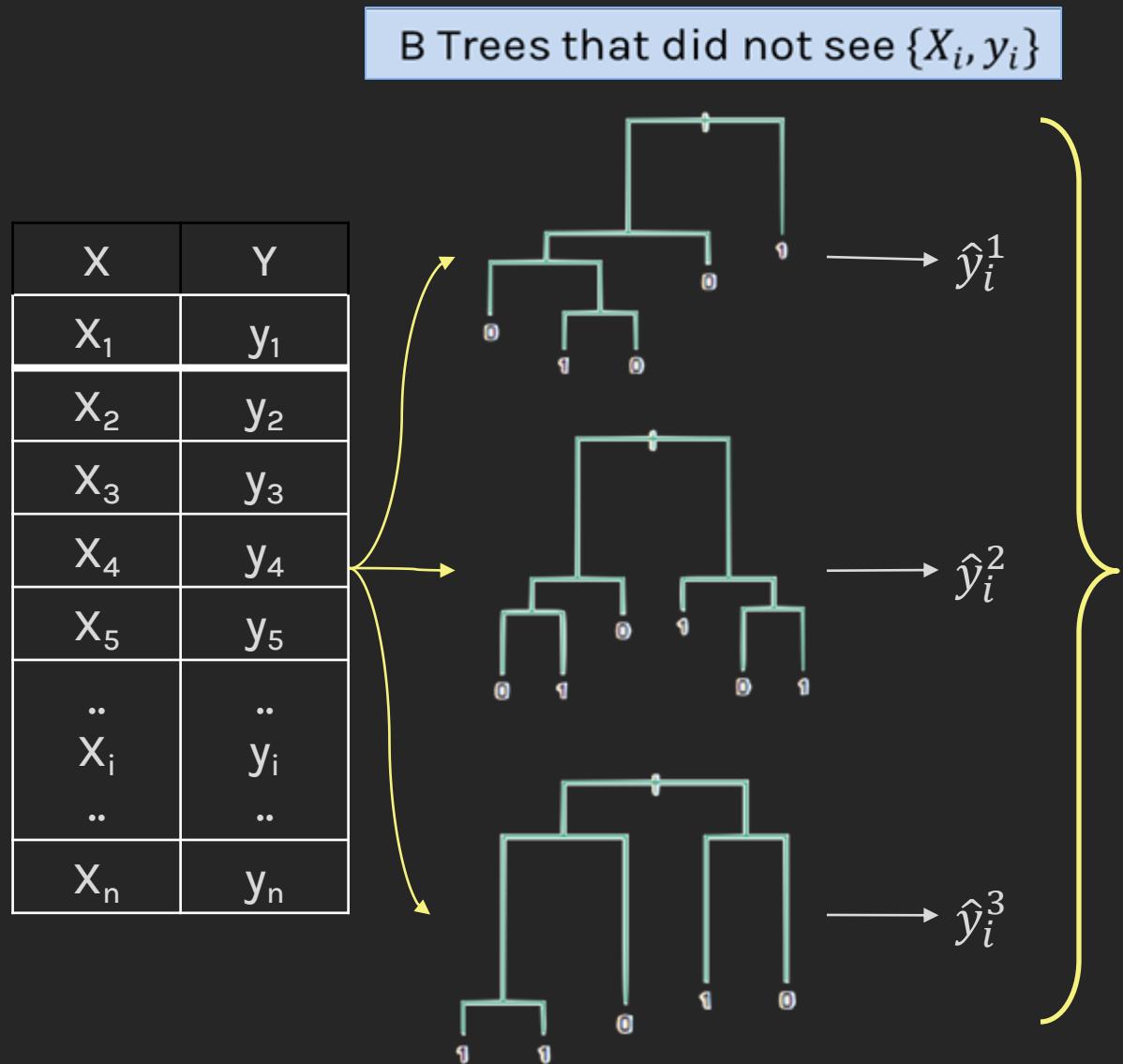
Decision Tree 2



Used and unused
data

| X | Y |
|-------|-------|
| X_1 | y_1 |
| X_2 | y_2 |
| X_3 | y_3 |
| X_4 | y_4 |
| X_5 | y_5 |
| • | • |
| • | • |
| • | • |
| X_n | y_n |

Point-wise out-of-bag error



- Identify observations the trained models have not seen
- Get the predictions for these observations from the models

Point-wise out-of-bag error

Point-wise
prediction

Classification

Point-wise
out-of-bag
error

$$\hat{y}_{i,pw} = \text{majority}(\hat{y}_i^j)$$

$$e_i = \mathbb{I}(\hat{y}_{i,pw} \neq y_i)$$

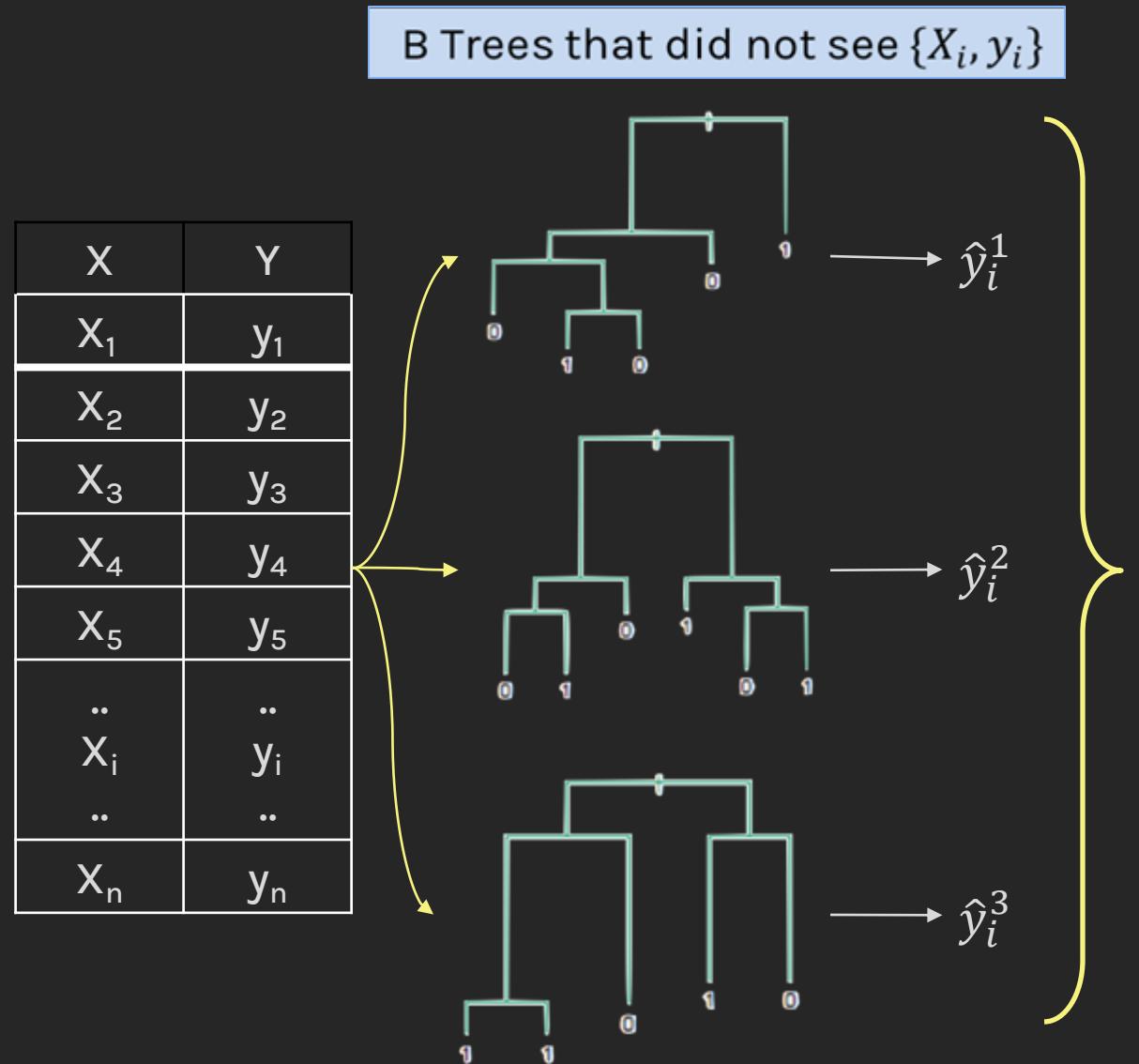
Take **majority** for
classification and **average** for
regression tasks as the
validation prediction for that
observation

Regression

$$\hat{y}_{i,pw} = \frac{1}{B} \sum_{j \in B} \hat{y}_{i,j}$$

$$e_i = (y_i - \hat{y}_{i,pw})^2$$

Point-wise out-of-bag error



Point-wise
prediction

Classification

$$\hat{y}_{i,pw} = \text{majority}(\hat{y}_i^j)$$

$$e_i = \mathbb{I}(\hat{y}_{i,pw} \neq y_i)$$

Regression

$$\hat{y}_{i,pw} = \frac{1}{B} \sum_{j \in B} \hat{y}_{i,j}$$

$$e_i = (y_i - \hat{y}_{i,pw})^2$$

Point-wise
out-of-bag
error

OOB Error

We average the point-wise out-of-bag errors over the full training set.

Classification

n

$$Error_{OOB} = \frac{1}{N} \sum_i^N e_i = \frac{1}{N} \sum_i^N \mathbb{I}(\hat{y}_{i,pw} \neq y_i)$$

Regression

$$Error_{OOB} = \frac{1}{N} \sum_i^N e_i = \frac{1}{N} \sum_i^N (y_i - \hat{y}_{i,pw})^2$$

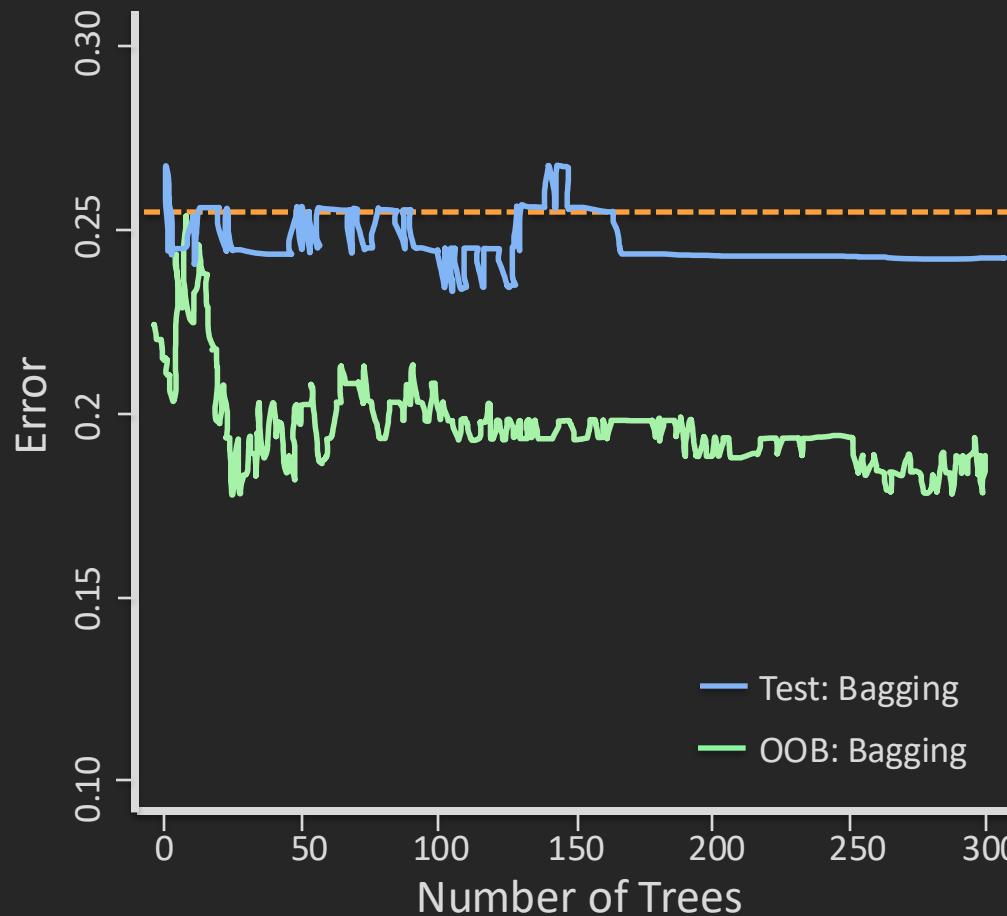
Out-of-Bag Error: Summary

With ensemble methods, we get a new metric for assessing the predictive performance of the model, the *out-of-bag error*.

Given a training set and an ensemble of models, we compute the *out-of-bag error* by

1. For each point x_i in the training set, we average the predicted outputs \hat{y}'_i 's. To do so we only use the B trees whose bootstrap training set excludes this point.
2. We compute the error of this averaged prediction. We call this the **point-wise out-of-bag error**.
3. We average the point-wise out-of-bag error over the full training set N .

Why OOB Error? COMPARING OOB AND CROSS VALIDATION



- While using the cross-validation technique, every validation set has already been seen in training by a few decision trees and hence there is a **leakage of data**.
- OOB Error prevents leakage and yields a better model with lower variance or less overfitting.
- There is also **lesser computational cost** for OOB as compared to CV for bagging.

Bagging

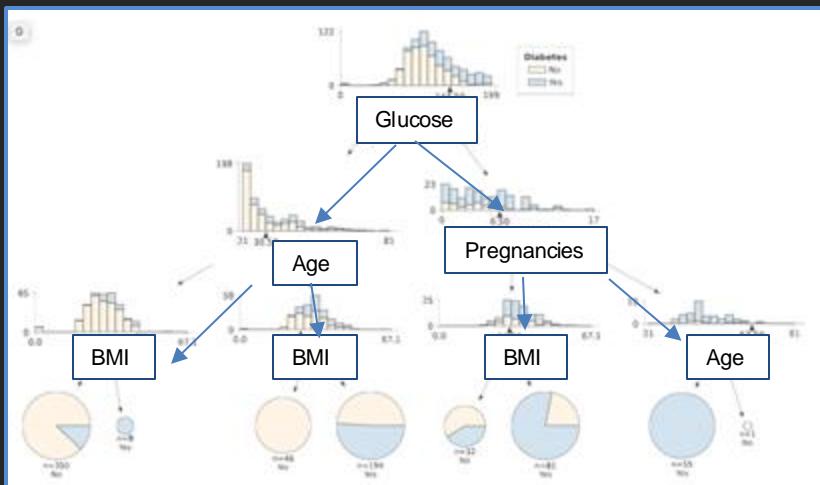
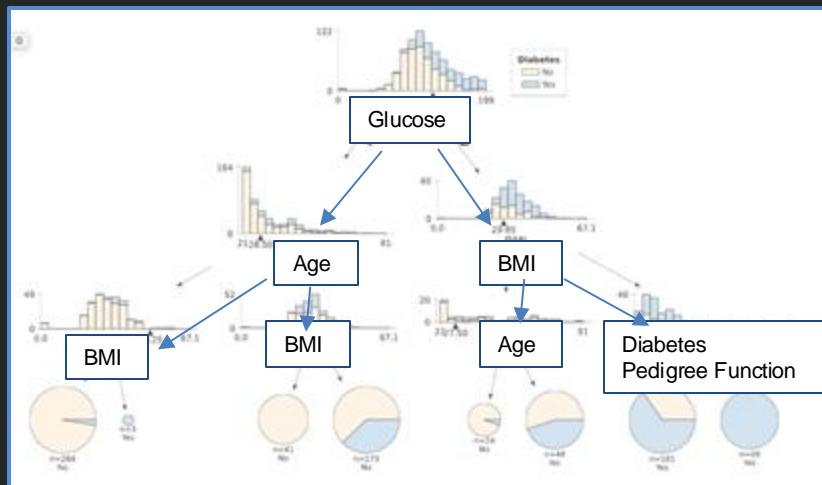
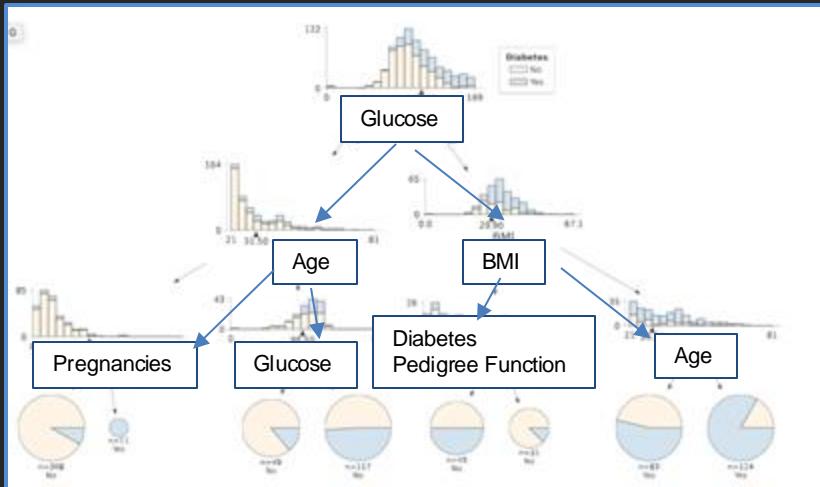
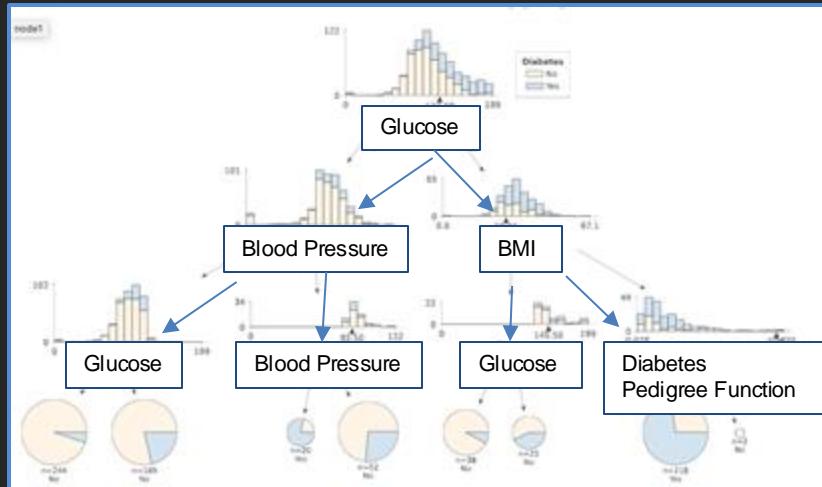
Interpretability:

Still an issue and we will address this later.

If the individual trees are too shallow, the ensembled model can still underfit. Even if we combine many underfitting trees we will still underfit.

If the individual trees are too large, the ensembled model could still overfit.

Drawbacks of Bagging



For each bootstrap, we build a decision tree.

Created by: Dr. Rahul Dave

Improving on Bagging

In practice, the trees in Bagging tend to be **highly correlated**.

- Suppose we have an **extremely strong predictor**, x_j , in the training set amongst **moderate predictors**. Then the greedy learning algorithm ensures that most of the models in the ensemble will choose to split on x_j in early iterations.
- However, we assumed (or hope) that each tree in the ensemble is **independently and identically distributed**.

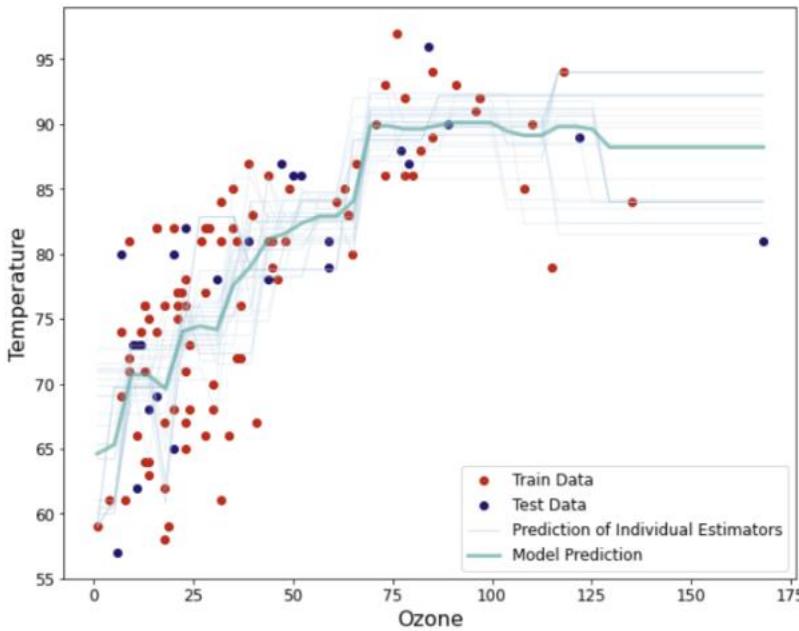
Next Monday, on 'Tree Mysteries Unveiled': Can trees ever truly be independent? 🌳 The secrets unraveled! Tune in and unlock the enigma... Only at the Monday Lecture!"





Exercise: Regression with Bagging

The aim of this exercise is to understand regression using Bagging.



Thank you



Random Forest And Variable Importance



CS1090A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai, Chris Gumb

Dylan Wu
Arashiyama Forest, Kyoto

Outline

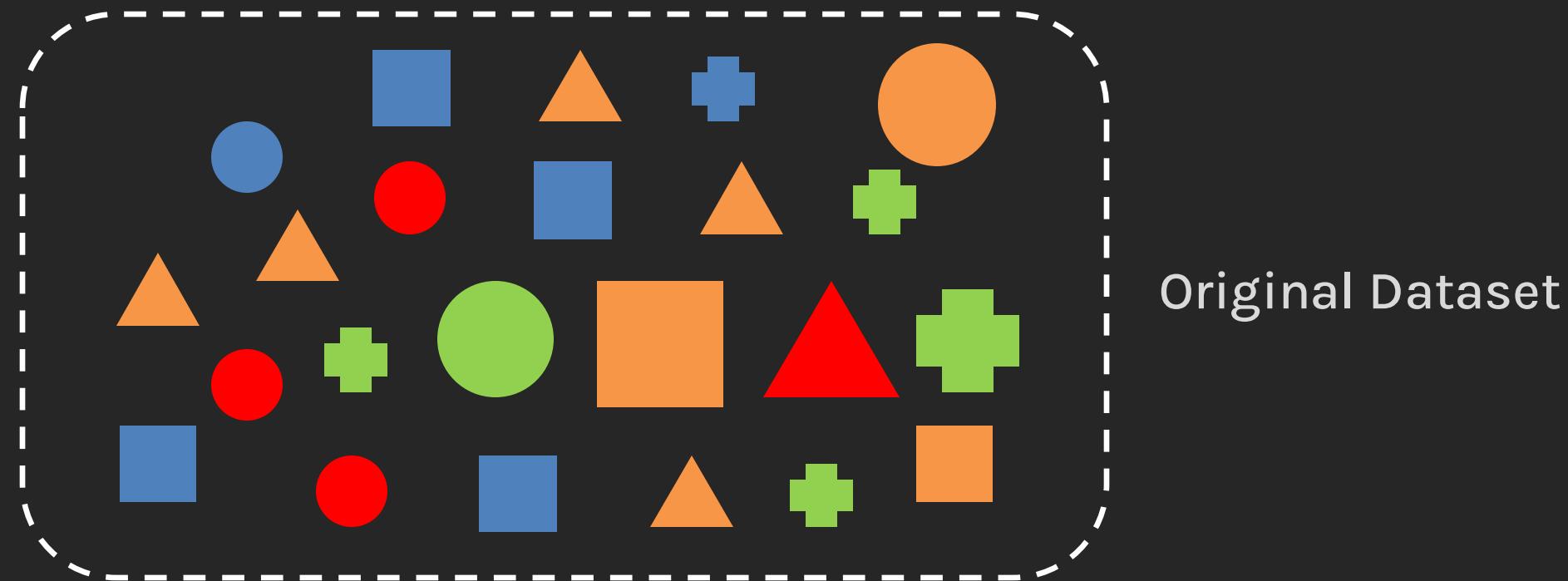
- Motivation
- Random Forest
- Variable Importance
- Missing Data (again)
- Class Imbalance
- Tree building algorithms

Outline

- Motivation
- Random Forest
- Variable Importance
- Missing Data (again)
- Class Imbalance
- Tree building algorithms

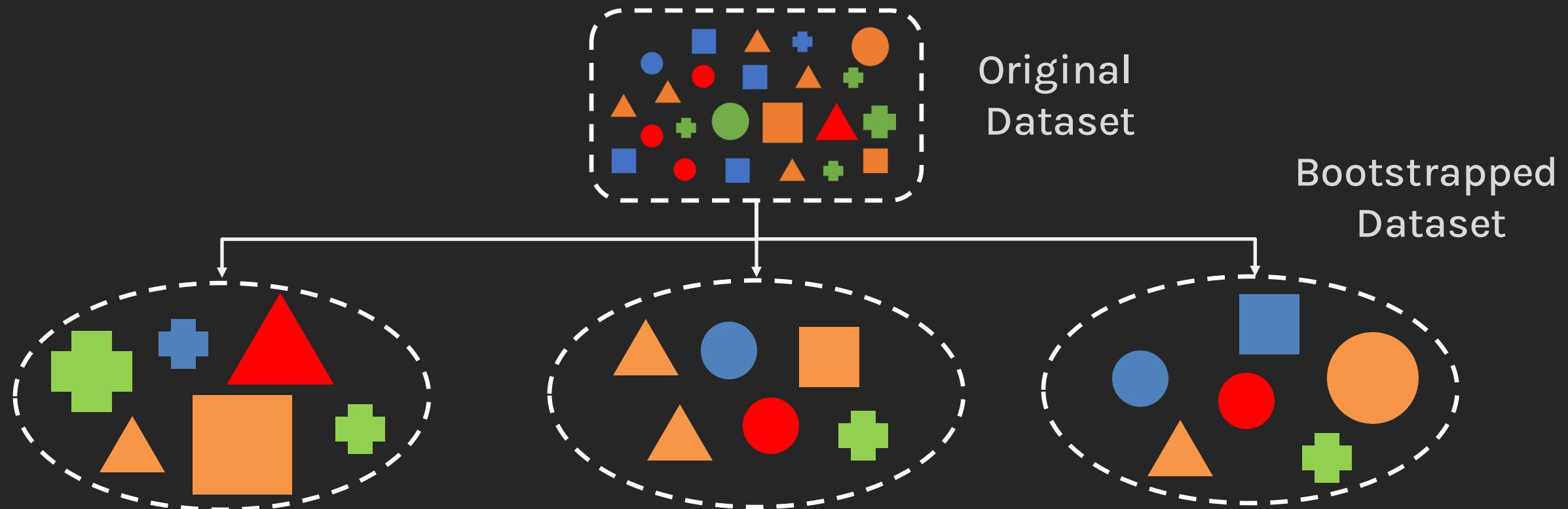
Motivation

Consider a dataset of geometric shapes with features of **Color**, **Shape** and **Size**, and we want to use the ensemble learning method, Bagging (Bootstrap Aggregating).



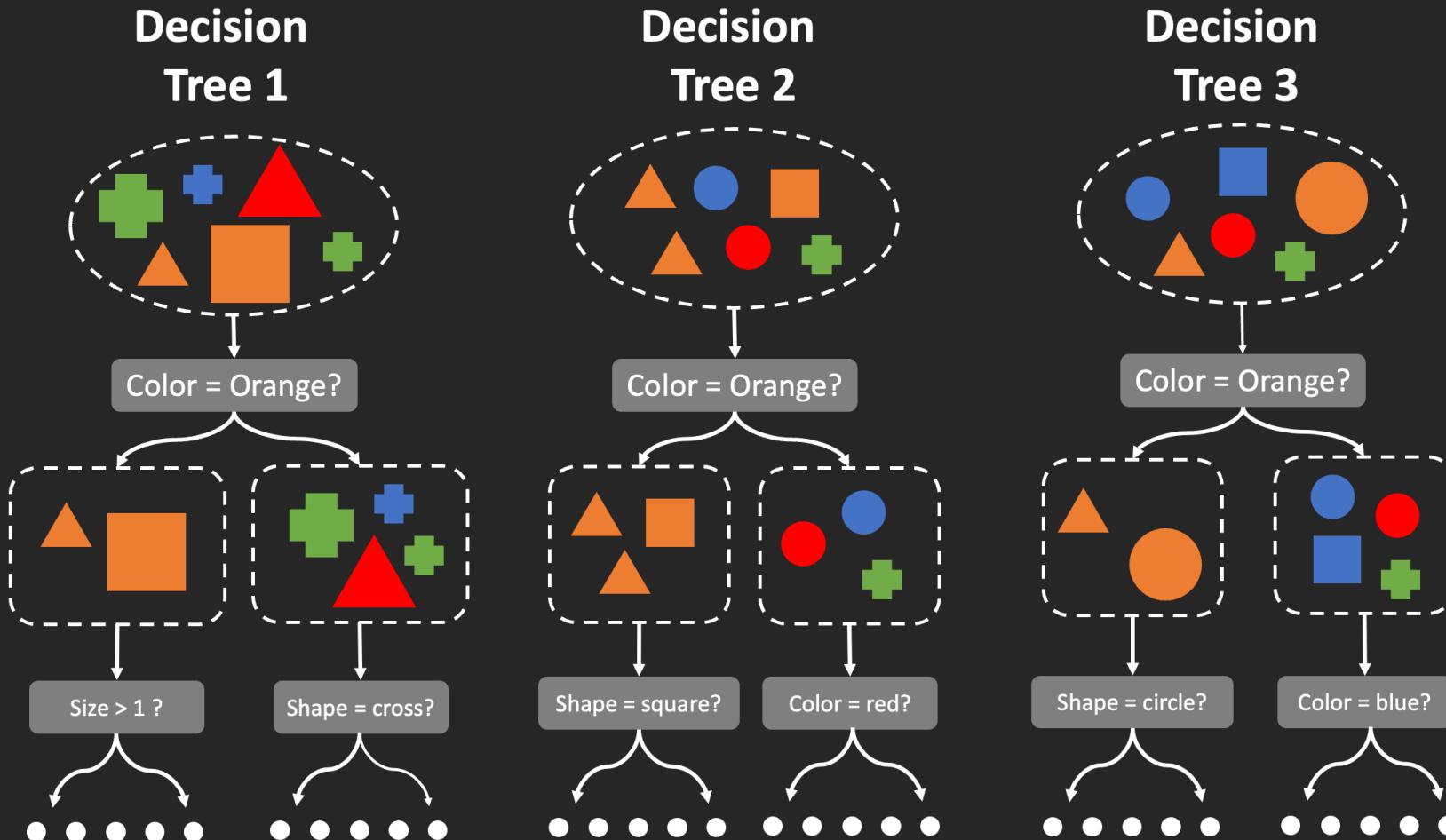
Motivation

Multiple subsets of the dataset are created using bootstrapping in the first step of Bagging.



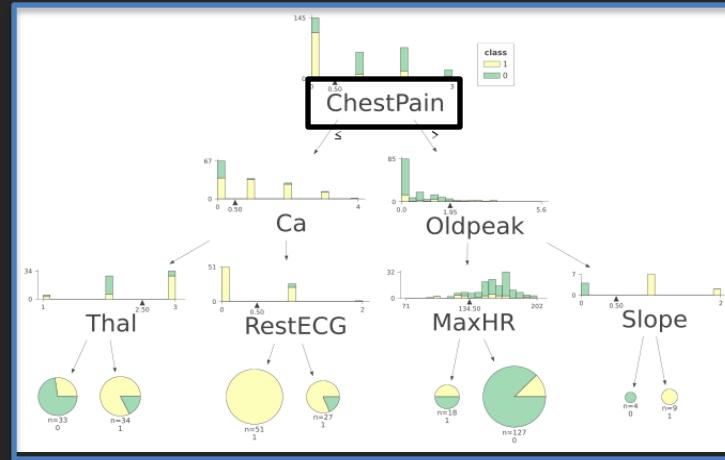
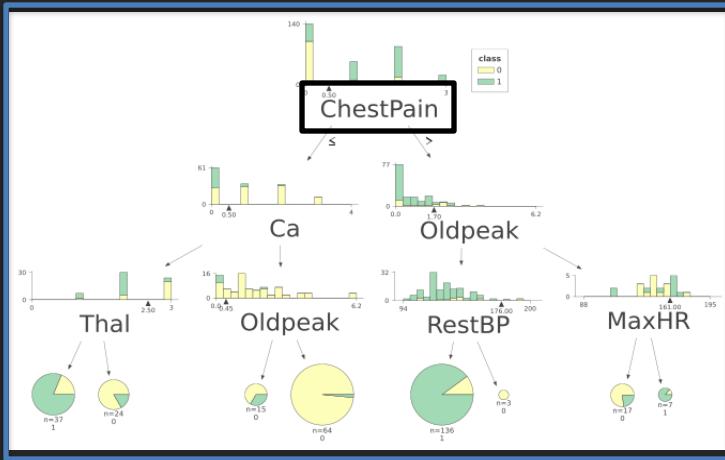
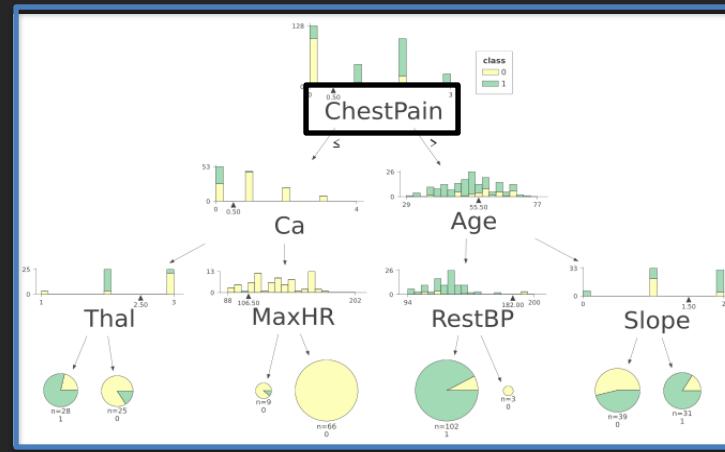
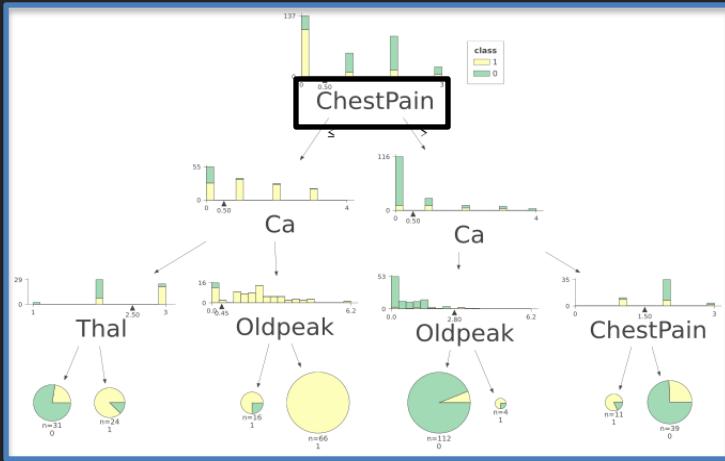
Motivation

Then multiple decision trees are fitted with each of the bootstrapped datasets.



Motivation

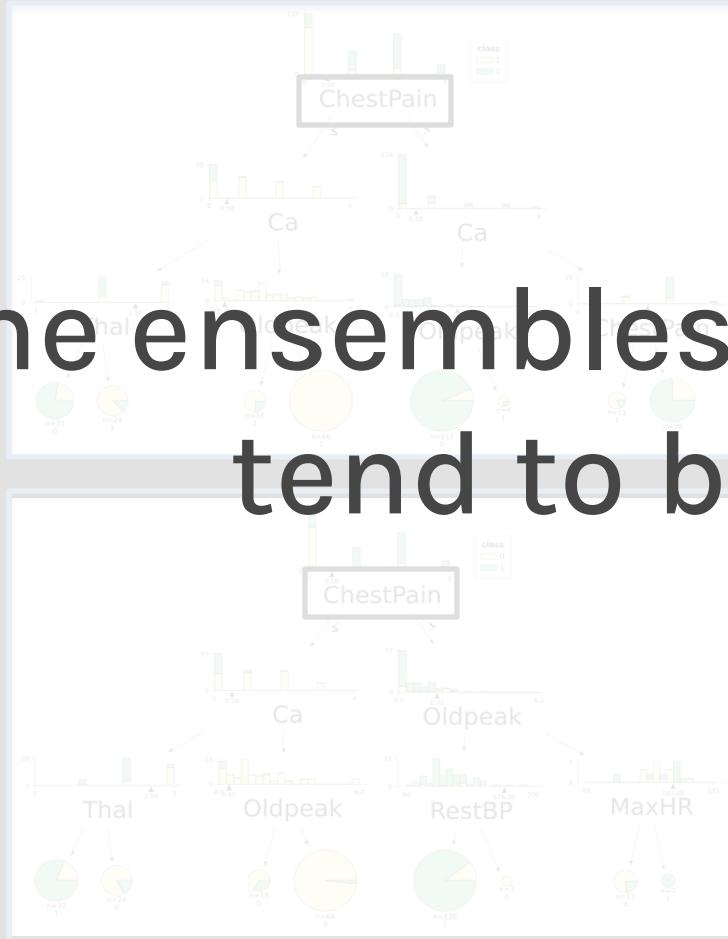
But as we have seen, these trees are correlated.



Motivation

Consider the following decision trees in a bagging model that predicts if a person has heart disease:

The ensembles of trees in bagging tend to be **correlated**.



Outline

- Motivation
- **Random Forest**
- Variable Importance
- Missing Data (again)
- Class Imbalance
- Tree building algorithms

**I Entered
A Random Forest**



**Now I see
The Future**

Random Forest

How can we avoid this issue then?

OR How do we de-correlate the trees?

Random forest is a modified form of bagging that creates ensembles of **independent** decision trees.

Then the question is, **how?**

Quiz time



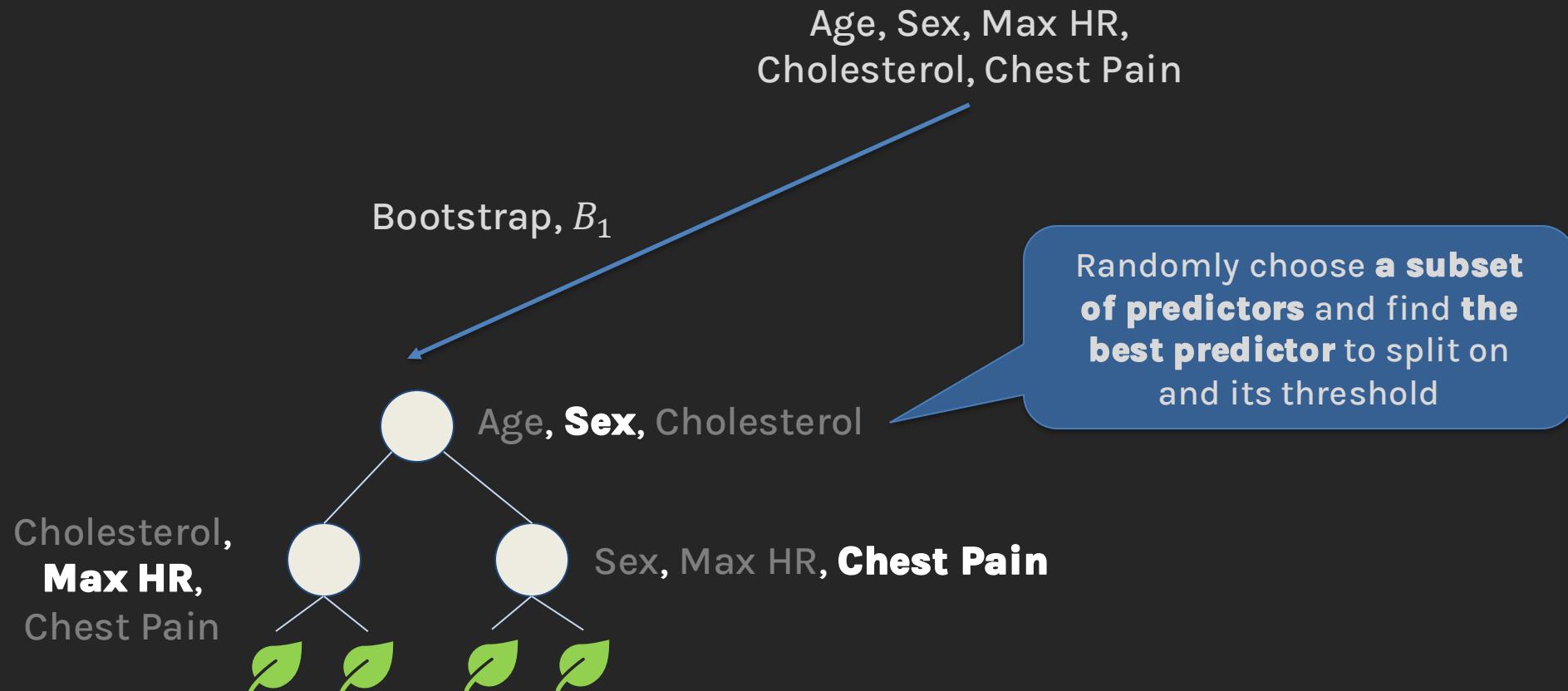
Which of the following(s) approaches do you think could be effective in reducing correlation among the trees?

Options

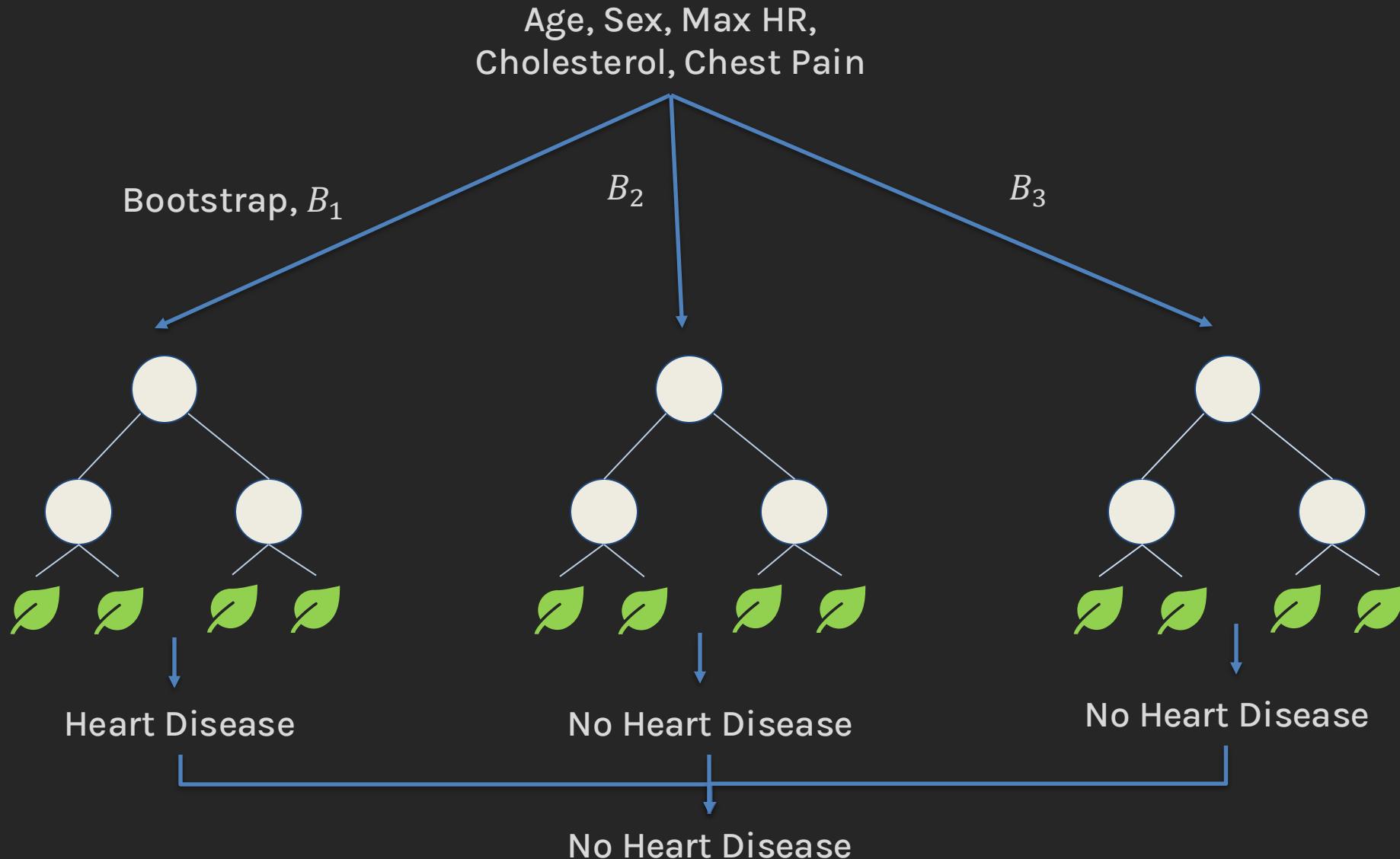
- A. Limit the depth of each tree in a unique way.
- B. Ensure that each tree's split does not include any of the splits used in the previous trees.
- C. Randomly select a subset of predictors to consider for splitting at each node.
- D. Randomly choose a single predictor for every split.
- E. Design the splits such that the probability of choosing a significant predictor is low, leading to an ensemble of weaker models.

Random Forest

Consider a dataset that contains the following predictors:



Random Forests



Random Forests

In summary, Random Forest works in the following ways:

1. Create B **bootstrapped datasets** with all J predictors (same as in bagging).
2. Initialize a random forest with B decision trees.
3. For each tree, at each split, we **randomly** select a **subset** of J' predictors from the full set of predictors ($J' < J$).
4. Amongst the J' **predictors**, we select the optimal predictor and the optimal threshold for the corresponding split.

Tuning Random Forests

Random forest models have multiple **hyper-parameters** to tune:

1. The **number of predictors** to randomly select at each split.
2. The **total number of trees** in the ensemble.
3. The **stopping criteria** - maximum depth, minimum leaf node size, etc.
4. The **splitting criterium** - gini, entropy

Tuning Random Forests

There are standard (default) values for each of random forest hyper-parameters recommended by long time practitioners.

For **THE NUMBER OF PREDICTORS**

$\sqrt{N_j}$ predictors for classification

$\frac{N_j}{3}$ predictors for regression

For **THE NUMBER OF TREES**, we use out-of-bag errors. With OOB, training and validation can be done in a single sequence - once the out-of-bag error stabilizes, adding more trees may not be beneficial.

Also, generally these parameters should be tuned through **OOB** (making them data and problem dependent).

Outline

- Motivation
- Random Forest
- Variable Importance
- Missing Data (again)
- Class Imbalance
- Tree building algorithms

Variable Importance for RF (and Bagging)

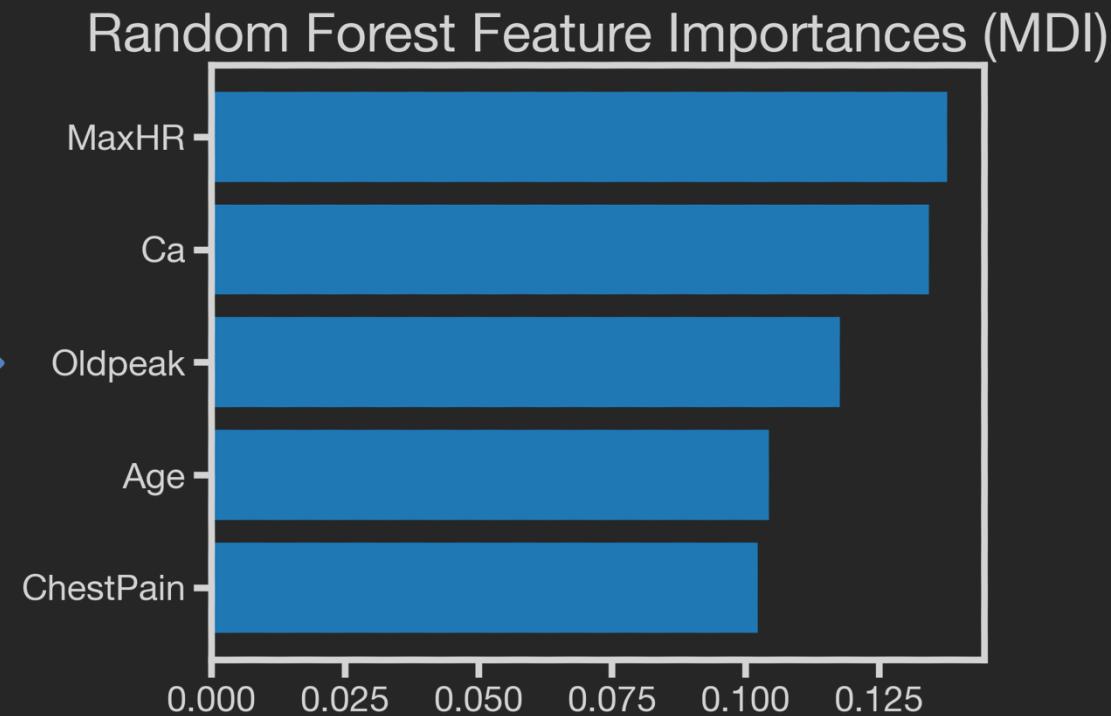
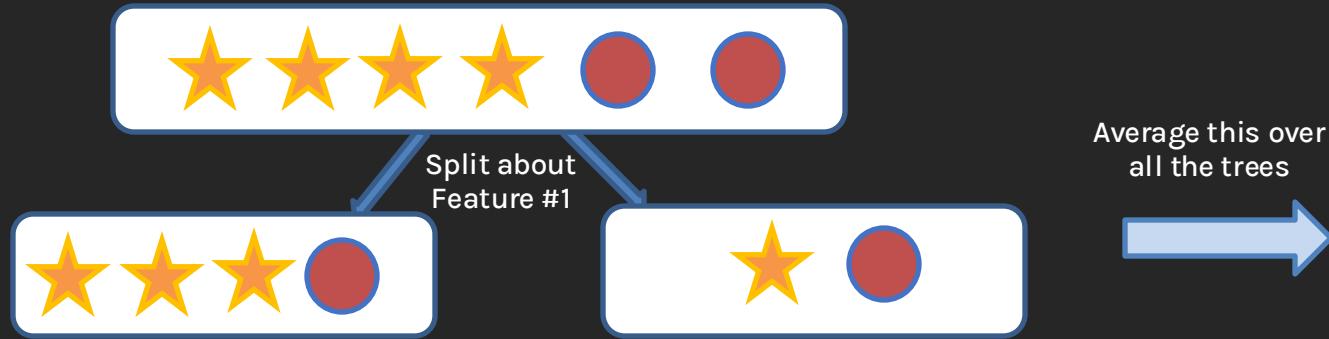
1. Mean Decrease in Impurity.
2. Permutation importance.
3. LIME, SHAP values [lab and reading]

Variable Importance for RF (and Bagging)

1. Mean Decrease in Impurity.
2. Permutation importance.
3. LIME, SHAP values [lab and reading]

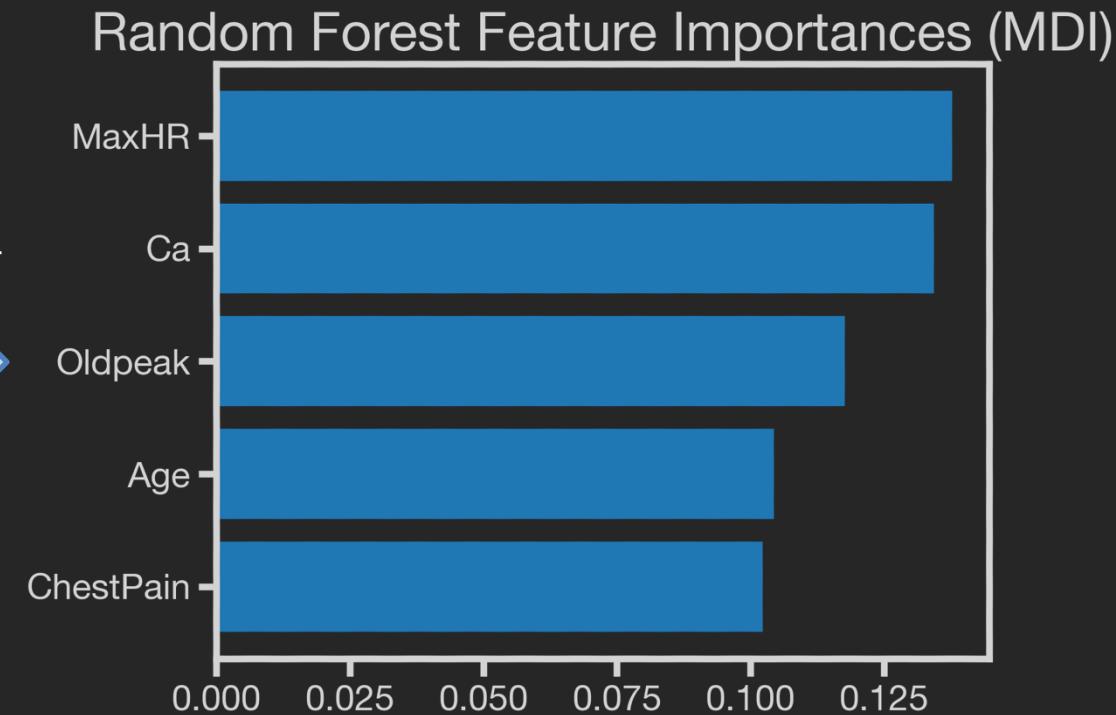
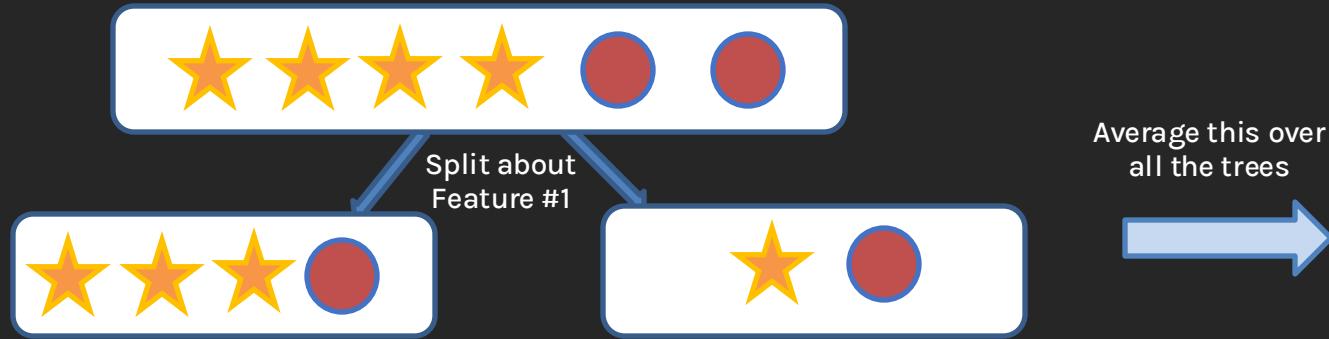
Mean Decrease in Impurity (MDI)

Decision trees make splits that maximize the decrease in impurity. By calculating the mean decrease in impurity for each feature across all trees, we arrive at the variable importance for a bagging or random forest model.



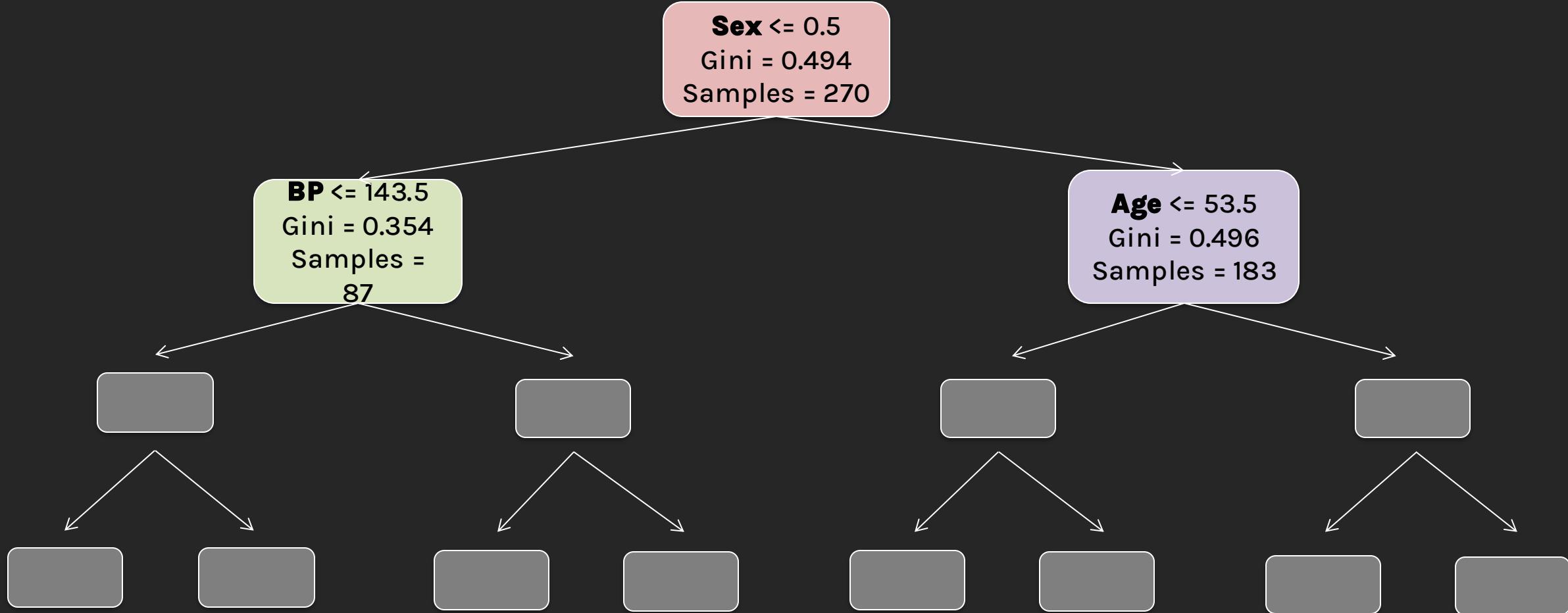
Mean Decrease in Impurity (MDI)

Instead of calculating at the individual tree level, we can calculate the MEAN decrease in impurity for each predictor across all trees.



Mean Decrease in Impurity (MDI)

Consider the following decision tree:



Mean Decrease in Impurity (MDI)

Step 1: Calculate the mean decrease in impurity for each node q in the decision tree.

Mean decrease in impurity for each node q

$$\Delta I_q = \left(\frac{n}{N} \right) \left[Gini_n - \sum_{m \in \text{Child}(n)} \left(\frac{m}{n} \right) Gini_m \right]$$

Fraction of n samples from the node out of the **whole dataset**

Sum over all children of node n

Mean Decrease in Impurity (MDI)

Step 1: Calculate the mean decrease in impurity for each node q in the decision tree.

$$\Delta I_q = \left(\frac{n}{N} \right) \left[Gini_n - \left(\frac{m_L}{n} \right) Gini_{m_L} - \left(\frac{m_R}{n} \right) Gini_{m_R} \right]$$

Mean Decrease in Impurity (MDI)

$$\Delta I_q = \left(\frac{n}{N} \right) \left[Gini_n - \sum \left(\frac{m}{n} \right) Gini_m \right]$$

Step 1: Calculate the mean decrease in impurity for each node q in the decision tree.



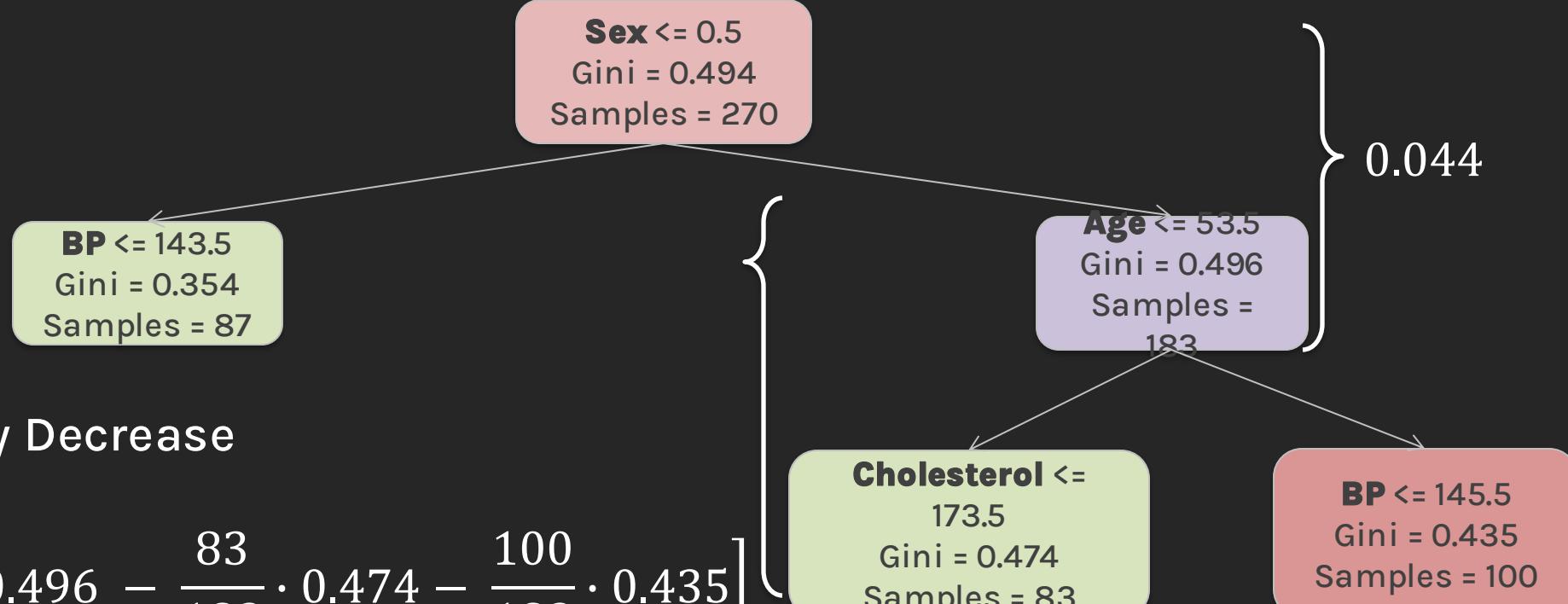
Impurity Decrease

$$\begin{aligned} &= \frac{270}{270} \left[0.494 - \frac{87}{270} \cdot 0.354 - \frac{183}{270} \cdot 0.496 \right] \\ &\approx 0.044 \end{aligned}$$

Mean Decrease in Impurity (MDI)

$$\Delta I_q = \left(\frac{n}{N} \right) \left[Gini_n - \sum \left(\frac{m}{n} \right) Gini_m \right]$$

Step 1: Calculate the mean decrease in impurity for each node q in the decision tree.



Impurity Decrease

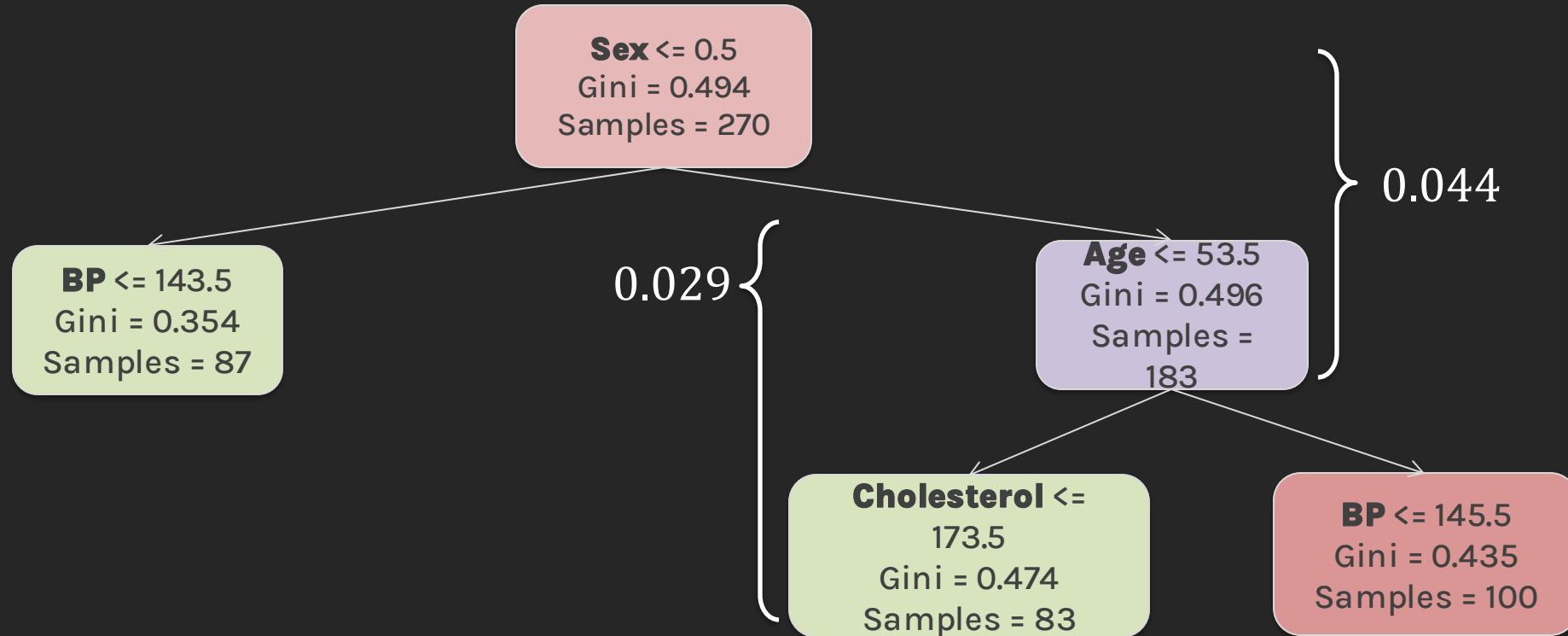
$$= \frac{183}{270} \left[0.496 - \frac{83}{183} \cdot 0.474 - \frac{100}{183} \cdot 0.435 \right]$$

≈ 0.029

Mean Decrease in Impurity (MDI)

$$\Delta I_q = \left(\frac{n}{N} \right) \left[Gini_n - \sum \left(\frac{m}{n} \right) Gini_m \right]$$

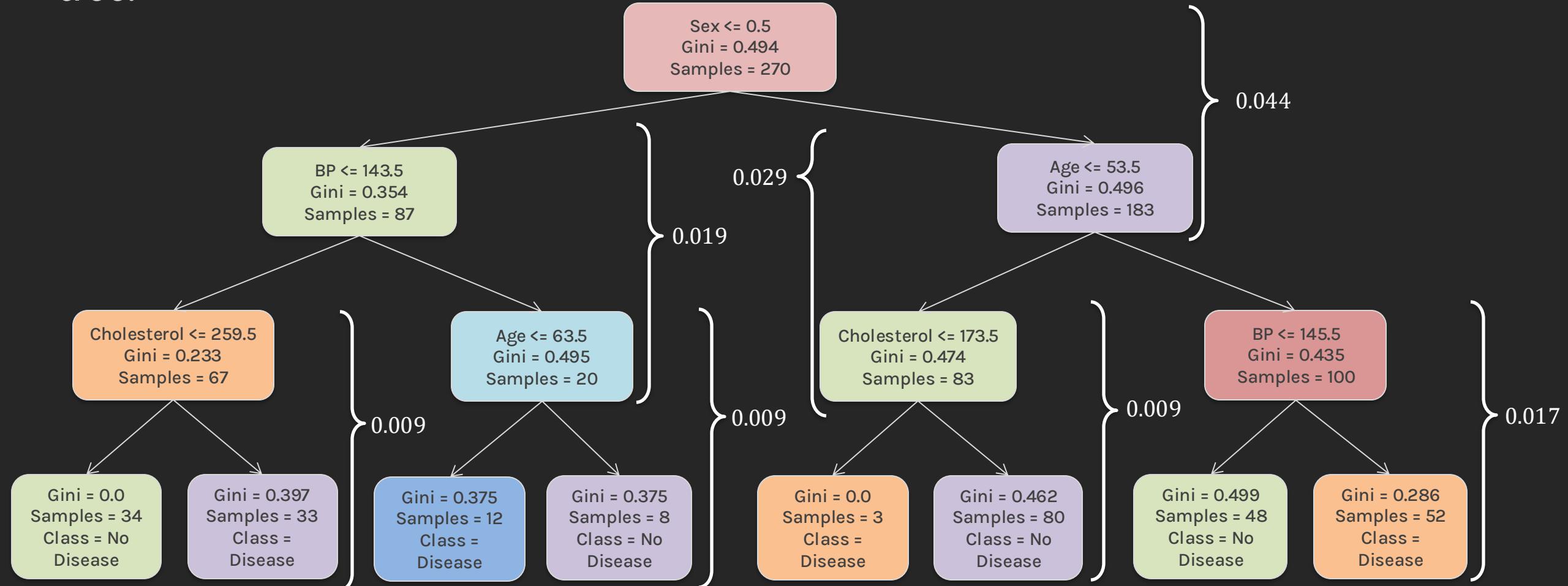
Step 1: Calculate the mean decrease in impurity for each node q in the decision tree.



Mean Decrease in Impurity (MDI)

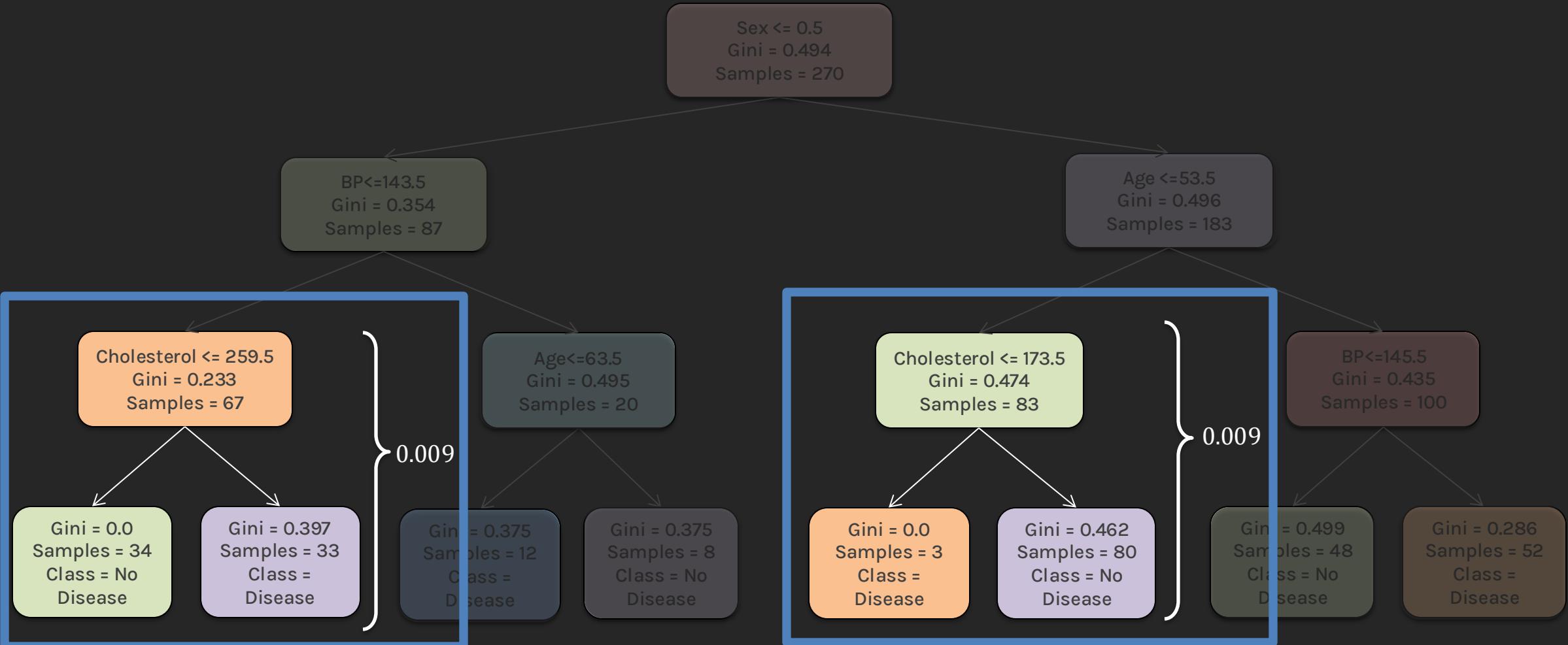
$$\Delta I_q = \left(\frac{n}{N} \right) \left[Gini_n - \sum \left(\frac{m}{n} \right) Gini_m \right]$$

Step 1: Calculate the mean decrease in impurity for each node q in the decision tree.



Mean Decrease in Impurity (MDI)

Let us try to compute the importance of the feature **cholesterol**:



Mean Decrease in Impurity (MDI)

Step 2: Calculate the importance of the feature by **summing up** the impurity decrease calculated in step 1 for each node in which that feature occurs.

$$\text{Feature Importance}_{cholesterol} = \sum_{n \in \text{nodes split on cholesterol}} \text{Impurity Decrease}_n$$
$$= 0.009 + 0.009 = 0.018$$

Mean Decrease in Impurity (MDI)

Step 3: Normalize this value between 0 and 1 by dividing by the sum of all feature importance values.

This ensures the sum of all feature importance in a decision tree adds up to 1.

$$\text{Norm Feature Importance}_{cholesterol} = \frac{\text{Feature Importance}_{cholesterol}}{\sum_{j \in \text{all features}} \text{Feature Importance}_j}$$

Mean Decrease in Impurity (MDI)

Step 4: To calculate the feature importance at the Random Forest or Bagging level, we **average** the normalized feature importance of the given predictor over **all the trees**.

$$\text{Norm RF/Bagging Feature Importance}_{cholesterol} = \frac{\sum_{t \in \text{all trees}} \text{Norm Feature Importance}}{\text{Total number of trees}}$$

Summary: Mean Decrease in Impurity (MDI)

For each feature j in the dataset:

Step 1: Calculate the **mean decrease in impurity** for **each node** n in the decision tree t .

Step 2: Calculate the **feature importance** by **summing up** the impurity decrease calculated in step 1 for each node n in which that feature j occurs.

Step 3: **Normalize** this value between 0 and 1 by dividing by the sum of all feature importance values.

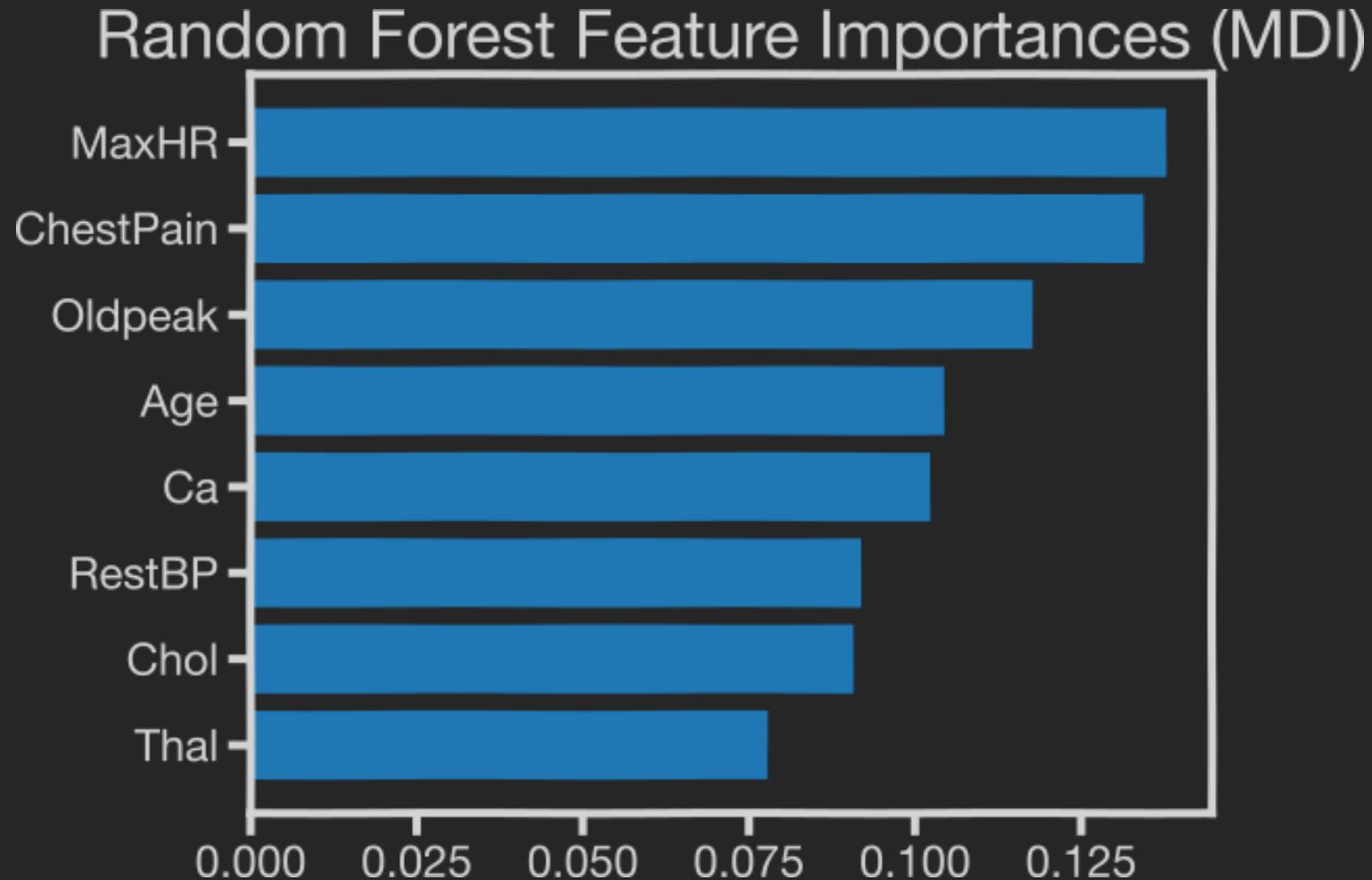
Step 4: At the Random Forest level, **average** over all the T trees.

$$F_j^{(t)} = \sum_{n \in \text{nodes}_j} I_n^{(t)}$$

$$\hat{F}_j^{(t)} = \frac{F_j^{(t)}}{\sum_i F_i^{(t)}}$$

$$\mathcal{F}_j = \frac{\sum_t \hat{F}_j^{(t)}}{T}$$

Summary: Mean Decrease in Impurity (MDI)



Variable Importance for RF (and Bagging)

1. Mean Decrease in Impurity.
2. Permutation importance.
3. LIME, SHAP values [lab and reading]

Permutation Importance

Consider the following dataset:

| Height (cm) | Weight (kg) | ... | Fitness Level (1 – 5) |
|-------------|-------------|-----|-----------------------|
| 150 | 65 | ... | 2 |
| 140 | 50 | ... | 3 |
| ... | ... | ... | ... |
| 170 | 70 | ... | 4 |
| 160 | 80 | ... | 1 |

Step 1: Record the validation/OOB accuracy of RF model:
Accuracy = 0.88

Permutation Importance

Step 1: Record the validation/OOB accuracy of RF model:

Accuracy = 0.88

| Height (cm) | Weight (kg) | ... | Fitness Level (1 – 5) |
|-------------|-------------|-----|-----------------------|
| 150 | 65 | ... | 2 |
| 140 | 50 | ... | 3 |
| ... | ... | ... | ... |
| 170 | 70 | ... | 4 |
| 160 | 80 | ... | 1 |

Step 2: Randomly permute the data for column j in the validation/OOB set.

Permutation Importance

Step 1: Record the validation/OOB accuracy of RF model:

Accuracy = 0.88

| Height (cm) | Weight (kg) | ... | Fitness Level (1 – 5) |
|-------------|-------------|-----|-----------------------|
| 150 | 70 | ... | 2 |
| 140 | 65 | ... | 3 |
| ... | ... | ... | ... |
| 170 | 80 | ... | 4 |
| 160 | 50 | ... | 1 |

Step 2: Randomly permute the data for column j in the validation/OOB set.

Permutation Importance

Step 1: Record the validation/OOB accuracy of RF model:

Accuracy = 0.88

| Height (cm) | Weight (kg) | ... | Fitness Level (1 – 5) |
|-------------|-------------|-----|-----------------------|
| 150 | 70 | ... | 2 |
| 140 | 65 | ... | 3 |
| ... | ... | ... | ... |
| 170 | 80 | ... | 4 |
| 160 | 50 | ... | 1 |

Step 2: Randomly permute the data for the column j (in this case Weight).

Record the validation/OOB accuracy of the RF model on the modified dataset:

Permuted Accuracy = 0.87

Permutation Importance

Step 1: Record the validation/OOB accuracy of RF model:

Accuracy = 0.88

Step 2: Randomly permute the data for the column j (in this case Weight).

Record the validation/OOB accuracy of the RF model on the modified dataset:

Permuted Accuracy = 0.87

Step 3: Repeat the previous step with K number of times and average all the accuracies.

Assume we permute the feature 3 times, we get:

Permuted Accuracy₁ = 0.82



Avg Permuted Accuracy = $\frac{0.82+0.87+0.86}{3} = 0.85$

Permuted Accuracy₂ = 0.87

Permuted Accuracy₃ = 0.86

Permutation Importance

Step 1: Record the validation/OOB accuracy of RF model:

Accuracy = 0.88

Step 2: Randomly permute the data for the column j (in this case Weight).

Record the validation/OOB accuracy of the RF model on the modified dataset:

Permuted Accuracy = 0.87

Step 3: Repeat the previous step with K number of times and average all the

Avg Permuted Accuracy= 0.85

Step 4: Calculate the difference between unpermuted and average permuted accuracy to get the importance of the feature in the random forest:

Difference = 0.88 - 0.85 = 0.03

Summary: Permutation Importance

For each feature j in the dataset:

Step 1: Record the validation/OOB (unpermuted) accuracy of RF model: s .

Step 2: For each repetition k in $1 \dots K$:

Randomly permute the data for the column j . Record the validation/OOB accuracy $s_{k,j}$ of the RF model on the modified dataset.

Step 3: Compute the average accuracy from all the permuted datasets

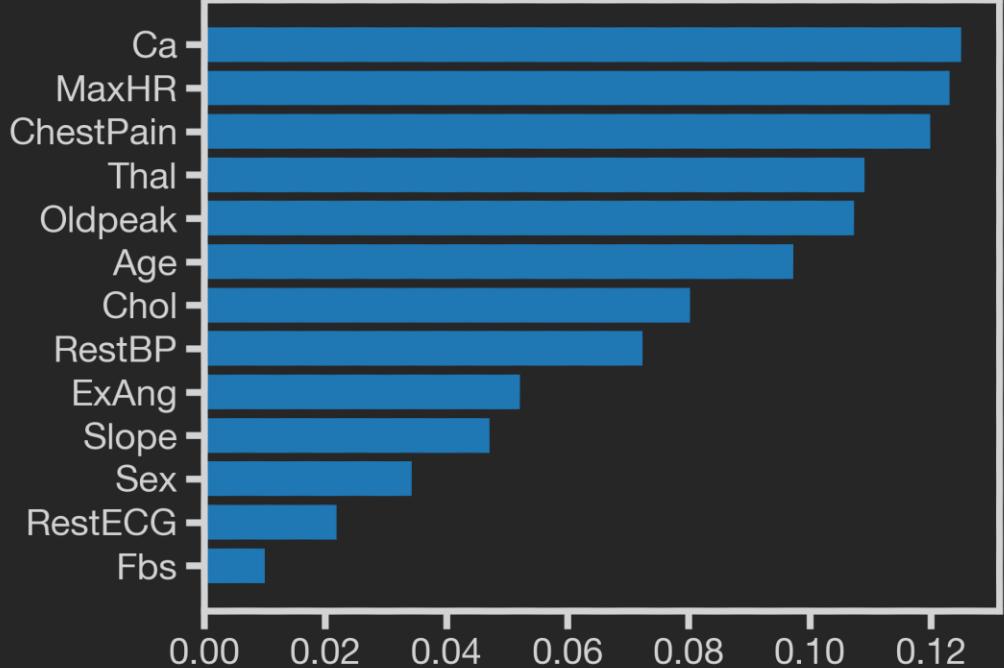
$$s_j = \frac{1}{K} \sum_{k=1}^K s_{k,j}$$

Step 4: Calculate the difference between unpermuted and average permuted accuracy to get the importance of the feature in the random forest.

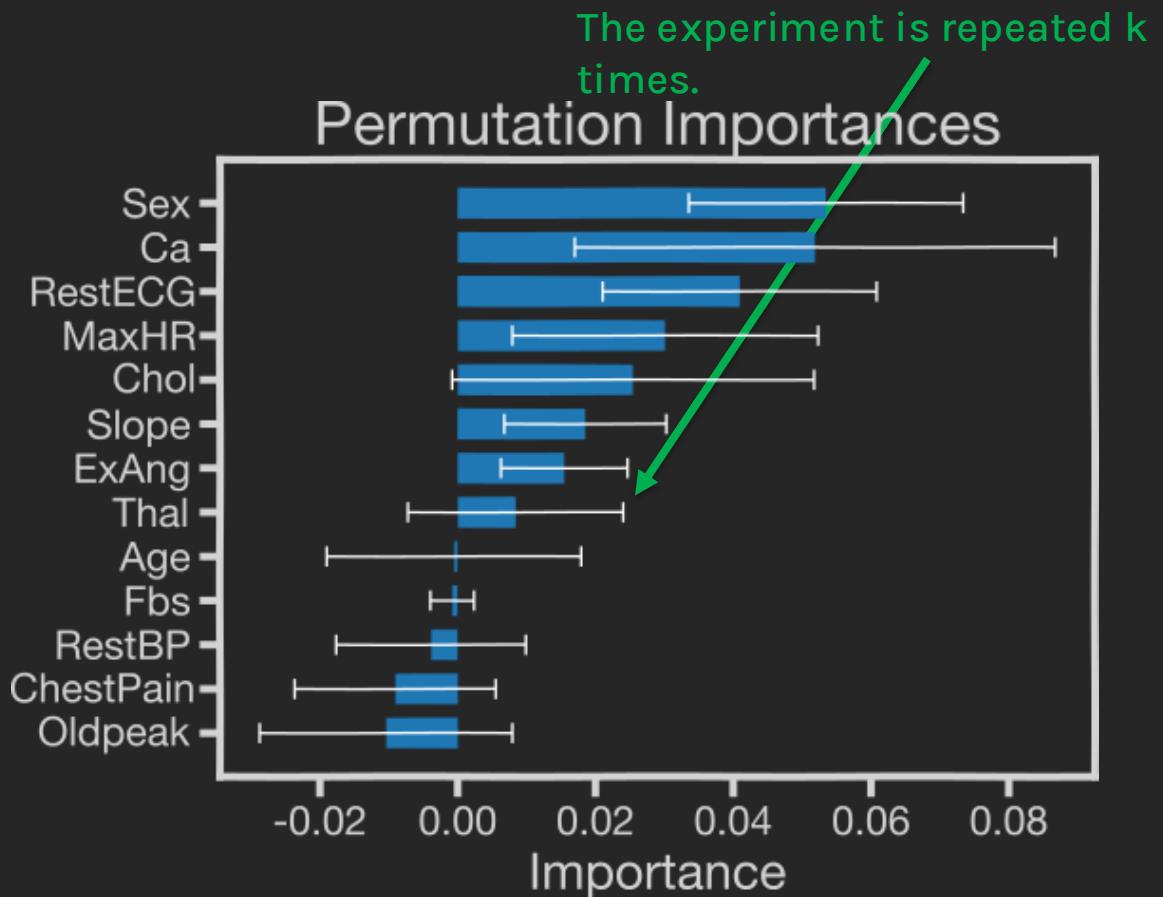
$$\text{RF Feature Importance}_j = s - s_j$$

MDI vs Permutation Importance

Random Forest Feature Importances (MDI)

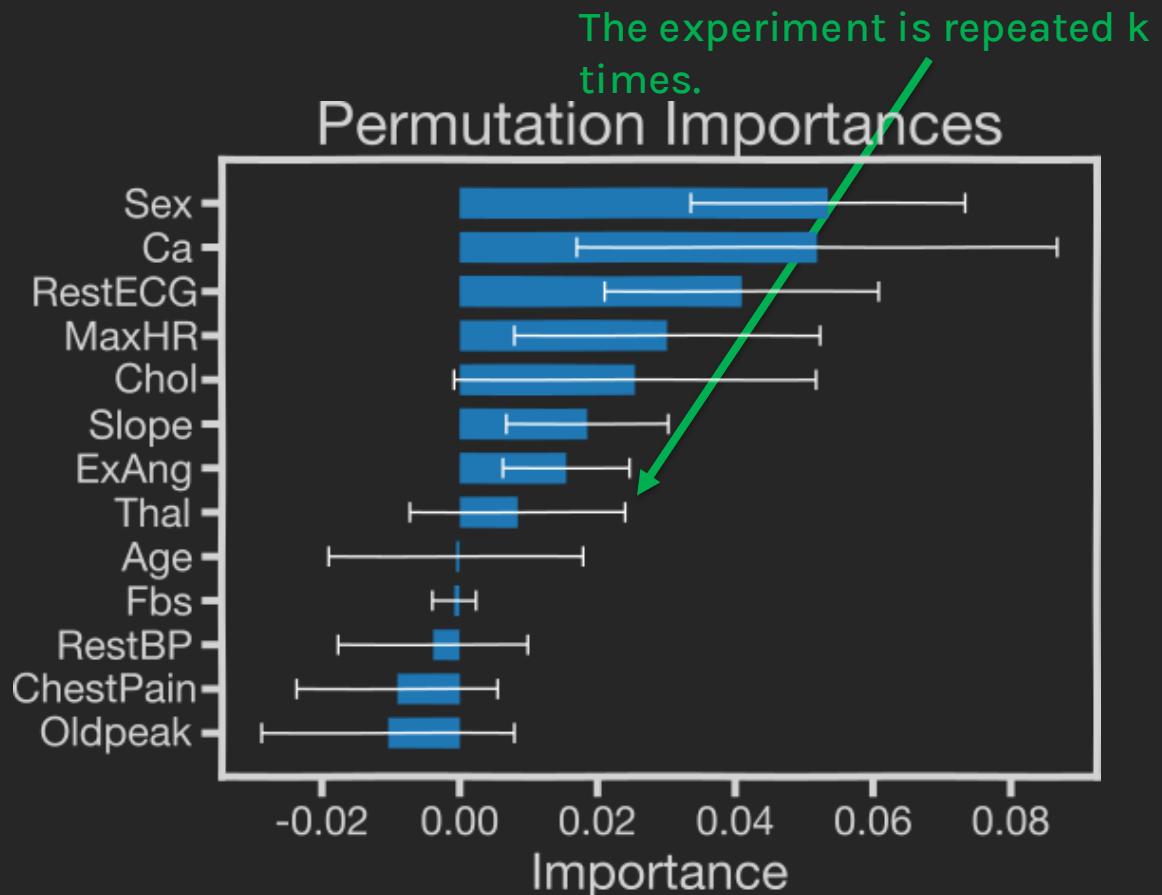
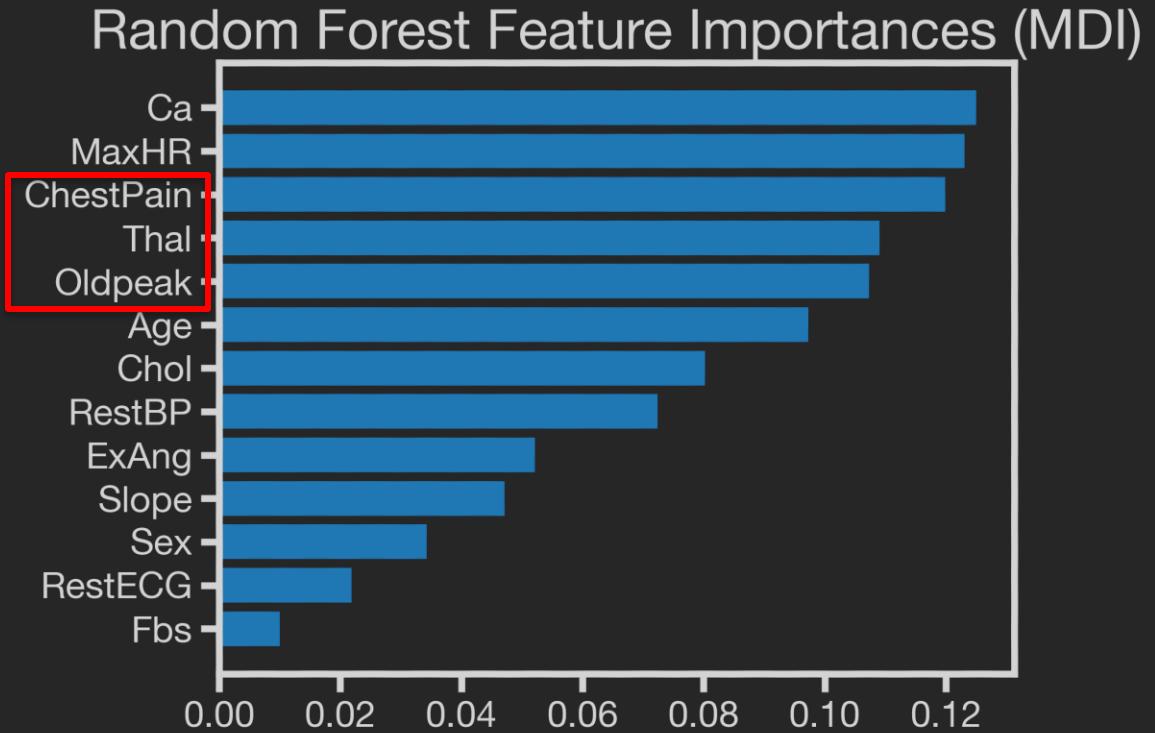


Permutation Importances



Features like ***ChestPain***, ***OldPeak***, ***Thal*** are ranked most important in MDI importance plot, but they are ranked low in permutation importance plot.

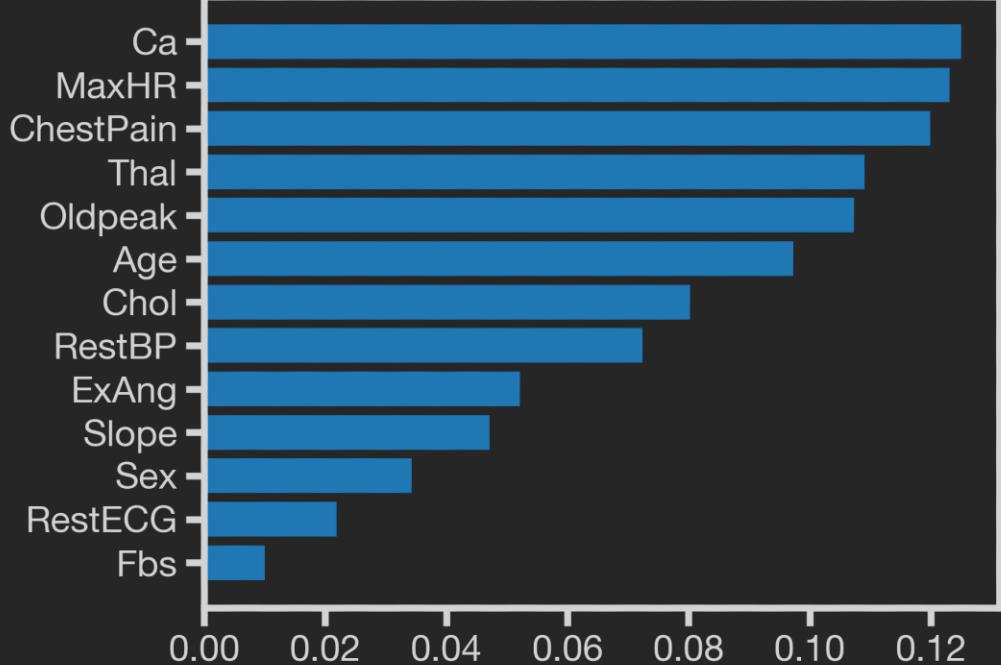
MDI vs Permutation Importance



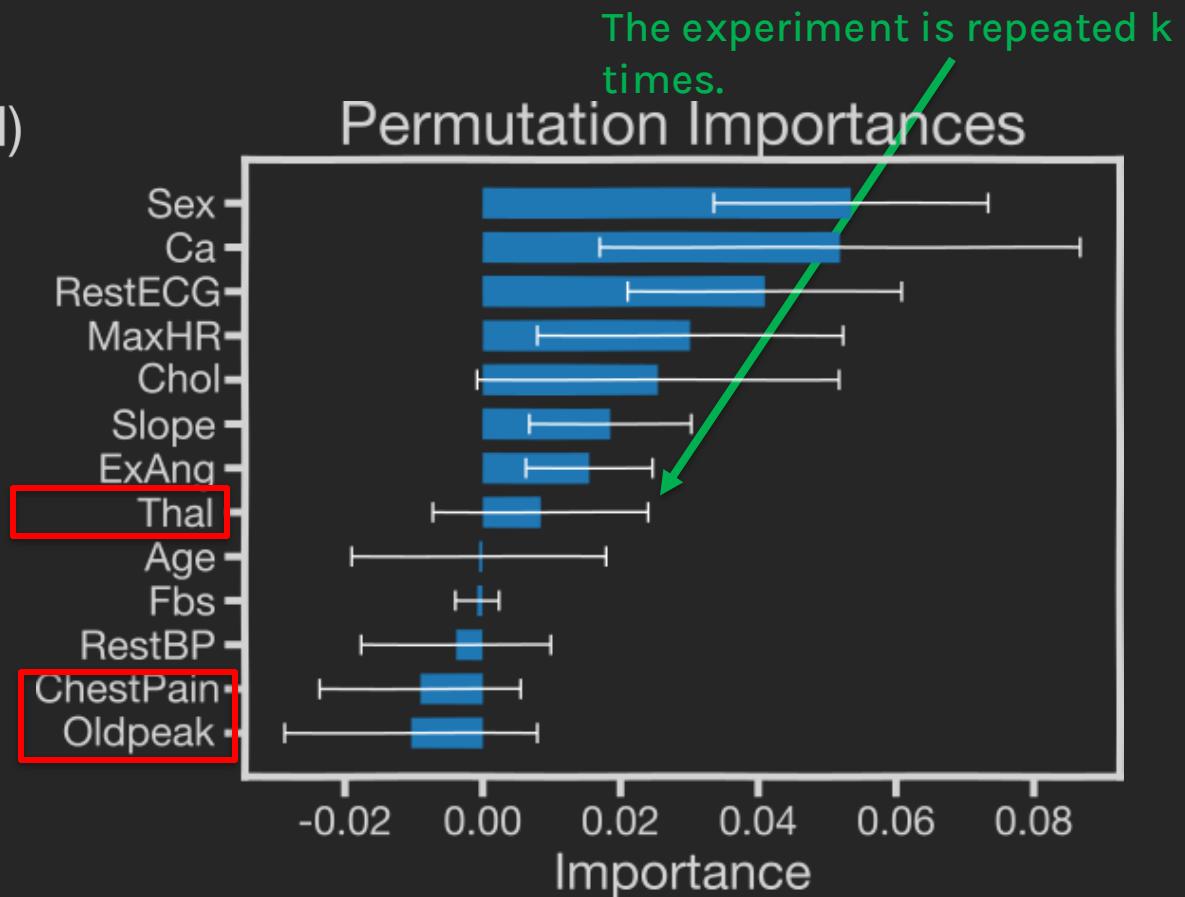
Features like ***ChestPain***, ***OldPeak***, ***Thal*** are ranked most important in MDI importance plot, but they are ranked low in permutation importance plot.

MDI vs Permutation Importance

Random Forest Feature Importances (MDI)



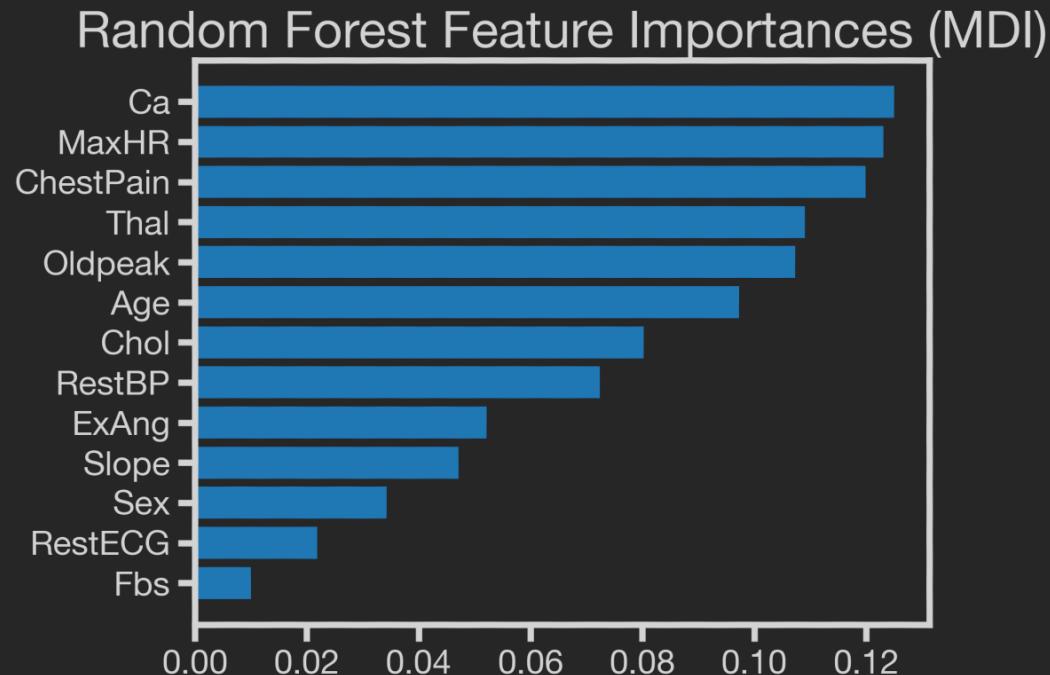
Permutation Importances



Features like ***ChestPain***, ***OldPeak***, ***Thal*** are ranked most important in MDI importance plot, but they are ranked low in permutation importance plot.

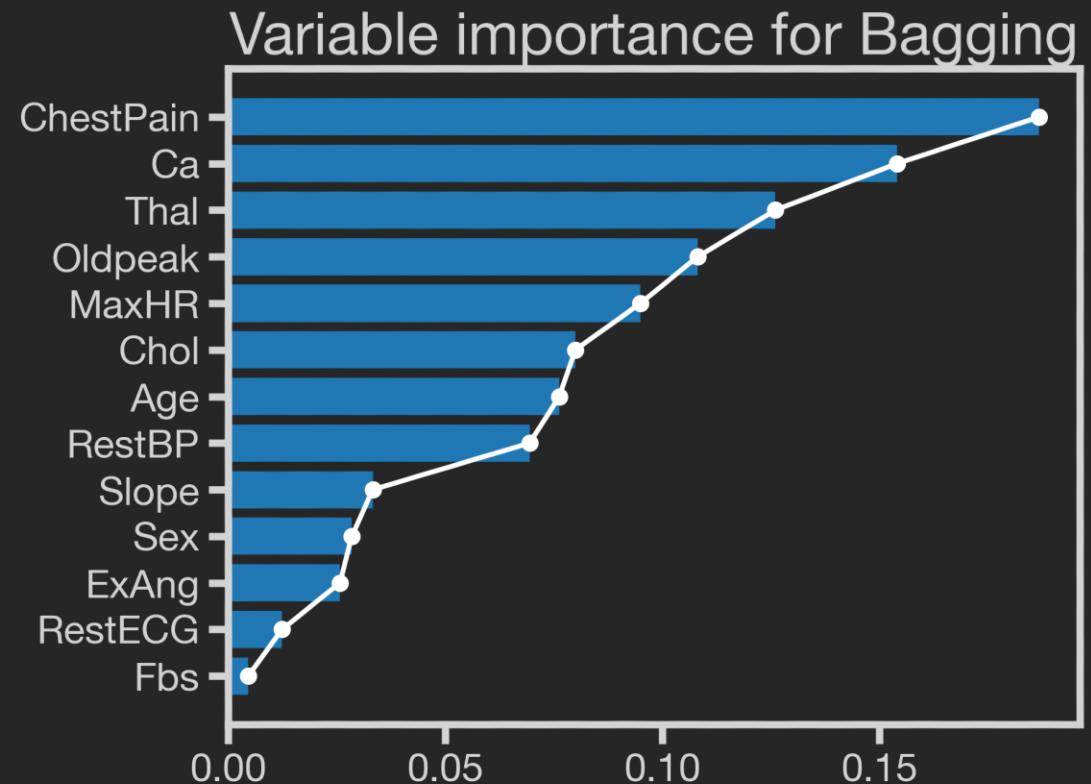
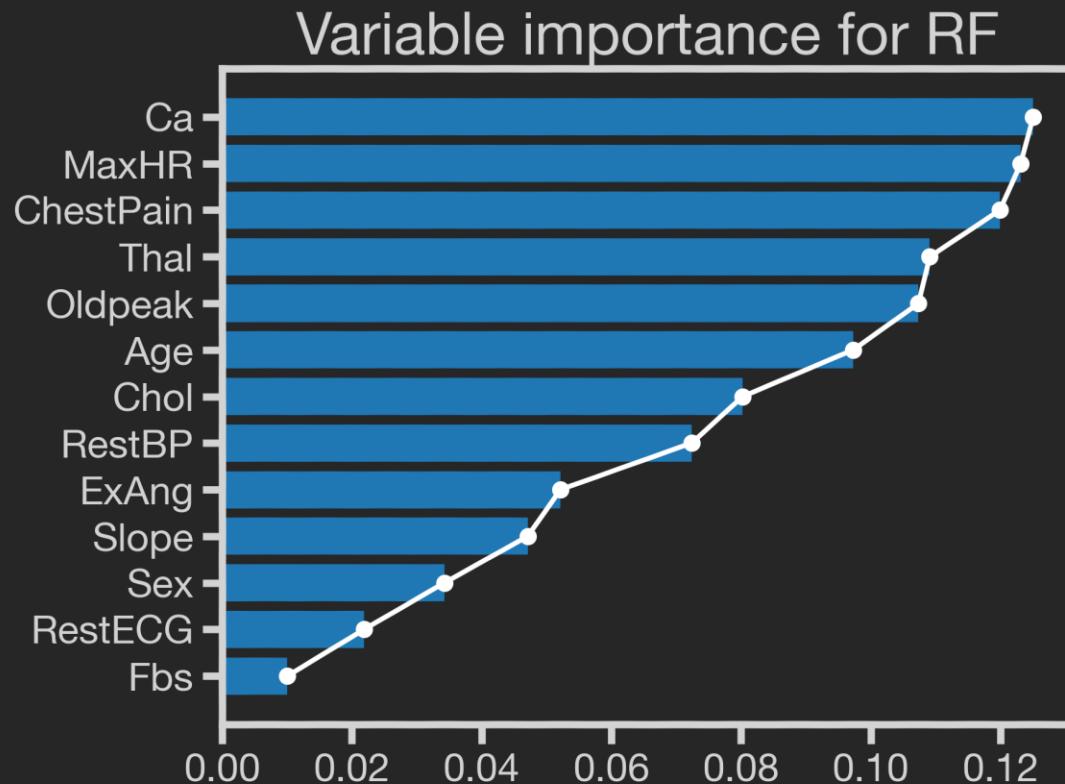
MDI vs Permutation Importance

- The biggest advantage of the MDI is **speed of computation**. All needed values are computed during the Random Forest training.
- The drawbacks of the method is its tendency to prefer (select as important) numerical features and categorical features with **high cardinality**. In the example shown, *Max HR* is selected as an important feature because of the high cardinality.



Variable Importance for bagging vs RF

Variable importance for RF is smoother than that for bagging due to randomness introduced by selecting a subset of predictors to choose from.

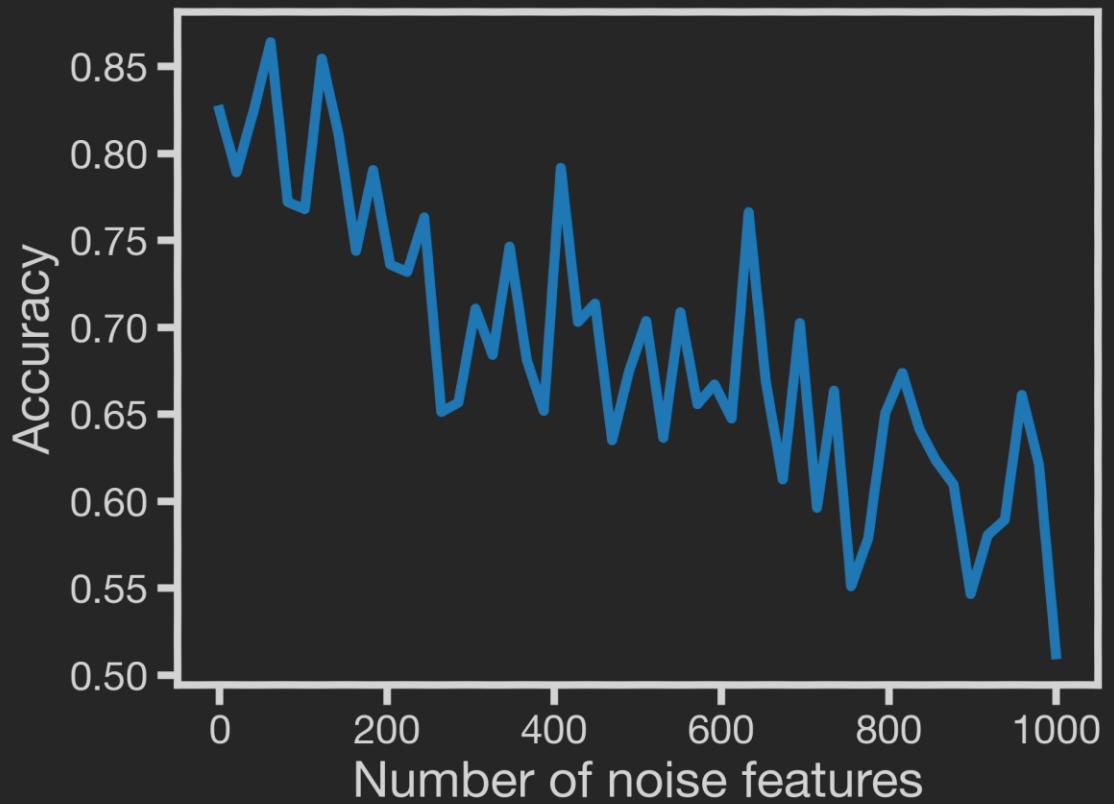


Final Thoughts on Random Forests

When the number of predictors is large, but the number of relevant predictors is small, random forests can perform poorly.

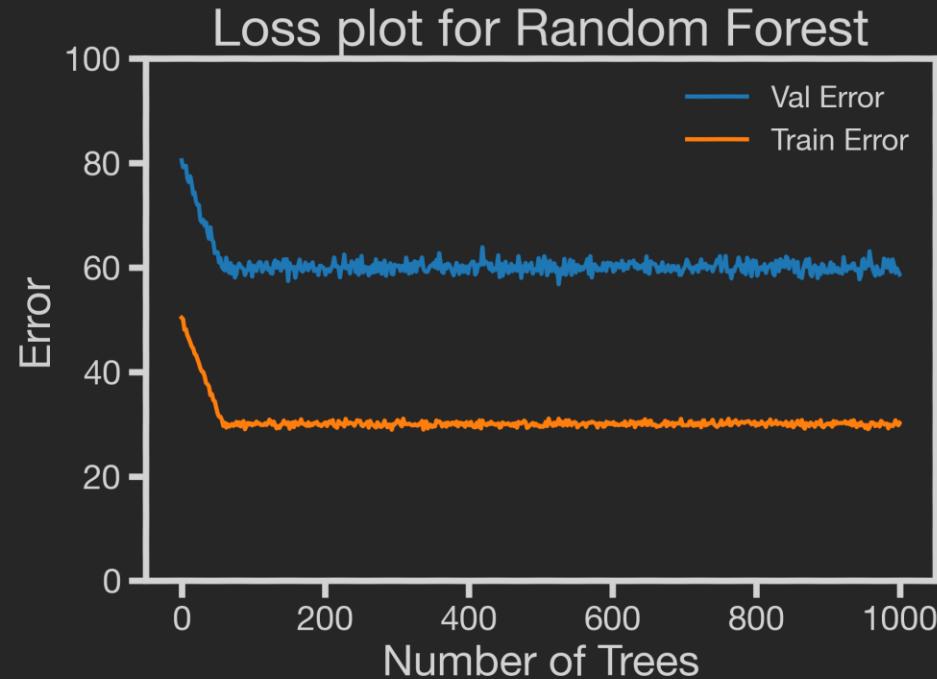
Question: Why?

In each split, the chances of selecting a relevant predictor will be low and hence most trees in the ensemble will be weak models.



Final Thoughts on Random Forests

Increasing the number of trees in the ensemble generally does not increase the risk of overfitting.



By decomposing the generalization error in terms of bias and variance, we see that increasing the number of trees produces a model that is at least as robust as a single tree.

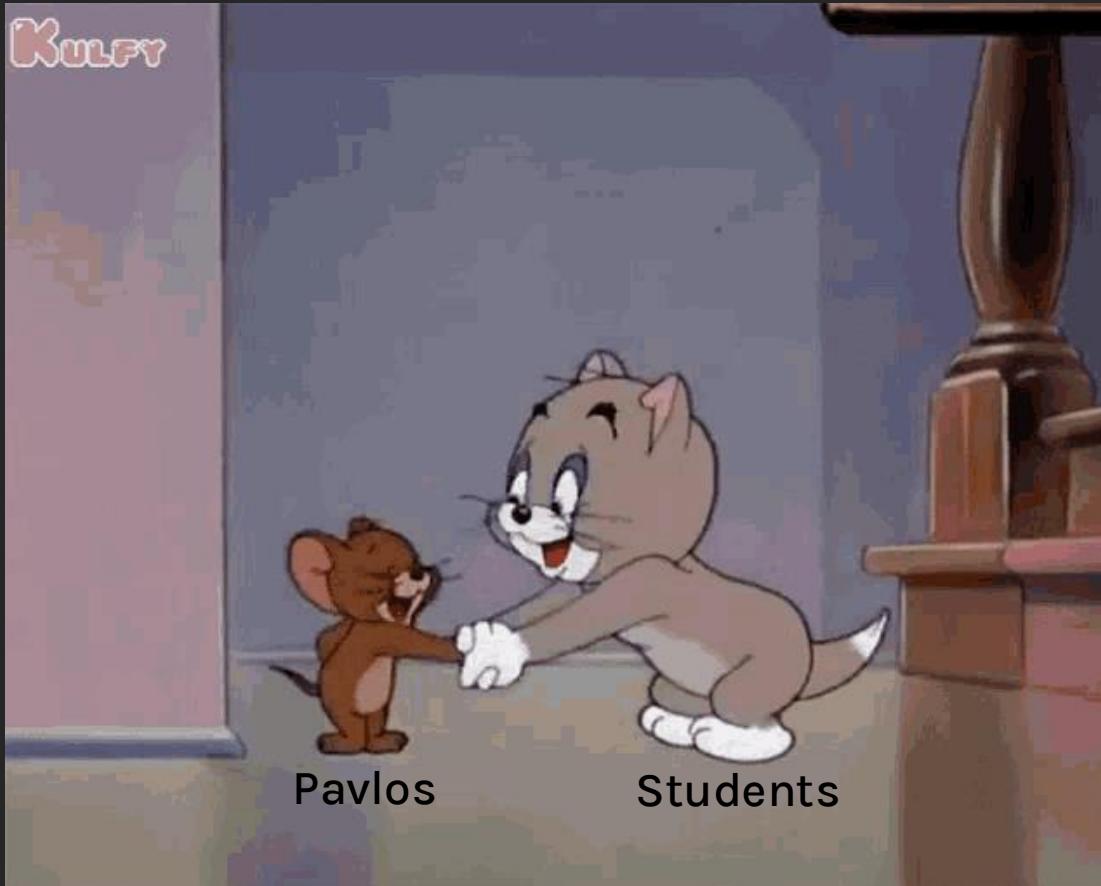
Final Thoughts on Random Forests

Random Forest Classifier (and bagging) can return probabilities.

Question: How?

The predicted class probabilities of an input sample is computed as the mean predicted class probabilities of the trees in the forest.

When you finally understand ‘Random Forest’



Thank you!





Missing Data

CS1090A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai, Chris Gumb



Samuel Tanner
Alaska

Outline

- Motivation
- Random Forest
- Variable Importance
- Missing Data (again)
- Class Imbalance
- Tree building algorithms

Missing Data: Example

Consider the below real-world dataset used to predict the presence of heart disease.

| Weight (kg) | Height (cm) | Chest Congested? | Good Blood Circulation? | Arteries Blocked? | Heart Disease |
|-------------|-------------|------------------|-------------------------|-------------------|---------------|
| 58.3 | 125.3 | No | No | No | No |
| 65.7 | 193.2 | Yes | No | Yes | Yes |
| NaN | 112 | No | Yes | No | No |
| 112 | 165.7 | No | No | NaN | Yes |
| 45 | 135 | No | No | No | No |
| 40 | 120 | No | No | No | Yes |

Missing Data: Example

Real-world datasets most often have **missing values**.

| Weight (kg) | Height (cm) | Chest Congested? | Good Blood Circulation? | Arteries Blocked? | Heart Disease |
|-------------|-------------|------------------|-------------------------|-------------------|---------------|
| 58.3 | 125.3 | No | No | | |
| 65.7 | 193.2 | Yes | No | | |
| NaN | 112 | No | Yes | | NO |
| 112 | 165.7 | No | No | NaN | Yes |
| 45 | 135 | No | No | No | No |
| 40 | 120 | No | No | No | Yes |

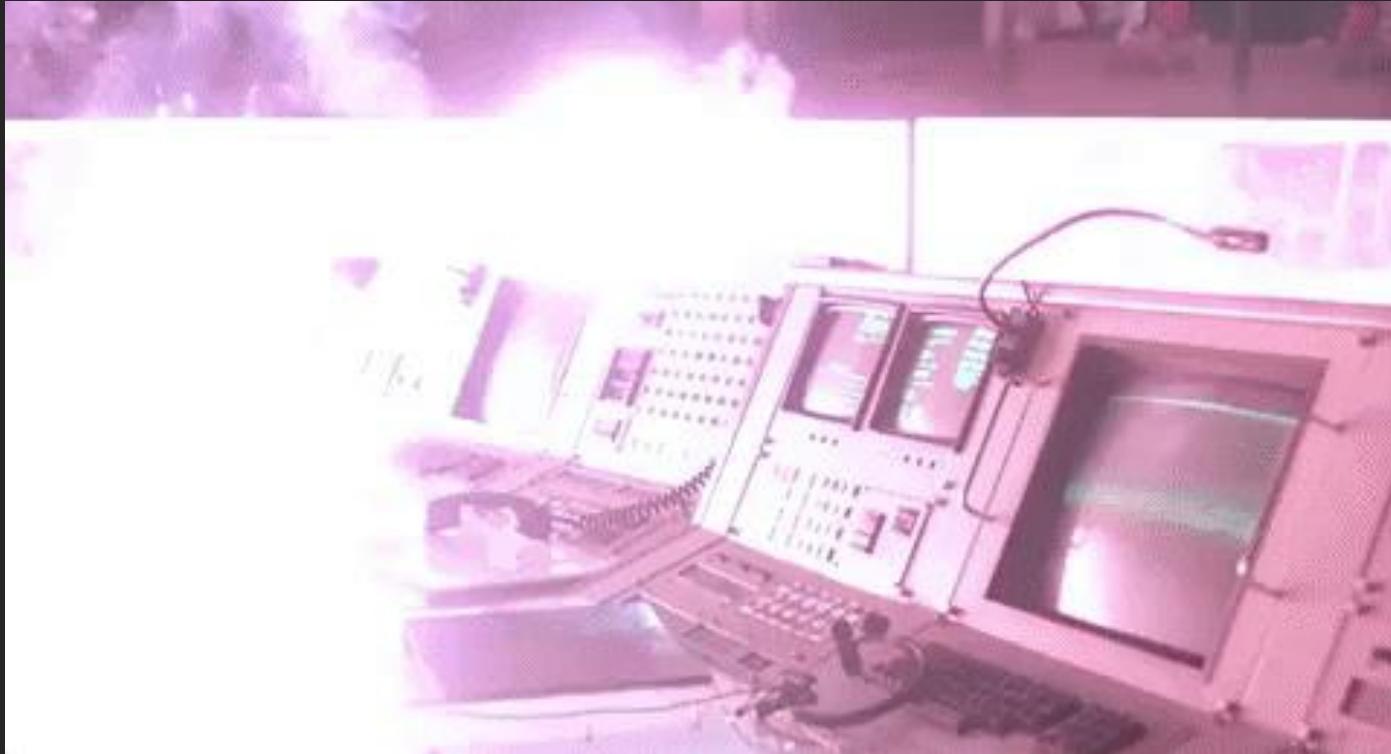
Missing values can occur due to a variety of reasons!

**Maybe someone forgot to fill out
the response sheet properly.**



I FORGOT!

...perhaps a device malfunction..



**...or maybe it was done
purposelv!**



*...or maybe it was done
purposely!*

BUT, HOW DO WE DEAL WITH MISSING DATA?

Missing Data: Example

There are many ways to deal with this missing data by performing imputation which we have learned.

However, in decision trees, we can handle missing values implicitly, which are called **surrogate splits!**

Introducing Surrogate Splits!



The basic idea is that during training, we find alternative splits, or “surrogate splits”, that can be used during prediction.

Missing Data: Decision Trees

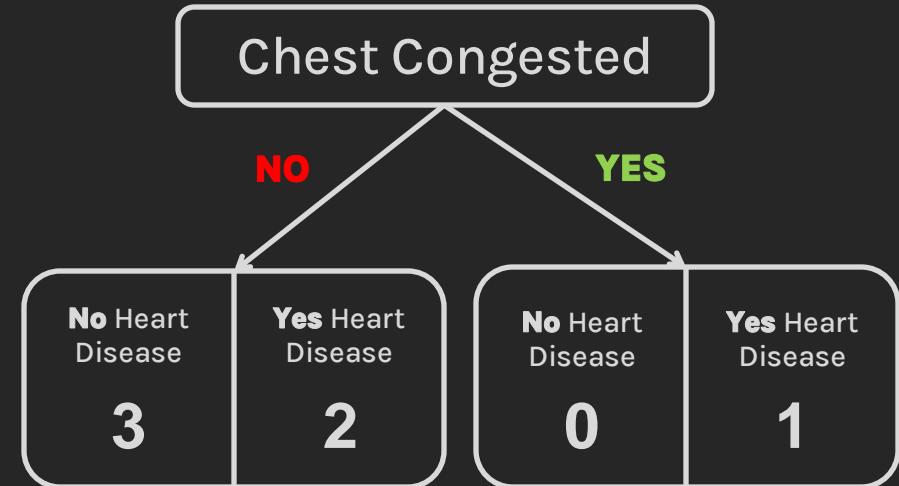
The basic idea is that during training, we find alternative splits, or “surrogate splits”, that can be used during prediction.

Missing Data: Decision Trees

As a first step, we do our usual tree thing.

Let's start with the predictor "Chest Congested" and split.

| Weight (kg) | Height (cm) | Chest Congested? | Good Blood Circulation? | Arteries Blocked? | Heart Disease |
|-------------|-------------|------------------|-------------------------|-------------------|---------------|
| 58.3 | 125.3 | No | No | No | No |
| 65.7 | 193.2 | Yes | No | Yes | Yes |
| 75.2 | 112 | No | Yes | No | No |
| 112 | 165.7 | No | No | Yes | Yes |
| 45 | 135 | No | No | No | No |
| 40 | 120 | No | No | No | Yes |



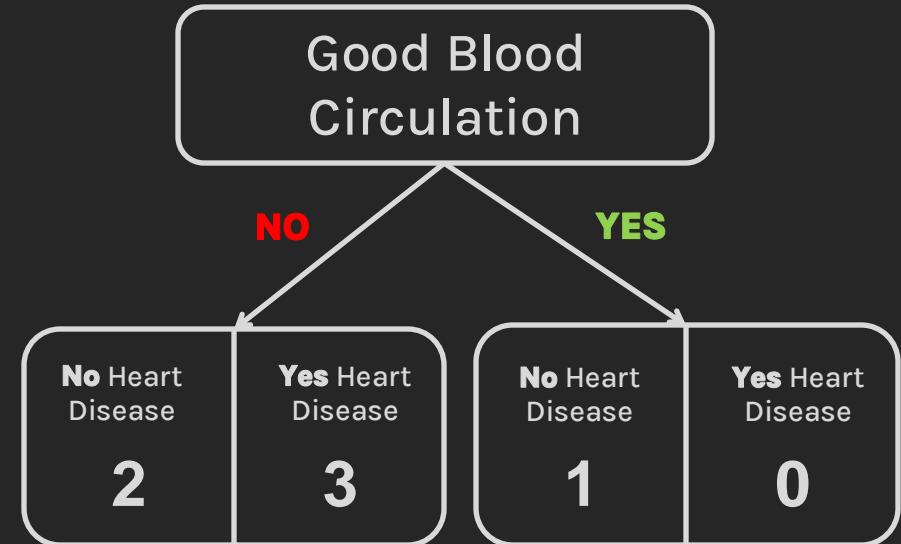
- WHEN "CHEST CONGESTED" IS FALSE, WE HAVE [3 NOs, 2 YESes].
- WHEN "CHEST CONGESTED" IS TRUE, WE HAVE [0 NOs, 1 YES].

We will be counting the number of "yes" or "no" in the response variable after each split.

Missing Data: Decision Trees

Then examine the split for “Good Blood Circulation”.

| Weight (kg) | Height (cm) | Chest Congested? | Good Blood Circulation? | Arteries Blocked? |
|-------------|-------------|------------------|-------------------------|-------------------|
| 58.3 | 125.3 | No | No | No |
| 65.7 | 193.2 | Yes | No | Yes |
| 75.2 | 112 | No | Yes | No |
| 112 | 165.7 | No | No | Yes |
| 45 | 135 | No | No | No |
| 40 | 120 | No | No | No |



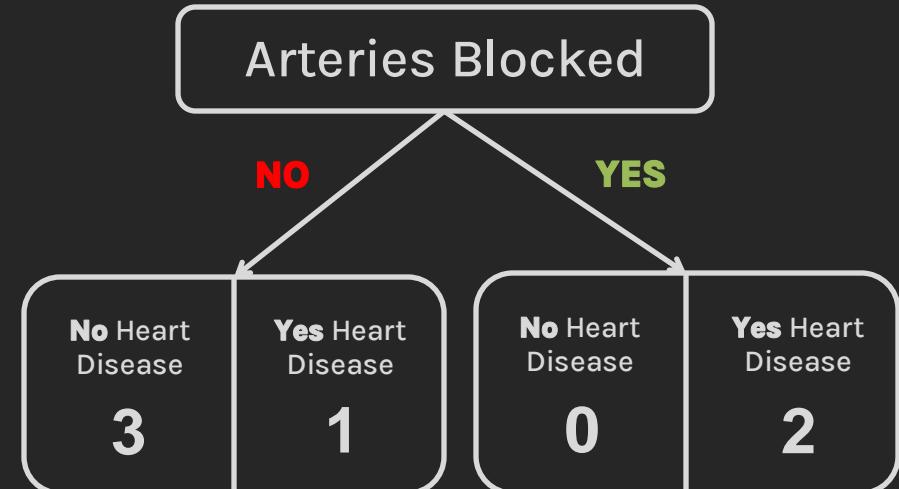
- WHEN "GOOD BLOOD CIRCULATION" IS FALSE, WE HAVE [2 NOs, 3 YESes].
- WHEN "GOOD BLOOD CIRCULATION" IS TRUE, WE HAVE [1 NO, 0 YESes]

Missing Data: Decision Trees

... for “Arteries Blocked”:

| Weight (kg) | Height (cm) | Chest Congested? | Good Blood Circulation? |
|-------------|-------------|------------------|-------------------------|
| 58.3 | 125.3 | No | No |
| 65.7 | 193.2 | Yes | No |
| 75.2 | 112 | No | Yes |
| 112 | 165.7 | No | No |
| 45 | 135 | No | No |
| 40 | 120 | No | No |

| Arteries Blocked? | Heart Disease |
|-------------------|---------------|
| No | No |
| Yes | Yes |
| No | No |
| Yes | Yes |
| No | No |
| No | Yes |



- WHEN "ARTERIES BLOCKED" IS FALSE, WE HAVE [3 NOs, 1 YES].
- WHEN "ARTERIES BLOCKED" IS TRUE, WE HAVE [0 NOs, 2 YESes].

Let's compare the split distribution of "*arteries blocked*" with the other two predictors: "*chest congested*" and "*good blood circulation*".

For "arteries blocked":

- When "blocked arteries" is false, we have [3 NOs, 1 YES].
- When "blocked arteries" is true, we have [0 NOs, 2 YESes].

For "chest congested":

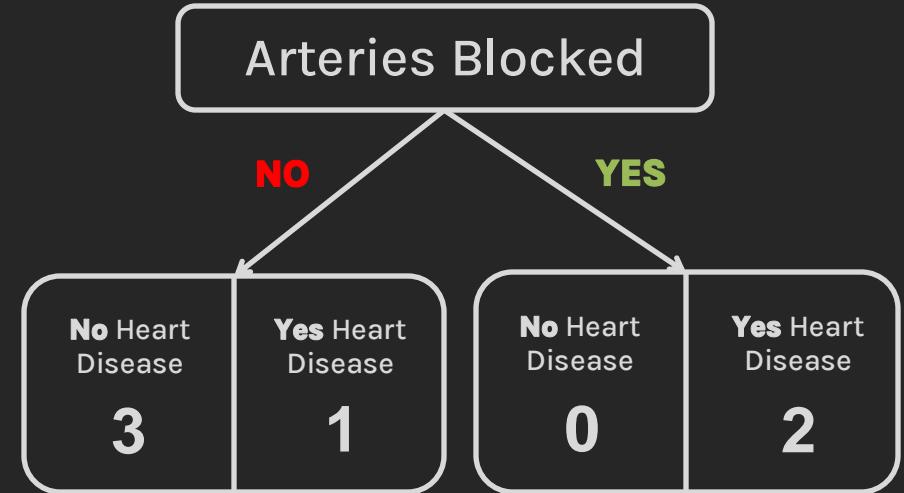
- When "chest congested" is false, we have [3 NOs, 2 YESes].
- When "chest congested" is true, we have [0 NOs, 1 YES].
- The difference in the distributions of chest-congested and blocked arteries is 2.

For "good blood circulation":

- When "good blood circulation" is false, we have [2 NOs, 3 YESes].
- When "good blood circulation" is true, we have [1 NO, 0 YESes].
- The difference in the distributions of good blood circulation

Which of these two closely resemble split for *Arteries Blocked*?

| Weight (kg) | Height (cm) | Chest Congested? | Good Blood Circulation? | Arteries Blocked? | Heart Disease |
|-------------|-------------|------------------|-------------------------|-------------------|---------------|
| 58.3 | 125.3 | No | No | No | No |
| 65.7 | 193.2 | Yes | No | Yes | Yes |
| Nan | 112 | No | Yes | No | No |
| 112 | 165.7 | No | No | Yes | Yes |
| 45 | 135 | No | No | No | No |
| 40 | 120 | No | No | No | Yes |



As “chest congested” shows the most similar split distribution to *arteries blocked*, it will be our second choice for a split during prediction if the value of *arteries blocked* is missing.

Missing Data: Decision Trees

Now that you understand the intuition behind it, what do you think is the surrogate for **Weight**?

| Weight (kg) | Height (cm) | | | | Heart Disease |
|-------------|-------------|-----|-----|-----|---------------|
| 58.3 | 125.3 | No | No | No | No |
| 65.7 | 193.2 | Yes | No | Yes | Yes |
| 75.2 | 112 | Yes | Yes | No | No |
| 112 | 165.7 | No | No | Yes | Yes |
| 45 | 135 | No | No | No | No |
| 40 | 120 | No | No | No | Yes |

The logical guess would
be height!

Missing Data: Decision Trees

During training, for every optimal split, we create a **rank** list of **surrogate splits**. This ranking is based on **similarity** between the split distributions of the optimal predictor and every other predictor.

Some Important Points about Surrogate Splits

- Surrogates can help us understand the primary splitter.
- Surrogates perform better when there is multi-collinearity.
- There is no guarantee that useful surrogates can be found!

Data Imbalance



CS1090A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai, Chris Gumb

Jose Garcia del Castillo
Hikkaduwa Beach, Sri Lanka

Outline

- Motivation
- Random Forest
- Variable Importance
- Missing Data (again)
- Class Imbalance
- Tree building algorithms

Class Imbalance

Training a RF (or any machine learning model) on an imbalanced dataset can introduce unique challenges to the learning problem.



Recap: F1-score

Accuracy is a great measure but only when you have **balanced datasets** (false negatives & false positives counts are close),

ALSO, **accuracy** is a good measure when **false negatives & false positives have similar costs**.

In the case of imbalance datasets, F1-score is a better metric

$$F1 = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}$$

Recap: Area Under the ROC curve

If the costs of false negatives & false positives are different, the ROC curve allows us to find the classification threshold that gives the best trade-off between FP rate and TP rate which we need in this case.

We summarize the ROC by computing the Area Under the ROC curve (AUC).



Dealing with Imbalanced classes

There are three main ways of dealing with imbalanced classes:
undersampling, oversampling and class weighting.

1. Undersampling

- i. Random Sampling

- ii. Near Miss

2. Oversampling

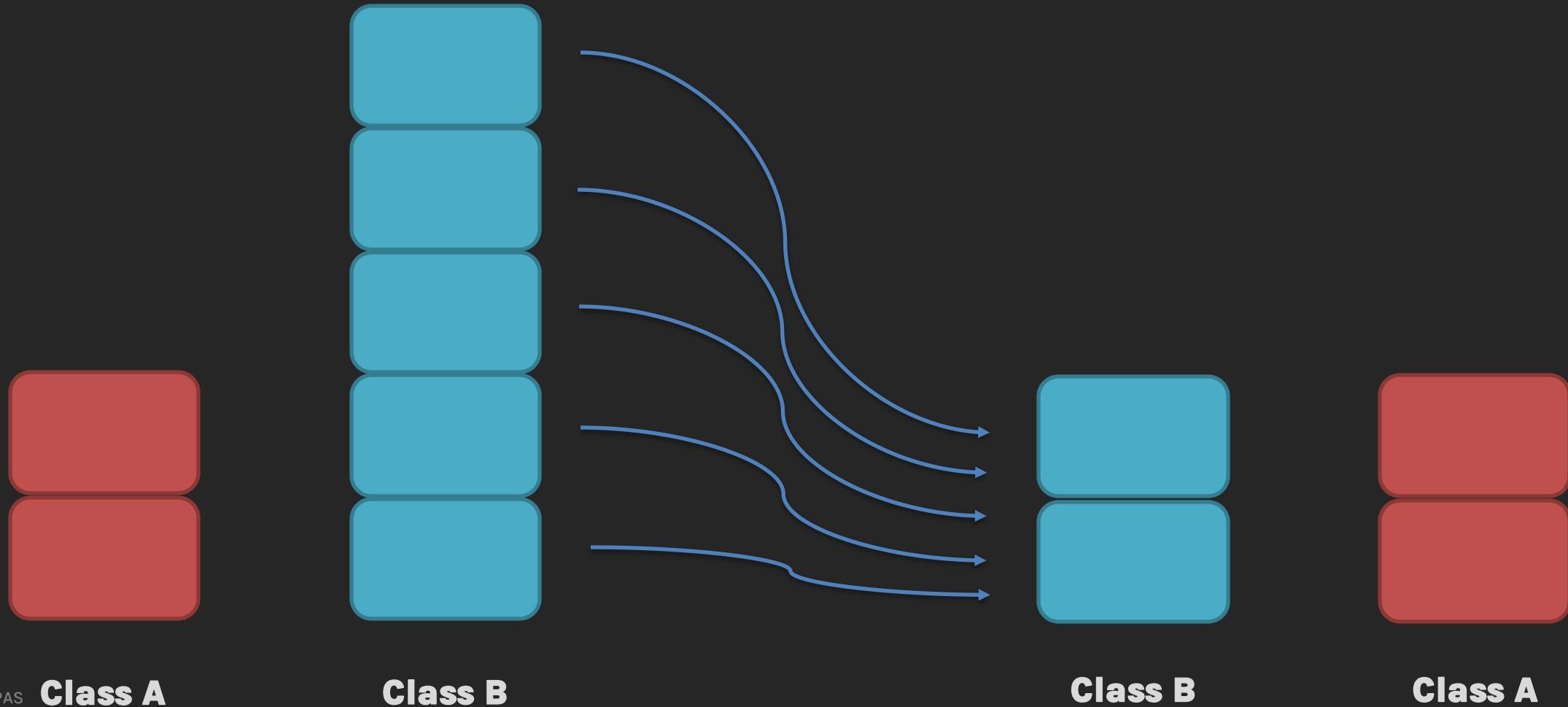
- i. Random Sampling

- ii. SMOTE

3. Class weighting

Dealing with Imbalanced classes

1. Undersampling



Dealing with Imbalanced classes

1. Undersampling

We **reduce** the number of samples in **majority class** to match the number of samples in minority class.

This can be done in two ways:

i. **Random Sampling:**

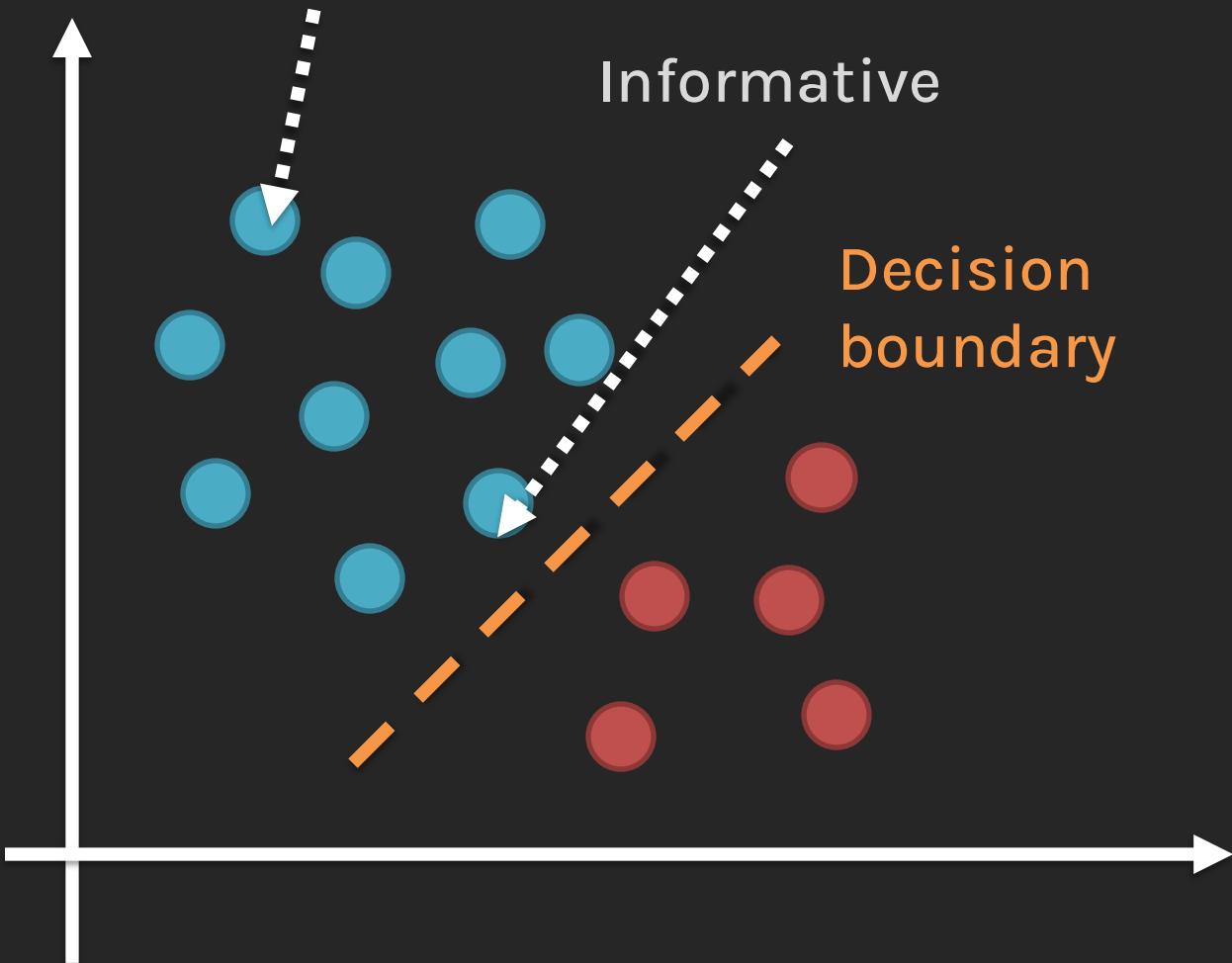
Randomly sample from majority class **with or without replacement**.

ii. **Near Miss:**

Select data points by using simple heuristics like finding samples from which the average distance to some data points of minority class is smallest. Read more about it [here](#).

Issue of random sampling

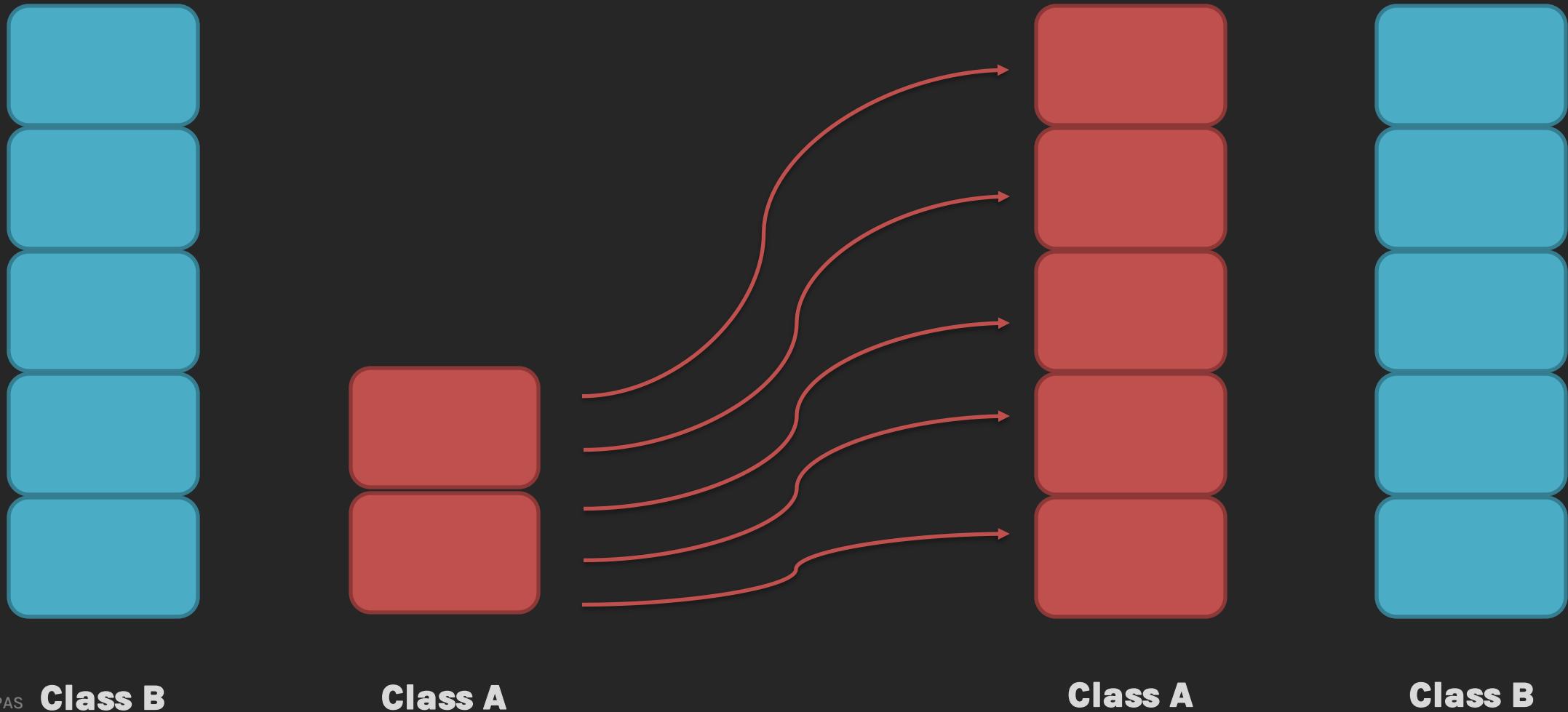
Not informative



- Random sampling can select data points that are not informative.
- Near miss, we can select more informative data points of the majority class; e.g., datapoints near the decision boundary in classification task.

Dealing with Imbalanced classes

2. Oversampling



Dealing with Imbalanced classes

2. Oversampling

We fight imbalanced data by generating new samples for minority class.

This can be done in two ways:

i. Random Sampling:

Randomly sample from minority class with replacement.

ii. SMOTE:

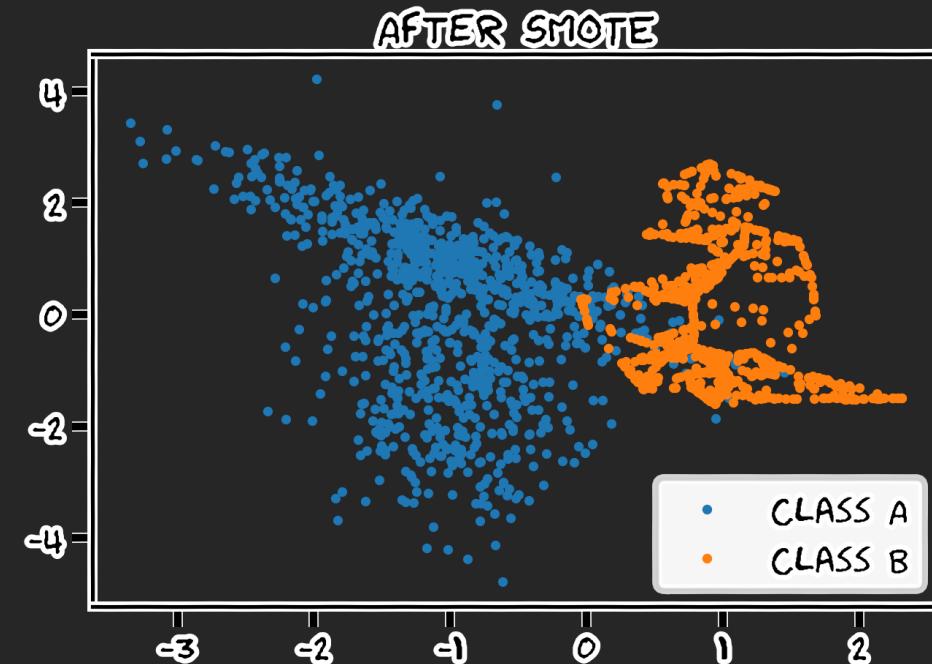
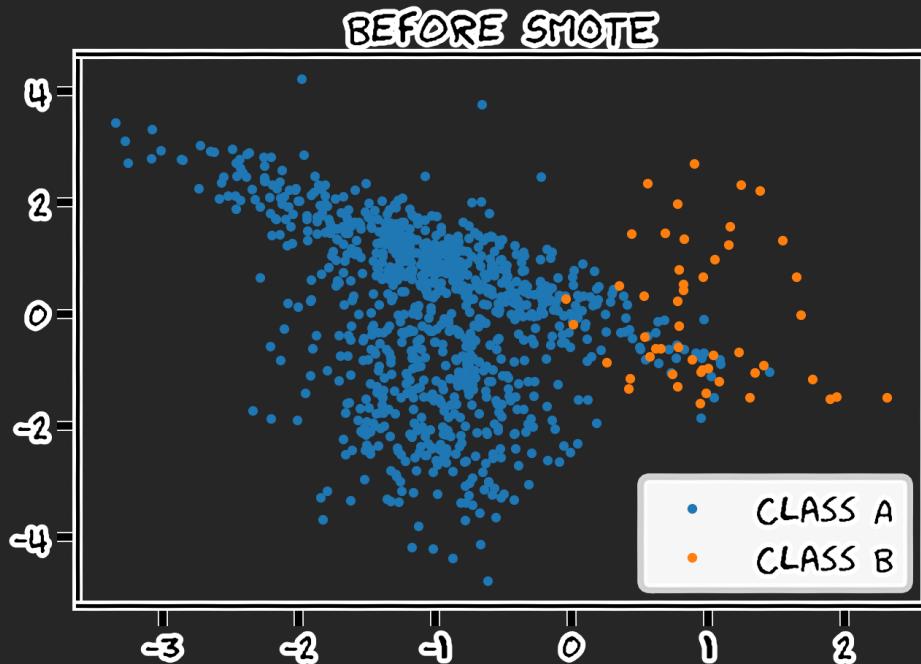
SMOTE is an improved alternative for oversampling.

SMOTE (Synthetic Minority Oversampling Technique):

ii. SMOTE:

SMOTE works by finding points that are closer in feature space.

Drawing a line between these points and generating new data points along this line.



Dealing with Imbalanced classes

3. Class weighting

A simple way to address the class imbalance is to provide a **weight** for each **class** which places more emphasis on the minority classes.

In sklearn we can provide the class weight as a dictionary or use
class_weight = balanced

Then it automatically adjust weights **inversely proportional** to class frequencies in the input data as:

$$W_k = \frac{N}{K \times N_K}$$

Where N is the total number of samples, N_k is the number of samples in class K and K is the total number of classes.

Outline

- Motivation
- Random Forest
- Variable Importance
- Missing Data (again)
- Class Imbalance
- Tree building algorithms [LAB TIME]

Thank you



Boosting, Gradient Boosting and AdaBoost

Introduction to Boosting

CS1090A Introduction to Data Science

Pavlos Protopapas, Natesh Pillai, Chris Gumb



Outline

- Introduction to Boosting
- Gradient Boosting
- Mathematical Formulation - Gradient Boosting

Outline

- **Introduction to Boosting**
- Gradient Boosting
- Mathematical Formulation - Gradient Boosting

Boosting



Apes together strong.

Anthony Goldbloom gives you the secret to winning Kaggle competitions

⌚ January 13, 2016 👤 Andrew Fogg 📁 Big Data

Kaggle has become the premier Data Science competition where the best and the brightest turn out in droves – Kaggle has more than 400,000 users – to try and claim the glory. With so many Data Scientists vying to win each competition (around 100,000 entries/month), prospective entrants can use all the tips they can get.

And who better than Kaggle CEO and Founder, Anthony Goldbloom, to dish out that advice? We caught up with him at Extract SF 2015 in October to pick his brain about how best to approach a Kaggle competition.

Anthony Goldbloom gives you the secret to winning Kaggle competitions

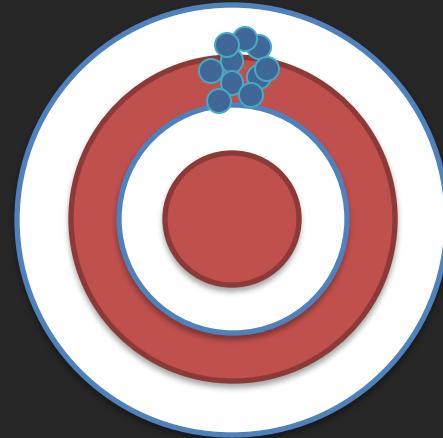
As long as Kaggle has been around, Anthony says, it has almost always been ensembles of decision trees that have won competitions.

It used to be random forest that was the big winner, but over the last six months a new algorithm called XGboost has cropped up, and it's winning practically every competition in the structured data category.

Recap: Decision Trees Issues

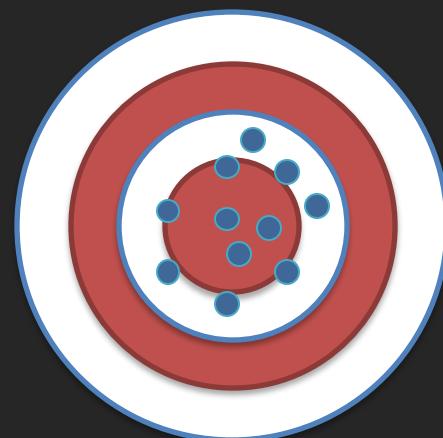
- Shallow trees:

Shallow trees (with very few leaves) suffer from high bias and do not train well.

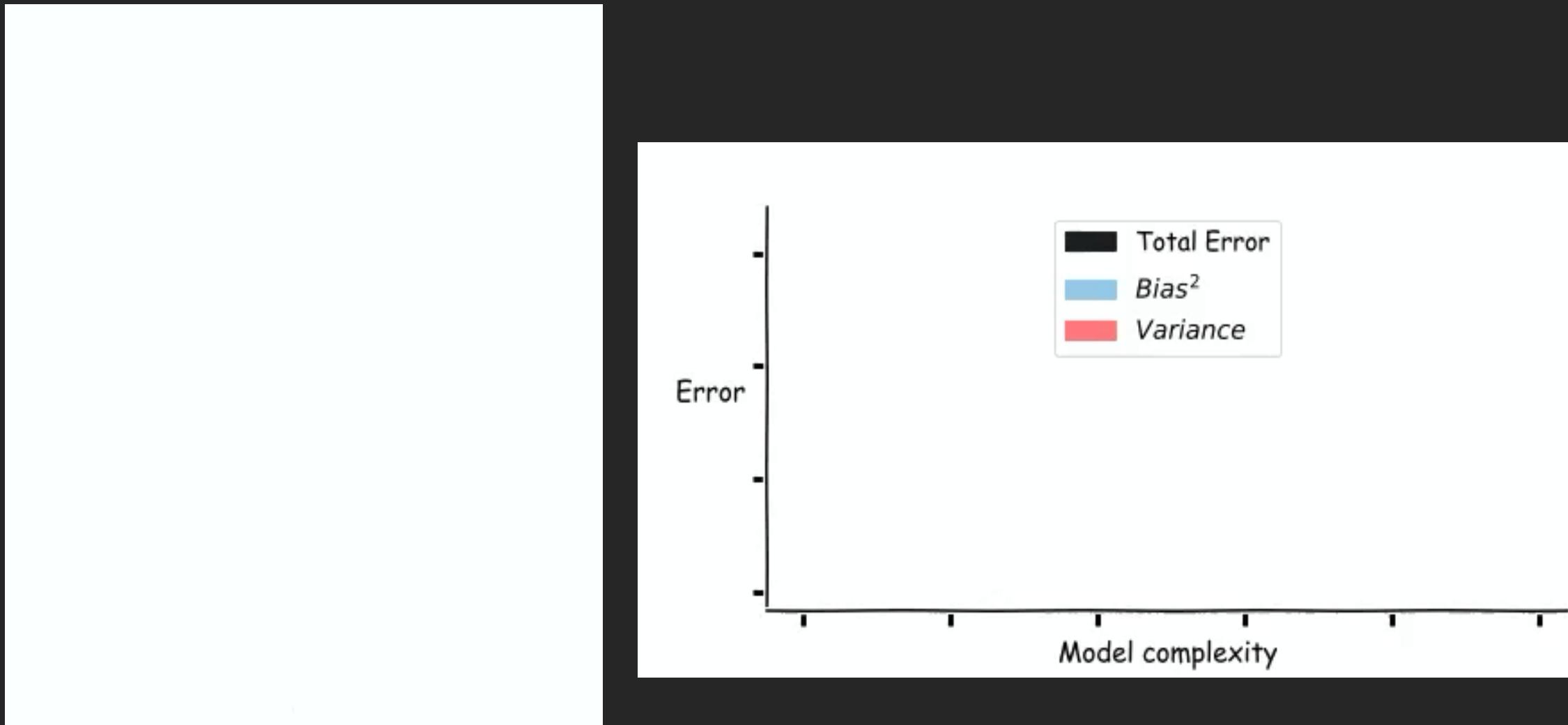


- Deep Trees:

Deep trees (with large number of nodes and leaves) have low bias but suffer from high variance leading to very low generalization error.



Recap: Decision Trees - Bias-Variance Trade-off



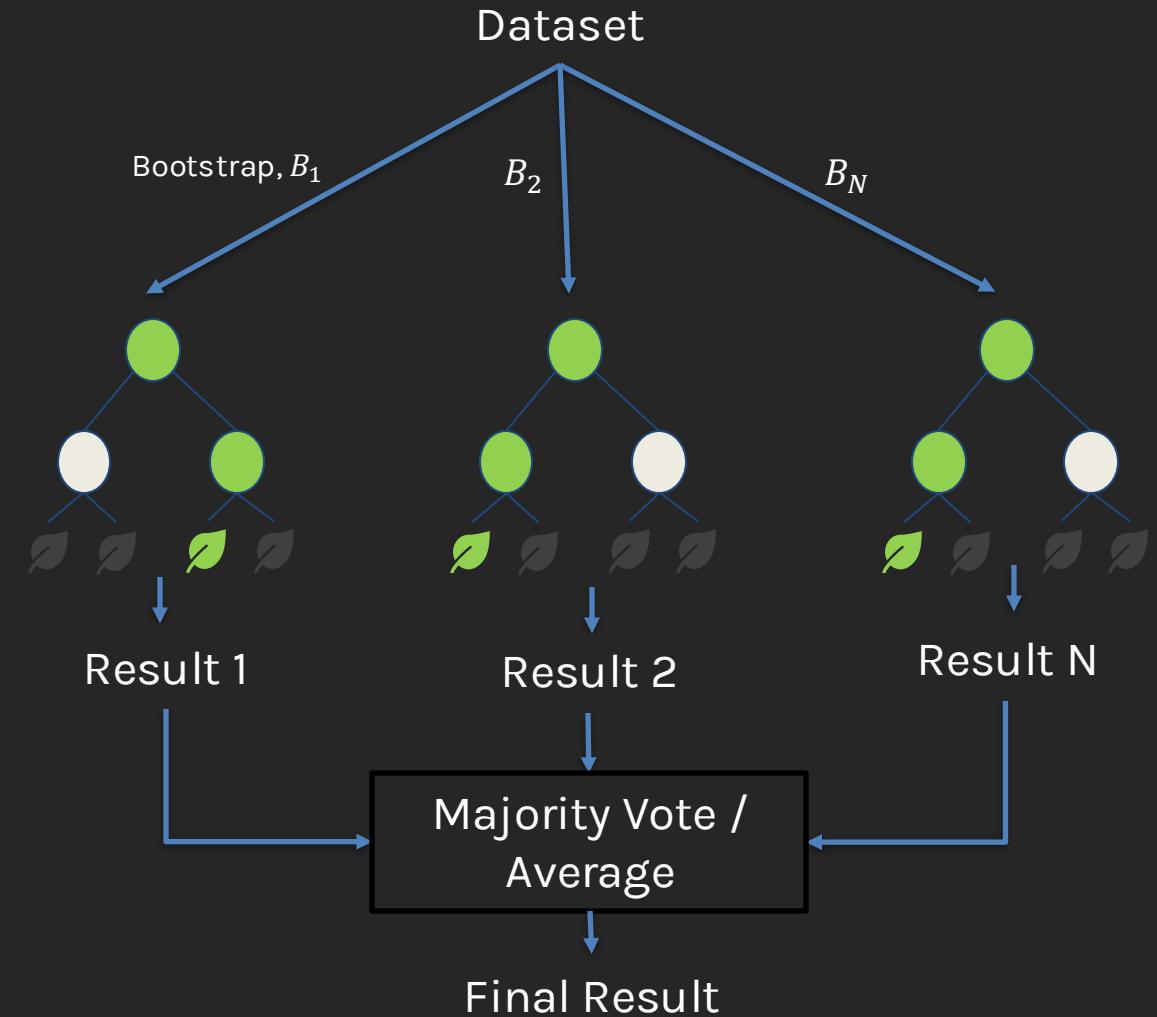
Random Forest Issues

- **Variance:**

Although variance reduction is better in RF than bagging, the generalization error is still high.

- **Inference Speed:**

Large number of trees can make the algorithm very slow and ineffective for real-time predictions.



Motivation for Boosting

Question: Could we address the shortcomings of single decision tree models in some other way?

For example, rather than performing **variance reduction** on complex trees, can we **decrease the bias** of simple trees - make them more expressive?

Can we learn from our mistakes?

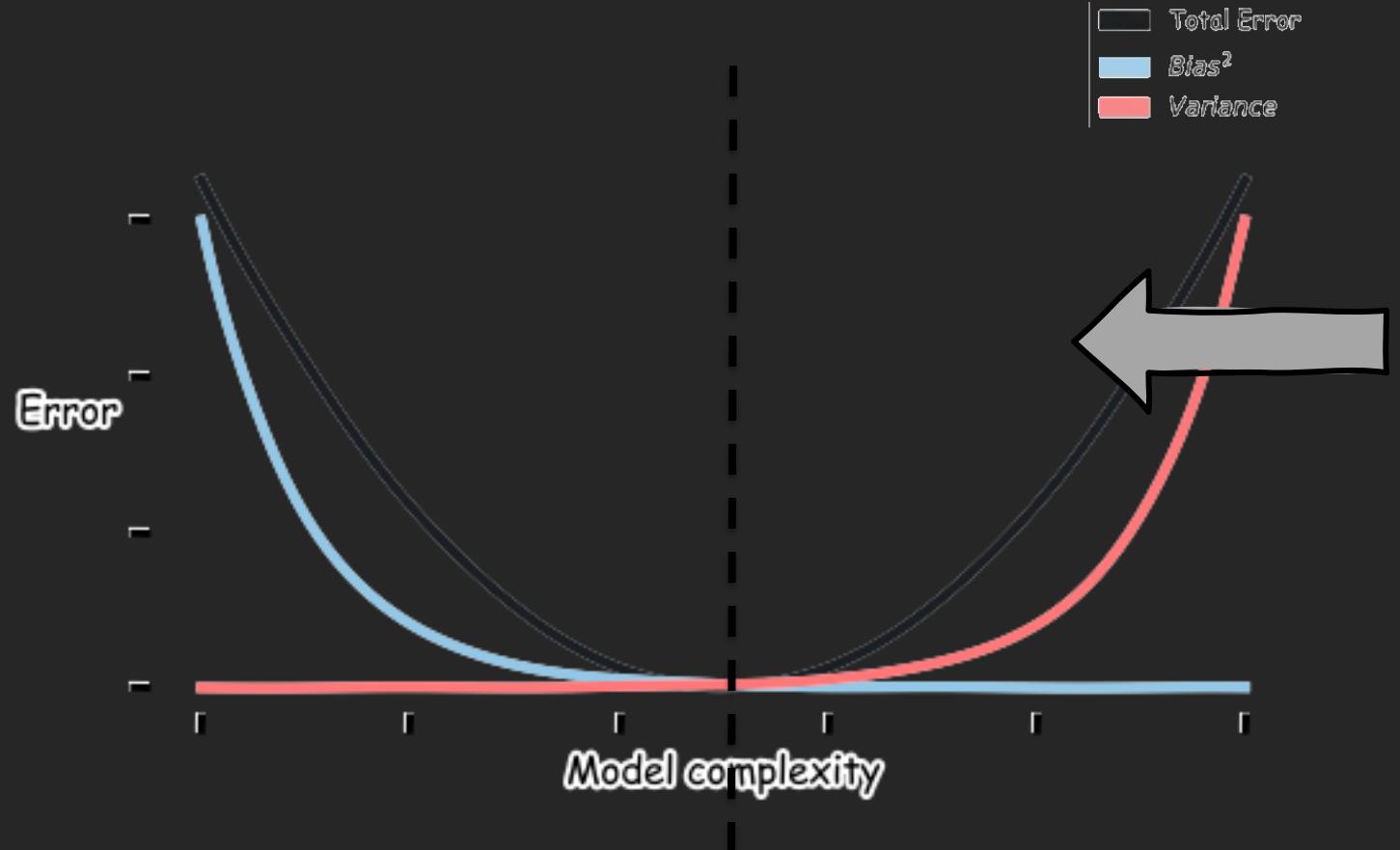
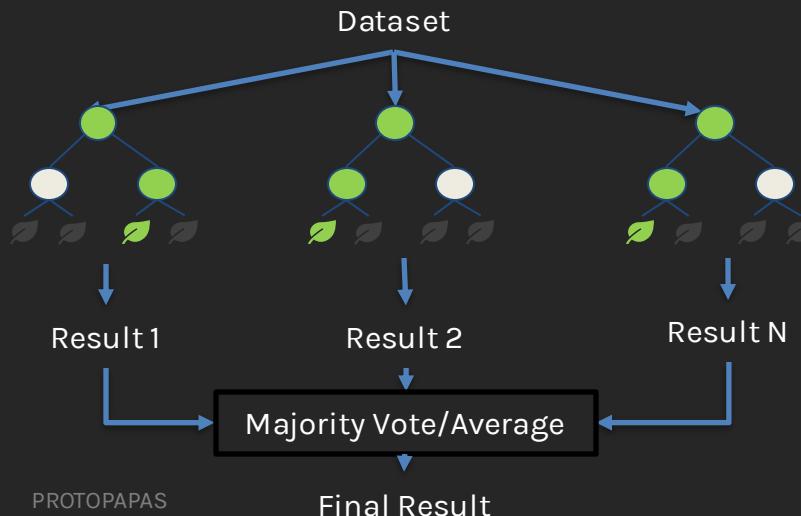
A solution to this problem, making an expressive model from simple trees, is another class of ensemble methods called **boosting**.

Random Forest - The only solution ?

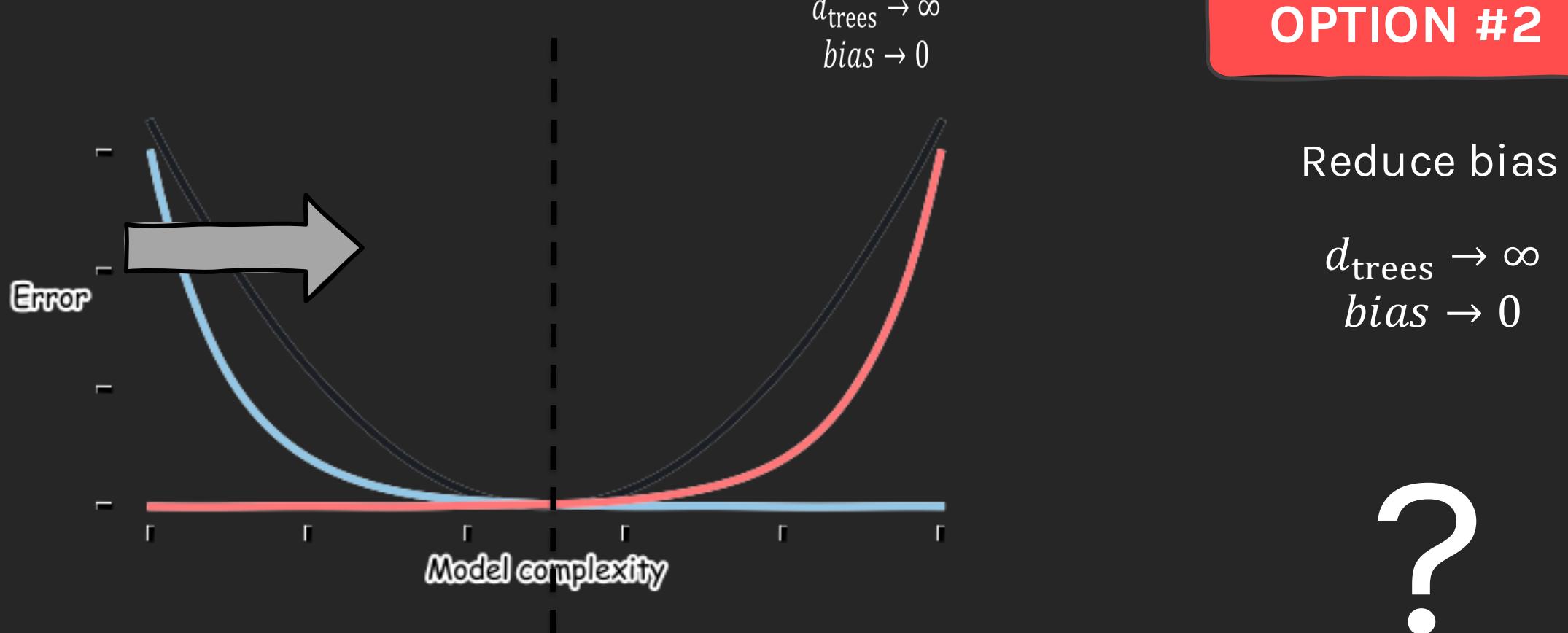
OPTION #1

Reduce variance

$$d_{\text{trees}} \rightarrow \infty$$
$$\text{var} \rightarrow 0$$



Random Forest - The only solution ?



Boosting

NEW IDEA 

- **Boosting** methods are general algorithms which combine several "**weak learners**" to produce a strong rule.
- The first implementation of Boosting was '**Adaboost**' invented by Robert Schapire and Yoav Freund in 1996.
- Boosting algorithms are fast, easy to compute and very accurate and are the de-facto optimization tree algorithms.



Rob Schapire & Yoav Freund

How does boosting work?

FINAL EXAM

TOPIC: BOOSTING DATE: DEC-11-2024

Q1:

Q2:

Q3:

...

Q10:

final score

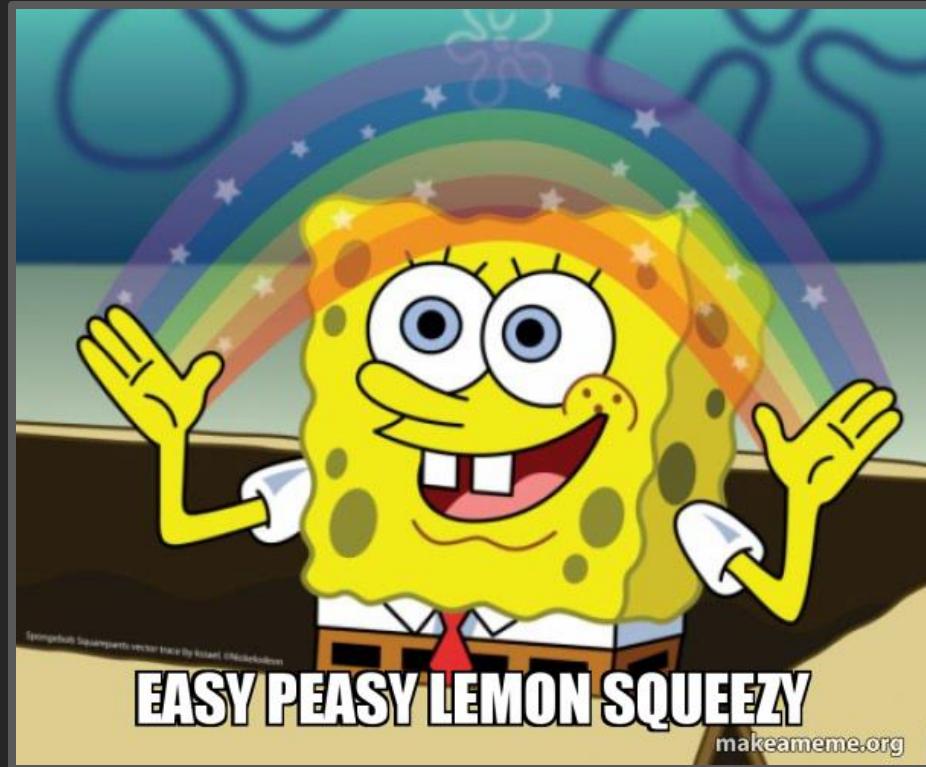
Passing grade is A



How does boosting work?

OPTION #1

1. Steal a time machine.
2. Go back to 1996 and meet Rob Schapire and Yoav Freund.
3. Befriend them by giving them stock trading tips from the future.
4. Follow their work for at least a decade to understand everything about boosting.
5. Return to the present and nail the test.
6. Repeat for another test.

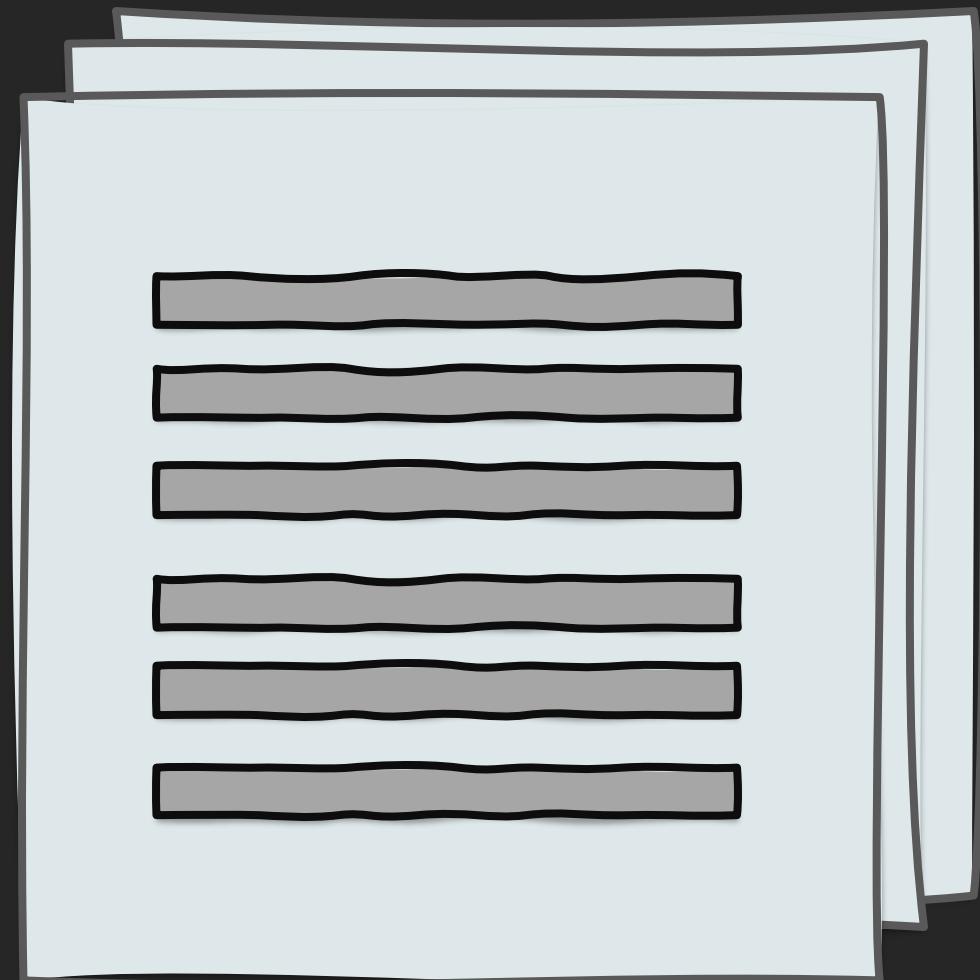


How does boosting work?

OPTION #2

STEP #1:

Go to the library and get
previous year question
papers.



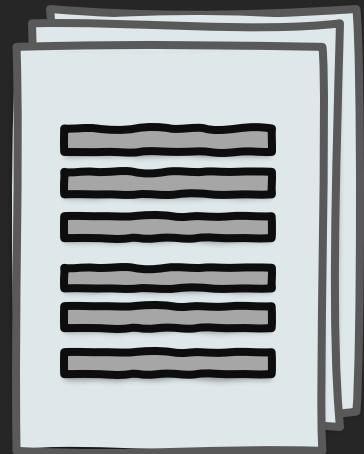
How does boosting work?

OPTION #2

STEP #2:

Find a helpful student
and ask them to give you
a "rule of thumb" to get at
least some answers right.

Don't ever choose option
D. Like never!



How does boosting work?

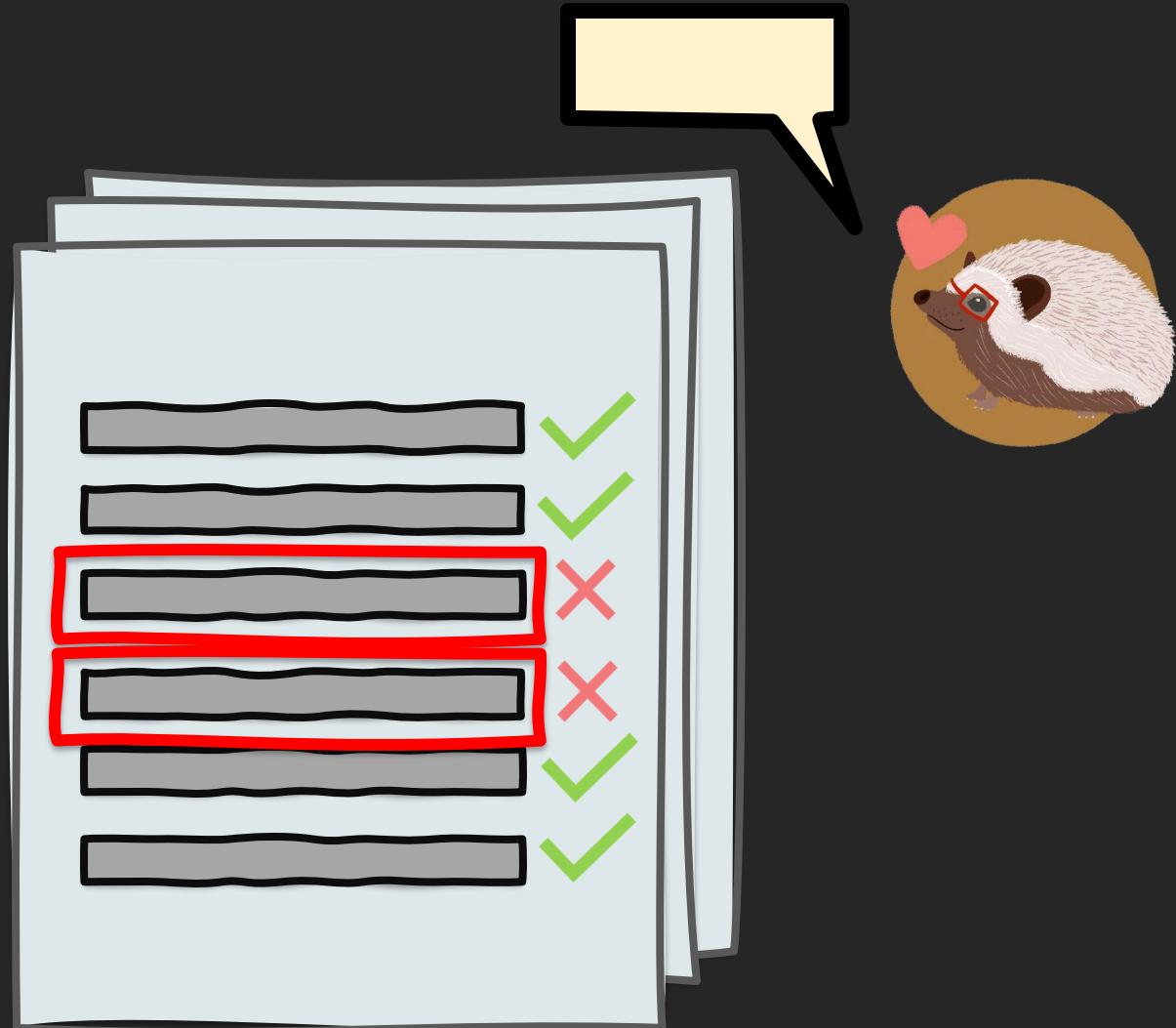
OPTION #2

Does the “rule” work?

Test out the rule.

It worked 60% of the time.

Not bad!!



How does boosting work?

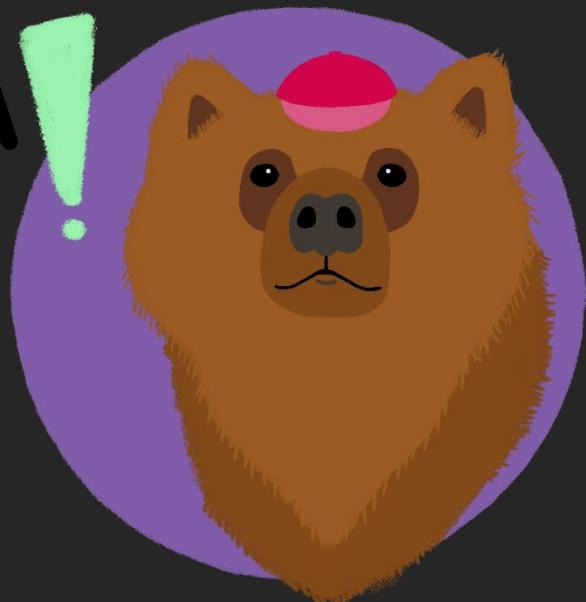
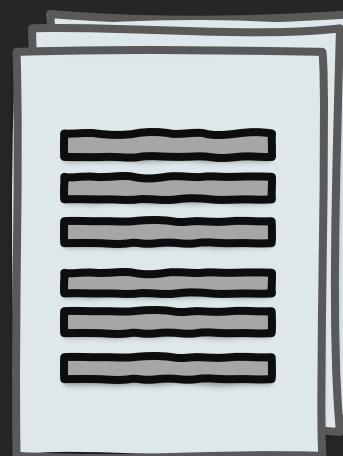
OPTION #2

STEP #3:

Find a TA and ask them to also give you a "rule of thumb" to get at least some answers right.

Make sure to focus on the ones you got wrong before!

If you see overfitting in the options, that's the right answer!



How does boosting work?

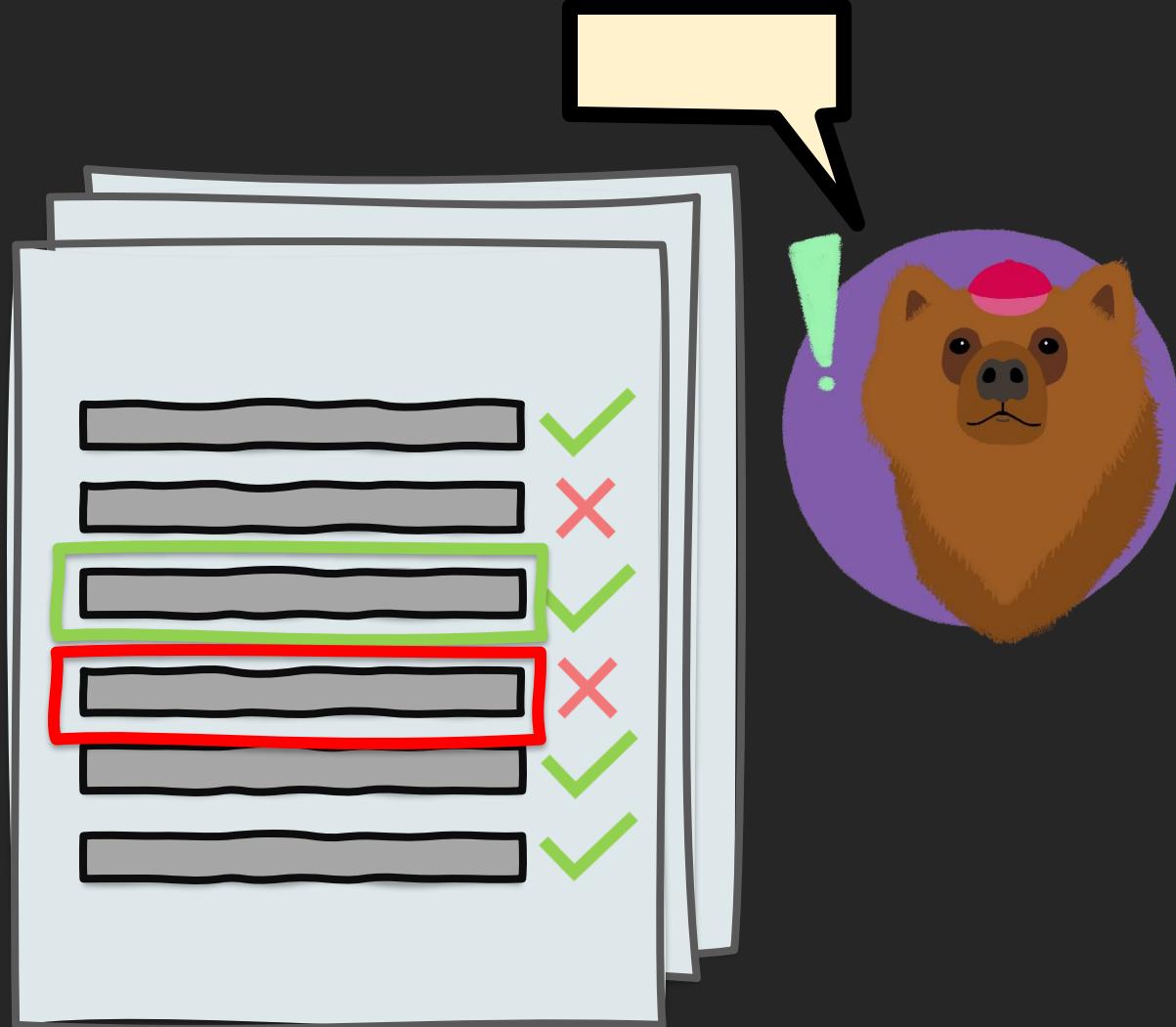
OPTION #2

Does the “rule” work?

Test out the new rule.

It works well on difficult problems!

But a few problems persist.



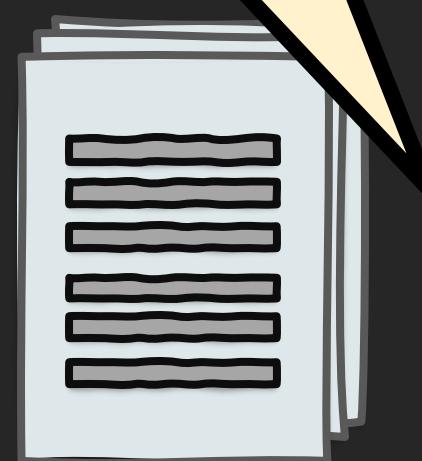
How does boosting work?

OPTION #2

STEP #4:

Call your favorite professor and focus on the ones you got wrong before!

The right answer is almost always cross-validation



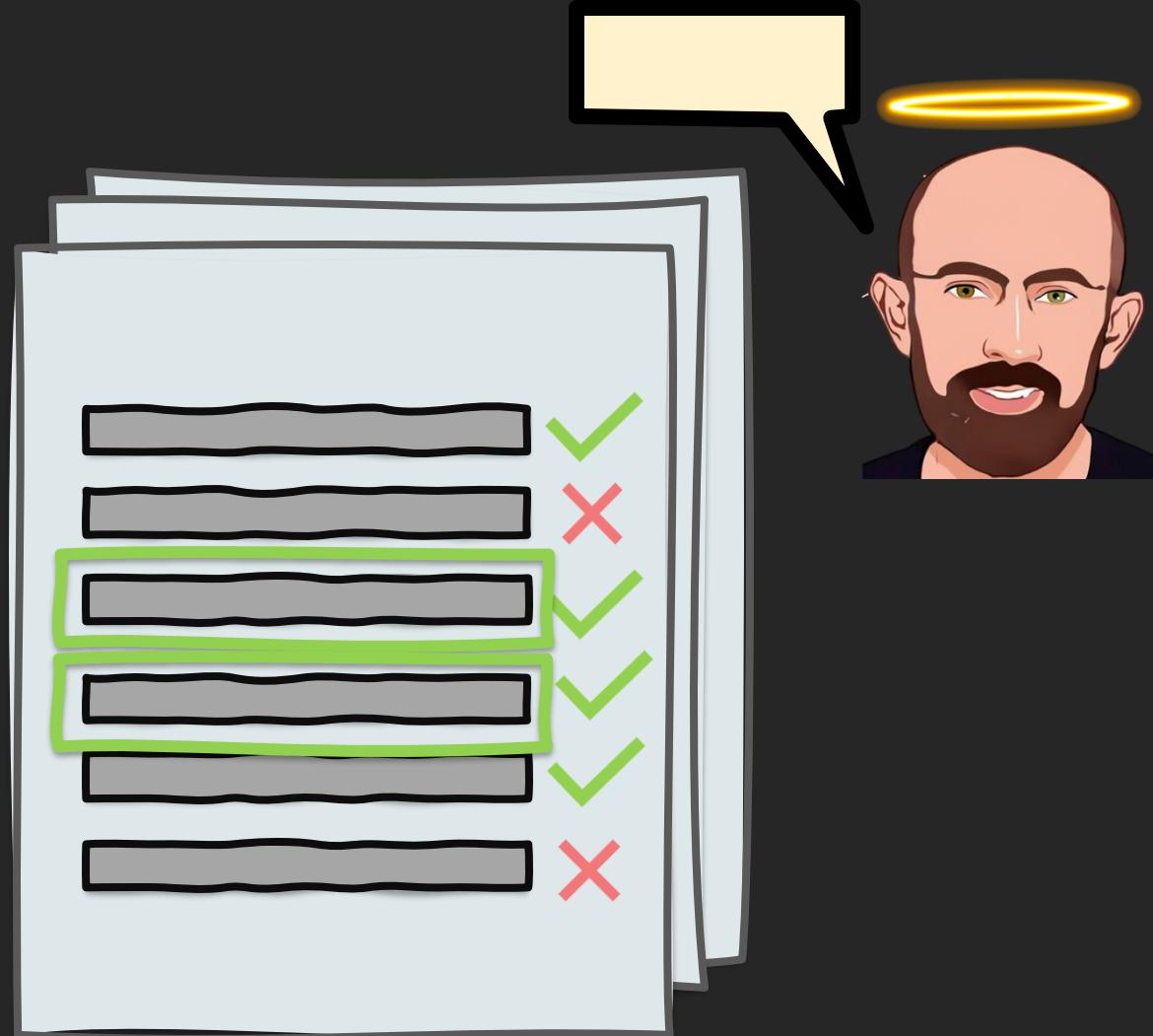
How does boosting work?

OPTION #2

Does the “rule” work?

Test out the new rule.

The new rule works well on
the difficult problems!

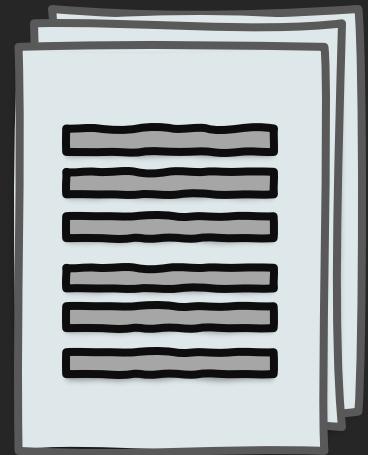


How does boosting work?

OPTION #2

STEP #5:

Combine the rules, but pay more **attention** to the ones that were more often right.



Accuracy: 60%



Accuracy: 64%



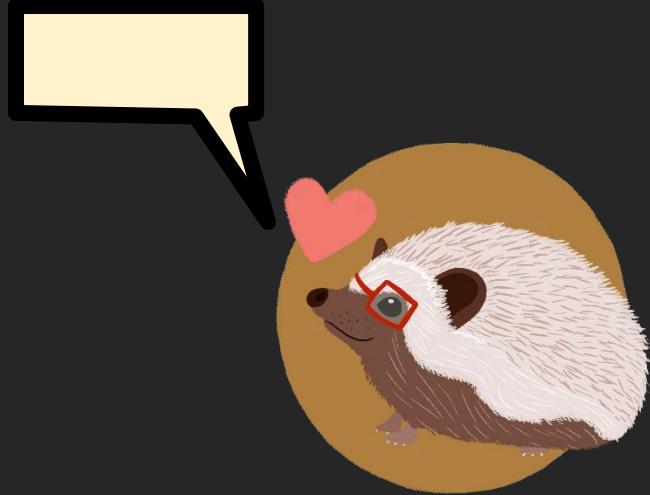
Accuracy: 70%



How does boosting work?

OPTION #2

Accuracy: 60%



Accuracy: 64%



Accuracy: 70%



$$Strategy = \alpha * Rule_1 + \beta * Rule_2 + \gamma * Rule_3$$

How does boosting work?

OPTION #2

FINAL STEP:

Take the test with these approximate rules, **weighted** by how well each rule performed.

A+



memegenerator.net

But how does it work?

Boosting, Gradient Boosting and AdaBoost

Gradient Boosting

CS1090A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai, Chris Gumb



Santiago Becerra
Agafay Dessert, Morocco

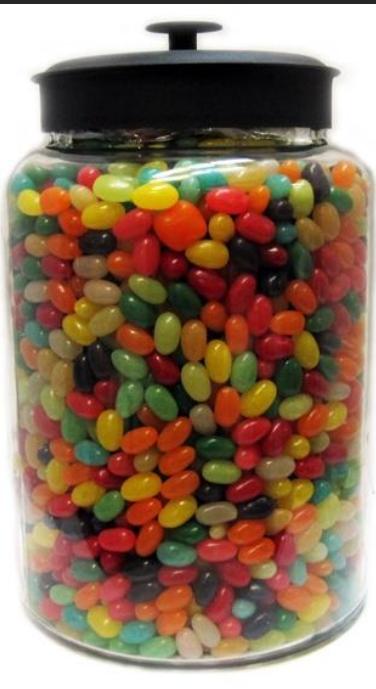
Outline

- Introduction to Boosting
- **Gradient Boosting**
- Mathematical Formulation - Gradient Boosting

Recap: Boosting

"Can a set of weak learners create a single strong learner?"

Leslie Gabriel Valiant



How many jellybeans do you see?

Recap: Boosting

The key intuition behind boosting is that one can take an ensemble of simple models $\{T_h\}_{h \in H}$ and **additively** combine them into a single, more complex model. Here, h is a weak learner and H is the hypothesis space of all possible weak learners.

Each model T_h might be a poor fit for the data, but a **linear combination** of the ensemble

$$T = \sum_h \lambda_h T_h$$

can be **expressive** and **flexible**.

Question: But which models should we include in our ensemble? What should the coefficients or weights in that linear combination be?

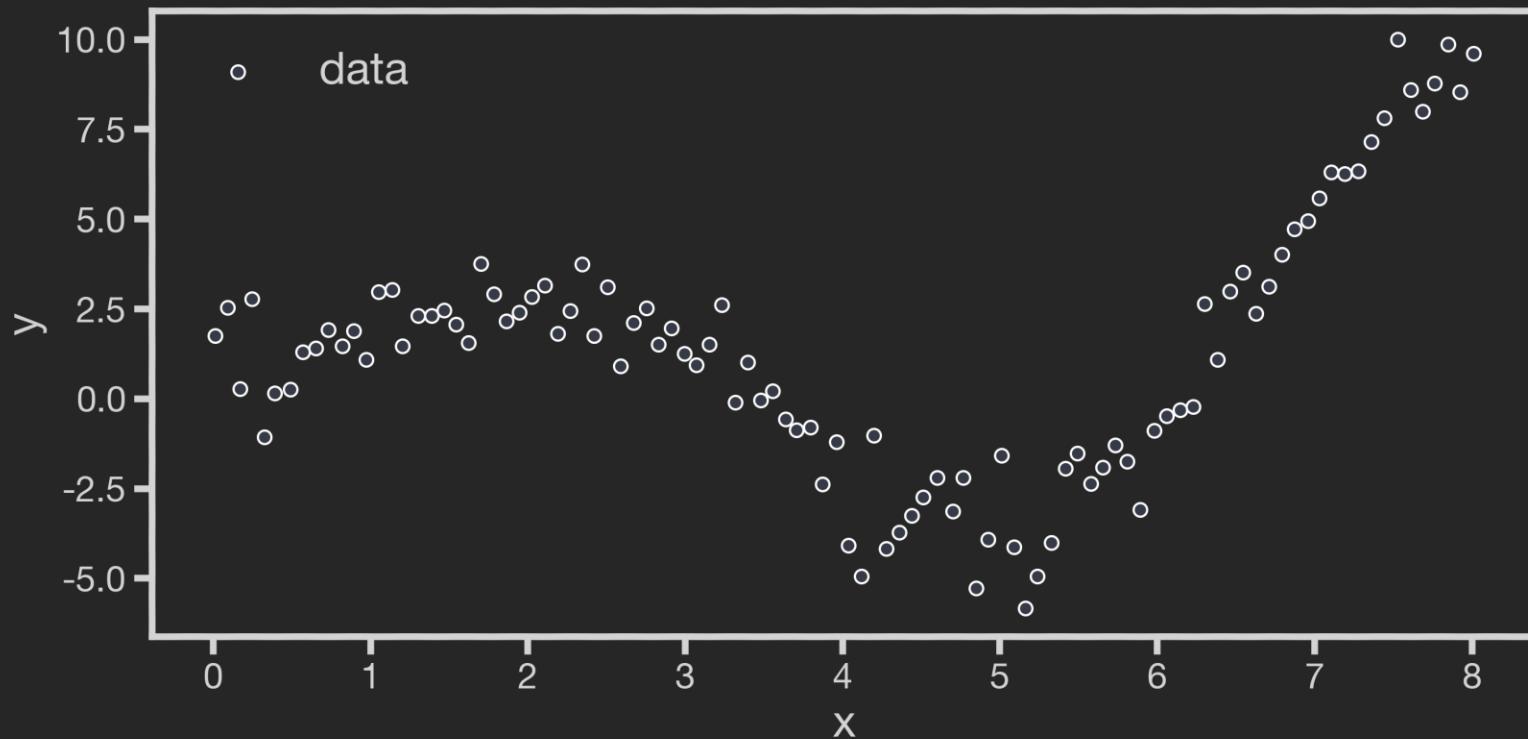
Gradient Boosting

There are two main ideas in gradient boosting:

- Gradient boosting is a method for iteratively building a complex model T by adding simple models.
- Each new simple model added to the ensemble compensates for the weaknesses of the current ensemble.

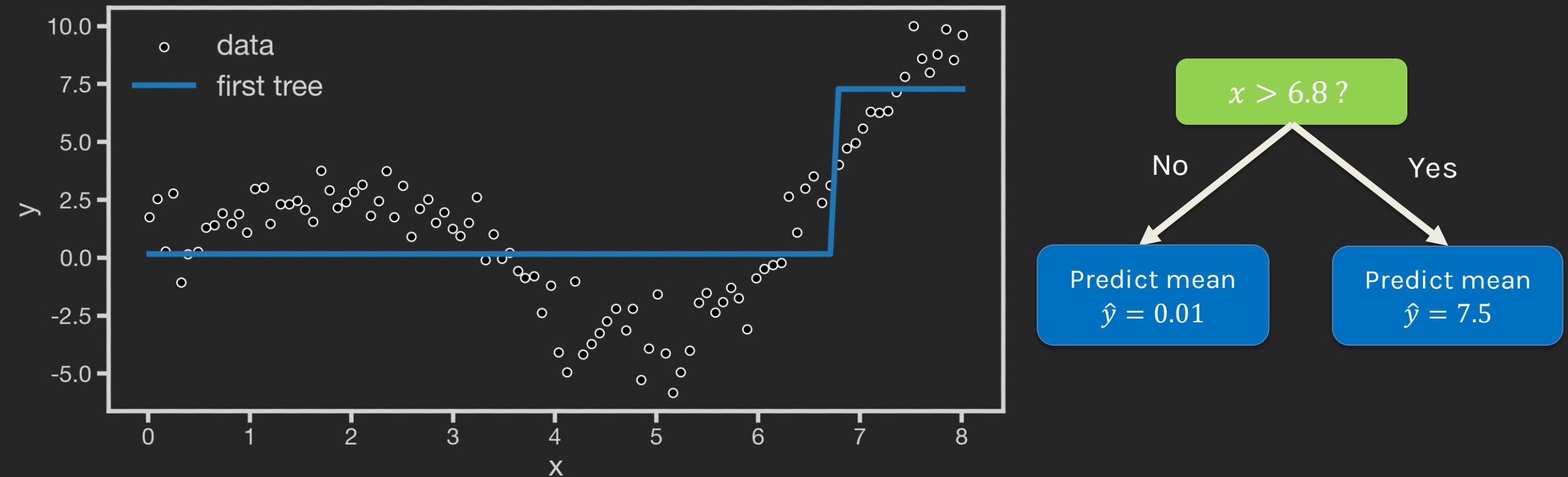
Gradient Boosting: Illustration

Consider the following dataset:



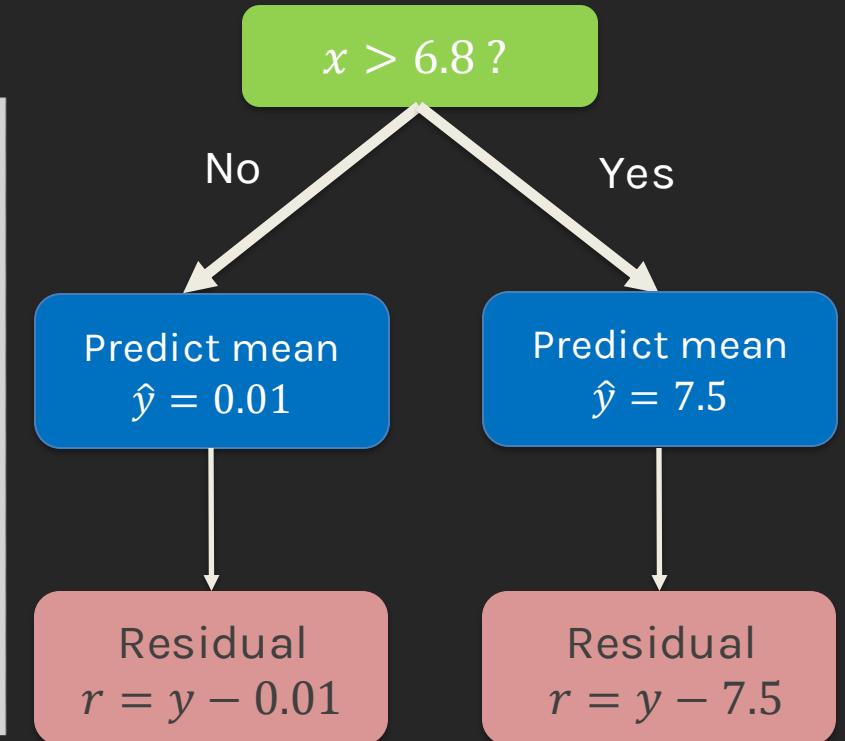
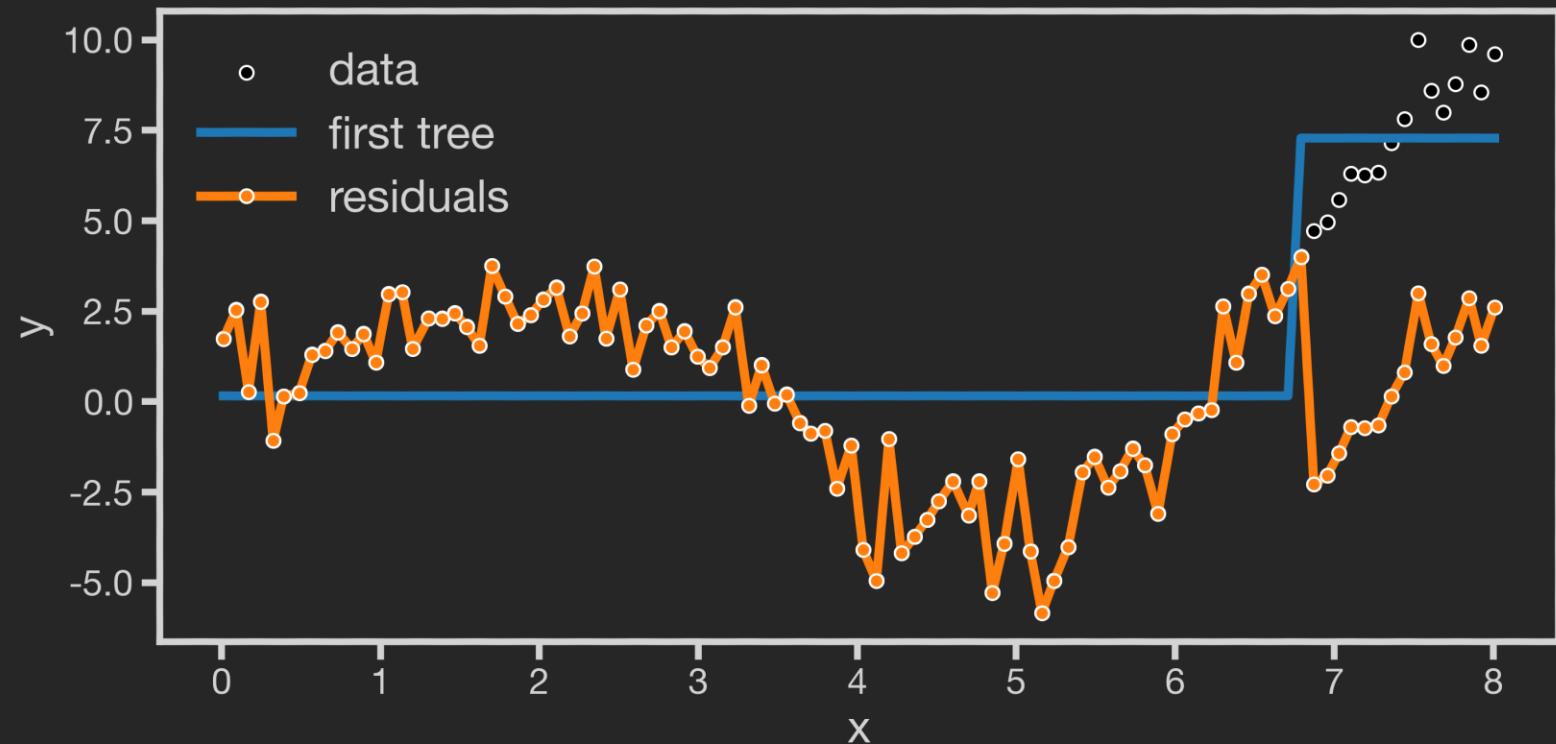
Gradient Boosting: Illustration

Step 1: Fit a simple model $T^{(0)}$ on the training data: $\{(x_1, y_1), \dots, (x_N, y_N)\}$.



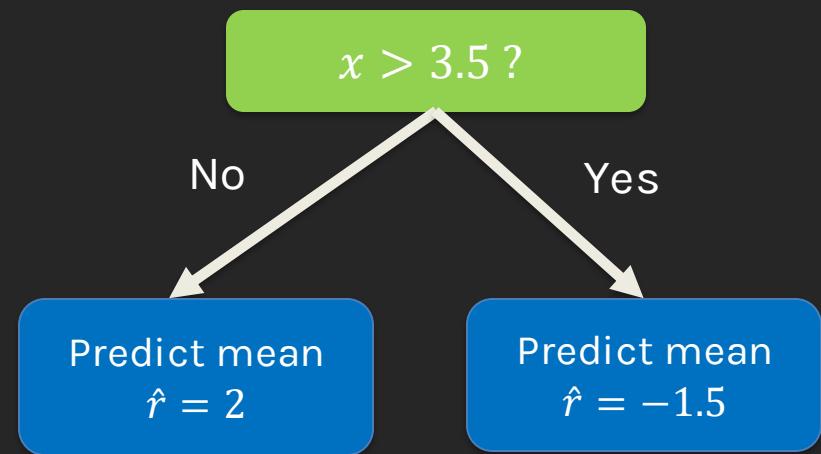
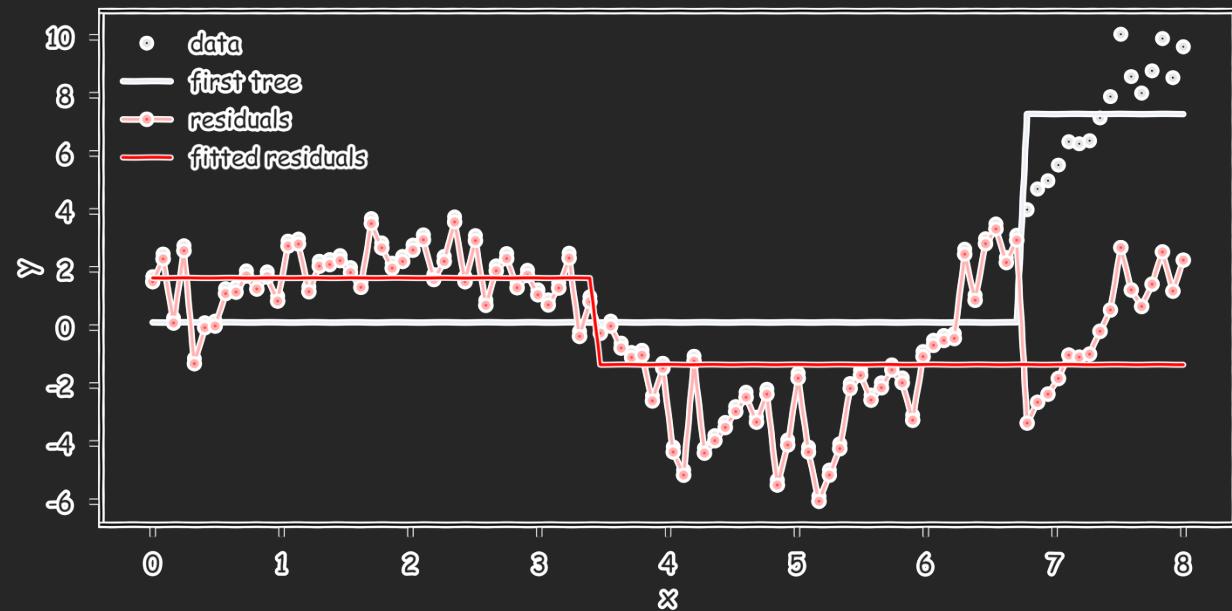
Gradient Boosting: Illustration

Step 2: Compute the residuals $\{r_1, \dots, r_N\}$ for $T^{(0)}$. Set $T \leftarrow T^{(0)}$.



Gradient Boosting: Illustration

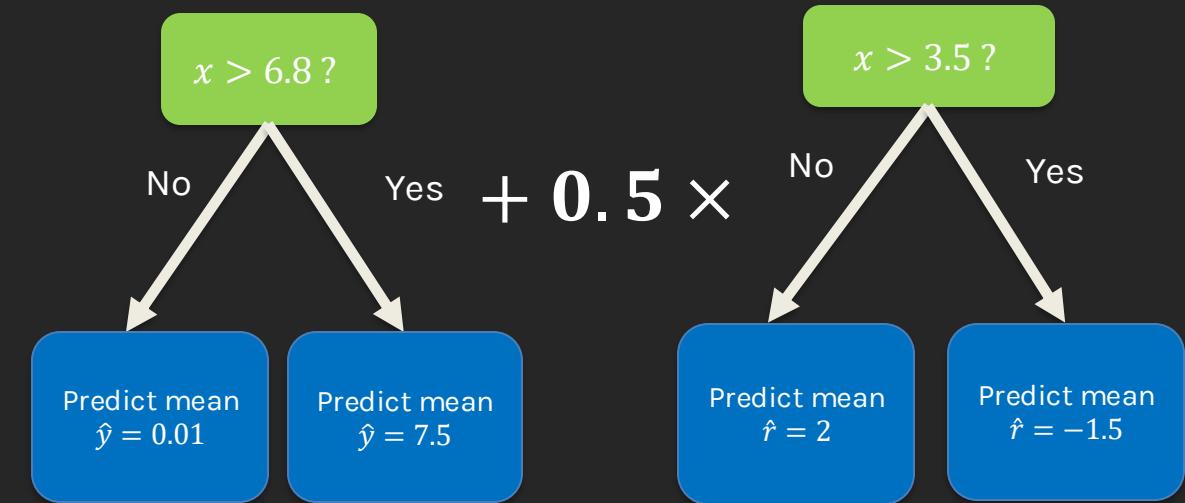
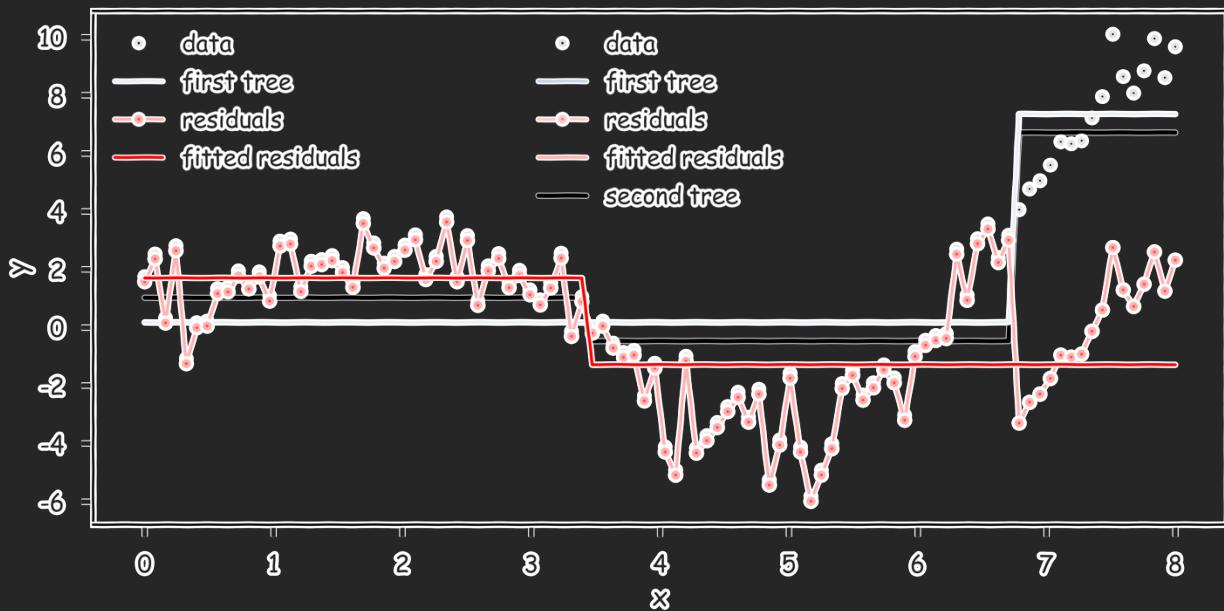
Step 3: Fit another model $T^{(1)}$ on: $\{(x_1, r_1), \dots, (x_N, r_N)\}$.



Gradient Boosting: Illustration

Step 4: Combine the two trees in step 1 and 3 by setting $T \leftarrow T + \lambda T^{(1)}$.

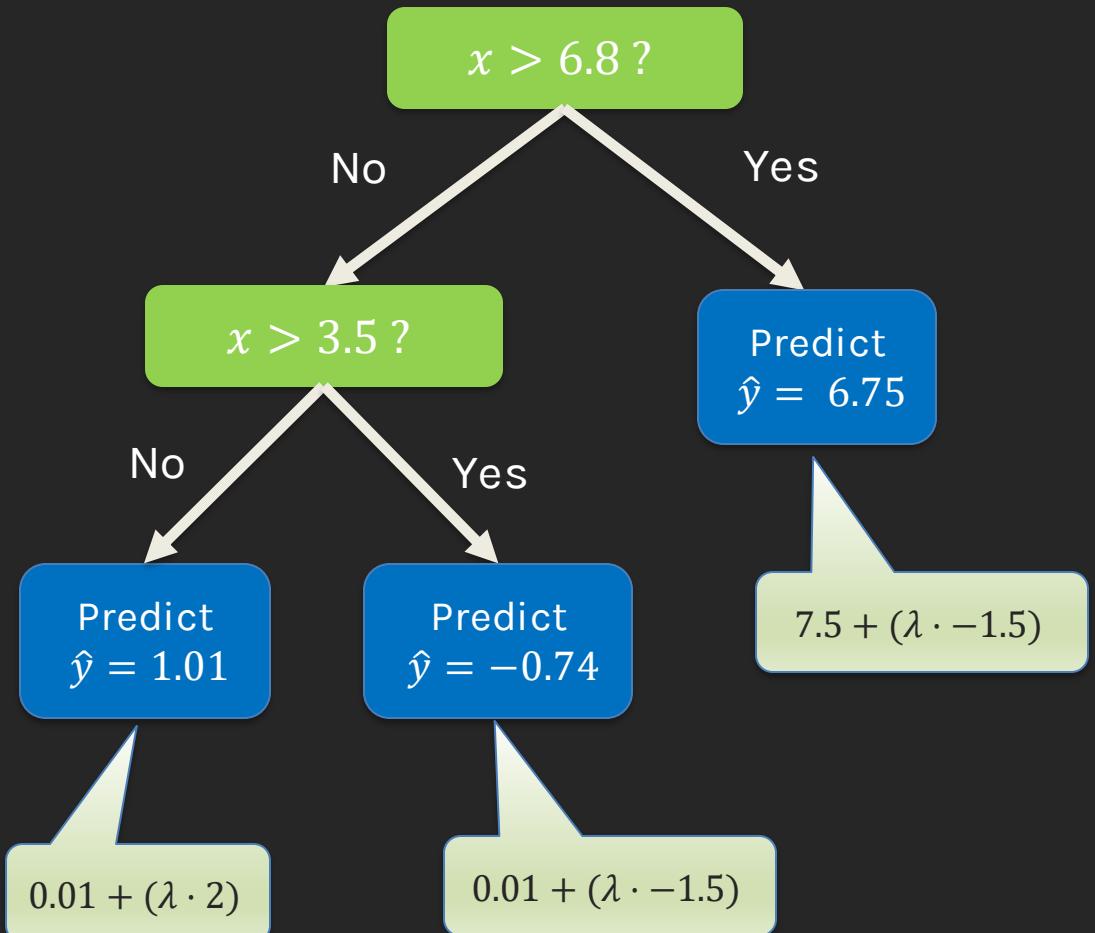
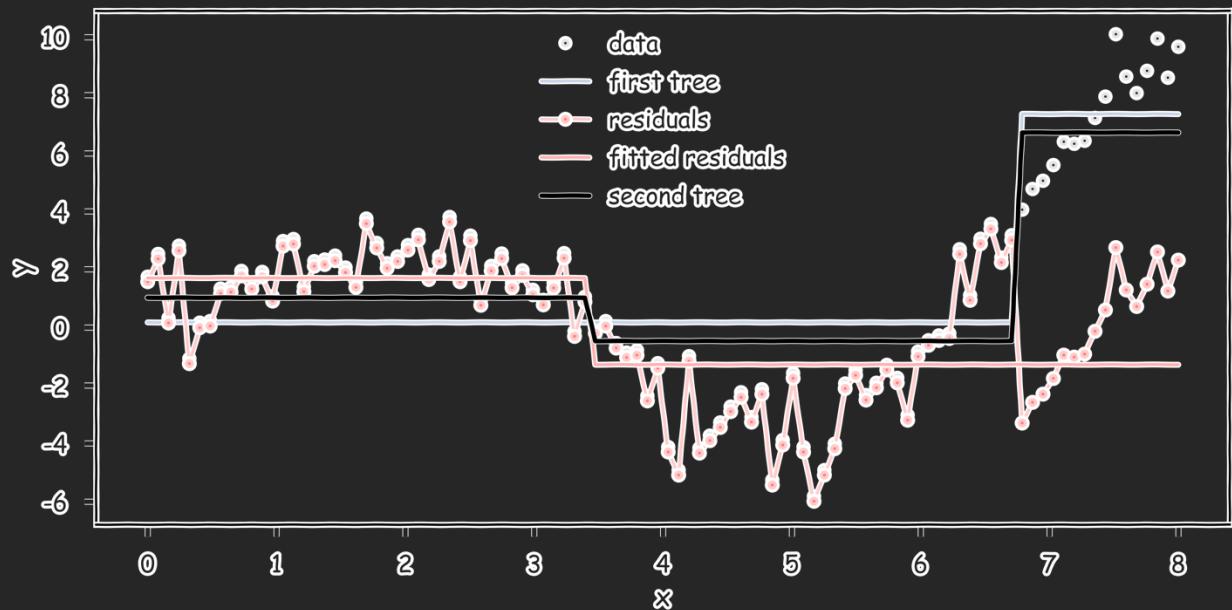
Assume $\lambda = 0.5$.



Gradient Boosting: Illustration

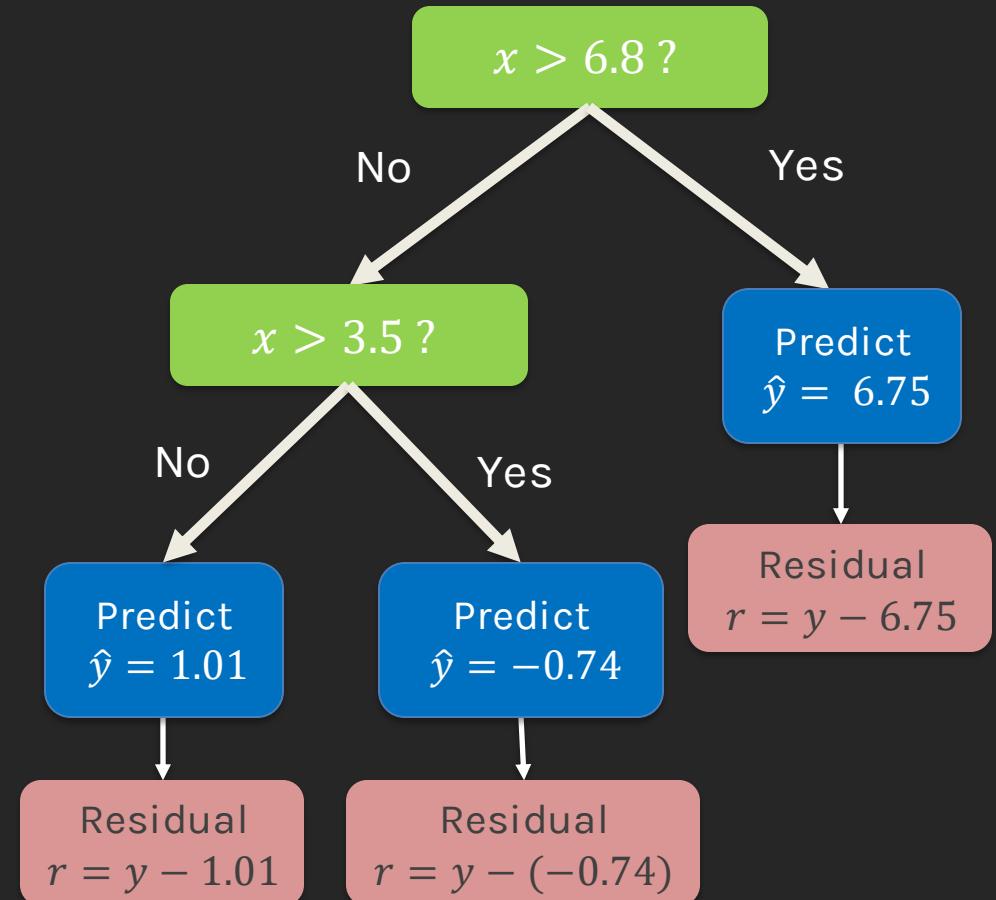
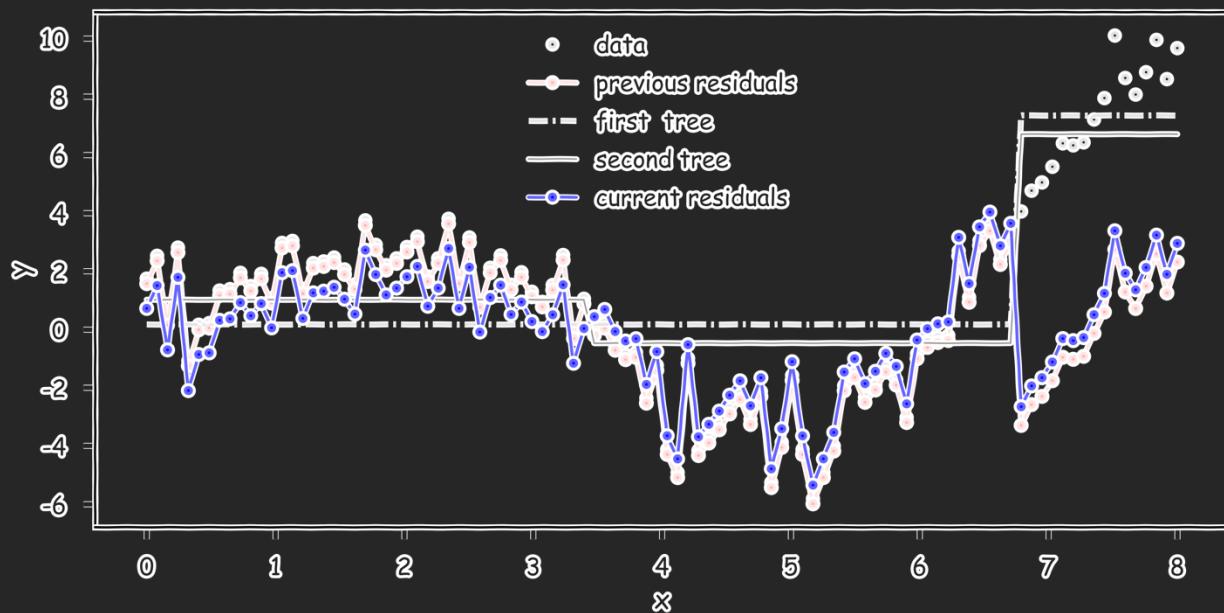
Step 4: Combine the two trees in step 1 and 3 by setting $T \leftarrow T + \lambda T^{(1)}$.

Assume $\lambda = 0.5$.



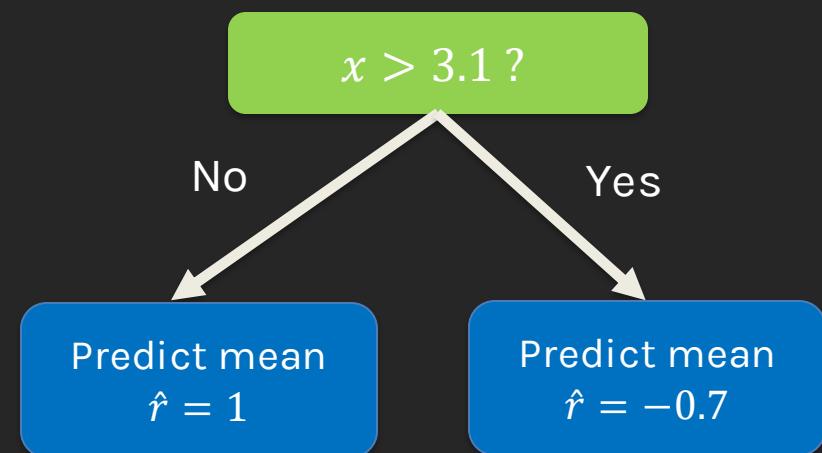
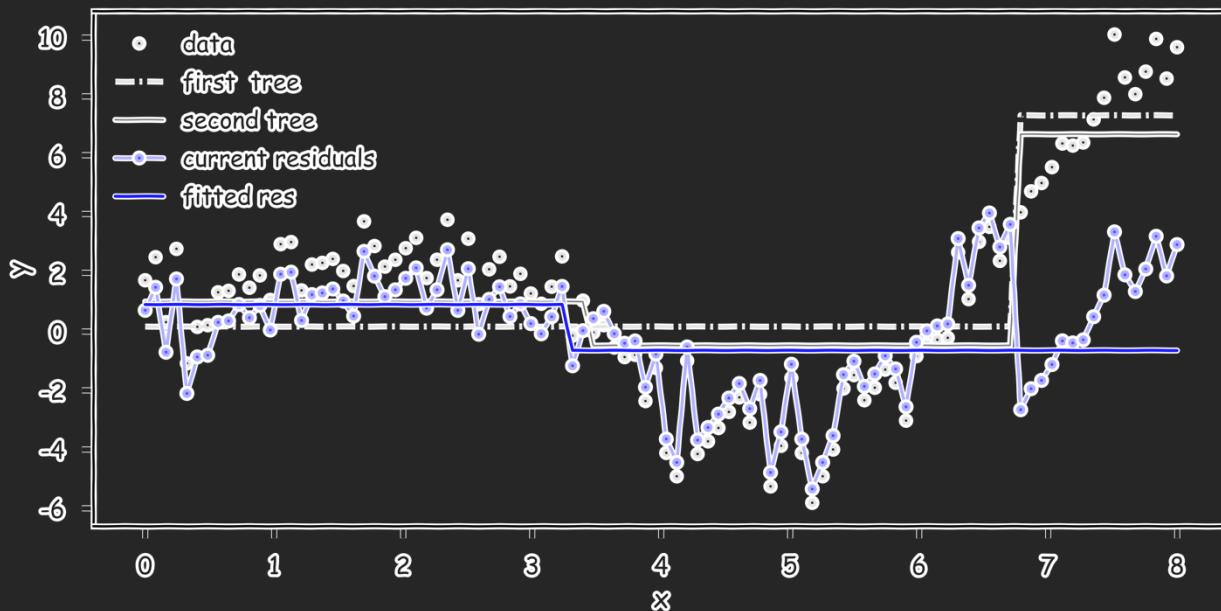
Gradient Boosting: Illustration

Step 5: Repeat step 2 on the new model by calculating the residuals $\{r_1, \dots, r_N\}$ for current model T .



Gradient Boosting: Illustration

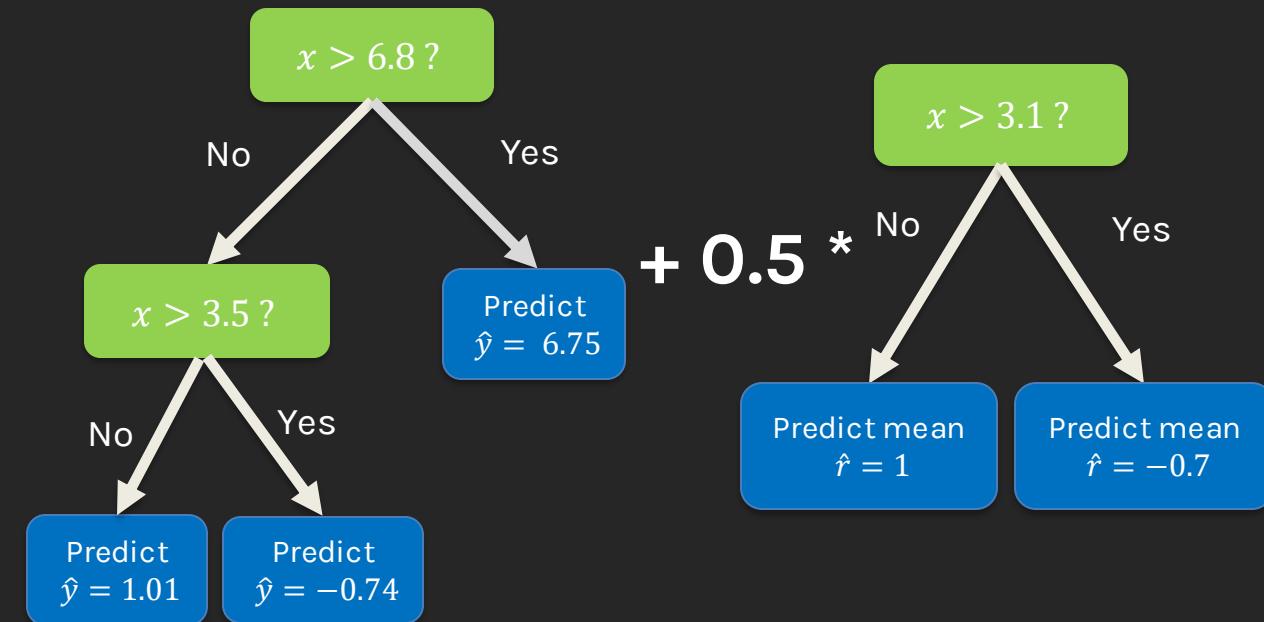
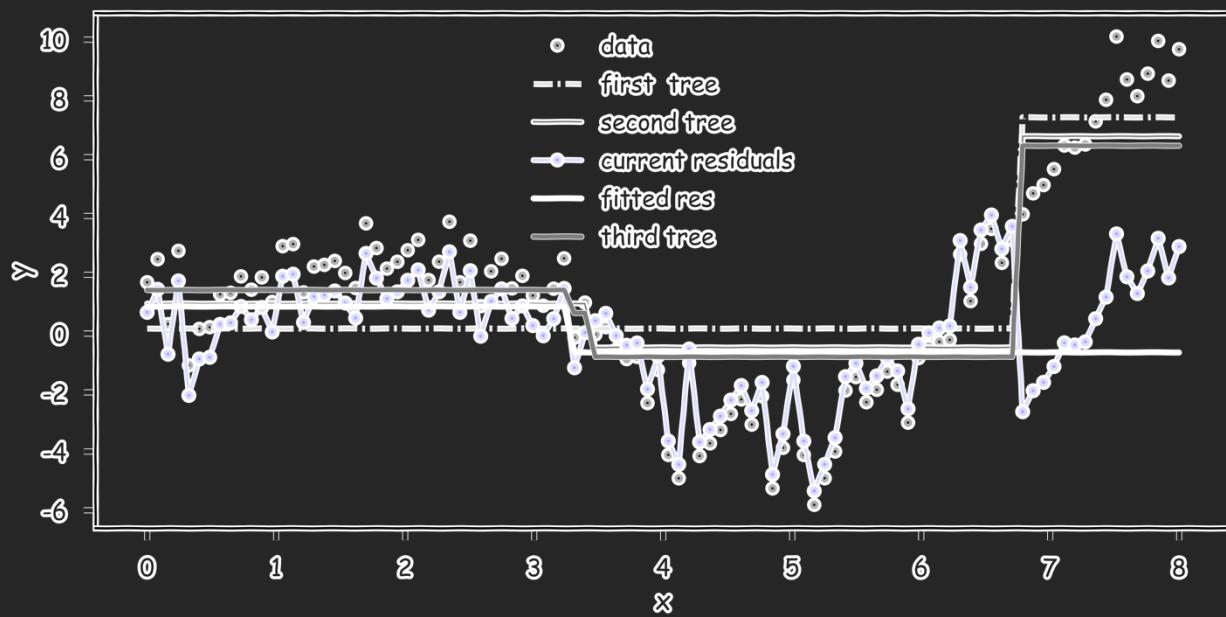
Step 6: Repeat step 3 and fit another model $T^{(2)}$ on the new residuals: $\{(x_1, r_1), \dots, (x_N, r_N)\}$.



Gradient Boosting: Illustration

Step 7: Combine the two trees in step 4 and 6 by setting $T \leftarrow T + \lambda T^{(2)}$.

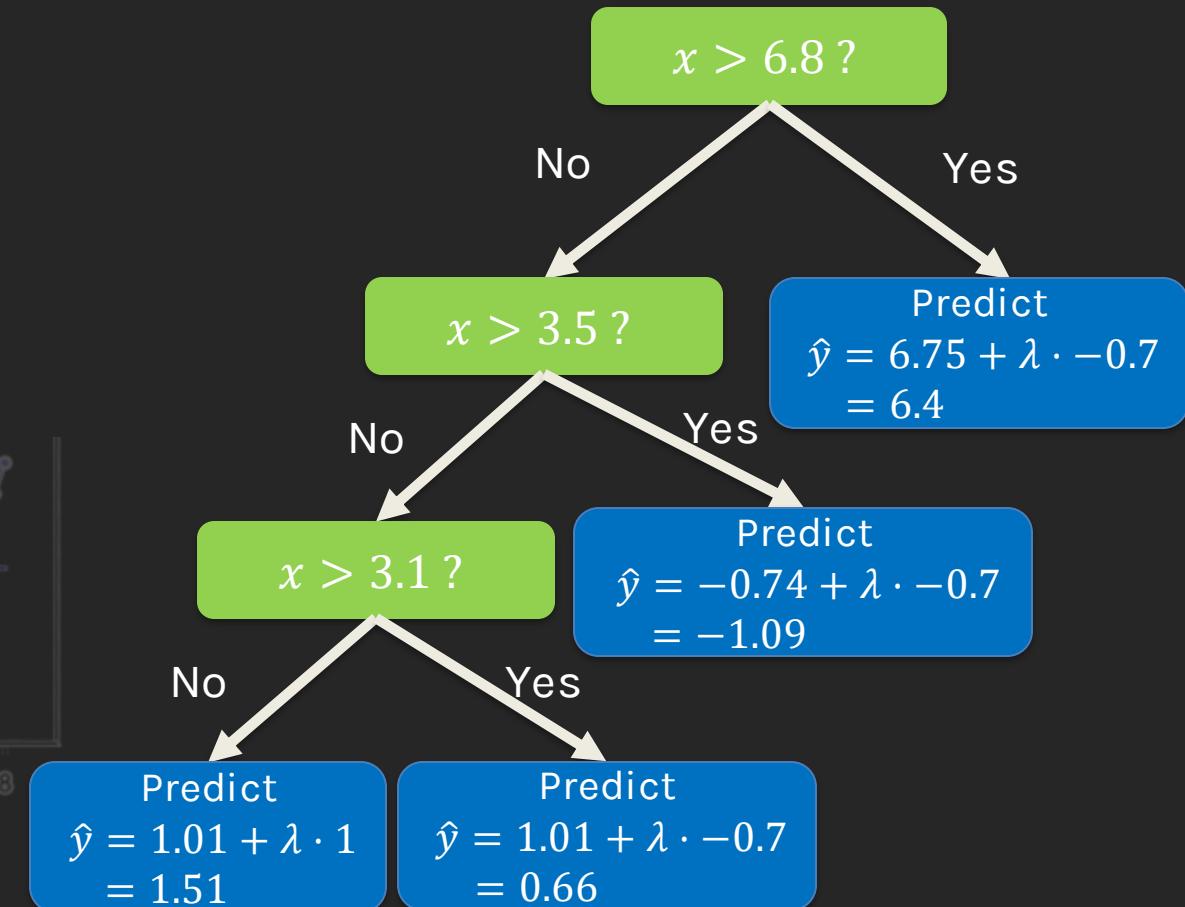
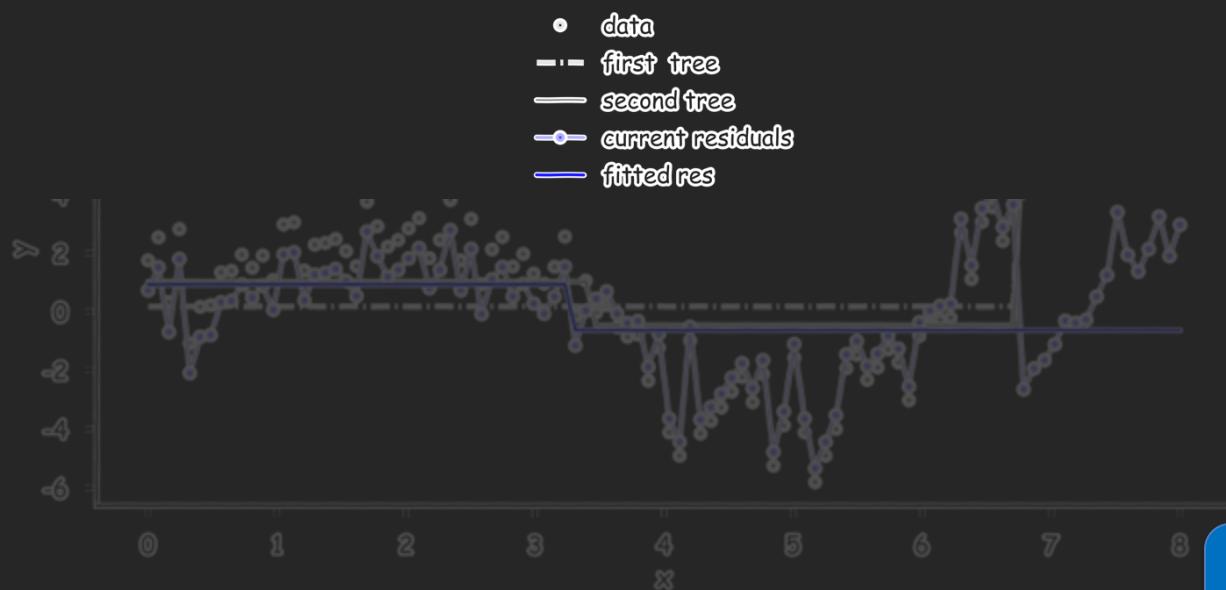
Assume $\lambda = 0.5$.



Gradient Boosting: Illustration

Step 7: Combine the two trees in step 4 and 6 by setting $T \leftarrow T + \lambda T^{(2)}$.

Assume $\lambda = 0.5$.



Gradient Boosting: Algorithm

Step 1: Fit a simple model $T^{(0)}$ on the training data $\{(x_1, y_1), \dots, (x_N, y_N)\}$. Set $T \leftarrow T^{(0)}$.

Step 2: Compute the residuals $\{r_1, \dots, r_N\}$ for T .

For $i = 1 \dots$ until stopping condition is met:

Step 3: Fit a simple model, $T^{(i)}$, to the current **residuals**.

That is, train $T^{(i)}$ using $\{(x_1, r_1), \dots, (x_N, r_N)\}$

Step 4: Set the current model $T \leftarrow T + \lambda T^{(i)}$.

Step 5: Compute residuals at step i , set $r_n \leftarrow r_n - \lambda T^{(i)}(x_n)$, $n = 1, \dots, N$

where λ is a constant called the **learning rate**.



Boosting, Gradient Boosting

Why Does Grad. Boosting work?



Outline

- Introduction to boosting
- Gradient Boosting
- **Mathematical Formulation - Gradient Boosting**

Why Does Gradient Boosting Work?

Intuitively, each simple model $T^{(i)}$ that we add to our ensemble T , models the residuals of T .

Thus, with each addition of $T^{(i)}$, the residuals are reduced.

$$r_n \leftarrow r_n - \lambda T^{(i)}(x_n)$$

Question: How?

Why Does Gradient Boosting Work?

In the previous example, we started off by modeling $T^{(0)}$ that gave us some predictions

$$T^{(0)}(x) = T_{pred}^{(0)}.$$

We then train $T^{(1)}$ on the residuals of $T^{(0)}$ such that $T_{pred}^{(1)} \approx \underbrace{y - T_{pred}^{(0)}}_{r^{(0)}}$.

Finally, we combined the models to form $T \leftarrow T^{(0)} + \lambda T^{(1)}$.

The residuals of this model would be:

$$\begin{aligned} y - T_{pred} &= y - T_{pred}^{(0)} - \lambda T_{pred}^{(1)} \\ &= r^{(0)} - \lambda T_{pred}^{(1)} \end{aligned}$$

Therefore, as we combine more models, the residuals keeps decreasing.

$$r_n \leftarrow r_n - \lambda T^{(i)}(x_n)$$

Why Does Gradient Boosting Work?

$$r_n \leftarrow r_n - \lambda T^{(i)}(x_n)$$

If we want to easily reason about convergence and how to choose λ and investigate the effect of λ on the model T , we need a bit more mathematical formalism.

In particular, how can we effectively **descend** through this optimization via an **iterative** algorithm?

We need to formulate gradient boosting as a type of **gradient descent**.

Review: A Brief Sketch of Gradient Descent

In **optimization**, when we wish to minimize a function, called the **objective function (or loss function)**, over a set of variables, we compute the **partial derivatives** of this function with respect to the variables.

If the partial derivatives are sufficiently **simple**, one could analytically find a common root - i.e., a point at which all the partial derivatives vanish; this is called a **stationary point**.

If the objective function has the property of being **convex**, then the stationary point is precisely the global minimum.

Review: A Brief Sketch of Gradient Descent

In practice, our objective functions are complicated, and analytically finding a stationary point is intractable.

Instead, we use an iterative method called **gradient descent**.

1. Initialize the variables at any value

$$x = [x_1, \dots, x_J]$$

2. Take the gradient of the objective function at the current variable values.

$$\nabla_x f(x) = \left[\frac{\partial f}{\partial x_1}(x), \frac{\partial f}{\partial x_2}(x), \dots, \frac{\partial f}{\partial x_J}(x) \right]$$

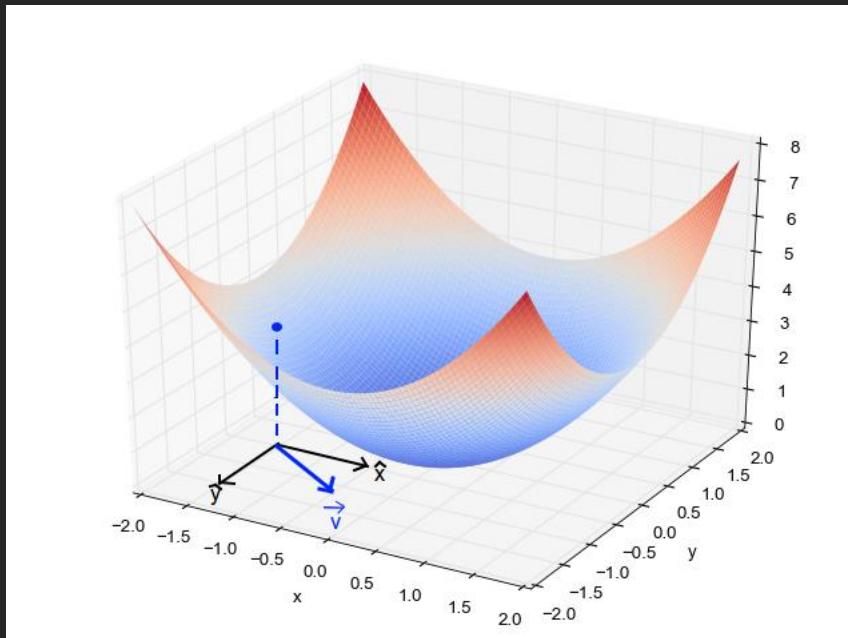
3. Adjust the variables by some negative multiple of the gradient.

$$x \leftarrow x - \lambda \nabla f(x) \text{ where } \lambda \text{ is the learning rate.}$$

Why Does Gradient Descent Work?

Claim: If the function is convex, this iterative methods will eventually move x close enough to the minimum, for an appropriate choice of λ .

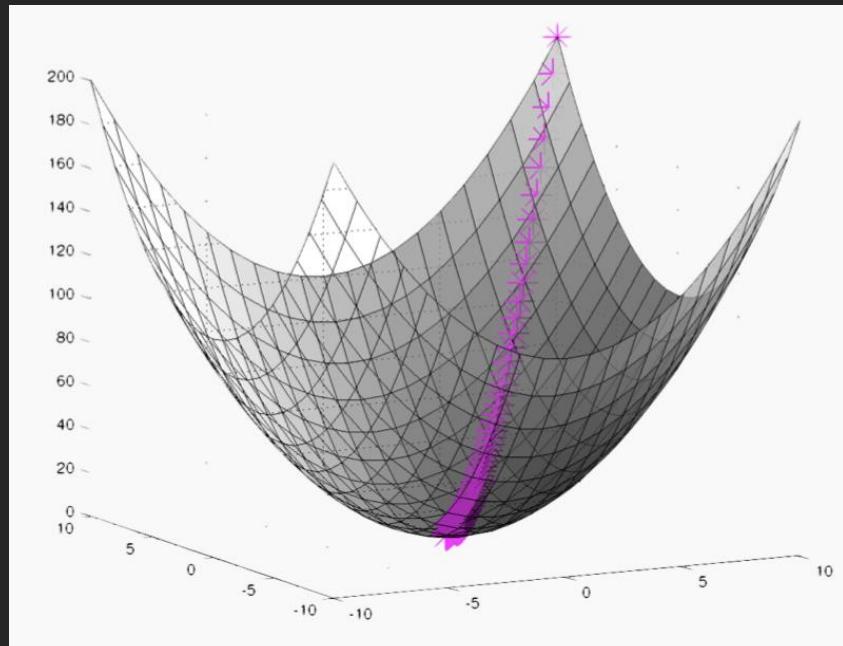
Why does this work? Recall, that as a vector, the gradient at a point gives the direction for the greatest possible rate of increase.



Why Does Gradient Descent Work?

Subtracting a λ multiple of the gradient from x , moves x in the opposite direction of the gradient (hence towards the steepest decline) by a step of size λ .

If f is convex, and we keep taking steps descending on the graph of f , we will eventually reach the minimum.

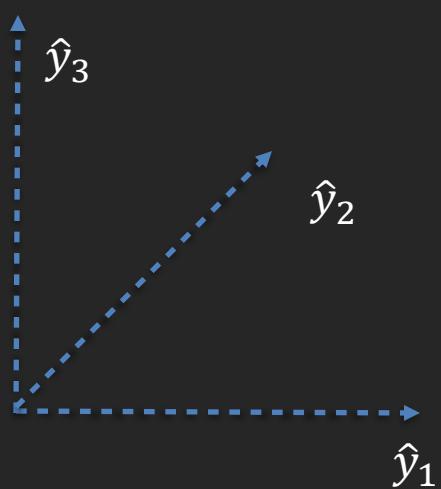


Gradient Boosting as Gradient Descent

Prediction space

Assume for now that we have 3 training examples $\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$.

Let us look at the **space of predictions**:



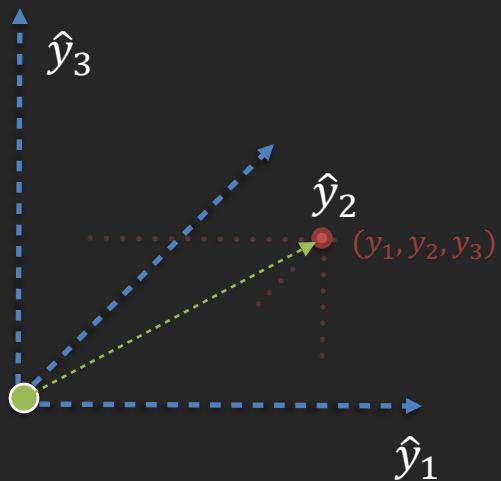
In this space, each point represents our set of predictions $y = (\hat{y}_1, \hat{y}_2, \hat{y}_3)$ for all datapoints. Note that here we have not assumed any relation between x and y whatsoever, we are just optimizing over the space of predictions.

The loss that we are optimizing for is the MSE:

$$L(y) = \frac{1}{3} [(y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + (y_3 - \hat{y}_3)^2]$$

Gradient Boosting as Gradient Descent

$$L(y) = \frac{1}{3}[(y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + (y_3 - \hat{y}_3)^2]$$



Here as the loss function is **convex**, we already know the **unique minimizer** $y^* = (y_1, y_2, y_3)$.

So if we start at the origin (or anywhere else), we will end at y^* when we optimize using gradient descent.

Gradient Boosting as Gradient Descent

More generally, we have:

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

Treating this as an optimization problem, we can try to directly minimize the MSE with respect to the predictions:

$$\begin{aligned}\nabla_{\hat{y}} MSE &= \left[\frac{\partial MSE}{\partial \hat{y}_1}, \dots, \frac{\partial MSE}{\partial \hat{y}_N} \right] \\ &= -2[y_1 - \hat{y}_1, \dots, y_N - \hat{y}_N] \\ &= -2[r_1, \dots, r_n]\end{aligned}$$

The update step for gradient descent would look like:

$$\hat{y}_n \leftarrow \hat{y}_n + \lambda r_n, \quad n = 1, \dots, N$$

Gradient Boosting as Gradient Descent (cont.)

There are two reasons why **minimizing** the MSE with respect to \hat{y}_n 's - the **predictions** - is not interesting:

Minimizing a convex function

- We know where the minimum MSE occurs: $\hat{y}_n = y_n$, for every n.
- Learning sequences of predictions, $\{\hat{y}_n^{(1)} \dots \hat{y}_n^{(i)} \dots\}$ does not produce a model. This is because we have, by no means, learned to map predictors to the prediction.

Here “i” is the iteration num (epoch)

Gradient Boosting as Gradient Descent

When trying to view Gradient Boosting as a form of Gradient Descent, we need to look at the **function space**.

Imagine a space where each point represents a function $f: X \rightarrow y$, then ideally, we want to optimize this **function space** to find the best estimator.

Represents a particular function

But we cannot do computation on infinite dimensional abstract spaces

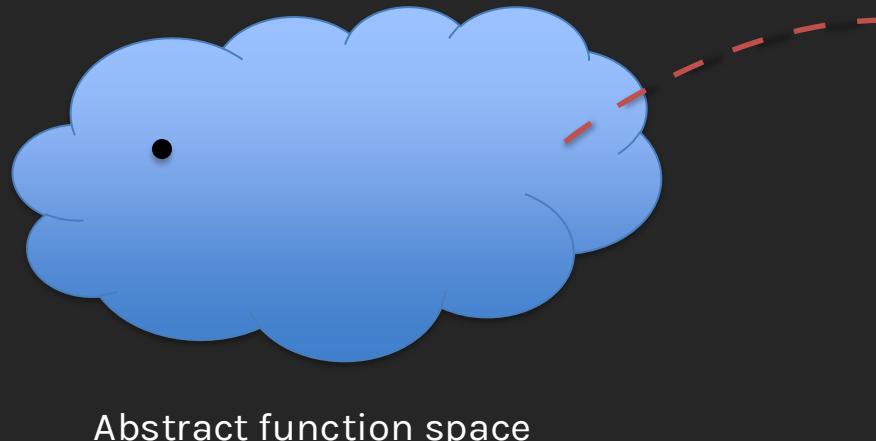


Abstract function space

Gradient Boosting as Gradient Descent

When trying to view Gradient Boosting as a form of Gradient Descent, we need to look at the **function space**.

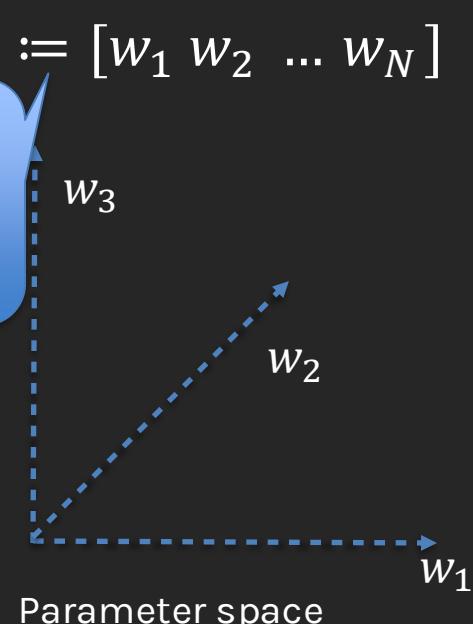
Imagine a space where each point represents a function $f: X \rightarrow y$, then ideally, we want to **optimize** over this **function space** to find out the best estimator.



In case of **parametric models**, we **optimize** our model in the **weight space** (parameters of our model)

$$\hat{f} := [w_1 \ w_2 \ \dots \ w_N]$$

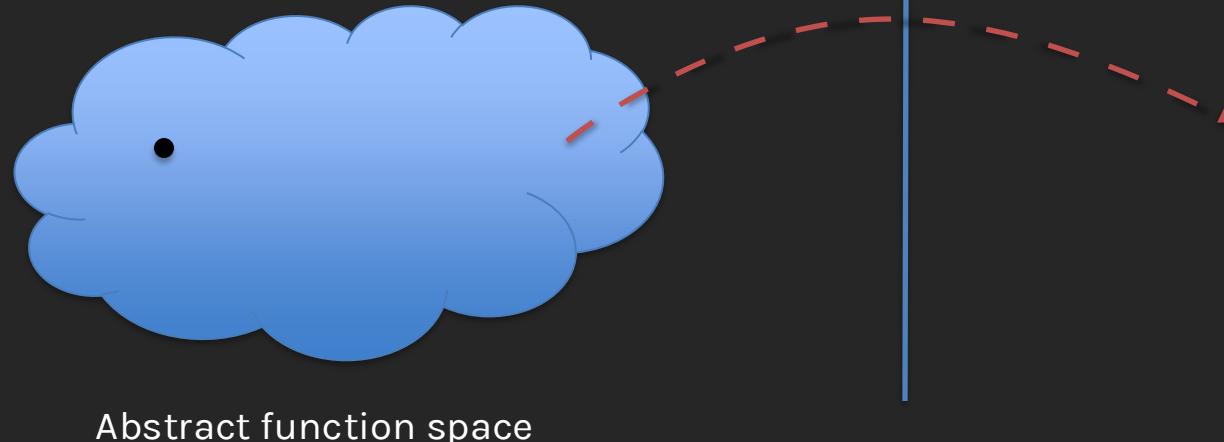
Means “ \hat{f} is represented by”



Gradient Boosting as Gradient Descent

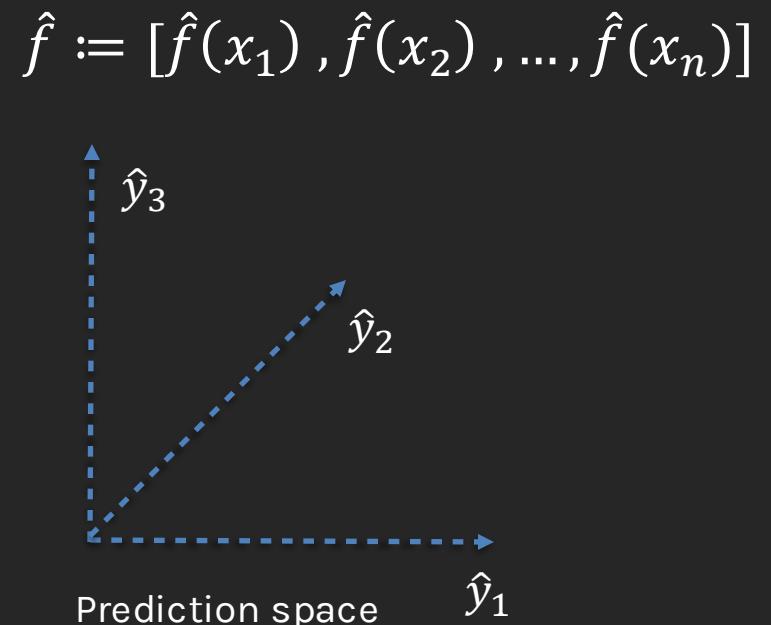
When trying to view Gradient Boosting as a form of Gradient Descent, we need to look at the **function space**.

Imagine a space where each point represents a function $f: X \rightarrow y$, then ideally, we want to **optimize** over this **function space** to find out the best estimator.



PROTOPAPAS

What about **non-parametric** models?
In boosting, we **optimize** in the **prediction space** of the model.

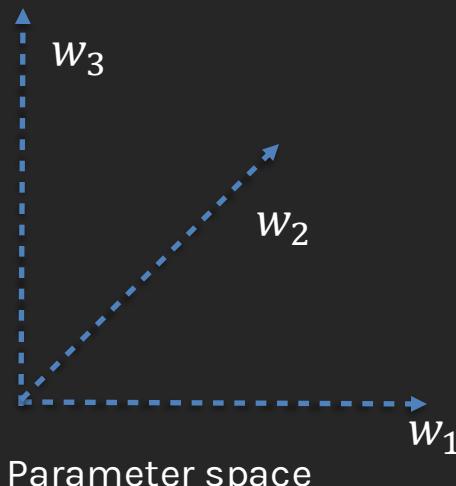


Gradient Boosting as Gradient Descent

In summary, for parametric models (say neural nets), gradient descent is done in the **parameter space**. Boosting can be thought of as doing gradient descent in the **prediction space**.

Parametric regime

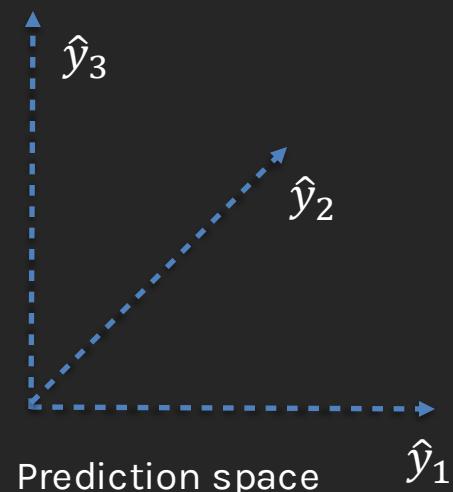
$$\hat{f} := [w_1, w_2, \dots, w_N]$$



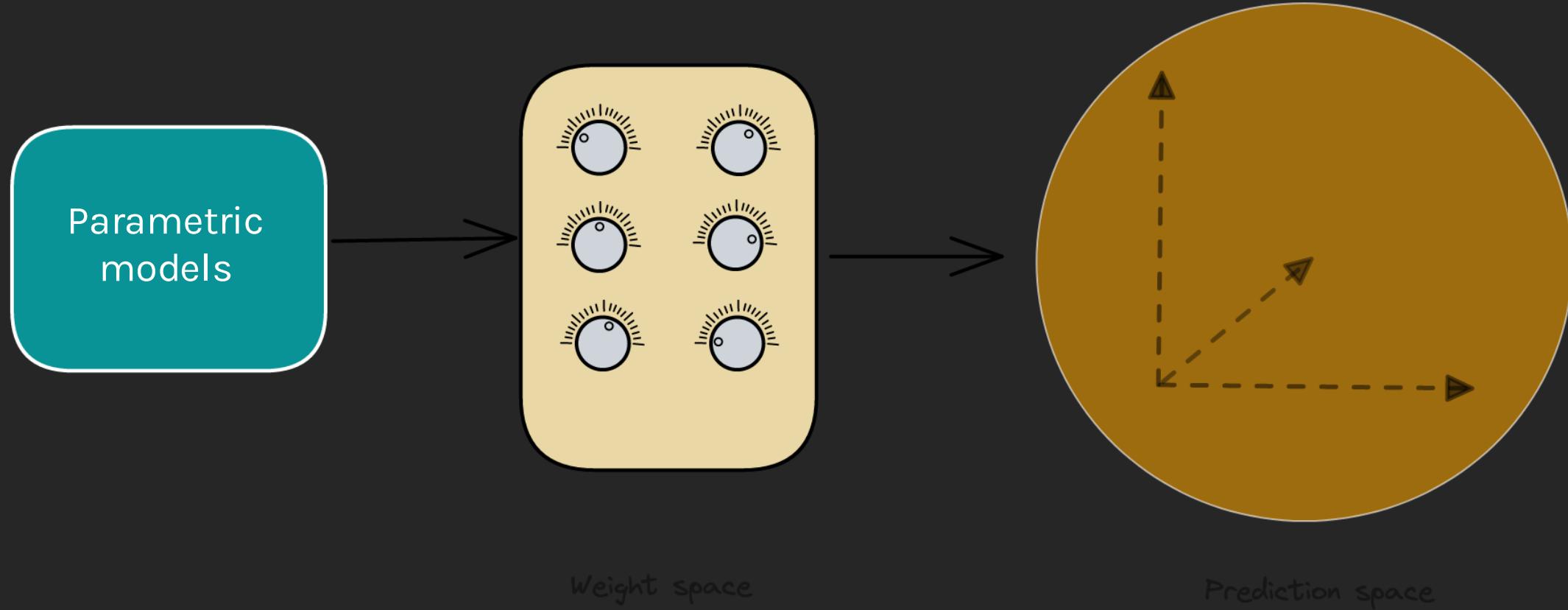
PROTOPAPAS

Non-parametric regime

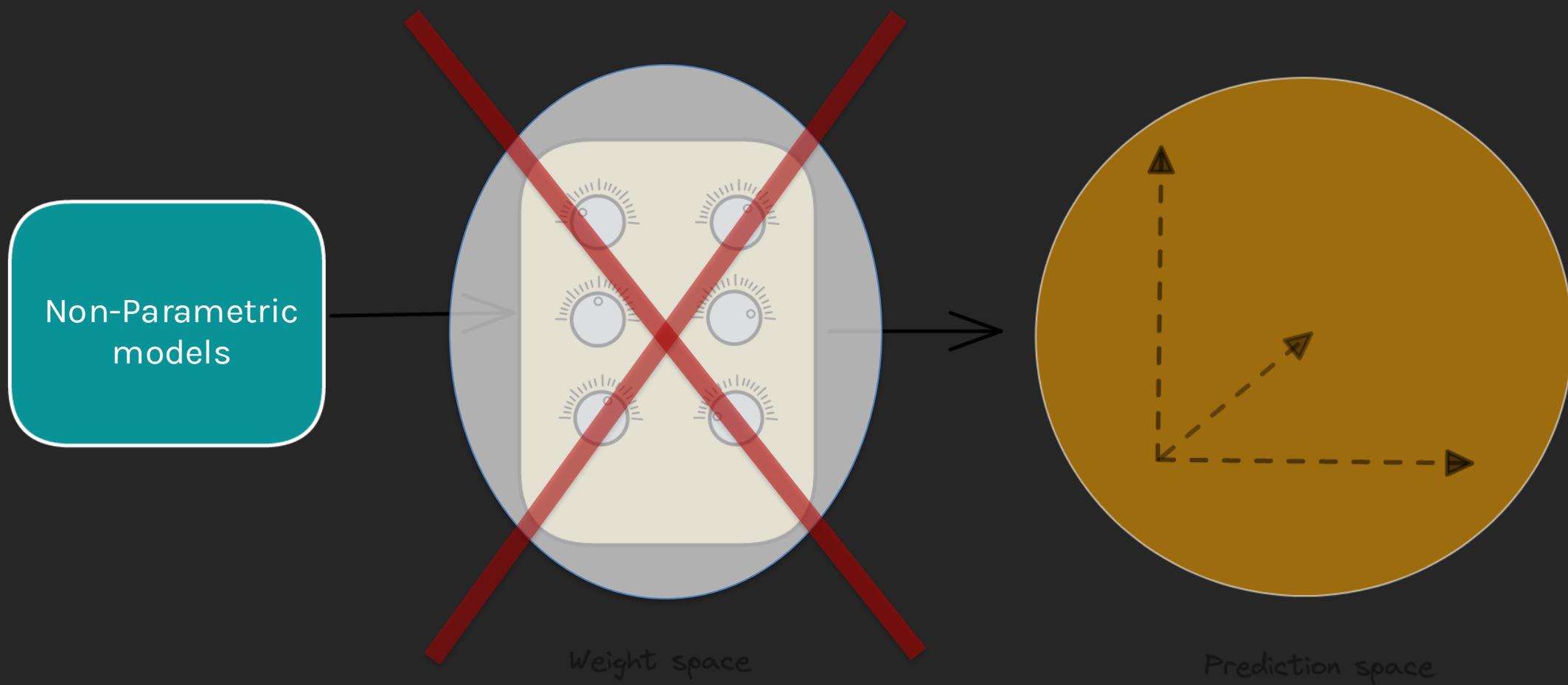
$$\hat{f} := [\hat{f}(x_1), \hat{f}(x_2), \dots, \hat{f}(x_n)]$$



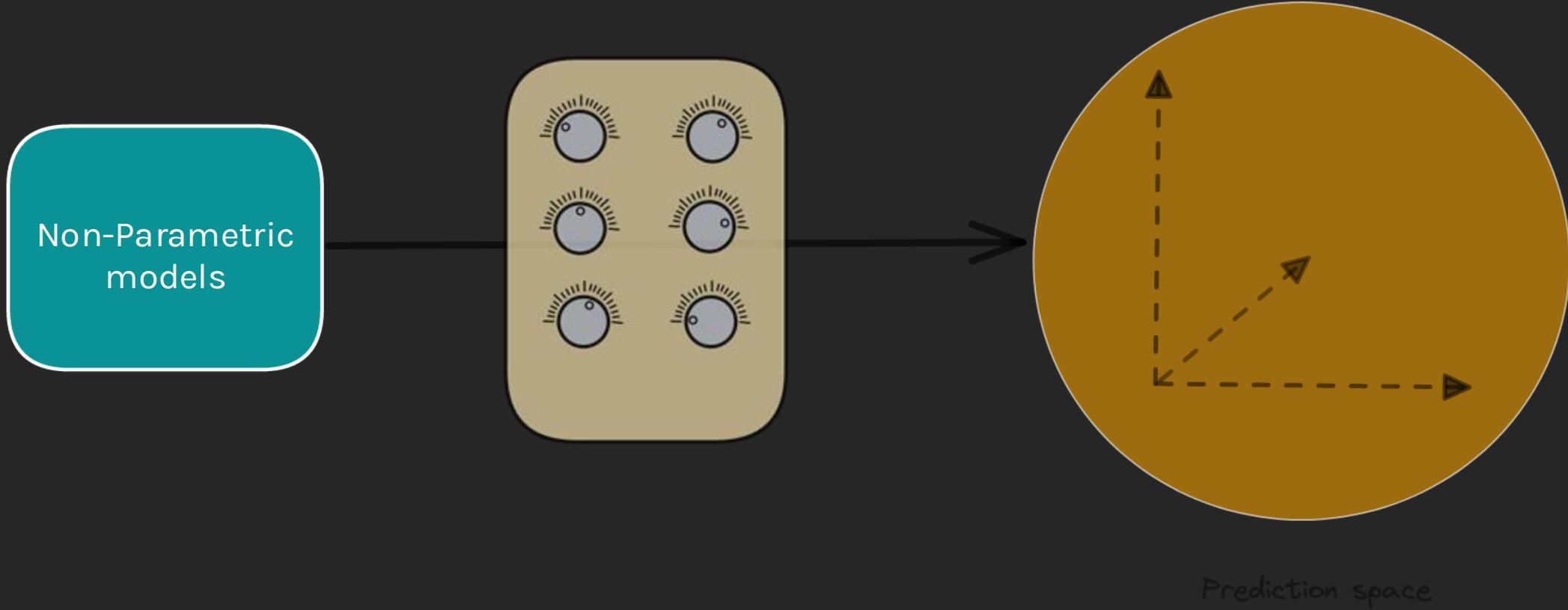
Gradient Boosting as Gradient Descent



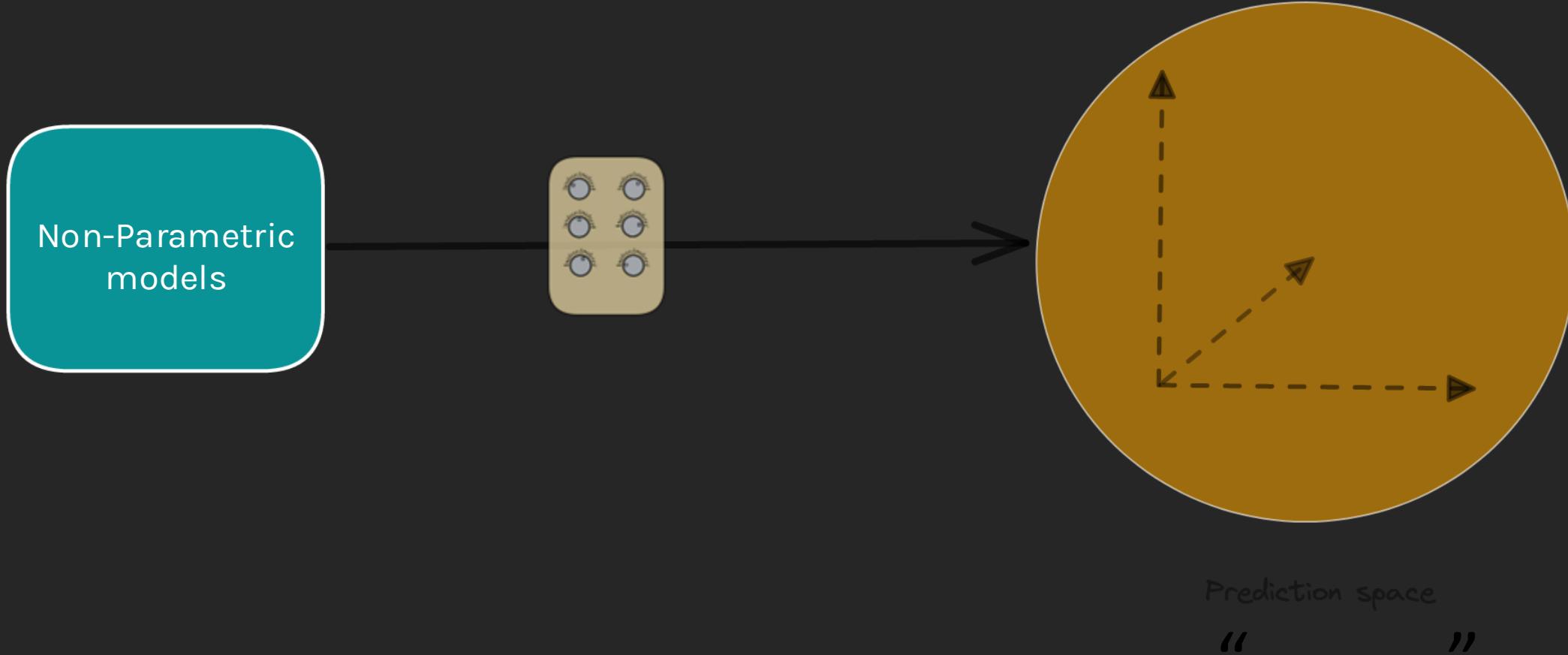
Gradient Boosting as Gradient Descent



Gradient Boosting as Gradient Descent



Gradient Boosting as Gradient Descent



Gradient Boosting as Gradient Descent

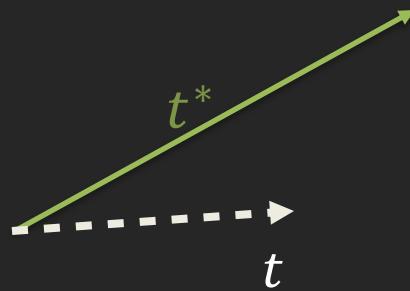
Let our current estimate be $T^{(n)}(x)$. In the prediction space, this function is just a vector of the prediction values. Then the optimal direction to take a step using Gradient Descent is along the negative gradient.

$$\begin{aligned} t^* &= -\frac{\partial \mathcal{L}}{\partial T^{(n)}(x)} \\ &= -\left[\frac{\partial \mathcal{L}}{\partial \hat{y}_1}, \frac{\partial \mathcal{L}}{\partial \hat{y}_2}, \dots, \frac{\partial \mathcal{L}}{\partial \hat{y}_N} \right] \\ &= -[(\hat{y}_1 - y_1), \dots, (\hat{y}_N - y_N)] \\ &= 2[r_1, r_2, \dots, r_N] \end{aligned}$$

Hence, the optimal direction to step in along the residuals.

Gradient Boosting as Gradient Descent

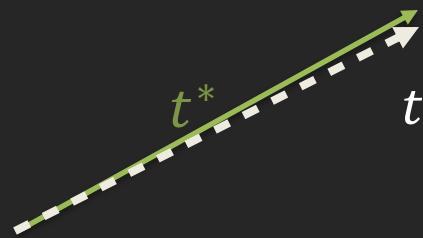
If we are restricting to a simple family of functions, we might not be able to always take a step in this **optimal direction** t^* . But we want to take as close a step to this direction as possible. Let \mathbb{H} be a simple family of models (say Decision Trees of `max_depth=5`).



Then we need to find some $t \in \mathbb{H}$ such that $t = [t(x_1), t(x_2), \dots, t(x_N)]$ is as close to $t^* = [r_1, r_2, \dots, r_N]$ as possible. Well, this is quite easy to do if we train another model in \mathbb{H} on the training set $\{(x_1, r_1), (x_2, r_2), \dots, (x_N, r_N)\}$.

Gradient Boosting as Gradient Descent

NOTE: If we were **not** restricting to a simple family of models (like using an unrestricted decision tree), we could directly take a step in the direction of the residuals and end up at y^* :

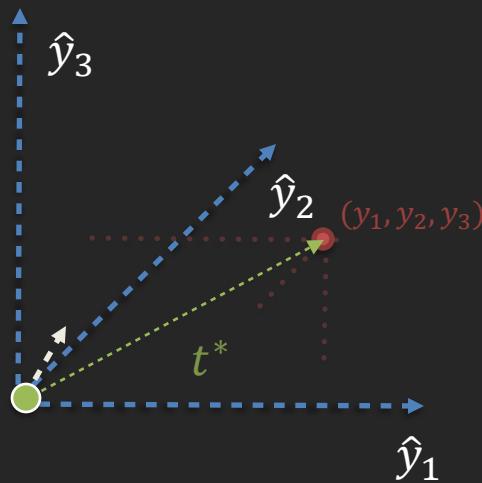


But this would mean we are completely **overfitting** on the training set. However, our goal is to decrease the bias **without increasing the variance**.

Gradient Boosting as Gradient Descent

How does this pan out graphically?

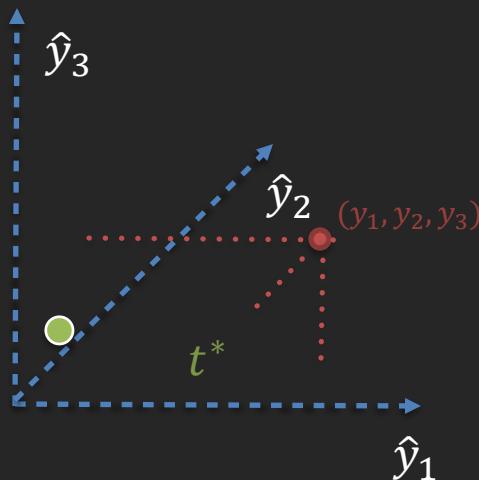
1. Say we start at the origin.
2. The optimal direction is along the residual vector (green line).
3. Because we restrict to a simple family of models, our approximations of the residuals are not exact and hence we take a step close to that direction.



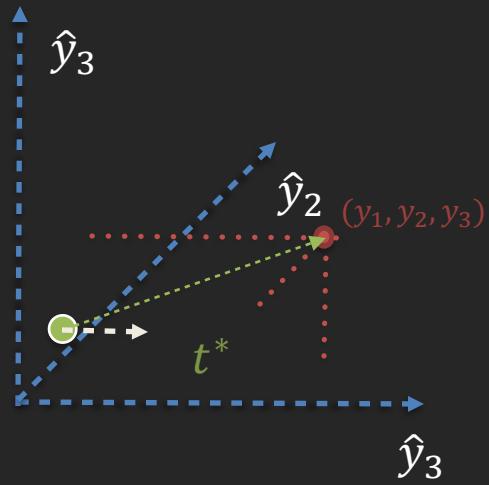
Gradient Boosting as Gradient Descent

How does this pan out graphically?

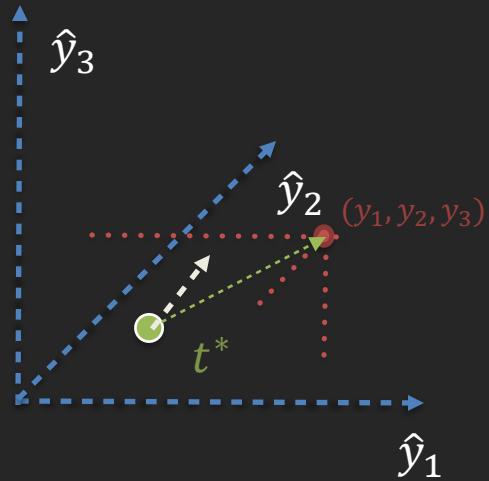
1. Say we start at the origin.
2. The optimal direction is along the residual vector (green line).
3. Because we restrict to a simple family of models, our approximations of the residuals are not exact and hence we take a step close to that direction.



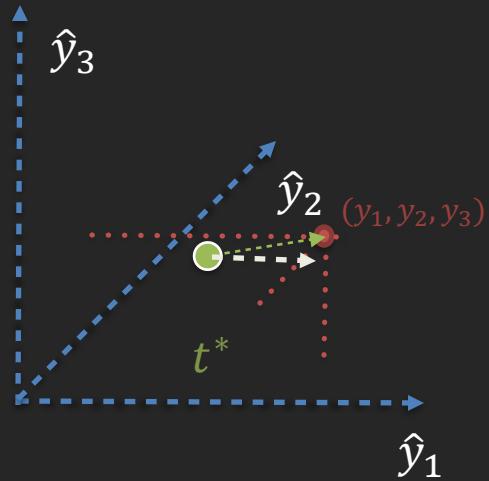
Gradient Boosting as Gradient Descent



Gradient Boosting as Gradient Descent



Gradient Boosting as Gradient Descent



Gradient Boosting as Gradient Descent

Hence instead of using the gradient (the residuals), we use an **approximation** of the gradient that depends on the predictors:

$$\hat{y}_n \leftarrow \hat{y}_n + \lambda \hat{r}_n(x_n) \quad \text{where } n = 1, 2, \dots, N$$

In gradient boosting, we use a **simple model** $\hat{r}_n(x_n)$, to **approximate the residuals** $r_n(x_n)$ in each iteration (which is the negative gradient of the loss in the prediction space).

Technical note: Gradient boosting is descending in a space of models or functions mapping x_n to y_n !

Gradient Boosting as Gradient Descent (cont.)

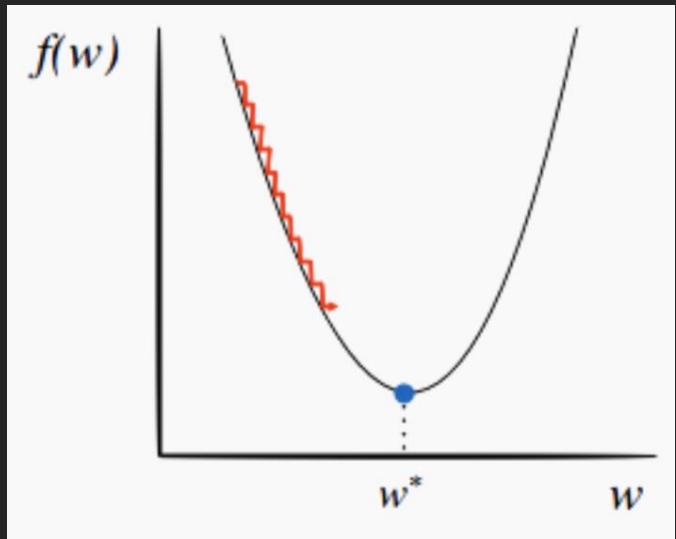
But why do we want to connect gradient boosting to gradient descent?

By making this connection, we can import the massive amount of techniques for studying gradient descent to analyze gradient boosting.

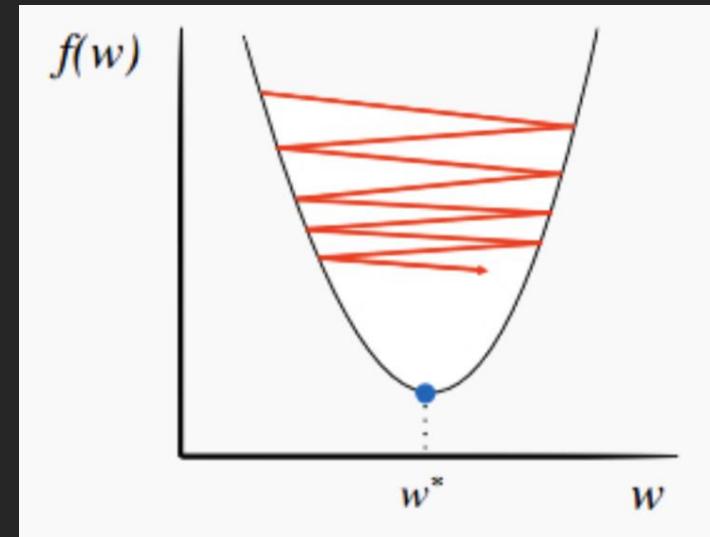
For example, we can easily reason about how to choose the learning rate λ in gradient boosting.

Choosing a Learning Rate

For a constant learning rate λ :



If λ is too small, it takes too many iterations to reach the optimum.



If λ is too large, the algorithm may ‘bounce’ around the optimum and never get sufficiently close.

Choosing a Learning Rate

Choosing λ :

- If λ is a constant, then it should be tuned through cross validation.
- For better results, use a variable λ . That is, let the value of λ depend on the gradient

We will learn this later:
RMSProp and Adam?

$$\lambda = h(||\nabla f(x)||)$$

where $\nabla f(x)$ is the magnitude of the gradient. So

- around the optimum, when the gradient is small, λ should be small
- far from the optimum, when the gradient is large, λ should be larger

Termination

Under ideal conditions, gradient descent iteratively approximates and converges to the optimum.

When do we terminate gradient descent?

- We can limit the number of iterations in the descent. But for an arbitrary choice of maximum iterations, we cannot guarantee that we are sufficiently close to the optimum in the end.
- If the descent is stopped when the updates are sufficiently small (e.g., the residuals of T are small), we encounter a new problem: the algorithm may never terminate!

Thank you



AdaBoost



CS109A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai, Chris Gumb

Mila Ivanovska
Bitola



Lecture Outline

- AdaBoost
- Mathematical Formulation - AdaBoost
- Final Thoughts on Boosting

Lecture Outline

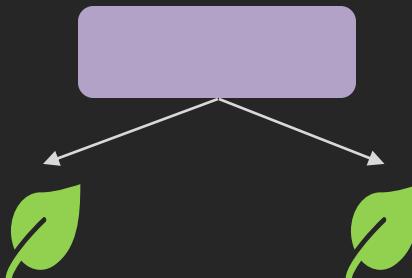
- AdaBoost
- Mathematical Formulation - AdaBoost
- Final Thoughts on Boosting

AdaBoost

There are two main ideas in AdaBoost:

- Idea #1: **Iteratively** build a complex model T by combining several **weak learners**.

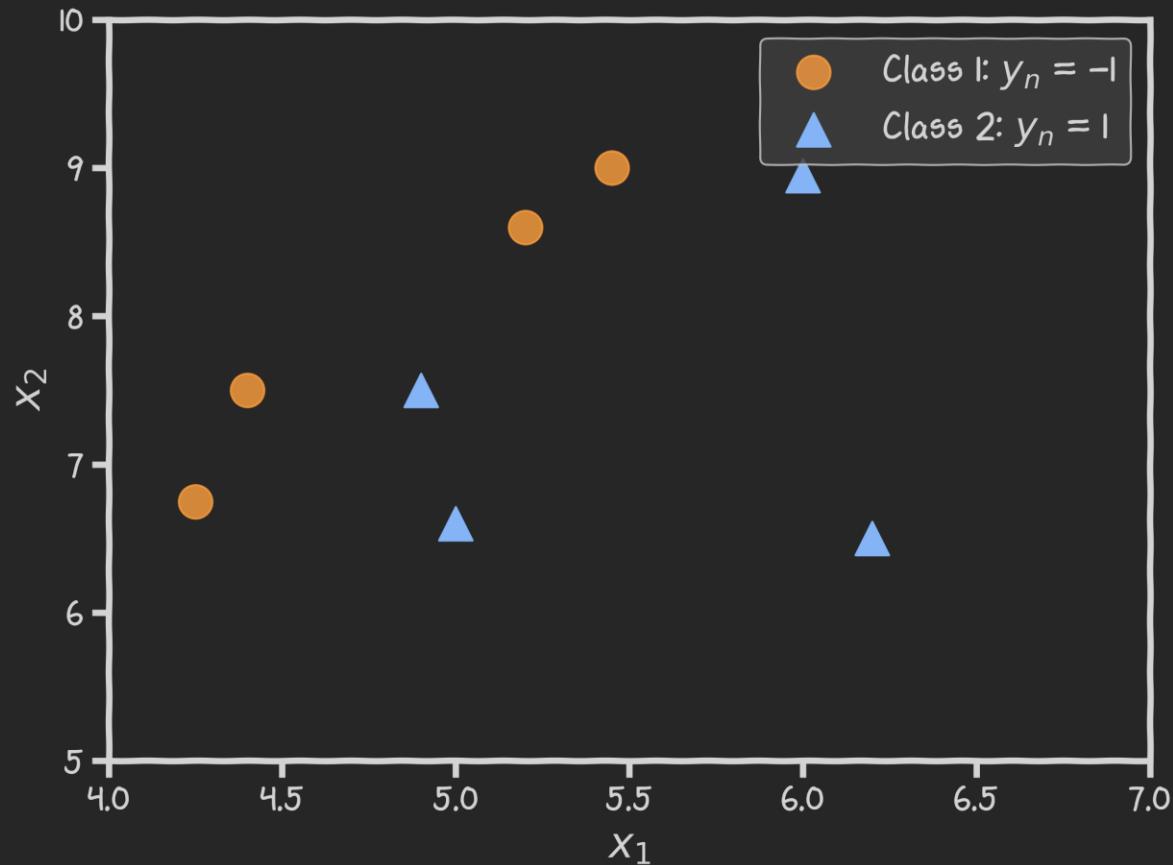
For trees, a weak learner is a tree with 1 node with 2 leaves. We call this a **stump**.



- Idea #2: Each new stump added to the ensemble model T learns from the **mistakes** of the **current model** T . This is done by assigning higher weights to the observations that the current model incorrectly classifies.

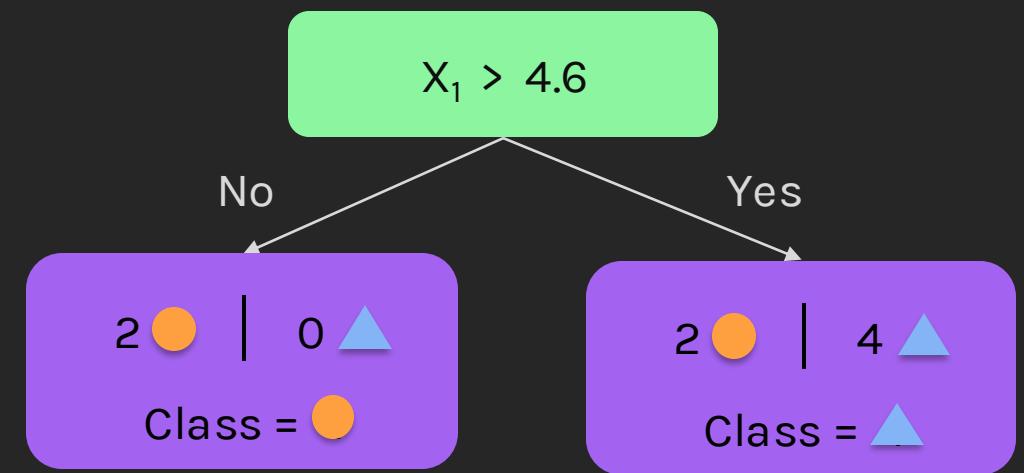
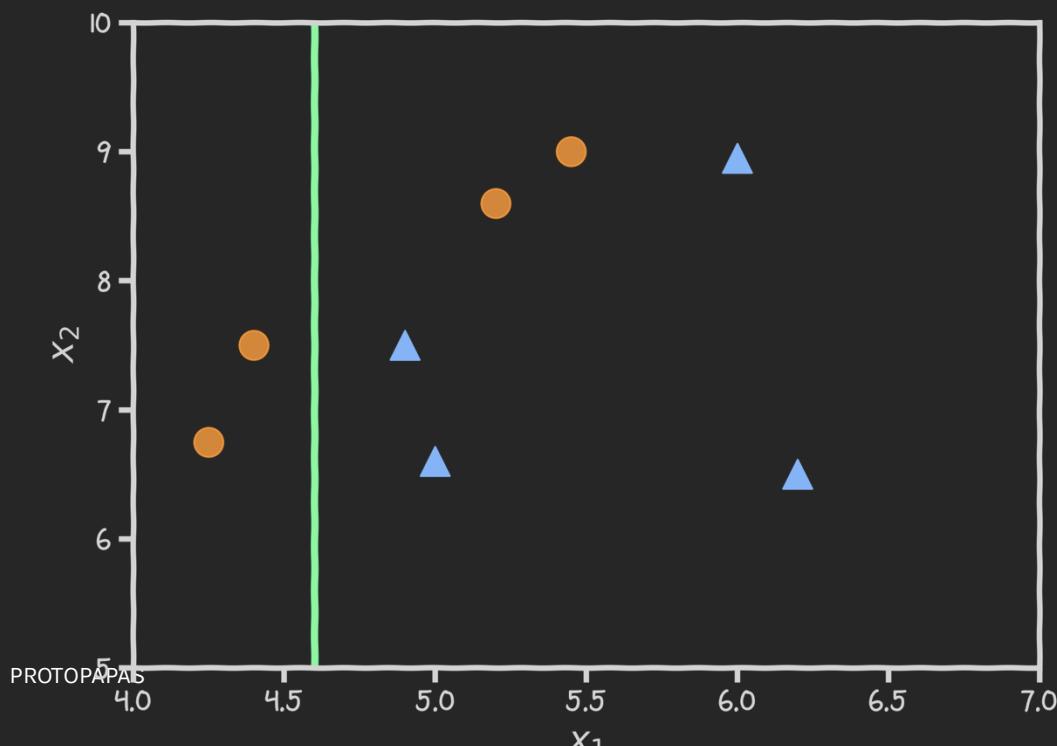
AdaBoost

Consider the following dataset:



AdaBoost

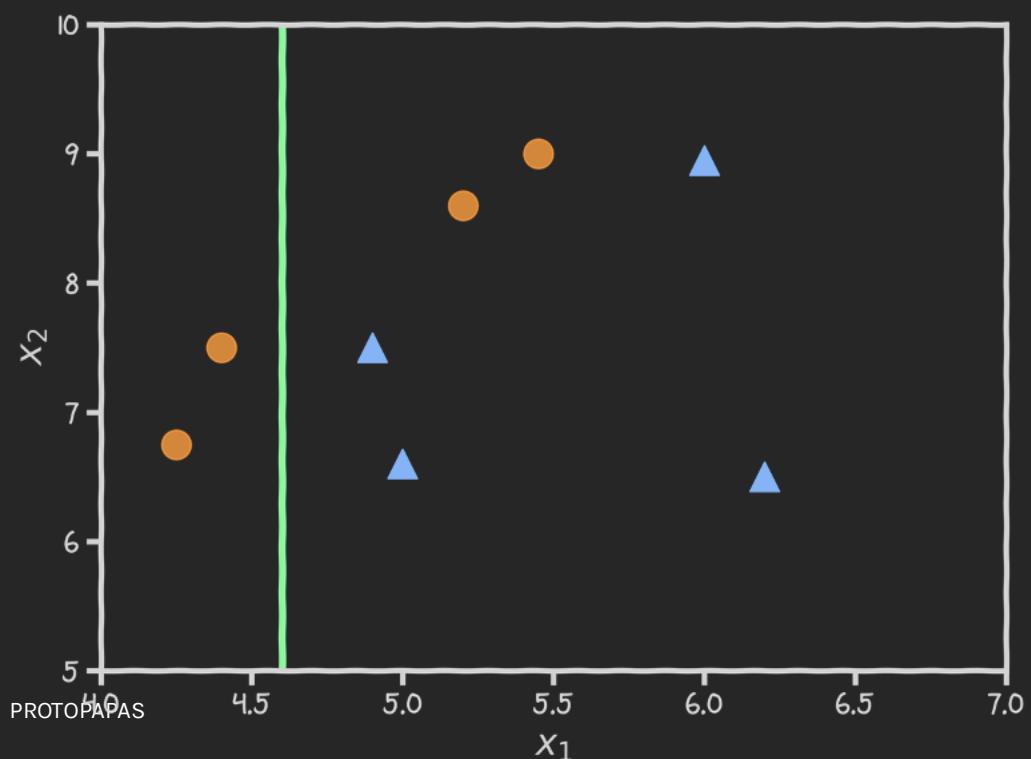
Step 1: Fit a stump $S^{(0)}$ on the dataset.



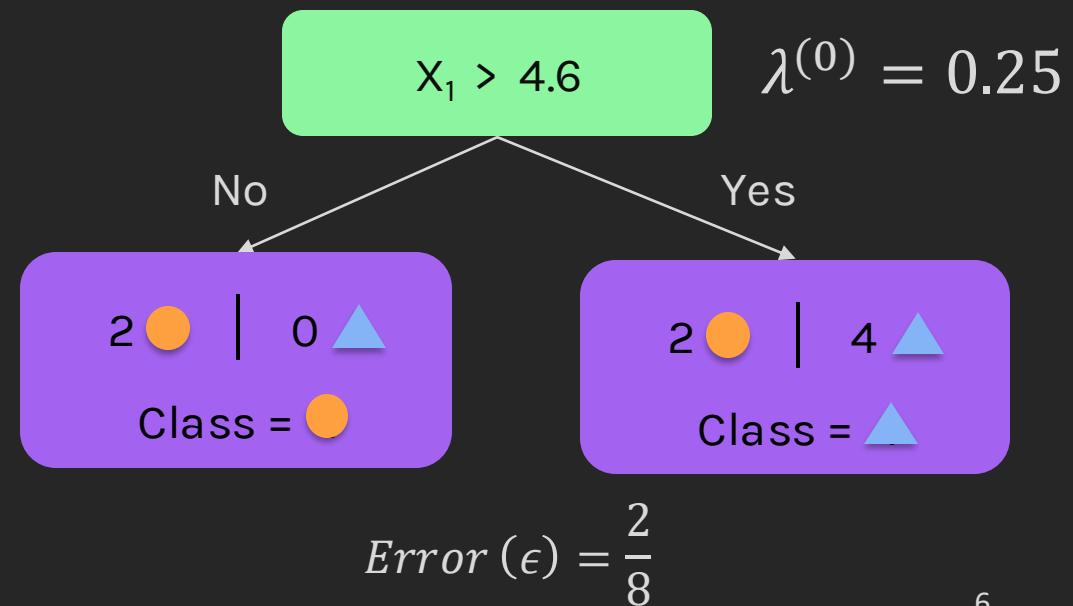
AdaBoost

Step 3: Now that we have the first weak learner, we will assign the stump a scaling factor, $\lambda^{(0)}$; The scaling factor indicates how much the stump **contributes to the entire ensemble**.

We assign λ to each successive stump because it offers some flexibility, and we can give more importance to stumps that perform better.



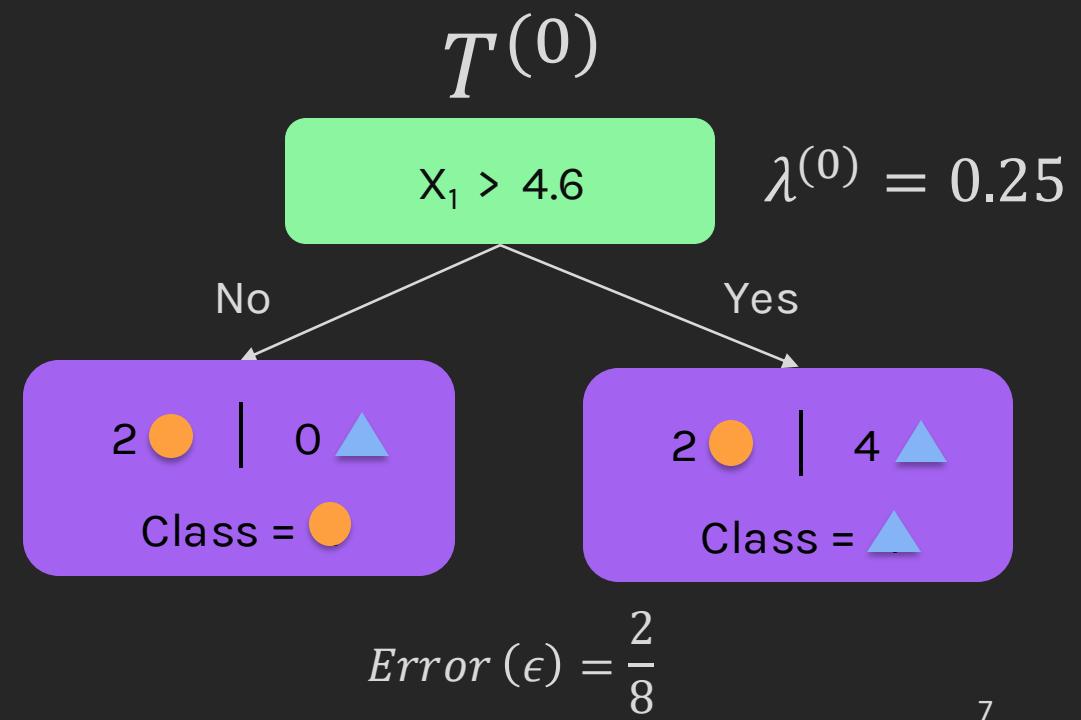
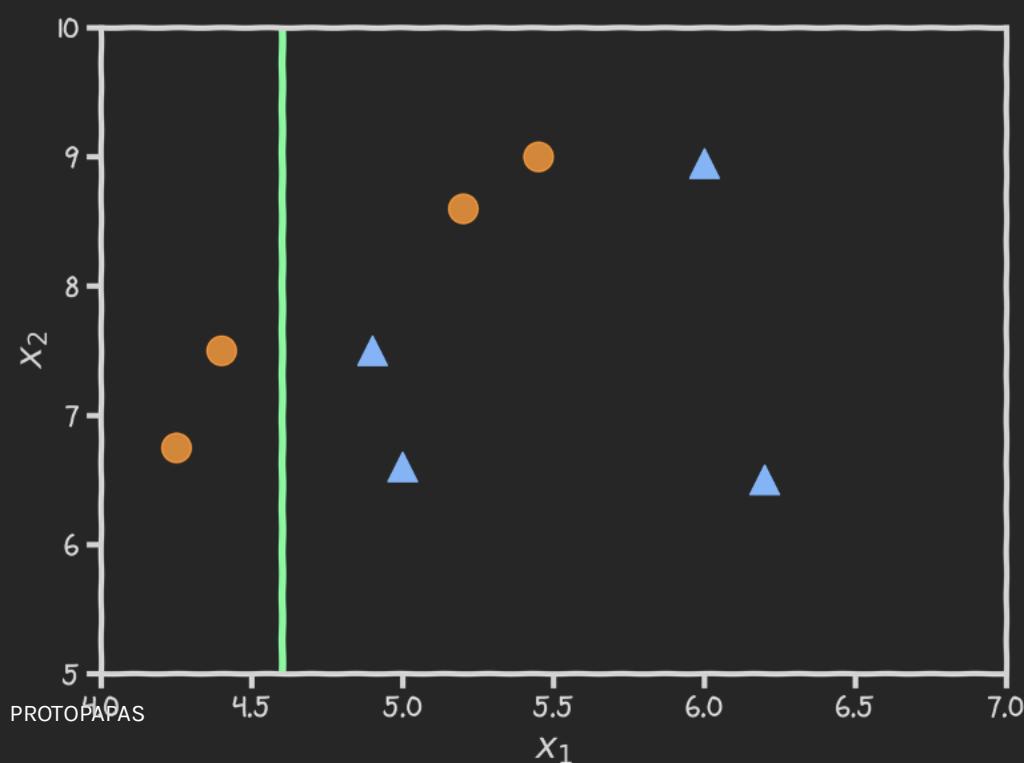
Let this model's scaling factor λ be 0.25.



AdaBoost

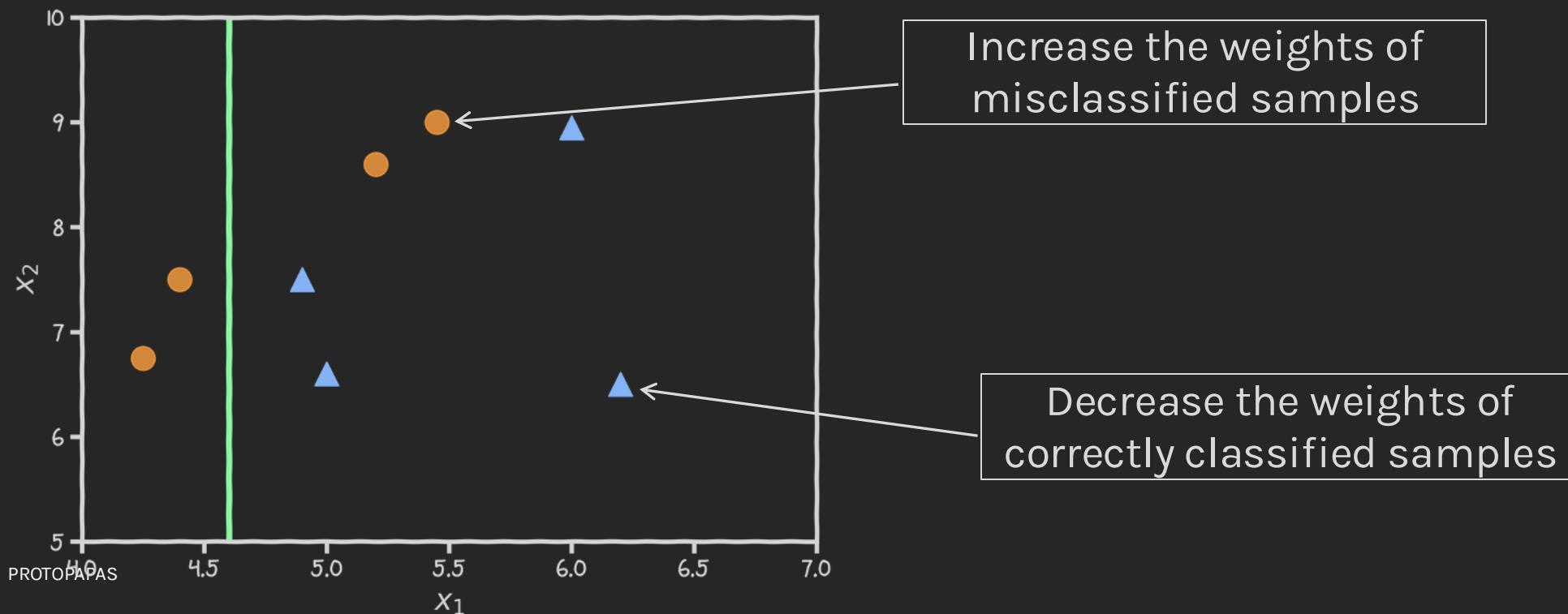
Step 4: Construct the **ensemble model** $T^{(0)}$ using:

$$T^{(i)} \leftarrow \begin{cases} \lambda^{(i)} S^{(i)} & i = 0 \\ T^{(i-1)} + \lambda^{(i)} S^{(i)} & i = 1, 2, \dots \end{cases}$$



AdaBoost

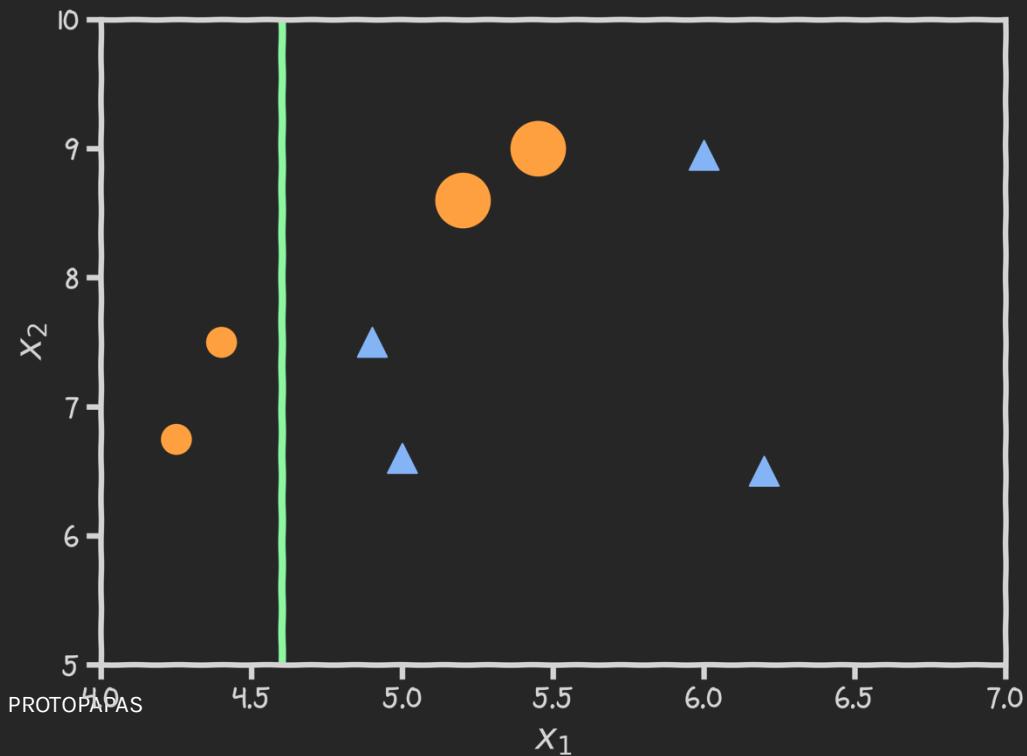
Step 5: Adjust the weights assigned to each data point to ensure the next stump focuses on the points **misclassified by the current model**.



AdaBoost

Step 5: Adjust the weights assigned to each data point to ensure the next stump focuses on the points **misclassified by the current model**.

$$w_n^{(1)} \leftarrow \frac{w_n^{(0)} e^{-\lambda y_n T^{(0)}(x_n)}}{Z} = \frac{w_n^{(1')}}{Z}$$



AdaBoost

ground truth -1 and 1

prediction from previous ensemble

$$w_n^{(1)} \leftarrow \frac{w_n^{(0)} e^{-\lambda y_n T^{(0)}(x_n)}}{Z} = \frac{w_n^{(1')}}{Z}$$

new weights

previous weights

normalizes the new weights

The diagram illustrates the AdaBoost weight update formula. The central equation is:

$$w_n^{(1)} \leftarrow \frac{w_n^{(0)} e^{-\lambda y_n T^{(0)}(x_n)}}{Z} = \frac{w_n^{(1')}}{Z}$$

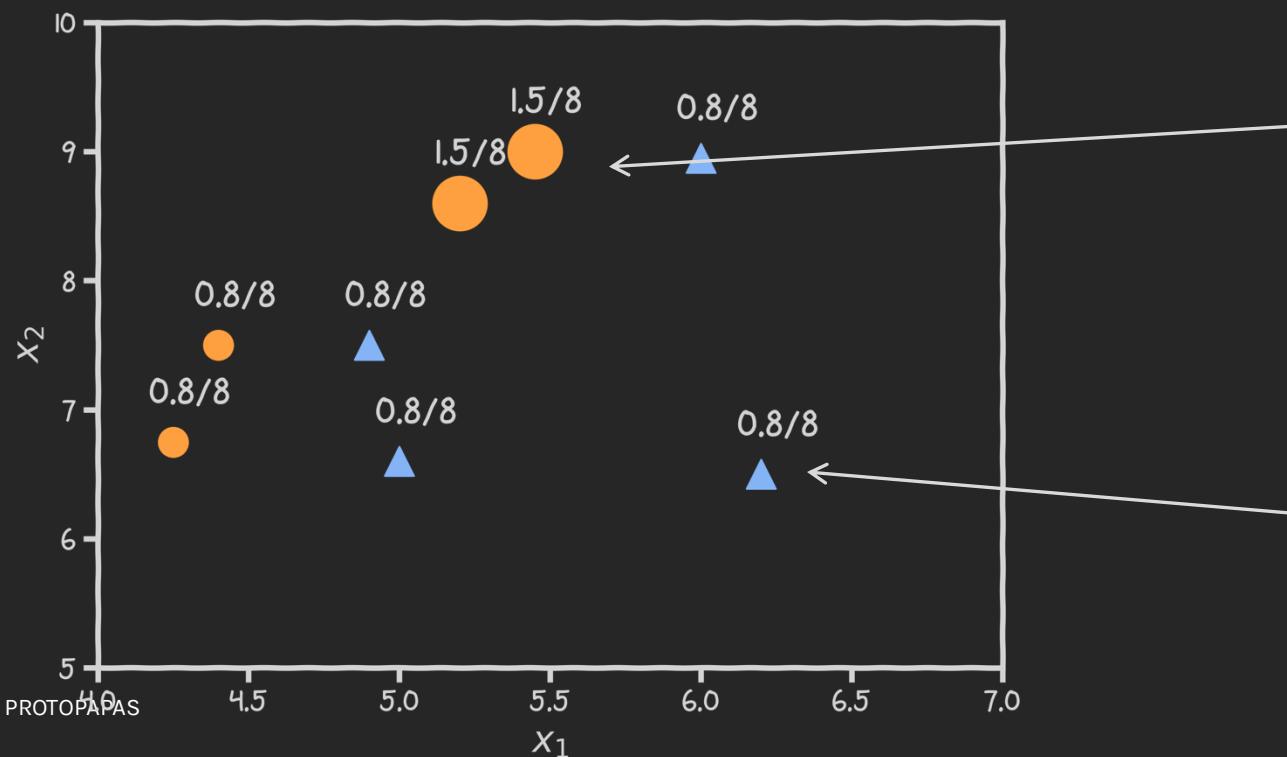
Annotations provide context for each term:

- "ground truth -1 and 1" is associated with the y_n term.
- "prediction from previous ensemble" is associated with the $T^{(0)}(x_n)$ term.
- "new weights" is associated with the final output $w_n^{(1)}$.
- "previous weights" is associated with the initial weight $w_n^{(0)}$.
- "normalizes the new weights" is associated with the normalization factor Z .

AdaBoost

Step 5: Adjust the weights assigned to each data point to ensure the next stump focuses on the points **misclassified by the ensemble**.

$$w_n^{(1)} \leftarrow \frac{w_n^{(0)} e^{-\lambda y_n T^{(0)}(x_n)}}{Z} = \frac{w_n^{(1')}}{Z}$$

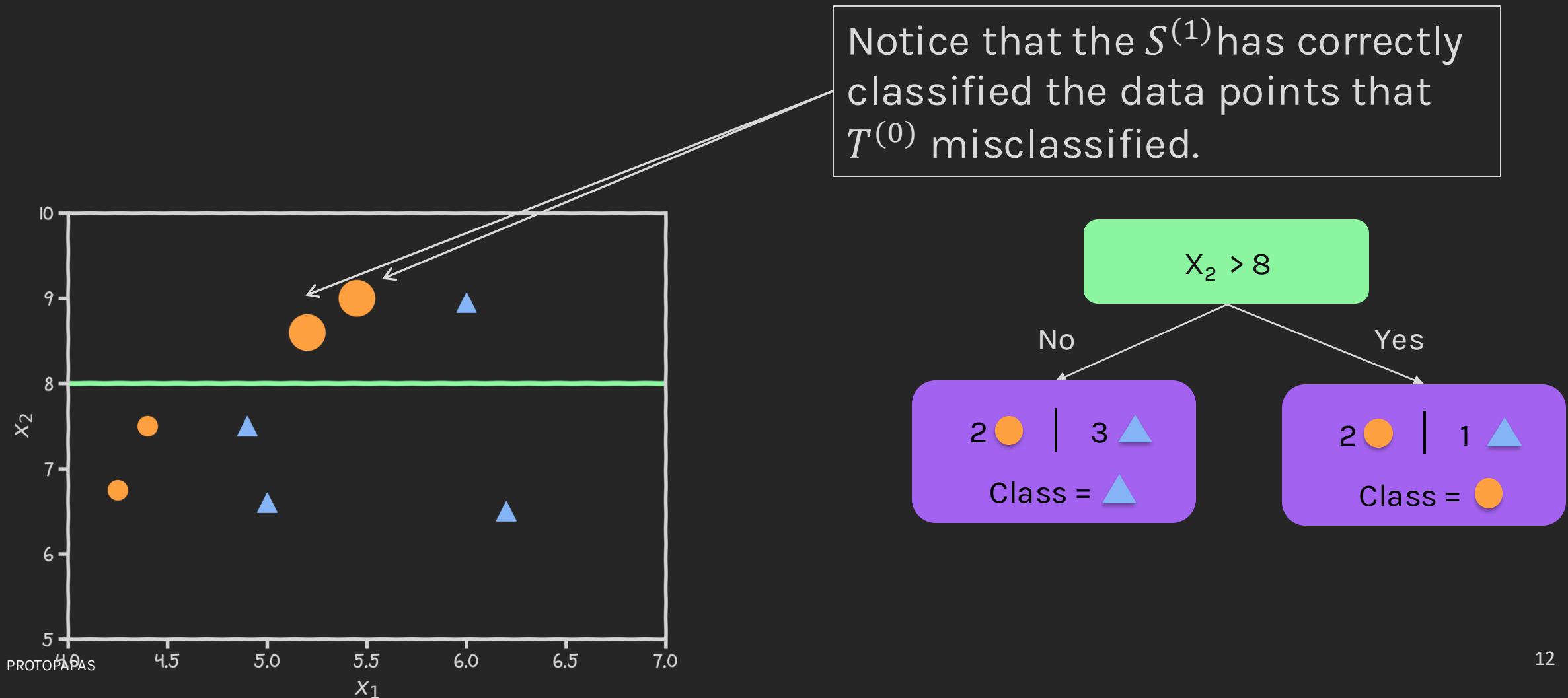


$$w^{(1)} \leftarrow \frac{w^{(1')}}{Z} \approx \frac{1.5}{8}$$

$$w^{(1)} \leftarrow \frac{w^{(1')}}{Z} \approx \frac{0.8}{8}$$

AdaBoost

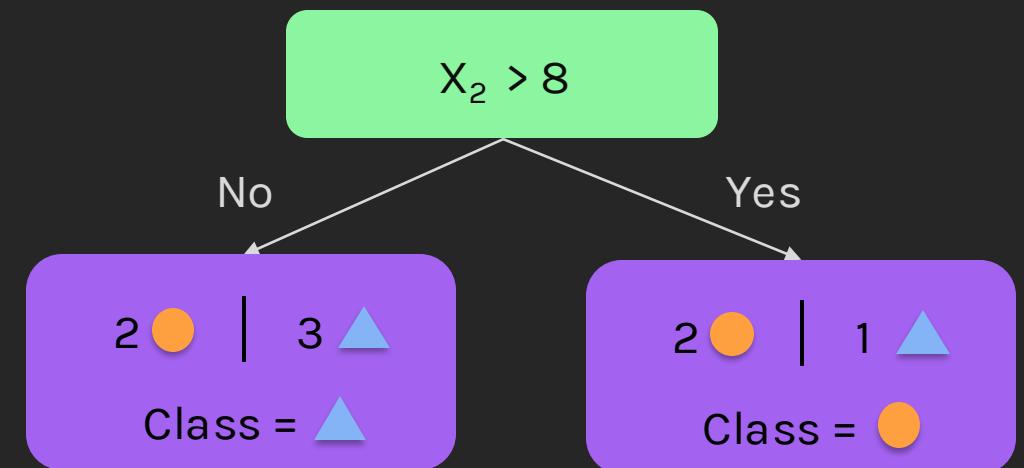
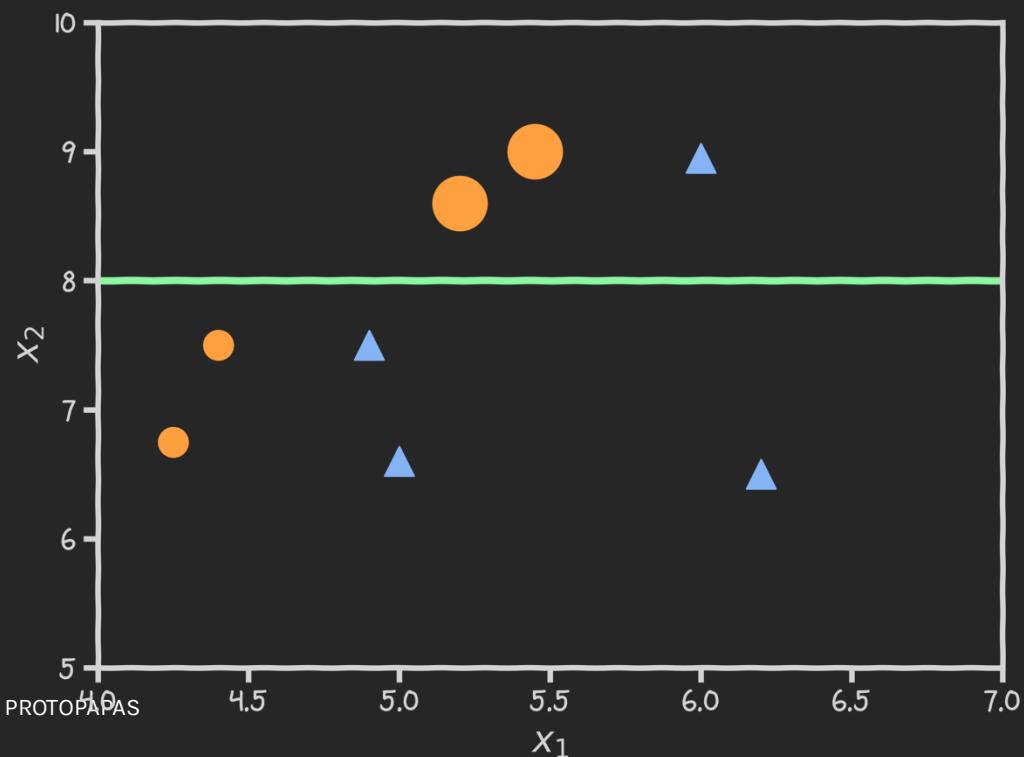
Step 6: Create another stump $S^{(1)}$ on the re-weighted data.



AdaBoost

Step 7: With the new weights, calculate the total error in the stump using:

$$\epsilon^{(1)} = \sum_{n=1}^N w_n^{(1)} \mathbb{I} (S^{(1)}(x_n) \neq y_n)$$



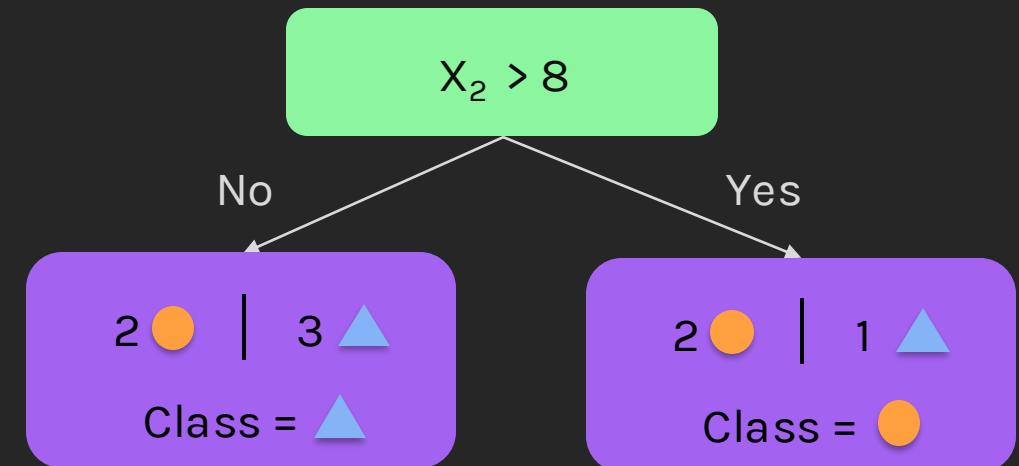
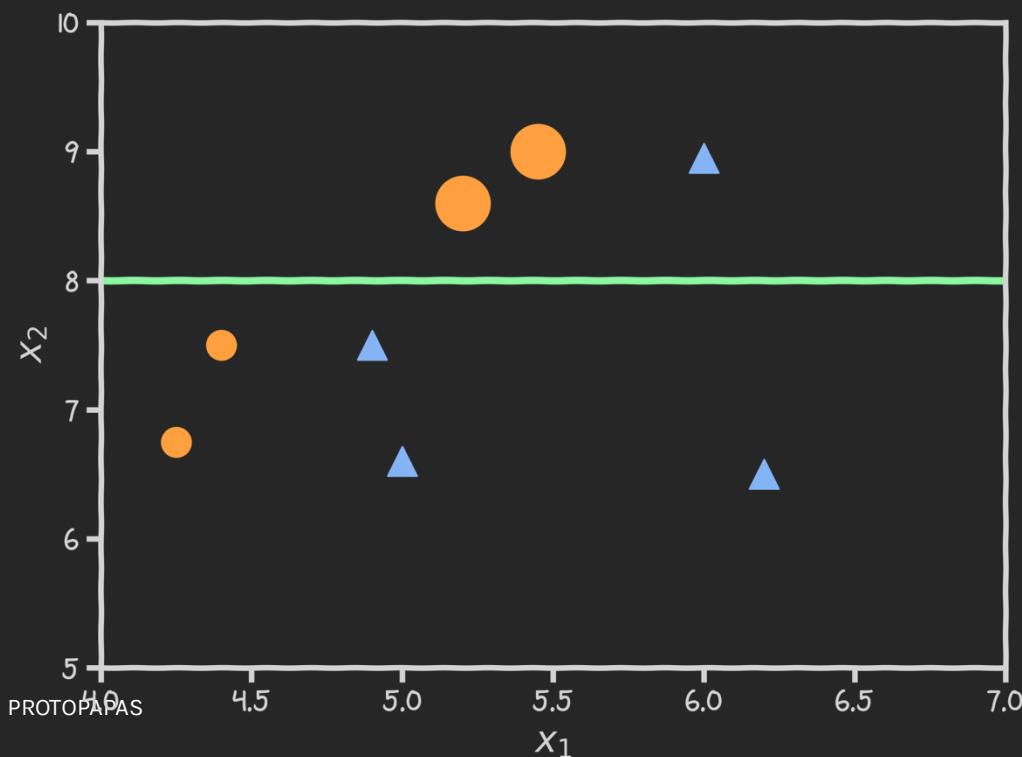
$$Error (\epsilon) = \frac{0.8}{8} + \frac{0.8}{8} + \frac{0.8}{8} = 0.3$$

AdaBoost

Step 8: Assign the stump a scale, $\lambda^{(1)}$, that indicates how much it **contributes to the entire ensemble**.

Let this STUMP'S weight $\lambda^{(1)}$ be 0.20.

$$\lambda^{(1)} = 0.20$$

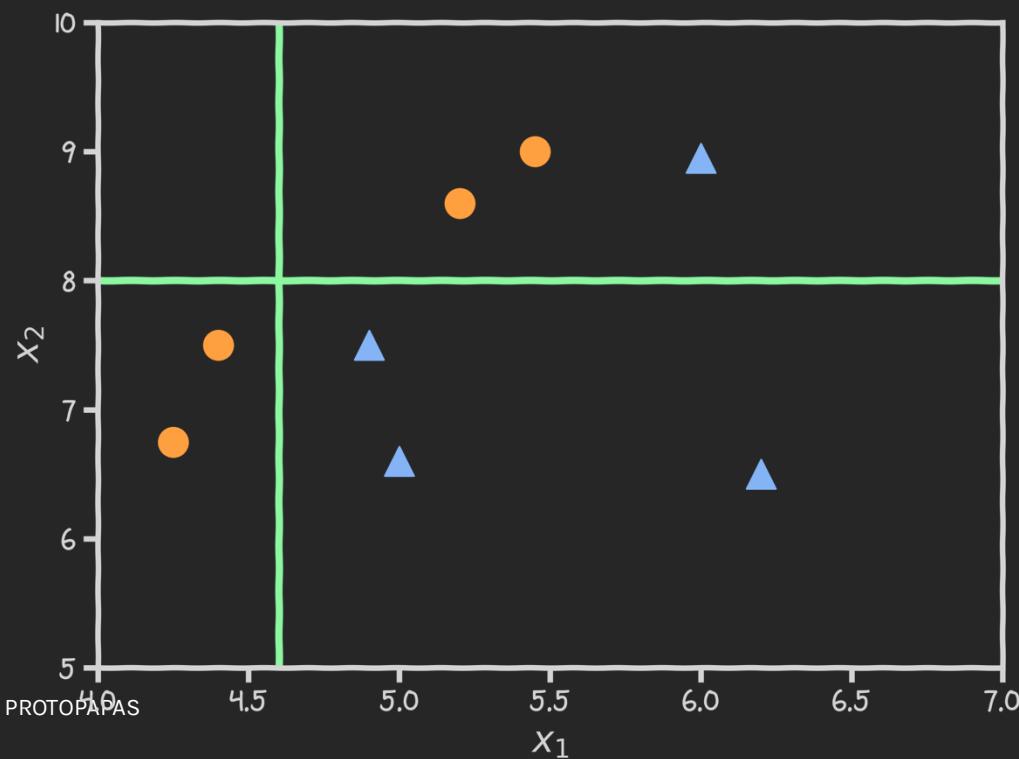


$$Error (\epsilon) = 0.3$$

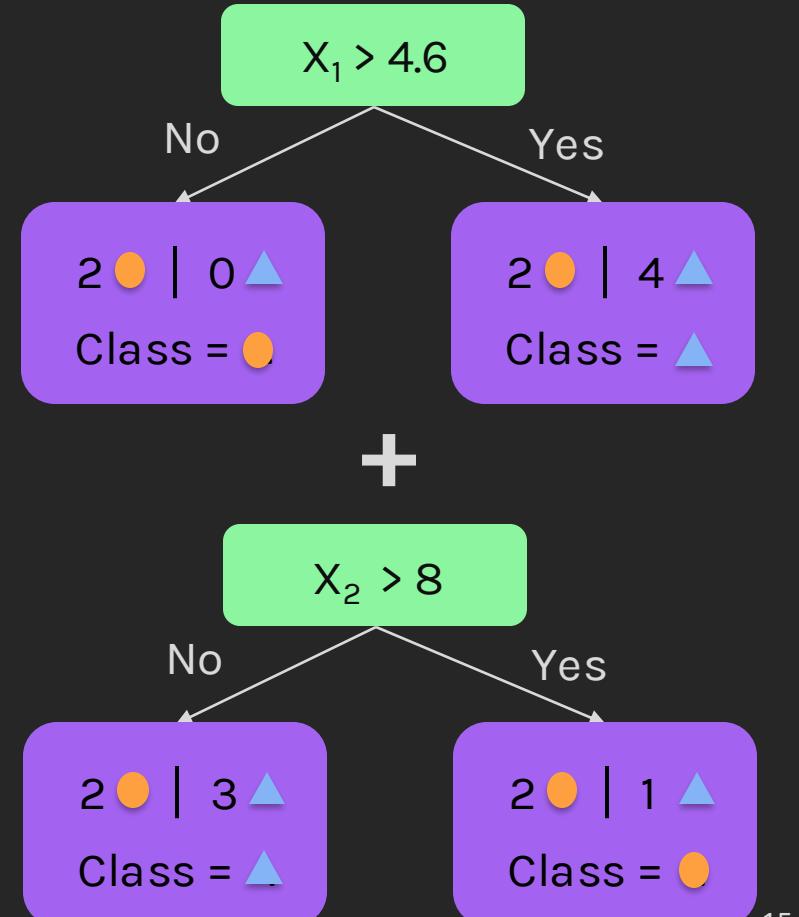
AdaBoost

Step 9: Construct the **ensemble model** $T^{(1)}$ using:

$$T^{(i)} \leftarrow \begin{cases} \lambda^{(i)} S^{(i)} & i = 0 \\ T^{(i-1)} + \lambda^{(i)} S^{(i)} & i = 1, 2, \dots \end{cases}$$



0.25 *

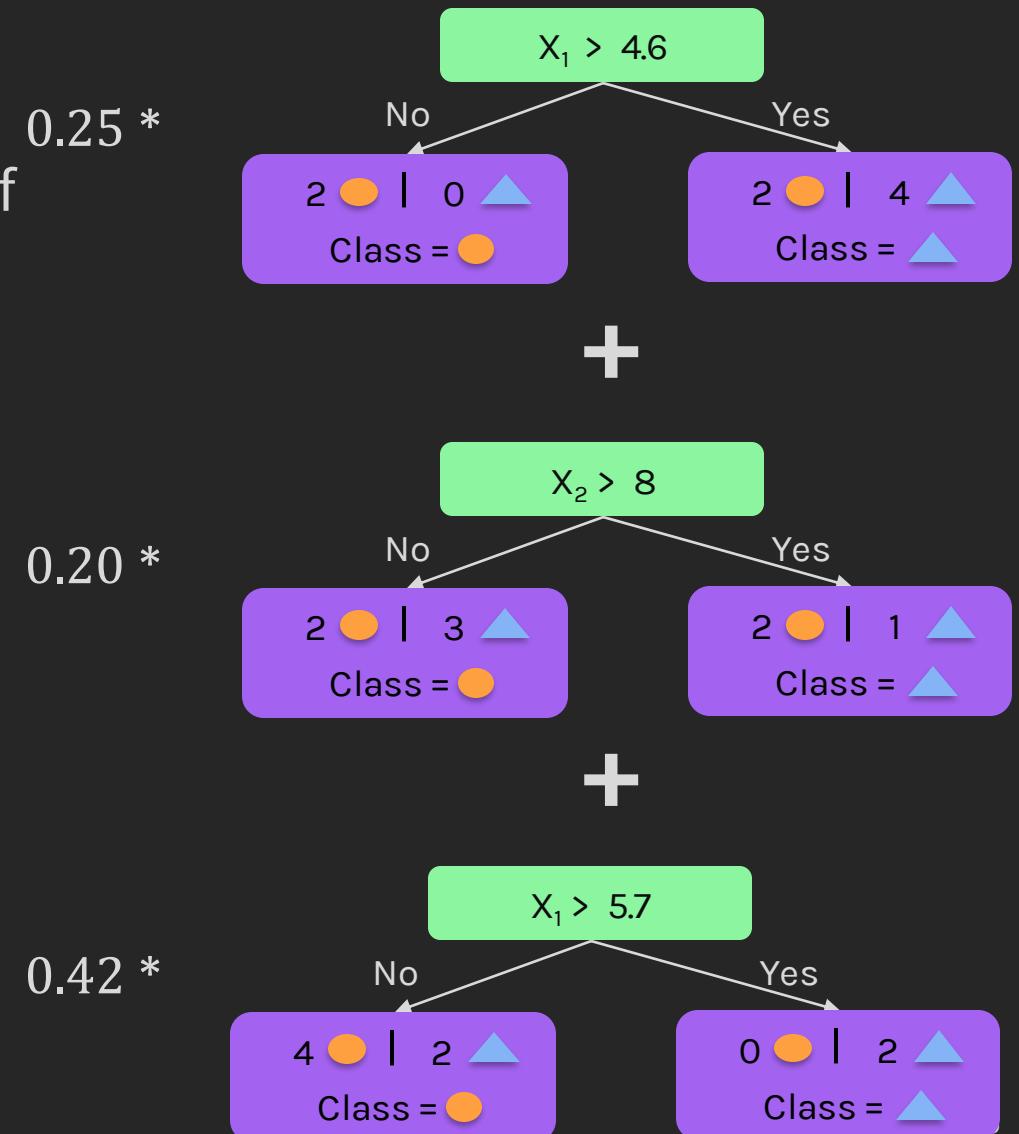
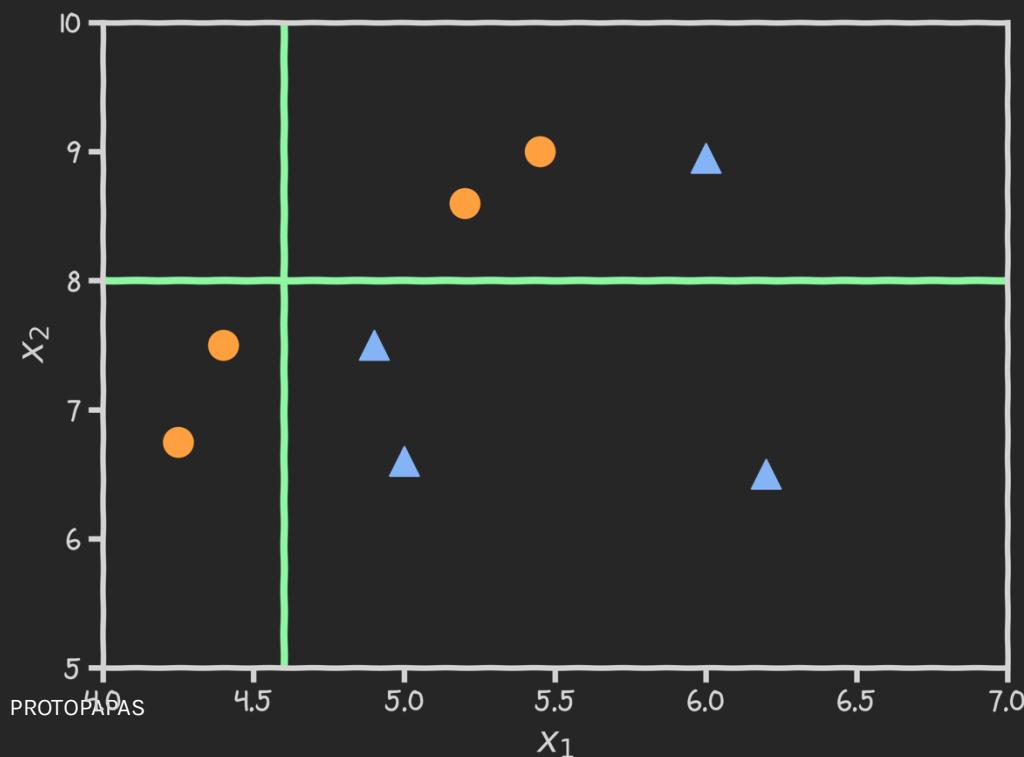


0.20 *

AdaBoost

Repeating the same process again, we get:

Thus, the ensemble keeps growing in terms of the number of stumps being used.



AdaBoost: SUMMARY

Given n , training data points, set $w=1/n$

For $i = 0 \dots$ until stopping condition is met:

- Train a weak learner $S^{(i)}$ using weights $w_n^{(i)}$.
- Calculate the total error, $\epsilon^{(i)}$, of the weak learner.
- Calculate the importance of each model $\lambda^{(i)}$.
- Combine all stumps into the new ensemble model, $T^{(i)}$:
- Adjust the weights assigned to each data point to ensure the next stump focuses on the points misclassified by the previous stump

AdaBoost

Step 1: Given training data $\{(x_1, y_1), \dots, (x_N, y_N)\}$, choose an initial distribution $w_n^{(0)} = 1/N$.

For $i = 0 \dots$ until stopping condition is met:

Step 2: Train a weak learner $S^{(i)}$ using weights $w_n^{(i)}$.

Step 3: Calculate the total error of the weak learner using:

$$\epsilon^{(i)} = \sum_{n=1}^N w_n^{(i)} \mathbb{I}(y_n \neq S^{(i)}(x_n))$$

Step 4: Calculate the importance of each model $\lambda^{(i)}$.

Step 5: Construct the ensemble model using:

$$T^{(i)} \leftarrow \begin{cases} \lambda^{(i)} S^{(i)} & i = 0 \\ T^{(i-1)} + \lambda^{(i)} S^{(i)} & i = 1, 2, \dots \end{cases}$$

Step 6: Adjust the weights assigned to each data point to ensure the next stump focuses on the points misclassified by the previous stump

$$w_n^{(i+1)} \leftarrow \frac{w_n^{(i)} e^{-\lambda^{(i)} y_n T^{(i)}(x_n)}}{Z}, \text{ where } T^{(i)}(x) = \text{sign}[T^{(i-1)}(x) + \lambda^{(i)} S^{(i)}(x)]$$

Final model: $T^{(i)}(x) = \text{sign} \left[\sum_{i=1}^T \lambda^{(i)} S^{(i)}(x) \right]$

AdaBoost

Step 1: Given training data $\{(x_1, y_1), \dots, (x_N, y_N)\}$, choose an initial distribution $w_n^{(0)} = 1/N$.

For $i = 0 \dots$ until stopping condition is met:

Step 2: Train a weak learner $S^{(i)}$ using weights $w_n^{(i)}$.

Step 3: Calculate the total error of the weak learner using:

GREAT PROFESSORS always KNOW your next questions!

$$\epsilon^{(i)} = \sum_{n=1}^N w_n^{(i)} \mathbb{I}(y_n \neq S^{(i)}(x_n))$$

Step 4: Calculate the importance of each model $\lambda^{(i)}$.

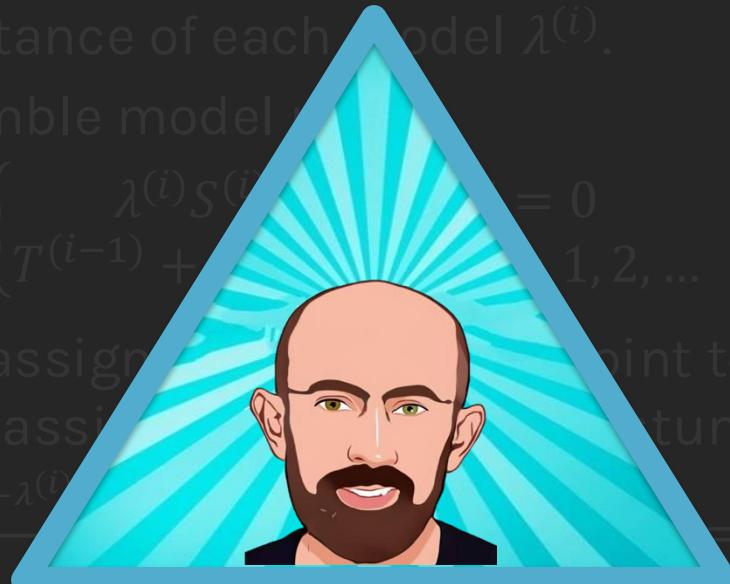
Step 5: Construct the ensemble model $T^{(i)}$:

$$T^{(i)} \leftarrow \begin{cases} \lambda^{(i)} S^{(i)} & \text{if } \epsilon^{(i)} < 0 \\ T^{(i-1)} + \lambda^{(i)} S^{(i)} & \text{if } i > 1, 2, \dots \end{cases}$$

Step 6: Adjust the weights assigned by the previous stump to ensure the next stump focuses on the points misclassified by the previous stump

$$w_n^{(i+1)} \leftarrow \frac{w_n^{(i)} e^{-\lambda^{(i)} y_n}}{\sum_n w_n^{(i)} e^{-\lambda^{(i)} y_n}} = sign[T^{(i-1)}(x) + \lambda^{(i)} S^{(i)}(x)]$$

Final model: $T^{(i)}(x) = sign \left[\sum_{j=1}^T \lambda^{(j)} S^{(j)}(x) \right]$



AdaBoost

Step 1: Given training data $\{(x_1, y_1), \dots, (x_N, y_N)\}$, choose an initial distribution $w_n^{(0)} = 1/N$.

For $i = 0 \dots$ until stopping condition is met:

Step 2: Train a weak learner $S^{(i)}$ using weights $w_n^{(i)}$. 

Step 3: Calculate the total error of the weak learner using:

$$\epsilon^{(i)} = \sum_{n=1}^N w_n^{(i)} \mathbb{I}(y_n \neq S^{(i)}(x_n))$$

Step 4: Calculate the importance of each model $\lambda^{(i)}$. 

Step 5: Construct the ensemble model using:

$$T^{(i)} \leftarrow \begin{cases} \lambda^{(i)} S^{(i)} & i = 0 \\ T^{(i-1)} + \lambda^{(i)} S^{(i)} & i = 1, 2, \dots \end{cases}$$

Step 6: Adjust the weights assigned to each data point to ensure the next stump focuses on the points misclassified by the previous stump

$$w_n^{(i+1)} \leftarrow \frac{w_n^{(i)} e^{-\lambda^{(i)} y_n T^{(i)}(x_n)}}{Z}, \text{ where } T^{(i)}(x) = \text{sign}[T^{(i-1)}(x) + \lambda^{(i)} S^{(i)}(x)]$$

Final model: $T^{(i)}(x) = \text{sign} \left[\sum_{i=1}^T \lambda^{(i)} S^{(i)}(x) \right]$

AdaBoost

- **How do I create a stump?**

In AdaBoost the stumps are created using simple decision trees with `max_depth = 1`.

- **How to use the normalized weights to make the stump?**

There are two options:

- A. Create a new dataset of same size of the original dataset with **repetition** based on the newly updated sample weight.
- B. Use a weighted version of the Gini index.

- **How do I calculate the scaling factor $\lambda^{(i)}$ of each stump?**

Next few slides...



PPPP BOOSTING

AdaBoost

Recall in gradient boosting for regression, we minimize the MSE loss.

In AdaBoost, we minimize a different function, we call **exponential loss**:

$$\text{Exp Loss} = \frac{1}{N} \sum_{n=1}^N e^{(-y_n \hat{y}_n)} \quad \text{where } y_n \in \{-1, 1\}$$

Exponential loss is differentiable with respect to \hat{y}_n and it is an upper bound of Error.

Choosing the Learning Rate

Unlike in the case of gradient boosting for regression, we can analytically solve for the optimal learning rate for AdaBoost, by optimizing:

$$\operatorname{argmin}_{\lambda} \frac{1}{N} \sum_{n=1}^N e^{(-y_n(T + \lambda^{(i)} S^{(i)}(x_n)))}$$

Doing so, we get: $\lambda^{(i)} = \frac{1}{2} \ln \left(\frac{1 - \epsilon}{\epsilon} \right)$ where

$$\epsilon = \sum_{n=1}^N w_n^{(i)} \mathbb{I}(y_n \neq T^{(i)}(x_n)) \quad \text{and} \quad w_n^{(i+1)} \leftarrow \frac{w_n^{(i)} e^{-\lambda^{(i)} y_n S^{(i)}(x_n)}}{Z}$$

Lecture Outline

- AdaBoost
- **Mathematical Formulation - AdaBoost**
- Final Thoughts on Boosting

Gradient Descent with Exponential Loss

$$\text{Exp Loss} = \frac{1}{N} \sum_{n=1}^N e^{(-y_n \hat{y}_n)}$$

Similar to what we did for gradient boosting, compute the gradient for the loss w.r.t the predictions:

$$\nabla_{\hat{y}} \text{Exp Loss} = [-y_1 e^{(-y_1 \hat{y}_1)}, \dots, -y_N e^{(-y_N \hat{y}_N)}]$$

Set $w_n = \exp(-y_n \hat{y}_n)$. Notice that when $y_n = \hat{y}_n$, the weight w_n is small; when $y_n \neq \hat{y}_n$, the weight is larger.

$$\nabla_{\hat{y}} \text{Exp Loss} = [-y_1 w_1, \dots, -y_N w_N]$$

This way, we see that the gradient is just a re-weighting applied to the target values.₂₆

Gradient Descent with Exponential Loss

The update step in the gradient descent is

$$\hat{y}_n \leftarrow \hat{y}_n + \lambda w_n y_n, \quad n = 1, \dots, N$$

Just like in gradient boosting, we approximate the gradient, $\lambda w_n y_n$, with a simple model, $S^{(i)}$, that depends on x_n .

This means training $S^{(i)}$ on a re-weighted set of target values,

$$\{(x_1, w_1 y_1), \dots, (x_N, w_N y_N)\}$$

That is, gradient descent with exponential loss means iteratively training simple models that **focuses on the points misclassified by the previous model**.

Comparison: Gradient Boosting and AdaBoost

GRADIENT BOOST

We minimize the **MSE**:

$$MSE = \frac{1}{N} \sum_{n=1}^N (y_n - \hat{y}_n)^2$$

Compute the gradient for the loss w.r.t predictions:

$$\nabla_{\hat{y}} MSE = -2[r_1, \dots, r_n]$$

The update step is:

$$\hat{y}_n \leftarrow \hat{y}_n + \lambda \hat{r}_n, \quad n = 1, \dots, N$$

ADABOOST

We minimize the **exponential loss**:

$$Exp\ Loss = \frac{1}{N} \sum_{n=1}^N e^{(-y_n \hat{y}_n)} \quad \text{where} \quad y_n \in \{-1, 1\}$$

Compute the gradient for the loss w.r.t predictions:

$$\nabla_{\hat{y}} Exp\ Loss = [-y_1 e^{(-y_1 \hat{y}_1)}, \dots, -y_N e^{(-y_N \hat{y}_N)}]$$

Setting $w_n = \exp(-y_n \hat{y}_n)$:

$$\nabla_{\hat{y}} Exp\ Loss = [-y_1 w_1, \dots, -y_N w_N]$$

The update step is:

$$\hat{y}_n \leftarrow \hat{y}_n + \lambda w_n y_n, \quad n = 1, \dots, N$$

Lecture Outline

- AdaBoost
- Mathematical Formulation - AdaBoost
- **Final Thoughts on Boosting**

Final thoughts on Boosting

- **Stopping condition:** Same as gradient boosting: maximum iteration (number of stumps) $n_estimators$, minimum improvement in loss, number of iterations without improvement in the loss.
- **Overfitting:** Unlike other ensemble methods like bagging and Random Forest, boosting methods like AdaBoost will overfit if run for many iterations. Some libraries implement regularization methods which is out of the scope in this course.
- **Hyper-parameters:** All parameters associated with the stump and the stopping conditions mentioned before.

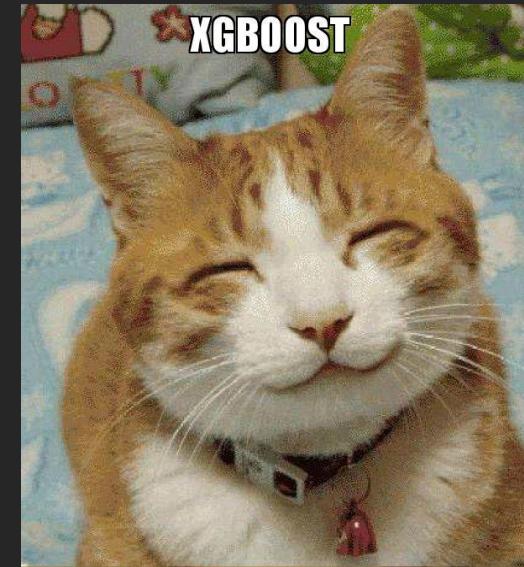
Final thoughts on Boosting

There are few implementations on boosting:

- **XGBoost**: An efficient Gradient Boosting Decision .
- **LGBM**: Light Gradient Boosted Machines. It is a library for training GBMs developed by Microsoft, and it competes with XGBoost.
- **CatBoost**: A new library for Gradient Boosting Decision Trees, offering appropriate handling of categorical features.

Everything you need to know about XGBoost (Extreme Gradient Boosting)!

- Highly optimized SoTA implementation of gradient boosting.
- **Regularization:** L1 and L2 regularization to prevent overfitting.
- **Parallelization:** Uses CPU cores for constructing trees in parallel.
- **Distributed Computing:** Scales well for large datasets using frameworks like Spark, Hadoop.
- **Cross-Validation:** Can tune hyperparameters easily using cross-validation.
- **Missing Values:** Handles missing values automatically.



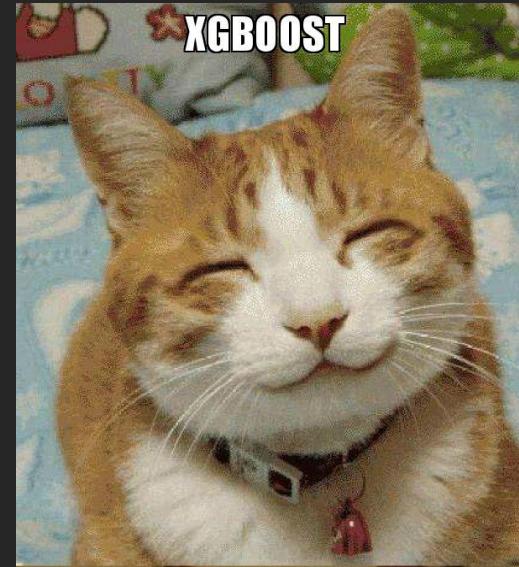
Everything you need to know about XGBoost (Extreme Gradient Boosting)!

Dataset Preparation: DMatrix, aka XGBoost's optimized data structure.

Model: XGBClassifier, XGBRegressor.

Hyperparameters:

- **Model Complexity:** {max_depth, min_child_weight, gamma}
- **Regularization:** {lambda, alpha}
- **Learning:** {learning_rate, n_estimators}
- **Loss Functions:** {objective, eval_metric}



The latest LLM



XGBoost





Students

Pavlos

Thank you

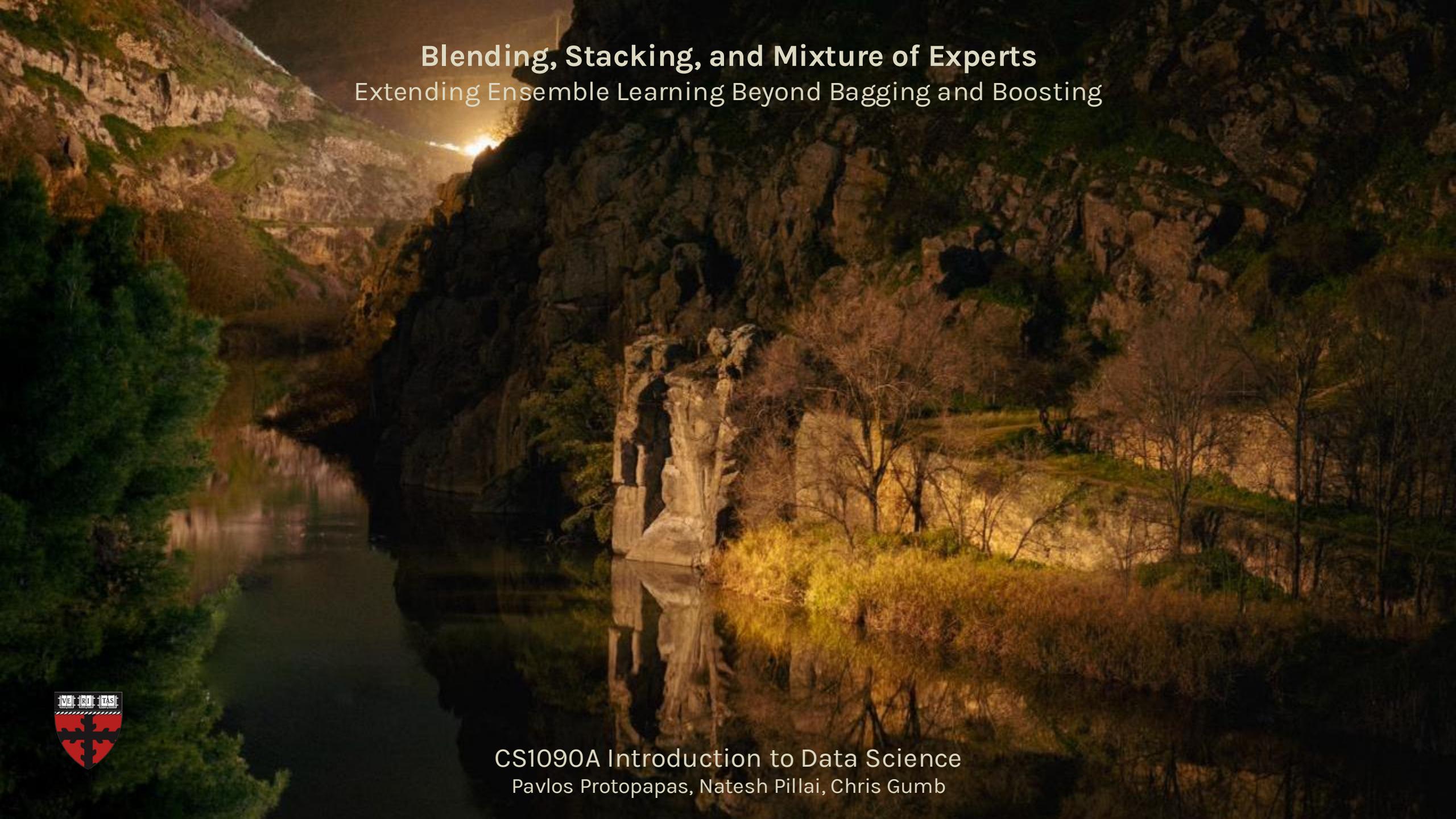


Likee



Thank you



The background of the slide is a photograph of a natural landscape. On the left, a dark, rocky cliff face rises, with patches of green vegetation and a small waterfall at the top. A bright sun is visible through a gap in the rocks, casting a warm glow over the scene. In the foreground, a calm body of water reflects the surrounding environment. To the right, there are more rocky outcrops and a dense forest of tall, thin trees. The overall atmosphere is serene and majestic.

Blending, Stacking, and Mixture of Experts

Extending Ensemble Learning Beyond Bagging and Boosting



CS1090A Introduction to Data Science
Pavlos Protopapas, Natesh Pillai, Chris Gumb

Outline

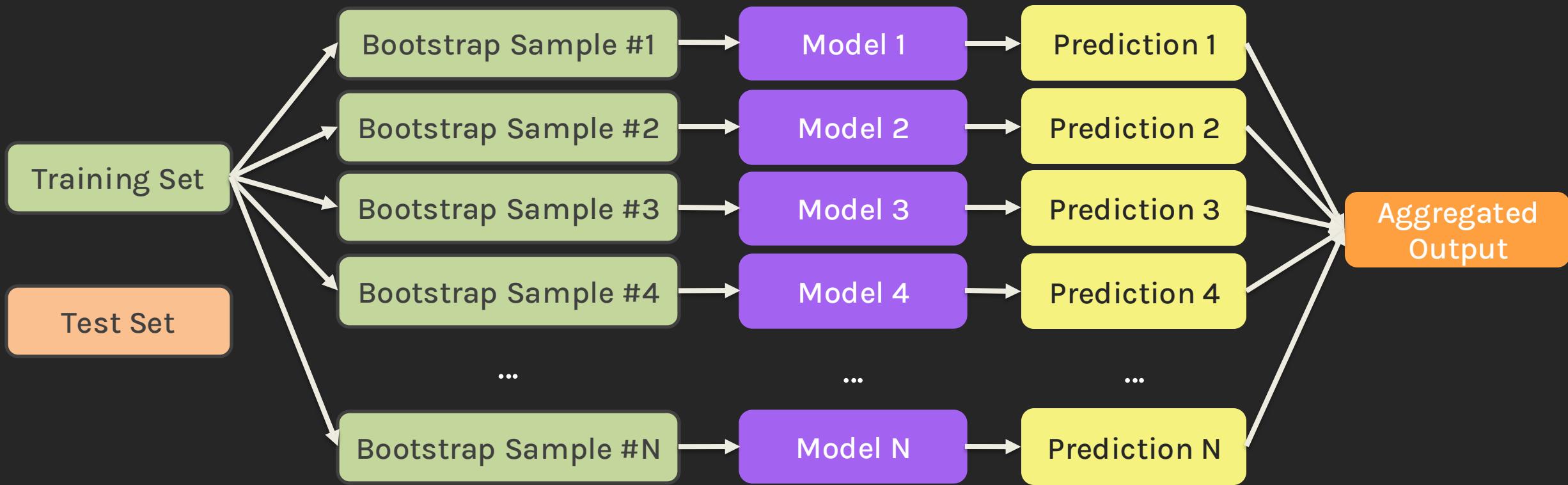
- Ensemble Methods
- Blending
- Stacking
- Mixture of Experts

Outline

- Ensemble Methods
- Blending
- Stacking
- Mixture of Experts

Recap: Bagging

Bagging (Bootstrap aggregating) trains models on each bootstrapped sample of the dataset and combines their predictions as the final output.



Recap: Bagging

Bagging (Bootstrap aggregating) trains models on each bootstrapped sample of the dataset and combines their predictions as the final output.

The base models are usually homogeneous ‘strong’ learners, i.e.,

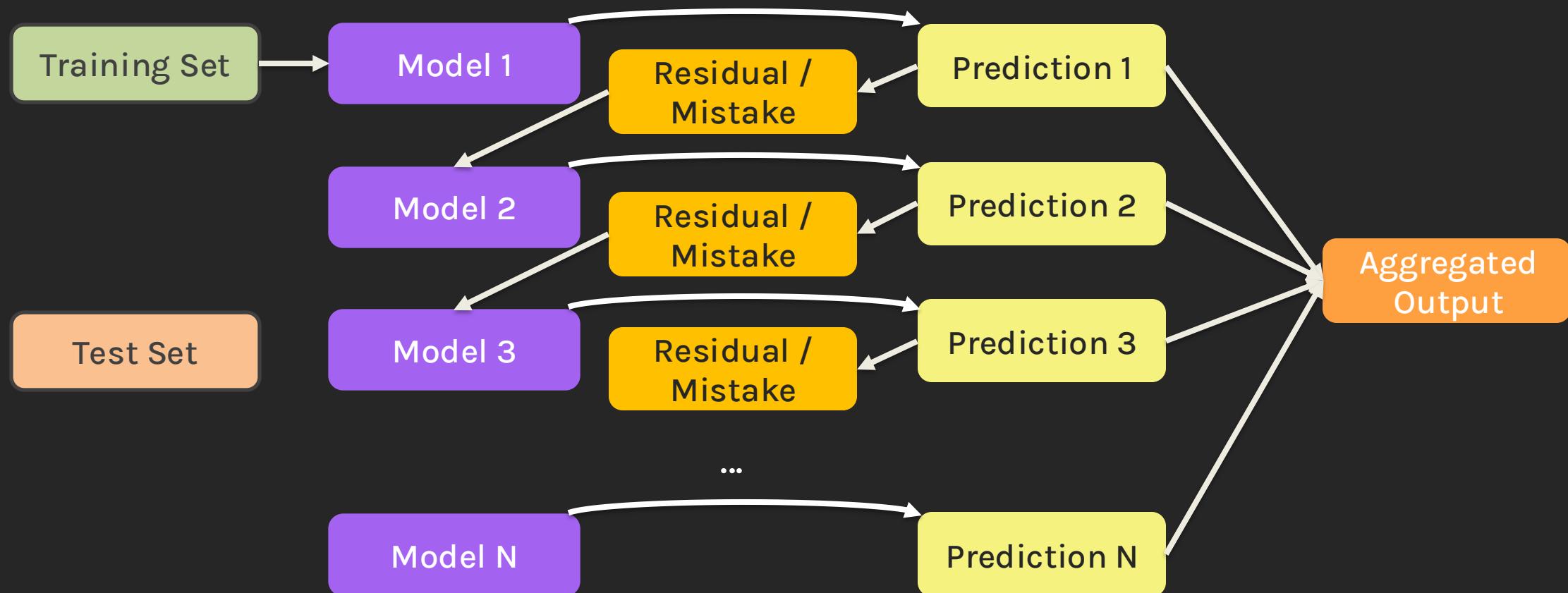
- They are the same type of models defined by same hyperparameters.
- They have low bias (and high variance) and tend to overfit.

Base models are trained in parallel, and their predictions are aggregated through:

- averaging for regression tasks
- majority voting for classification tasks

Recap: Boosting

In Boosting, base models are fit sequentially on the emphasis of residuals/mistakes from the previous ensemble, and final output is aggregated with different weights on base models' predictions.



Recap: Ensemble Learning

In Boosting, base models are fit sequentially on the emphasis of residuals/mistakes from the previous ensemble, and final output is aggregated with different weights on base models' predictions.

The base models are usually homogeneous ‘weak’ learners, i.e.,

- They are the same type of models defined by same hyperparameters.
- They have high bias and tend to underfit.

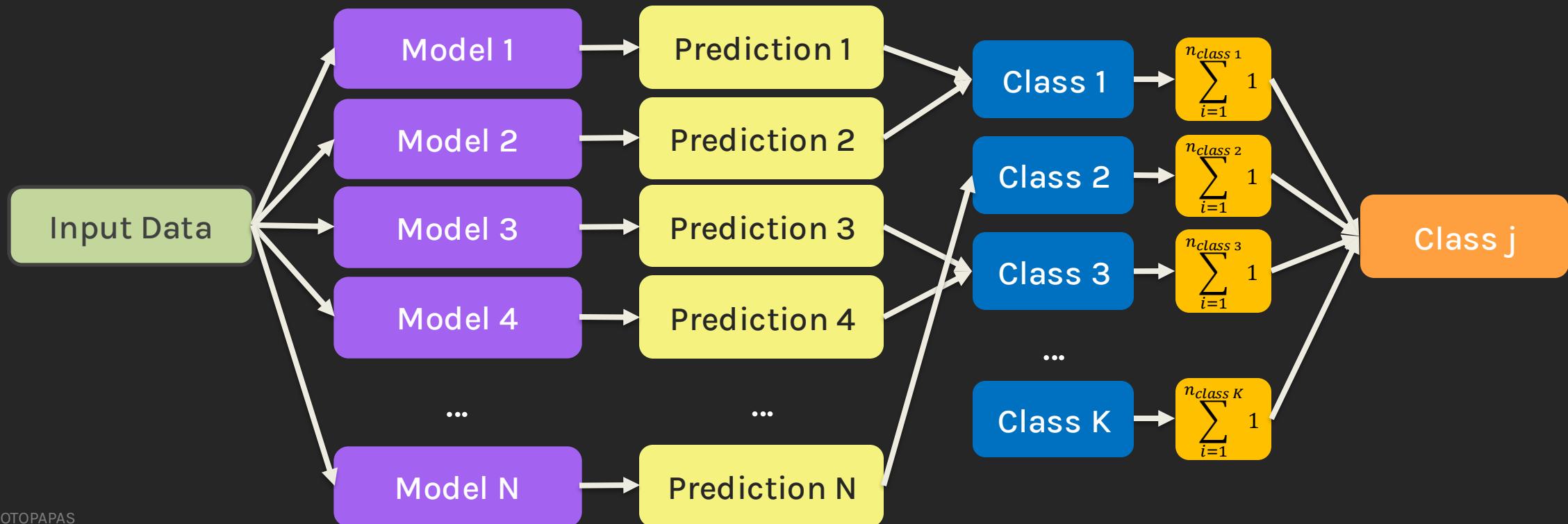
The predictions from base models are aggregated through:

- weighted averaging for regression tasks
- weighted voting for classification tasks

Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

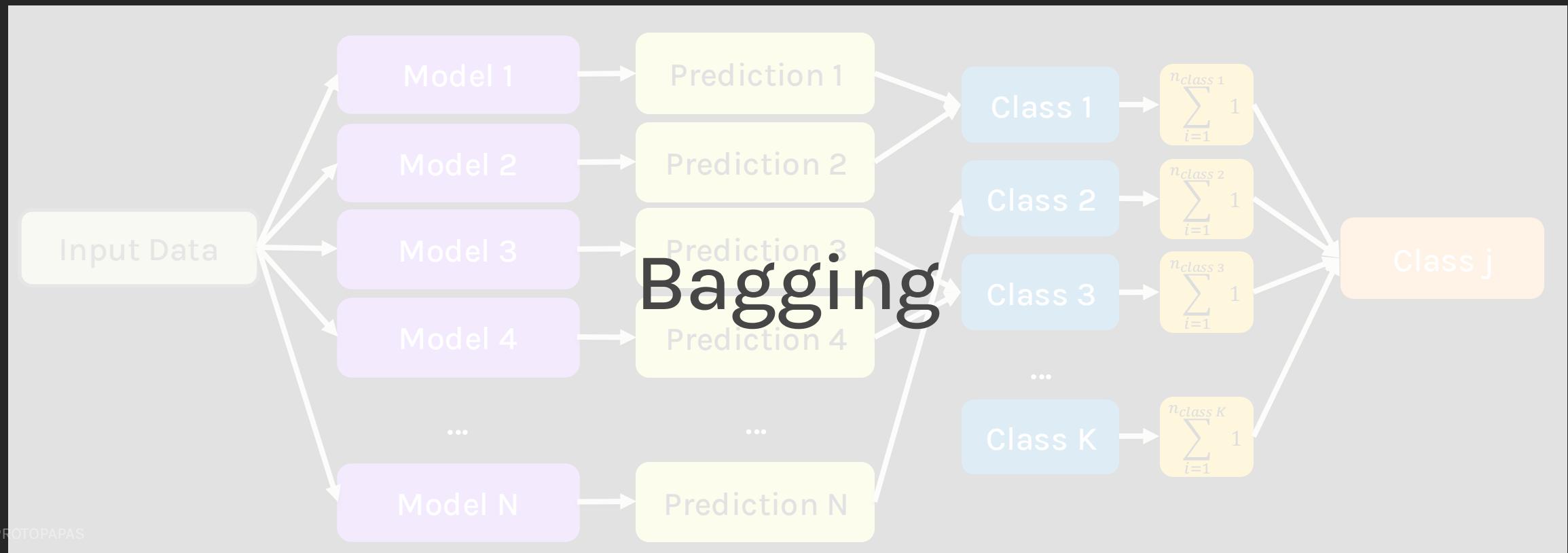
- Majority Voting (classification)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

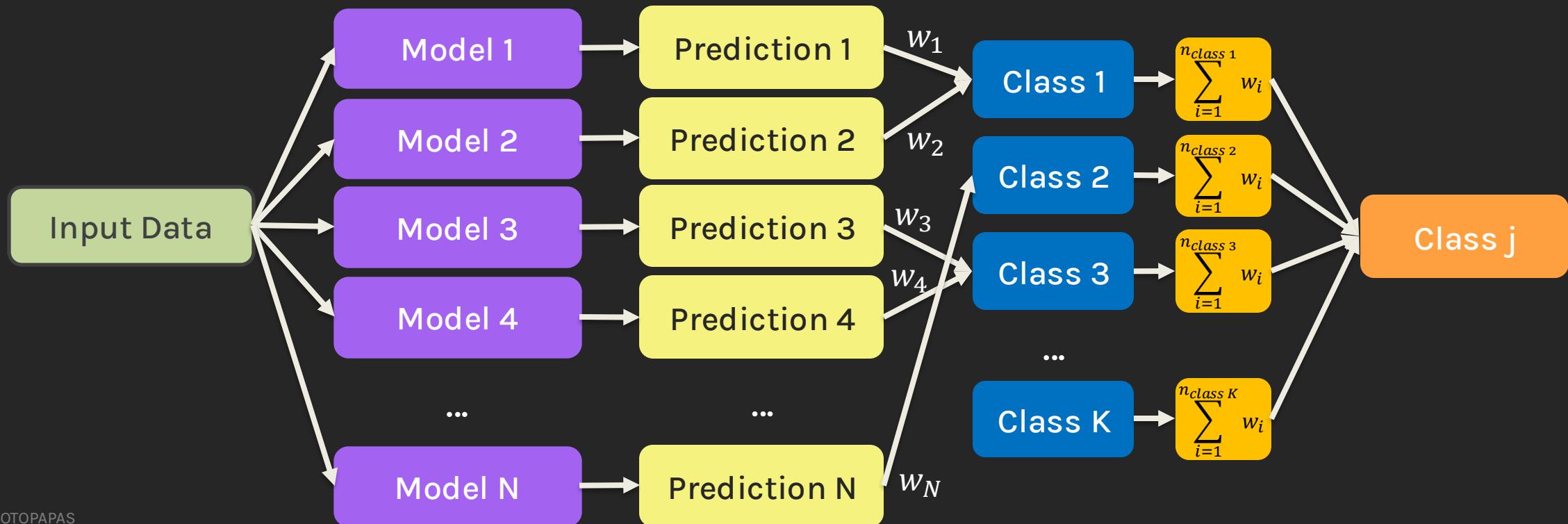
- Majority Voting (classification)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

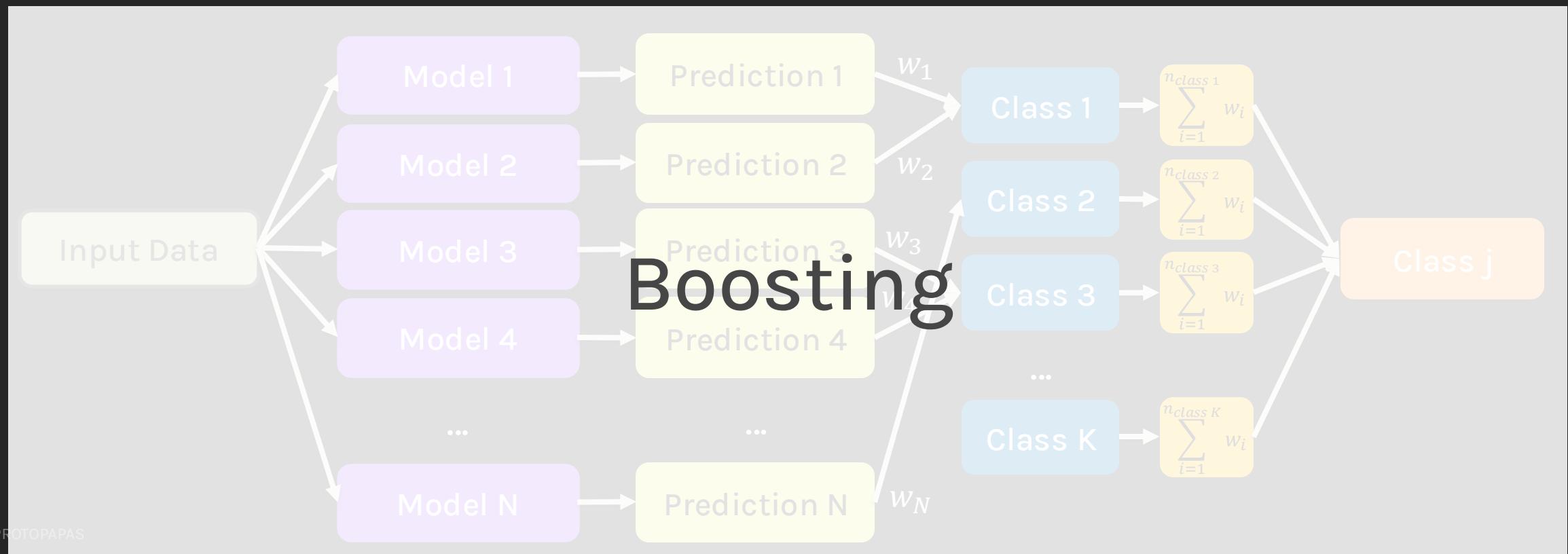
- Weighted Voting (classification)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

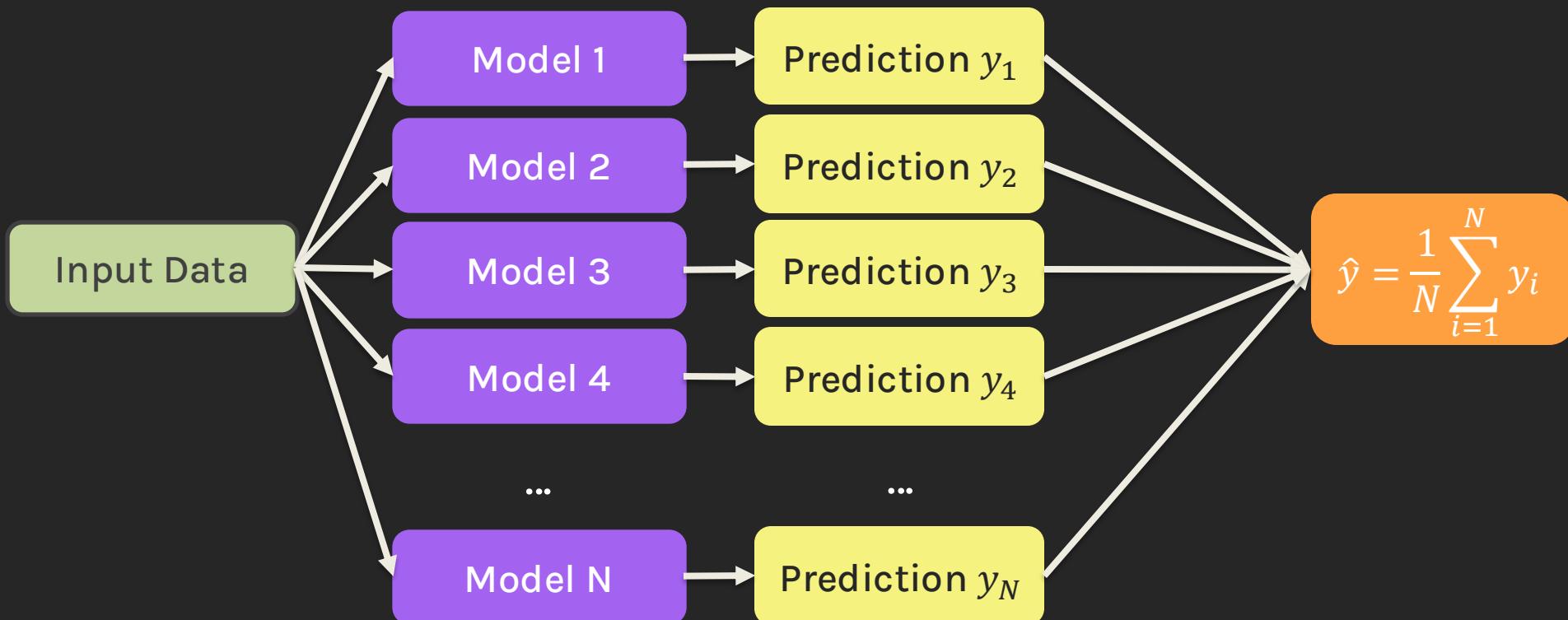
- Weighted Voting (classification)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

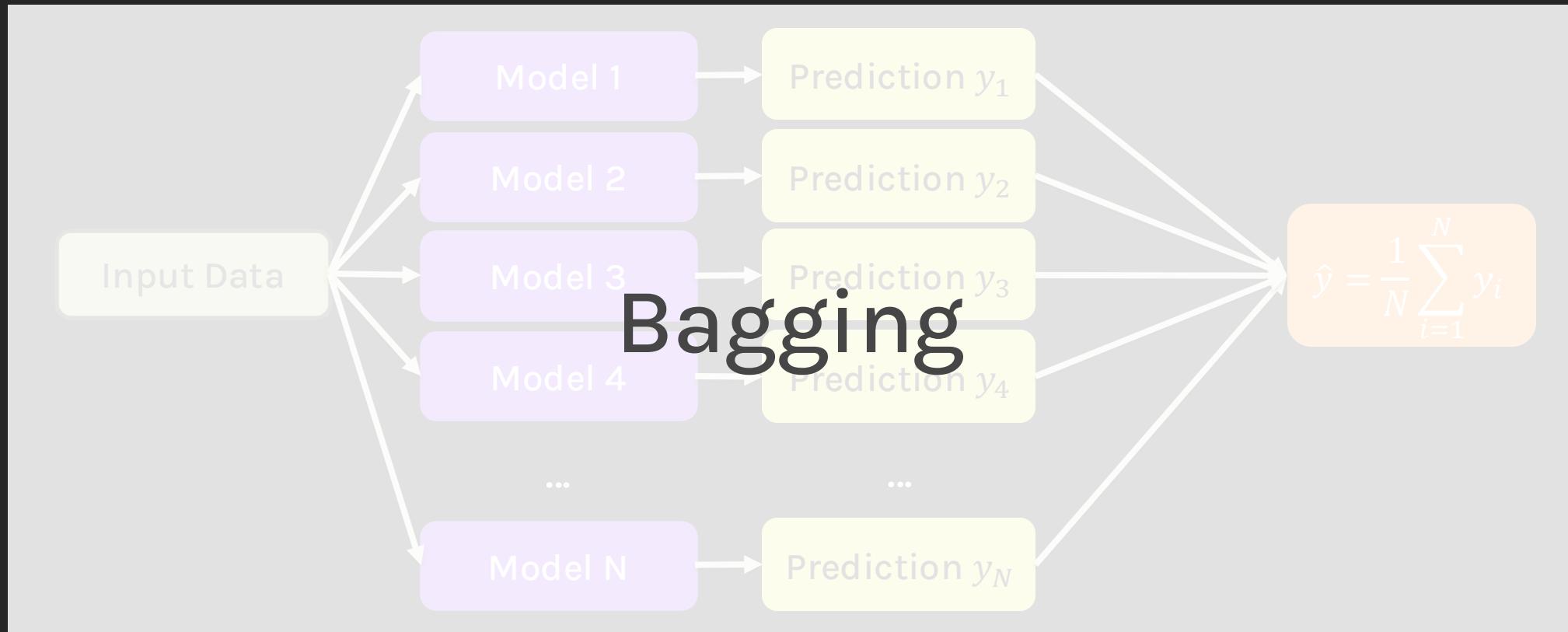
- Simple Averaging (regression)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

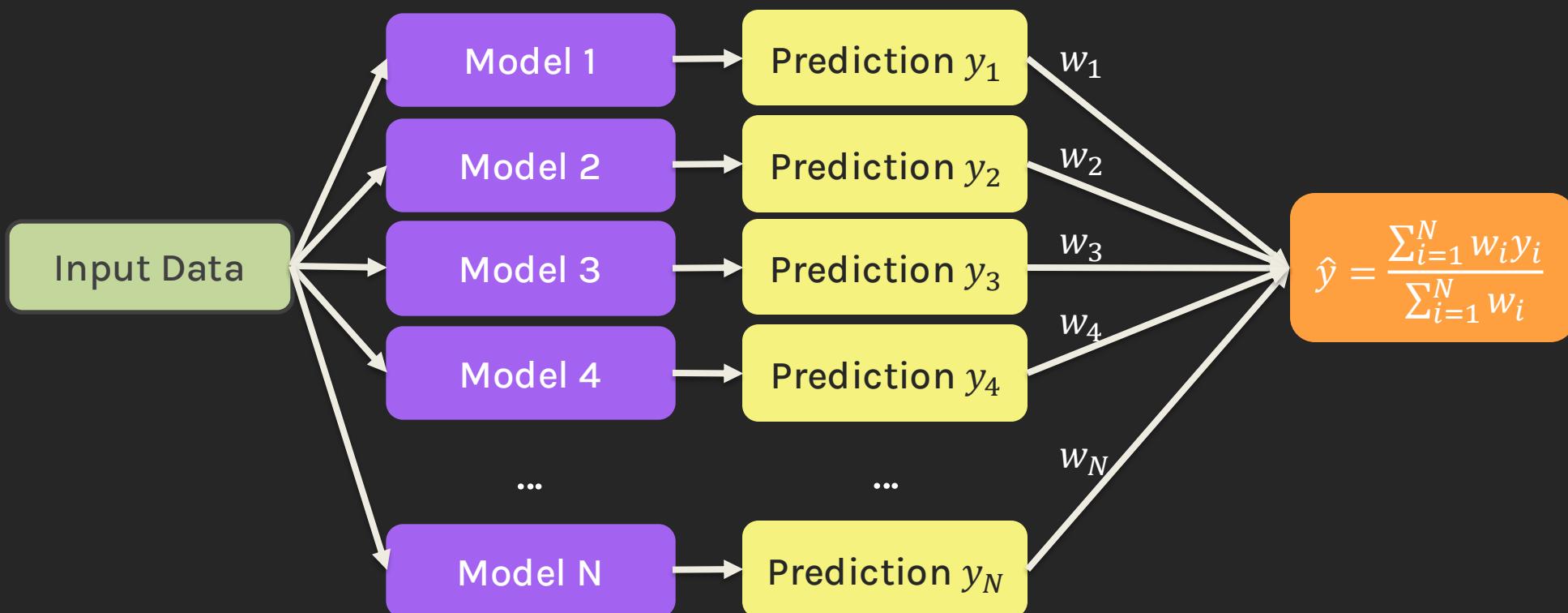
- Simple Averaging (regression)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

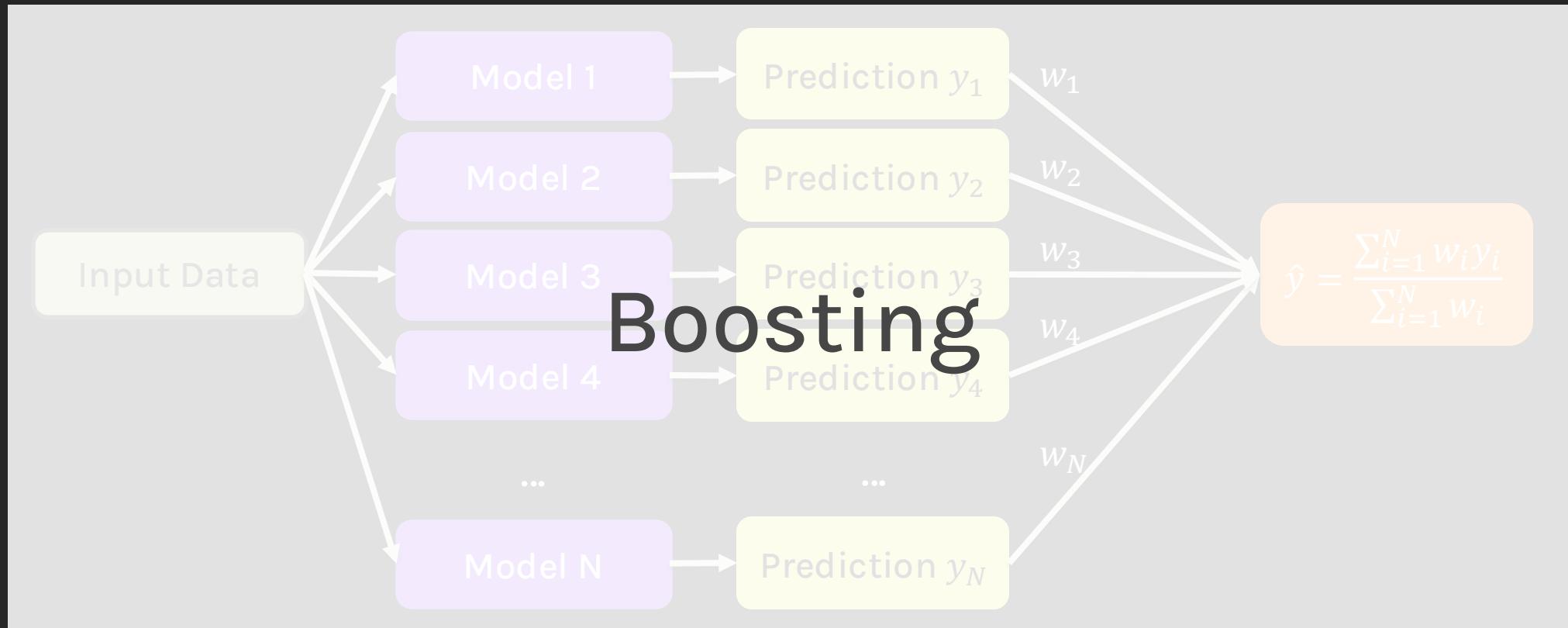
- Weighted Averaging (regression)



Recap: Ensemble Methods

So far, to aggregate the predictions of base models, we use voting and averaging, including

- Weighted Averaging (regression)



Can we aggregate outputs in a different way?

YES!

Can we use different models as base learners?

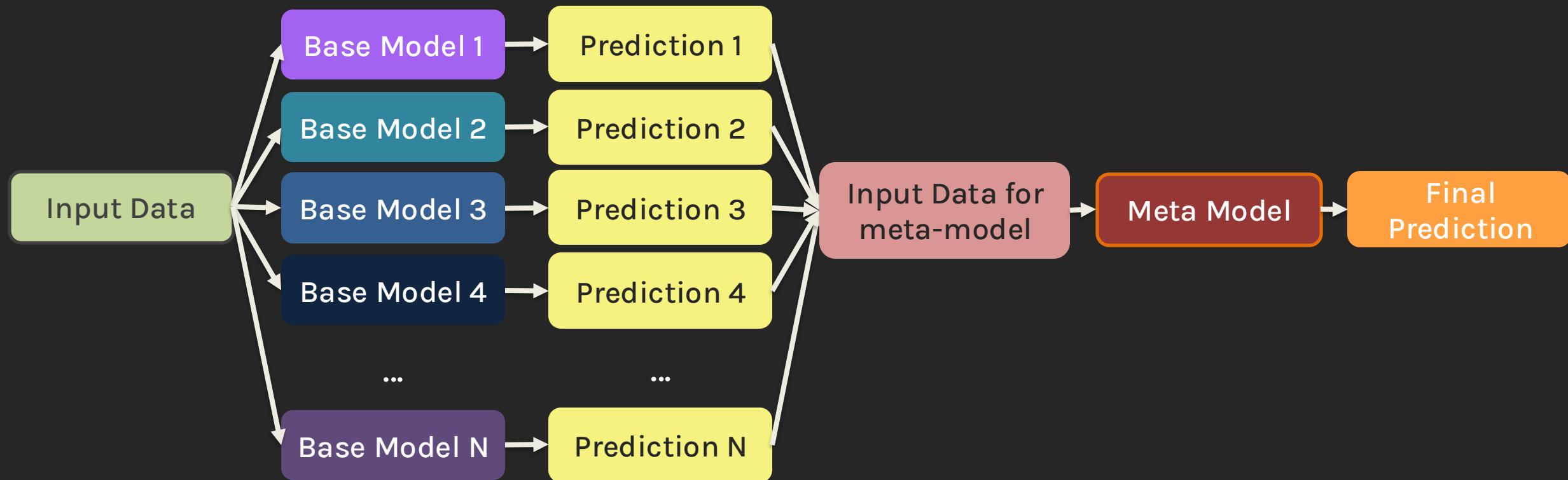
YES!

Outline

- Ensemble Methods
- **Blending**
- Stacking
- Mixture of Experts

Blending

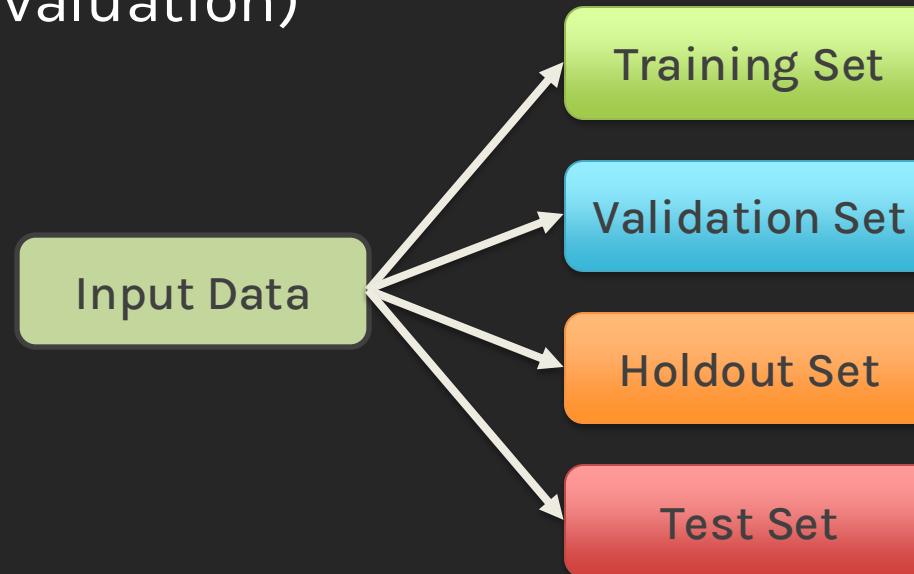
Blending trains **heterogeneous learners** (i.e., different models) on the dataset in parallel, and it further trains a **meta-model** on the base models' outputs for making predictions.



Blending

Step 1: Split the dataset into

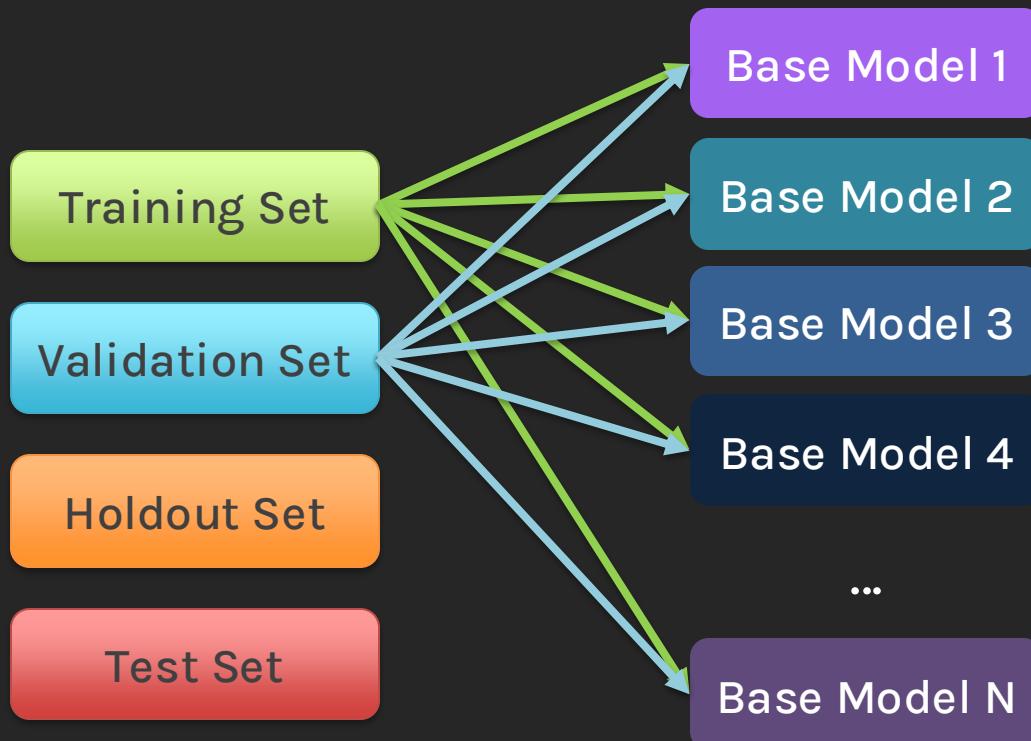
- Training set (to train base models)
- Validation set (to validate base models and generate predictions for training meta model)
- Holdout set (to validate meta model)
- Test set (for evaluation)



Blending

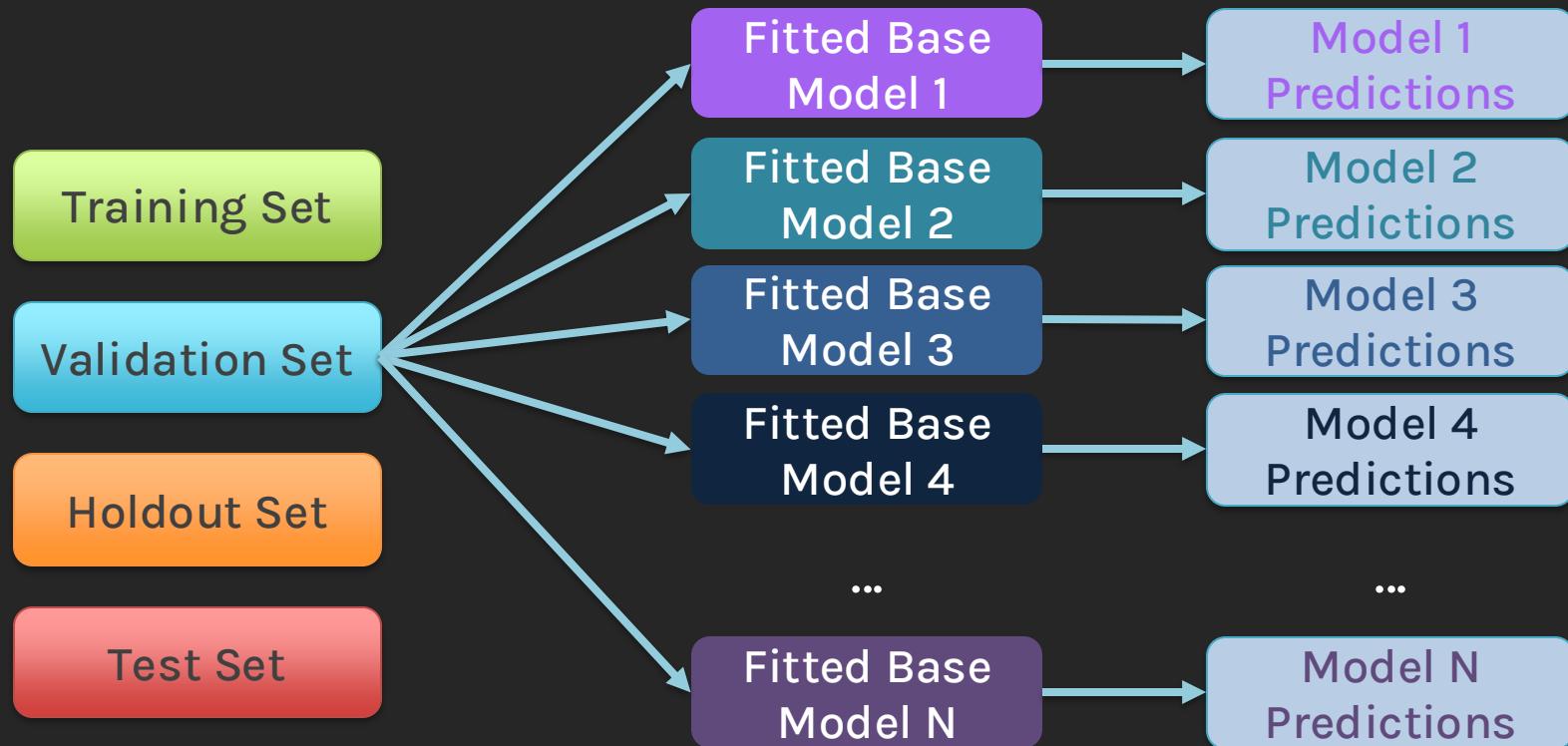
Step 2:

- Fit base models on **training set** and
- Validate with **validation set**



Blending

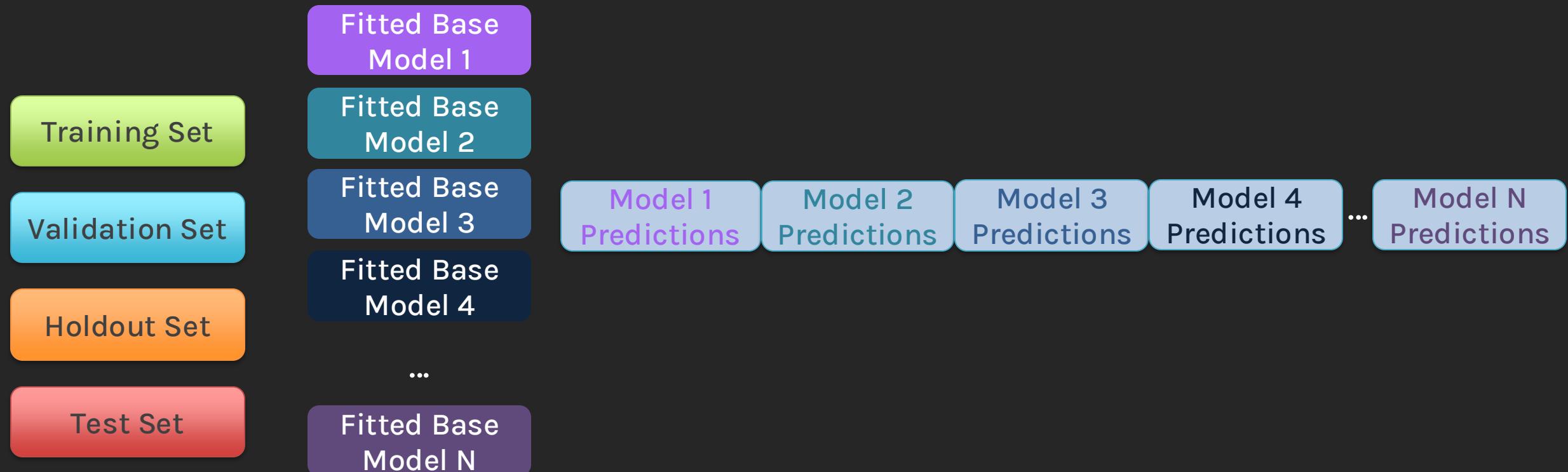
Step 3: Generate base model predictions on validation set and holdout set.



Blending

Step 3: Generate base model predictions on validation set and holdout set.

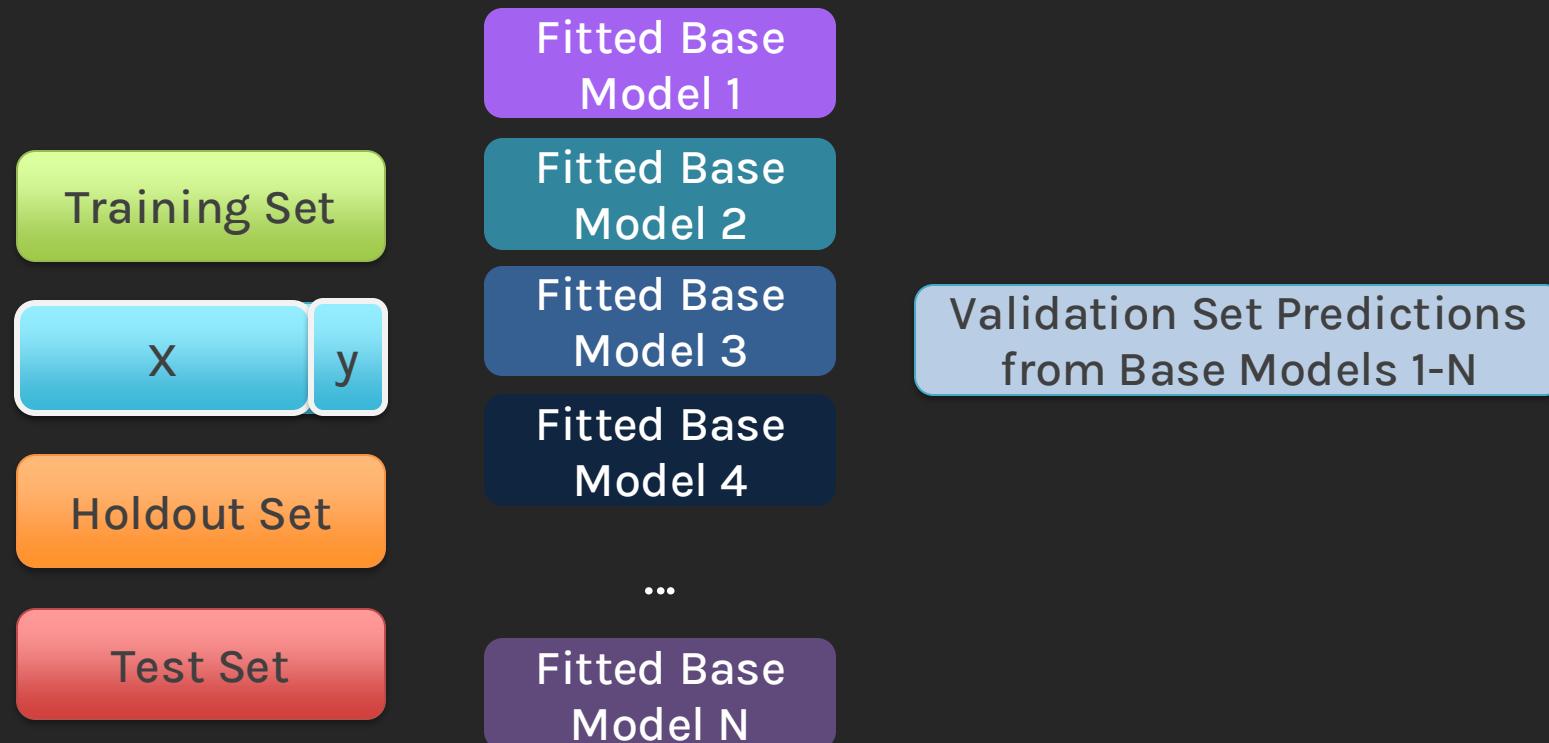
Step 4: Concatenate predictions and the original data into a new dataset.



Blending

Step 3: Generate base model predictions on validation set and holdout set.

Step 4: Concatenate predictions and the original data into a new dataset.

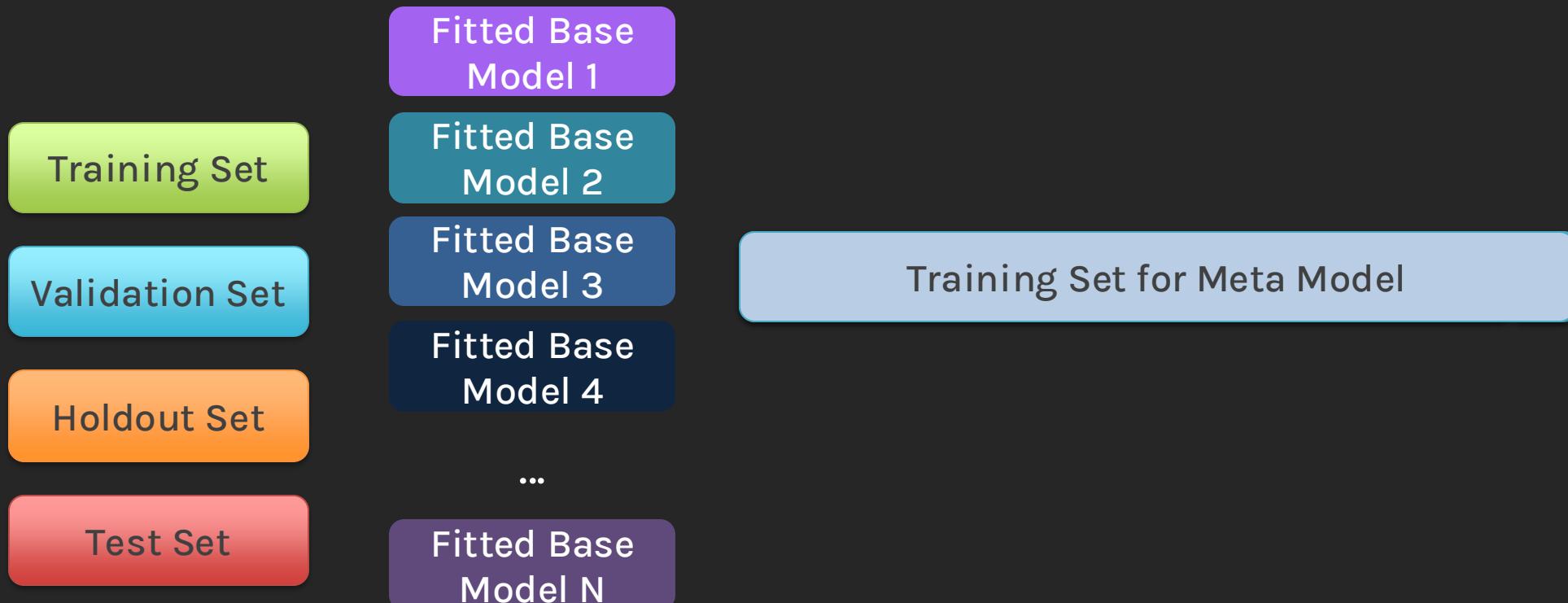


Blending

Step 3: Generate base model predictions on validation set and holdout set.

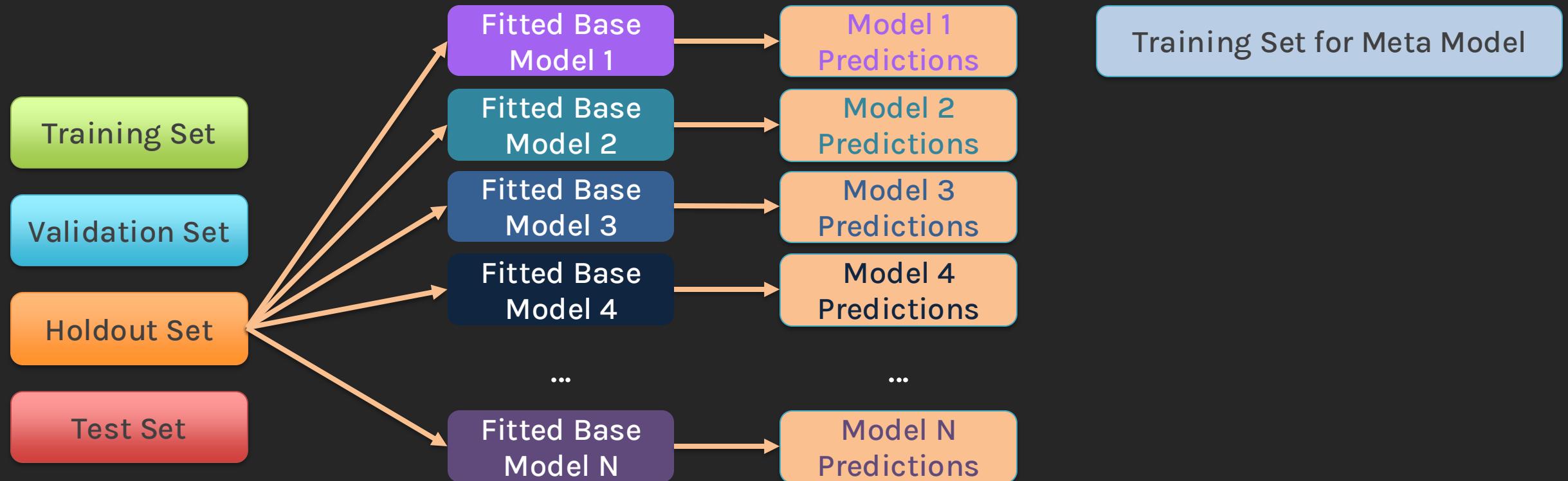
Step 4: Concatenate predictions and the original data into a new dataset.

Step 5: Combined features from validation set will be used for training meta model.



Blending

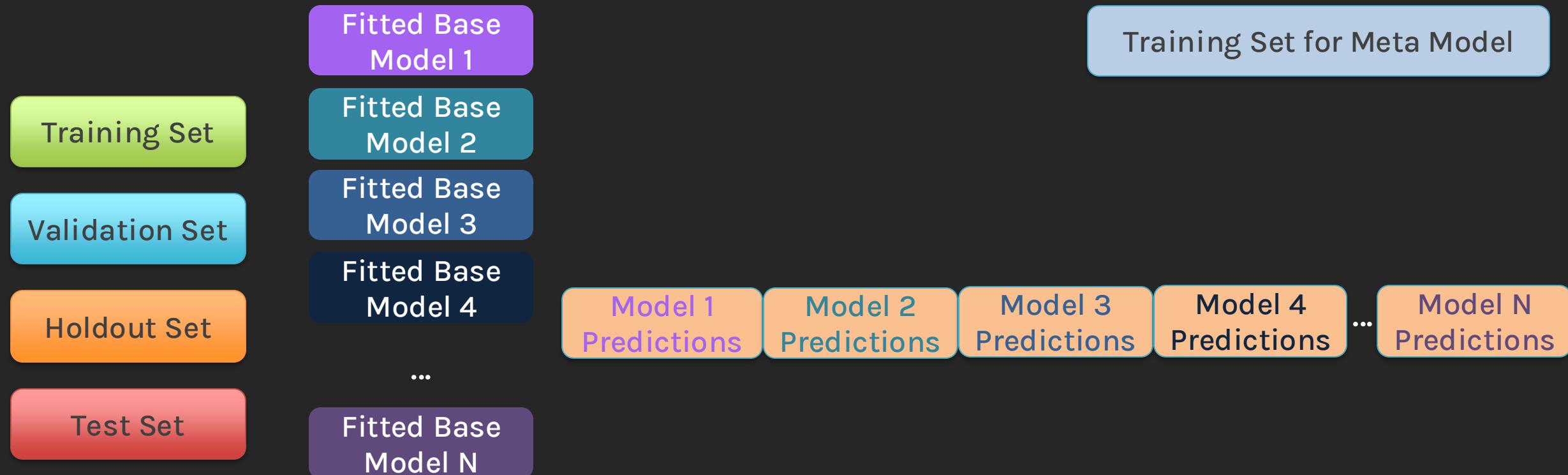
Step 3: Generate base model predictions on validation set and holdout set.



Blending

Step 3: Generate base model predictions on validation set and holdout set.

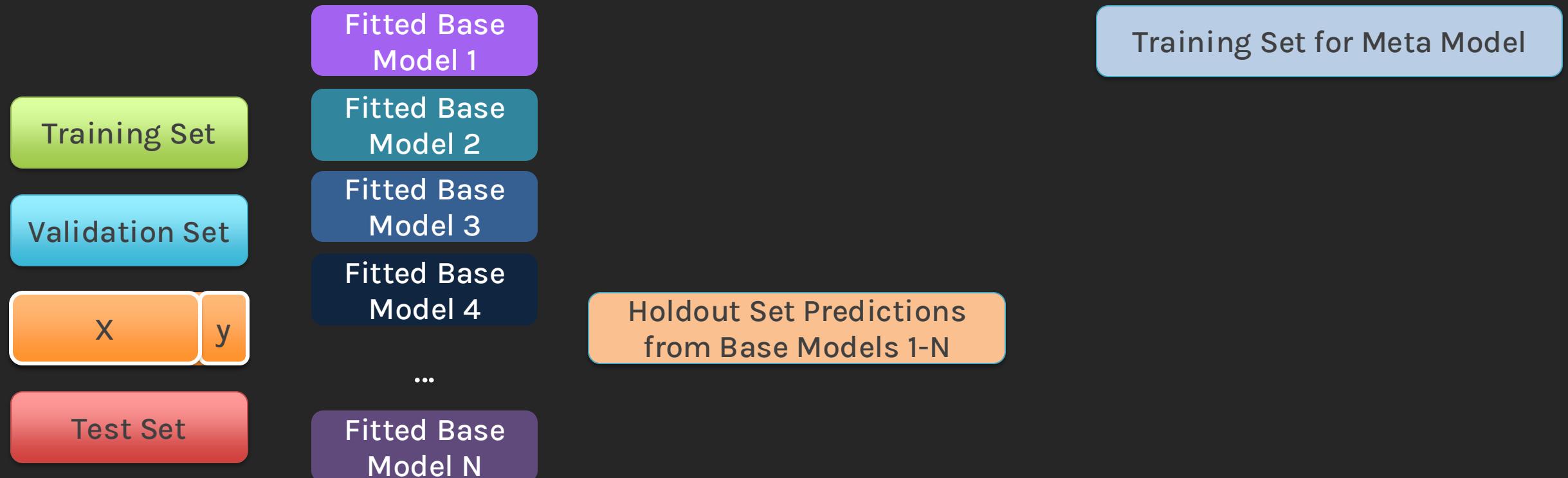
Step 4: Concatenate predictions into a new dataset.



Blending

Step 3: Generate base model predictions on validation set and holdout set.

Step 4: Concatenate predictions into a new dataset.

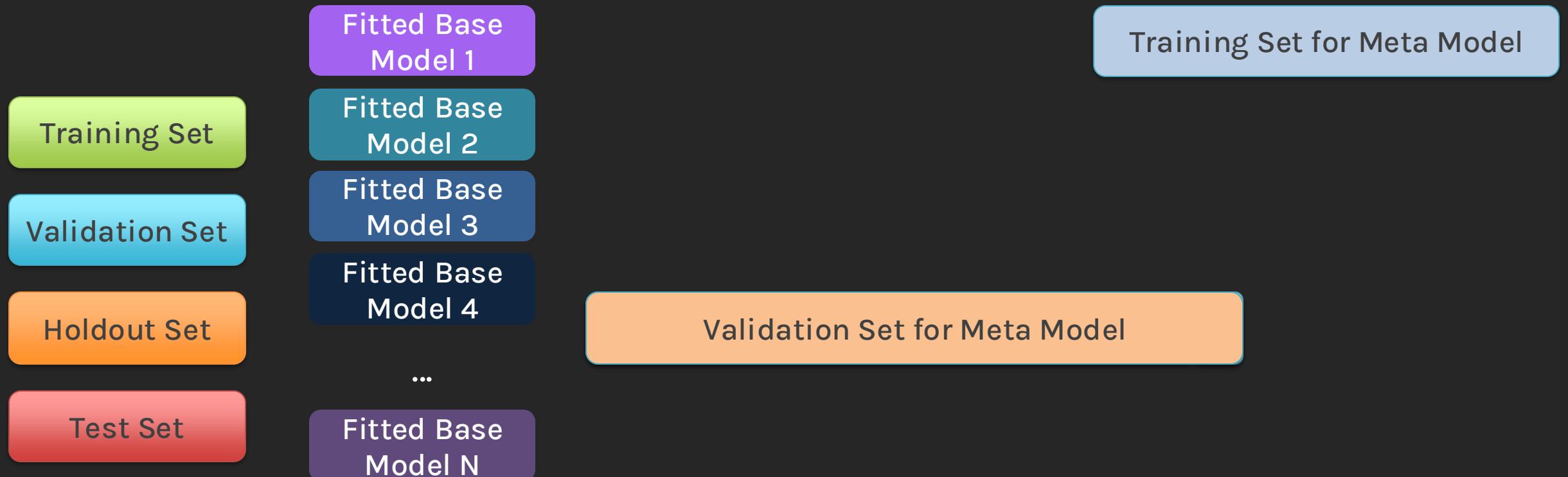


Blending

Step 3: Generate base model predictions on validation set and holdout set.

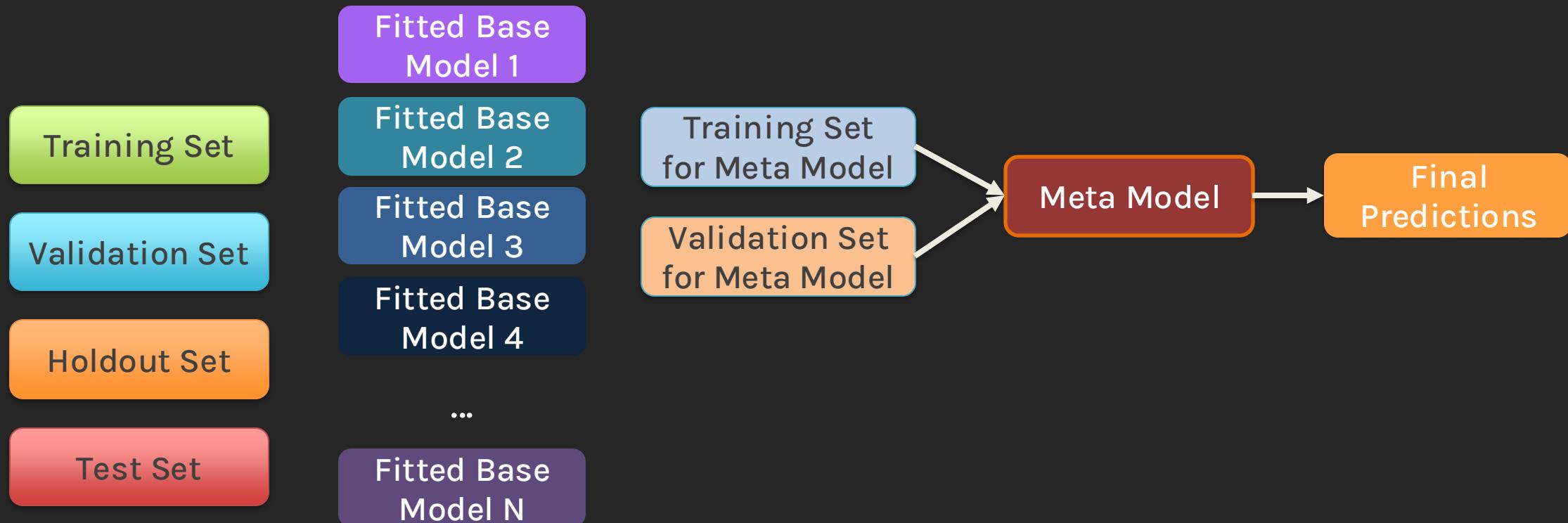
Step 4: Concatenate predictions into a new dataset.

Step 5: Combined features from holdout set will be used for validating meta model.



Blending

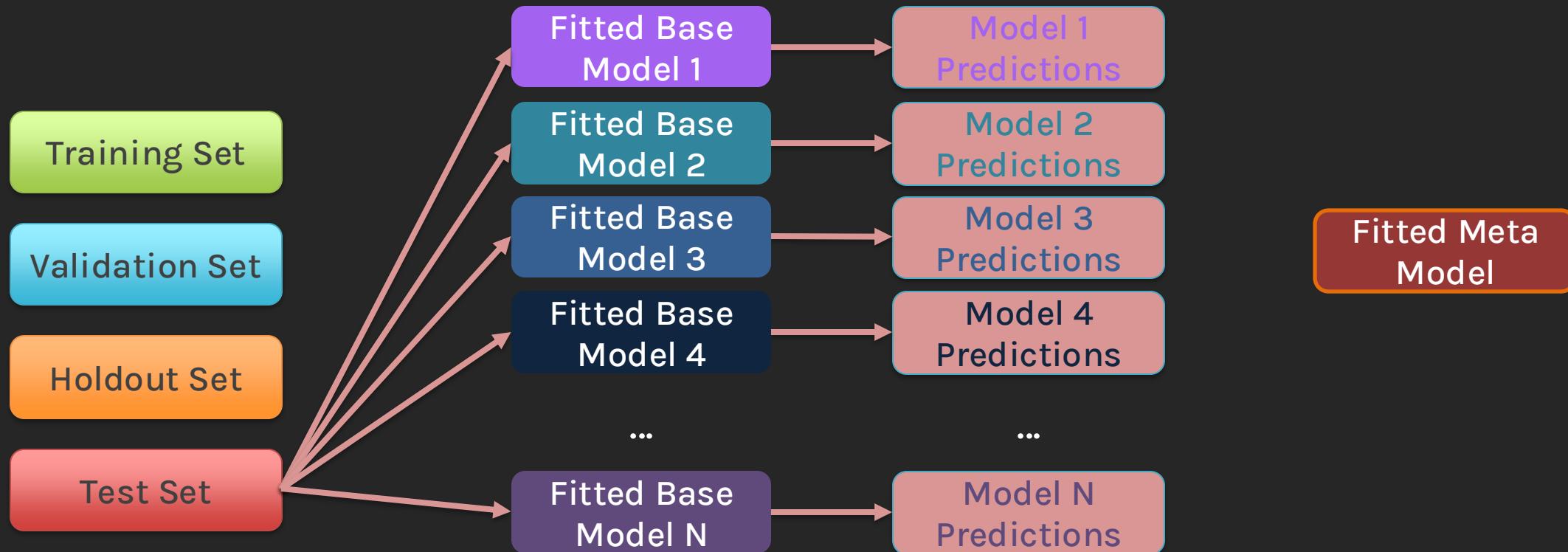
Step 6: Fit meta model with training and validation sets built with original data and base model predictions.



Blending

During inference (e.g., evaluation on test set):

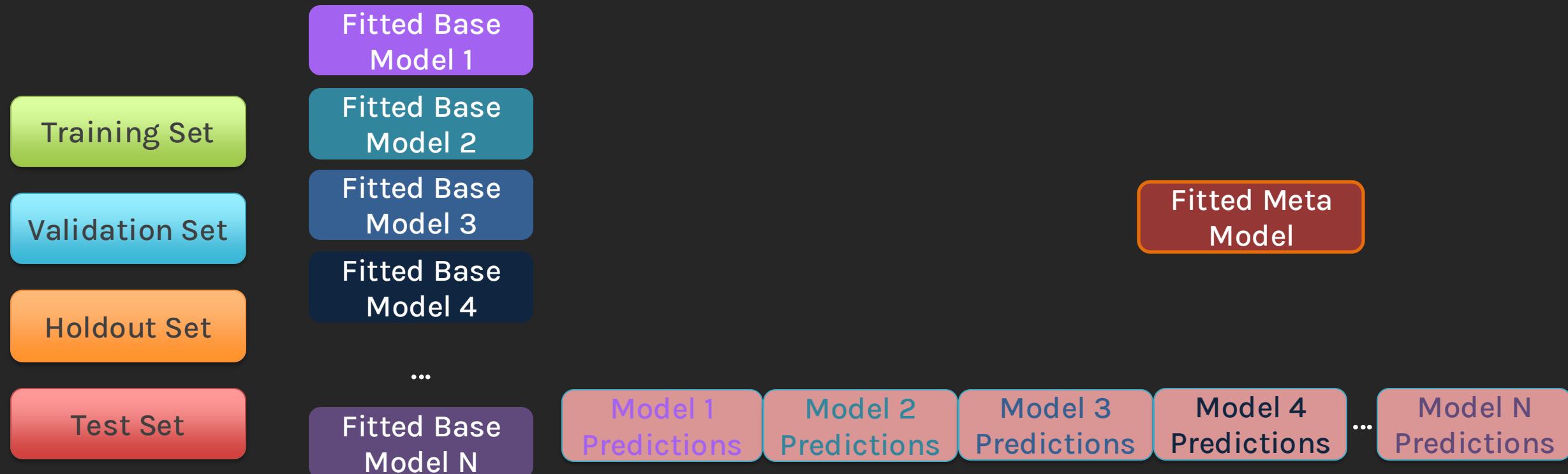
- Generate predictions from base models



Blending

During inference (e.g., evaluation on test set):

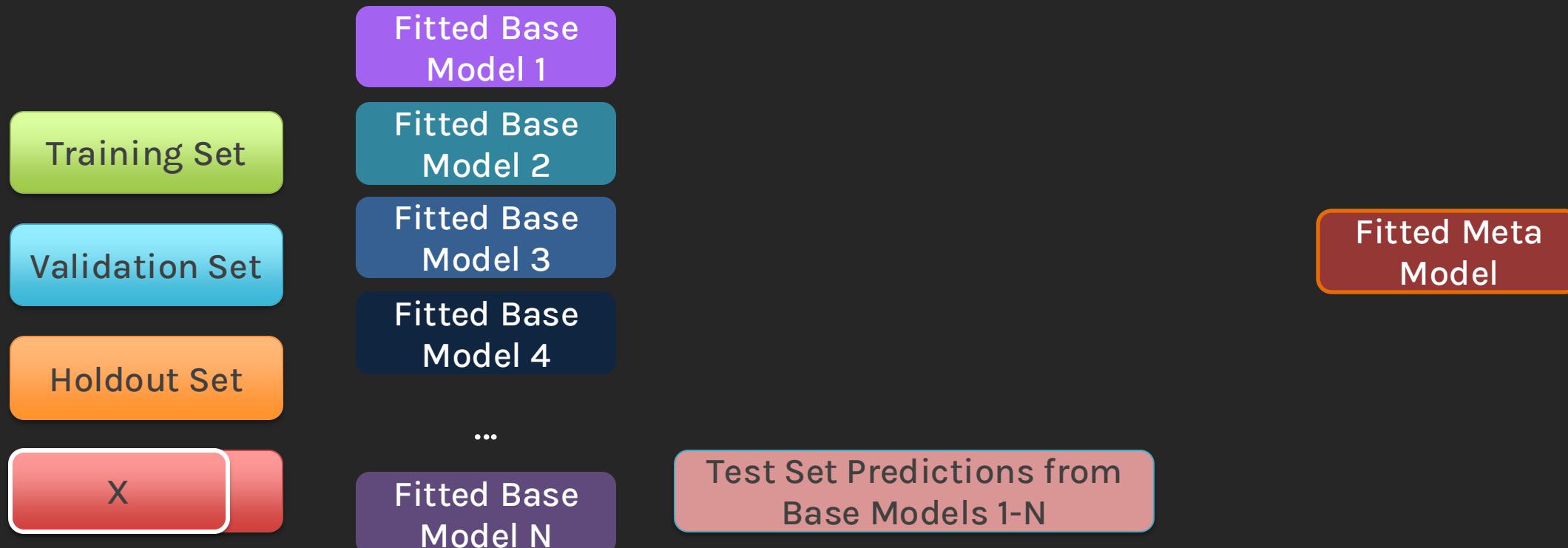
- Generate predictions from base models



Blending

During inference (e.g., evaluation on **test set**):

- Generate predictions from base models
- Combine model predictions and original data



Blending

During inference (e.g., evaluation on **test set**):

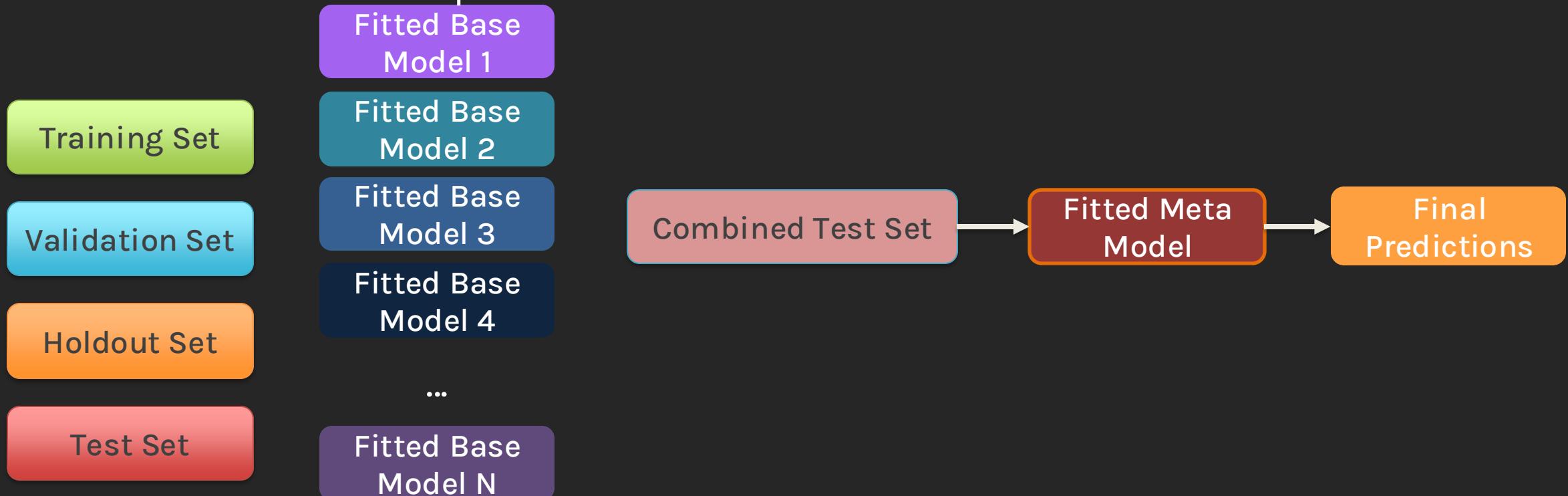
- Generate predictions from base models
- Combine model predictions and original data



Blending

During inference (e.g., evaluation on test set):

- Generate predictions from base models
- Combine model predictions and original data
- Generate model predictions



Blending

Compared with Bagging and Boosting, Blending provides flexibility

- Allows combining models of different types
- Meta model learns how to best combine diverse model outputs

Blending models can be more interpretable if meta-model is simple (e.g. linear regression), where contribution of each base model can be explicitly visible.

But we are dividing the dataset into more pieces for training blending models, and the amount of data for training meta model is only the size of validation set (usually 10%-25% of original data).

Question: Can we get more data for training meta model?

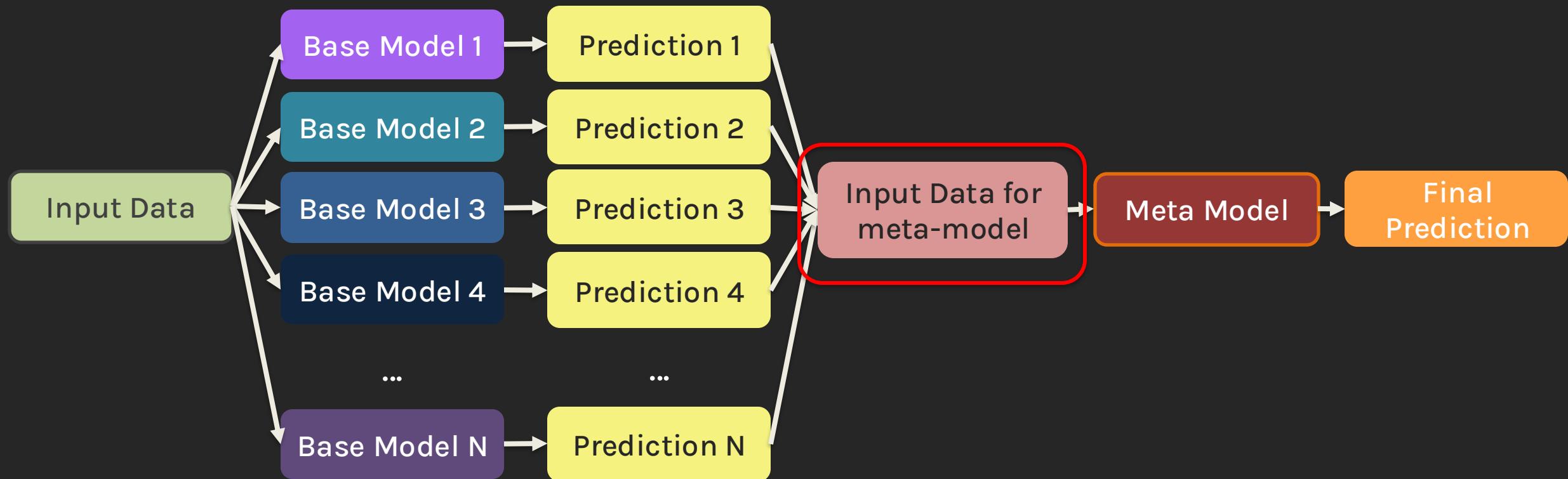
Outline

- Ensemble Methods
- Blending
- **Stacking**
- Mixture of Experts

Stacking

Similar to Blending, Stacking also trains heterogeneous learners on the dataset, and a meta model is fit with base models' predictions.

The difference is how we construct data for meta model.



Stacking

Like Blending, Stacking also trains heterogeneous learners on the dataset, and a meta model is fit with base models' predictions.

The difference is how we construct data for meta model.

“Stacked generalization works by deducing the biases of the generalizer(s) with respect to a provided learning set. This deduction proceeds by generalizing in a second space whose inputs are (for example) the guesses of the original generalizers when taught with part of the learning set and trying to guess the rest of it, and whose output is (for example) the correct guess.” Stack Generalization 1992 - D. Wolpert

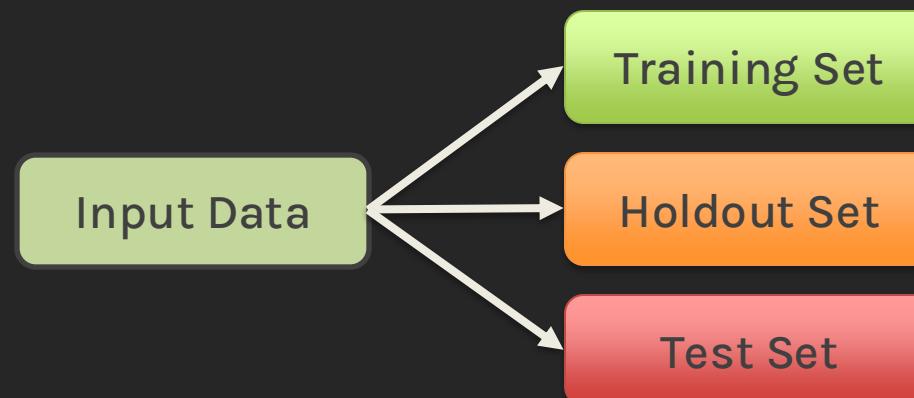
Stacking

Step 1: Similar to blending, we first split the dataset into

- Training set (to train base models)
- Holdout set (to validate meta model)
- Test set (for evaluation)

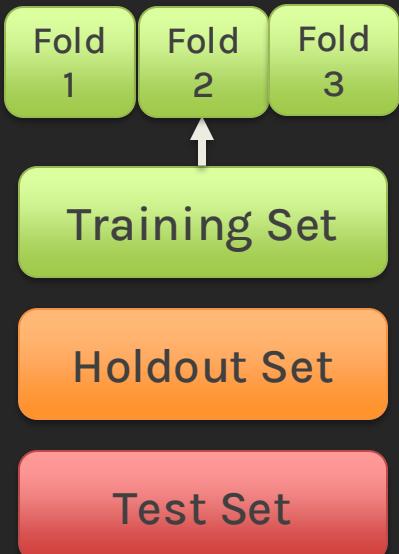
Note that we don't have validation set here anymore, because we will be doing

CROSS VALIDATION!



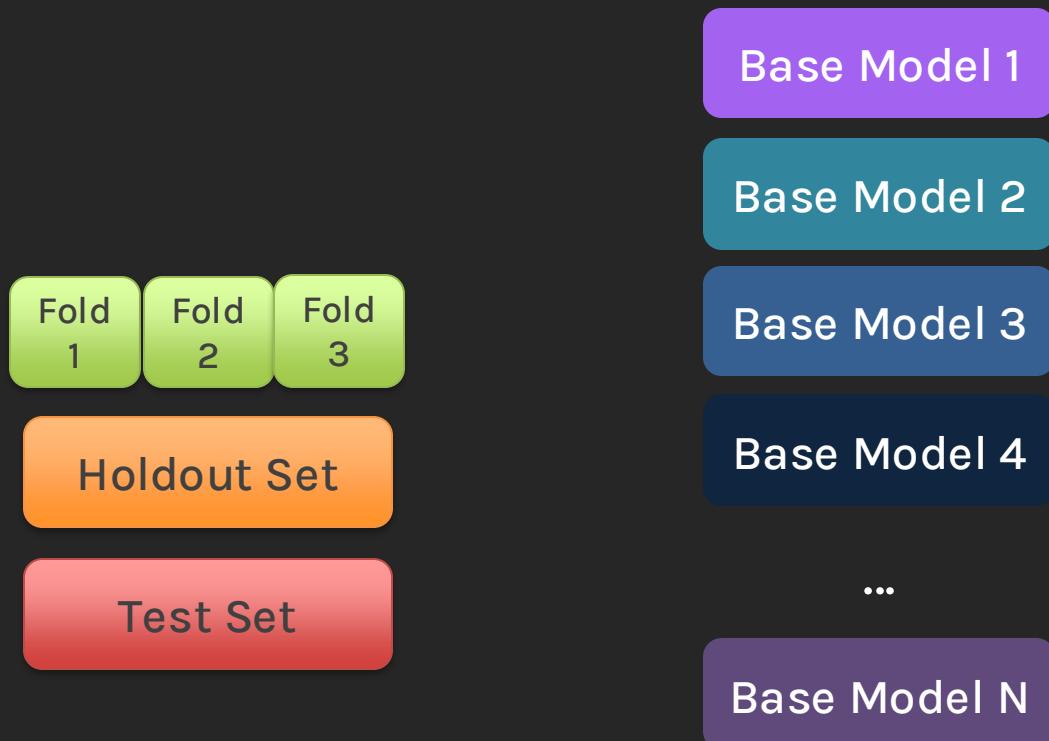
Stacking

Step 1: In cross validation, we need to split the training data into folds.
For simplicity here, let's assume we want $k = 3$ folds.



Stacking

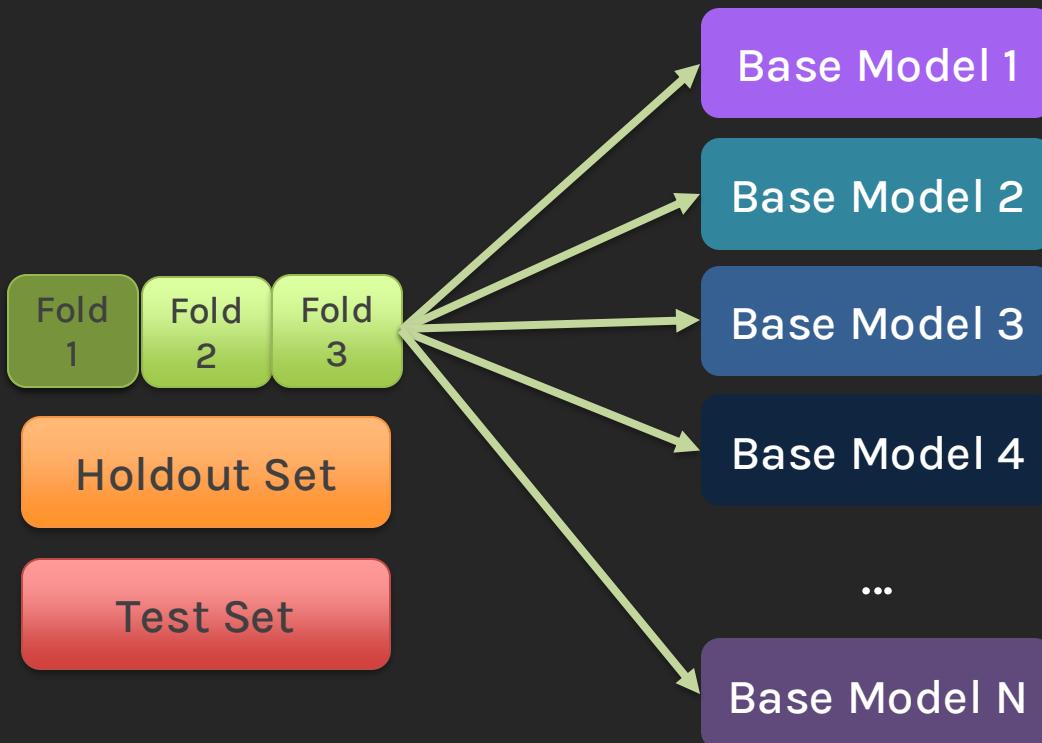
Step 2: Fit base models with cross validation. While we do validation with each fold, collect these predictions to build our training set for meta model. In other words, we concatenate the output of `cross_val_predict()` from each base model into a new dataset.



Stacking

Step 2:

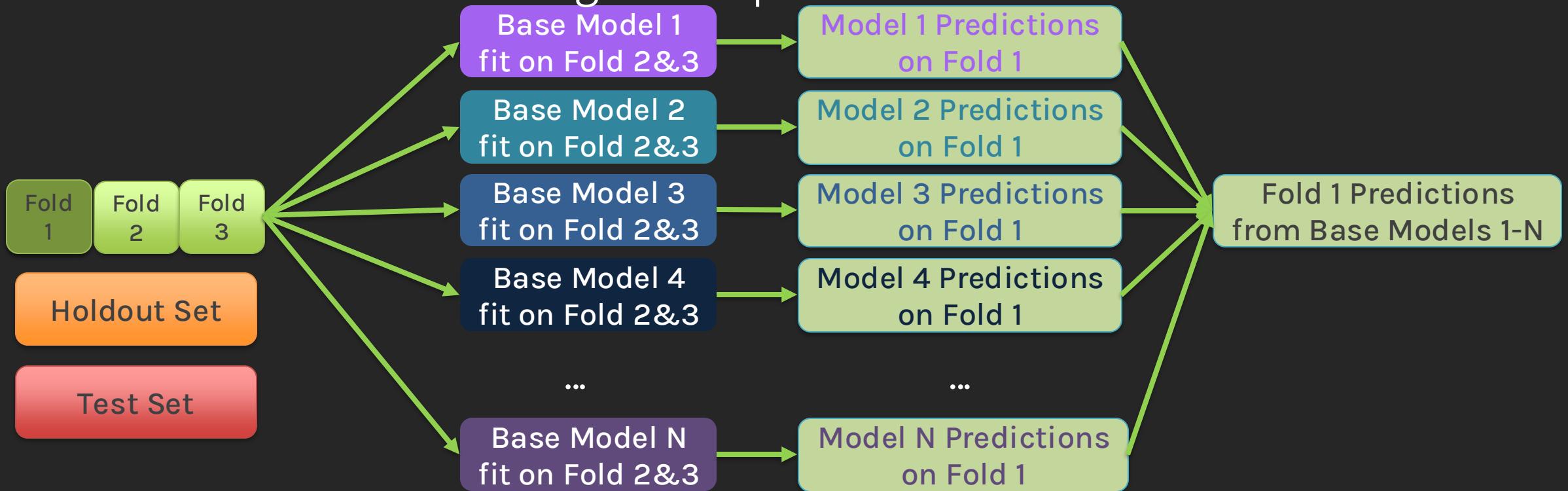
- We firstly take Fold 1 as our validation fold.
- Train on Fold 2 and 3.



Stacking

Step 2:

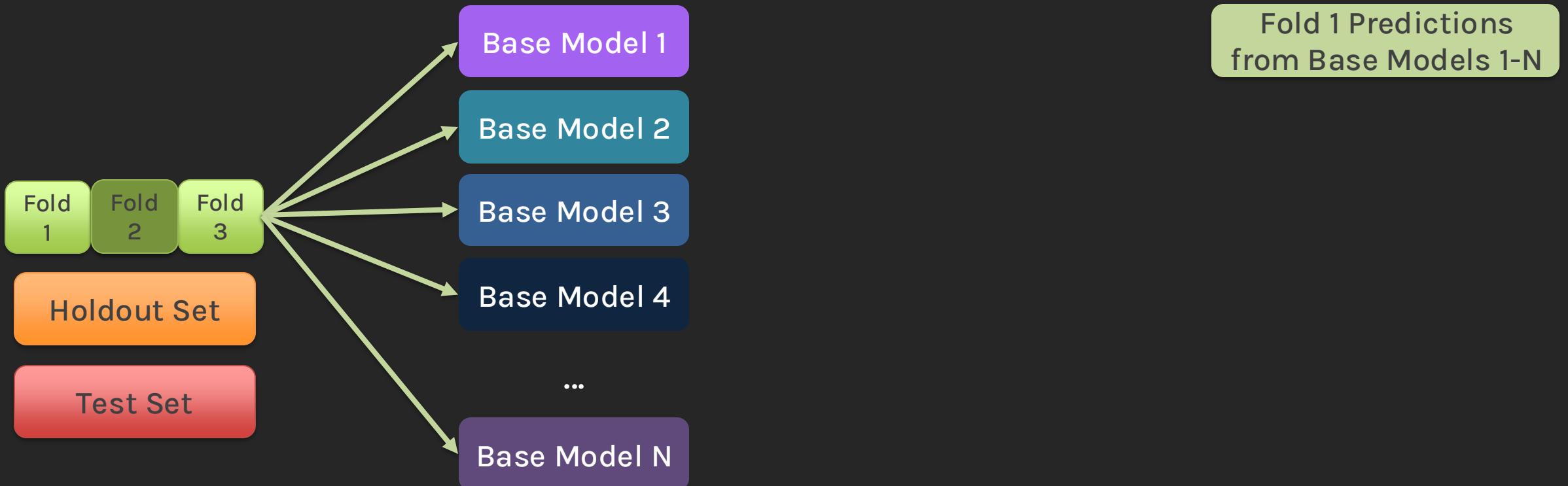
- We firstly take Fold 1 as our validation fold.
- Train on Fold 2 and 3.
- Validate on Fold 1 and generate predictions.



Stacking

Step 2:

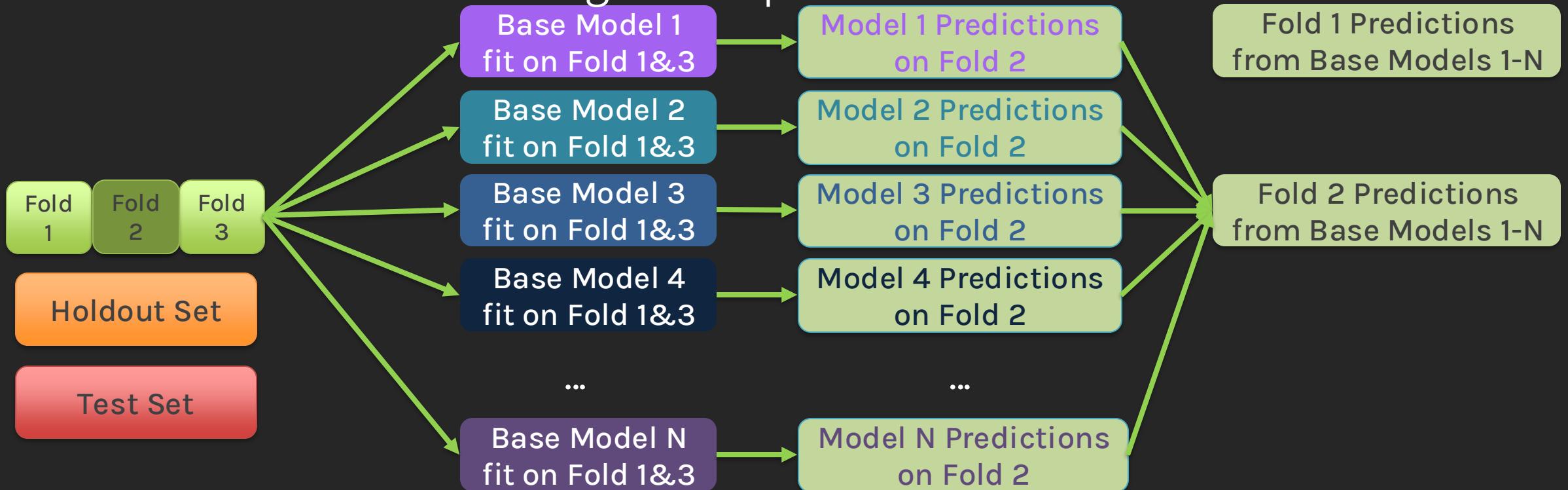
- Next, we take Fold 2 as our validation fold.
- Train on Fold 1 and 3.



Stacking

Step 2:

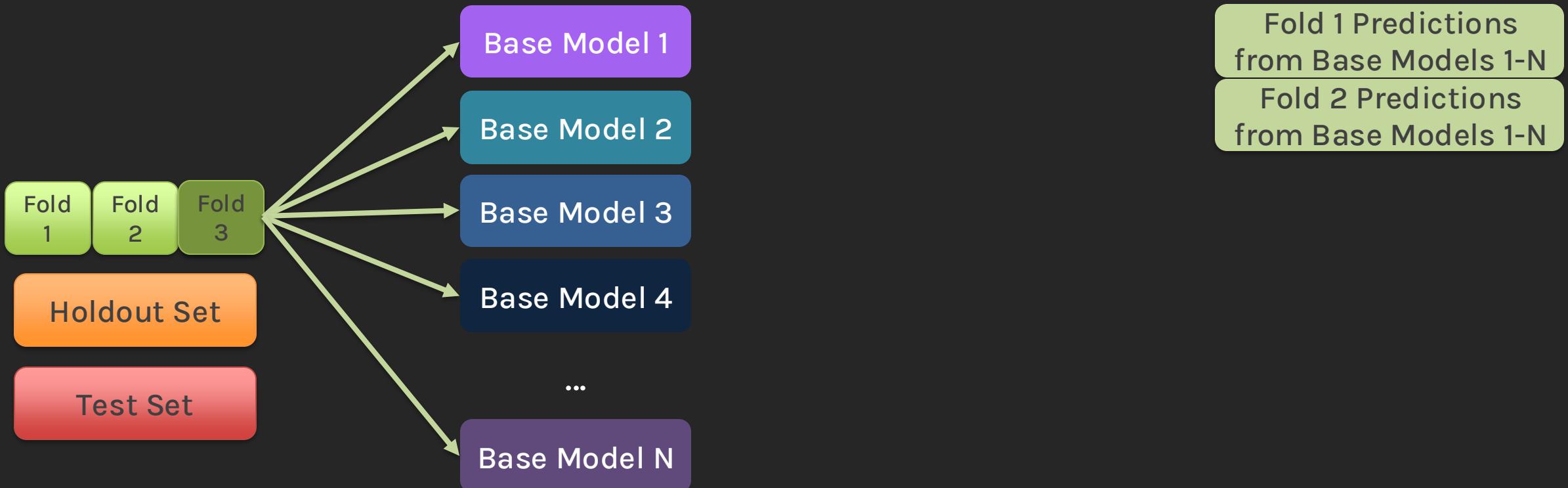
- Next, we take Fold 2 as our validation fold.
- Train on Fold 1 and 3.
- Validate on Fold 2 and generate predictions.



Stacking

Step 2:

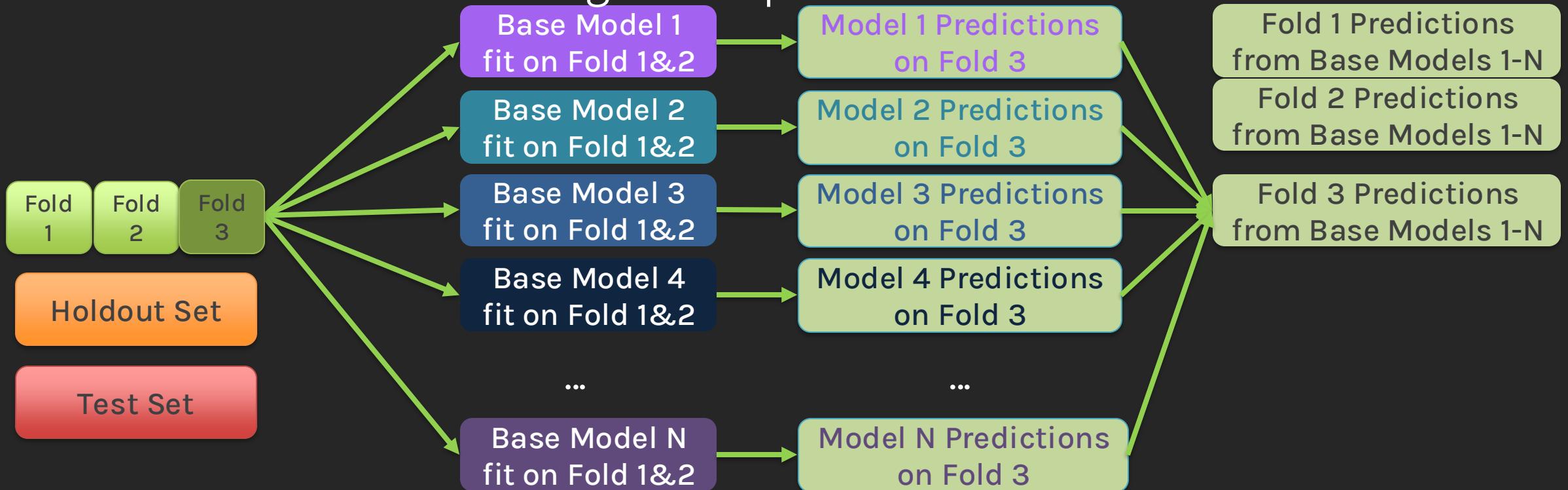
- Now we take Fold 3 as our validation fold.
- Train on Fold 1 and 2.



Stacking

Step 2:

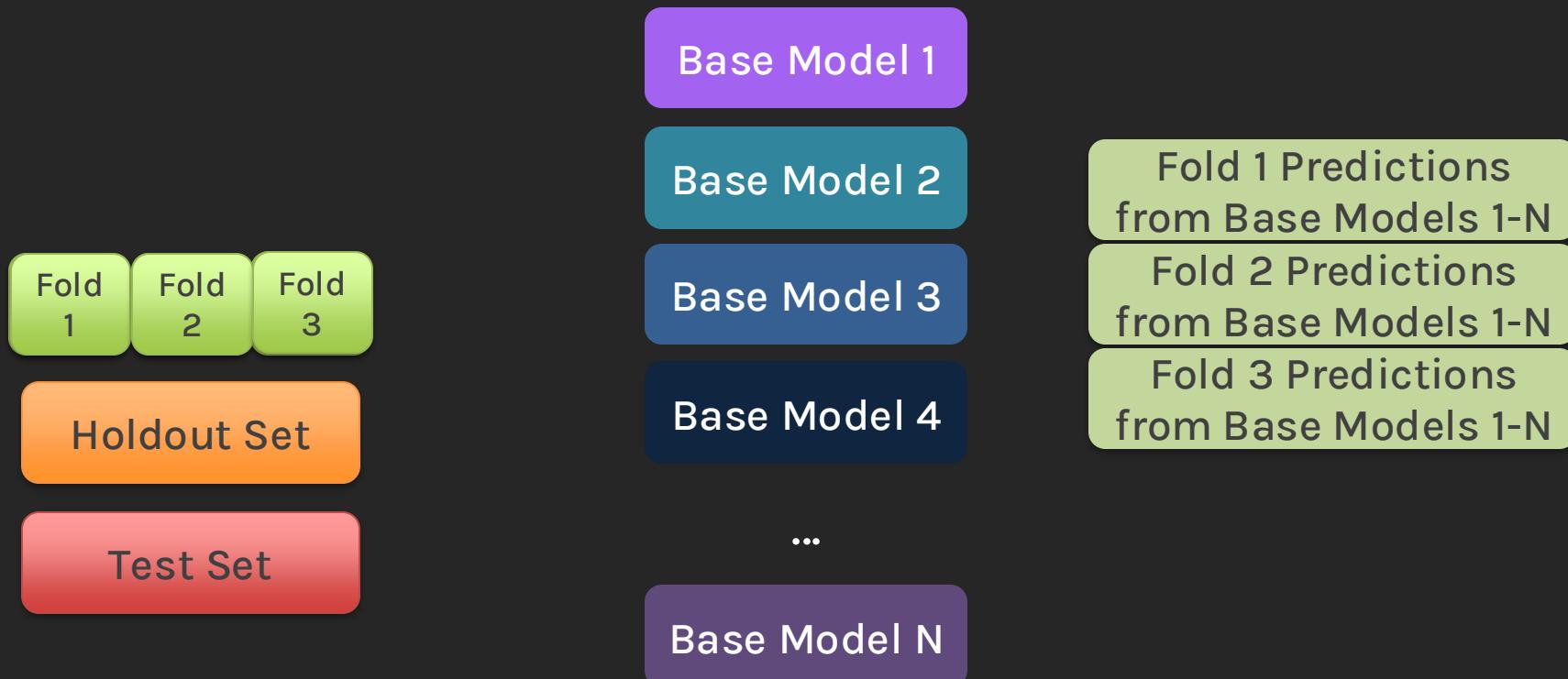
- Now we take Fold 3 as our validation fold.
- Train on Fold 1 and 2.
- Validate on Fold 3 and generate predictions.



Stacking

Step 2:

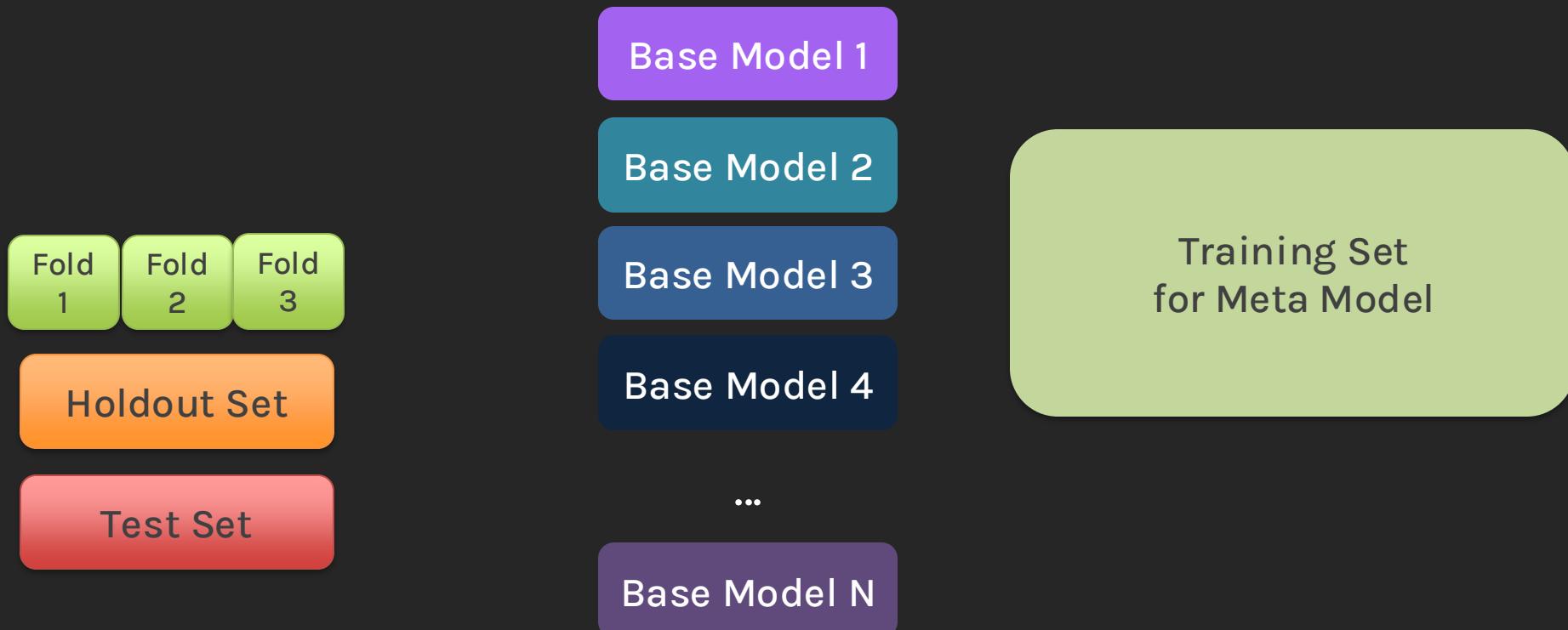
Combine the training folds with the base model predictions, and this is the training set for meta model. Note that different from blending, the size of training set is the same as the original training set!



Stacking

Step 2:

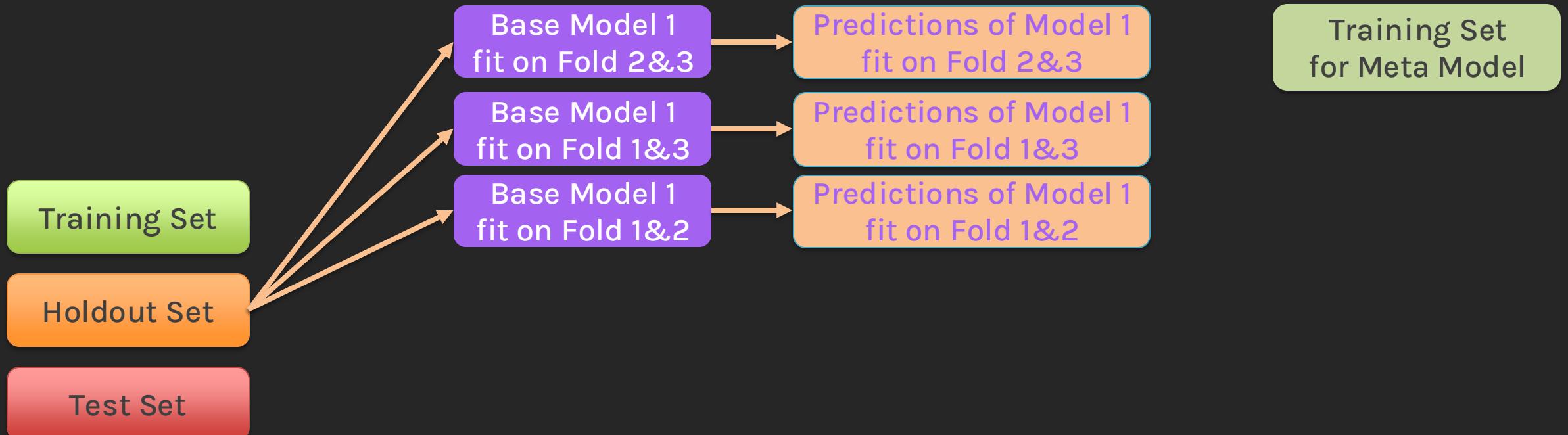
Combine the training folds with the base model predictions, and this is the training set for meta model. Note that different from blending, the size of training set is the same as the original training set!



Stacking

Step 2:

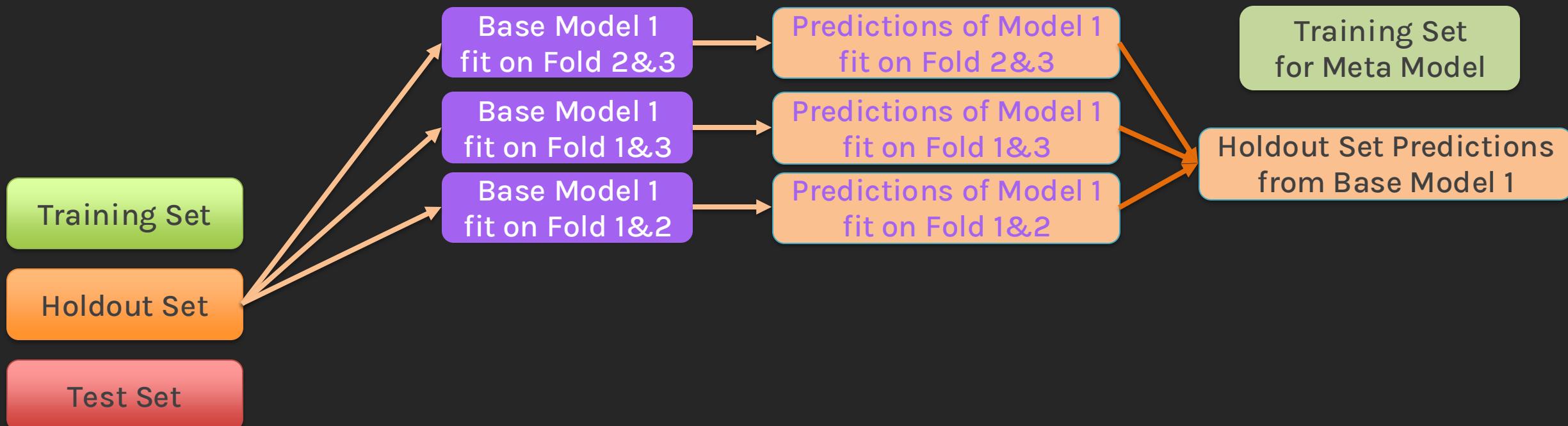
Meanwhile, in each fold round of cross validation, we want to make predictions on **holdout set**.



Stacking

Step 3:

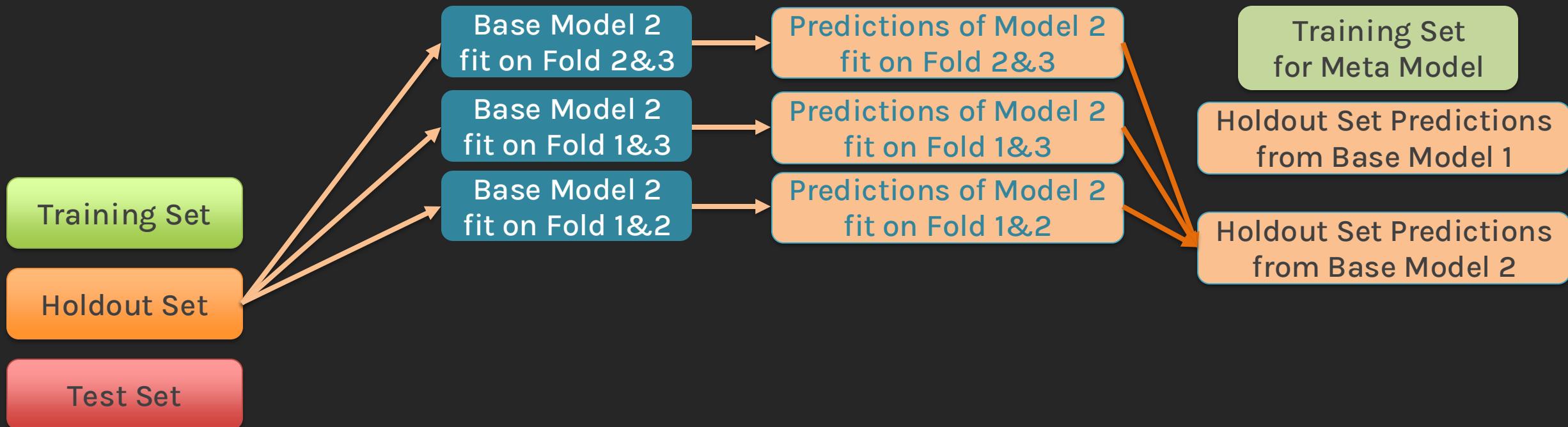
Meanwhile, in each fold round of cross validation, we want to make predictions on **holdout set**. Then we take the **average** across the folds (i.e., we have one group of holdout set predictions per base model).



Stacking

Step 3:

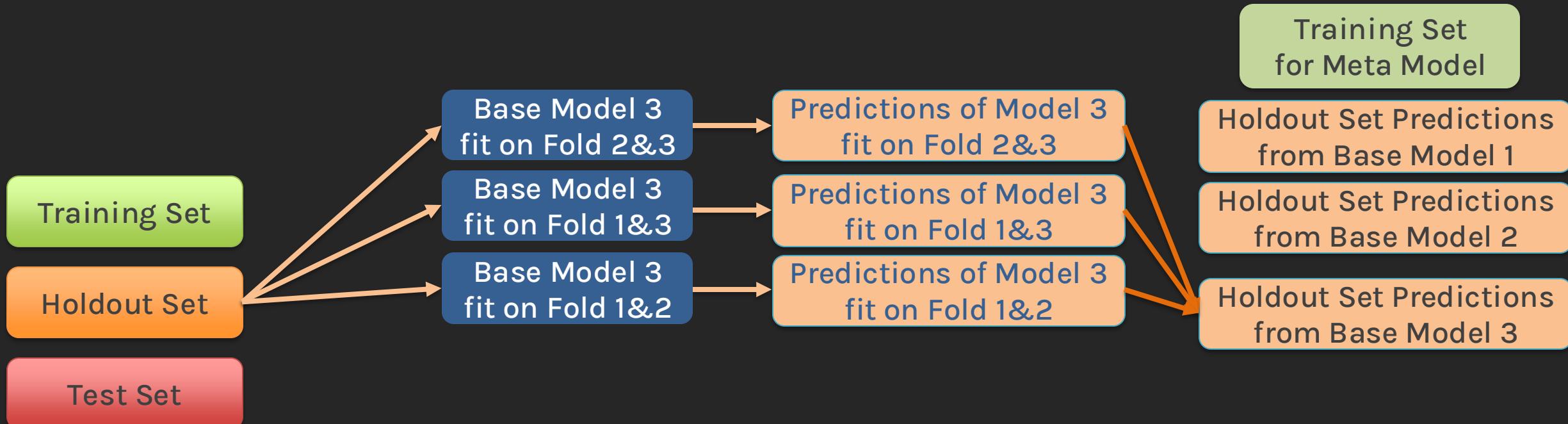
We repeat for all other base models. Generate holdout set predictions and take the average.



Stacking

Step 3:

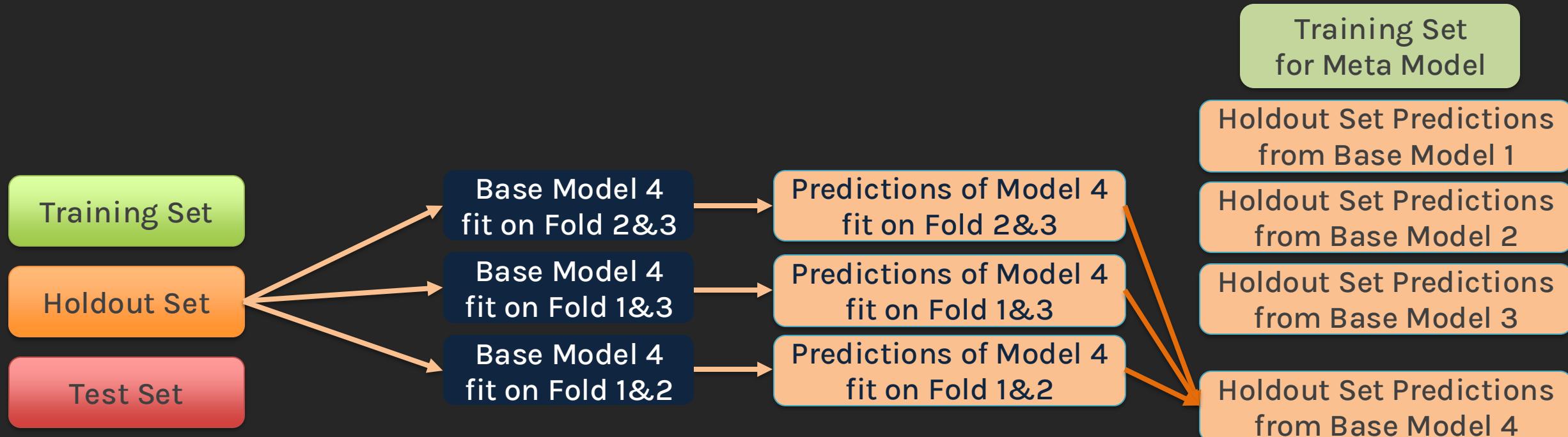
We repeat for all other base models. Generate holdout set predictions and take the average.



Stacking

Step 3:

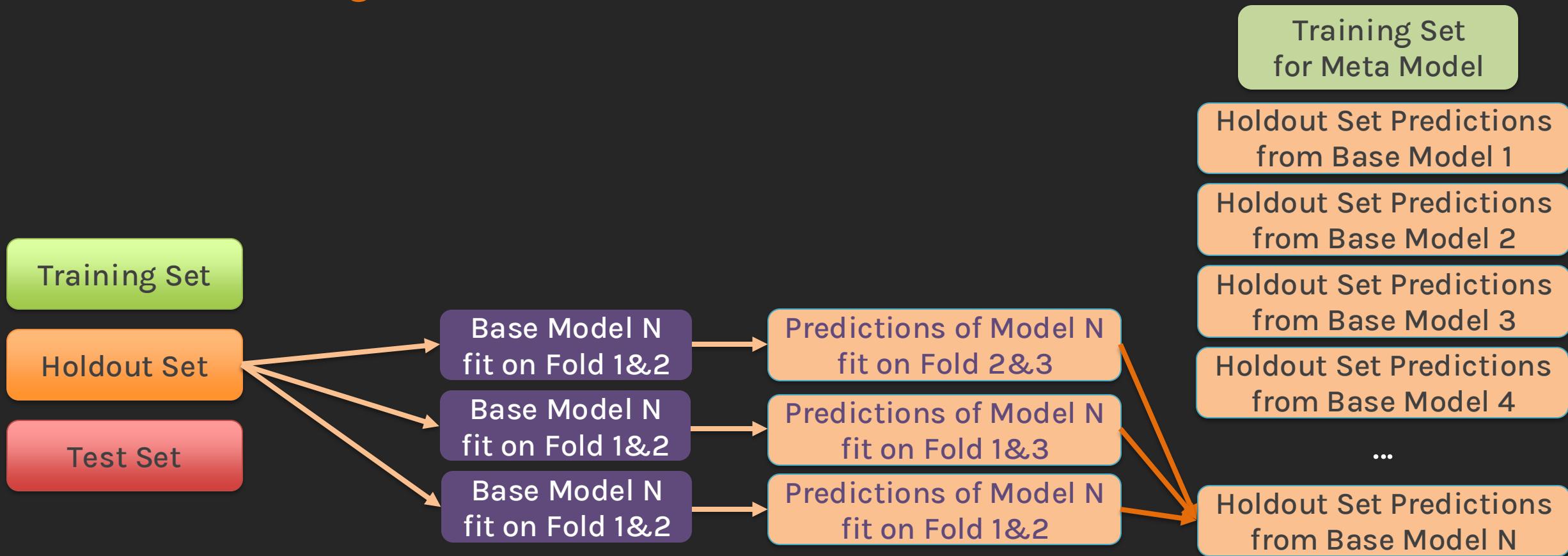
We repeat for all other base models. Generate holdout set predictions and take the average.



Stacking

Step 3:

We repeat for all other base models. Generate holdout set predictions and take the average.



Stacking

Step 3:

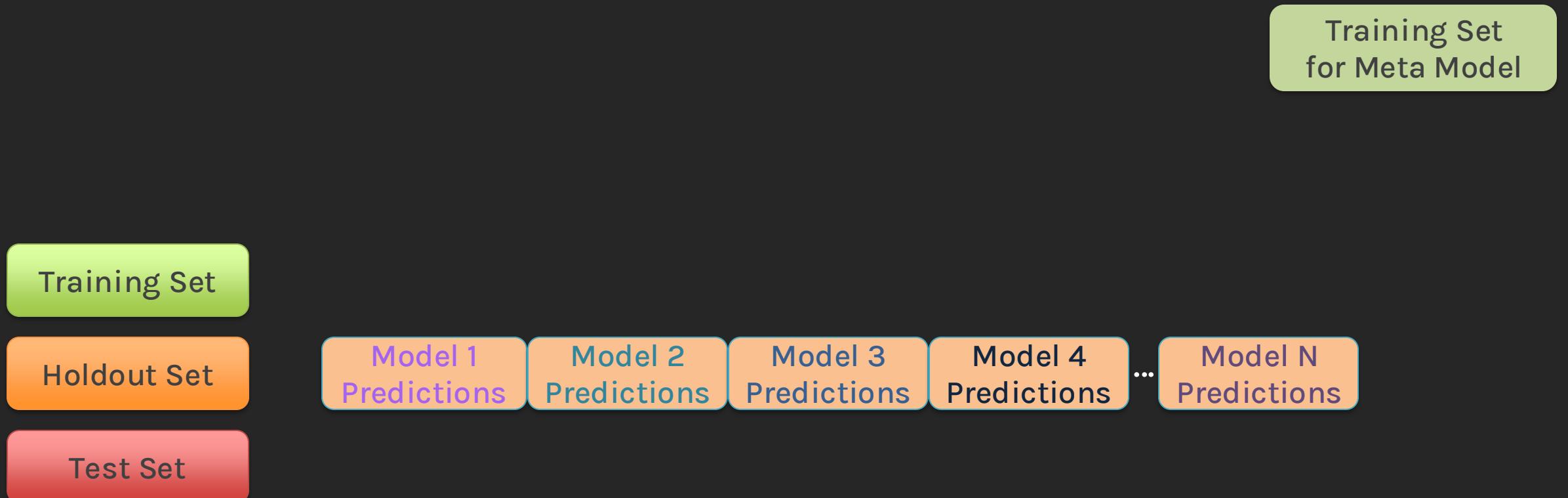
We repeat for all other base models. Generate holdout set predictions and take the average.



Stacking

Step 3:

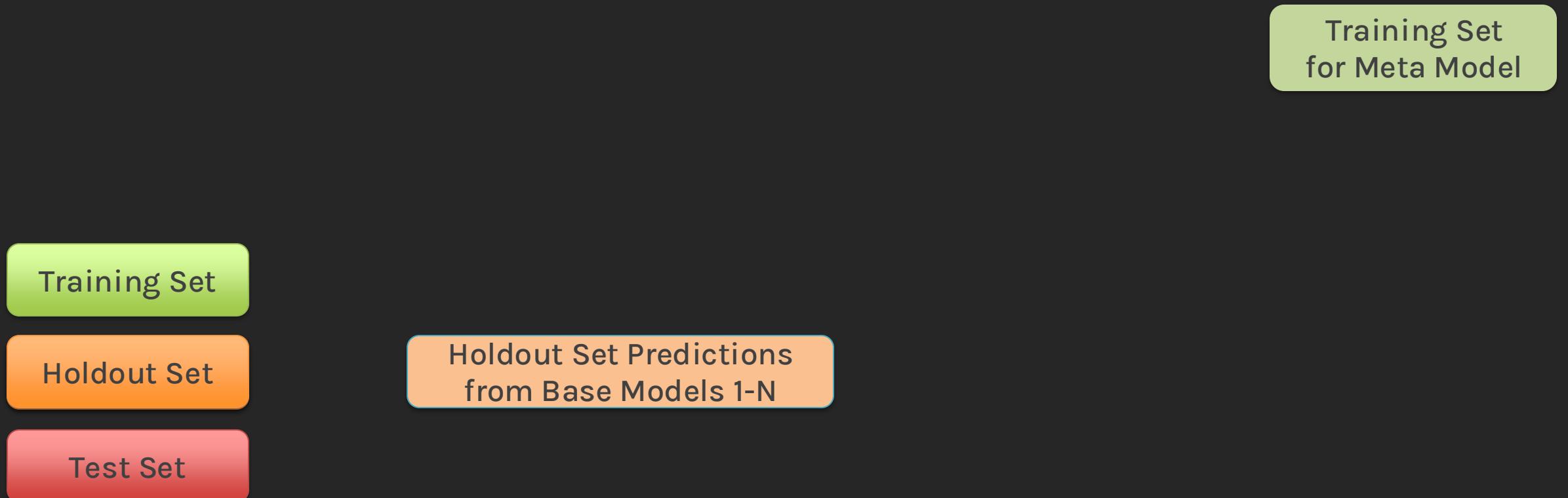
Combined features from holdout set will be ready for validating meta model.



Stacking

Step 3:

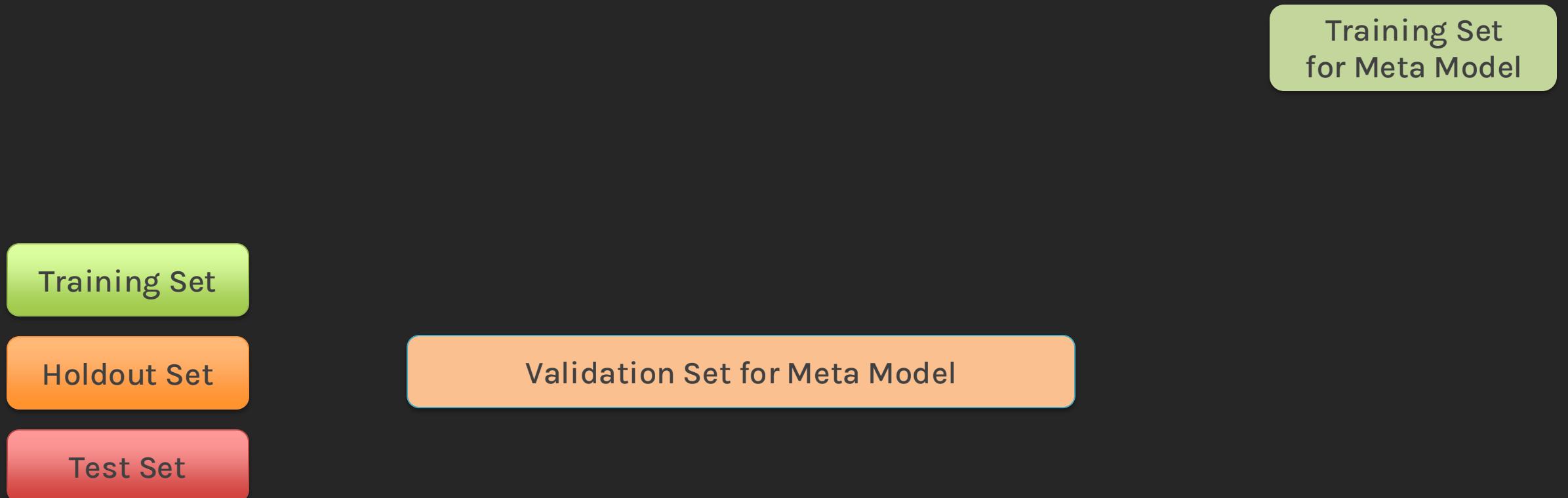
Combined features from holdout set will be ready for validating meta model.



Stacking

Step 3:

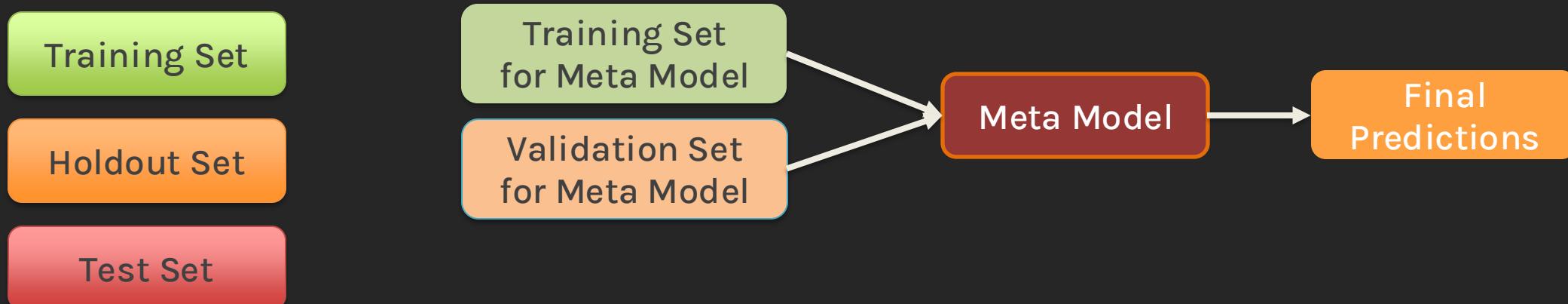
Combined features from holdout set will be ready for validating meta model.



Stacking

Step 4:

The remaining steps of fitting meta model and making predictions are the same as blending.



Important Note with sklearn

Note that we have used both the base model predictions and original data for training meta model when introducing blending and stacking.

In practice, we can also use the combination of only base model predictions and response variables (only y).

Although less features are provided to meta model, this avoids meta model from overfitting to the original data and makes it learn the right balance of predictions from base models.

This is controlled by a Boolean argument passthrough in sklearn's StackingRegressor() and StackingClassifier(). The default value is False (meaning meta model doesn't see the original data).

How to combine outputs from base models?

Model outputs can be different depending on the tasks.

Regression tasks are easier to understand, as the model outputs a single value for each sample.

The outputs of classifiers are usually in the format of probabilities for each class. How do we combine them?

- For binary classification, class 0 and class 1 probabilities are perfectly collinear, so we only keep one of them (e.g. take predicted probability for class 1 as the model output).
- For multi-class classification, all probabilities sum to one for each sample, so we keep ‘num_classes-1’ classes. Usually the probabilities for class 0 (first column) are dropped.

A Quick Summary

With ensemble methods, we can combine models to obtain a more accurate and/or more robust model, including

- Bagging and Boosting with homogeneous models
- Blending and Stacking with heterogeneous models

The base models in the ensemble are combined in a cooperative way, and to achieve this, we need all base models to fit on the entire training set and learn data distributions and patterns over the complete range.

This could be potentially hard for base models if the dataset contains several different regimes, which have different relationships between input and output.

Is it possible to cover different input regions with different learners?

Outline

- Ensemble Methods
- Blending
- Stacking
- **Mixture of Experts**

Mixture of Experts

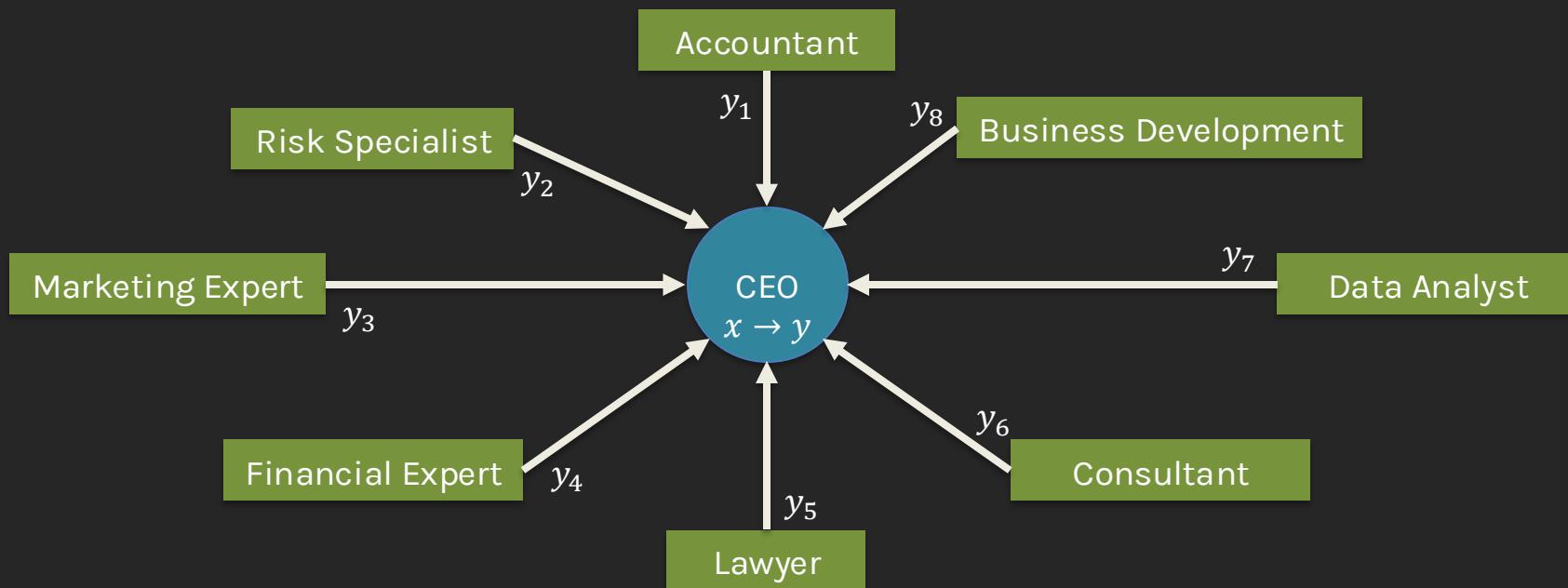
Intuition: Consider the case of a physical examination, and the goal is to have the diagnosis if the patients have any severe diseases.

Patients will go to different consultation rooms, and there will be experts of different medical fields waiting to check different indicators.

You will get a report in the end summarized by some professional doctors or even computer programs, perhaps with an emphasis on some particular sections you want to pay more attention to.

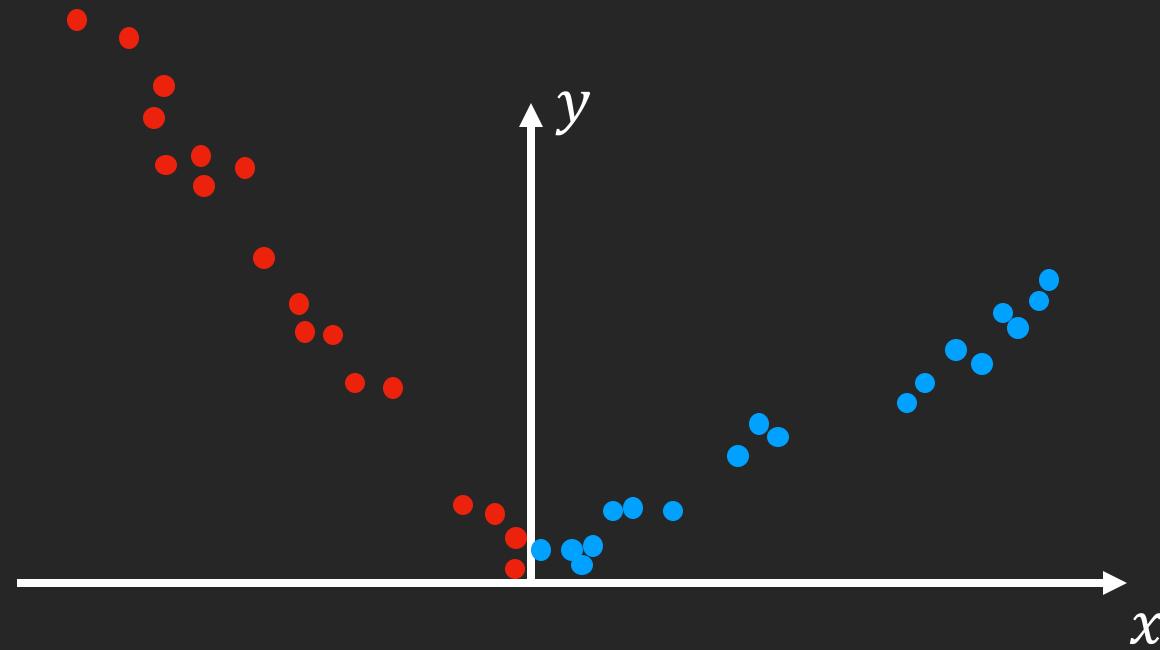
Mixture of Experts

Another Intuition: Consider a regular company with all the different departments. If the CEO wants to make an important decision, he/she may need to consult the professionals from each department to get a comprehensive plan.



Specialization instead of Cooperation

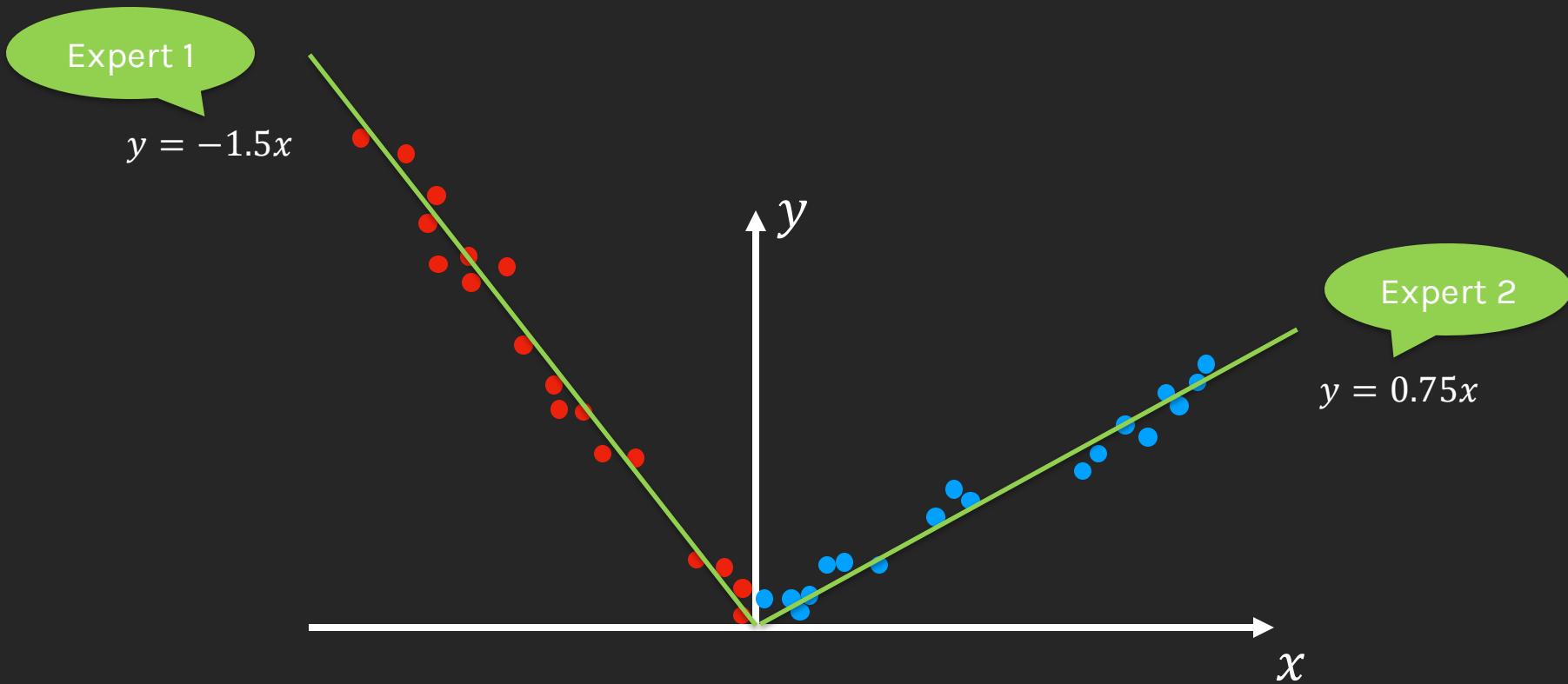
Given the following data example, we have two different data regimes for positive x and negative x values.



Specialization instead of Cooperation

Given the following data example, we have two different data regimes for positive x and negative x values.

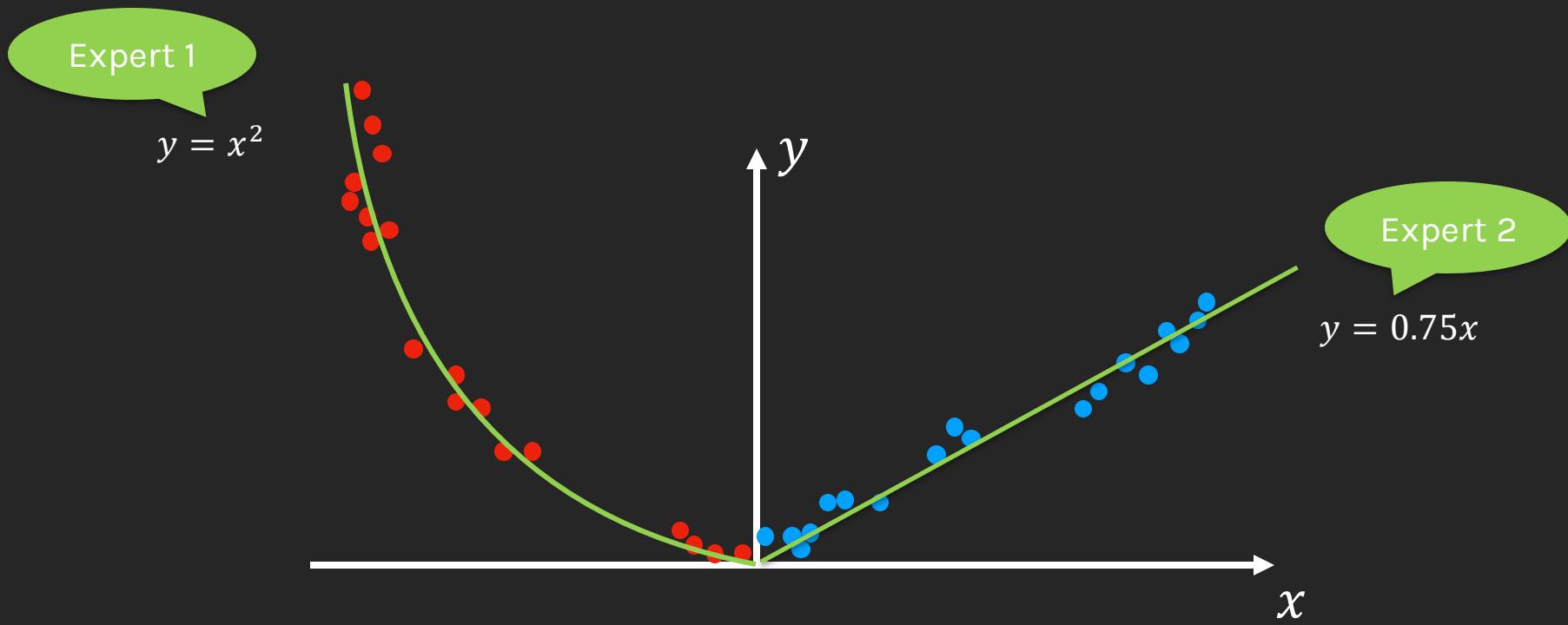
This can be easy if we have two different learners fit on each input region!



Specialization instead of Cooperation

Given the following data example, we have two different data regimes for positive x and negative x values.

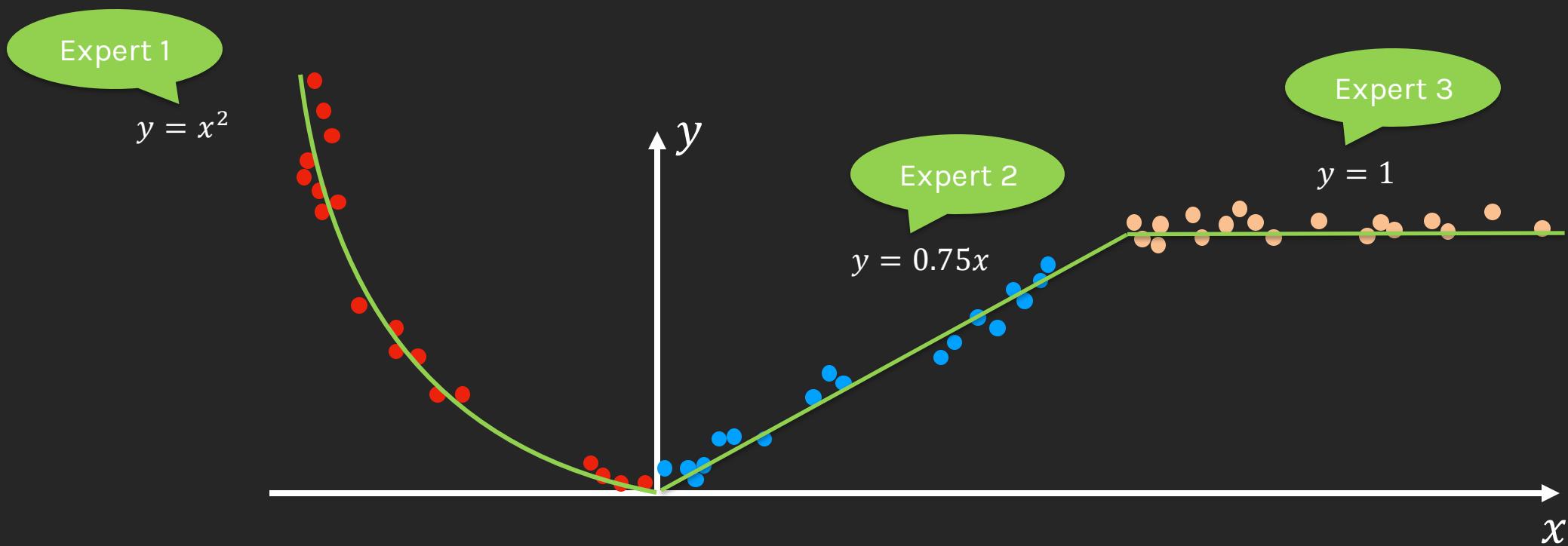
It is still easy even different families of models are involved in two regions!



Specialization instead of Cooperation

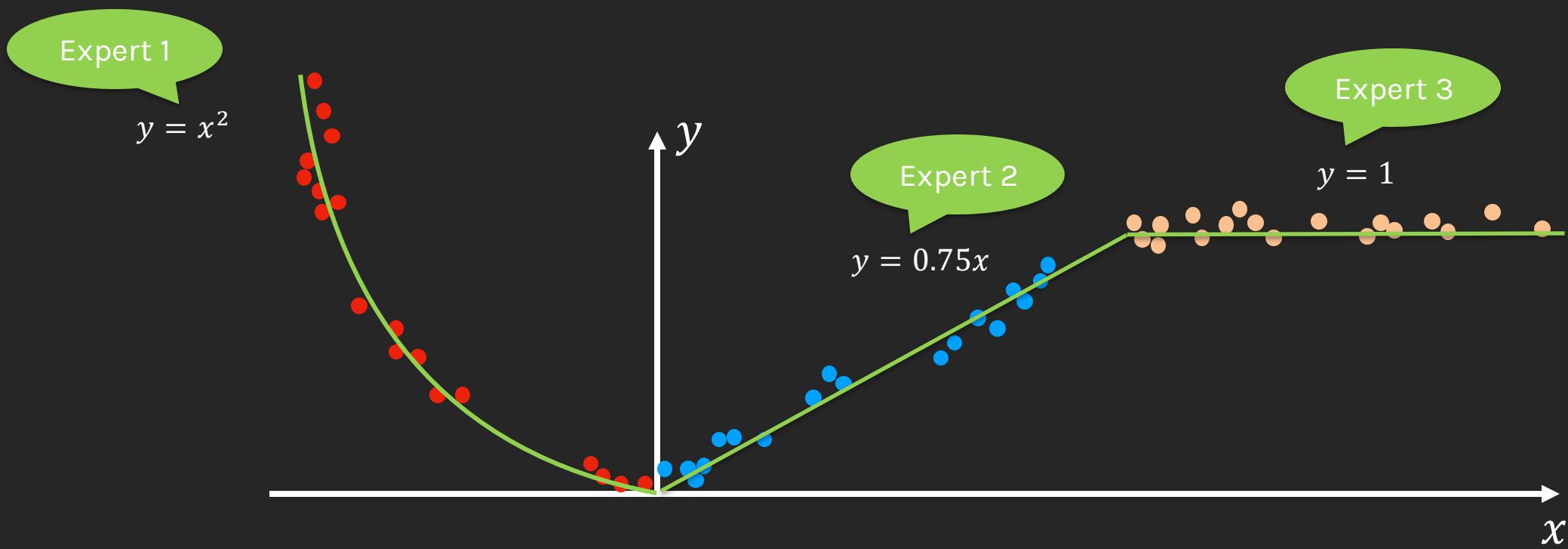
Given the following data example, we have two different data regimes for positive x and negative x values.

More data relationships? Just get another expert!



Specialization instead of Cooperation

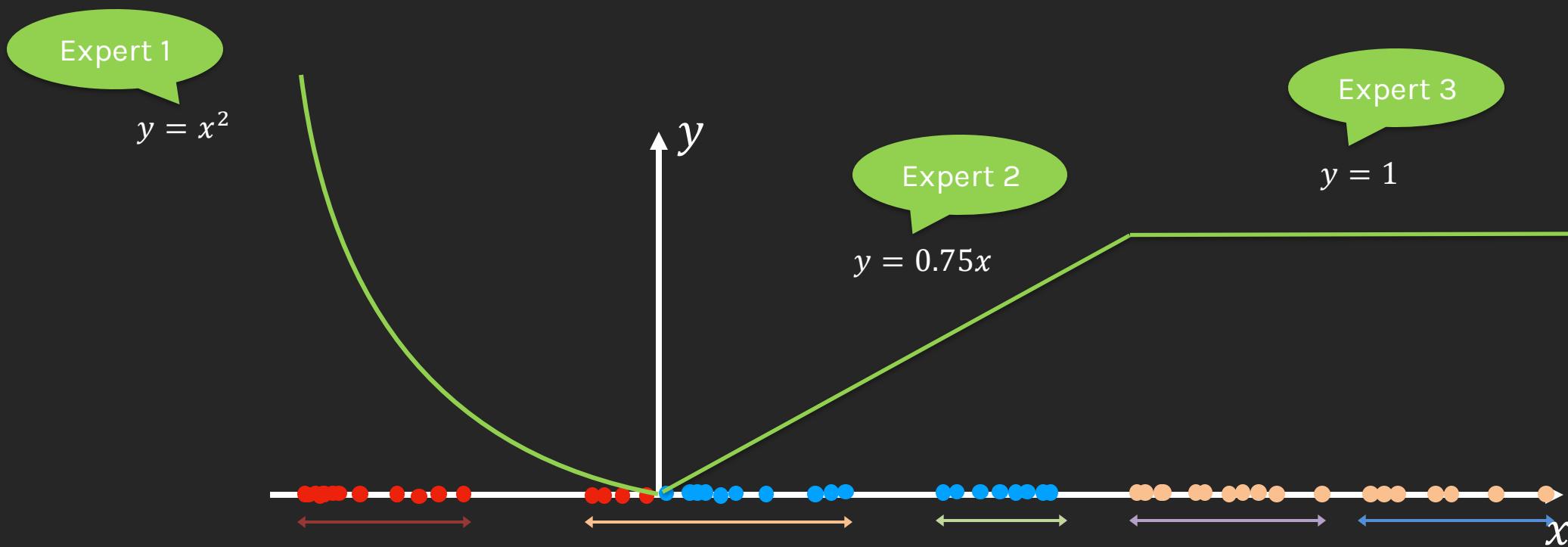
It is easy to tell that 3 experts are needed here based on (x, y) distributions.



Specialization instead of Cooperation

It is easy to tell that 3 experts are needed here based on (x, y) distributions.
But to identify the domains, we usually only have this to start with.

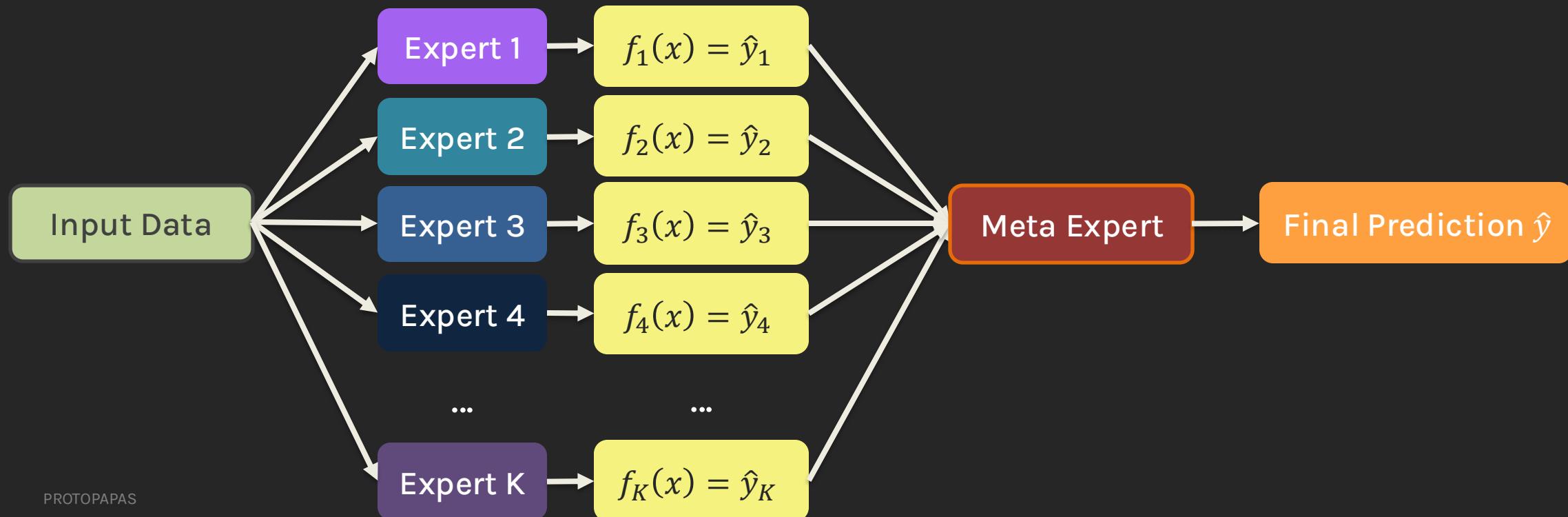
How do we assign these points to the correct expert?



Mixture of Experts

We begin with a framework similar to blending/stacking.

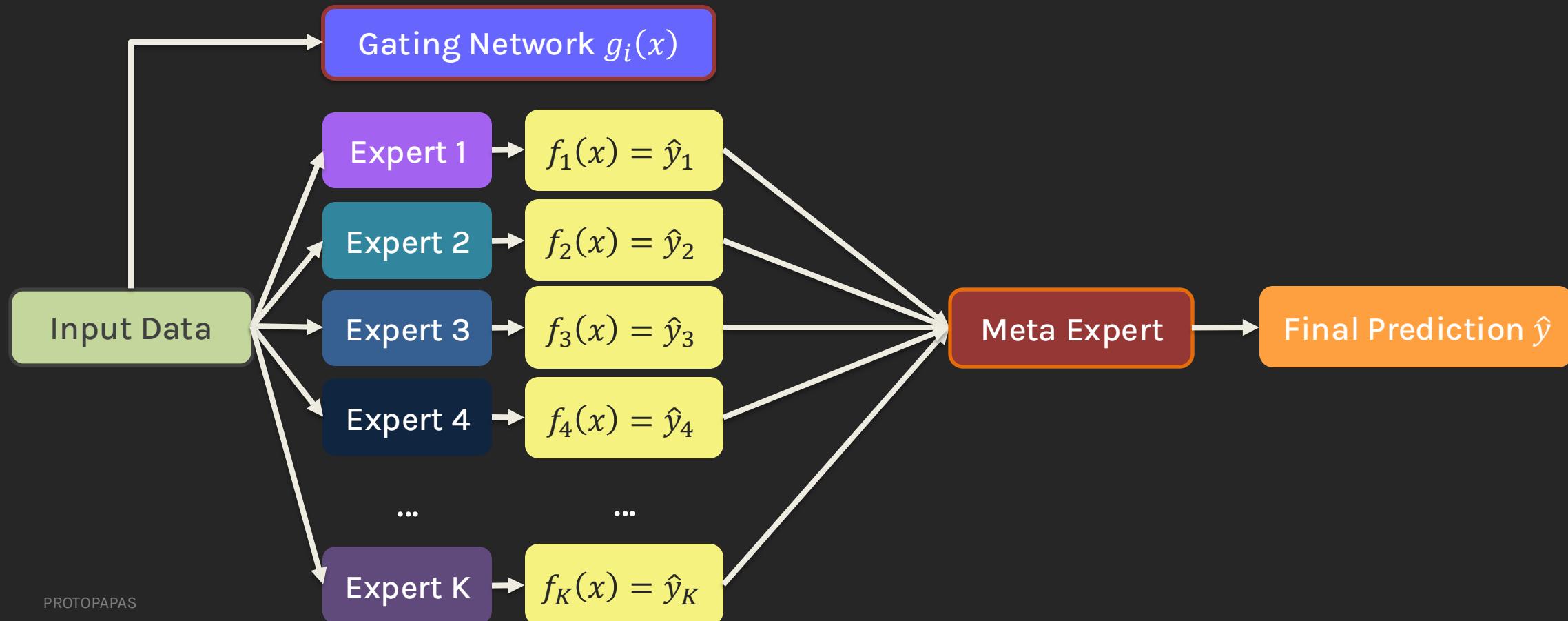
- Each trained expert will make predictions \hat{y}_i
- A meta expert will learn from experts to make final prediction \hat{y} .



Mixture of Experts

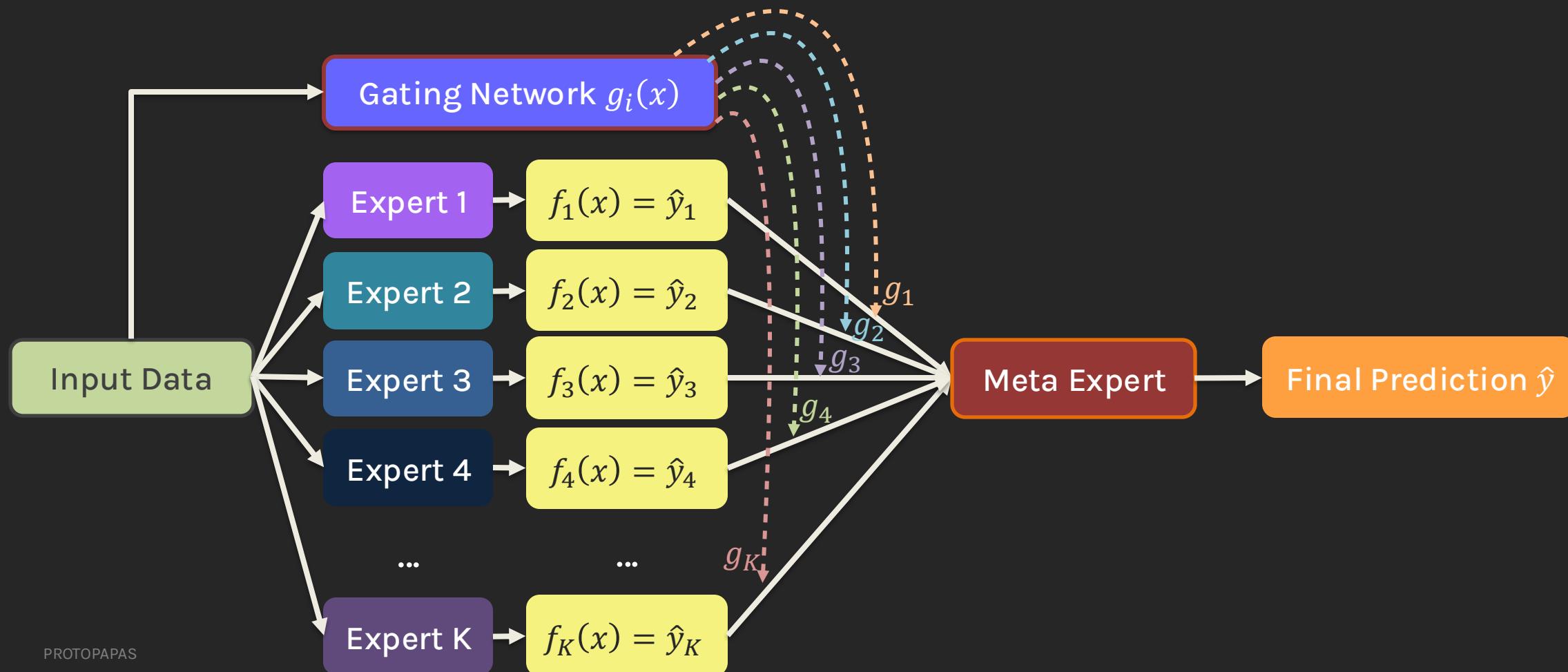
What makes Mixture of Experts special and different?

We have a *gating network*!



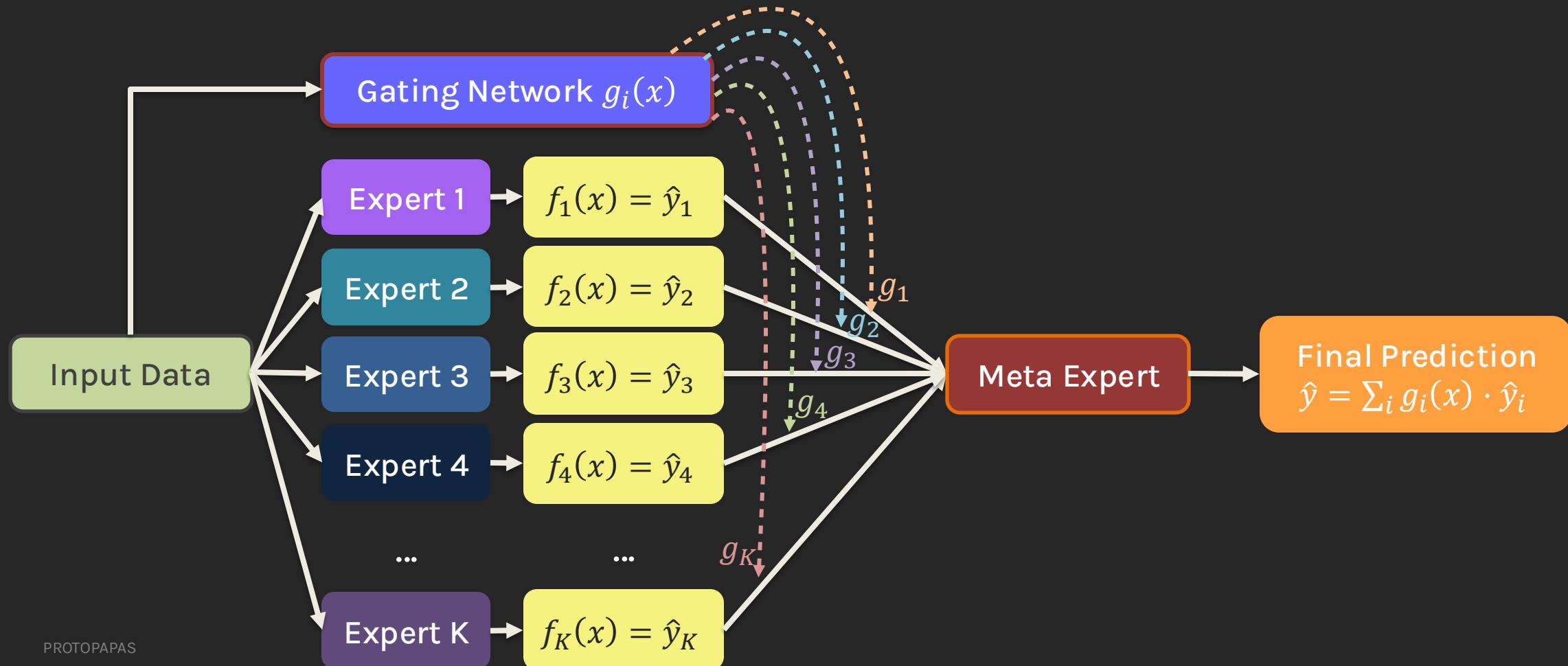
Mixture of Experts

Gating network uses gating functions g_i to decide what combination of experts to use.



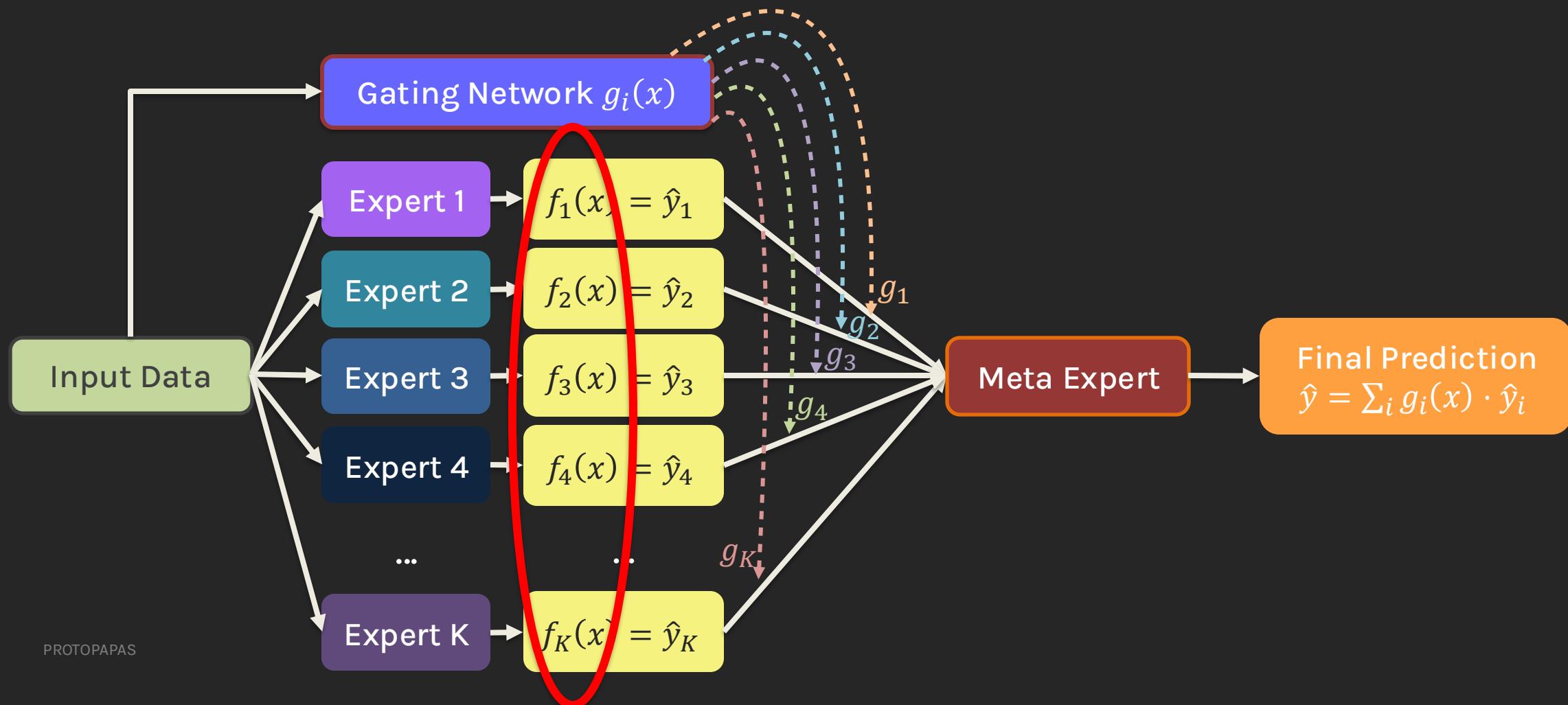
Mixture of Experts

Gating network uses gating functions g_i to decide what combination of experts to use.



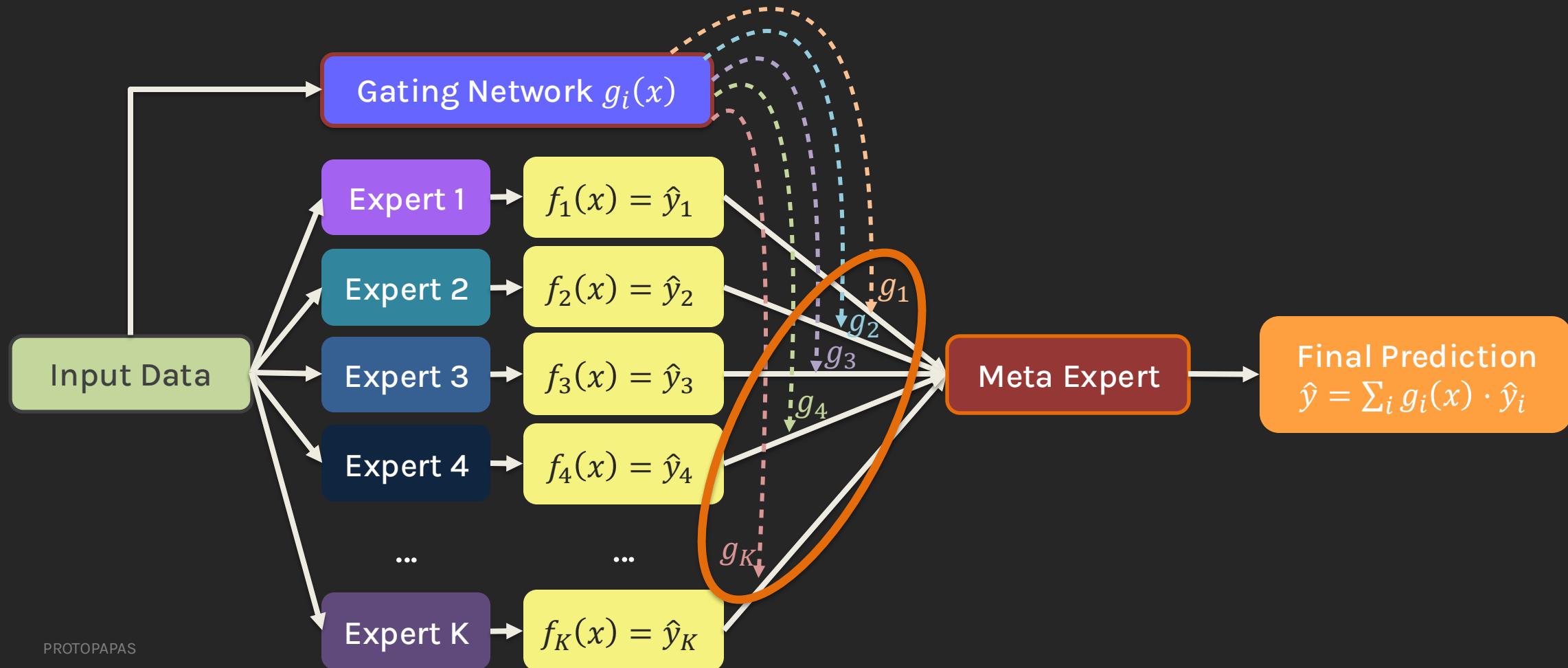
Mixture of Experts

Fitting the model involves 1) learning the parameters of each expert,



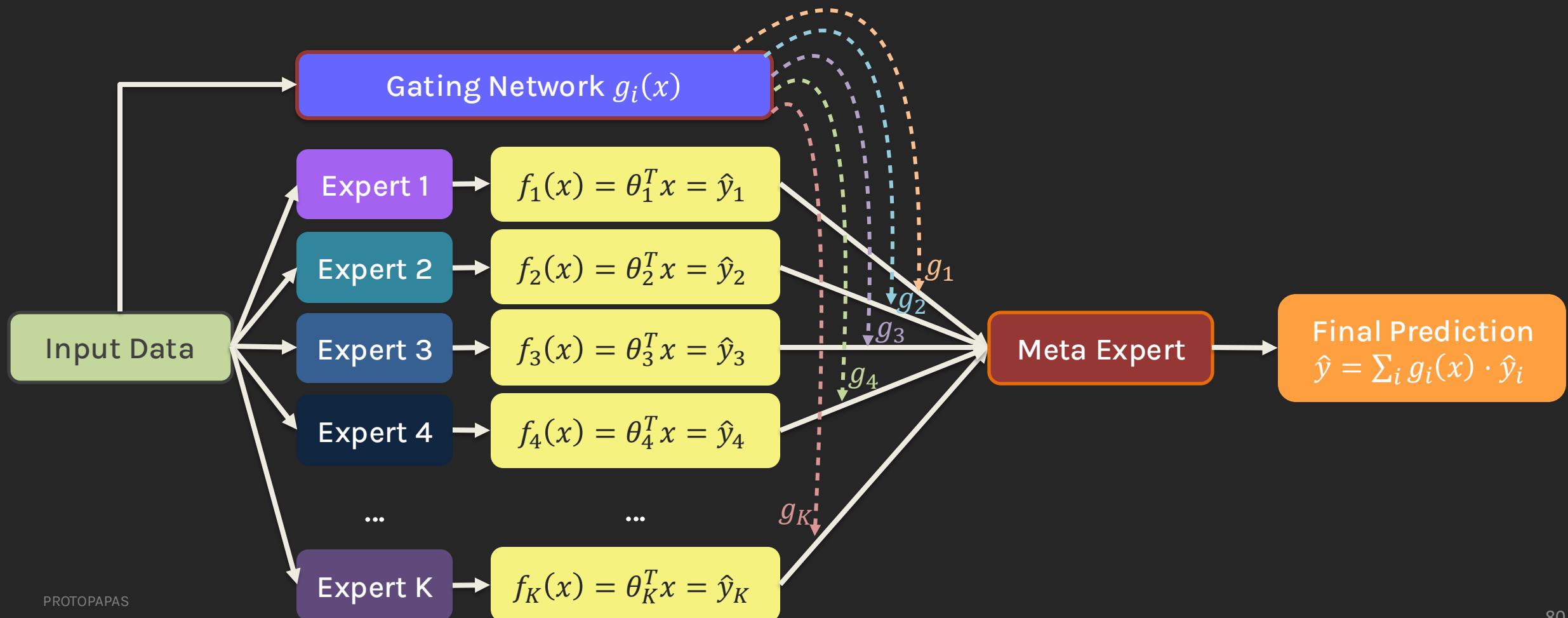
Mixture of Experts

Fitting the model involves 1) learning the parameters of each expert, and 2) learning the parameters of the *gating network*.



Mixture of Experts

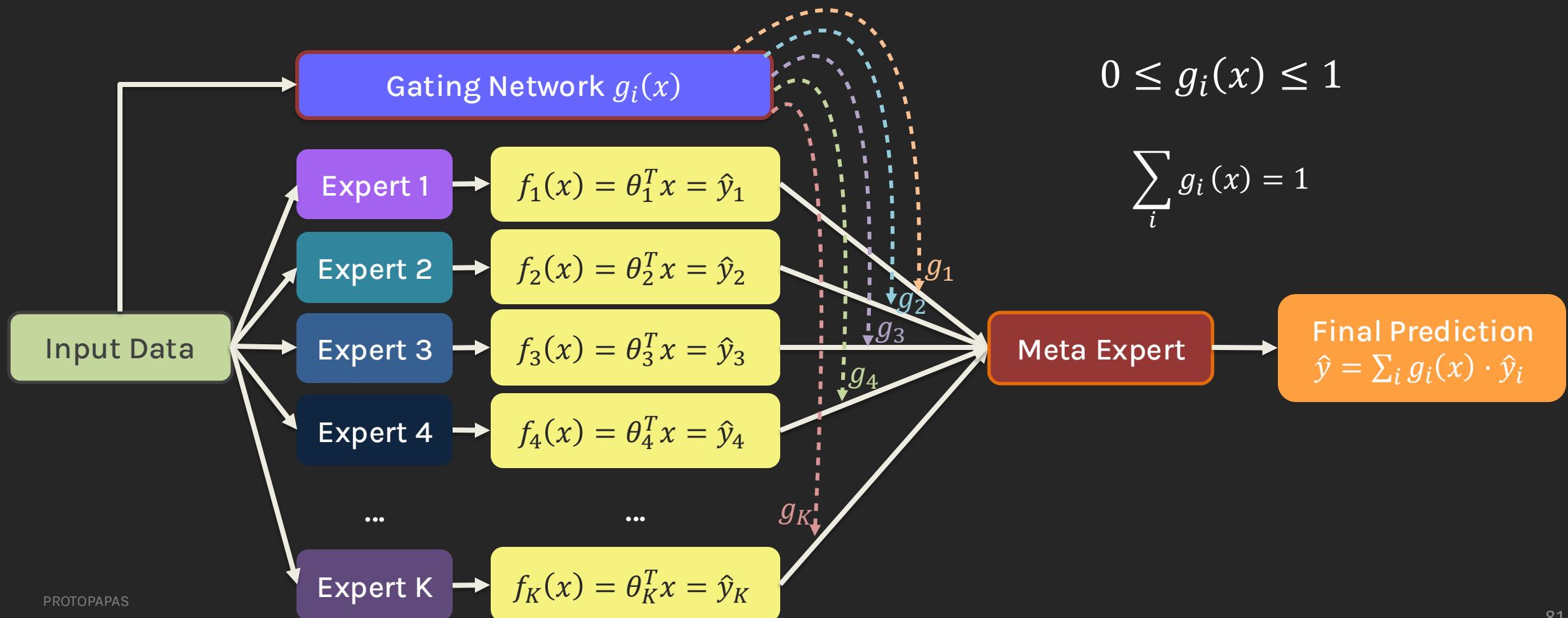
For simplicity here, let's assume the experts are a family of linear models.
Note that these models don't have to be the same/homogeneous.



Mixture of Experts

What about the gating network?

Are there any desired properties of the gating functions?

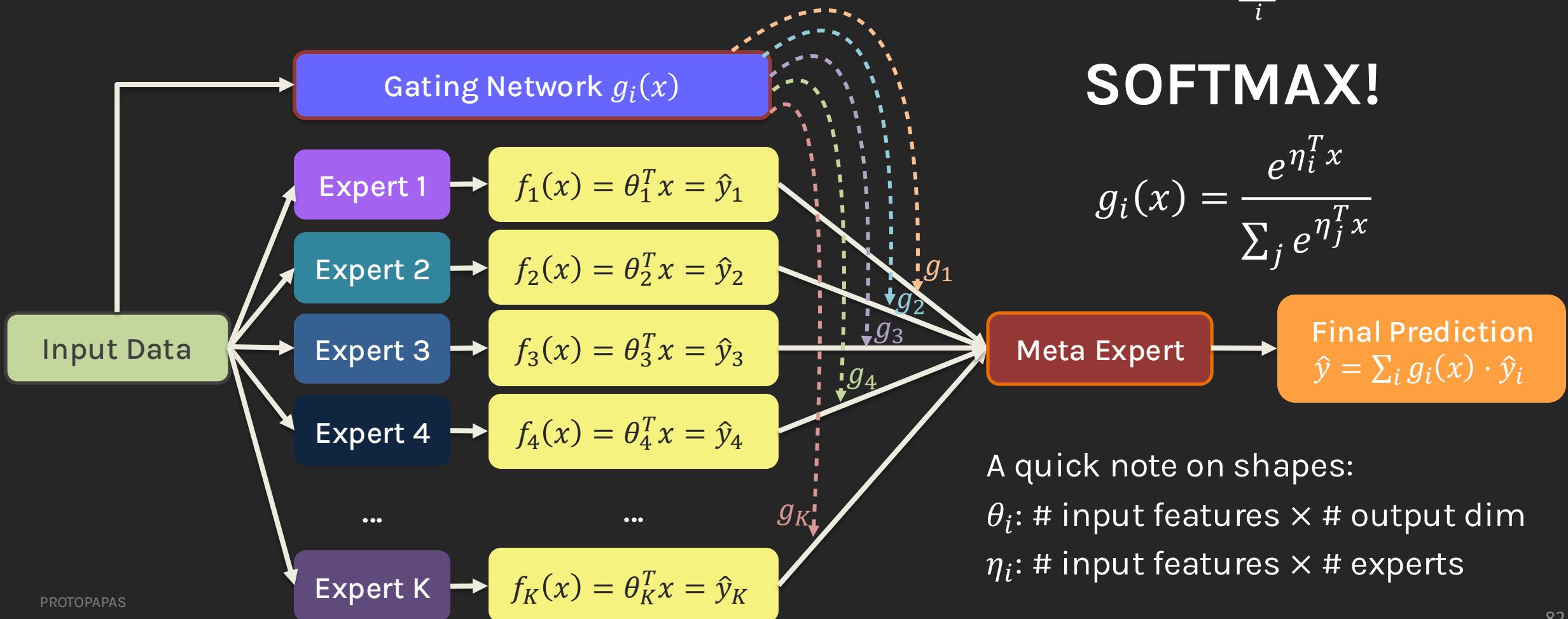


Mixture of Experts

Are there any functions that satisfy these requirements and help us parameterize the network?

$$0 \leq g_i(x) \leq 1$$

$$\sum_i g_i(x) = 1$$



Mixture of Experts

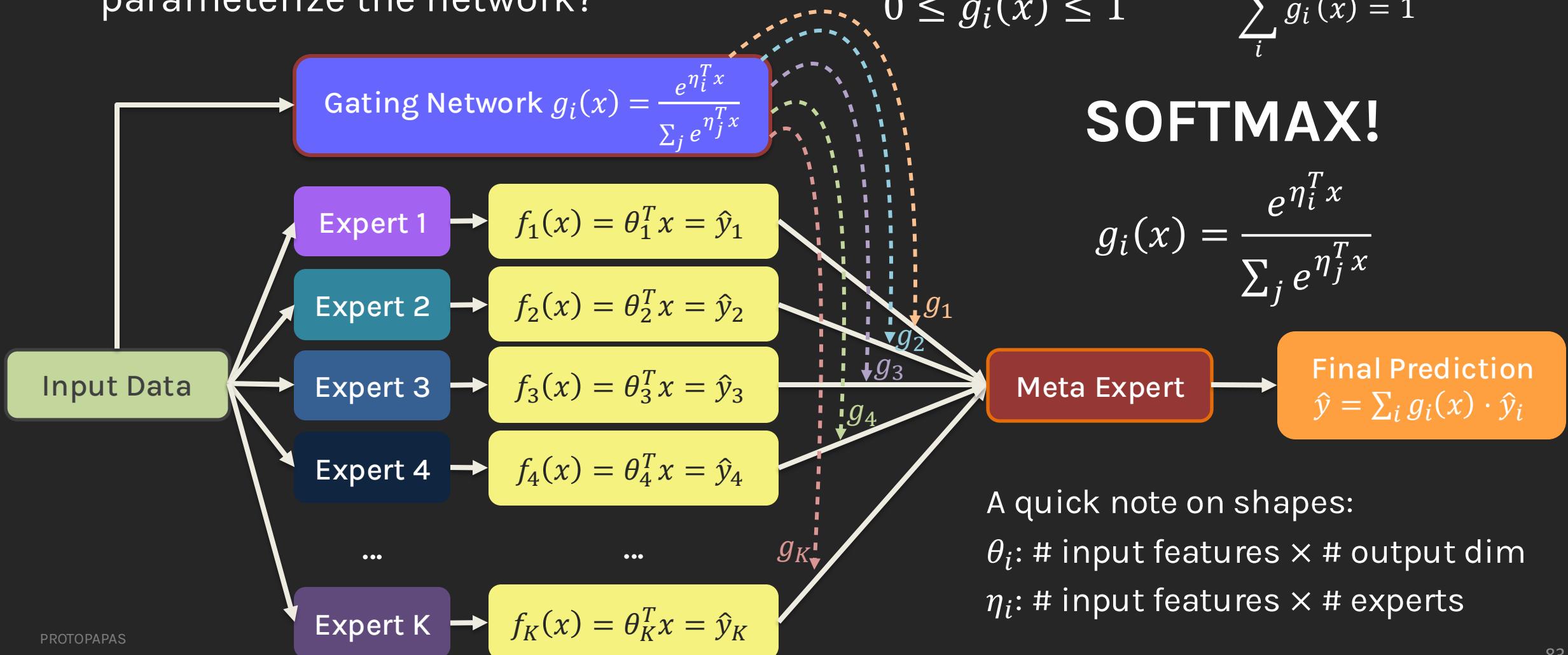
Are there any functions that satisfy these requirements and help us parameterize the network?

$$0 \leq g_i(x) \leq 1$$

$$\sum_i g_i(x) = 1$$

SOFTMAX!

$$g_i(x) = \frac{e^{\eta_i^T x}}{\sum_j e^{\eta_j^T x}}$$



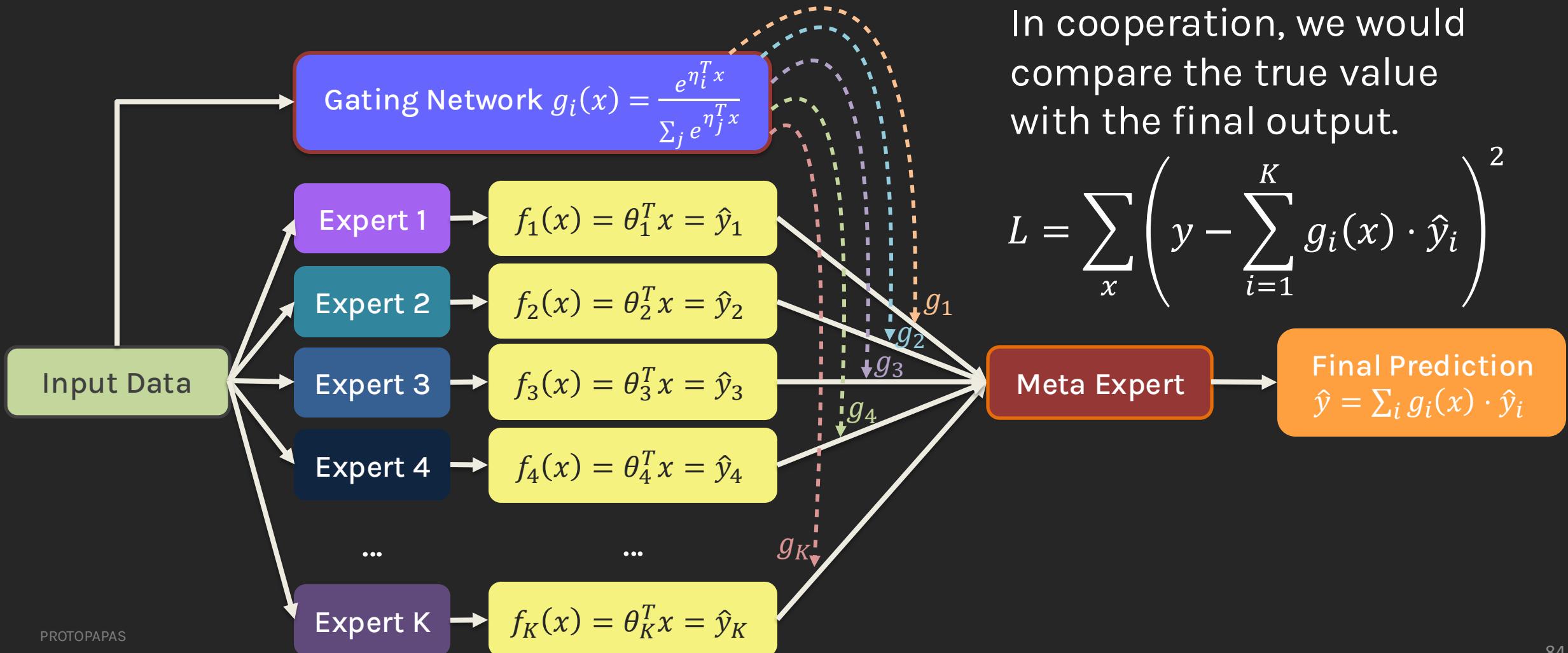
A quick note on shapes:

θ_i : # input features \times # output dim

η_i : # input features \times # experts

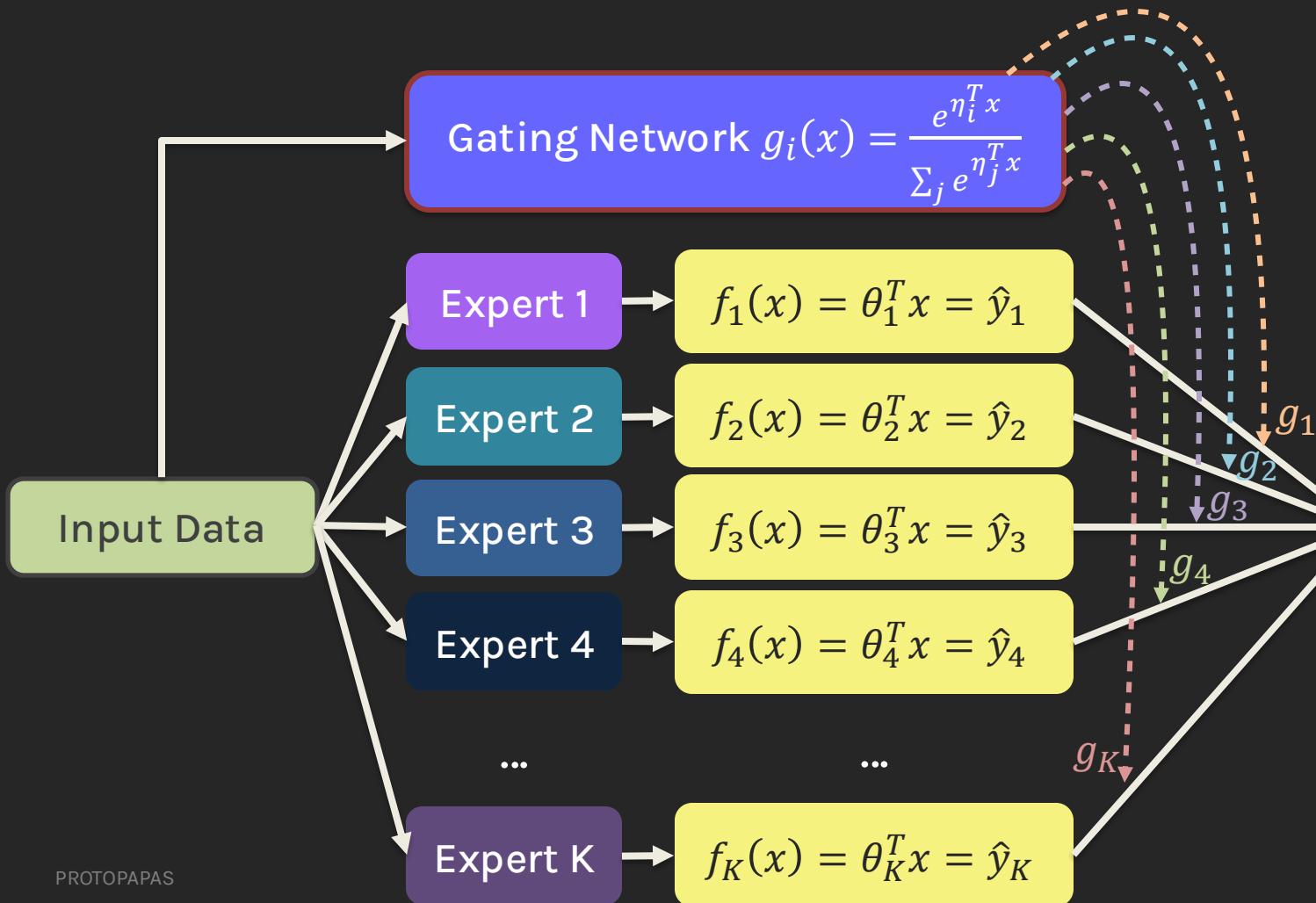
Mixture of Experts

All the parameters are ready. How does the model learn and evaluate?



Mixture of Experts

All the parameters are ready. How does the model learn and evaluate?



In cooperation, we would compare the true value with the final output.

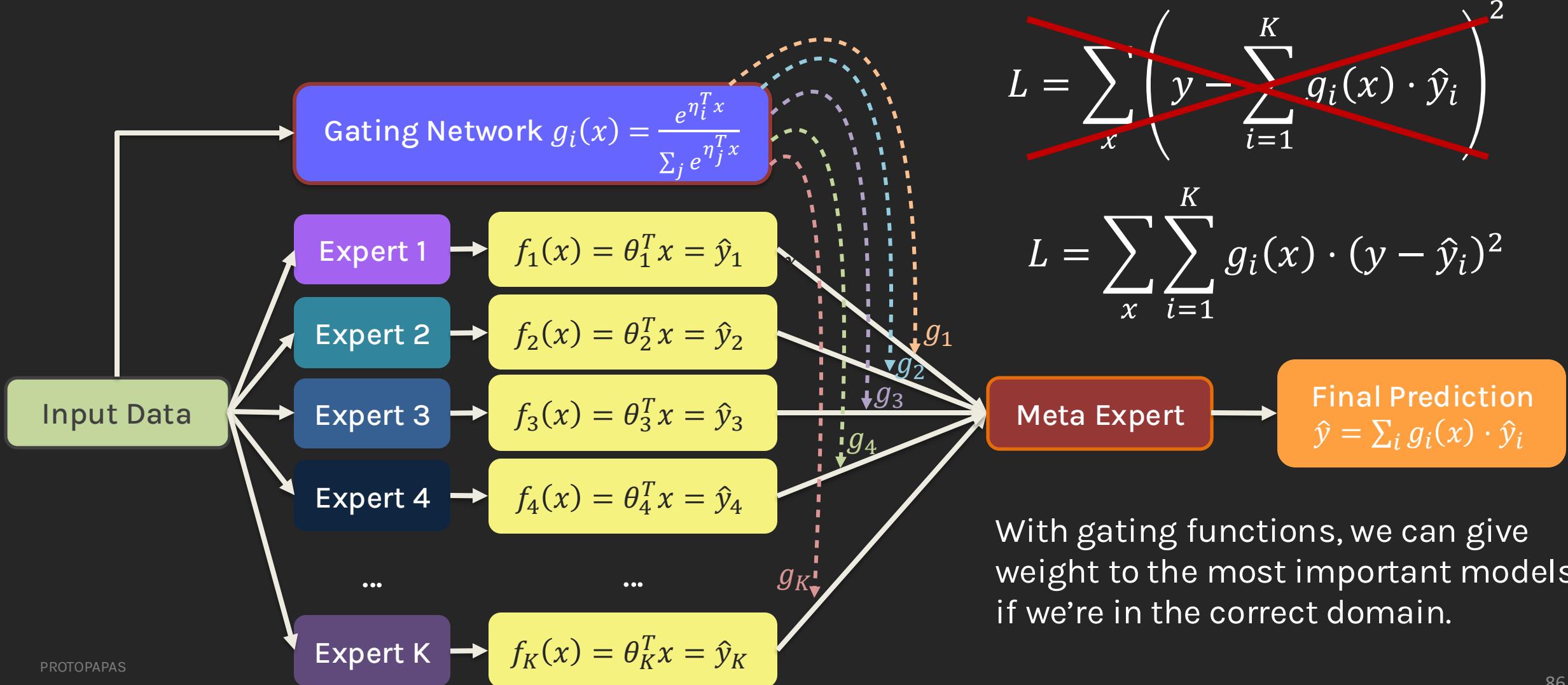
$$L = \sum_x \left(y - \sum_{i=1}^K g_i(x) \cdot \hat{y}_i \right)^2$$



But this goes against our goal of having specialized experts!

Mixture of Experts

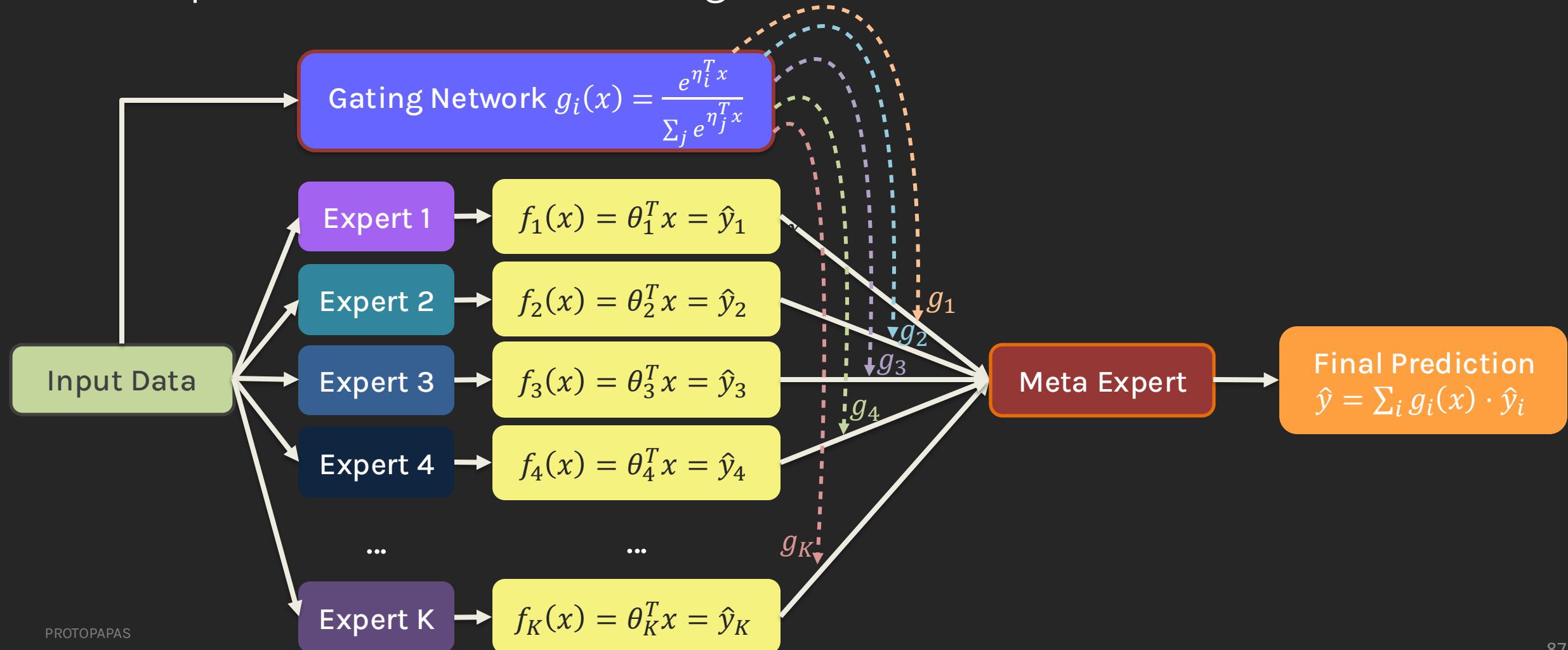
All the parameters are ready. How does the model learn and evaluate?



Mixture of Experts

With the loss function, how do we update our model?
Compute the derivatives and do gradient descent!

$$L = \sum_x \sum_{i=1}^K g_i(x) \cdot (y - \hat{y}_i)^2$$

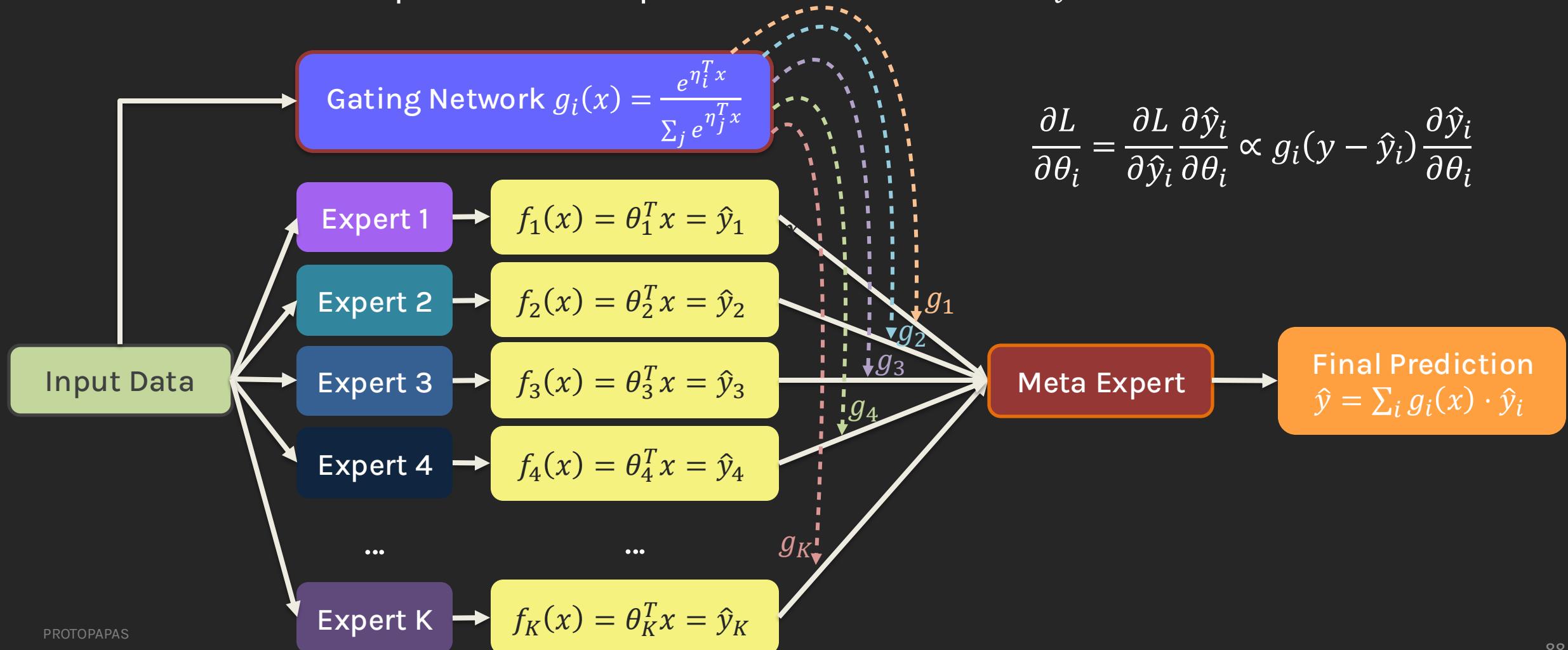


Mixture of Experts

For simplicity, let's focus on only one sample x .

To train each expert, we take partial derivatives to θ_i .

$$L = \sum_{i=1}^K g_i(x) \cdot (y - \hat{y}_i)^2$$



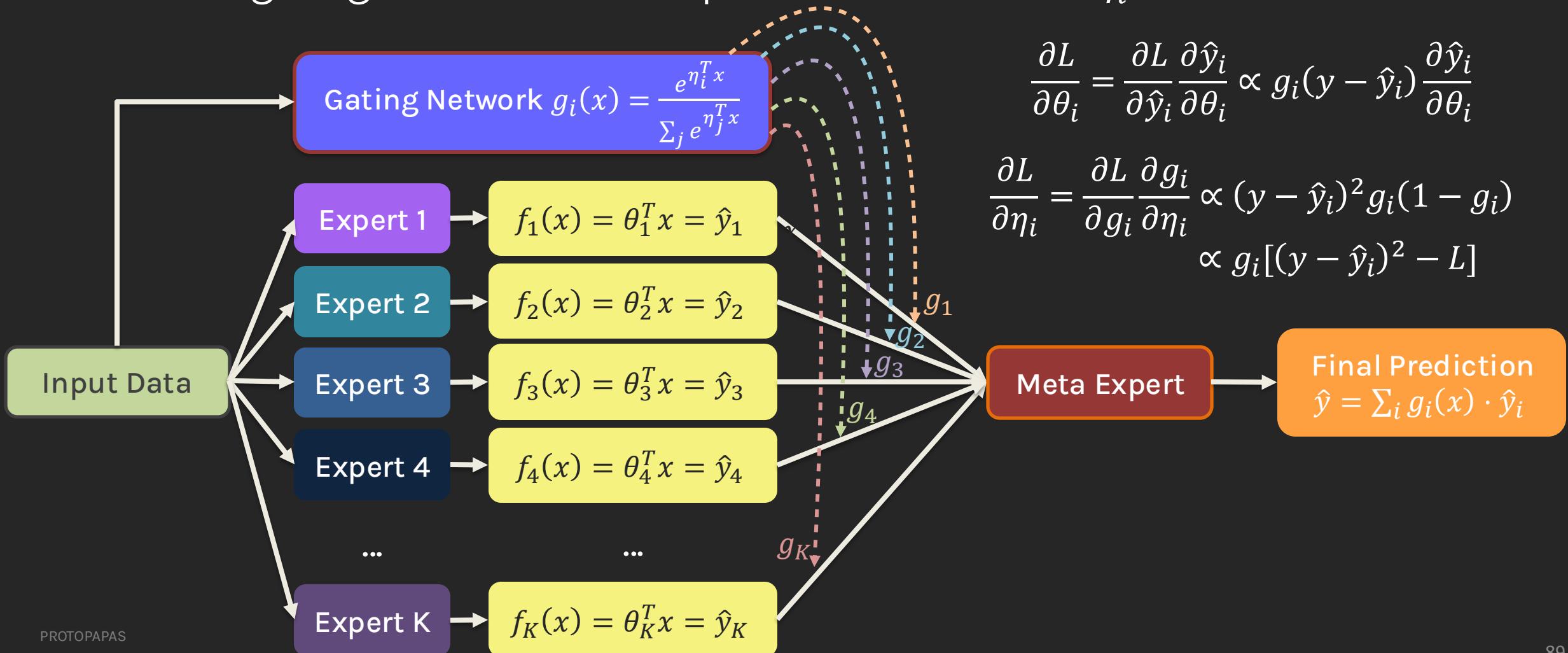
$$\frac{\partial L}{\partial \theta_i} = \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial \theta_i} \propto g_i(y - \hat{y}_i) \frac{\partial \hat{y}_i}{\partial \theta_i}$$

Mixture of Experts

For simplicity, let's focus on only one sample x .

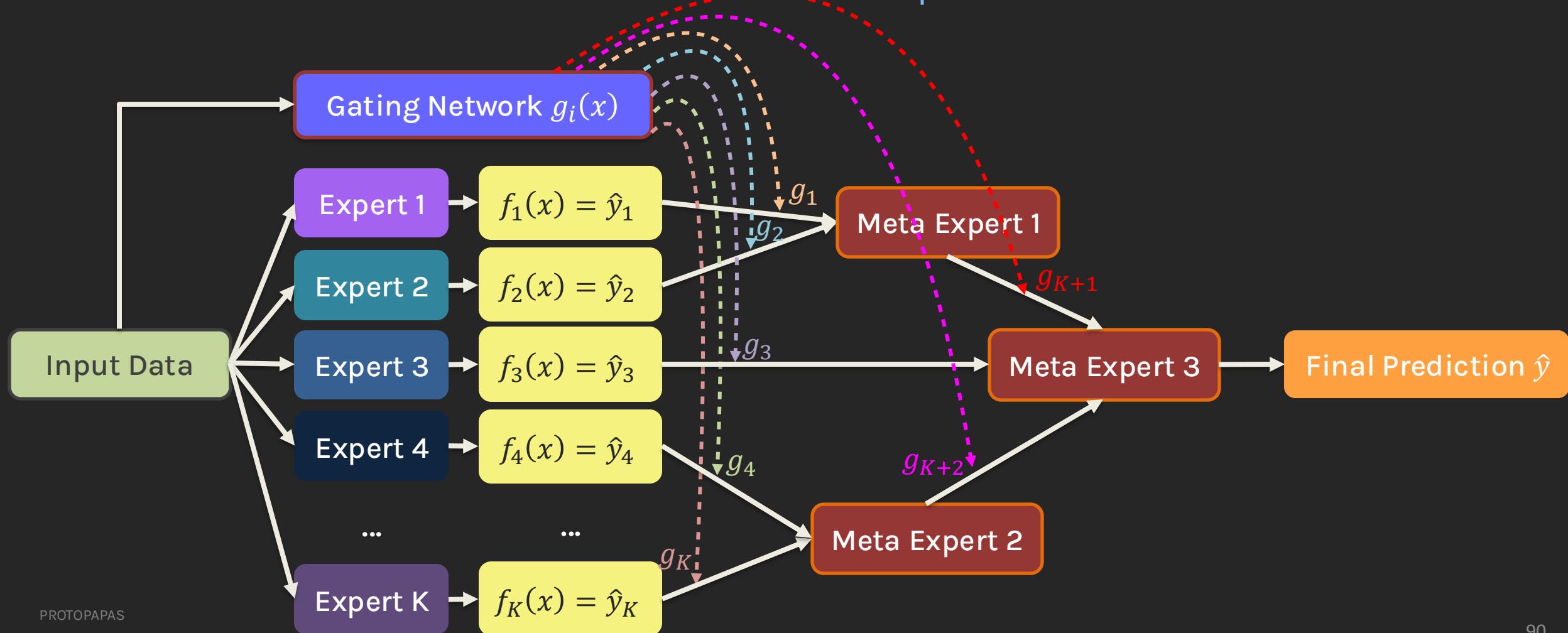
To train gating network, we take partial derivatives to η_i .

$$L = \sum_{i=1}^K g_i(x) \cdot (y - \hat{y}_i)^2$$



Hierarchical Mixture of Experts

If the output is conditioned on multiple levels of gating functions, the mixture is called a **hierarchical mixture of experts**.





GIFWAVE.COM

