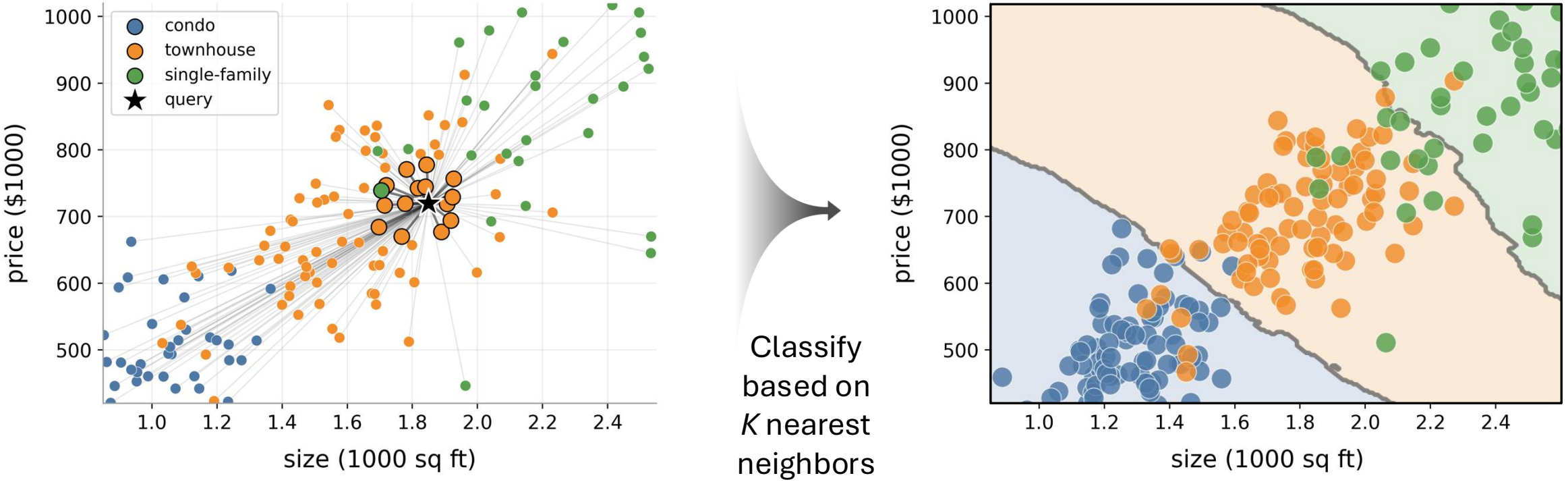
COM SCI M148, Lecture 11:
Logistic Regression

Prof. Nathan Szymanski
nszymanski@seas.ucla.edu
Department of Materials Science

UCLA

Samueli
School of Engineering

From last time: KNN classification



- **No model training is needed**, but we must store all of the data and recompute distances every time a new point is queried.
- What if we'd like to train a model up front, then get rid of the training data and reuse the model to do inference on new data? Sounds familiar…

Recall: Parametric *vs.* Non-parametric models

Parametric models

- We assume $f(x)$ has a specific form – *e.g.*, a polynomial.
- The fitted model is completely described by a finite set of parameters (β); the training data is not needed for prediction.

Non-Parametric models

- We make *no assumption* about the form $f(x)$ takes.
- Predictions *depend directly* on the stored training data, which is actively referenced when making predictions.

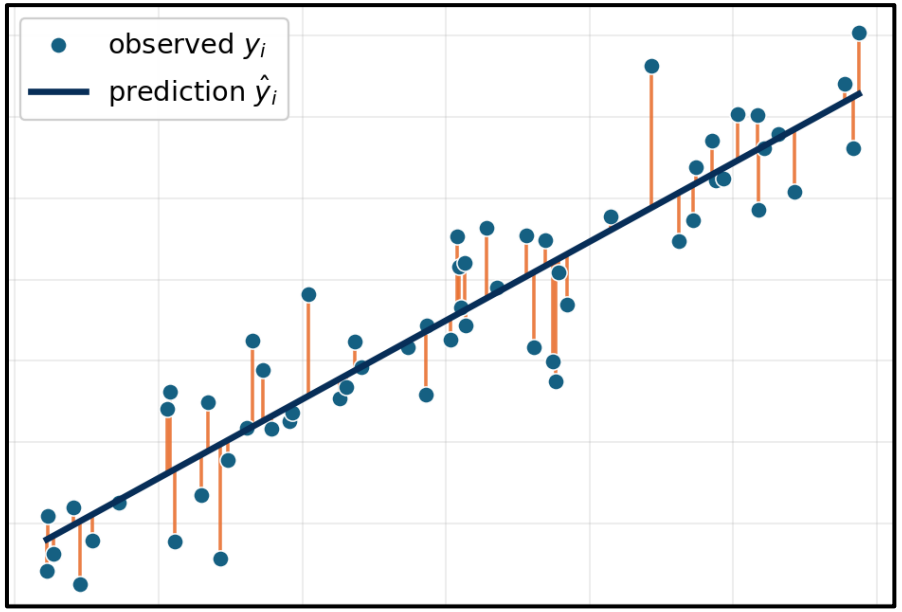
Today: A parametric model for classification

Parametric models

- We assume $f(x)$ has a specific form – *e.g.*, a polynomial.
- The fitted model is completely described by a finite set of parameters (β); the training data is not needed for prediction.

$$\hat{y} = \beta_0 + \sum_i \beta_i x_i$$

Remember: the coefficients provide ***interpretability.*** How much of an effect does each feature have?



Can we develop a similar **parametric model for classification?**

Let's start with binary classification (2 classes)

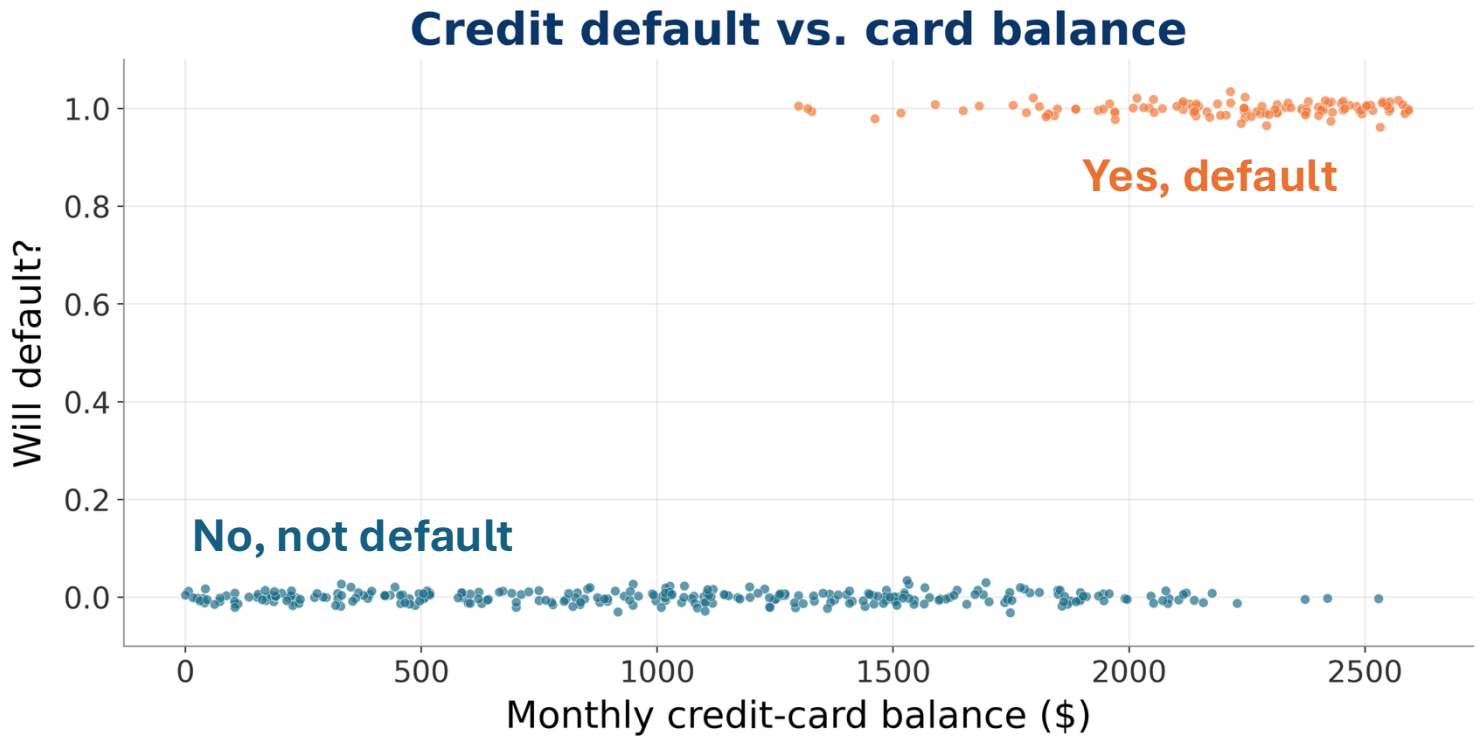
Input features: x → **Learned function:** $f(x)$

- **Class 1** — Make $y = 0$
- **Class 2** — Make $y = 1$

This is a function that takes numbers as input and produces a **number as output**

EX: Based on their monthly credit balance, will a customer default?

Monthly balance	Defaulted
\$512	No ($y = 0$)
\$1020	No ($y = 0$)
…	…
\$2345	Yes ($y = 1$)



Our function should give class *probabilities*

Our learned function $f(x)$ should be a probability P ranging from 0 to 1, where:

Probability of $y = 1$ given x

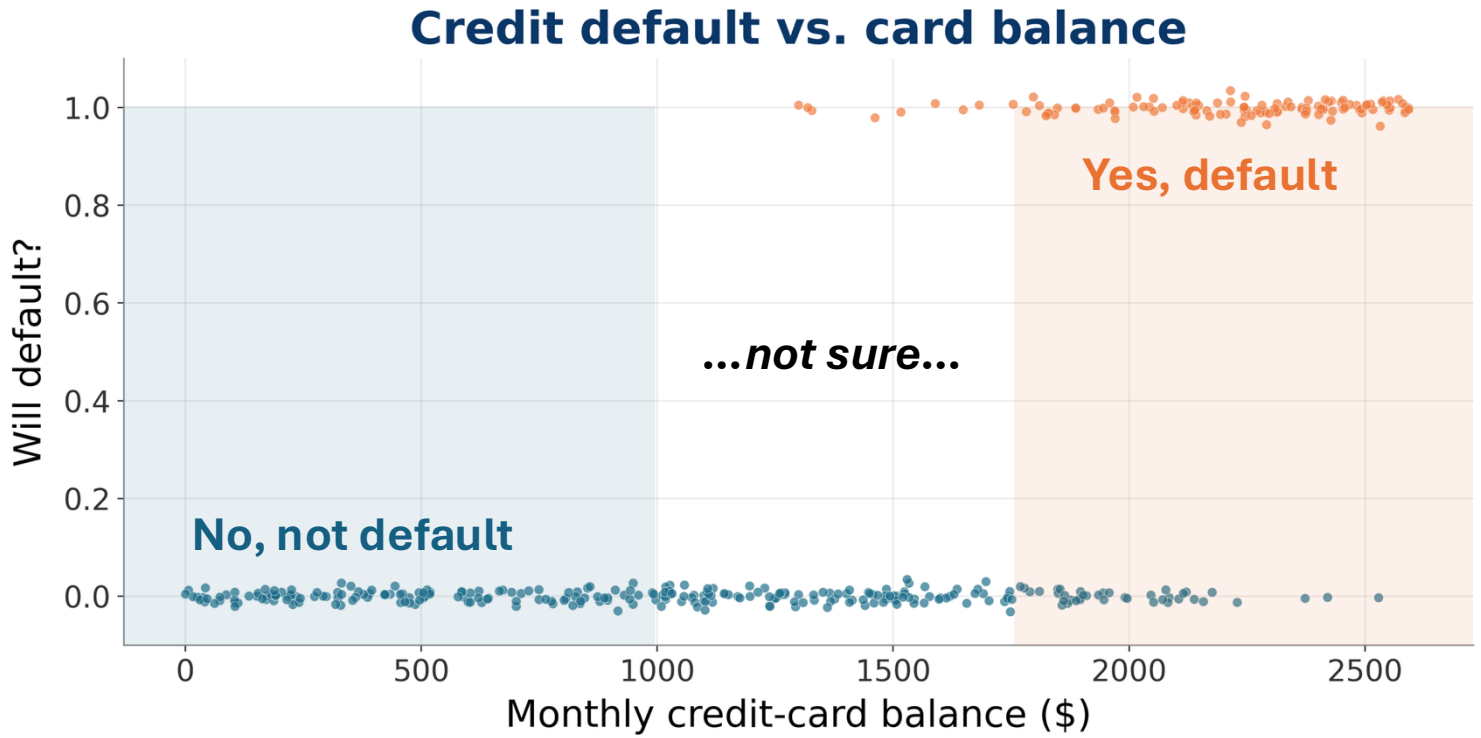
$P(y = 1|x) = 0 \rightarrow$ we are certain the customer will *not* default

$P(y = 1|x) = 1 \rightarrow$ we are certain the customer *will* default

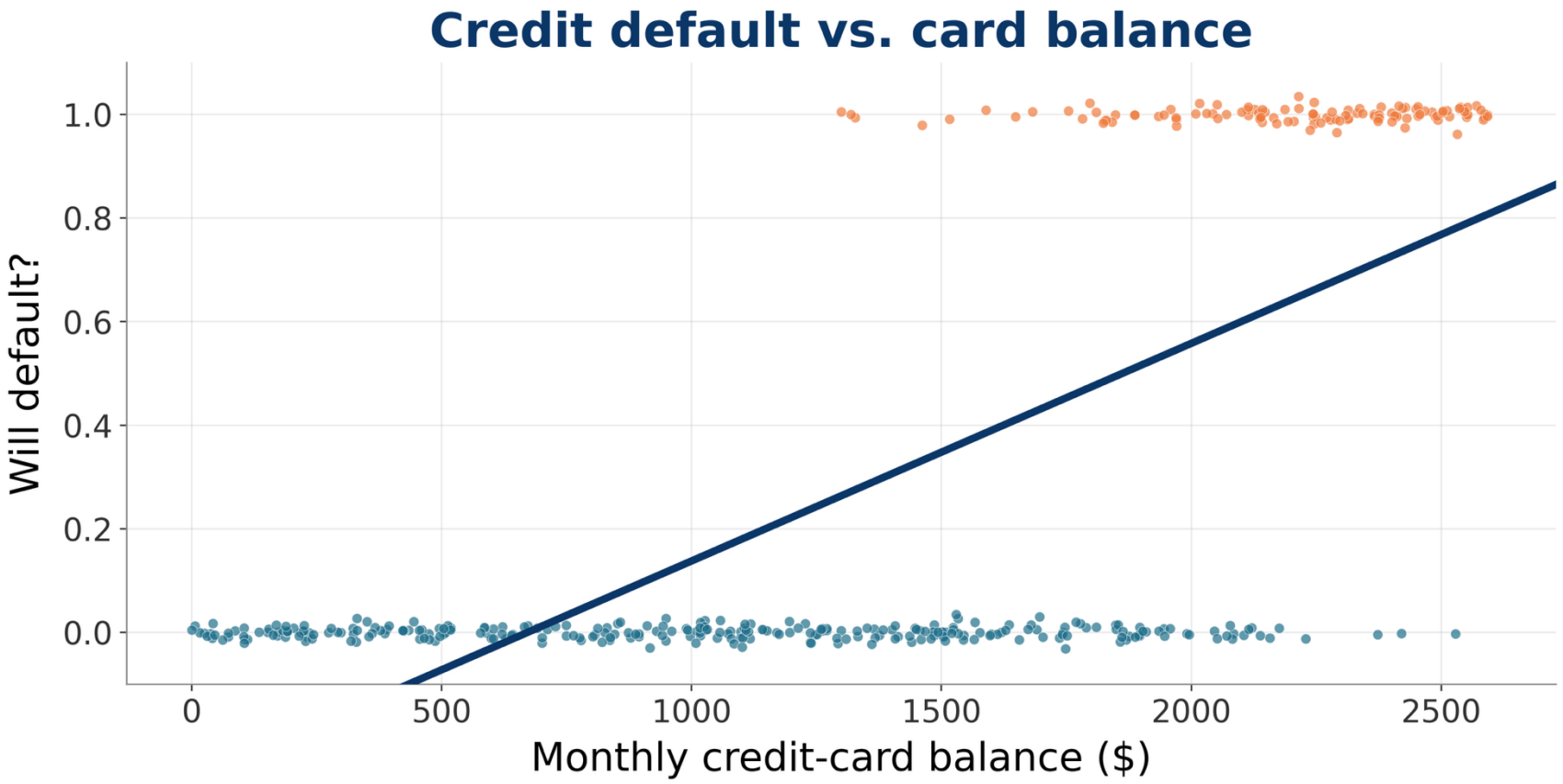
$P(y = 1|x) = = 0.5 \rightarrow$ we are not sure…

EX: Based on their monthly credit balance, will a customer default?

Monthly balance	Defaulted
\$512	No ($y = 0$)
\$1020	No ($y = 0$)
…	…
\$2345	Yes ($y = 1$)



Linear regression is not a great choice

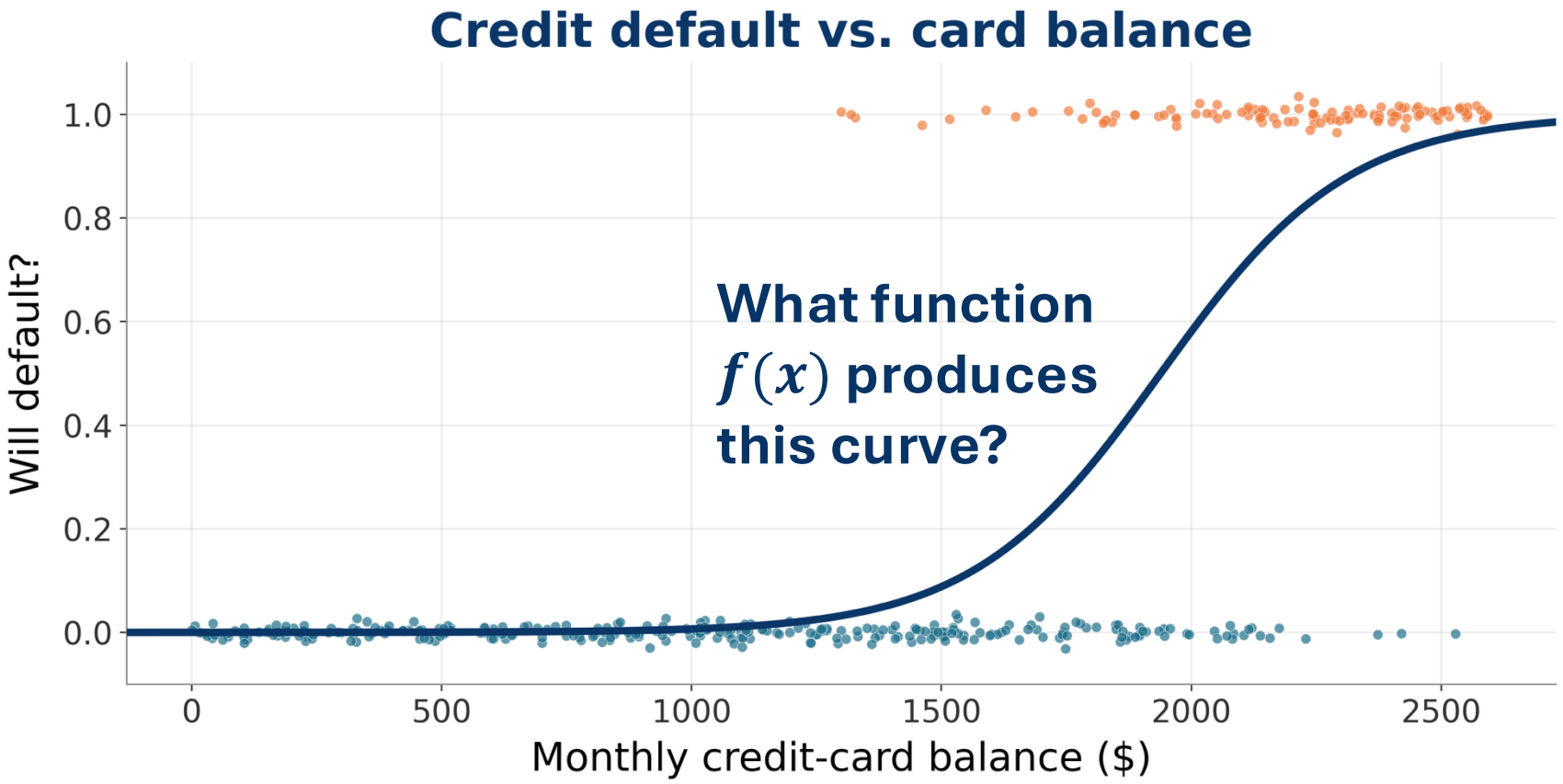


Suppose we use linear regression to solve this problem...

$$y = \beta_0 + \beta_1 x$$

1. It produces **y-values outside the $[0, 1]$ range**...Not meaningful as a probability
2. We generally expect a smooth but curved change in probability as x varies (not linear)

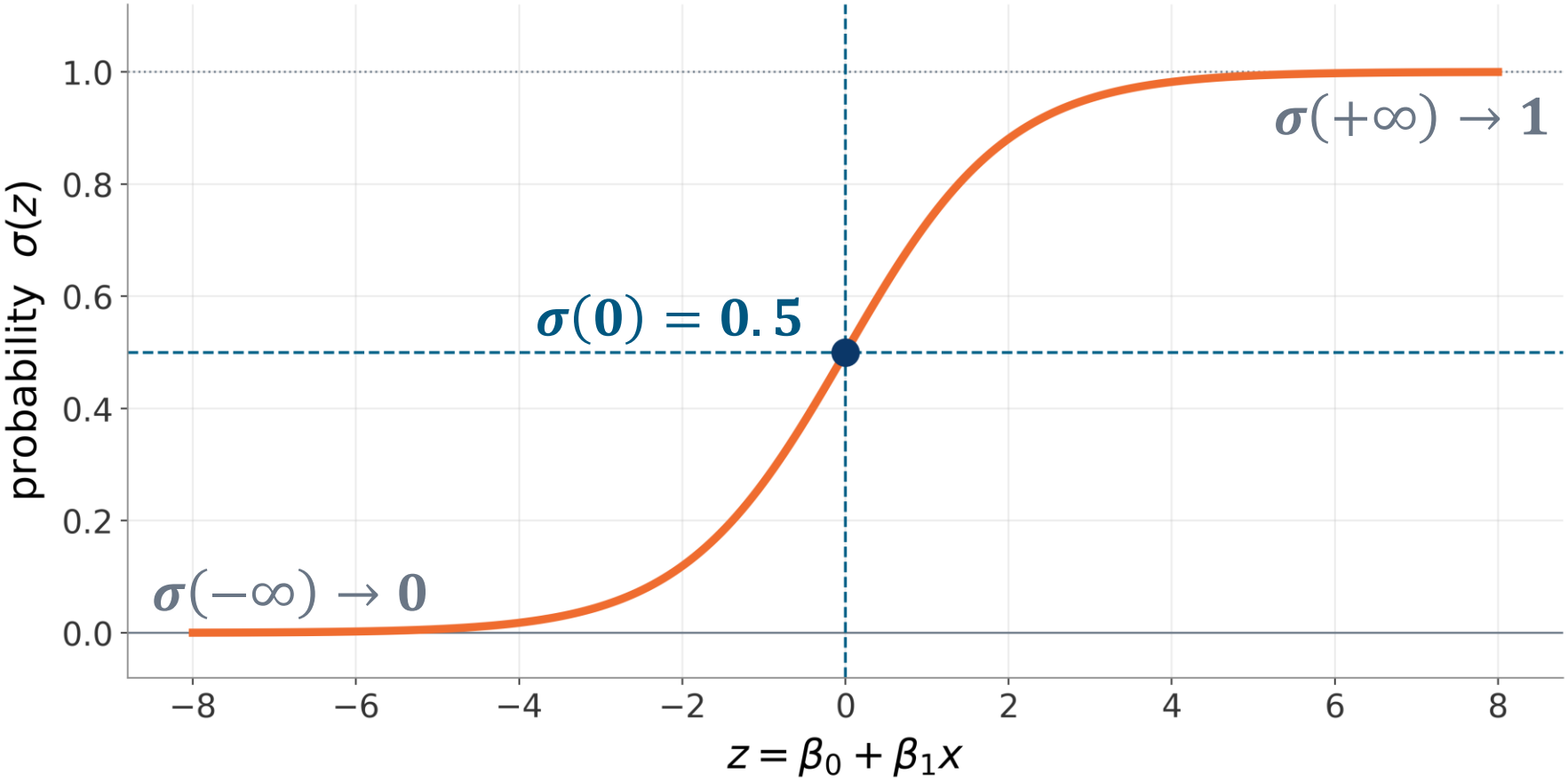
Logistic regression is the right tool for this



Now solved using **logistic regression**

1. It produces **y-values within the $[0, 1]$ range**...These are **meaningful probabilities**
2. It produced an **"S-shaped" curve** that better reflect the real trend in probability

This is based on the logistic (sigmoid) function



Sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

This gives rise to the S-shaped curve on the left

Some nice properties of the sigmoid function:

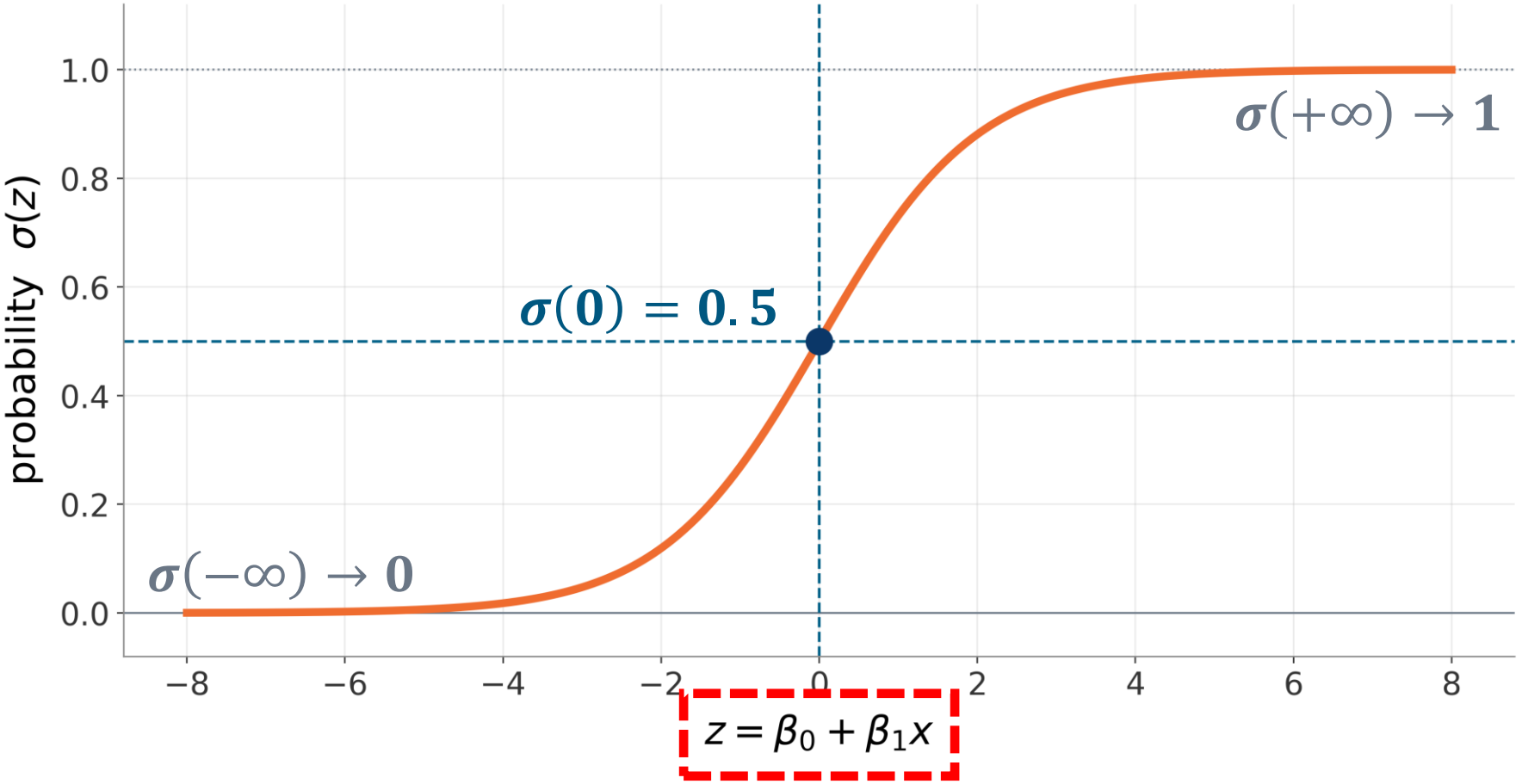
$z \in (-\infty, \infty) \rightarrow \sigma(z) \in (0, 1)$

σ is monotonically increasing

Flat tails near 0 and 1

Nice derivative: $\sigma'(z) = \sigma(z)[1 - \sigma(z)]$

This is based on the logistic (sigmoid) function



Sigmoid function:

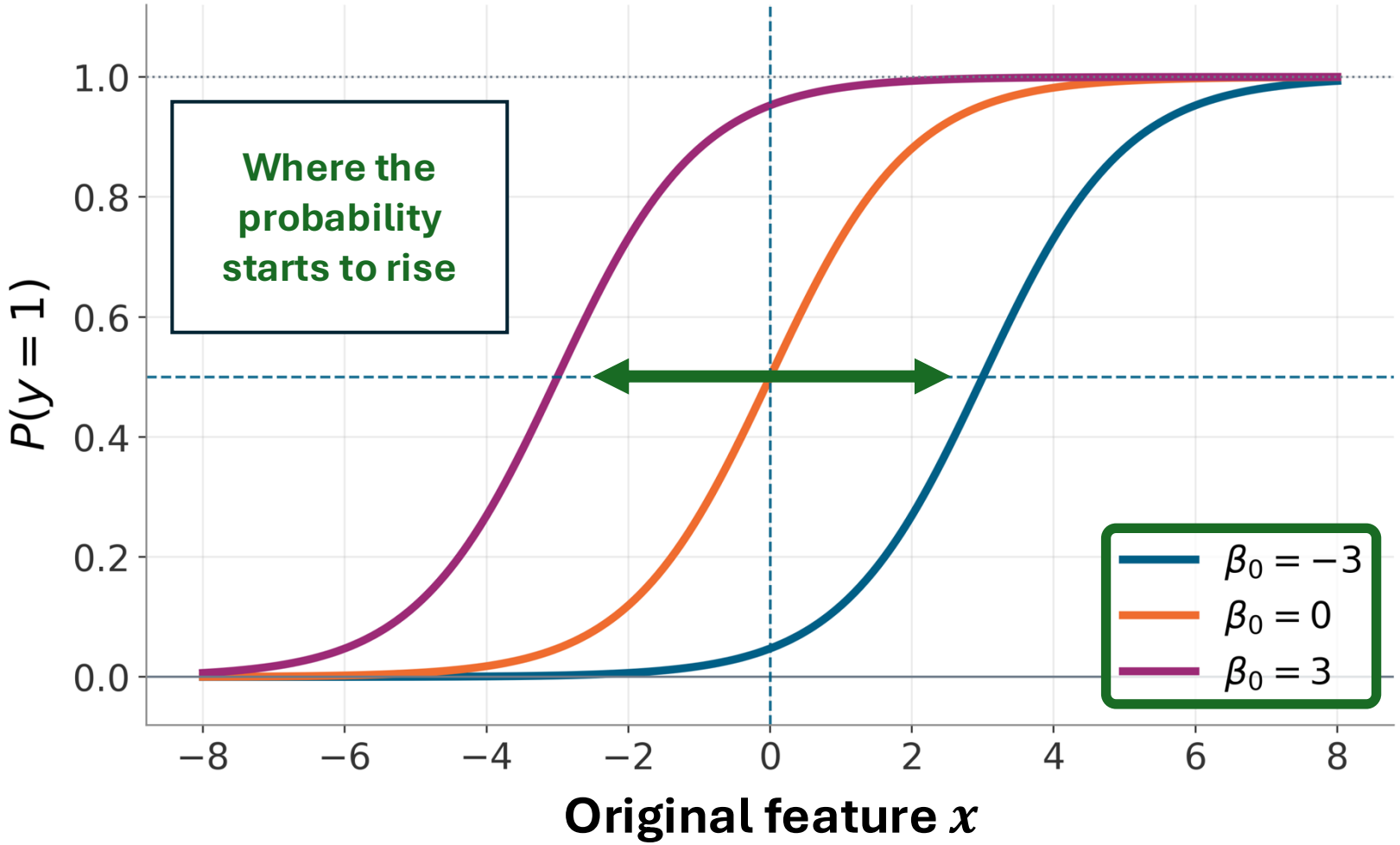
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

We don't modify sigmoid itself. We modify what gets plugged into it.

These provide the tunable parameters to modify our curve and fit the data!

Plug this into the sigmoid function

How β_i controls the sigmoid curve



Sigmoid rewritten:

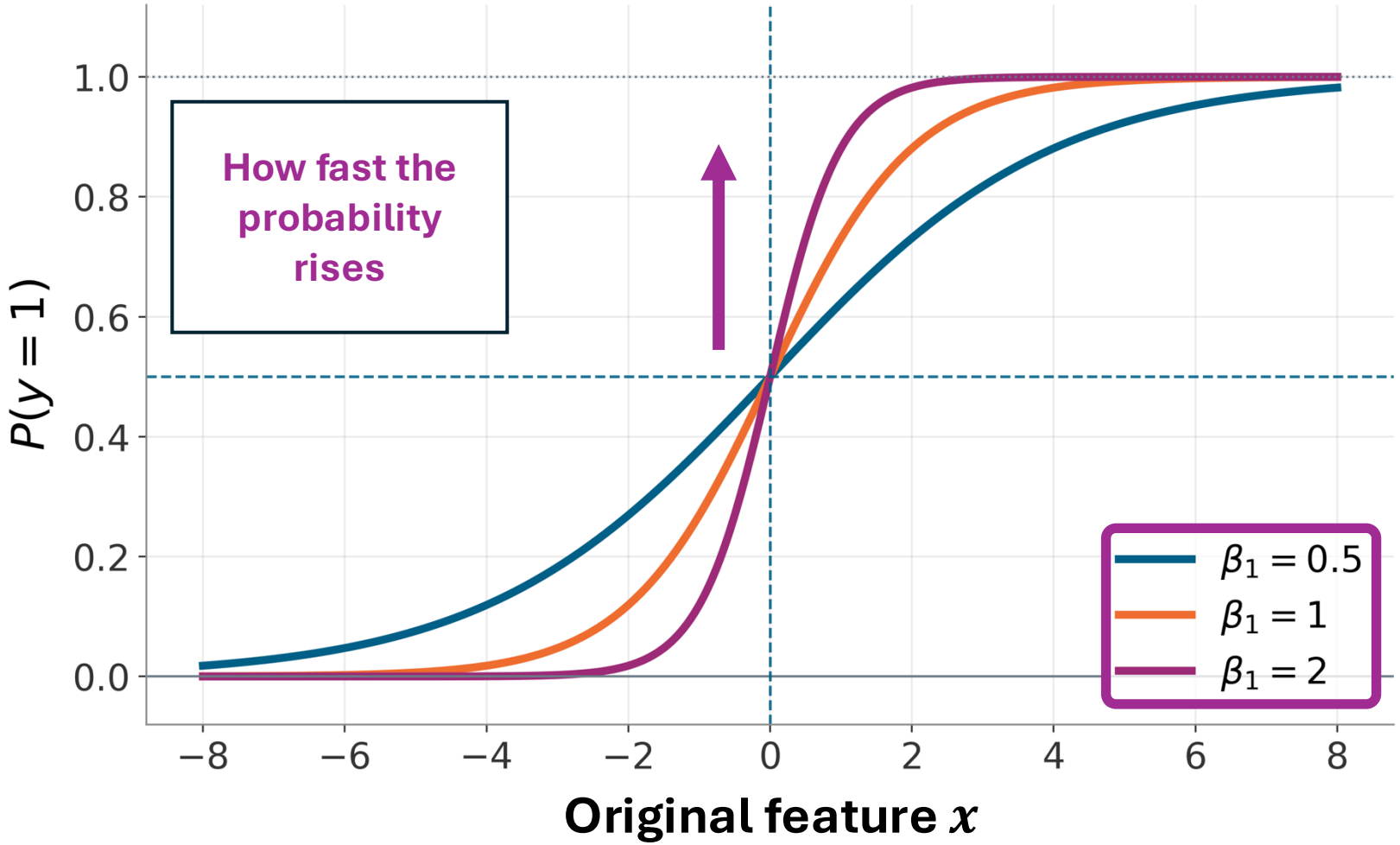
$$\sigma(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

This sigmoid is rewritten as **a function of x, not z**

β_0 shifts the *x*-value where $\sigma = 0.5$

β_0 controls the position (offset) of the curve along the *x*-axis

How β_i controls the sigmoid curve



Sigmoid rewritten:

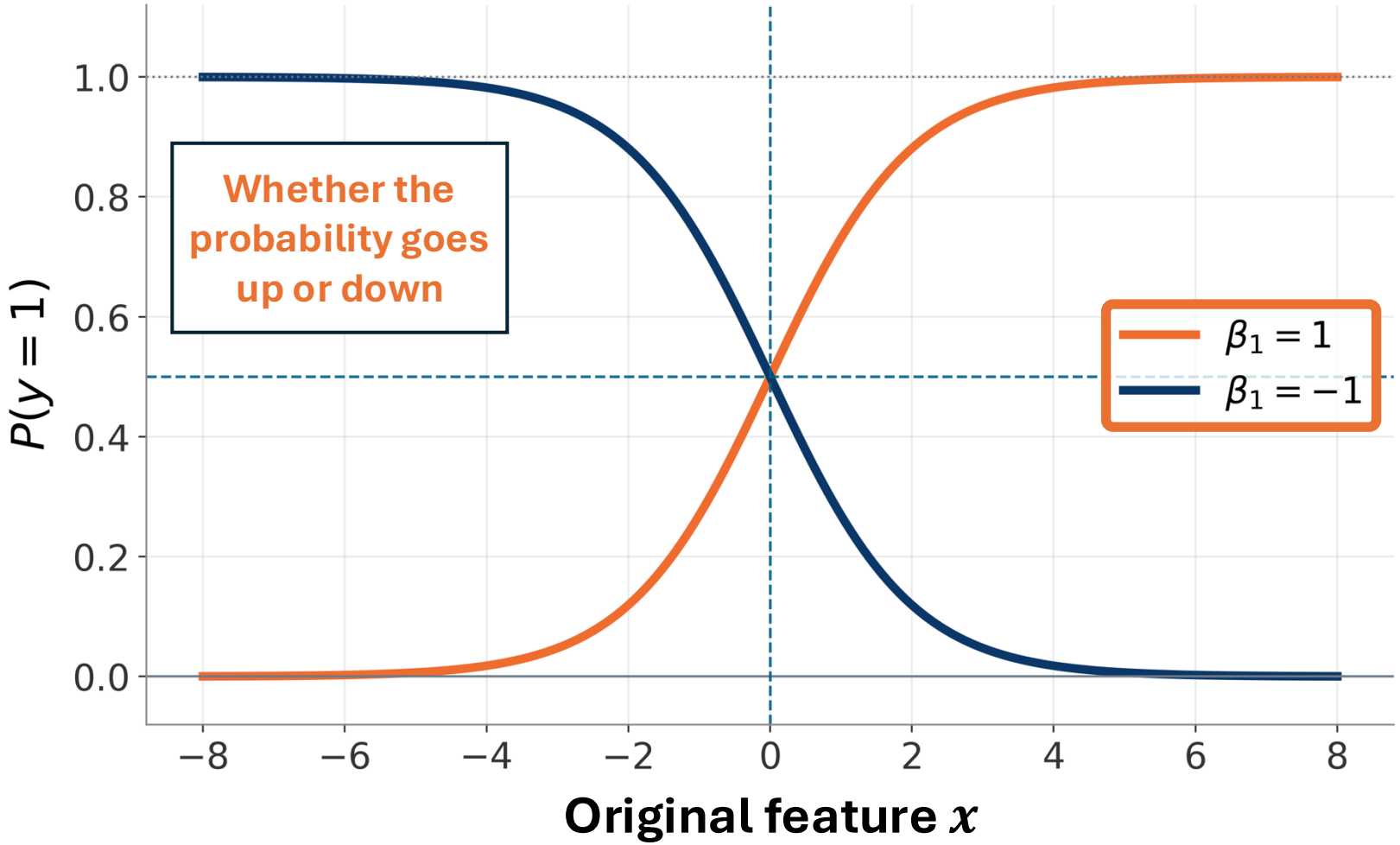
$$\sigma(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

This sigmoid is rewritten as **a function of x, not z**

Larger β_1 leads to a steeper rise in σ

β_1 controls the "steepness" of the curve – how fast it rises

How β_i controls the sigmoid curve



Sigmoid rewritten:

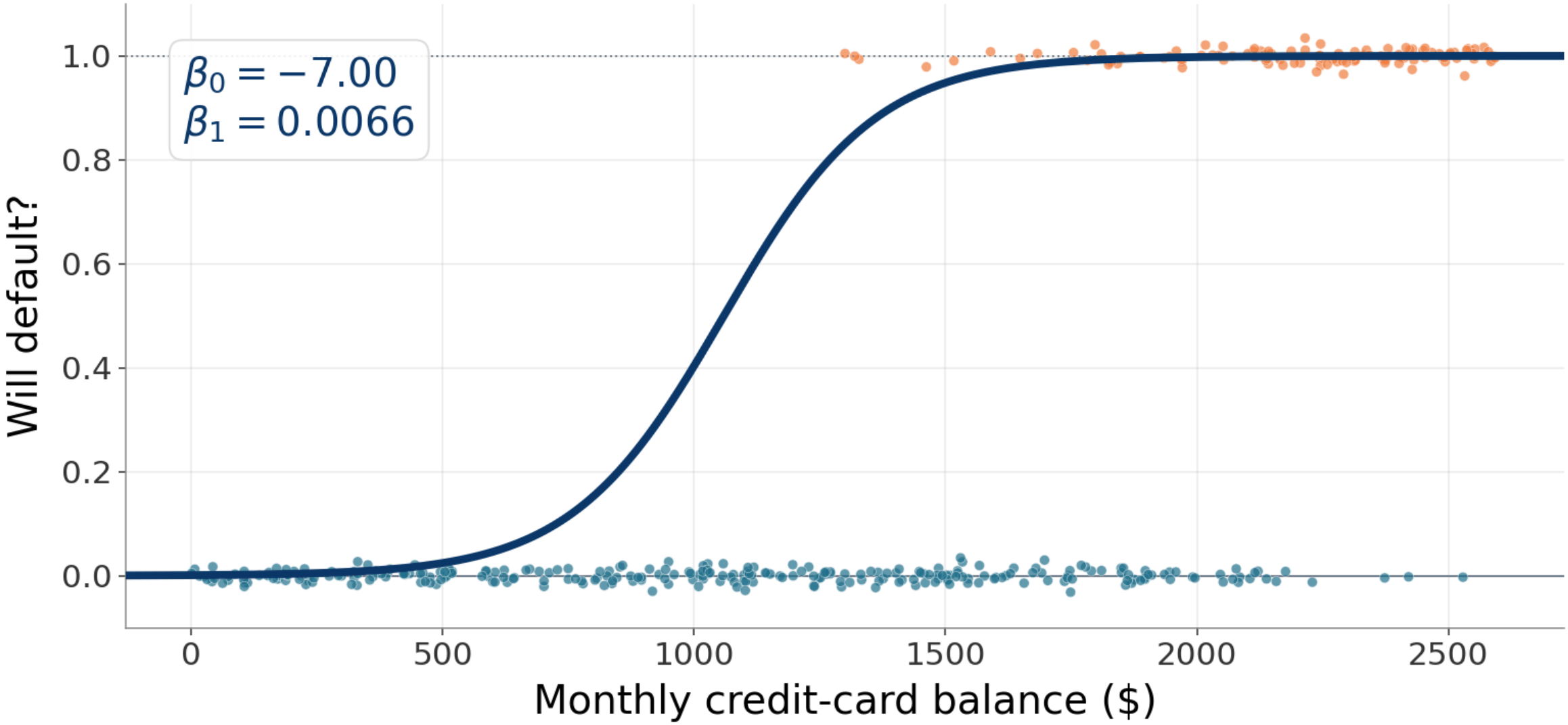
$$\sigma(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

This sigmoid is rewritten as **a function of x, not z**

Negative β_1 causes probability to decrease with increasing x

The sign of β_1 flips the curve – whether it rises or falls with x

Logistic regression = Finding the right β_i

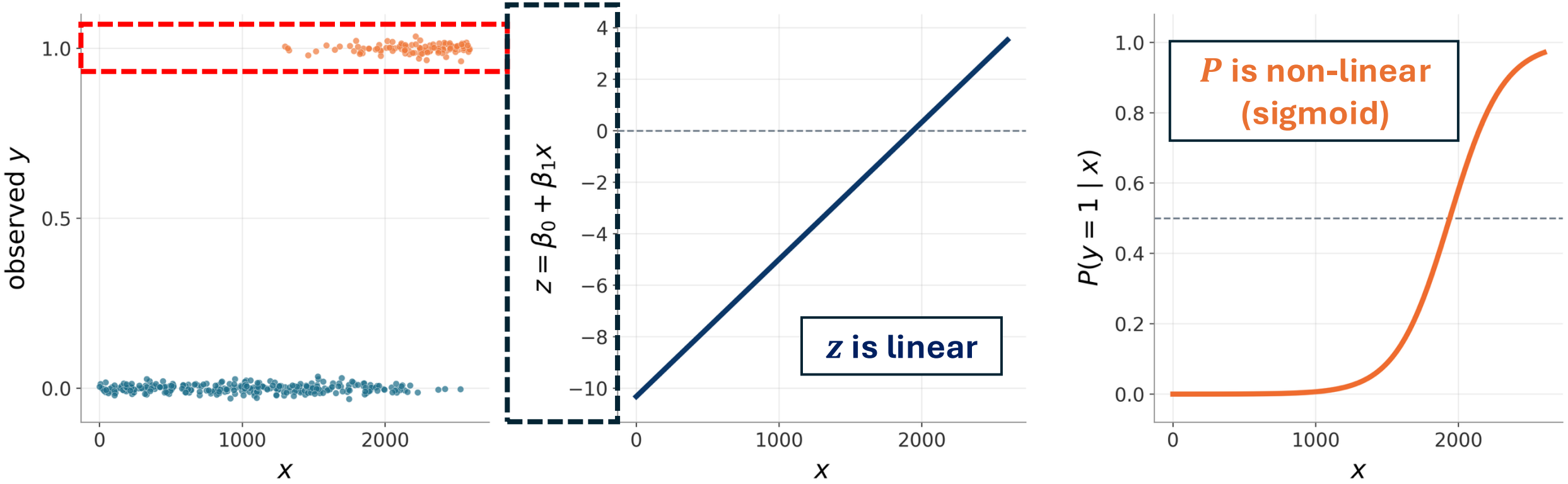


Questions?

What exactly does z represent?

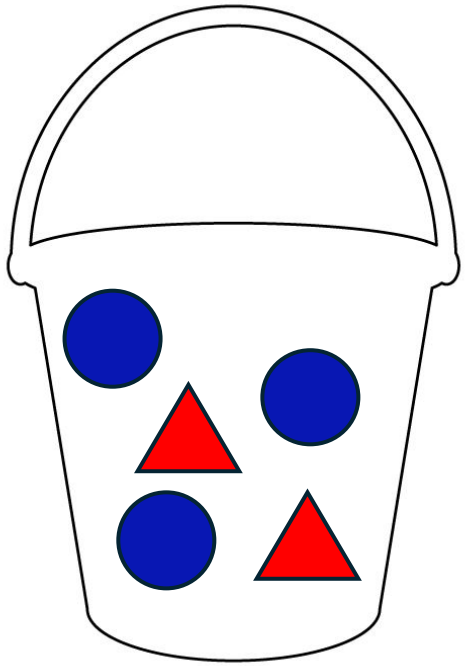
$$z = \beta_0 + \beta_1 x$$

Each step in logistic regression



- We are *not* fitting a line directly to the binary 0/1 data. We choose the coefficients β_i so that the resulting sigmoid probabilities fit the observed class labels well.
- **Clearly, $z = \beta_0 + \beta_1 x$ is not probability…Instead, it is the log-odds of $y = 1$**

What are “odds” in the first place?



We randomly choose from a bucket with:

- **3 blue circles** and **2 red triangles**

→ **The probability of choosing a blue circle is $P = 3/5 = 0.6$**

→ **The probability of choosing a red triangle is $P = 2/5 = 0.4$**

How can we re-frame this in terms of odds?

→ **The odds of choosing a blue circle are 3 to 2 … (1.5)**

→ **The odds of choosing a red triangle are 2 to 3 … ($2/3 \approx 0.66$)**

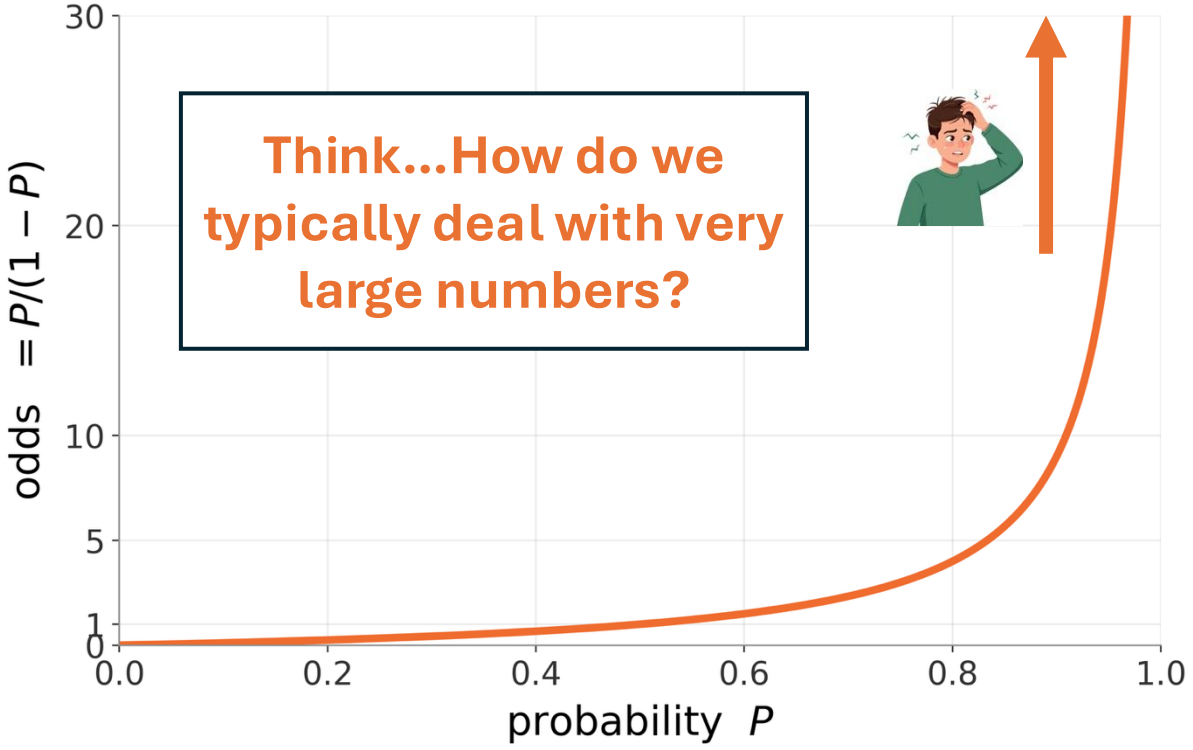
Mathematically, we can relate **the odds to the probabilities:**

$$\textbf{odds} = \frac{P}{1 - P}$$

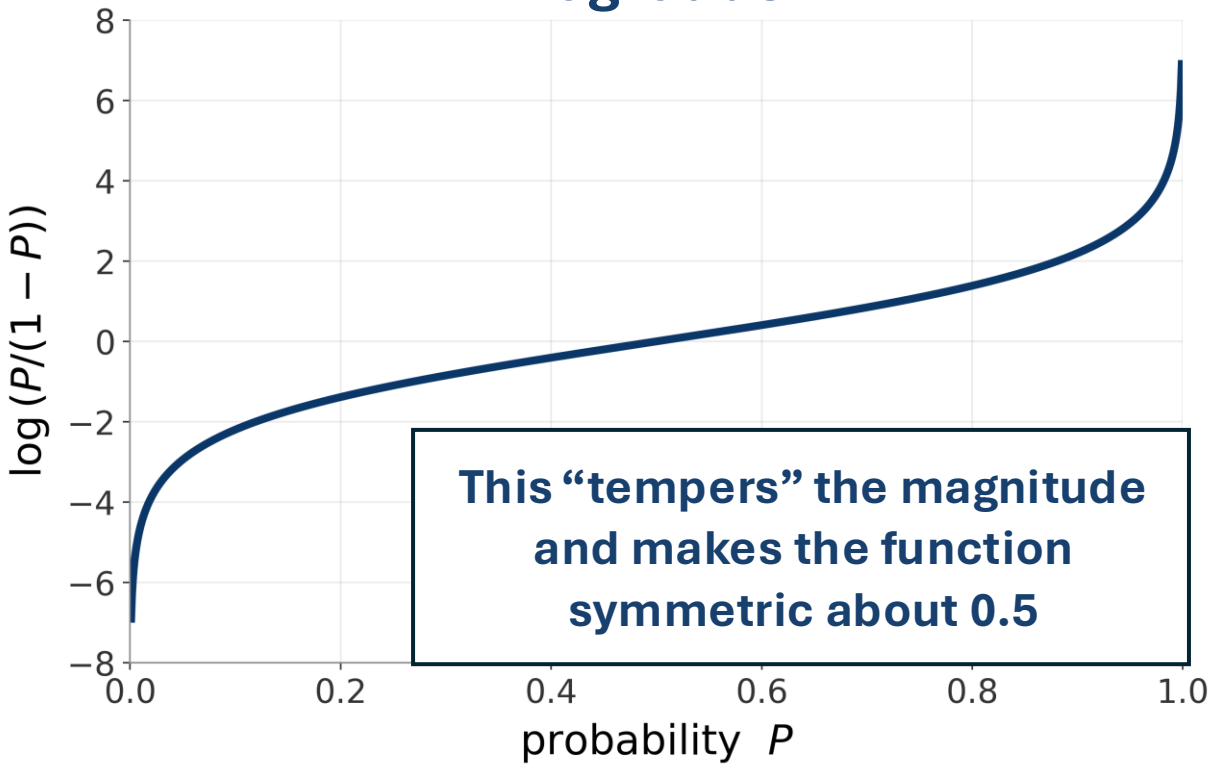
Note that odds can range anywhere from 0 to ∞

Odds grow very large as $P \to 1$

Odds



Log-odds



$$\frac{P}{1-P} \in (0, \infty) \quad \textbf{vs.} \quad \beta_0 + \beta_1 x \in (-\infty, \infty)$$

$$\log\left(\frac{P}{1-P}\right) \in (-\infty, \infty)$$

It's not practical to model odds directly

But we *can* model the log-odds directly

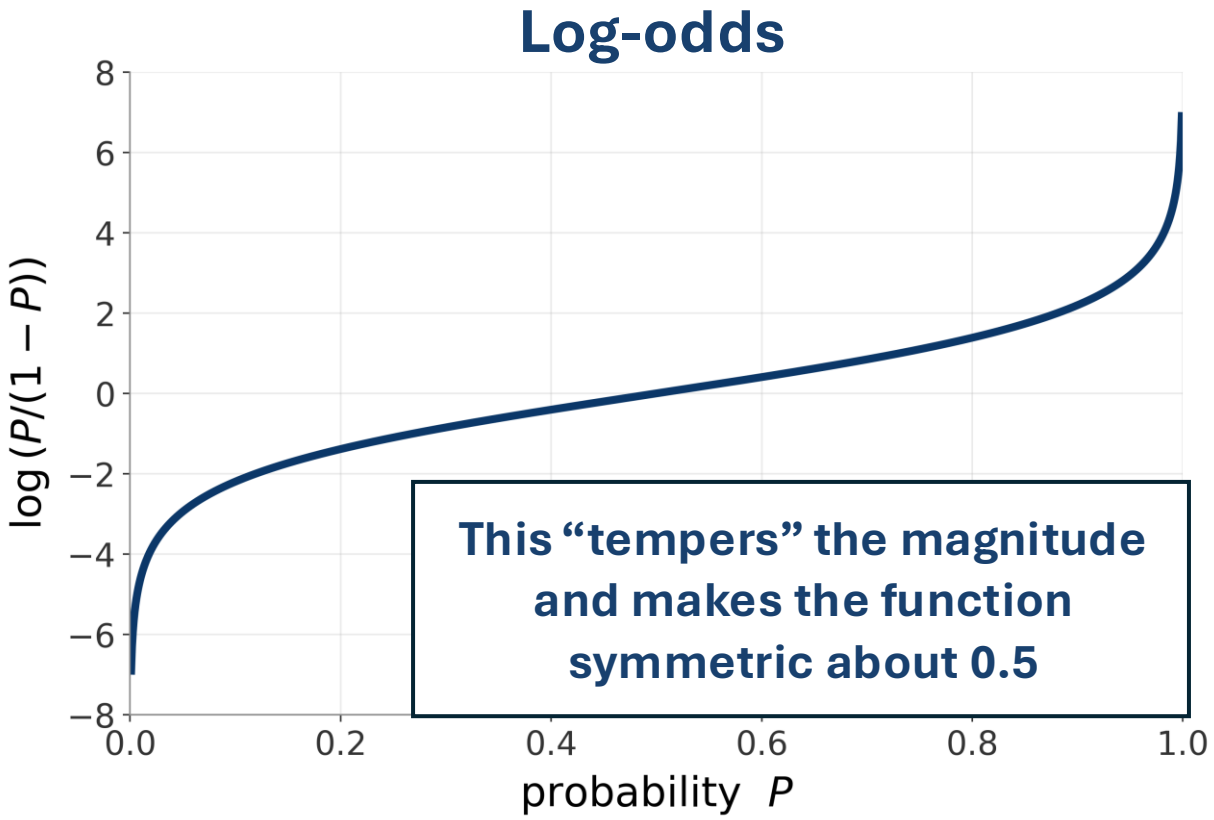
The logistic unit (logit)

A **logit** is what we call the function that produces the log-odds from a raw probability: $\mathbf{logit}(P)$

Logistic regression models the log-odds by **setting the logit equal to a linear function of the features:**

$$\mathbf{logit}(P) = \log\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x + \cdots$$

Do you see how the sigmoid function comes into play? (log ↔ exp)



$$\log\left(\frac{P}{1-P}\right) \in (-\infty, \infty)$$

But we *can* model the log-odds directly

The logistic unit (logit) and the sigmoid function

A **logit** is a function that maps a probability P (from 0 to 1) into a continuous, unbounded value: the **log-odds** (from $-\infty$ to $+\infty$)

Logistic regression models the log-odds by **setting the logit equal to a linear function of the features:**

$$\text{logit}(P) = \log\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x + \cdots$$

Do you see how the sigmoid function comes into play?

$$\log\left(\frac{P}{1-P}\right) = \underbrace{\beta_0 + \beta_1 x + \cdots}_{\text{Call this } z}$$

Logit

$$z = \log\left(\frac{P}{1-P}\right) \qquad e^z = \frac{P}{1-P}$$

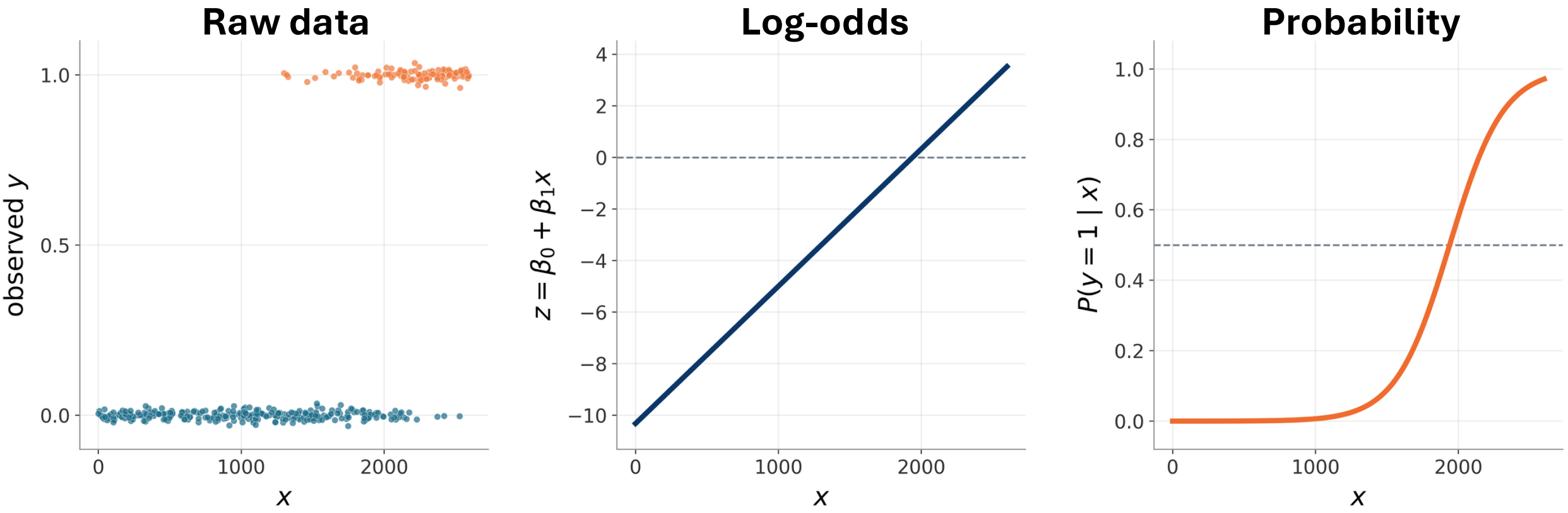
$$e^z(1-P) = P \qquad e^z - e^z P = P$$

$$e^z = P + e^z P = P(1+e^z)$$

$$P = \frac{e^z}{e^z+1} = \frac{1}{1+e^{-z}}$$

Sigmoid!

Putting it all together for logistic regression



The probability is NOT a linear function of the features x, but the log-odds are.

This is the basis of logistic regression! Now it's just a matter of calculating β_i

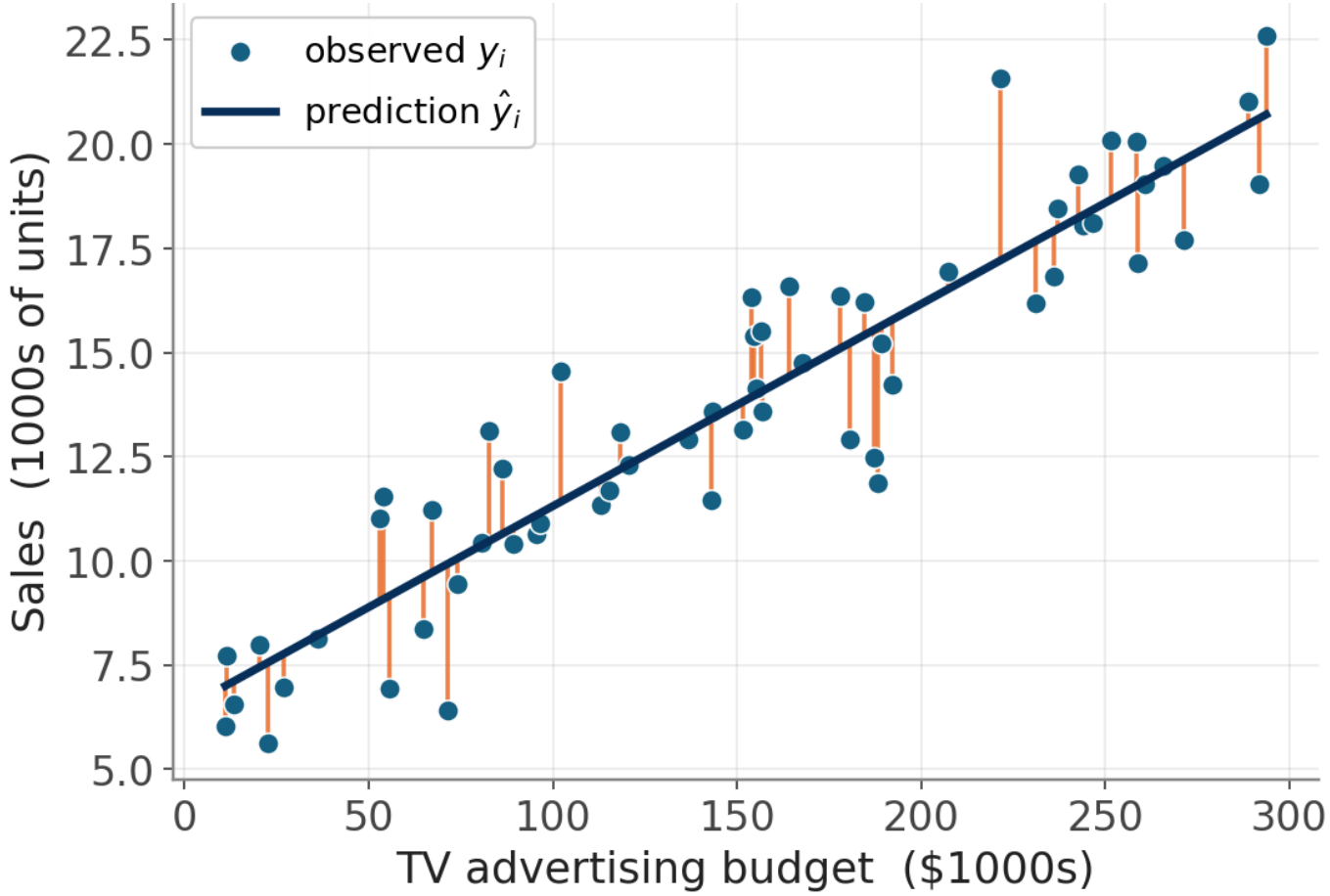
Fitting the Model

How to find the coefficients (β_i) that produce a desired probability curve

Recall: Solving β_i for linear regression

$$\hat{y} = \beta_0 + \beta_1 x$$

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2$$



Minimize the Mean Squared Error

$$\frac{d\mathcal{L}}{d\beta_0} = 0$$

$$\frac{d\mathcal{L}}{d\beta_1} = 0$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Linear regression has a nice, closed-form solution

For logistic regression, we have two challenges

Challenge #1: The predictions are non-linear in the coefficients, and we **cannot** solve this analytically by setting the derivatives equal to zero

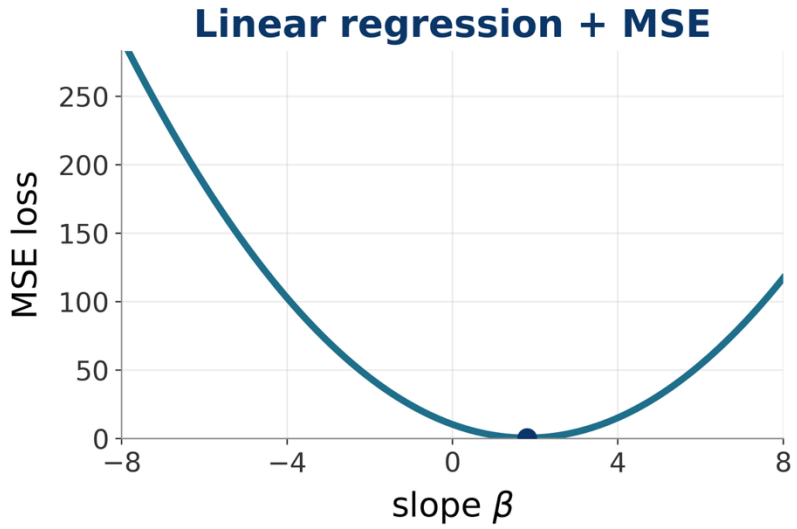
$$P(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

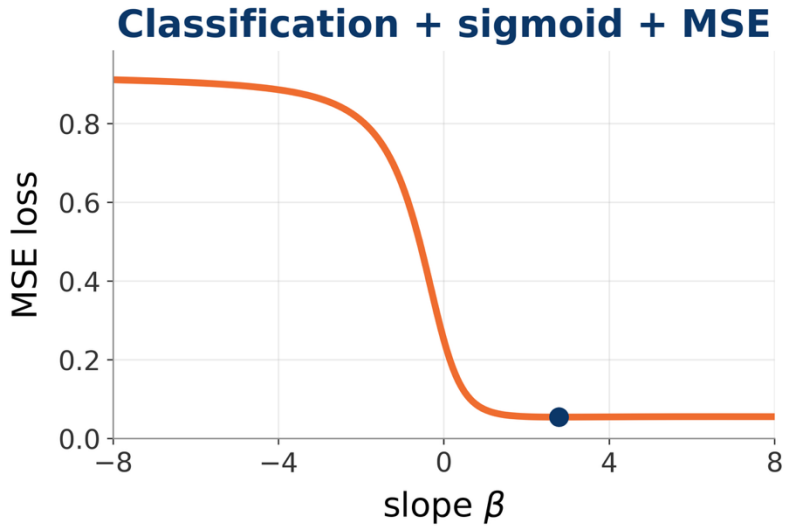
$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\left[y_i - \left(\frac{1}{1+e^{-(\beta_0+\beta_1 x)}}\right)\right]^2$$



We are now **non-linear in the coefficients!**

Challenge #2: The mean squared error is **not** the best loss for classification

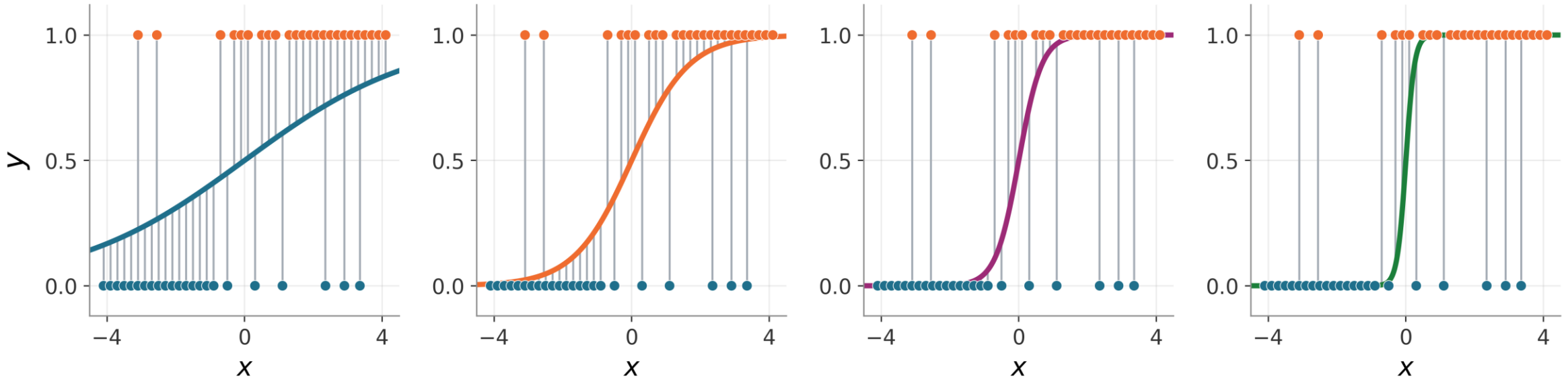


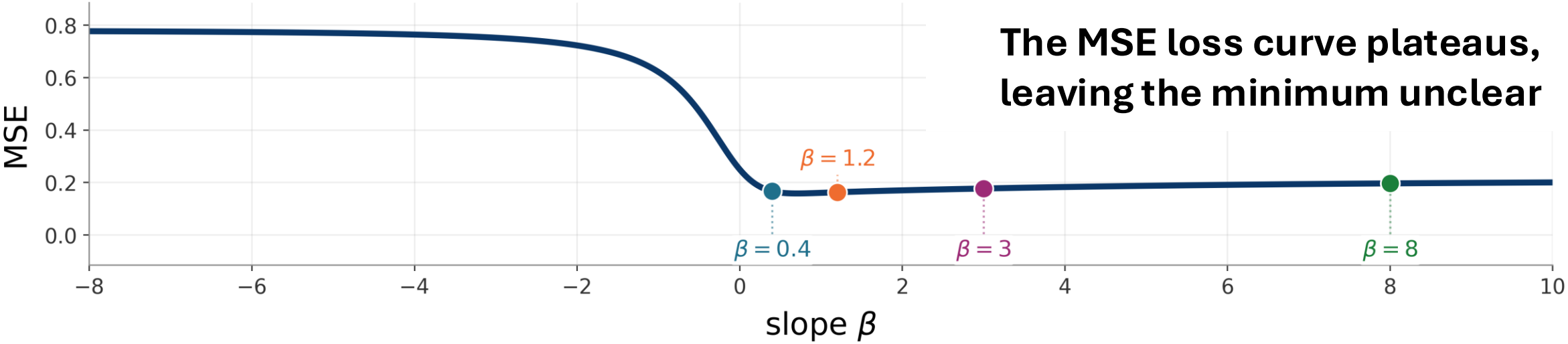


See the right panel:

MSE(β) is a non-convex loss function for logistic regression

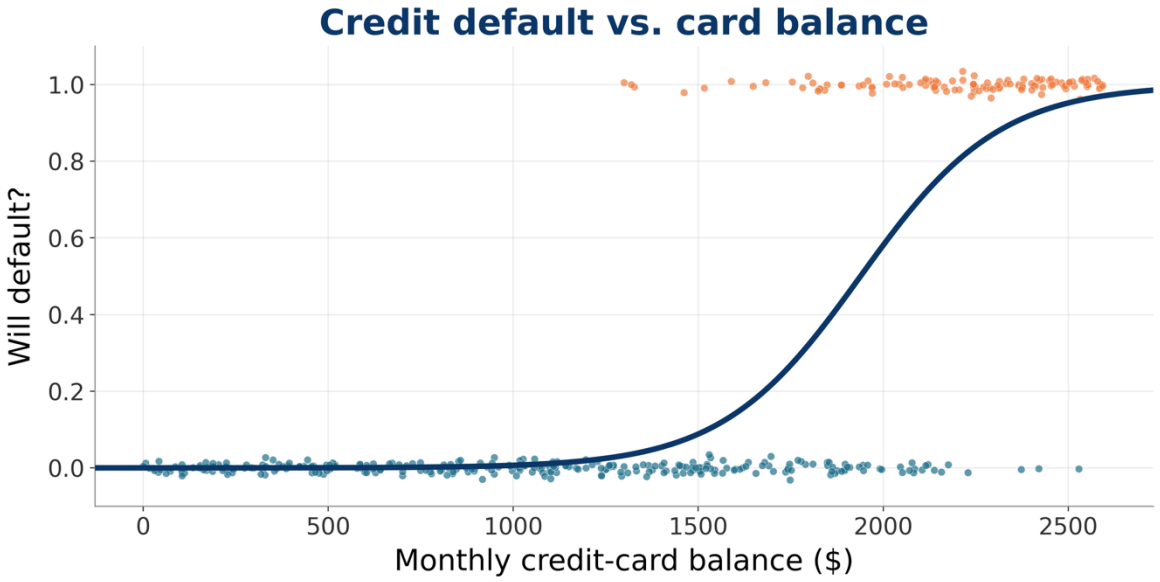
Why the MSE loss is non-convex





The MSE loss curve plateaus, leaving the minimum unclear

Building a loss for logistic regression



Logistic regression predicts the **probability** that an observation i belongs to class 1:

$$\boldsymbol{P_i(y = 1|x_i) = \sigma(\beta_0 + \beta_1 x_i)}$$

If the label $y_i = 1$, then **we want P_i large**

If the label $y_i = 0$, then **we want P_i small**

Probabilities must sum to 1. This means that for binary classification, we can write:

Actual label	Probability assigned to outcome
$y_i = 1$	P_i
$y_i = 0$	$1 - P_i$

We combine these into <u>one expression</u>:

$$\boldsymbol{P(y_i|x_i) = P_i^{y_i}(1 - P_i)^{1-y_i}}$$

Case 1: $\boldsymbol{P(y_i = 1|x_i) = P_i^1(1 - P_i)^0 = P_i}$

Case 2: $\boldsymbol{P(y_i = 0|x_i) = P_i^0(1 - P_i)^1 = 1 - P_i}$

The likelihood

The **likelihood** measures how well our coefficients explains the labels we observed.

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} p_i^{y_i}(1 - p_i)^{1-y_i}$$

A product over all observations

We **maximize the likelihood** because we want the coefficients that assign the highest probability to the labels we actually observed.

This assumes **independent observations:**

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

The likelihood is the **joint probability of observing all labels (y_i) in the dataset**, given our choice of coefficients (β)

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmax}}\, L(\boldsymbol{\beta})$$

This means are goal is to find β where $L(\beta)$ is as large as possible

The *log*-likelihood

In practice, it is easier to maximize the **log likelihood…**

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}$$

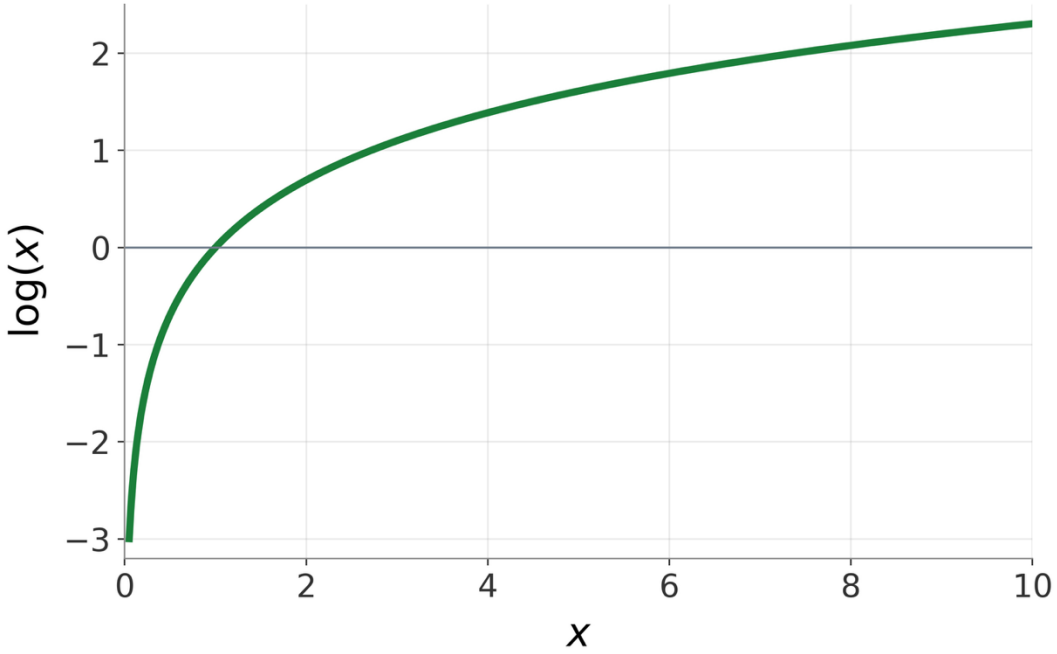
$$l(\boldsymbol{\beta}) = \log L(\boldsymbol{\beta})$$

- The logarithm is monotonically increasing, so it won't change where our maximum occurs (*i.e.*, what β we get)

$$\underset{\beta}{\operatorname{argmax}}\, L(\beta) = \underset{\beta}{\operatorname{argmax}}\, l(\beta)$$

- Logarithms turn products into sums, which are easier to differentiate

$$\log(AB) = \log A + \log B$$



The *log*-likelihood

In practice, it is easier to maximize the **log likelihood…**

$$L(\beta) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i}$$

$$l(\beta) = \log L(\beta)$$

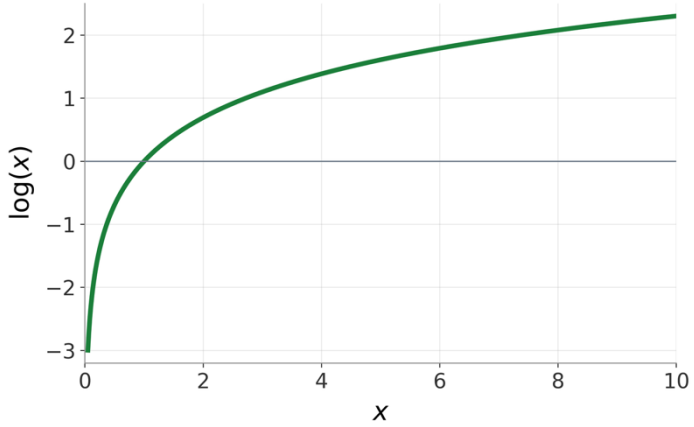
$$l(\beta) = \log \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} = \sum_{i=1}^{n} \log\left[p_i^{y_i}(1-p_i)^{1-y_i}\right]$$

$$= \sum_{i=1}^{n} \log\left(p_i^{y_i}\right) + \log(1-p_i)^{1-y_i} = \sum_{i=1}^{n} y_i \log(p_i) + (1-y_i)\log(1-p_i)$$

The log likelihood equation

The *log*-likelihood

You can see why we'd like this to be maximal.
Probabilities should align with true labels!



If $y_i = 1$, this expression is maximal when $p_i = 1$ (otherwise, its negative)

Actual label of sample i

Modeled probability of sample i

$$= \sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i)\log(1 - p_i)$$

The log likelihood equation

Cross-entropy

Machine-learning algorithms are usually framed as **minimizing a loss** rather than maximizing an objective. We also prefer to work with **normalized quantities.**
To this end, we define **the average binary cross-entropy loss:**

$$J(\boldsymbol{\beta}) = -\frac{1}{n} l(\boldsymbol{\beta}) = -\frac{1}{n} \sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

n is total # of samples

Other names for it include the ***log loss*** and the ***negative log-likelihood***

Maximizing the log-likelihood = **Minimizing the cross-entropy**

Cross-entropy: An example

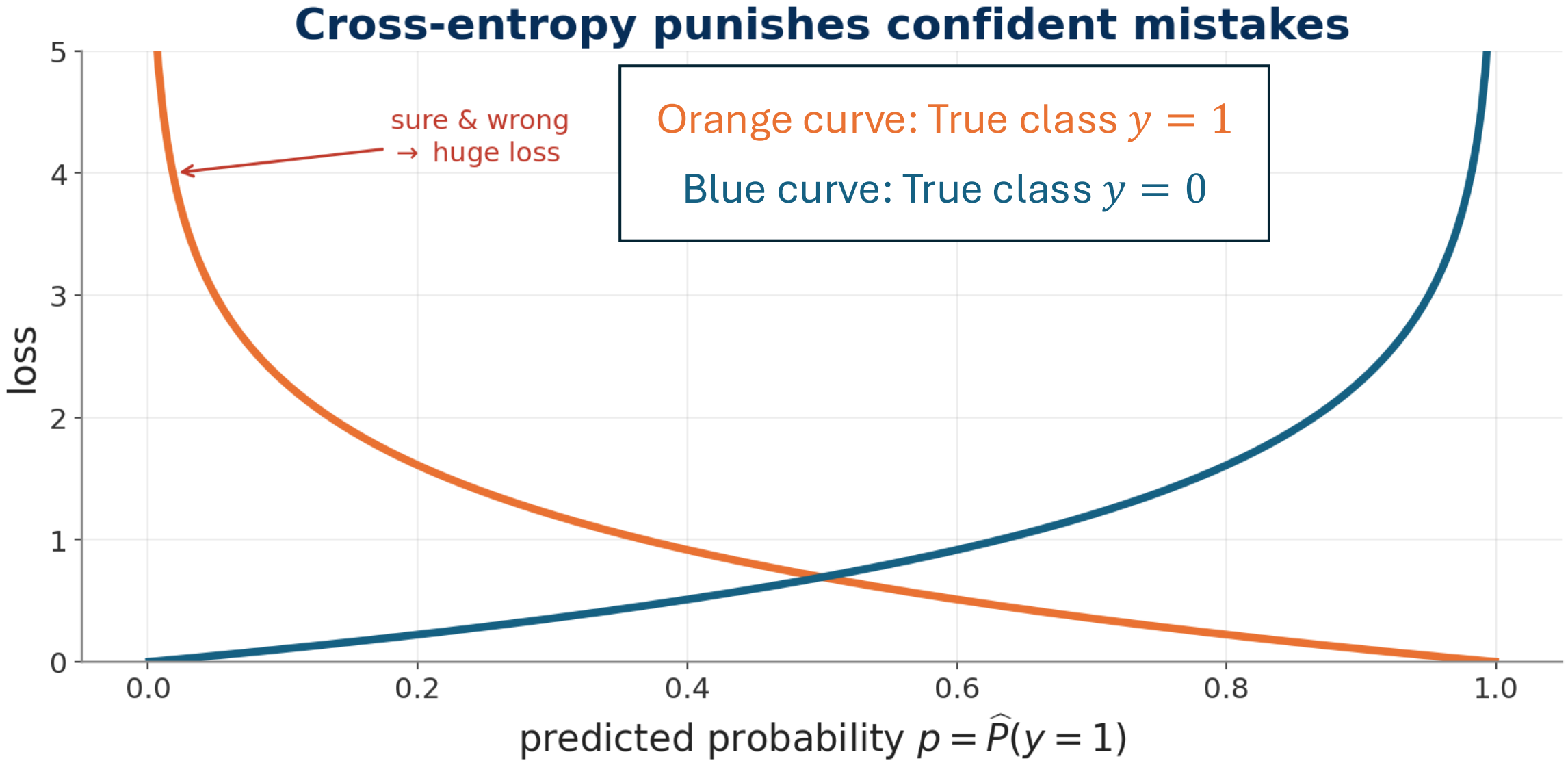
Let's consider the **binary cross-entropy of just one observation (y):**

$$J = --y\log(p) - (1-y)\log(1-p)$$

Suppose **the true label of the observation is class $y = 1$.** Then we can calculate what the loss would be for various probabilities given by our model on that sample.

Actual label	Predicted P	Cross-entropy	
$y = 1$	0.99	0.01	**This is what we want!**
$y = 1$	0.9	0.10	
$y = 1$	0.5	0.69	
$y = 1$	0.01	4.61	**This is bad...**

Cross-entropy as a plot for two classes



Relating cross-entropy to the coefficients

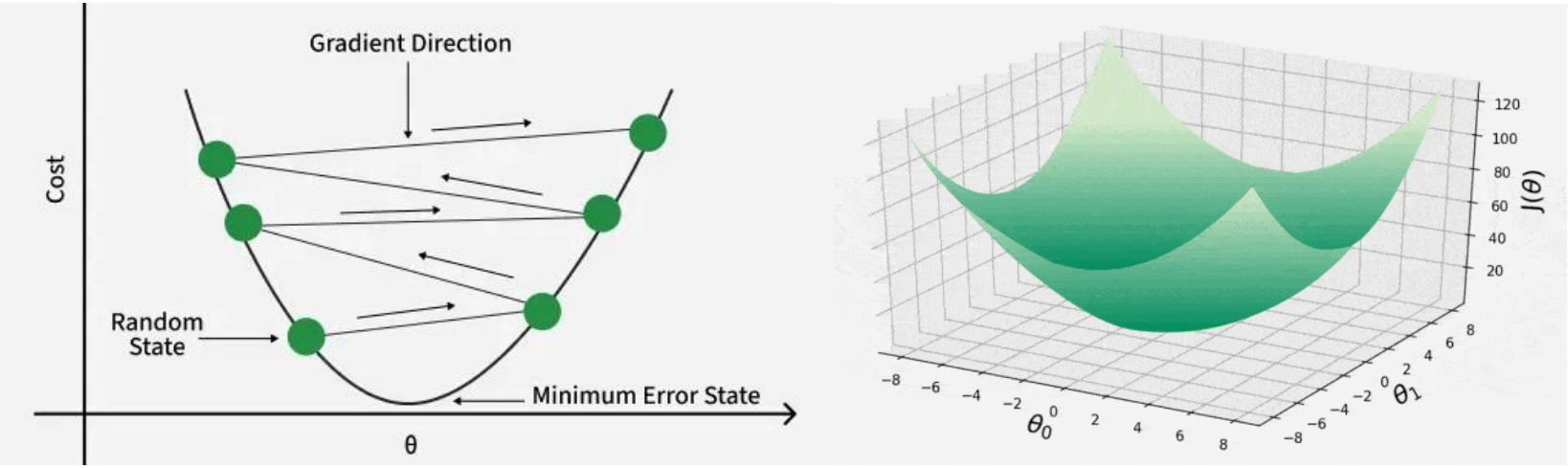
$$J(\beta) = -\frac{1}{n} l(\beta) = -\frac{1}{n} \sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

$$p_i = \sigma(\beta_0 + \beta_1 x_i)$$

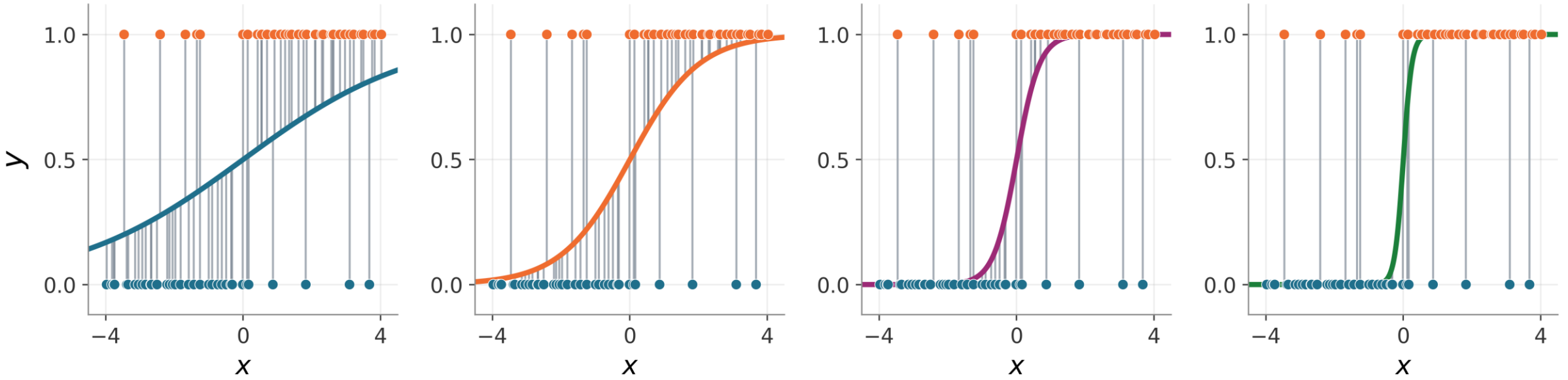
Plug in $p_i = \sigma(z)$ where $z = \beta_0 + \beta_1 x$

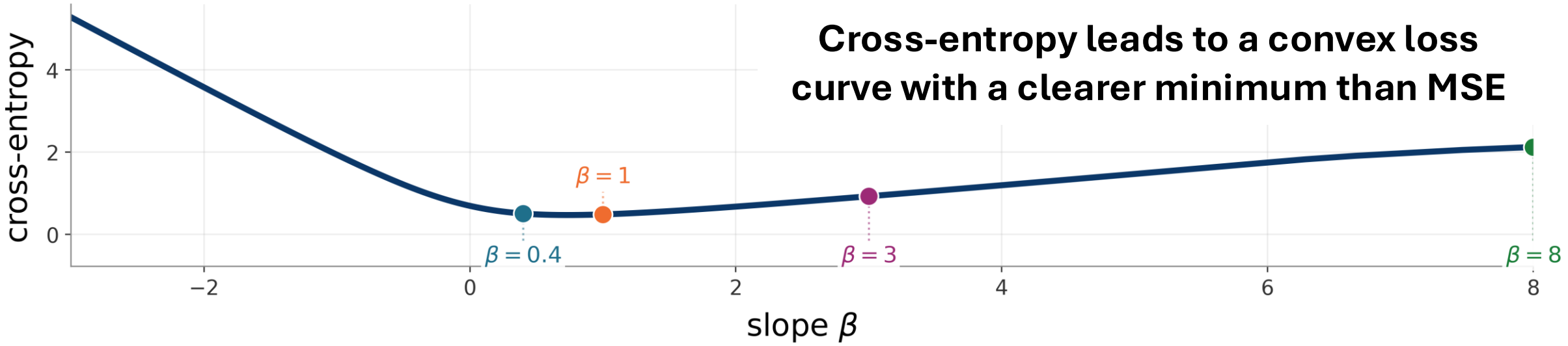
$$J(\beta) = -\frac{1}{n} l(\beta) = -\frac{1}{n} \sum_{i=1}^{n} y_i \log[\sigma(\beta_0 + \beta_1 x_i)] + (1 - y_i) \log[1 - \sigma(\beta_0 + \beta_1 x_i)]$$

We cannot solve this analytically, but we can minimize it using **gradient descent.**



Cross-entropy loss *vs.* β





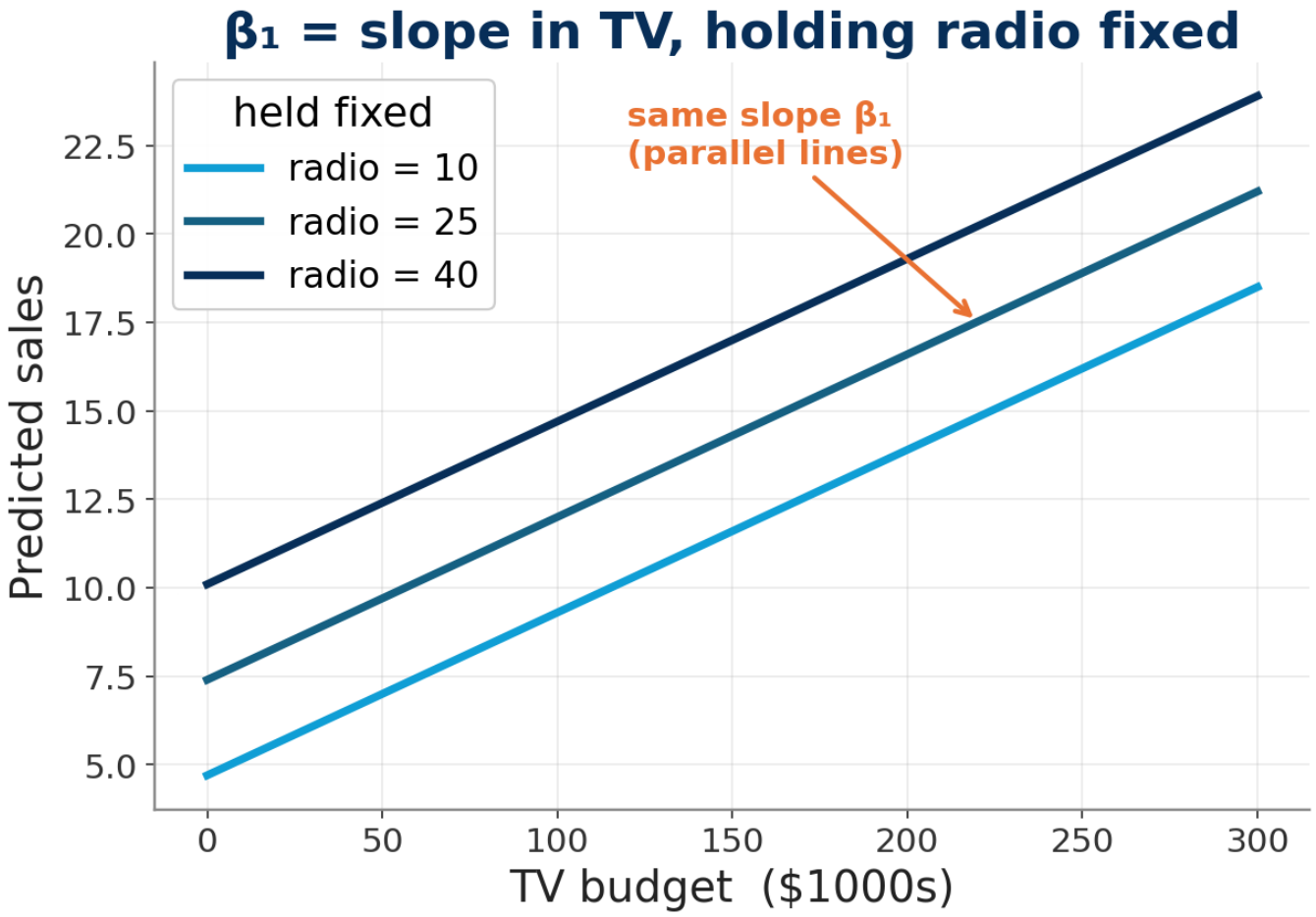
Cross-entropy leads to a convex loss curve with a clearer minimum than MSE

Questions?

Interpreting Coefficients

Recall the meaning of β_i from linear regression

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$



Each β_i represents the change in $\hat{y}$ per one-unit increase in x_i, while holding all other predictors fixed.

Larger $\beta_i \rightarrow$ feature i has a greater impact on the output.

EX: "With a fixed radio budget, how much increased sales do we get per extra $1000 spent on TV?"

Logistic regression: β_i relates to log-odds

Our coefficients are directly related to **the log-odds**, not the class probability

$$\textbf{logit}(P) = \log\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_i x_i$$

β_0 is the baseline log-odds of class 1 when all features are zero. It can be positive (if $P > 0.5$), negative ($P < 0.5$), or zero when the odds are even ($P = 0.5$)

For all others: one-unit increase in x_i changes the log-odds of class 1 by β_i

$\beta_i > 0$: increasing x_i makes class 1 **more likely**

$\beta_i < 0$: increasing x_i makes class 1 **less likely**

$\beta_i = 0$: x_i has **no effect** on the predicted odds

Log-odds are a bit difficult to interpret, so we can also exponentiate β_i to get a direct effect on the odds

From log-odds to odds

Our coefficients are directly related to **the log-odds**, not the class probability

$$\text{logit}(P) = \log\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ldots + \beta_i x_i$$

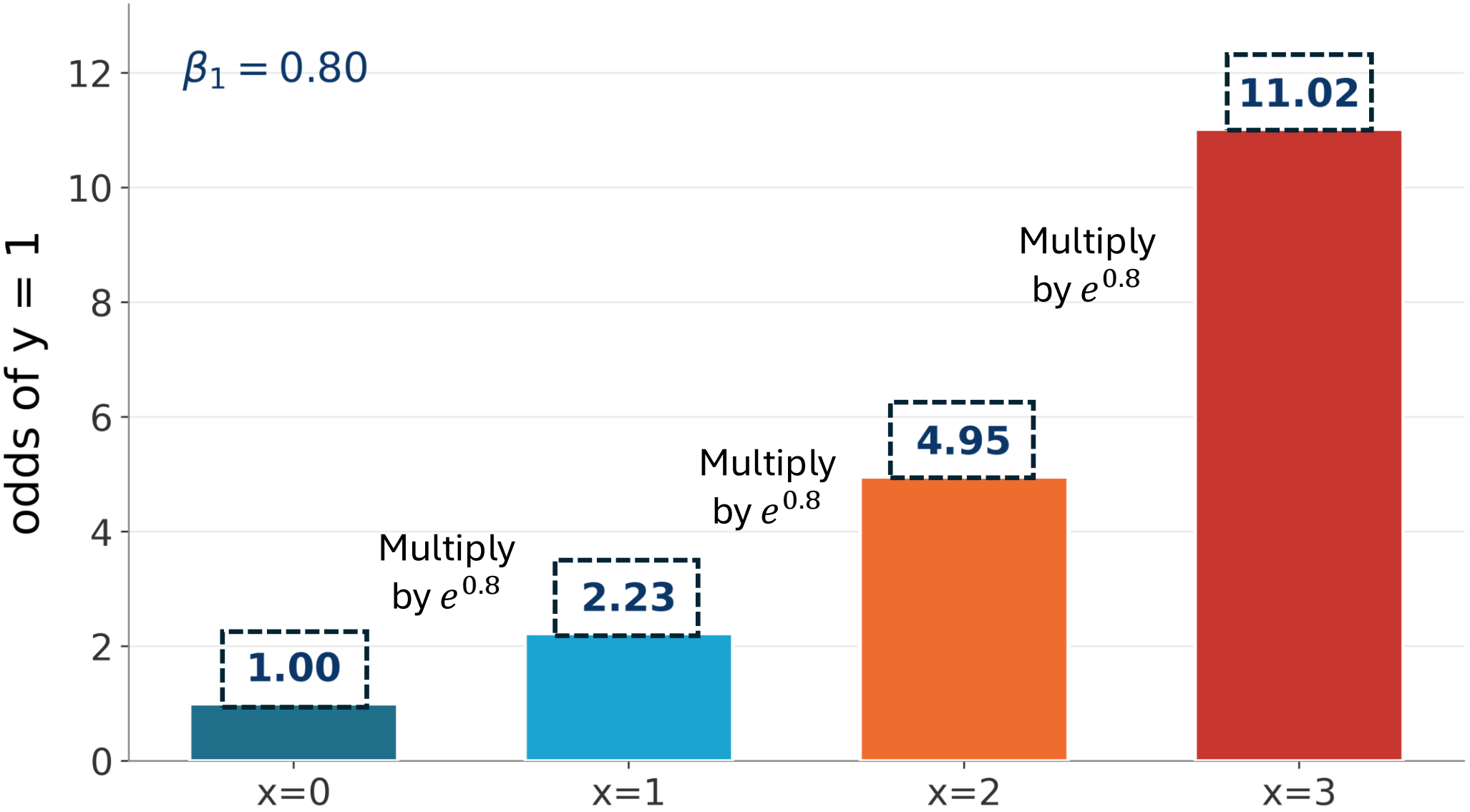
Suppose these are held constant

$$e^{\beta_0+\beta_1 x_1+\beta_2 x_2 \ldots+\beta_i x_i} = e^{\beta_0} e^{\beta_1 x_1} e^{\beta_2 x_2} \ldots e^{\beta_i x_i} = \frac{P}{1-P}$$

The odds of class 1

A one-unit increase in x_i multiplies the odds of class 1 by e^{β_i}

Relating coefficients to the odds



A one-unit increase in x_i multiplies the odds of class 1 by e^{β_i}

Not a linear increase!

A worked example with mixed variable types

Question. How likely is a student to pass a course given two predictors:

$$\log\left(\frac{P_{\text{pass}}}{1-P_{\text{pass}}}\right) = \log\left(\frac{P_{\text{pass}}}{P_{\text{fail}}}\right) = \beta_0 + \beta_1 x_{\text{hrs studied}} + \beta_2 x_{\text{prerequisite}}$$

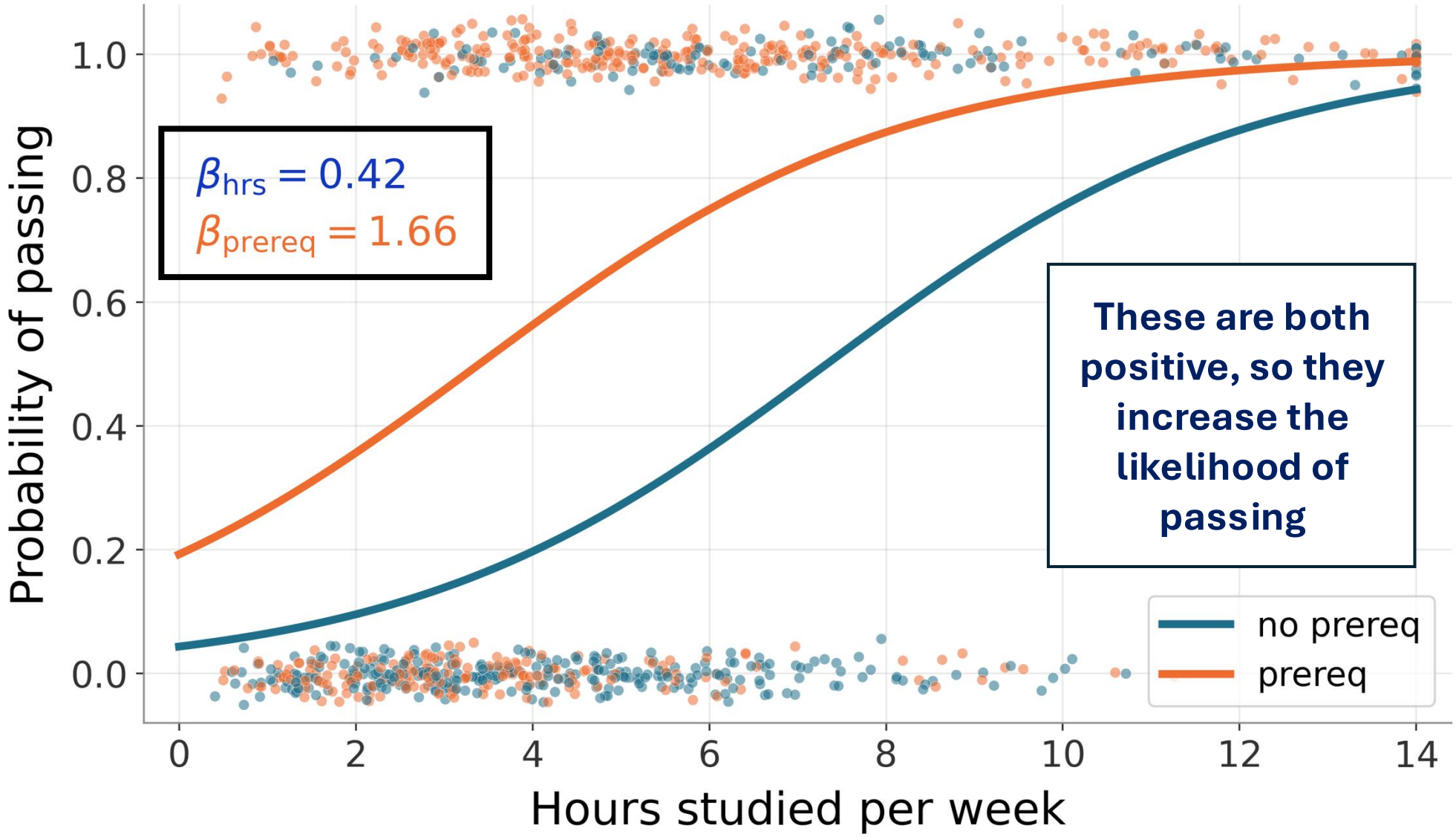
How many hours did they study each week?

Did they take the perquisite course? **(categorical)**

$$x_{\text{prerequisite}} = \begin{cases} 0 \text{ if not} \\ 1 \text{ if yes} \end{cases}$$

Hrs Studied	Took prereq?	Passed?
2	No	No
6	Yes	Yes
3	Yes	Yes
…	…	…

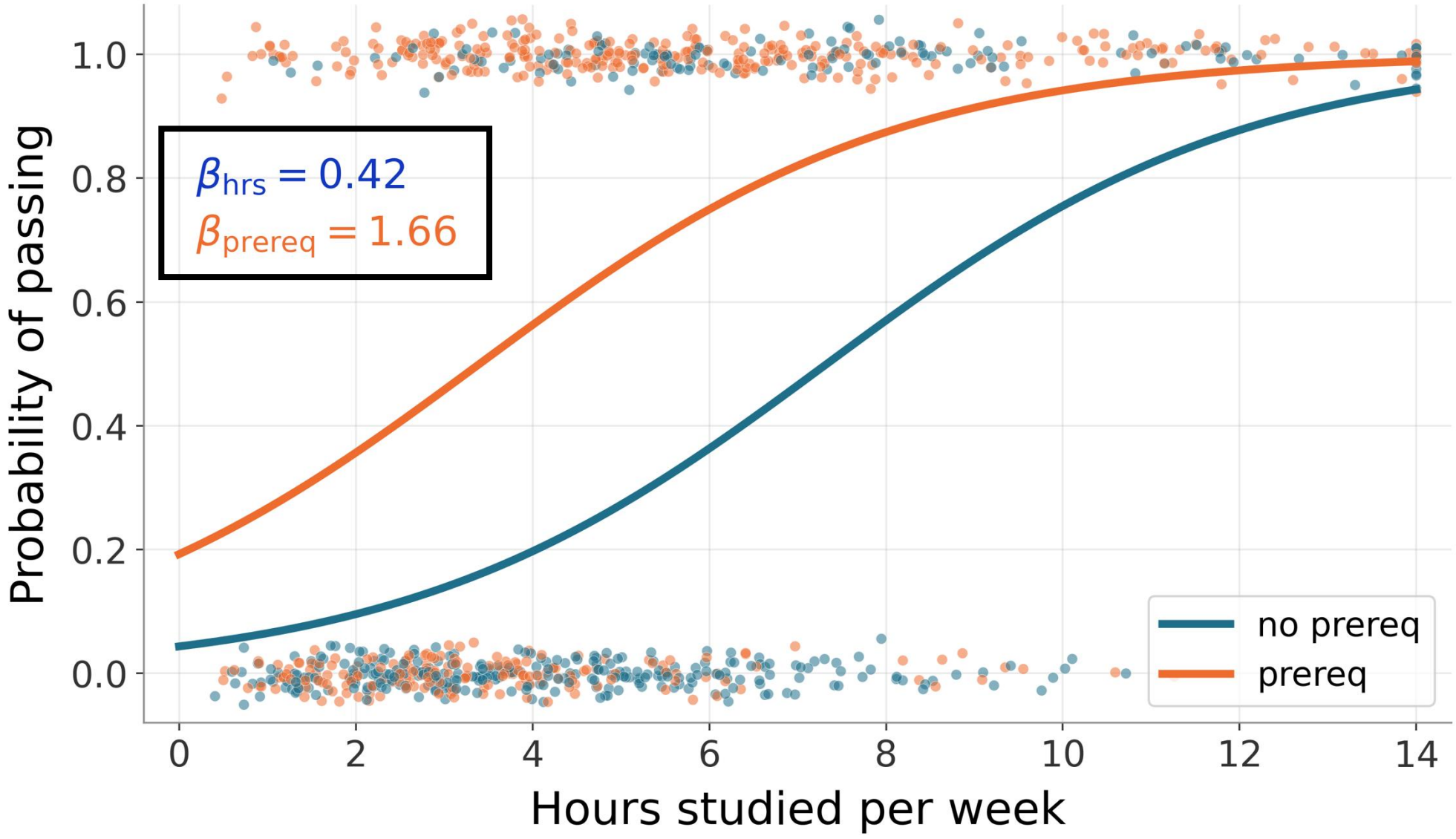
A worked example with mixed variable types



$\beta_{prereq} > \beta_{hrs}$

Taking the prereq has a larger effect than just one extra hour of studying.

A worked example with mixed variable types



$\beta_{\text{hrs}} = 0.42$

For every +1 hour we study, our odds of passing are multiplied by:

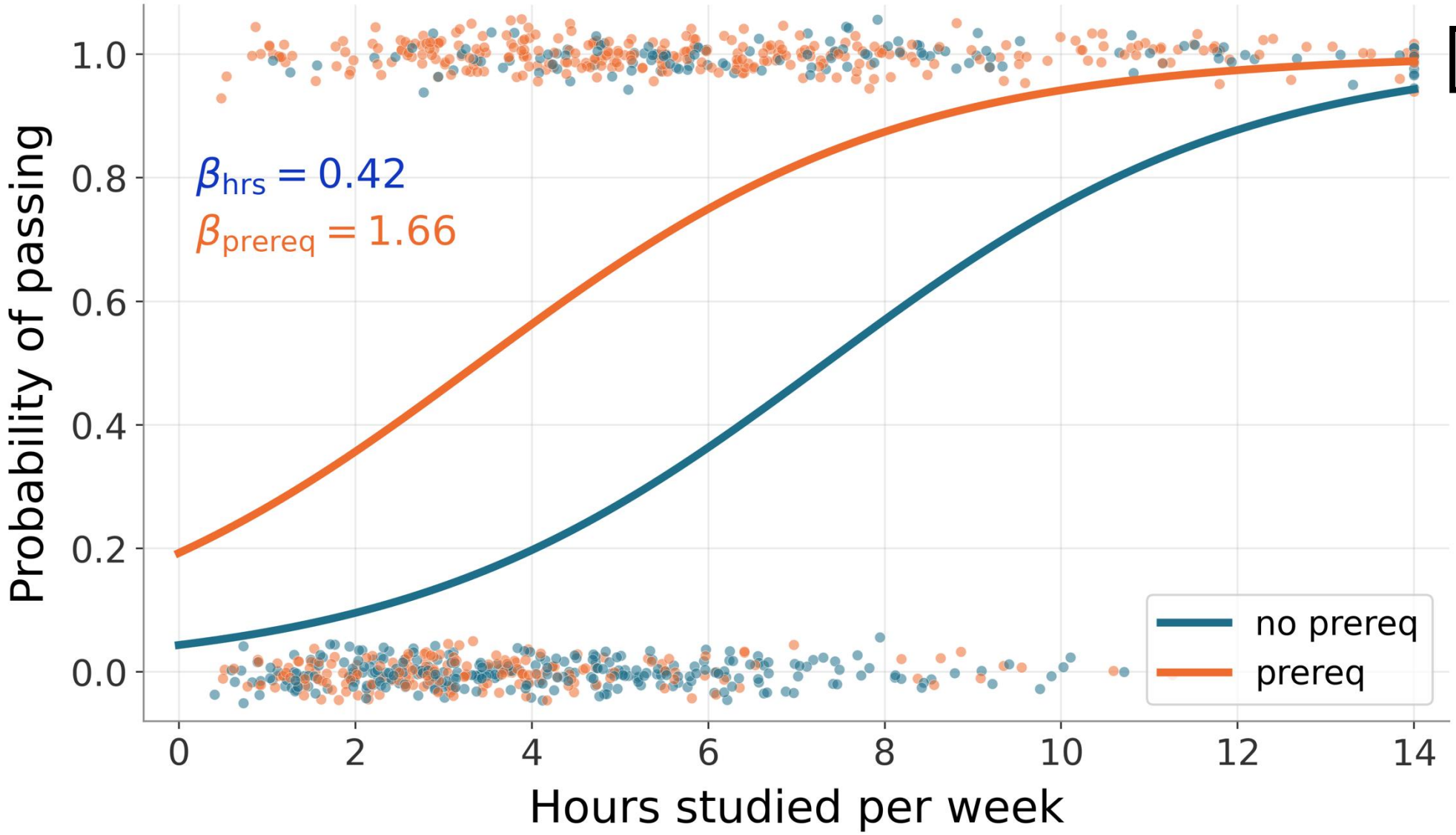
$e^{0.42} \sim 1.53$

$\beta_{\text{prereq}} = 1.66$

By taking the prereq, we multiply our odds by:

$e^{1.66} \sim 5.26$

A worked example with mixed variable types



$$\boxed{\beta_0} + \beta_1 x_{\text{hrs}} + \beta_2 x_{\text{prereq}}$$

$$\beta_0 = -3.1$$

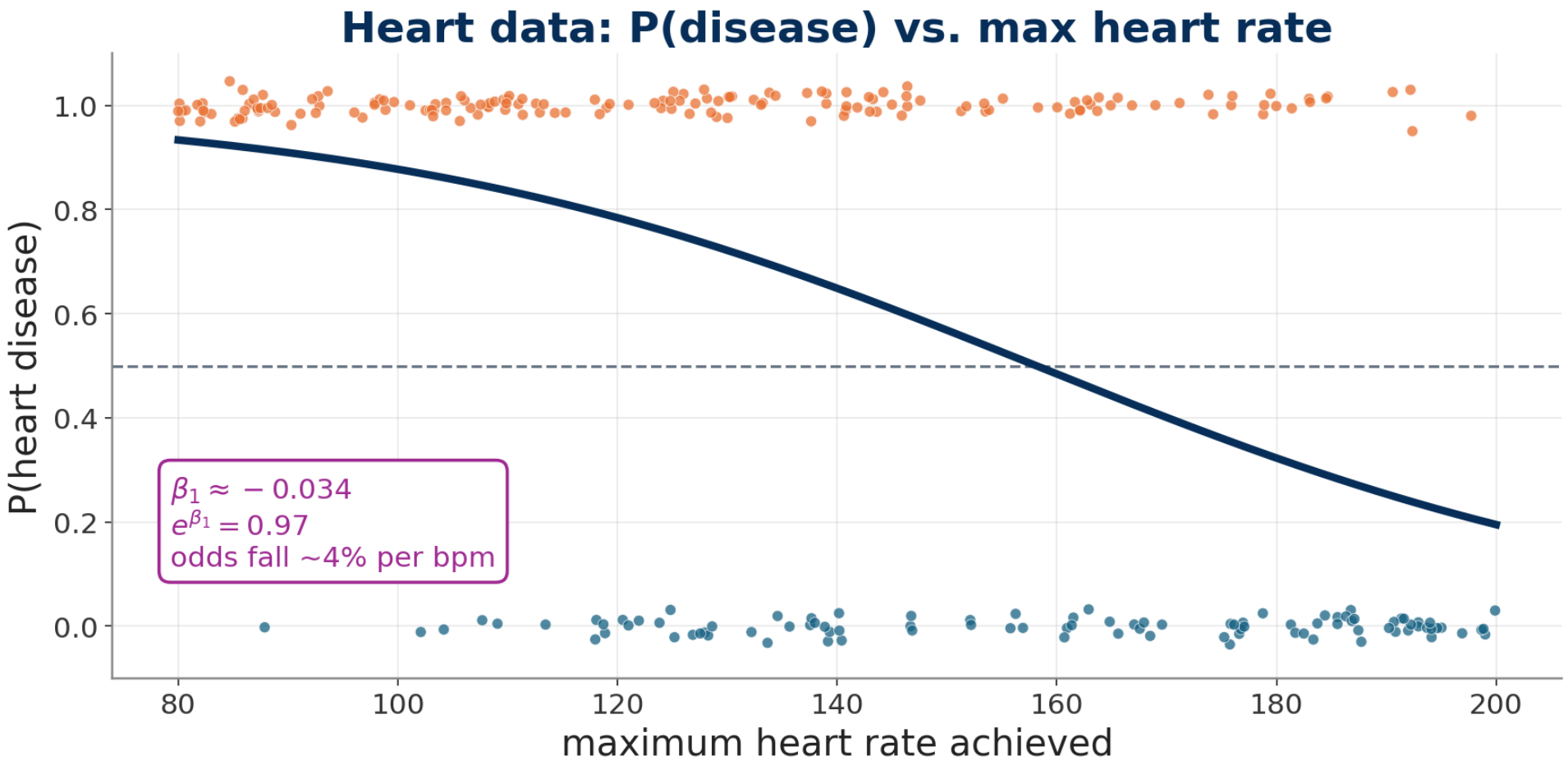
This is the baseline: our log-odds if we don't take the prereq nor study…

$$\log\left(\frac{P_{\text{pass}}}{P_{\text{fail}}}\right) = -3.1$$

$$P_{\text{fail}} = 1 - P_{\text{pass}}$$

$$P_{\text{pass}} = 4.3\%$$

Another example: Heart disease



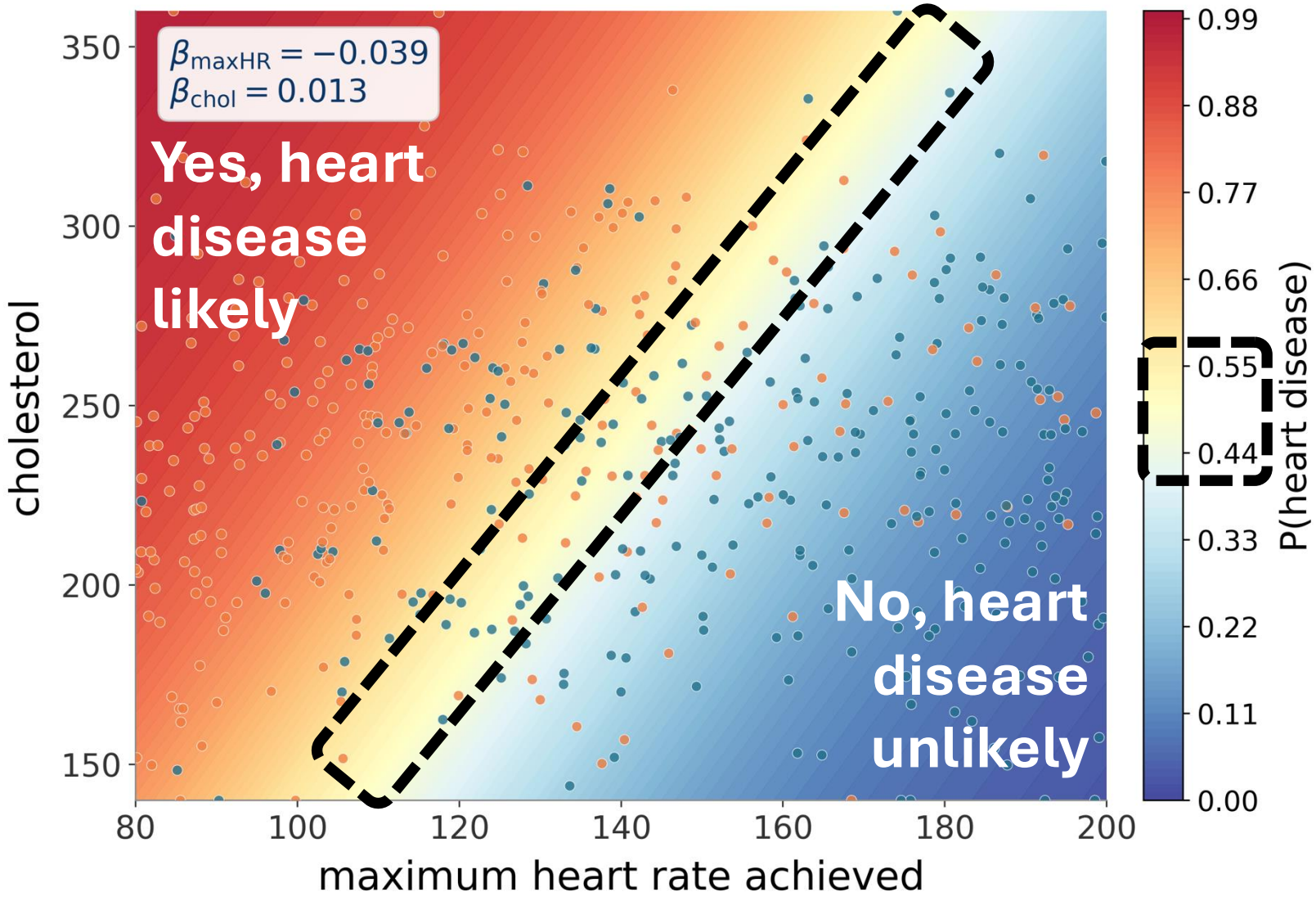
In this case:

$\boldsymbol{\beta_{\text{heart-rate}} < 0}$

If you can achieve a higher heart rate → you have a ***lower*** chance of heart disease

$$\boldsymbol{\beta_{\text{rate}} = -0.034 \text{ per bpm} \rightarrow e^{-0.034} \approx 0.96 \rightarrow \text{the odds fall by about 4\% per bpm}}$$

Visualizing results in two dimensions



$$\log\left(\frac{P_{\text{disease}}}{1-P_{\text{disease}}}\right) = \beta_0 + \beta_1 x_{\text{HR}} + \beta_2 x_{\text{chol}}$$

Now we added another feature (cholesterol)

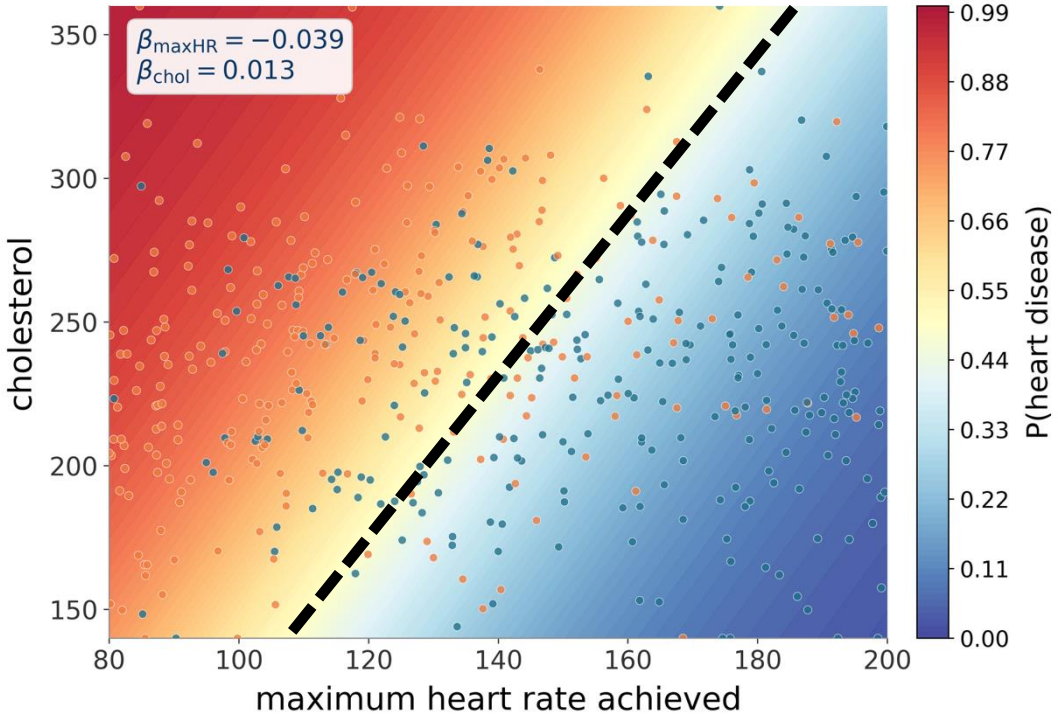
$\beta_{\text{chol}} > 0 \rightarrow$ higher cholesterol raises the odds of heart disease

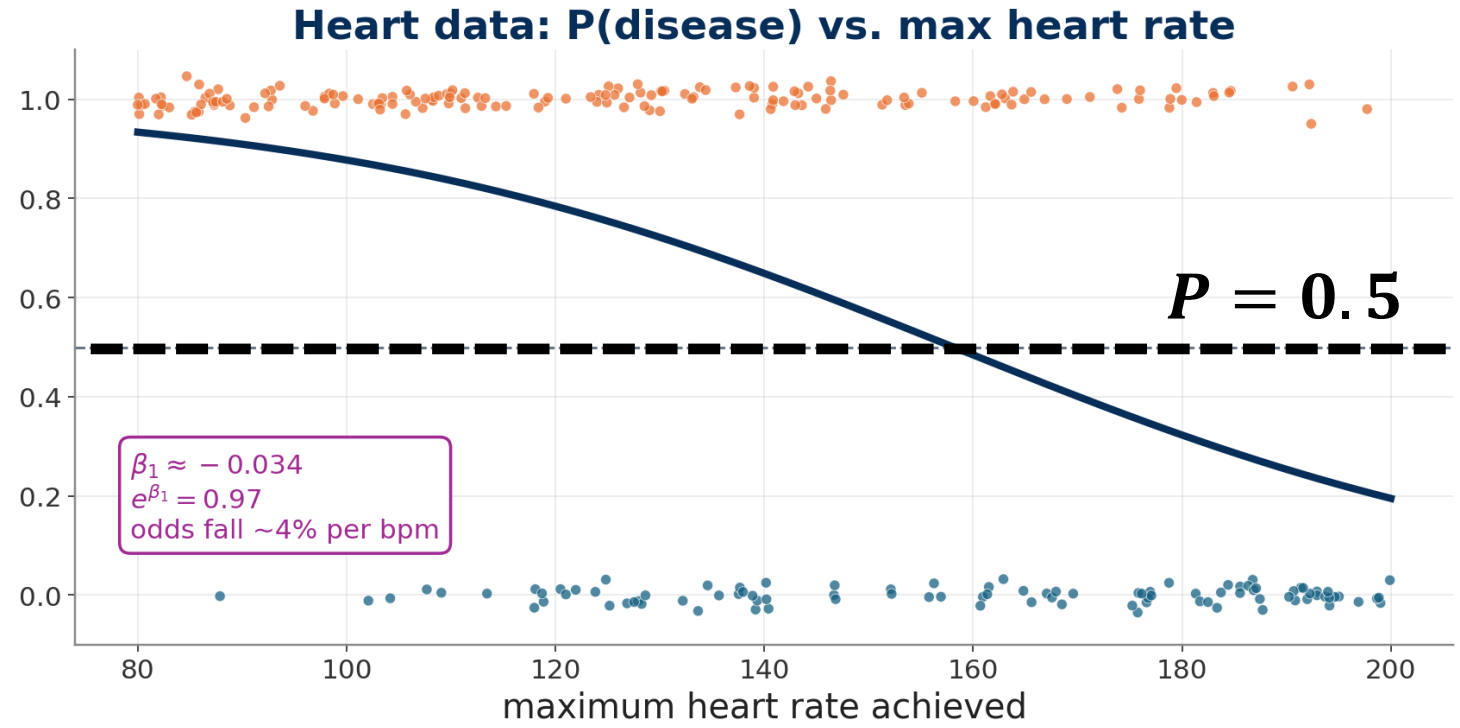
The *decision boundary* (equal odds, $P = 0.5$)

The Decision Boundary

Where the model switches classes

The decision boundary relies on a threshold

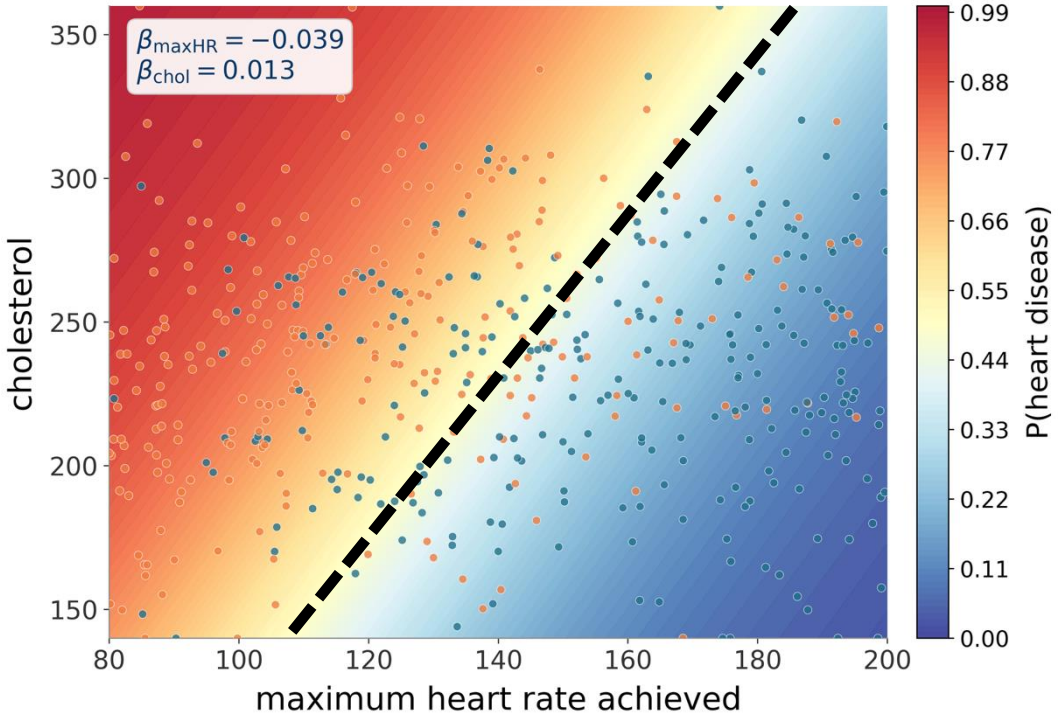




We'd like to do **classification**: (has heart disease) *vs.* (does *not* have heart disease)

By default, we'll **use $P = 0.5$ (50%) as the decision boundary** separating each class

Solving for the boundary: It's just a line!



$$\log\left(\frac{P_{\text{disease}}}{1-P_{\text{disease}}}\right) = \beta_0 + \beta_1 x_{\text{HR}} + \beta_2 x_{\text{chol}}$$

$$\beta_0 = 2.431,\ \beta_1 = -0.039,\ \beta_2 = 0.013$$

To solve for the boundary, set $P_{\text{disease}} = 0.5$ and solve for x_{chol} as a function of x_{HR}

$$\log\left(\frac{0.5}{1-0.5}\right) = \log(1) = 0 = \beta_0 + \beta_1 x_{\text{HR}} + \beta_2 x_{\text{chol}}$$

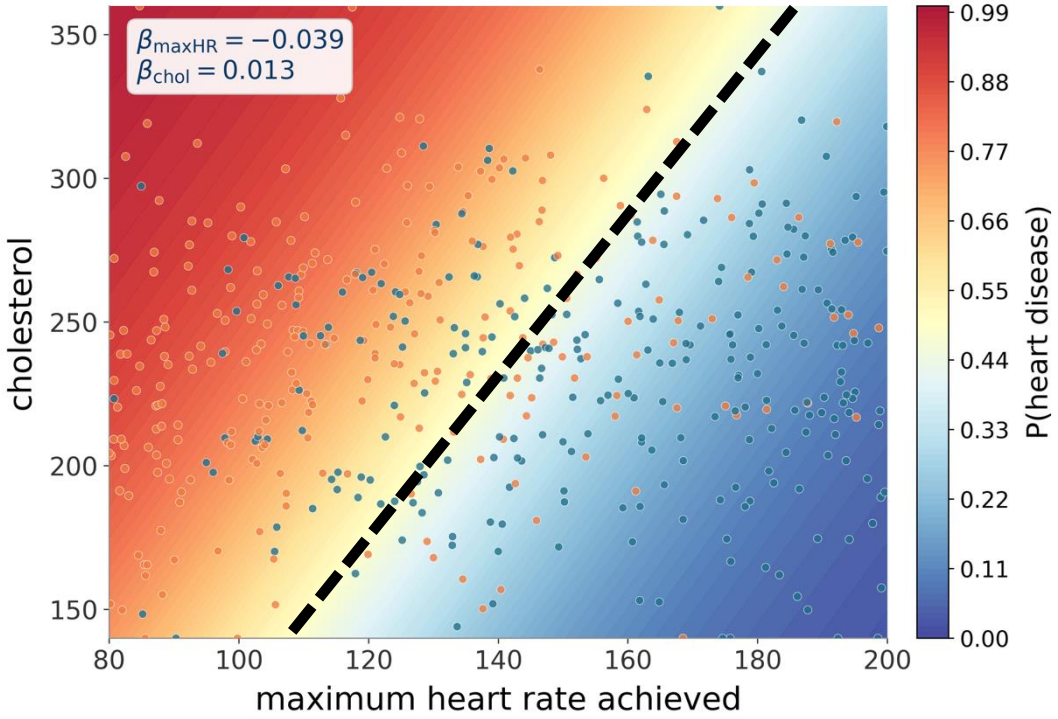
$P = 0.5$ means the log-odds is zero

$$-\beta_0 - \beta_2 x_{\text{chol}} = \beta_1 x_{\text{HR}}$$

$$x_{\text{HR}} = -\frac{\beta_0}{\beta_1} - \frac{\beta_2}{\beta_1} x_{\text{chol}}$$

The equation of a line!
Nice and interpretable.

Solving for the boundary: It's just a line!



This is very useful. Once we train our model, we can ask:

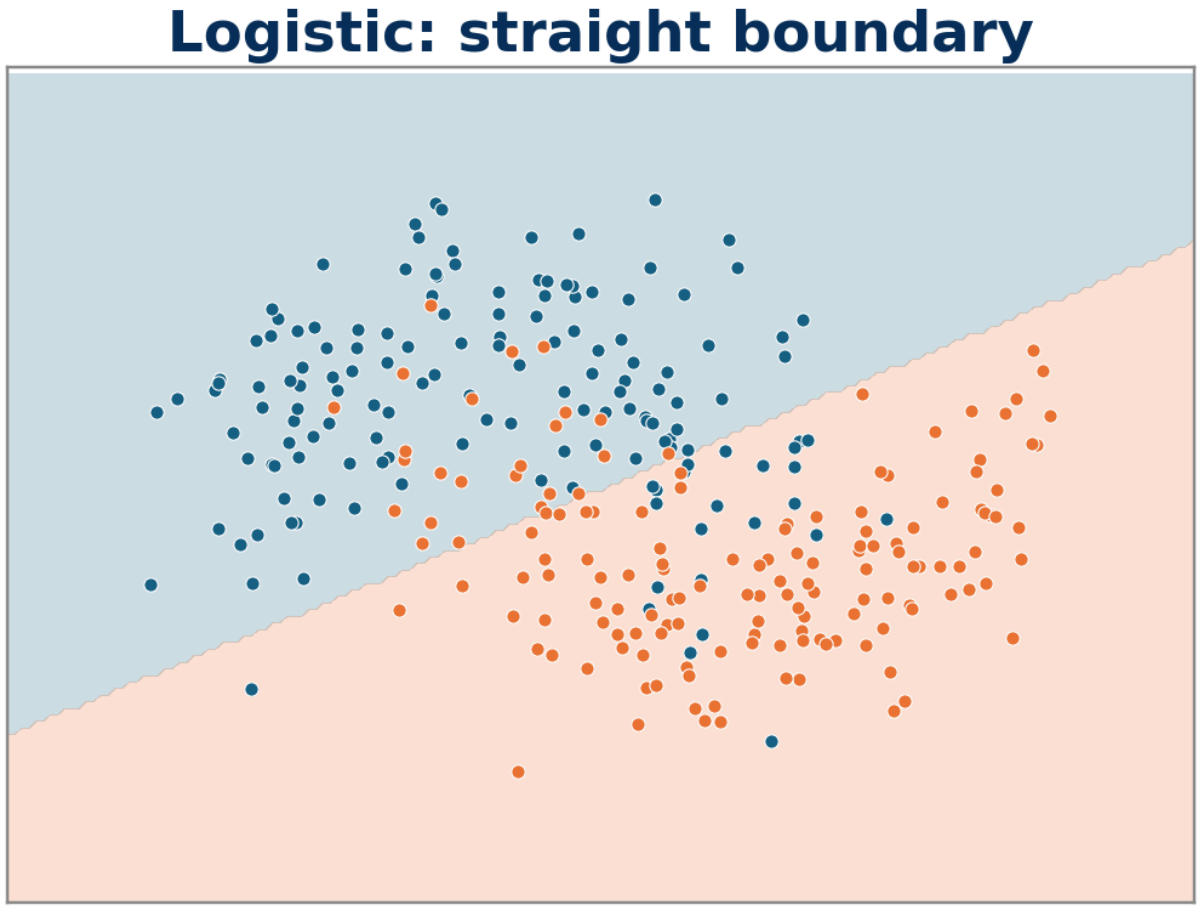
Given a cholesterol, what is the max heart rate we must achieve so that heart disease is unlikely?

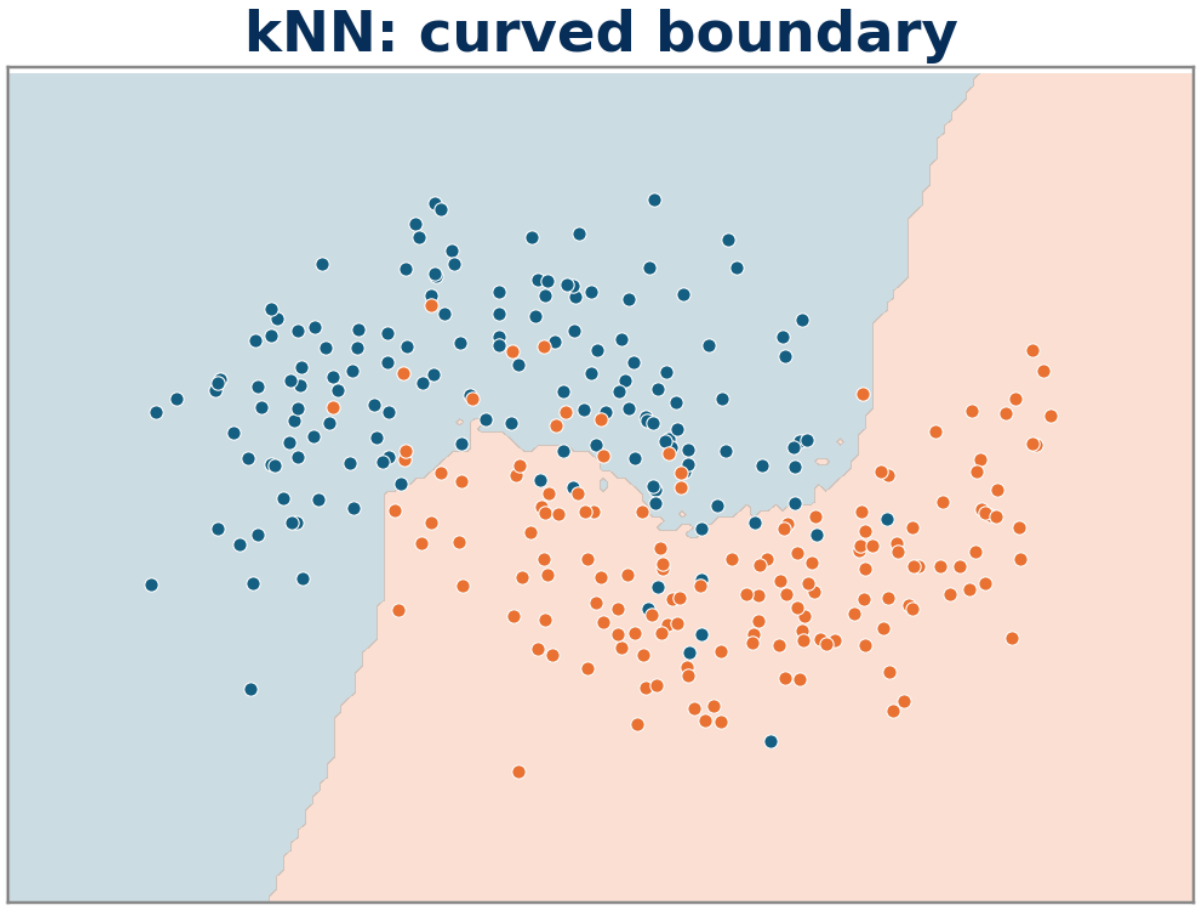
$$-\beta_0 - \beta_2 x_{\text{chol}} = \beta_1 x_{\text{HR}}$$

$$x_{\text{HR}} = -\frac{\beta_0}{\beta_1} - \frac{\beta_2}{\beta_1} x_{\text{chol}}$$

The equation of a line!
Nice and interpretable.

Logistic regression *vs.* KNN for classification





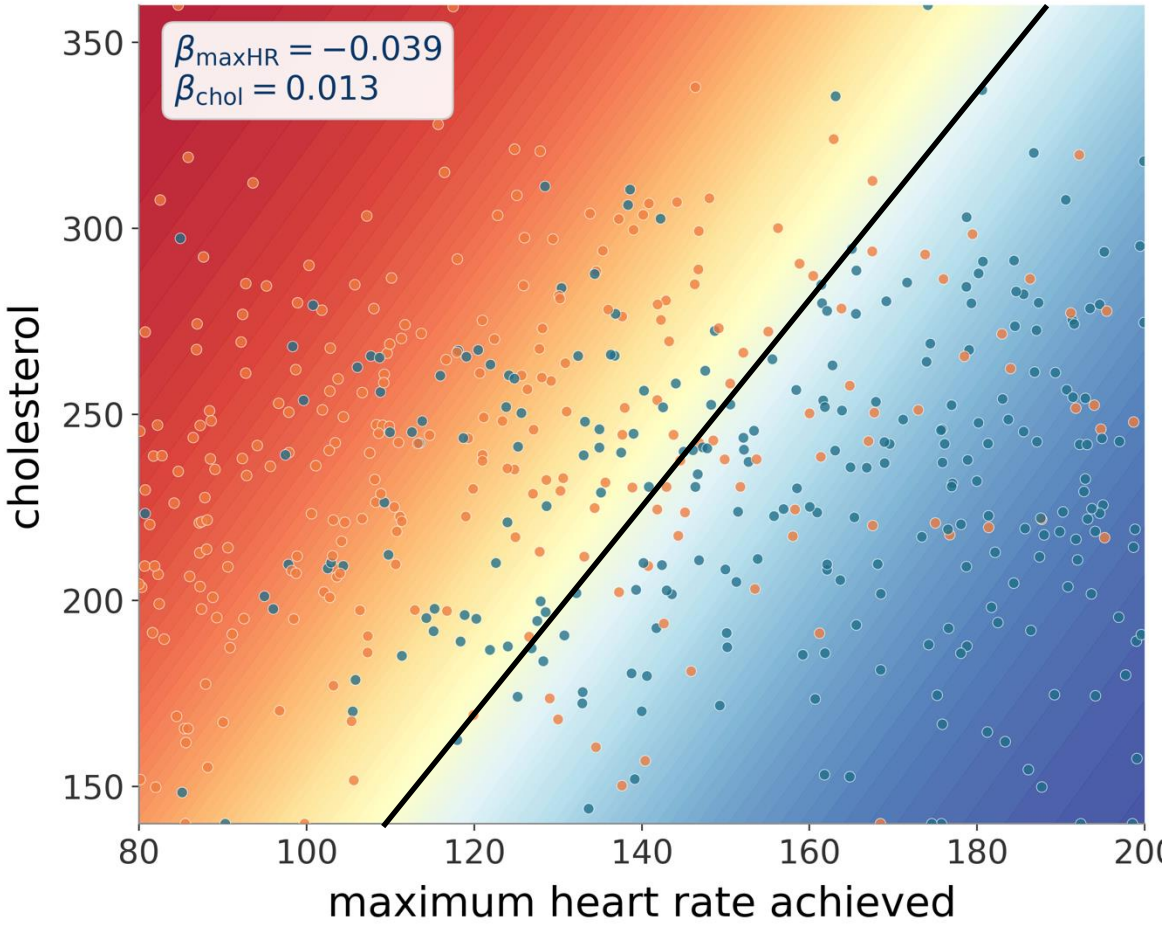
But keep in mind logistic regression does not necessarily need to stay linear.

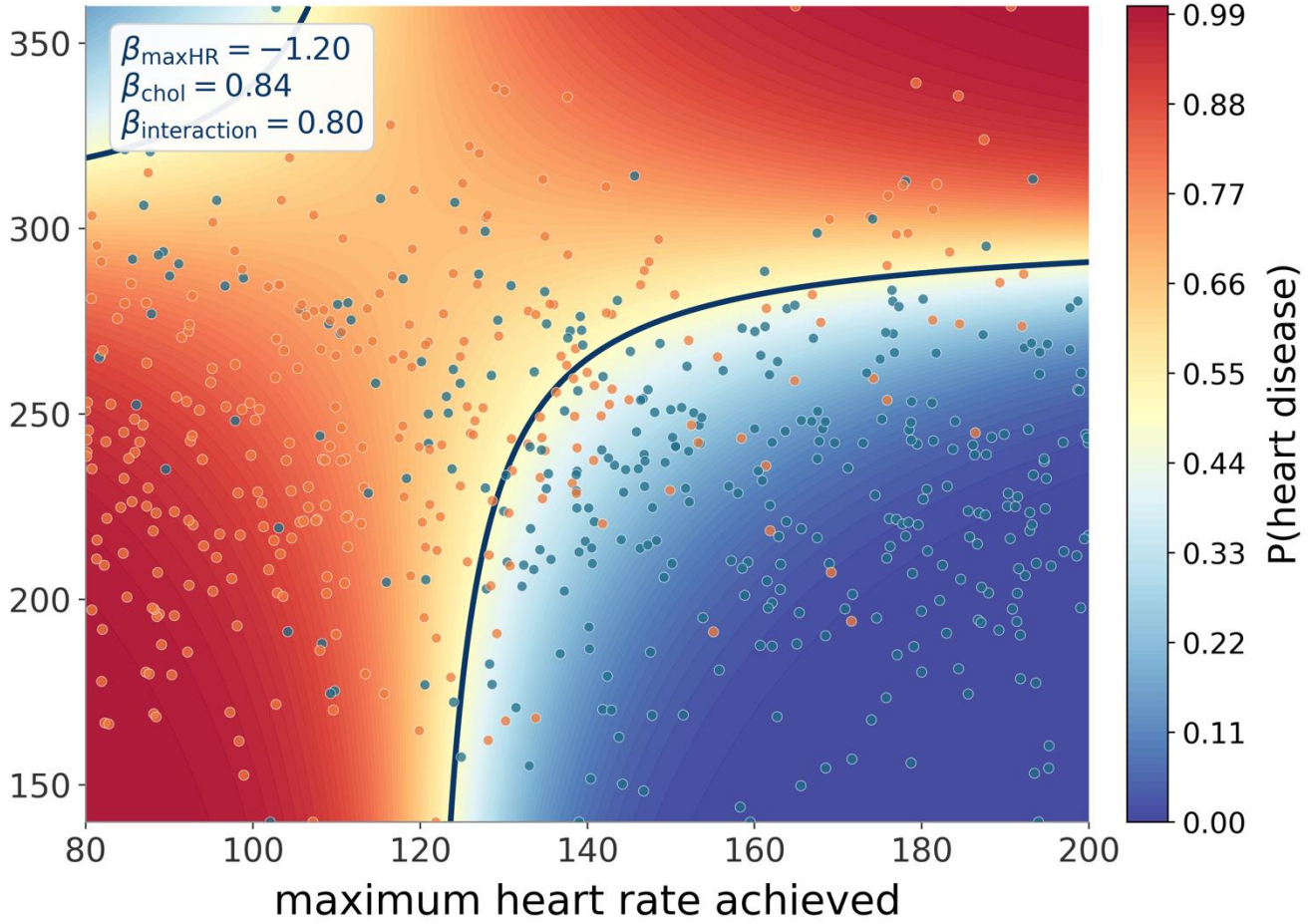
What did we do previously? **Add higher-order and interaction terms!**

Non-linearity in logistic regression

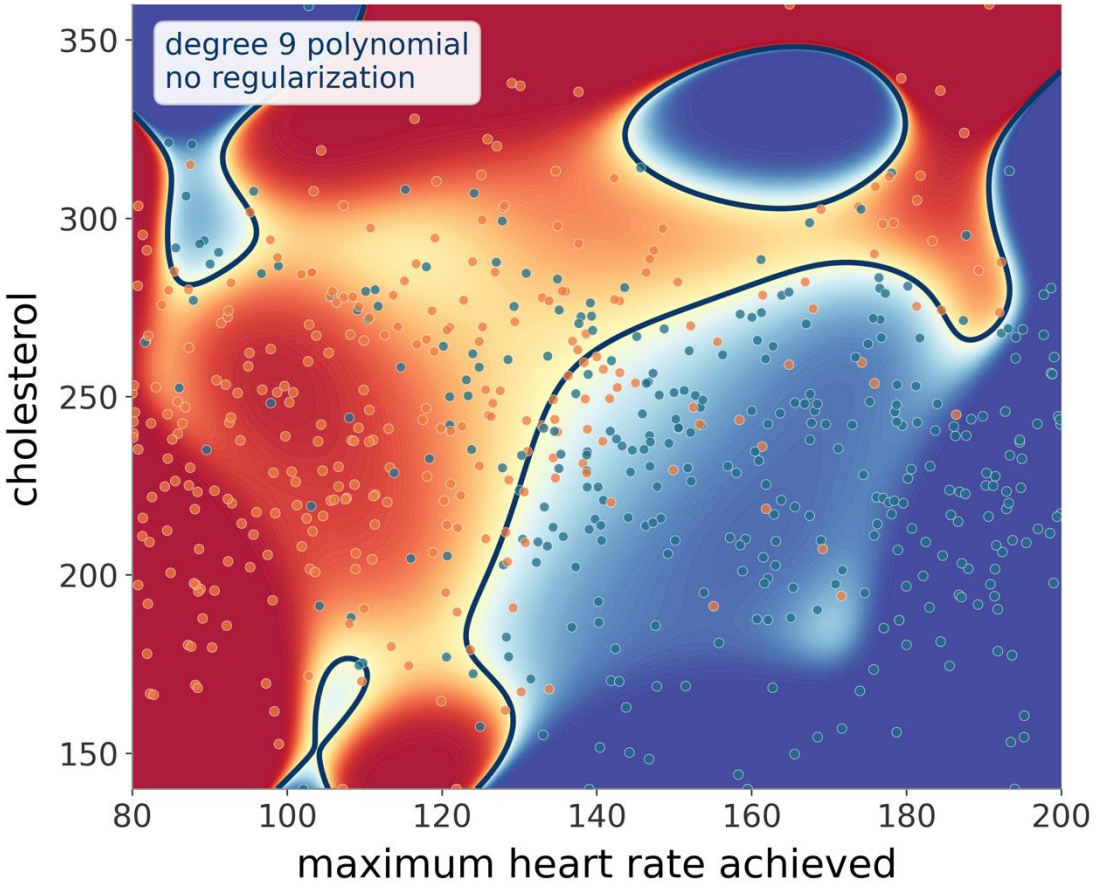
$$\log\left(\frac{P_{\text{disease}}}{1-P_{\text{disease}}}\right) = \beta_0 + \beta_1 x_{\text{HR}} + \beta_2 x_{\text{chol}}$$

$$\beta_0 + \beta_1 x_{\text{HR}} + \beta_2 x_{\text{chol}} + \beta_3 x_{\text{HR}} x_{\text{chol}}$$



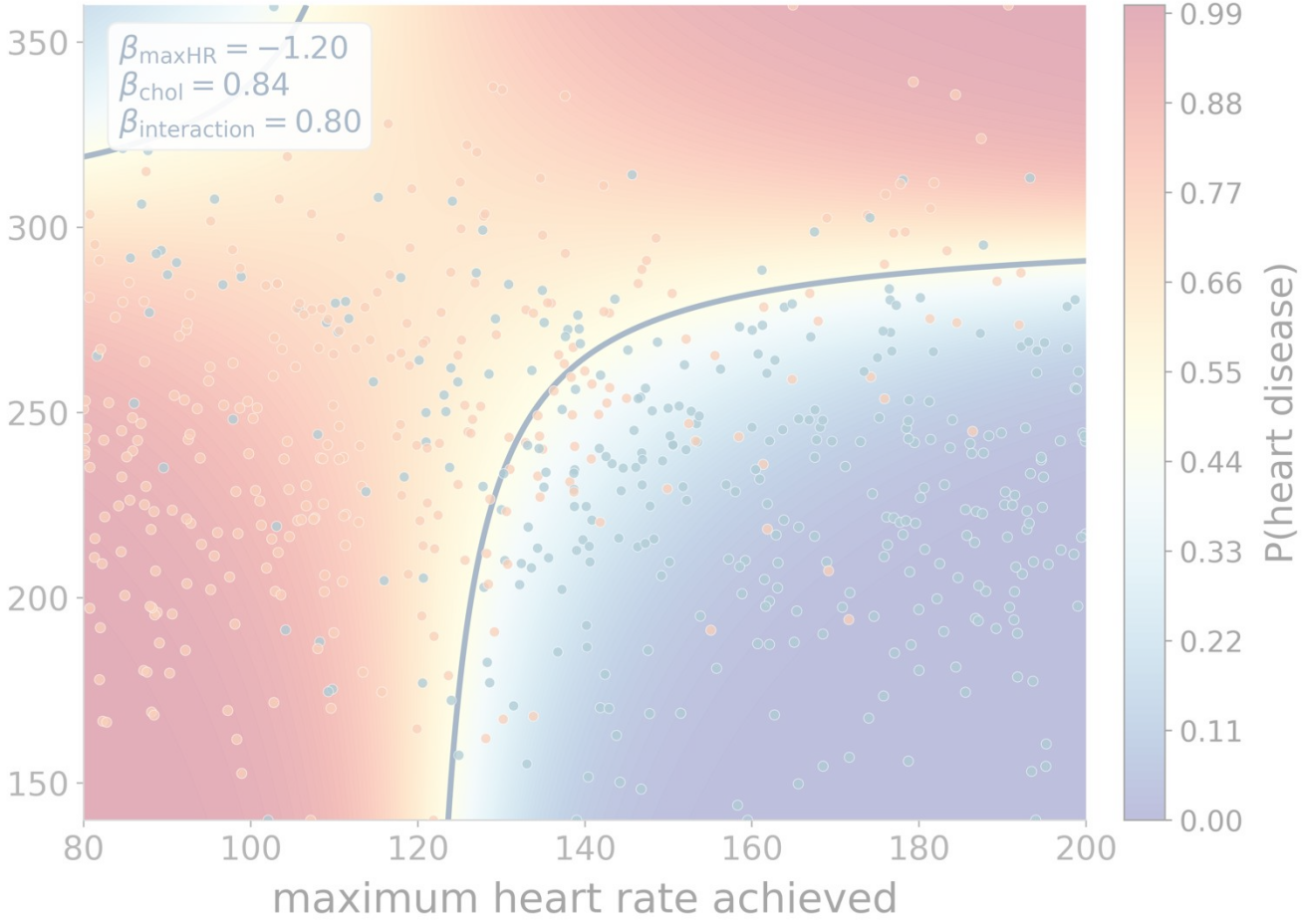


Non-linearity in logistic regression



Just like with linear regression, we can easily overfit with higher order terms

$$\beta_0 + \beta_1 x_{\text{HR}} + \beta_2 x_{\text{chol}} + \beta_3 x_{\text{HR}} x_{\text{chol}}$$



Regularization for logistic regression

Before: we find β to **minimize the cross-entropy**

$$J(\beta) = -\frac{1}{n} l(\beta) = -\frac{1}{n} \sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

Now, with regularization: We **minimize a weighted sum of the cross-entropy and the coefficient magnitudes**

$$\mathcal{L} = J(\beta) + \lambda R(\beta_i)$$

L_2-norm

$$R(\beta_i) = \sum_i \beta_i^2$$

Ridge regression

L_1-norm

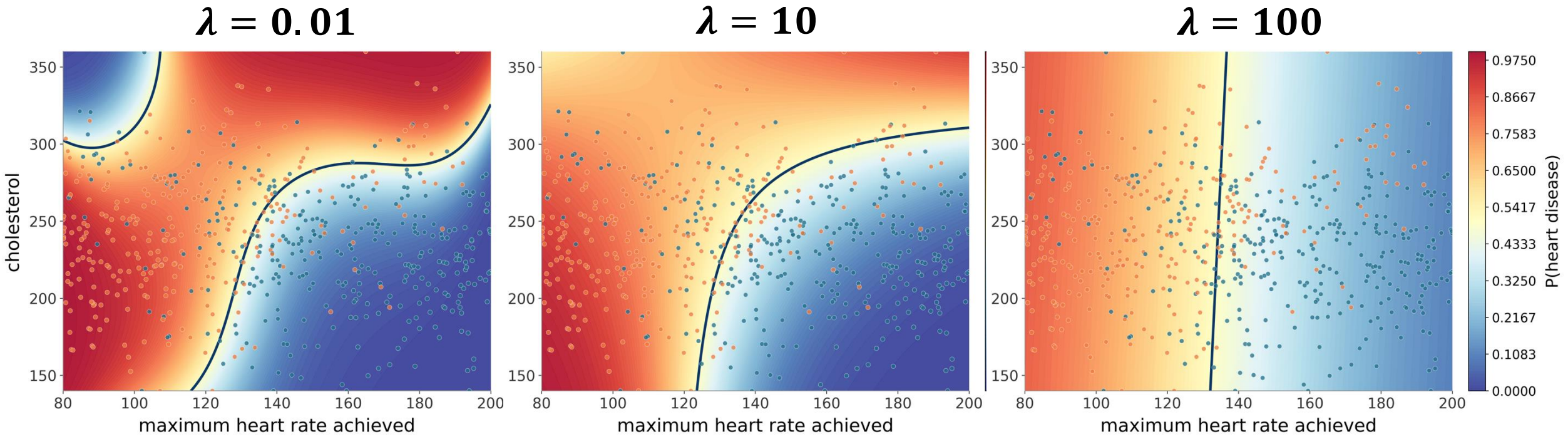
$$R(\beta_i) = \sum_i |\beta_i|$$

Lasso regression

Remember:

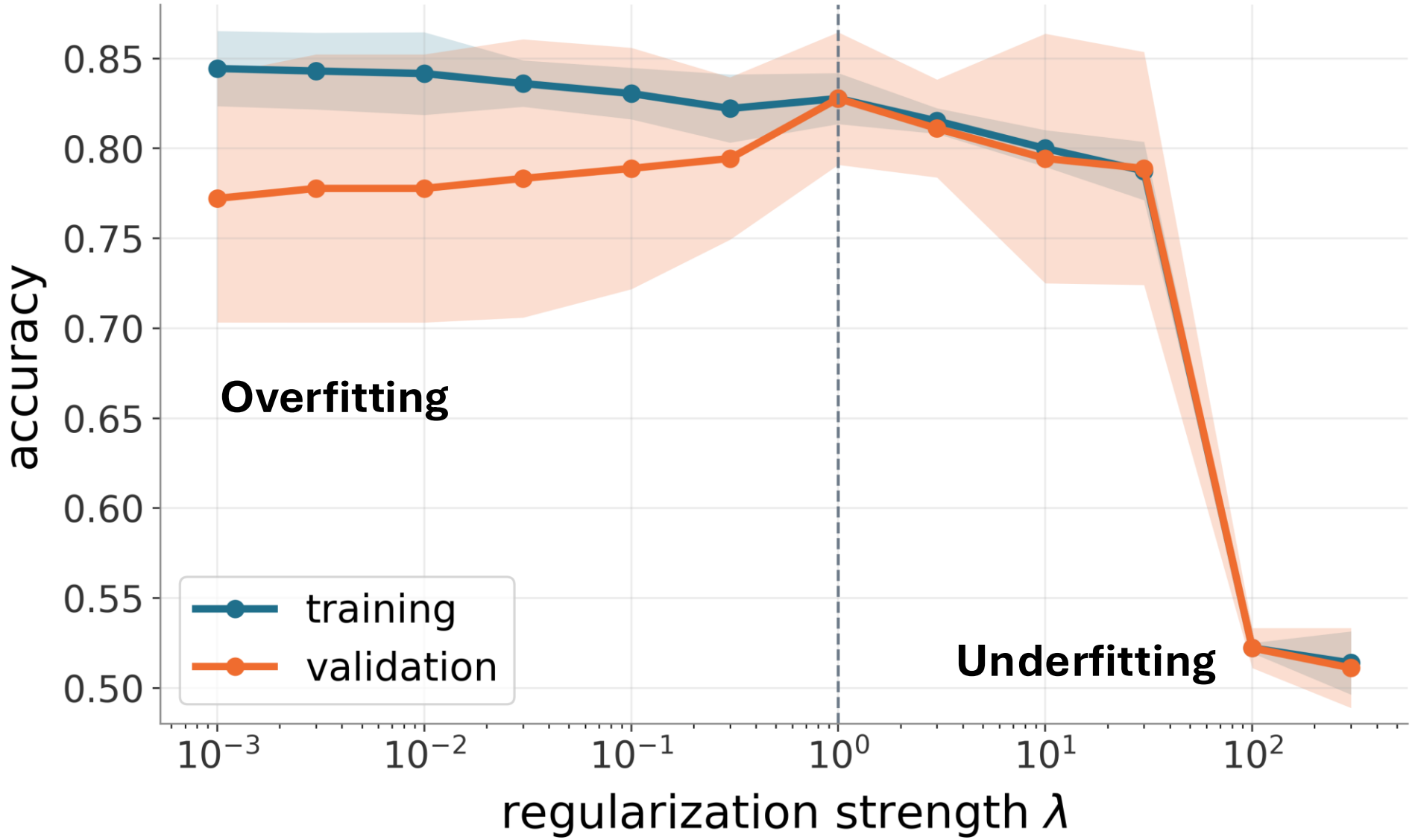
- Ridge keeps all coefficients
- Lasso forces some to zero

Stronger regularization (λ) = smoother boundary



As we've discussed, you should choose the "right" λ by doing **cross-validation.**

Cross-validation for logistic regression



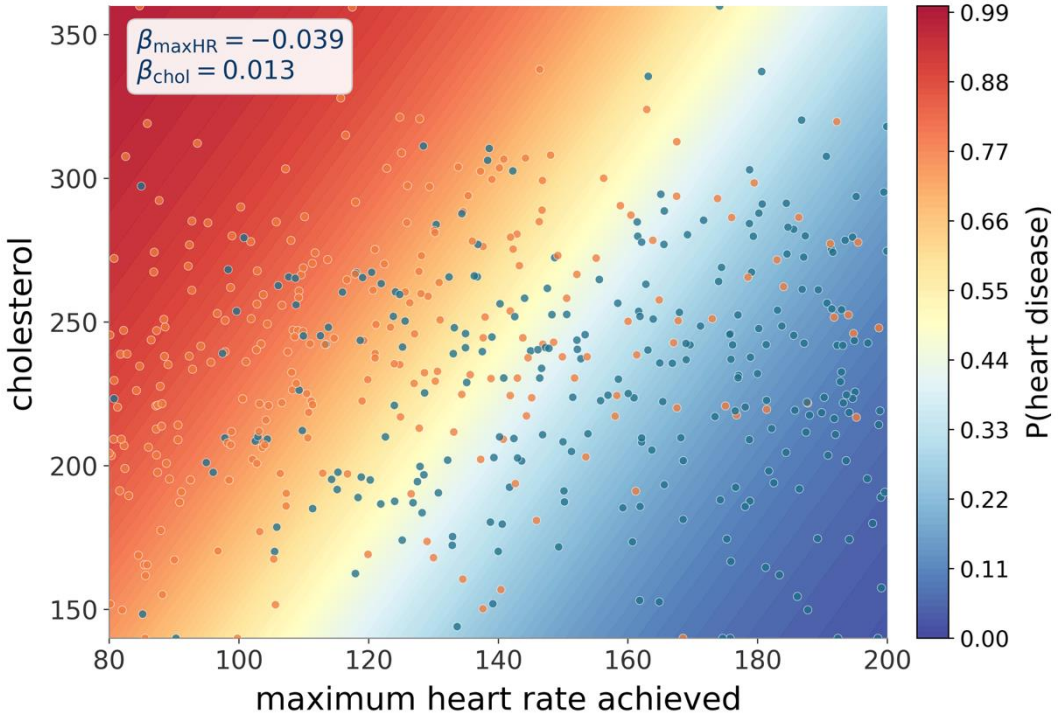
Remember that training accuracy will always decrease as we regularize.

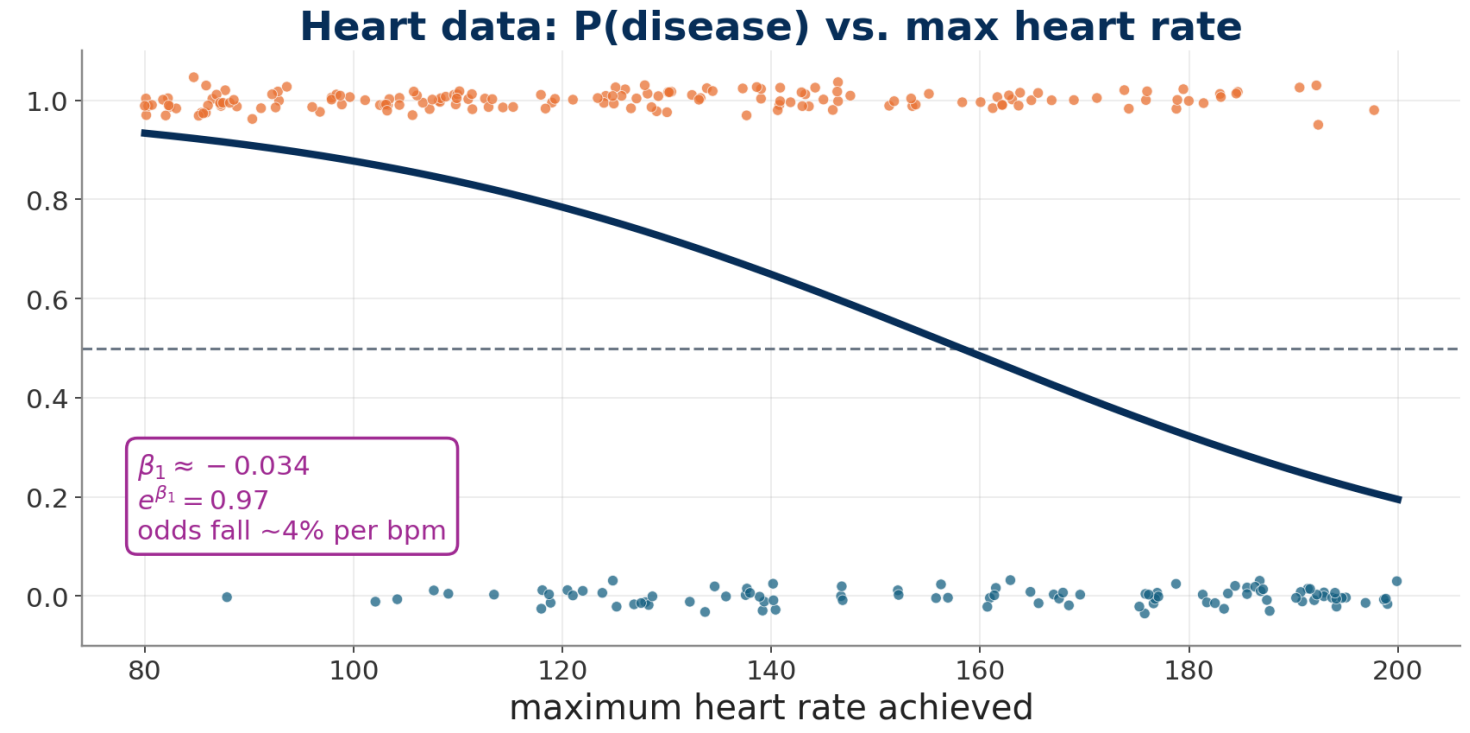
What we care about is validation accuracy!

What if we have > 1 class?

Multiclass logistic regression

From binary to multiclass





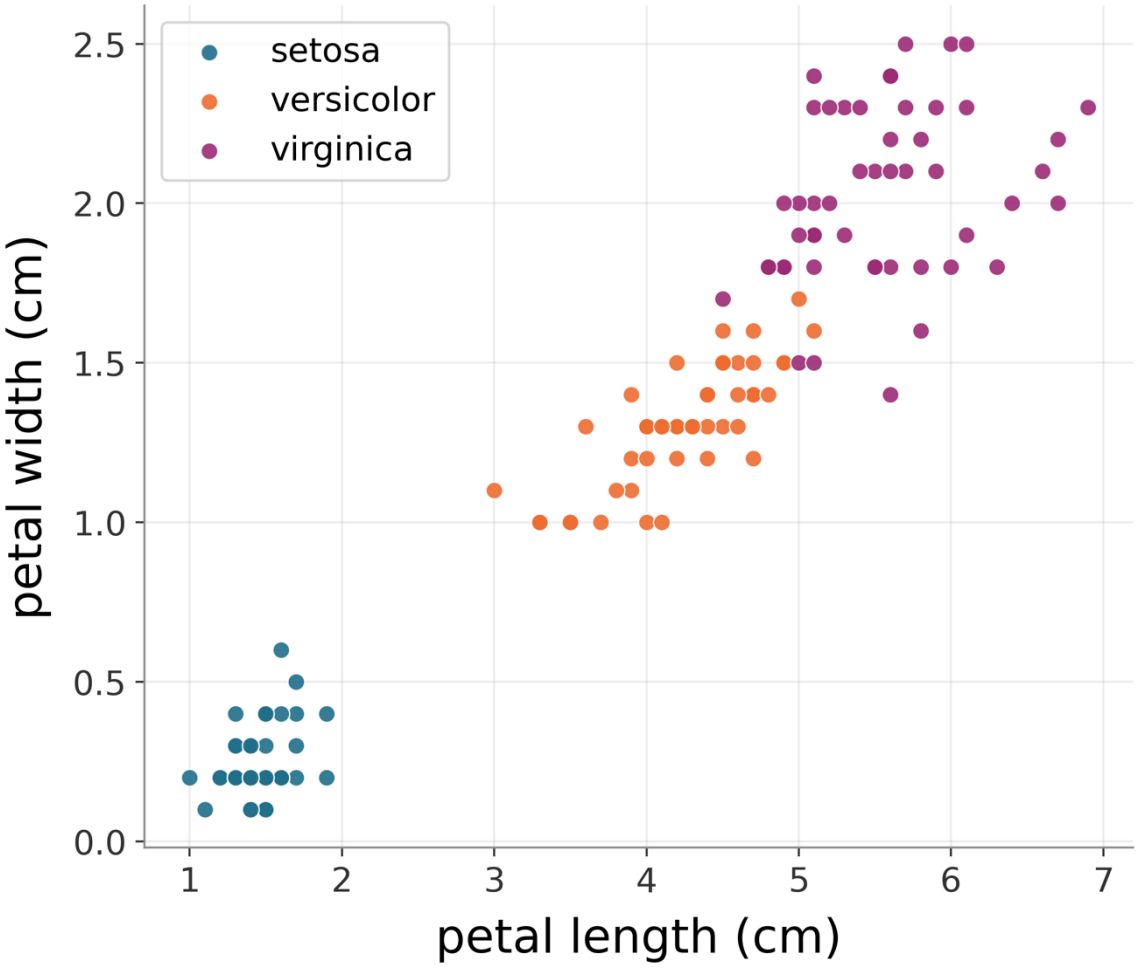
We've so far only covered **binary problems** – there are **two classes** to choose from:

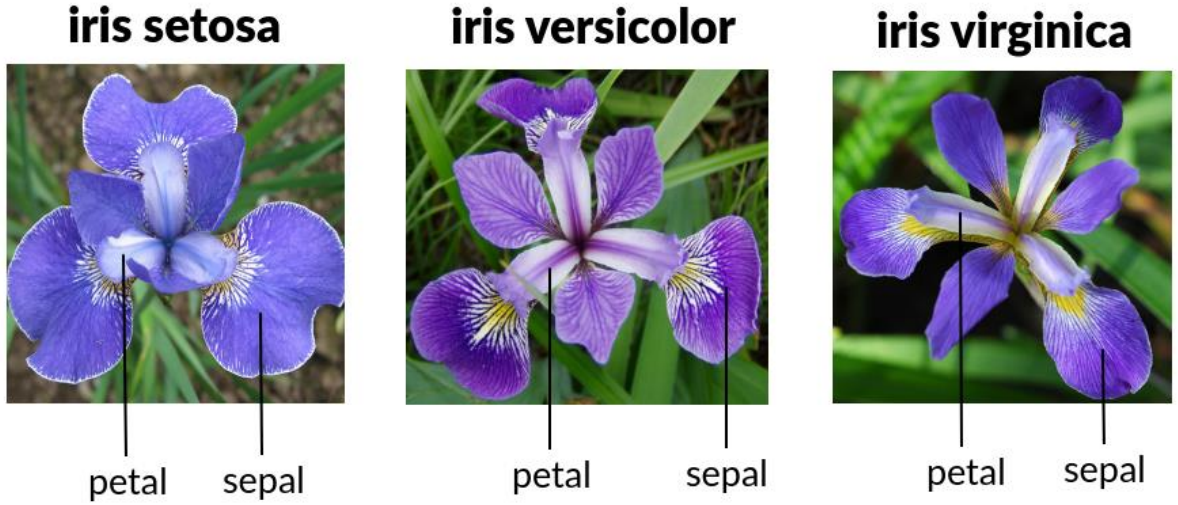
Will you contract heart disease? **Yes ($y = 1$)** or **No ($y = 0$)**

And because **probabilities must sum to 1**, we only had to model one: $P(y = 1|x)$

From binary to multiclass

There are of course many situations with **> 2 classes to choose from (multiclass)**





Three classes, each with their own probability:

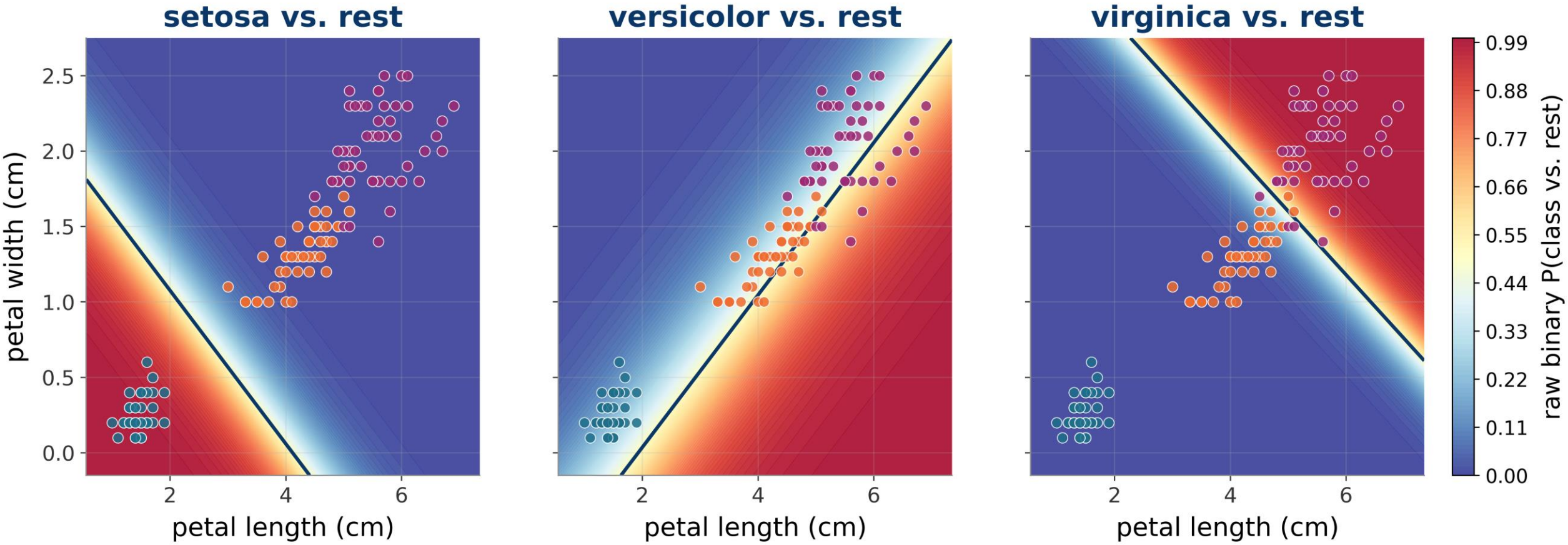
$$P(y = \text{setosa}|x), \qquad P(y = \text{versicolor}|x),$$

$$P(y = \text{virginica}|x)$$

Where x are features describing the flower

Strategy #1: One-*vs.*-Rest (OvR)

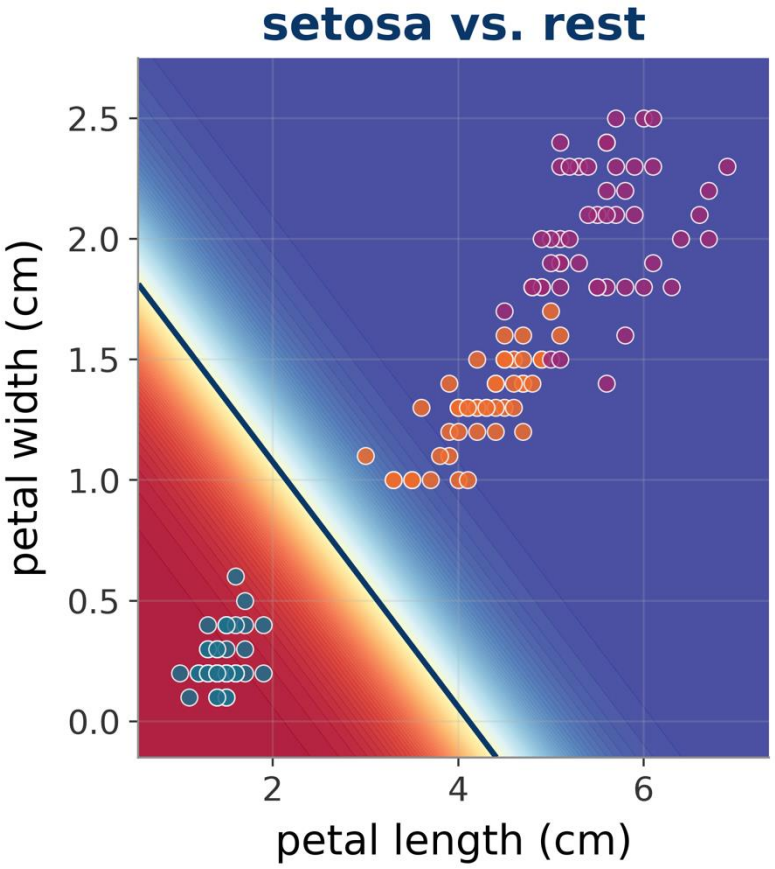
For K classes, fit K separate binary logistic-regression models



These are completely independent of one another! **Probabilities may not sum to 1**

Strategy #1: One-*vs.*-Rest (OvR)

For K classes, fit K <u>separate</u> binary logistic-regression models



$$\log\left(\frac{P_{\text{Setosa}}}{P_{\text{Not Setosa}}}\right) = \beta_0 + \beta_1 x_{\text{petal length}} + \beta_2 x_{\text{petal width}}$$

We model whether a flower ***IS* a Setosa** or ***IS NOT* a Setosa**, regardless of which class the "*Is not* Setosa" flower belongs to...

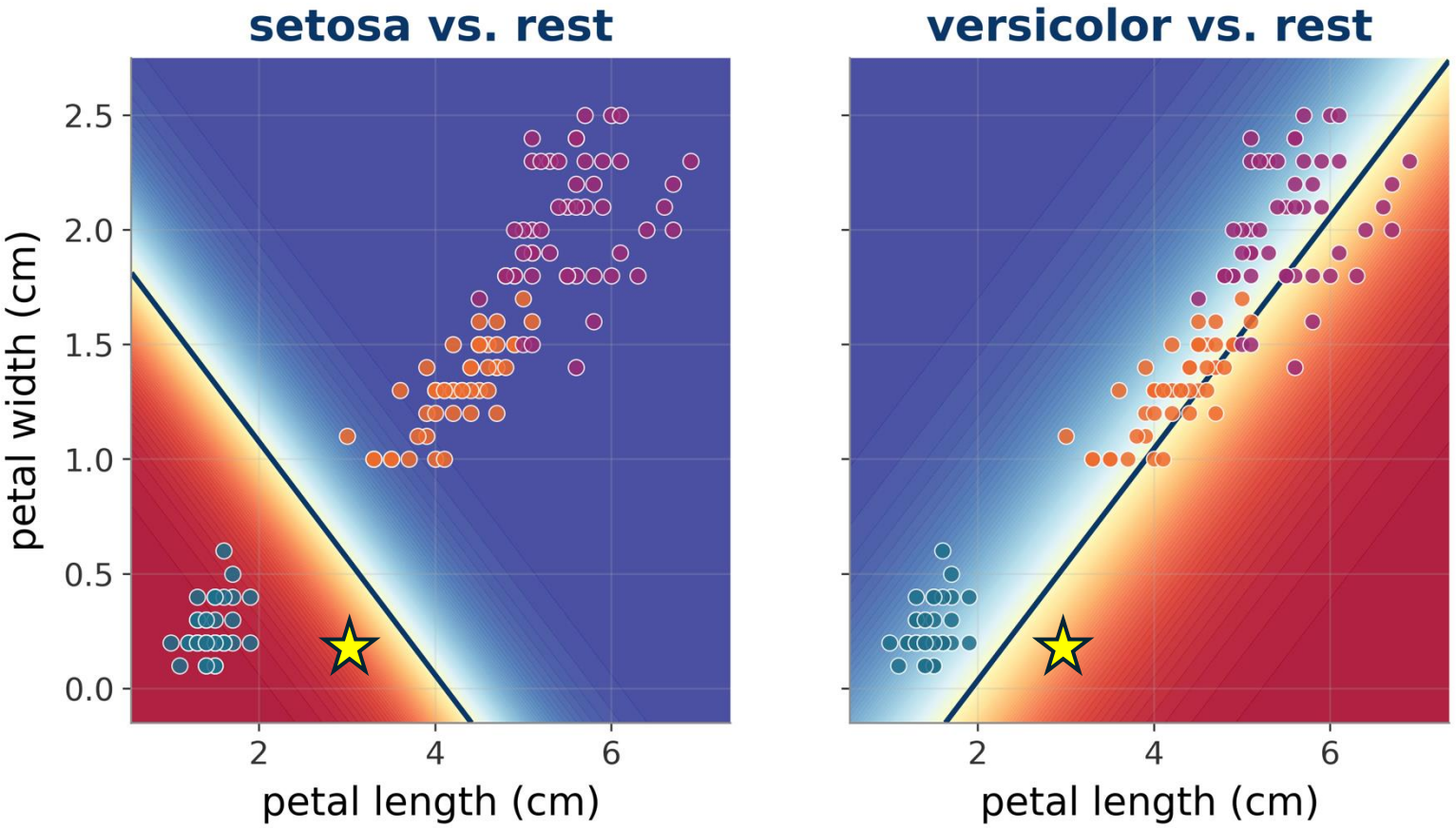
Do the same for each class. Then, given a new data point (x) we separately compute three probabilities:

$$P(y = \text{setosa}|x), P(y = \text{versicolor}|x),$$
$$P(y = \text{virginica}|x)$$



How to make final decision?

Strategy #1: One-*vs.*-Rest (OvR)



Consider a new sample with a **petal length of 3 cm** and a **petal width of 0.2 cm…**

$$P(y = \text{setosa}|x) = 0.84$$
$$P(y = \text{versicolor}|x) = 0.72$$

Based on just these two classes, we would label the new sample as **setosa** since it has a higher probability

But these are not strictly class probabilities – they don't sum to 1

Strategy #2: Multinomial

Create a **joint model** that optimizes all nine coefficients simultaneously

$$z^{\text{Setosa}} = \beta_0^{\text{Setosa}} + \beta_1^{\text{Setosa}} x_{\text{petal length}} + \beta_2^{\text{Setosa}} x_{\text{petal width}}$$

$$z^{\text{Versi}} = \beta_0^{\text{Versi}} + \beta_1^{\text{Versi}} x_{\text{petal length}} + \beta_2^{\text{Versi}} x_{\text{petal width}}$$

$$z^{\text{Virg}} = \beta_0^{\text{Virg}} + \beta_1^{\text{Virg}} x_{\text{petal length}} + \beta_2^{\text{Virg}} x_{\text{petal width}}$$

Then pass these scores (z^k) through the **softmax function** to get probabilities:

$$P(y = k) = \frac{e^{z^k}}{\sum_{i=1}^{K} e^{z^k}}$$

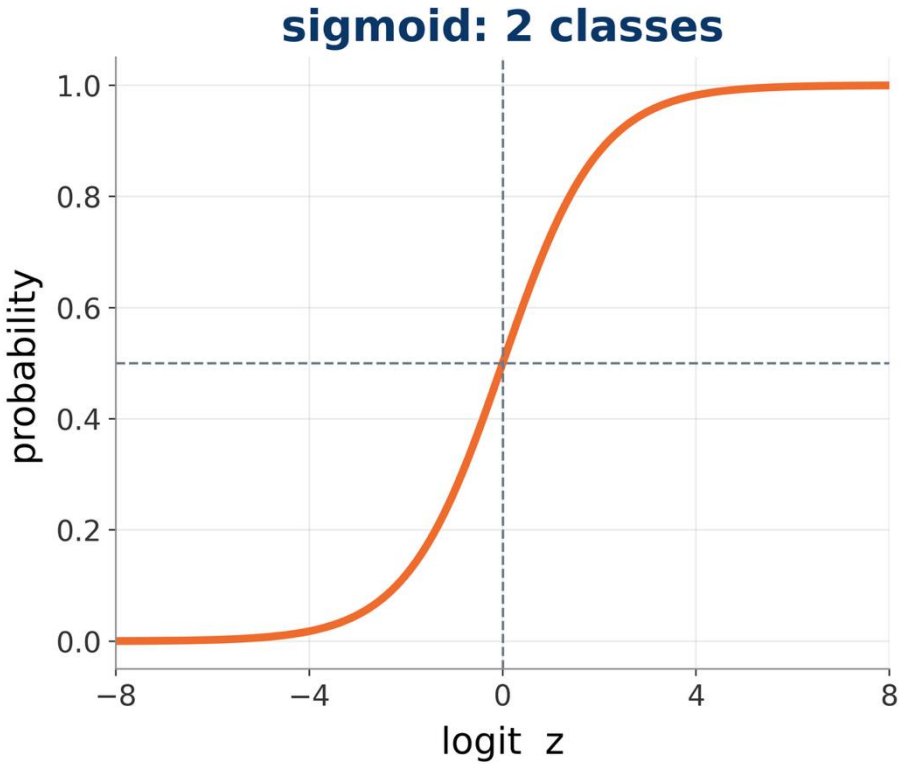
k = each class

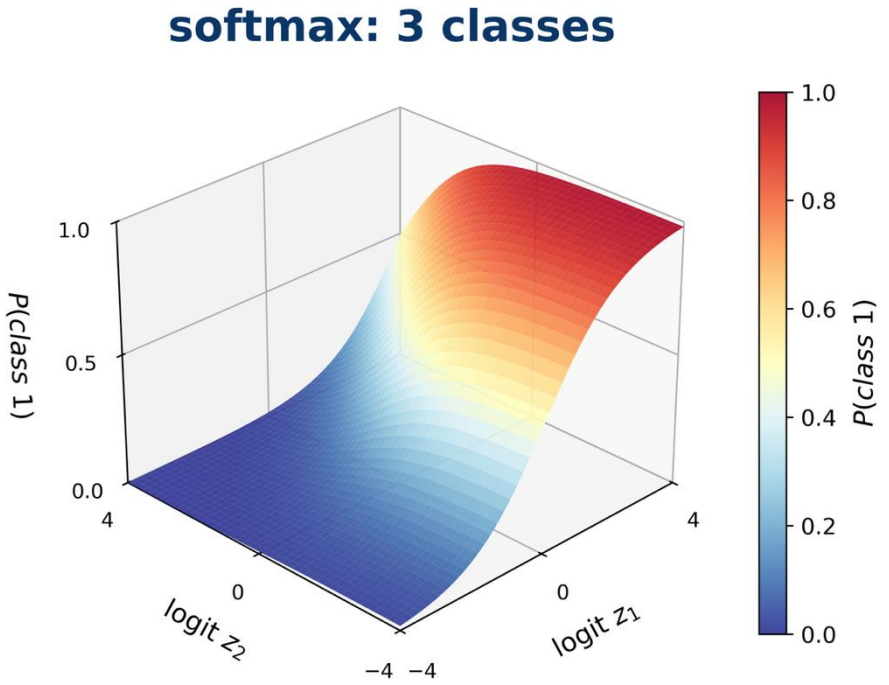
$$P(y = \text{Setosa}|x) = \frac{e^{z^{\text{Setosa}}}}{e^{z^{\text{Setosa}}} + e^{z^{\text{Versi}}} + e^{z^{\text{Virg}}}}$$

$$P(y = \text{Versi}|x) = \frac{e^{z^{\text{Versi}}}}{e^{z^{\text{Setosa}}} + e^{z^{\text{Versi}}} + e^{z^{\text{Virg}}}}$$

And so on…This ensure **probabilities sum to 1**

The softmax function





$$P(y = k) = \frac{e^{z^k}}{\sum_{i=1}^{K} e^{z^k}}$$

k = each class

Think of this as a more general version of the **sigmoid** function. If we only have two classes:

$$P(y = 1) = \frac{e^{z^1}}{e^{z^1} + e^{z^2}} = \frac{1}{1 + e^{-(z^1 - z^2)}} = \sigma(z^1 - z^2)$$

Strategy #2: Multinomial

Create a **joint model** that optimizes all nine coefficients simultaneously

$$z^{\text{Setosa}} = \beta_0^{\text{Setosa}} + \beta_1^{\text{Setosa}} x_{\text{petal length}} + \beta_2^{\text{Setosa}} x_{\text{petal width}}$$

$$z^{\text{Versi}} = \beta_0^{\text{Versi}} + \beta_1^{\text{Versi}} x_{\text{petal length}} + \beta_2^{\text{Versi}} x_{\text{petal width}}$$

$$z^{\text{Virg}} = \beta_0^{\text{Virg}} + \beta_1^{\text{Virg}} x_{\text{petal length}} + \beta_2^{\text{Virg}} x_{\text{petal width}}$$

By minimizing the multiclass cross-entropy

$$J(\boldsymbol{\beta}) == -\frac{1}{n}\sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i)\log(1 - p_i)$$

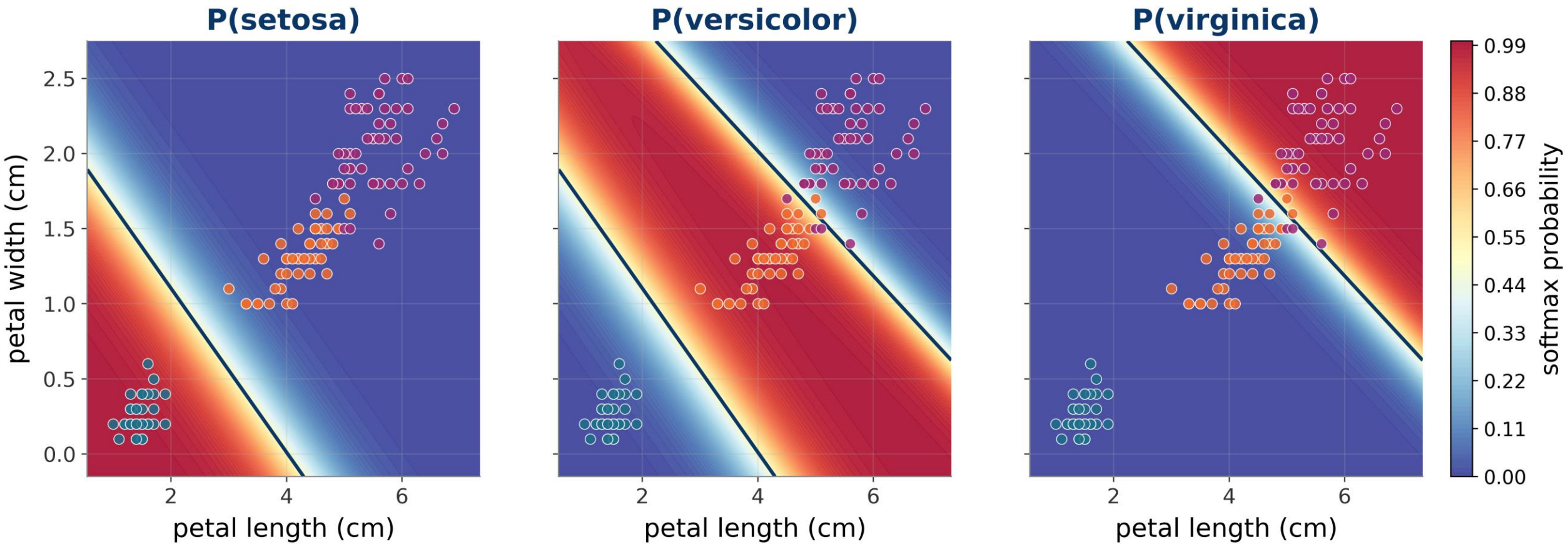
Binary cross-entropy

$$J(\boldsymbol{\beta}) == -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik} \log(p_{ik})$$

Same idea: Penalize the model when it assigns low probability to the true class

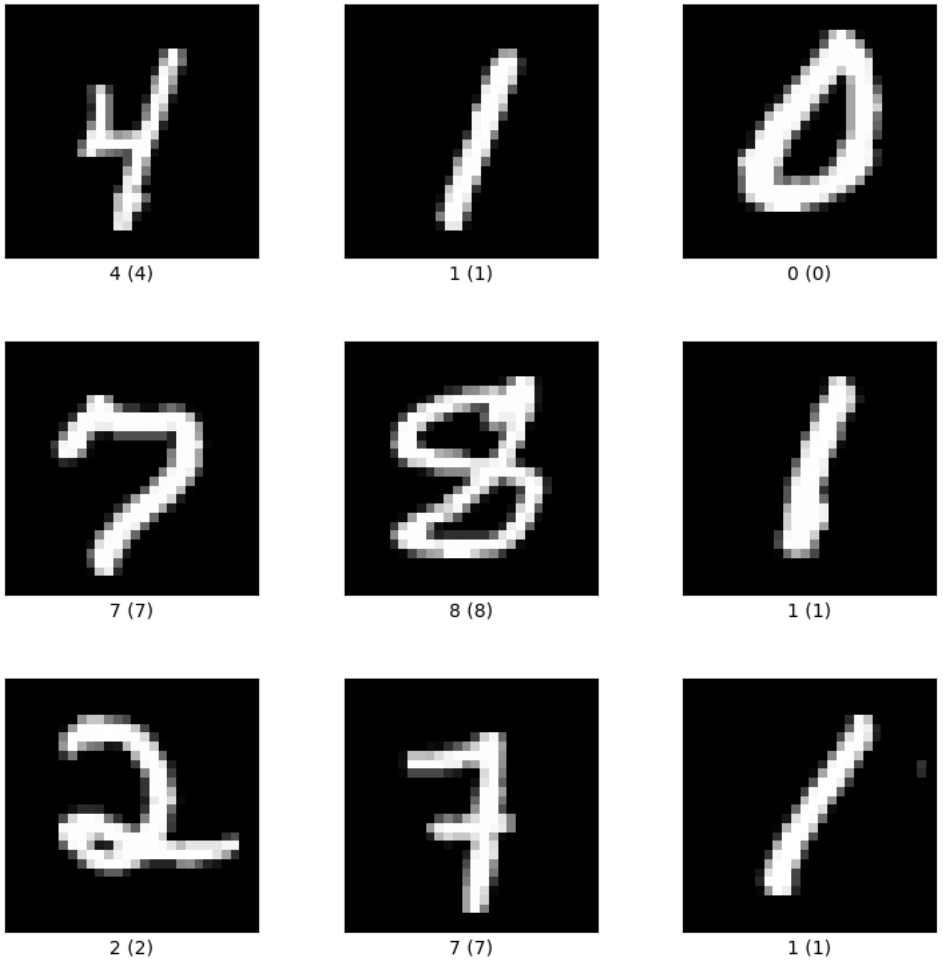
Now sum over all samples (n) *and* classes (K)

Results from Multinomial on the Iris dataset



Use multinomial logistic regression when probabilities should sum to 1

MNIST: Much better results than KNN!



	0	1	2	3	4	5	6	7	8	9
0	62	0	0	0	0	0	0	0	0	0
1	0	61	0	1	0	0	0	0	1	1
2	0	0	62	0	0	0	0	0	0	0
3	0	0	3	57	0	3	0	0	1	0
4	0	0	0	0	59	0	0	2	1	1
5	0	0	0	0	0	62	1	0	0	1
6	0	1	0	0	0	0	61	0	1	0
7	0	0	0	0	2	0	0	60	0	1
8	0	3	1	1	0	0	0	0	56	0
9	0	0	0	1	0	2	0	1	0	59

Multinomial logistic regression achieves 95% accuracy on a hold-out test set

Questions?

What to know for HW + Exams

- Distinguish **probability, odds, and log-odds**
- Understand the loss functions (log-likelihood and cross-entropy)
- Logistic regression calculations for simple binary cases
- How to **interpret coefficients β_i in logistic regression**
- Basics of single *vs.* multiclass logistic regression