Lecture #13: Classification

aka STAT109A, AC209A, CSCIE-109A

CS109A Introduction to Data Science

Pavlos Protopapas, Kevin Rader and Chris Gumb

Outline

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

Hypothesis Testing

Hypothesis testing is a formal process through which we evaluate the validity of a statistical hypothesis by considering evidence **for** or against the hypothesis gathered by **random sampling** of the data. The steps are:

1. State the hypotheses, typically a **null hypothesis**, H_0 and an **alternative hypothesis**, H_A, that is the negation of the former.
2. Choose a type of analysis, i.e. how to use sample data to evaluate the null hypothesis. Typically, this involves choosing a single test statistic.
3. **Sample** data and compute the test statistic.
4. Use the value of the test statistic (or the p-value) to either reject or not reject the null hypothesis.
5. Restate the conclusion in context of the problem.

p-value

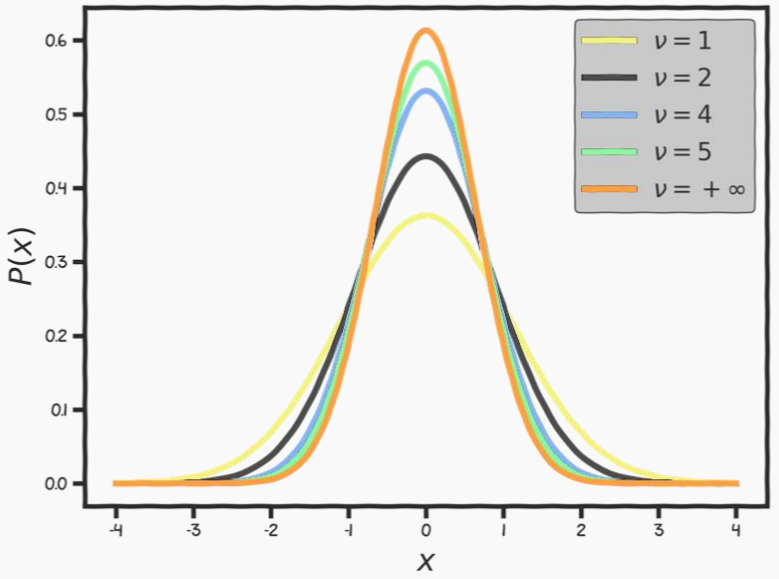
How extreme is extreme for a t -test statistic? We determine the probability of observing our test statistic, or a more extreme one, if the H_0 were true.

We call this probability the p-value:

$$p - value = P(|t_{df=n-p-1}| \geq |t - stat|)$$

Small p-value indicates that it is unlikely to observe such a substantial association between the predictor and the response due to chance. It is common to use p-value<0.05 as the threshold for significance.

To calculate the p-value we use the cumulative distribution function (CDF) of the student-t. `stats model` a python library has a build-in function `stats.t.cdf()` which can be used to calculate this.



Student's t-distribution, where ν is the degrees of freedom (number of data points minus (number of predictors + 1) = $n - (p + 1)$).

Hypothesis Testing via statsmodels

1. State Hypotheses:

$H_0: \beta_1 = 0, \qquad H_A: \beta_1 \neq 0$

2. Choose test statistic: (t-test)

3. Sample data and estimate:

$$t = \frac{\hat{\beta}_1 - 0}{\widehat{SE}\left(\hat{\beta}_1\right)} = \frac{0.5898}{0.023} = 25.211$$

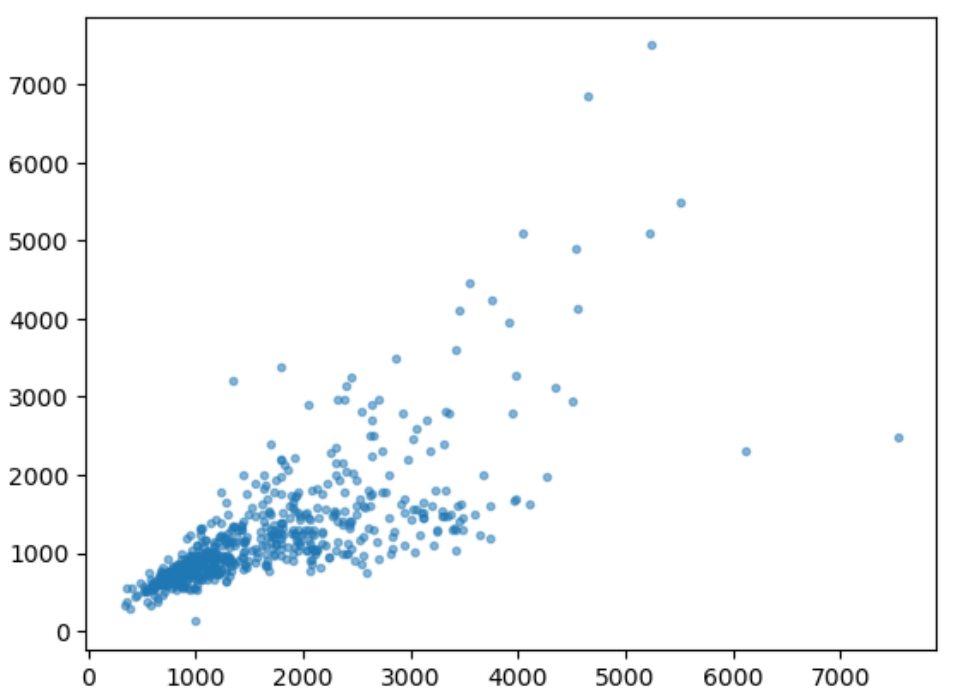
4. Reject or do not reject the H_0

$$p = P\left(\left|t_{df=n-2=590}\right| > 25.211\right) < 0.001$$

since p-value< 0.05, reject H_0

5. Restate the conclusion in context:

Evidence suggests that housing prices are truly [positively] associated with size of the home.



	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	247.4382	45.388	5.452	0.000	158.296	336.581
sqft	0.5898	0.023	25.211	0.000	0.544	0.636

Omnibus:	325.423	**Durbin-Watson:**	1.725
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	4390.598
Skew:	2.123	**Prob(JB):**	0.00
Kurtosis:	15.648	**Cond. No.**	3.95e+03

Permutation Tests: not just a side note

The **permutation test** allows us to directly use the data and statistic we care about but not have to bring along the baggage of distributional assumptions.

In classical hypothesis testing, the steps are:

1. State Hypotheses (H_0 and H_A)
2. Choose test statistic (often a t-test or z-test)
3. Sample/collect your data and estimate
4. Reject or do not reject the H_0 hypothesis
5. Restate the conclusion in context of the problem

The process for the permutation test is exactly the same, just the way the sampling distribution is built is through a form of resampling from the data rather than relying on probability theory.

Permutation Tests: the hypotheses

The most general form of the hypotheses:

H_0: The distribution of outcomes (Y) is not related to the value of X.

H_A: The distribution of outcomes (Y) is associated with the value of X.

Assumptions:

- Independence of observations.

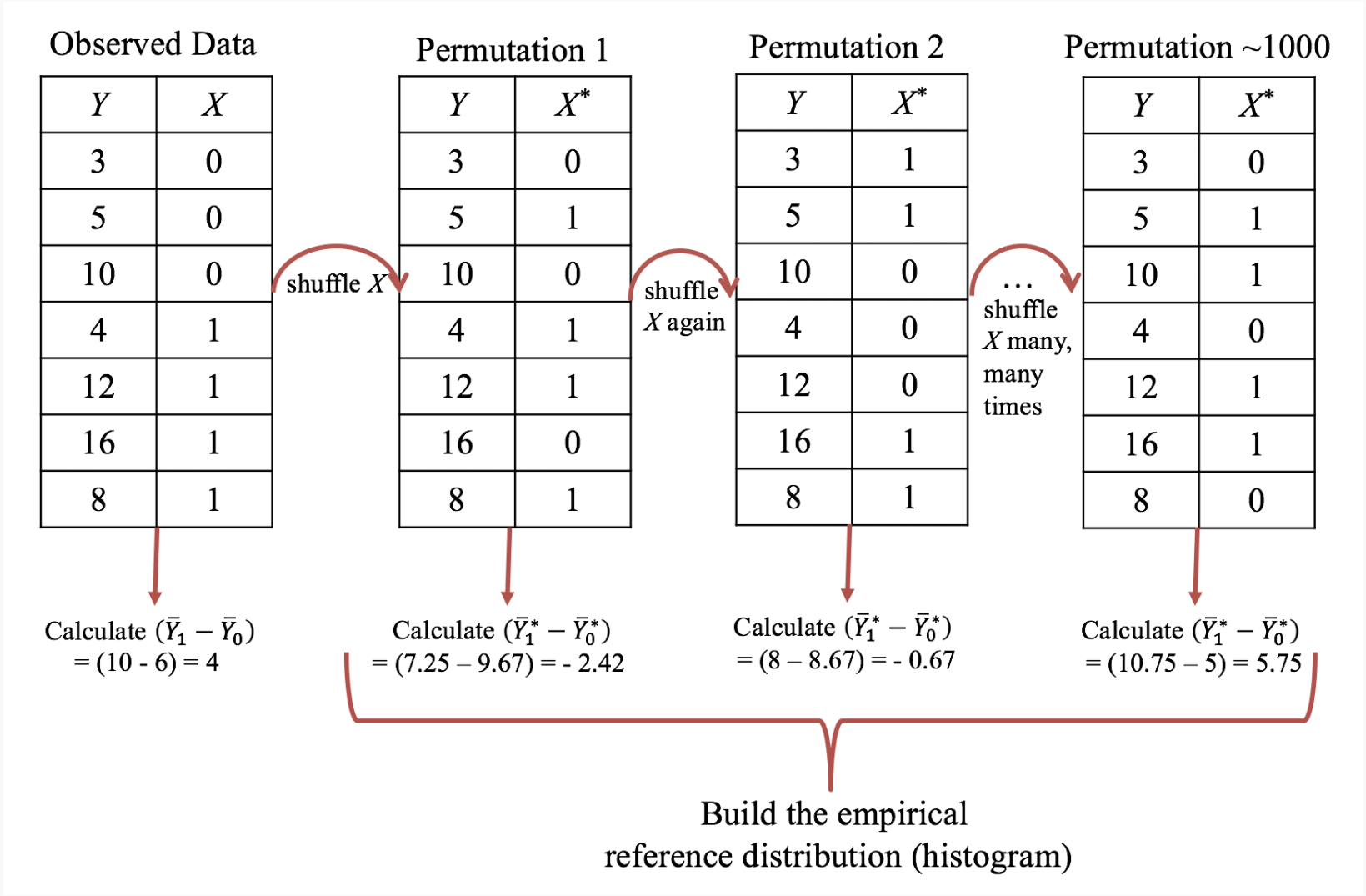
Test statistic: estimated slope, $\hat{\beta}_1$, between Y and X (or, any other statistic).

If you want a different set of hypotheses (comparison of means, comparison of medians, etc.), then that (i) will inform what test statistic to use and (ii) require you to possibly make more assumptions (similar shapes and spreads).

Permutation Tests: the main idea

- Just like in the bootstrap, the permutation test empirically creates a sampling distribution by using the data to build many potential **new** samples of data.
- But unlike the bootstrap, the permutation test relies on the null hypothesis (of **no effect** or **no difference**) when performing the resampling.
- In the two-sample case, if the null hypothesis is true, this results in **exchangeability** across observations: if the predictor variable were to change, this should have no bearing on the outcome/response measurement.
- In practice, all that you need to do is randomly 'redistribute' the responses across the predictors, thus preserving *n* but artificially forcing there to be no relationship between response (Y) and group status (X).

Permutation Tests: a diagram (for a binary predictor)



Permutation Tests: calculating the p-value

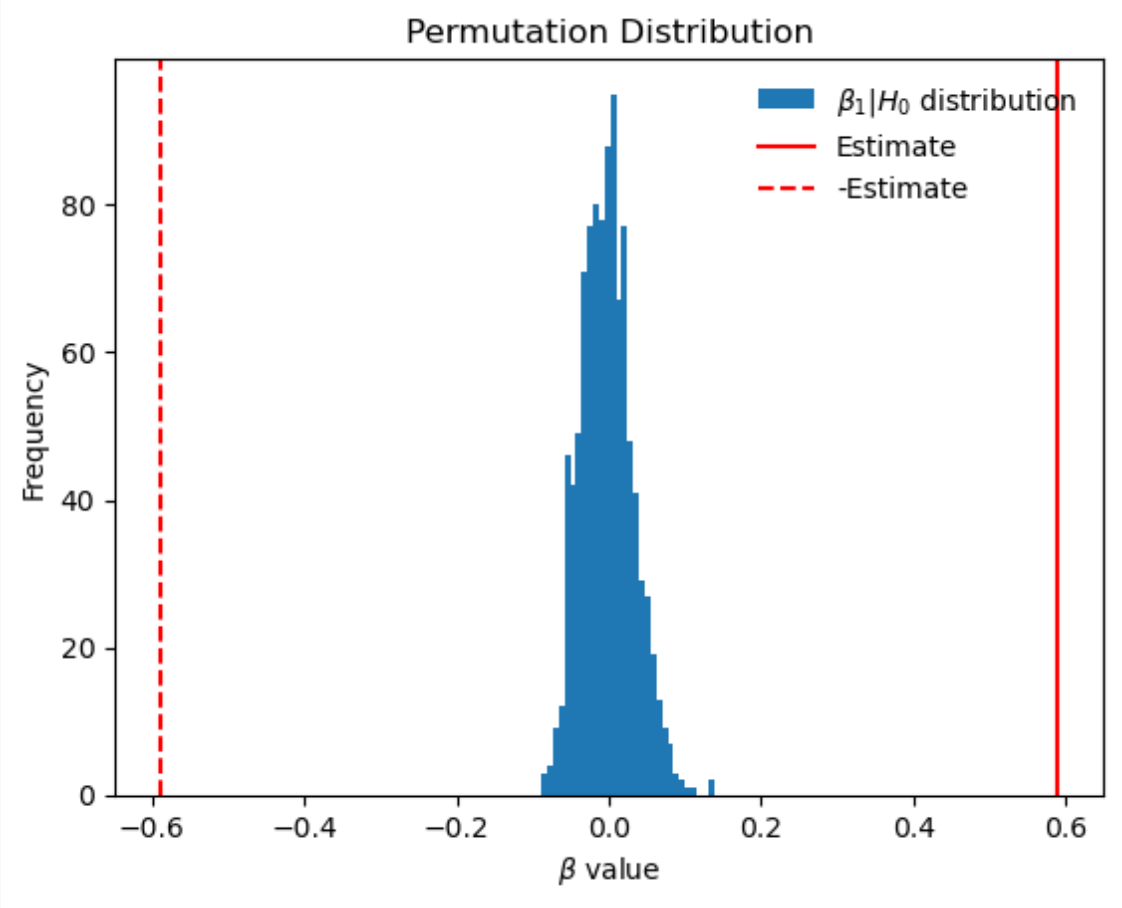
- Once the test statistic is calculated many, many times via re-permuting the observed responses across groups (via re-shuffling the group variable), then the observed statistic is compared for its **extremity** in that permutation reference/sampling distribution, in empirical probability terms. More specifically, we calculate how often the permuted statistic (calculated under the null) is more extreme than what was actually observed in the data.

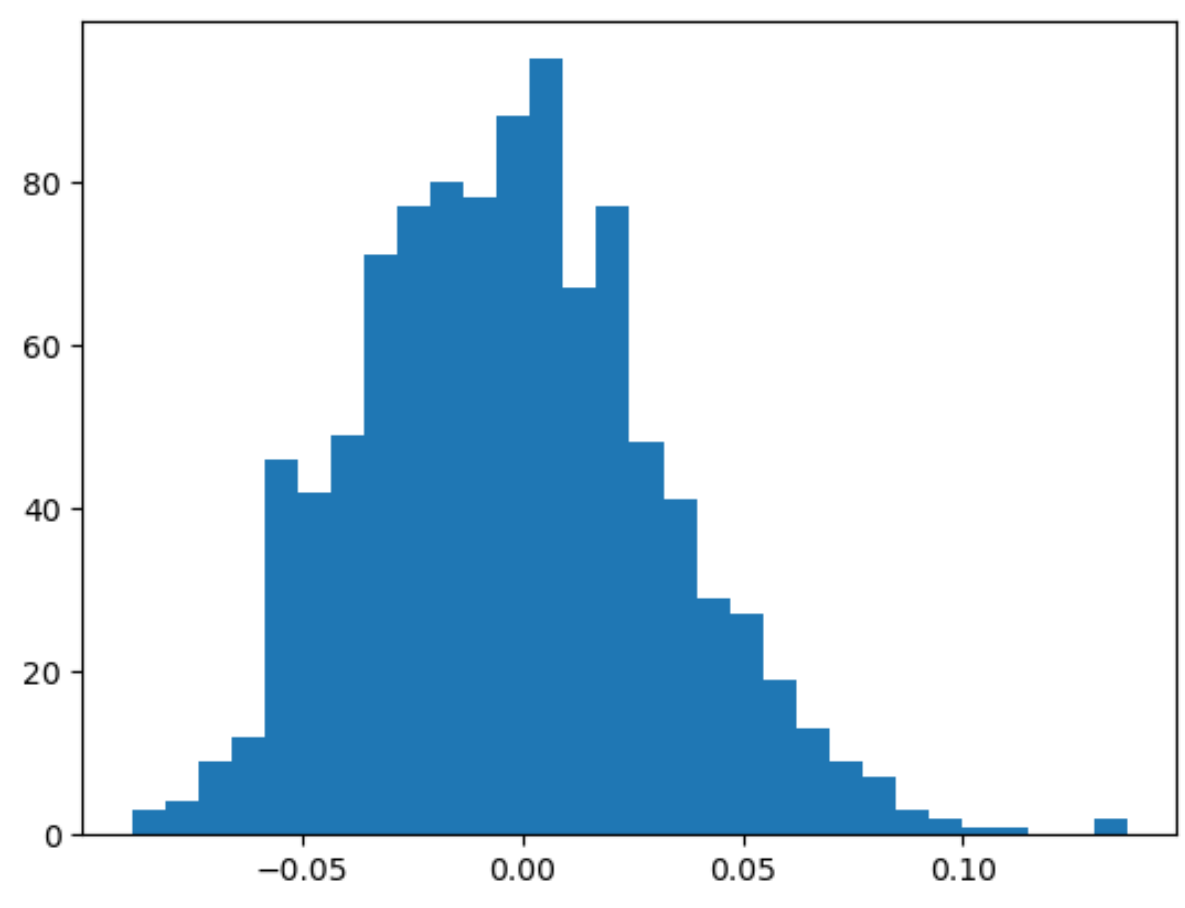
- Note: there is no distributional assumptions about the observations! Thus, the permutation test is an alternative to the -test if normality looks iffy.

Permutation Tests: an example

```
#permuations
nsims = 1000
X = homes['sqft']
y = homes[['price']]
indices = np.arange(0,len(homes))
beta1_permute = []
for i in np.arange(0,nsims):
    np.random.shuffle(indices)
    y_permute = y.iloc[indices]
    permute_ols = sk.linear_model.LinearRegression().fit(X = homes[['sqft']], y = y_permute)
    beta1_permute.append(permute_ols.coef_[0][0])
```

Permutation Tests: reference distribution





There's a package for that (in scipy):
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.permutation_test.html

Permutation vs. bootstrap

What are the main differences between the bootstrap and permutation methods to performing resampling?

- **The goal:** bootstrapping is done to estimate (like calculate CIs) while permutation-ing is done to test a specific hypothesis.

- **The implementation:** just like in classic methods, estimation (bootstrapping) is performed without a null hypothesis and thus uses an approach the relies on whatever world the data lives in (presumably H_A), while hypothesis testing (permutation testing) is performed strictly assuming the null hypothesis is true.

That perspective on the implementation is why the bootstrapping just samples from the observed data directly (not caring about Type I or Type II error), while permutation testing carefully resamples under the null condition (thus attempting to preserve Type I error).

Note: 'inverting' a bootstrapped CI to perform testing is reasonable…but could lead to inflated (or deflated) Type I error.

Interpreting Interactions

Write out the model statement:

Interpret the coefficient estimates:

Does it appear that there is truly an interaction effect? [tricky]

```
interaction_ols = smf.ols(formula = "price ~ sqft * type",
                          data = homes).fit()
interaction_ols.summary()
```

OLS Regression Results

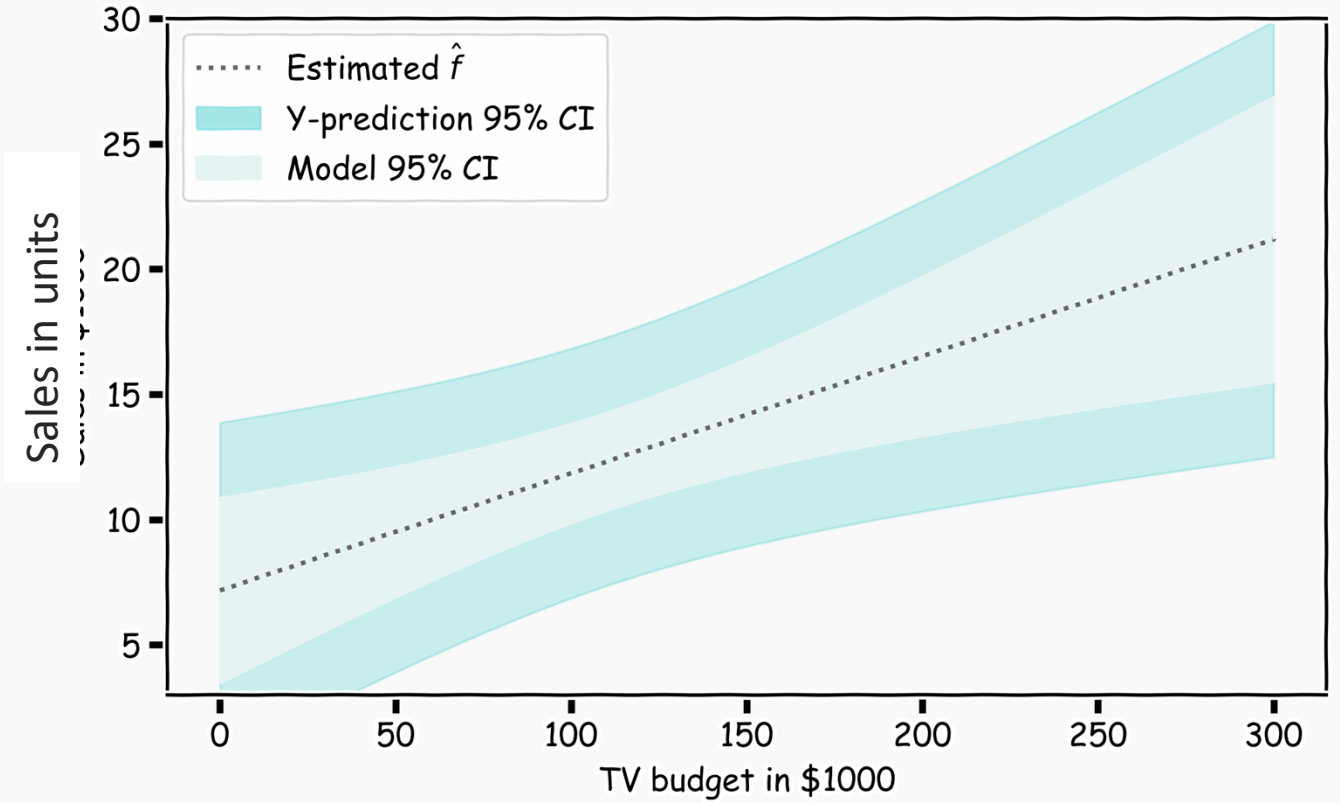
Dep. Variable:	price	**R-squared:**	0.738
Model:	OLS	**Adj. R-squared:**	0.734
Method:	Least Squares	**F-statistic:**	234.5
Date:	Wed, 15 Oct 2025	**Prob (F-statistic):**	4.70e-165
Time:	07:19:03	**Log-Likelihood:**	-4386.6
No. Observations:	592	**AIC:**	8789.
Df Residuals:	584	**BIC:**	8824.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	170.5182	52.997	3.217	0.001	66.430	274.606
type[T.multifamily]	142.0626	154.846	0.917	0.359	-162.060	446.185
type[T.singlefamily]	-708.8103	111.900	-6.334	0.000	-928.586	-489.035
type[T.townhouse]	-34.2107	191.736	-0.178	0.858	-410.787	342.365
sqft	0.6659	0.040	16.516	0.000	0.587	0.745
sqft:type[T.multifamily]	-0.2863	0.062	-4.615	0.000	-0.408	-0.164
sqft:type[T.singlefamily]	0.4769	0.057	8.298	0.000	0.364	0.590
sqft:type[T.townhouse]	0.0543	0.107	0.509	0.611	-0.155	0.264

Omnibus:	196.241	**Durbin-Watson:**	1.842
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1512.812
Skew:	1.248	**Prob(JB):**	0.00
Kurtosis:	10.423	**Cond. No.**	2.56e+04

Uncertainty in predicting a new Y

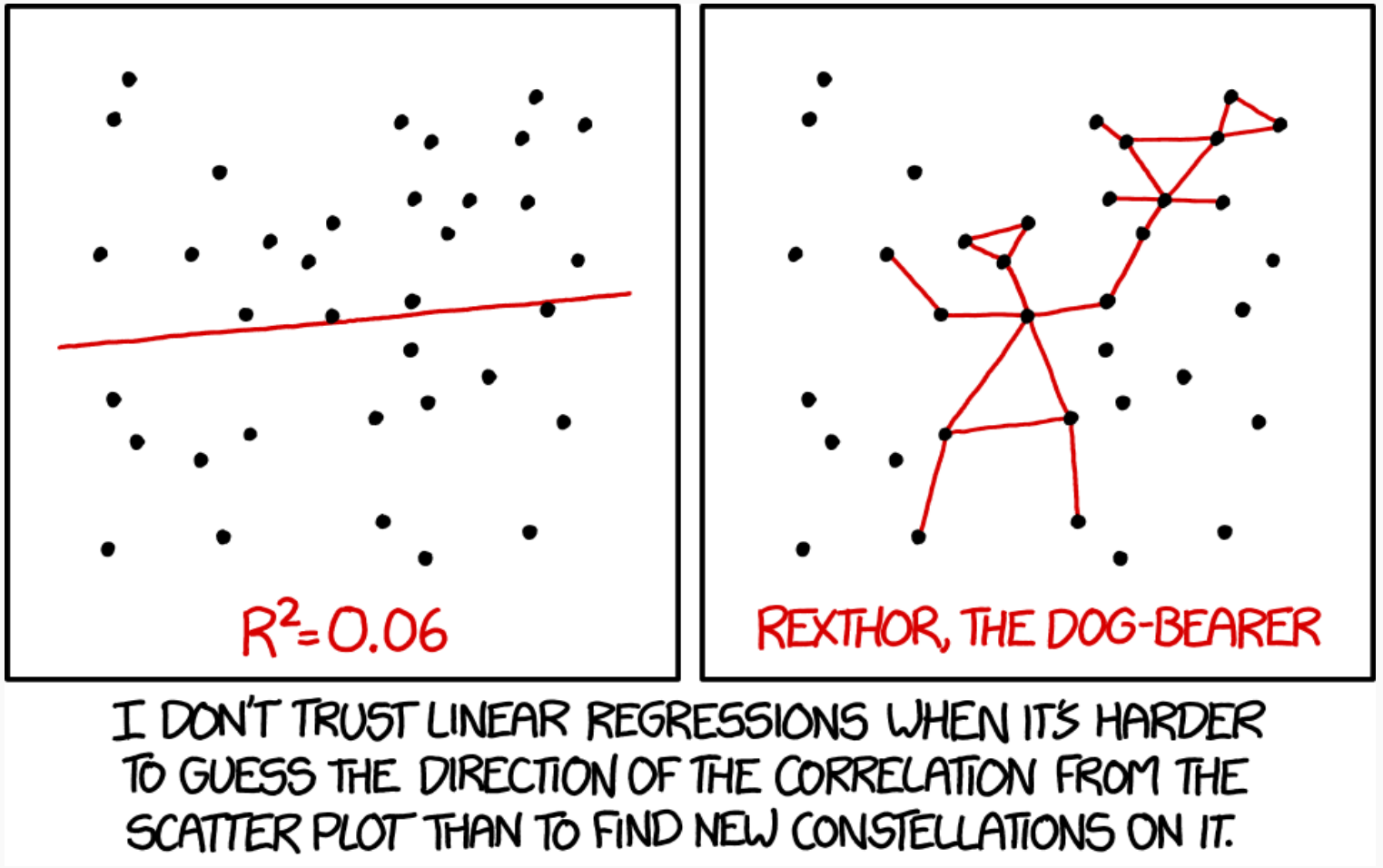
- For a given x, we have a distribution of models $f(x)$
- For each of these $f(x)$, the prediction for $y \sim N(f(x), \sigma_\epsilon)$
- The prediction intervals are then …



Confidence in predicting $\hat{y}$

Even if we knew $f(x)$, the response value cannot be predicted perfectly because of the random error in the model (irreducible error).

How much will Y vary from $\hat{Y}$? We use prediction intervals to answer this question.

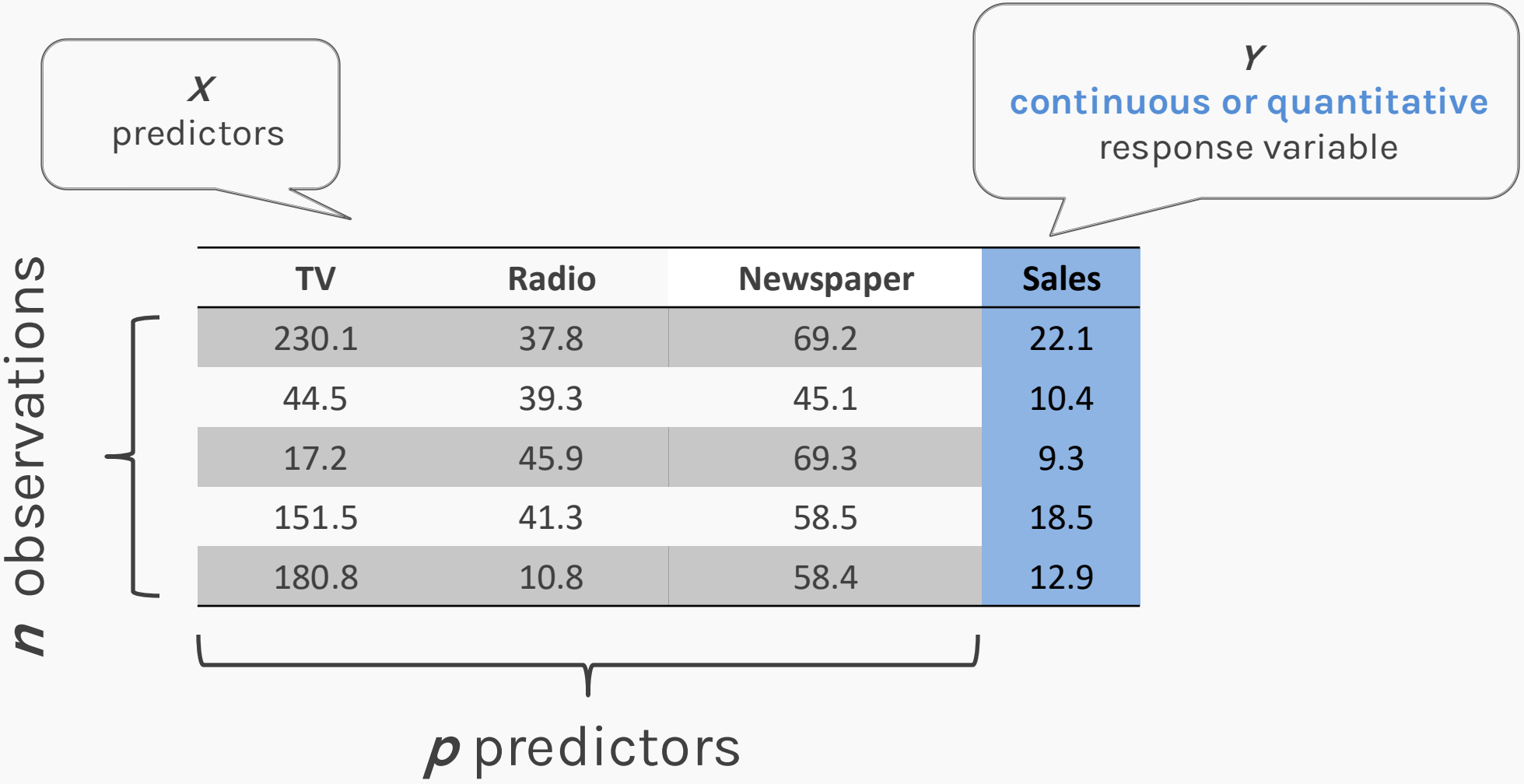


Outline

- Review
- **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)



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

What is Classification?

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

X
predictors

Y
Yes or No
response variable

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	reversable	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

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	reversable	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

No indicates **absence** of heart disease

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	reversable	Yes
37	1	nonanginal	130	250	187	0	normal	No

What is Classification?

In summary,

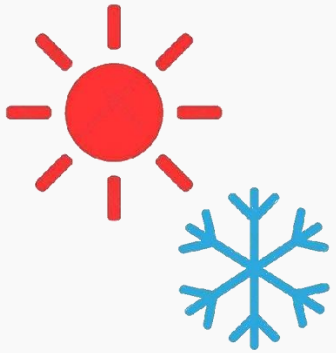
Regression

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

- Review
- 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.

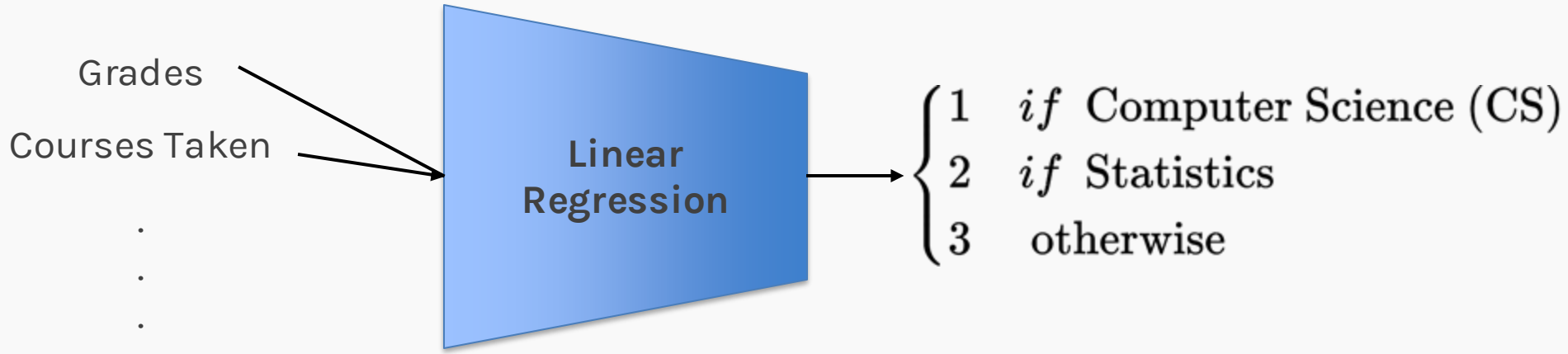
This categorical variable can't be used as is, but it could be encoded to be quantitative.

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

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

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.

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

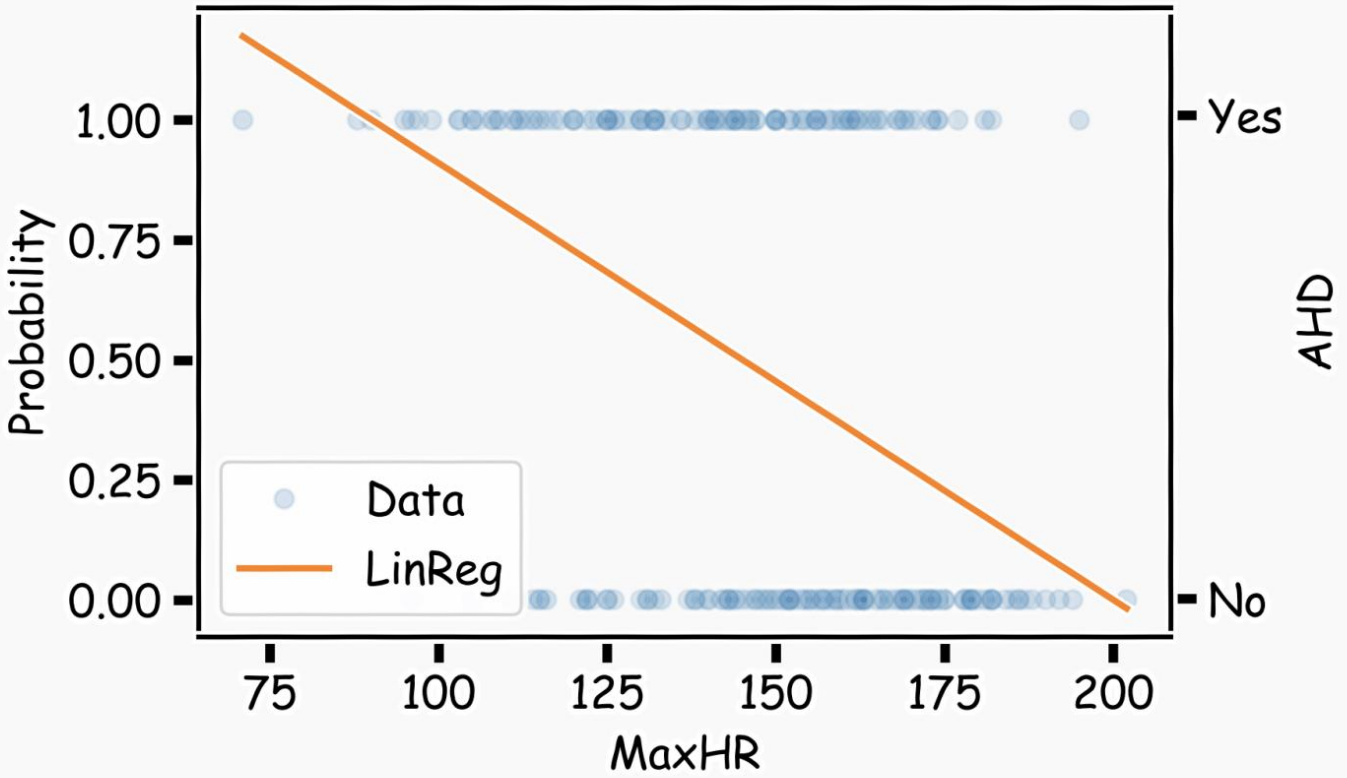
- Linear regression could be used to predict the probability $P(y = 1)$ directly from a set of predictors such as sex, cholesterol levels, etc.

- 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.

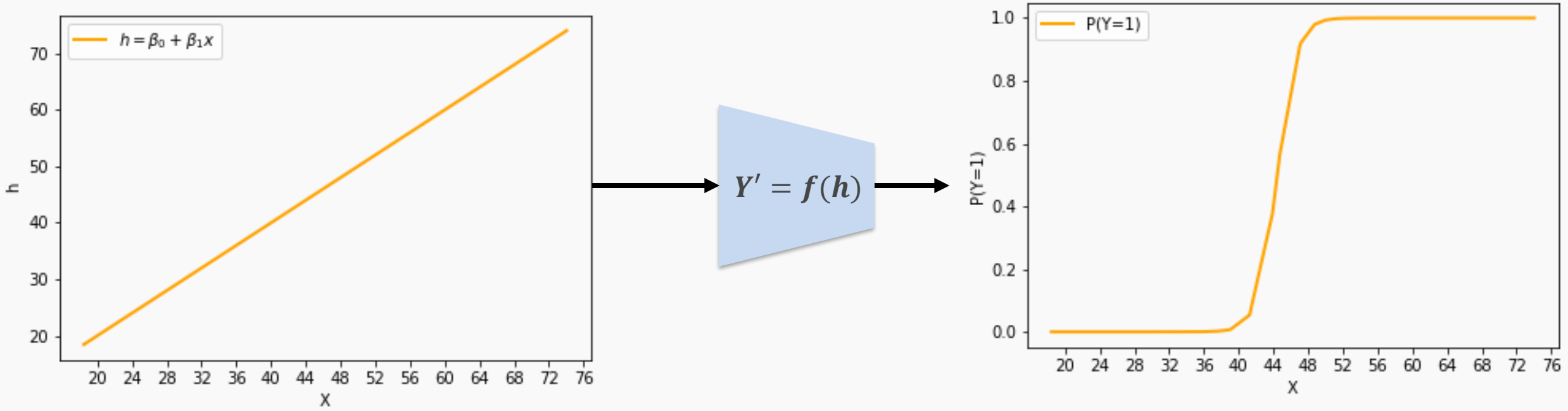


Outline

- Review
- 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:

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

Logistic Regression

- Logistic Regression addresses the problem of estimating a probability, $P(y = 1)$, to be outside the range of $[0,1]$.

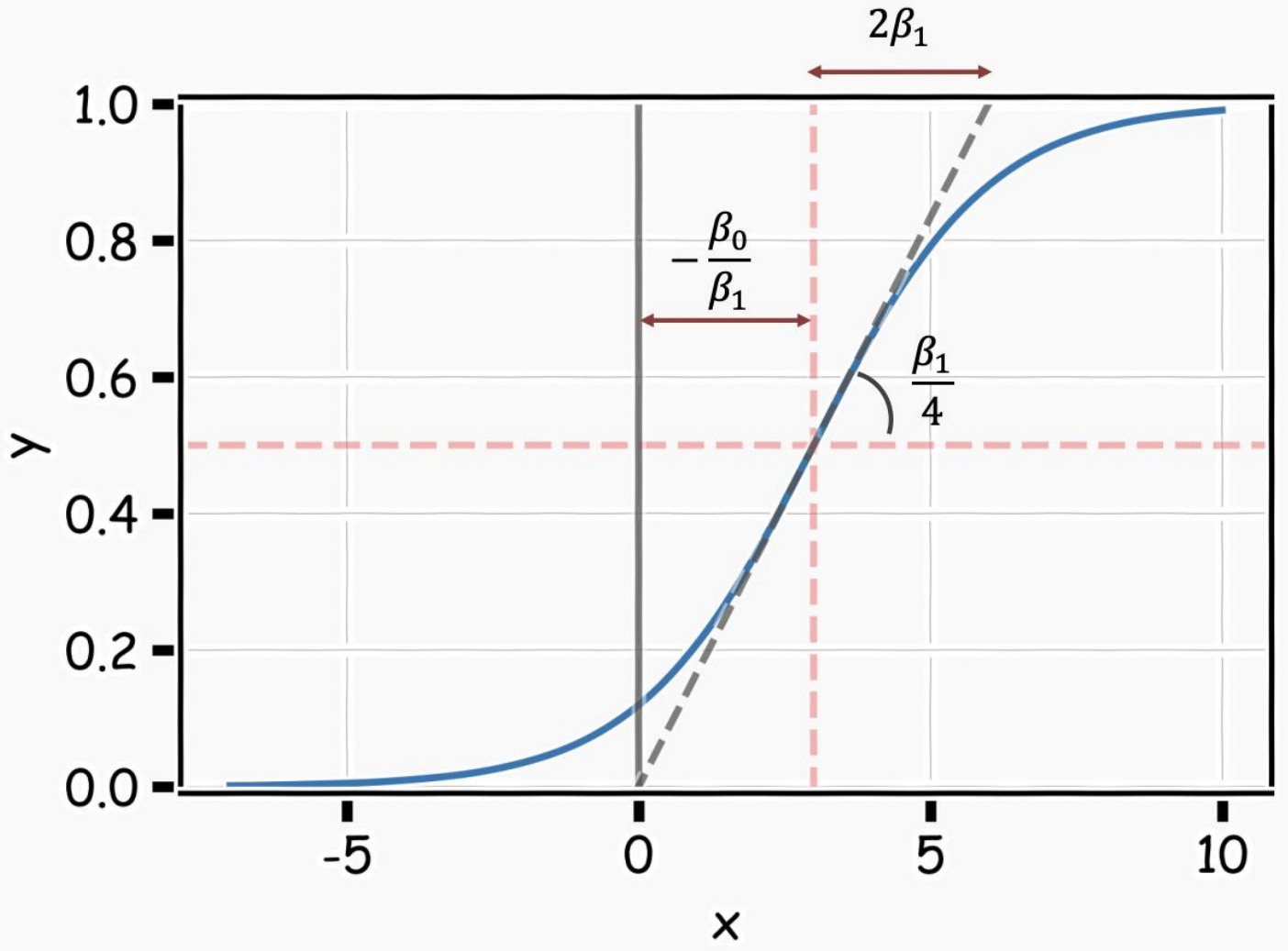
- The logistic regression model uses a function, called the **logistic** function, to model $P(y = 1)$:

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

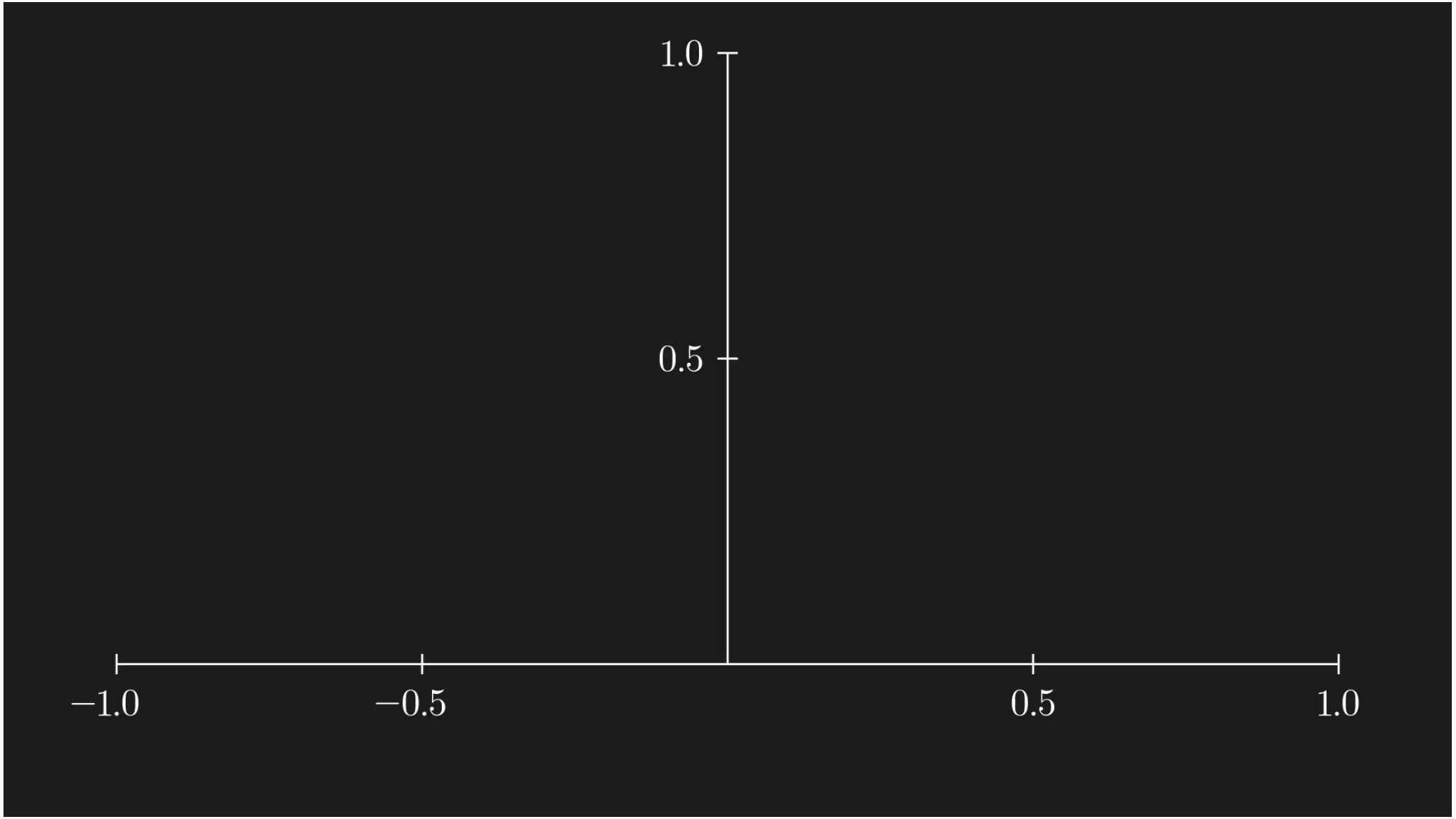
What are the parameters of this model?

Logistic Regression

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



Sigmoid Animation



Interpretation of β's

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.

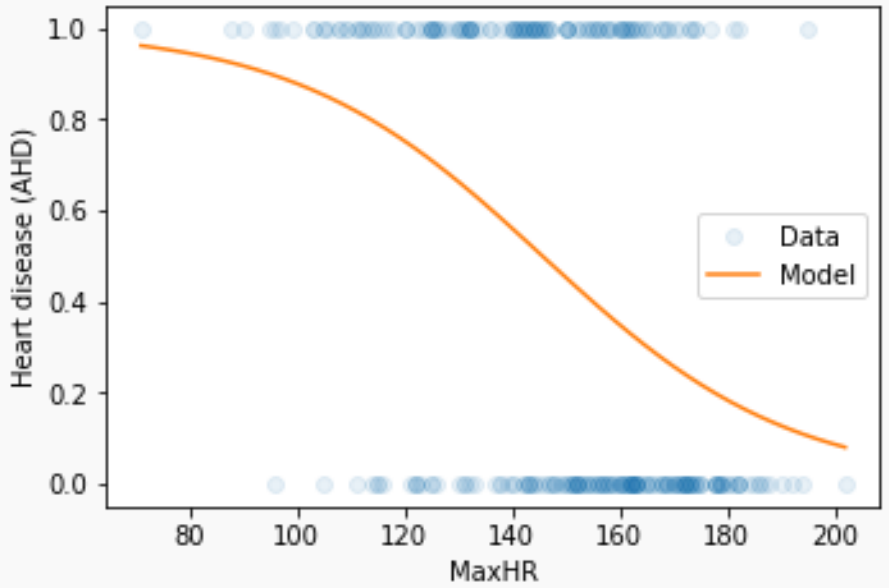
A one unit change in X is associated with a β_1 change in the log-odds of $\mathrm{P}(Y=1)$; or better yet, a one unit change in X is associated with an e^{β_1} change in the odds that $Y=1$.

A First Logistic Regression Model in sklearn

Here is a logistic regression output to predict Y = AHD from X = 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?

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

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?

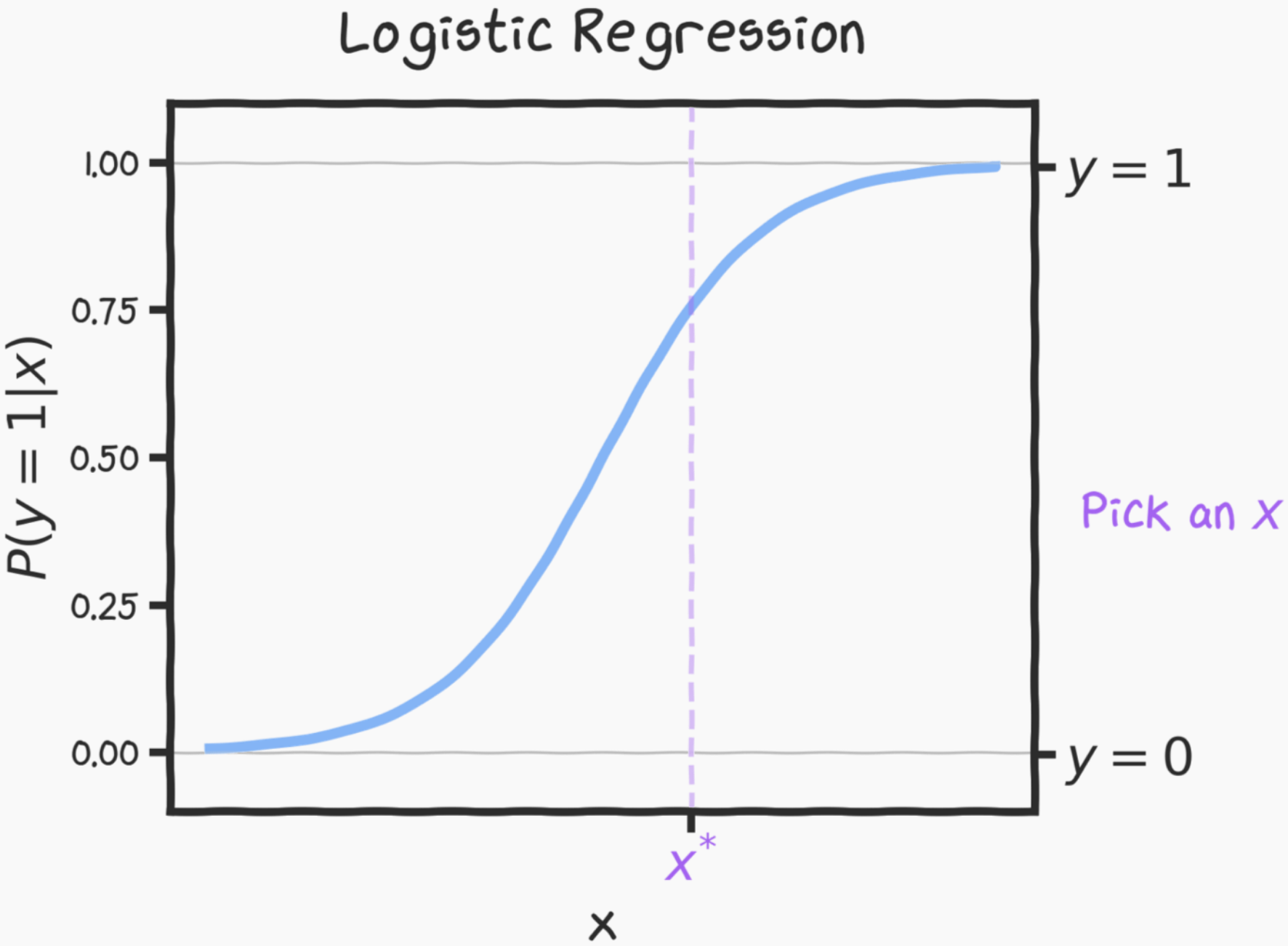
  MSE

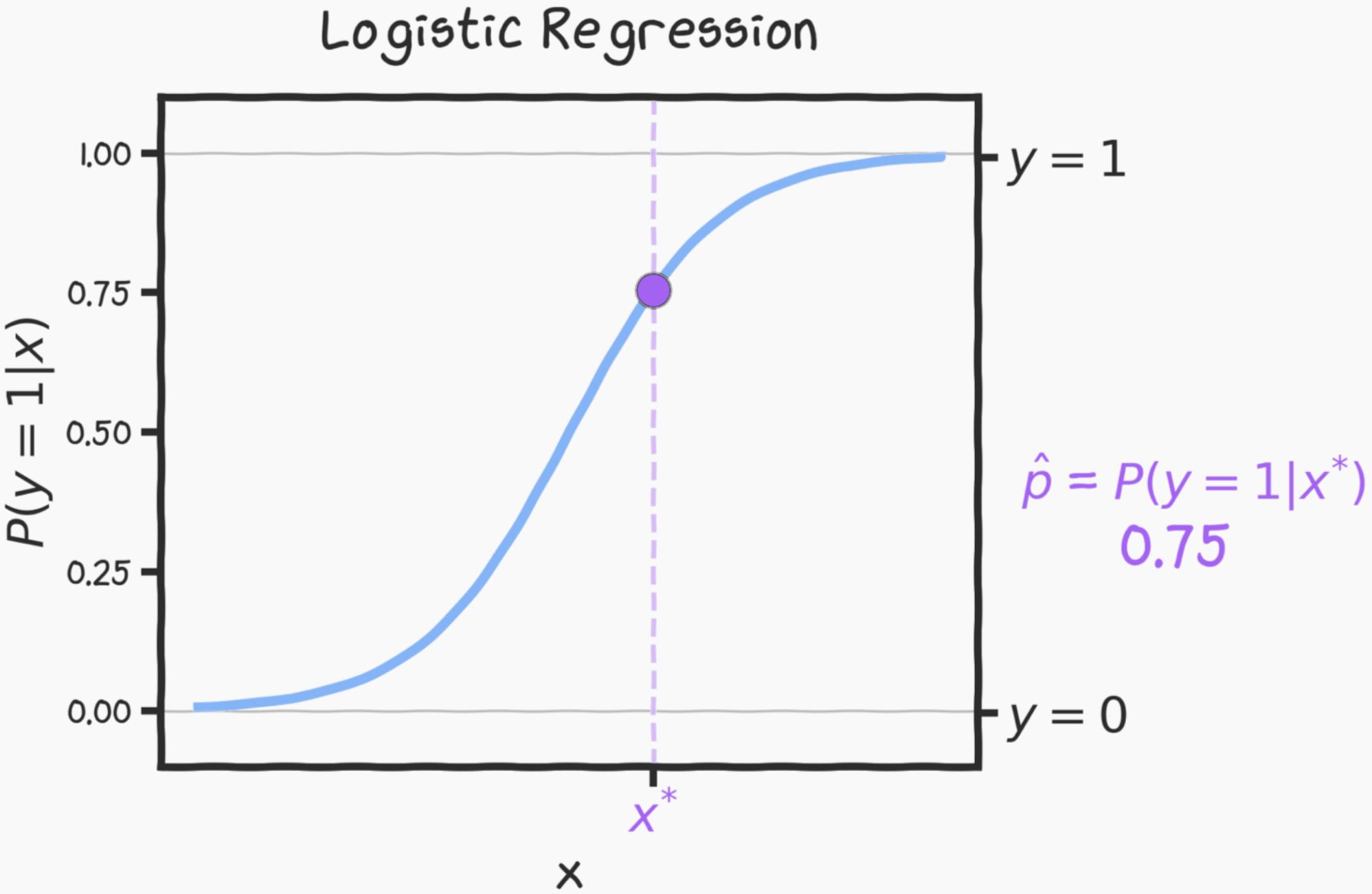
- What was the probabilistic perspective on linear regression?

  Normal Distribution

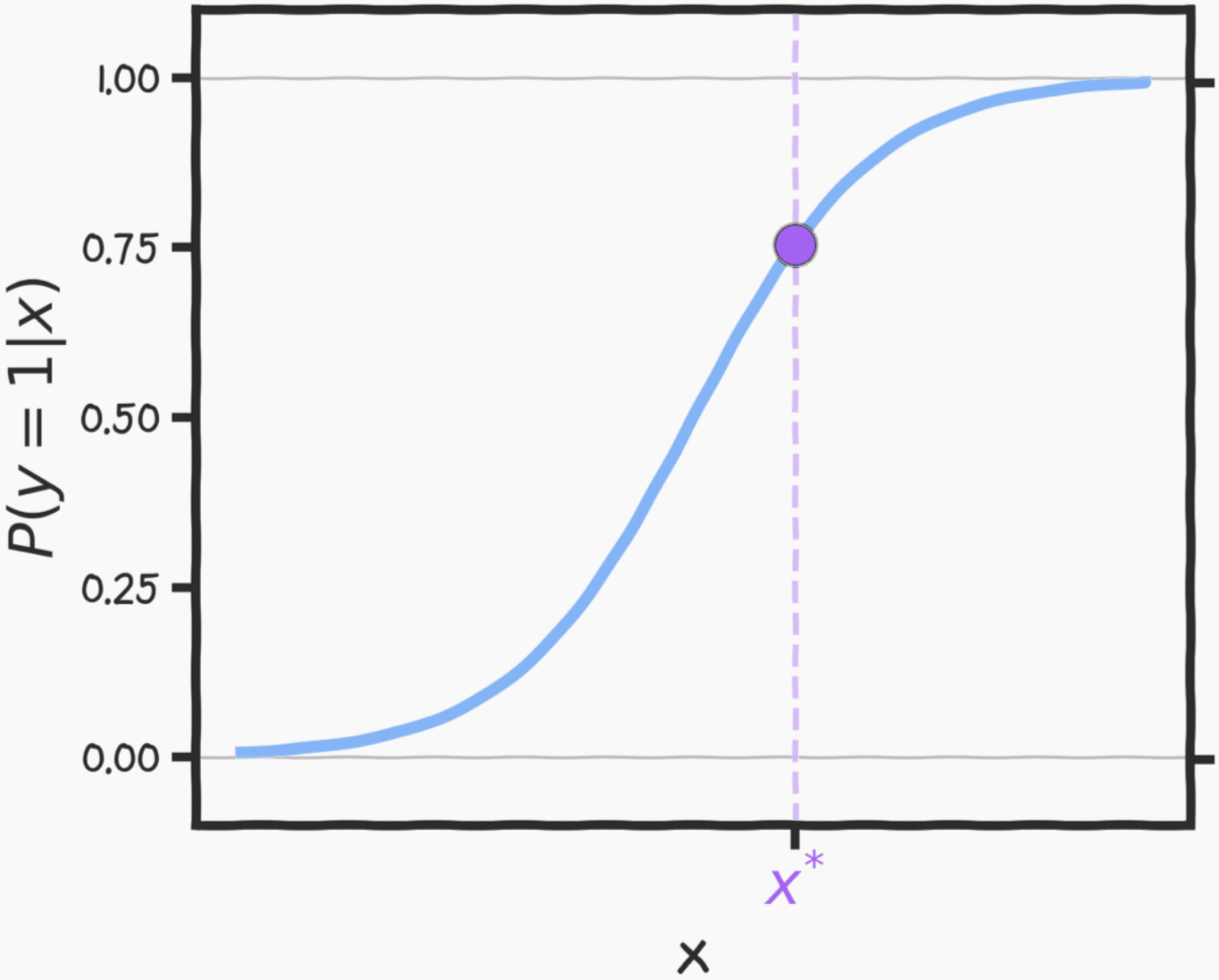
Logistic Regression also has a likelihood-based approach to estimating parameter coefficients.


Logistic Regression
1.00
0.75
0.50
0.25
0.00
$P(y = 1|x)$
x
$y = 1$
$y = 0$


Logistic Regression
1.00
0.75
0.50
0.25
0.00
$P(y=1|x)$
$y=1$
$y=0$
Pick an x
x^*
x


Logistic Regression
1.00
0.75
0.50
0.25
0.00
$P(y=1|x)$
$y=1$
$y=0$
x^*
x
$\hat{p} = P(y=1|x^*)$
0.75

Logistic Regression

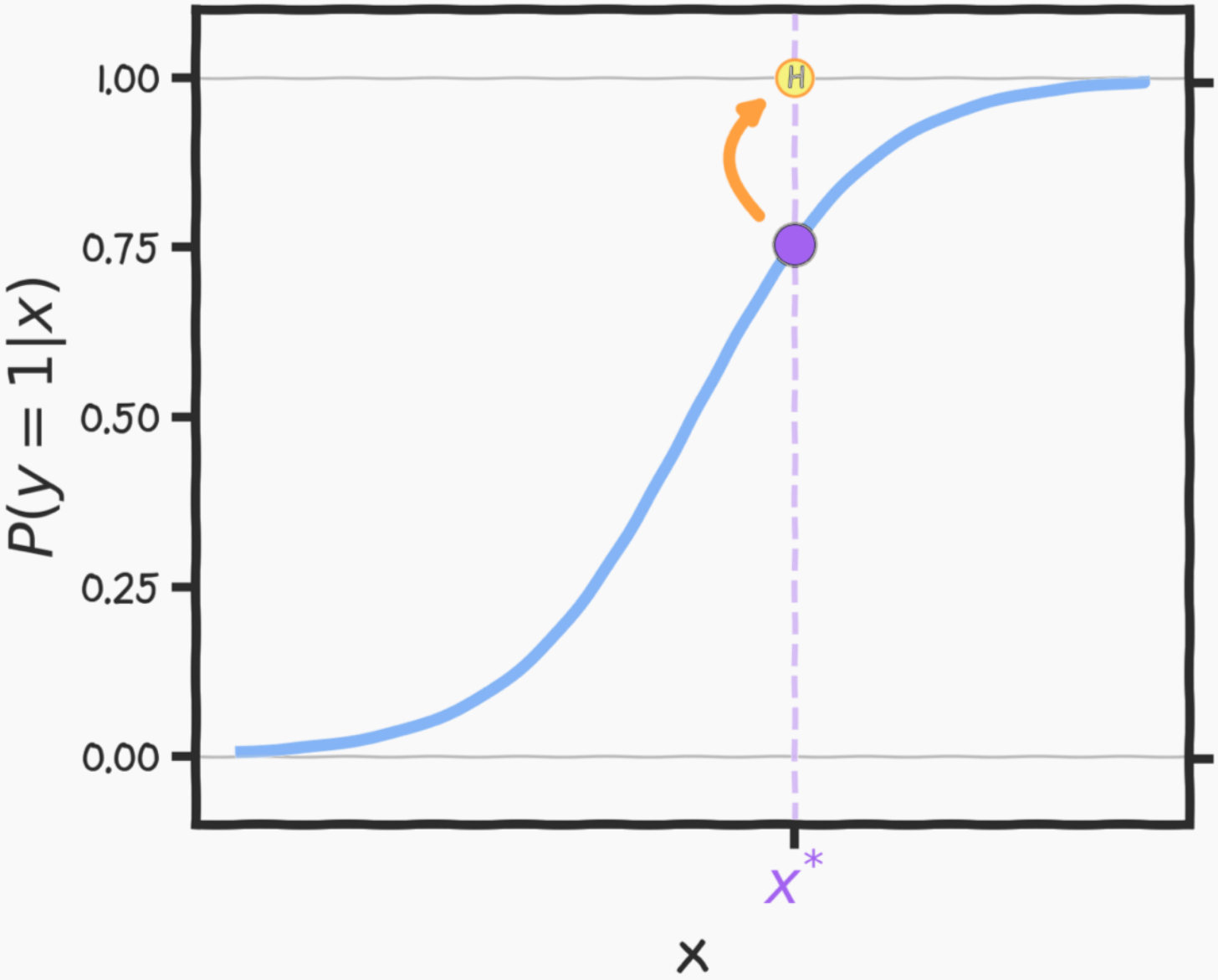


Flip a Coin with a
$\hat{P}$ Probability of Heads

$\hat{p} = P(y=1|x^*)$

0.75

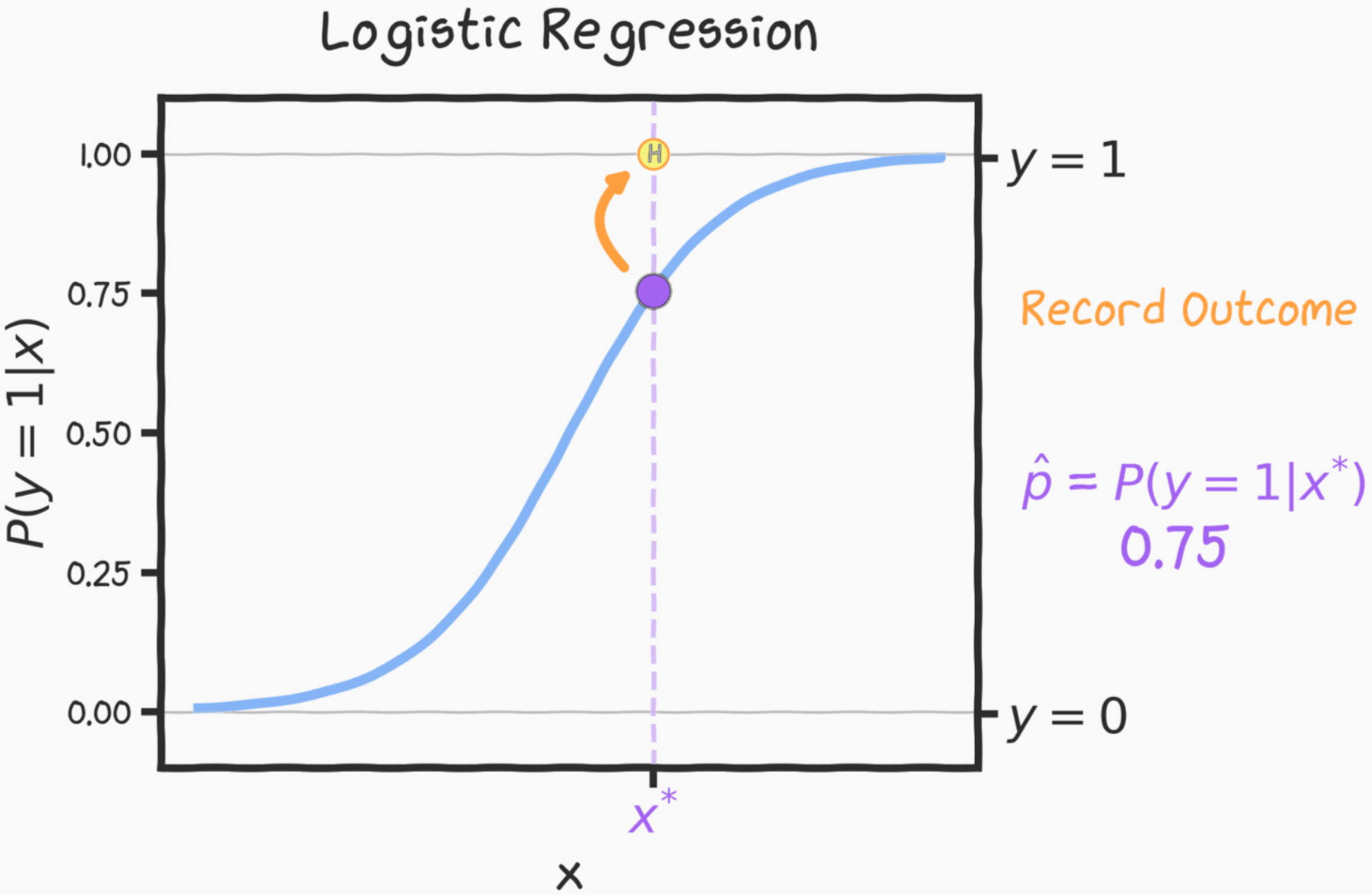
Logistic Regression

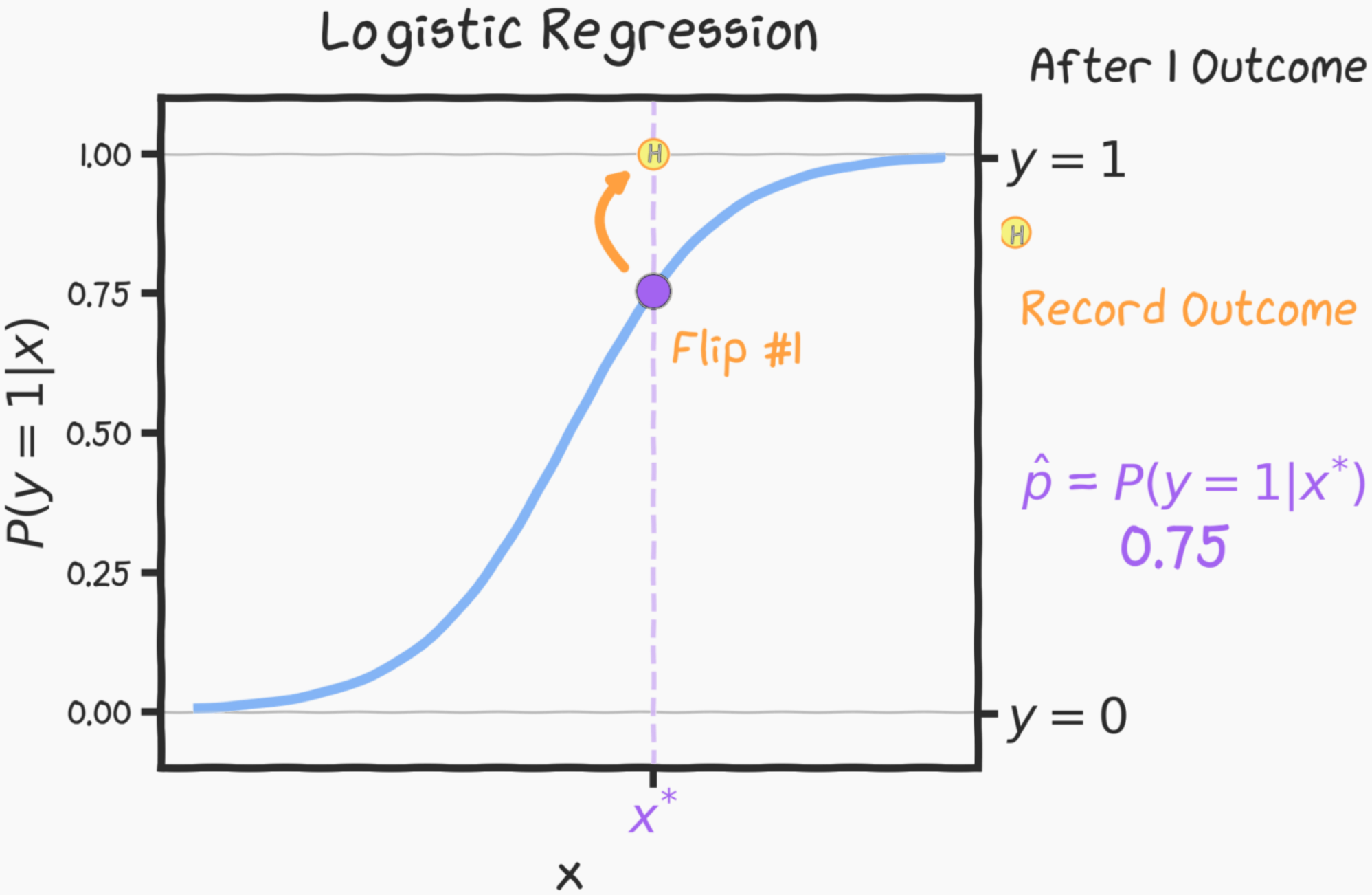


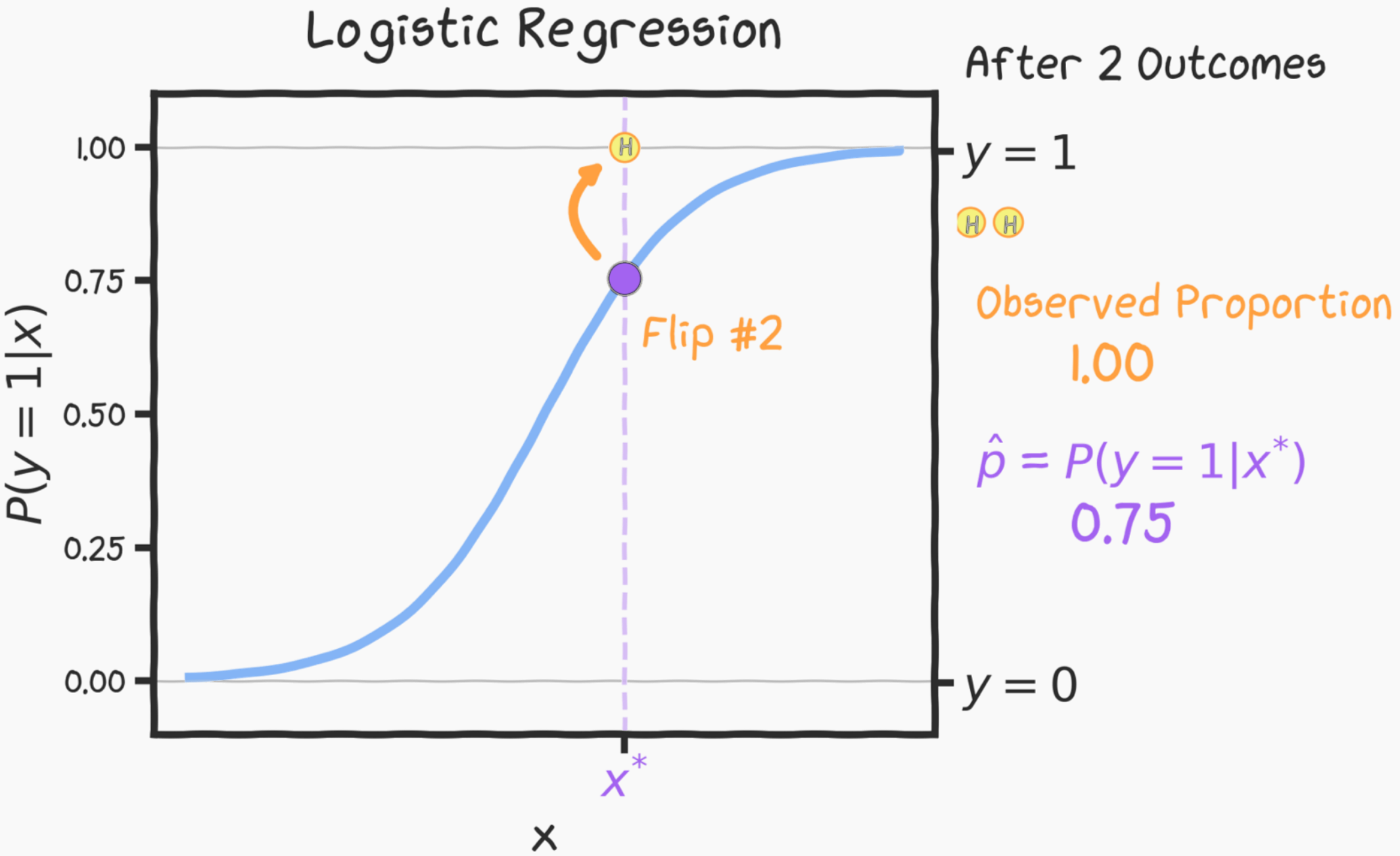
Flip a Coin with a
$\hat{P}$ Probability of Heads

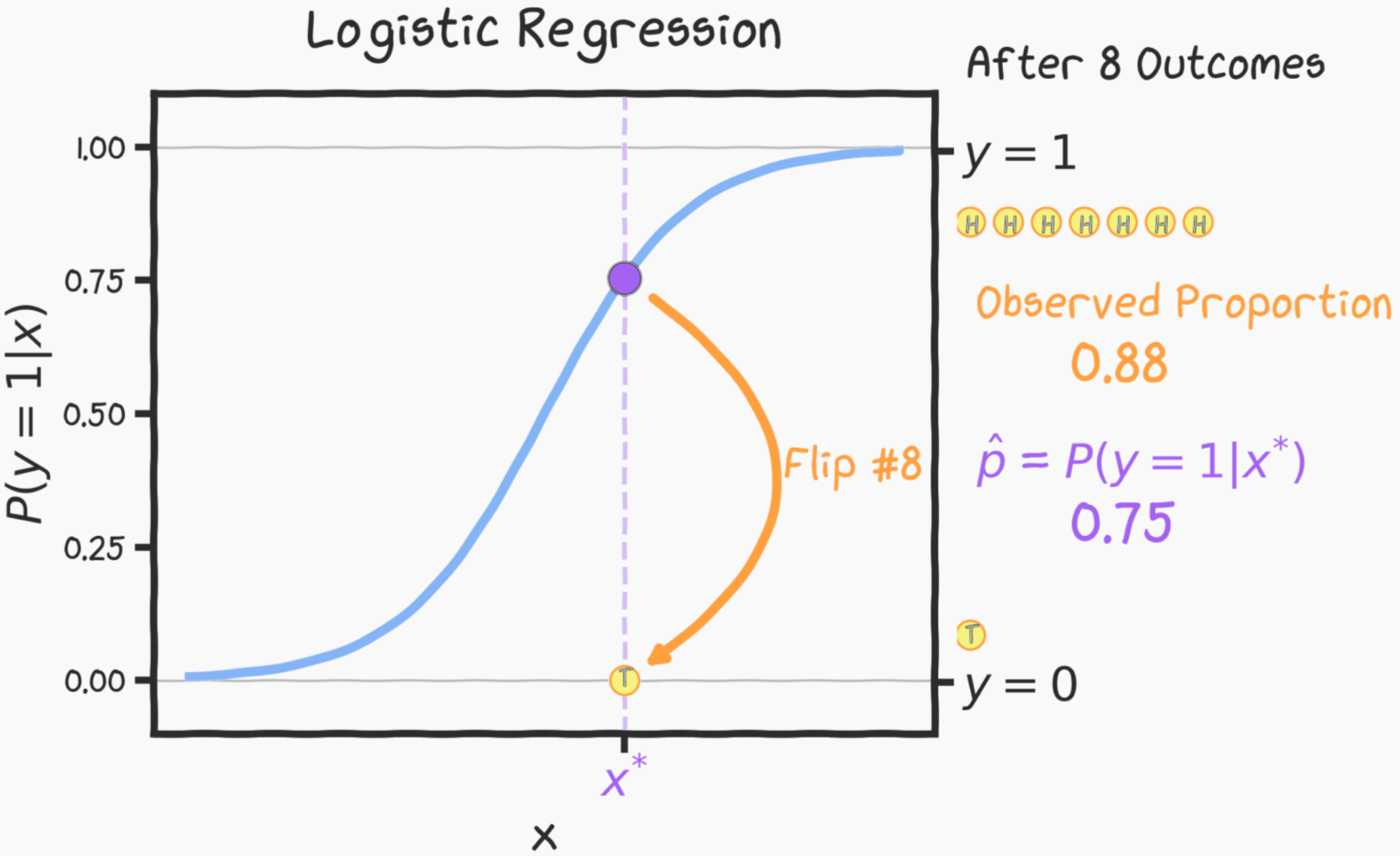
$\hat{p} = P(y=1|x^*)$

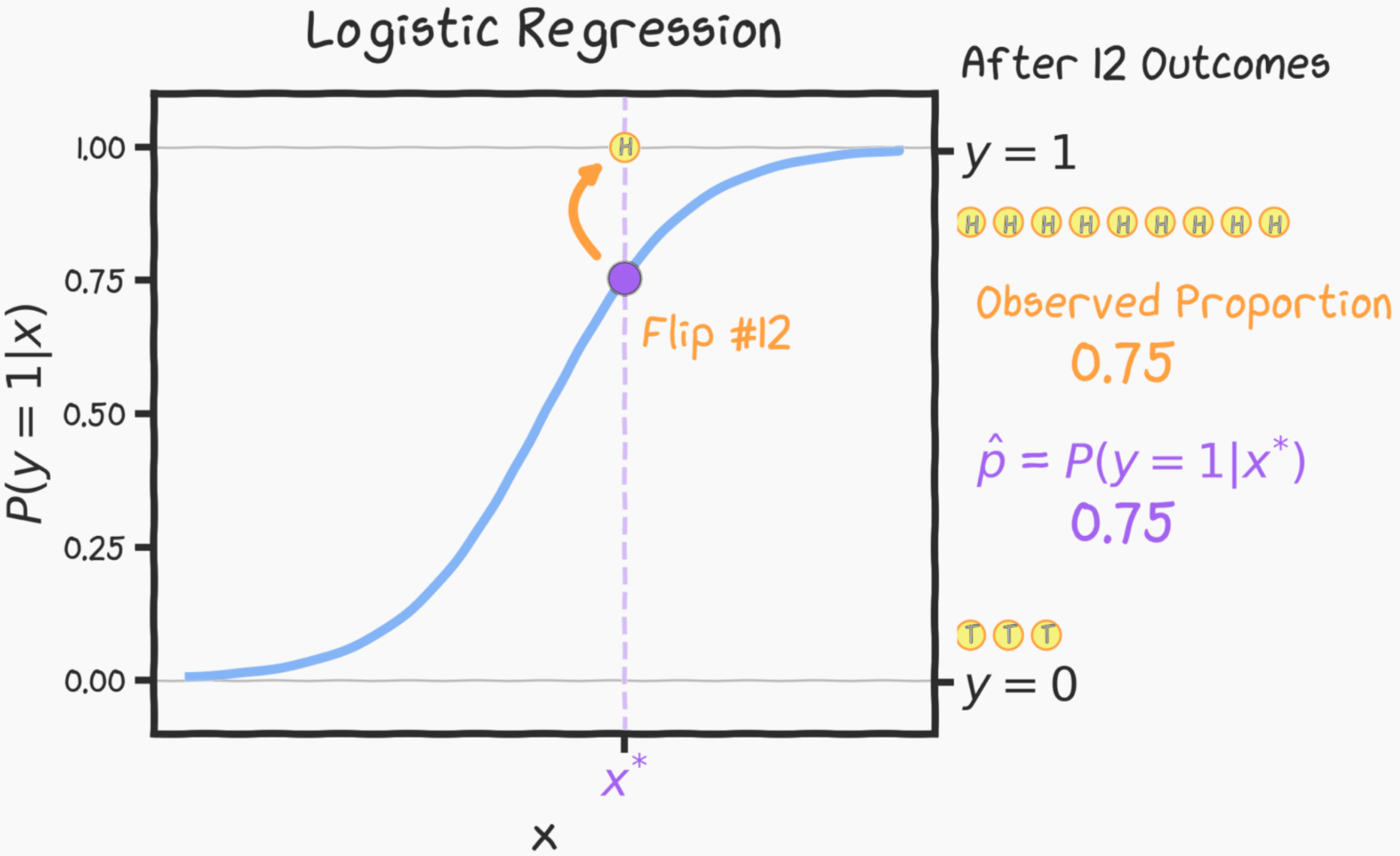
0.75

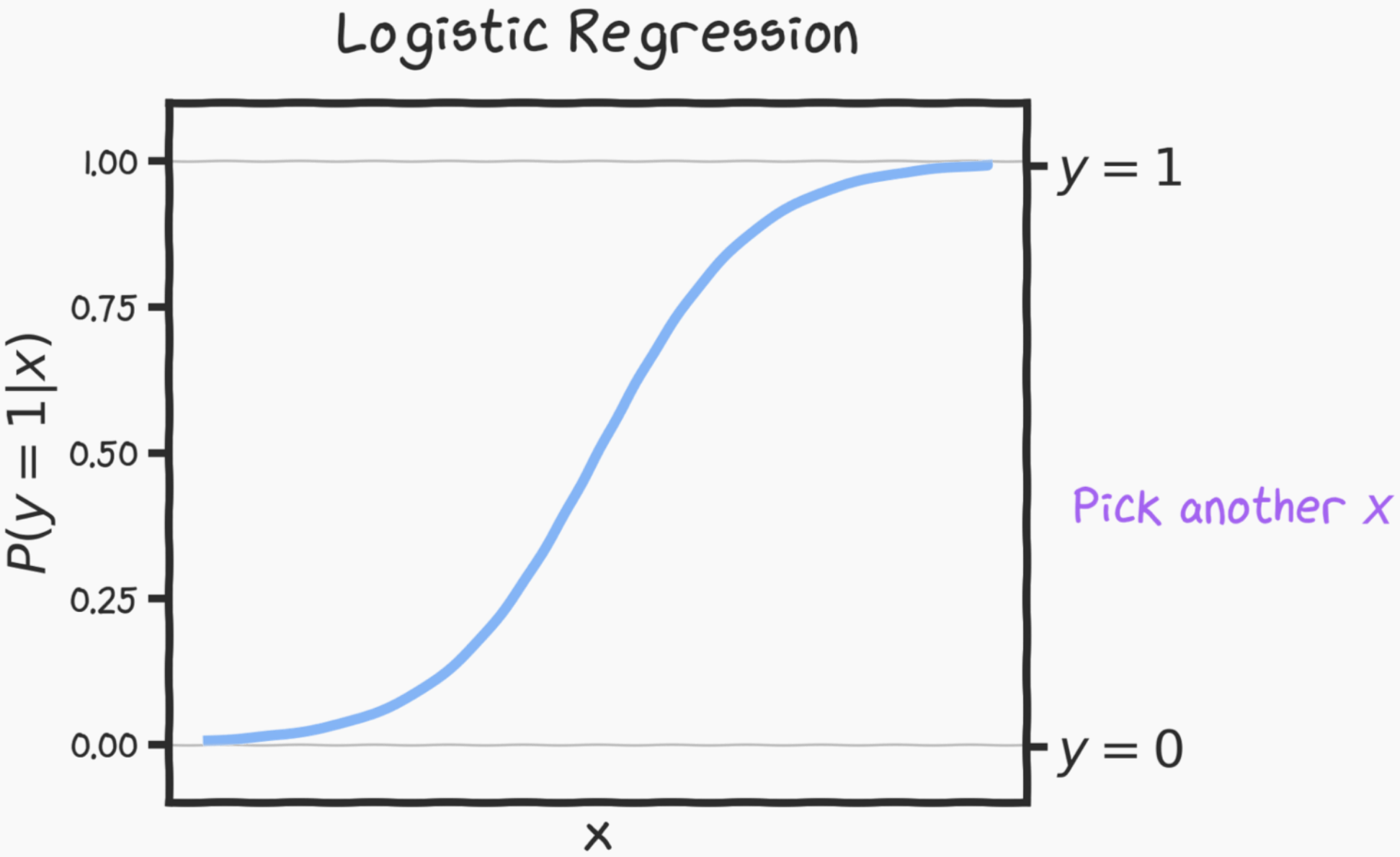

Logistic Regression
1.00
0.75
0.50
0.25
0.00
$P(y=1|x)$
H
$y=1$
$y=0$
Record Outcome
$\hat{p} = P(y=1|x^*)$
0.75
x^*
x

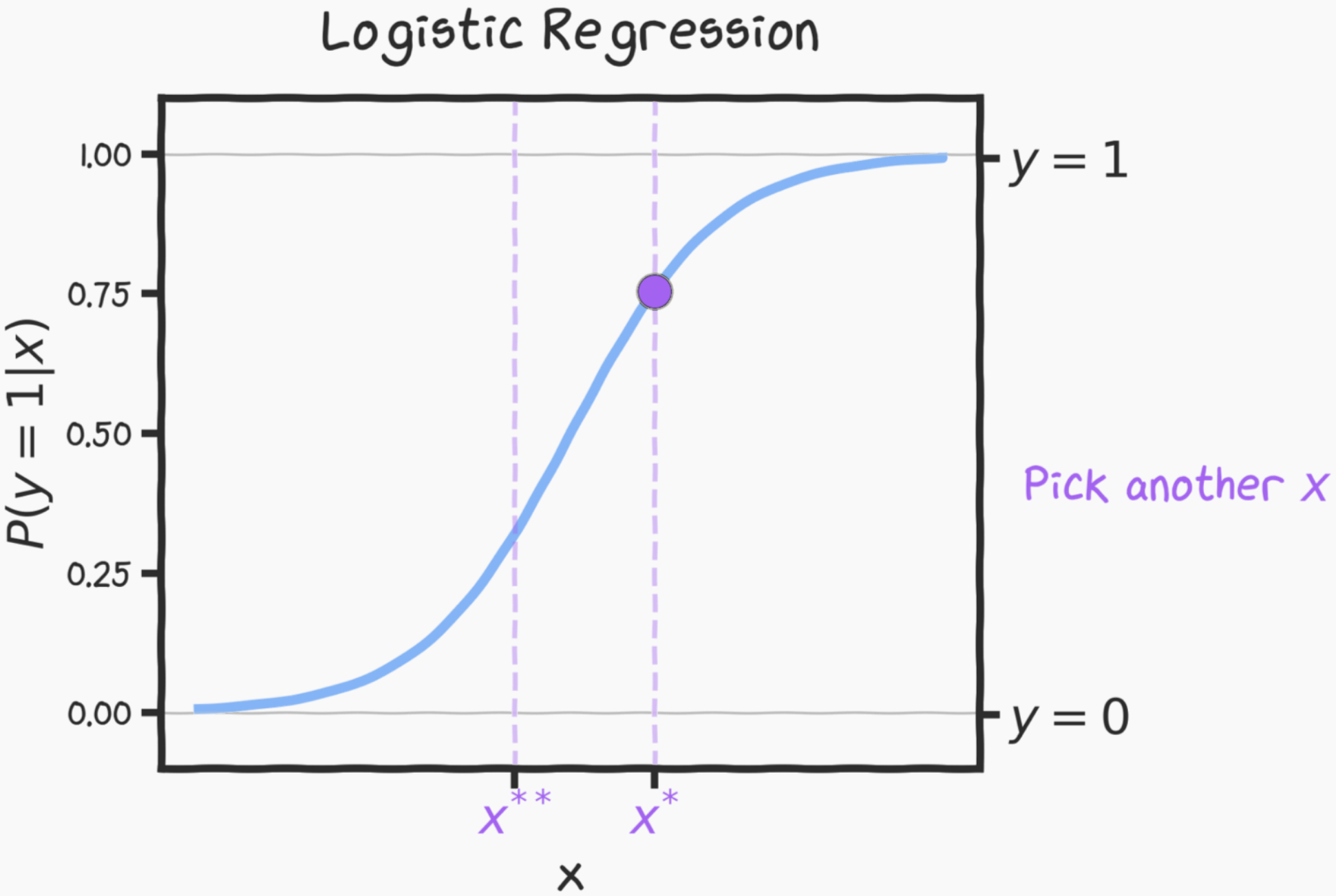

Logistic Regression
After 1 Outcome
1.00
0.75
0.50
0.25
0.00
$P(y=1|x)$
$y=1$
$y=0$
H
H
Flip #1
Record Outcome
$\hat{p} = P(y=1|x^*)$
0.75
x^*
x


Logistic Regression
After 2 Outcomes
1.00
0.75
0.50
0.25
0.00
$P(y=1|x)$
$y=1$
$y=0$
H
H H
Flip #2
Observed Proportion
1.00
$\hat{p} = P(y=1|x^*)$
0.75
x^*
x

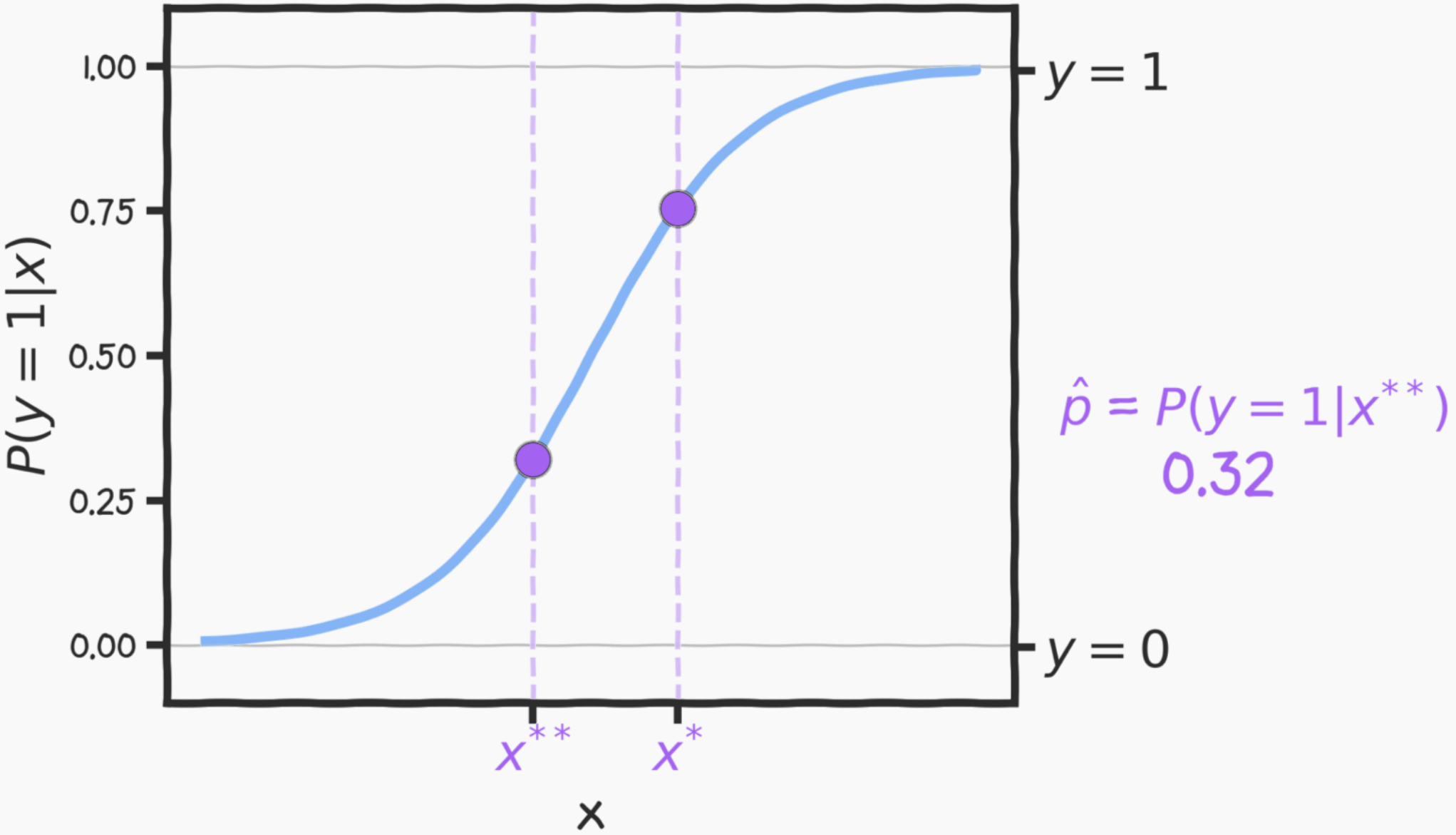

Logistic Regression
After 8 Outcomes
1.00
0.75
0.50
0.25
0.00
$P(y=1|x)$
$y=1$
H H H H H H H
Observed Proportion
0.88
Flip #8
$\hat{p} = P(y=1|x^*)$
0.75
T
$y=0$
x^*
x


Logistic Regression
After 12 Outcomes
1.00
0.75
0.50
0.25
0.00
$P(y=1|x)$
$y=1$
$y=0$
Flip #12
Observed Proportion
0.75
$\hat{p} = P(y=1|x^*)$
0.75
x^*
x


Logistic Regression
1.00
0.75
0.50
0.25
0.00
$P(y = 1|x)$
x
$y = 1$
$y = 0$
Pick another x


Logistic Regression
1.00
0.75
0.50
0.25
0.00
$P(y=1|x)$
$y=1$
$y=0$
Pick another x
x^{**}
x^{*}
x

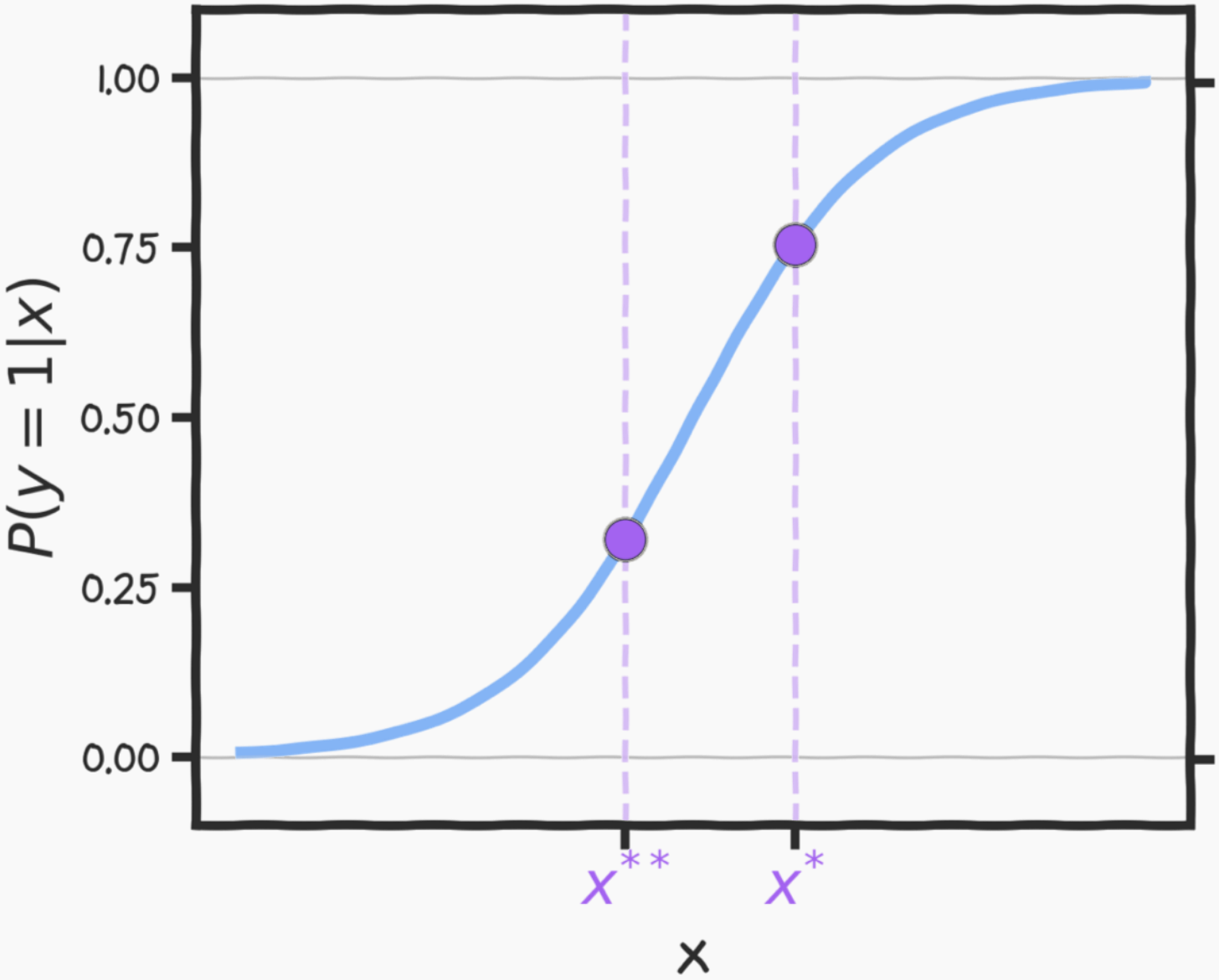
Logistic Regression



$\hat{p} = P(y=1|x^{**})$

0.32

Logistic Regression

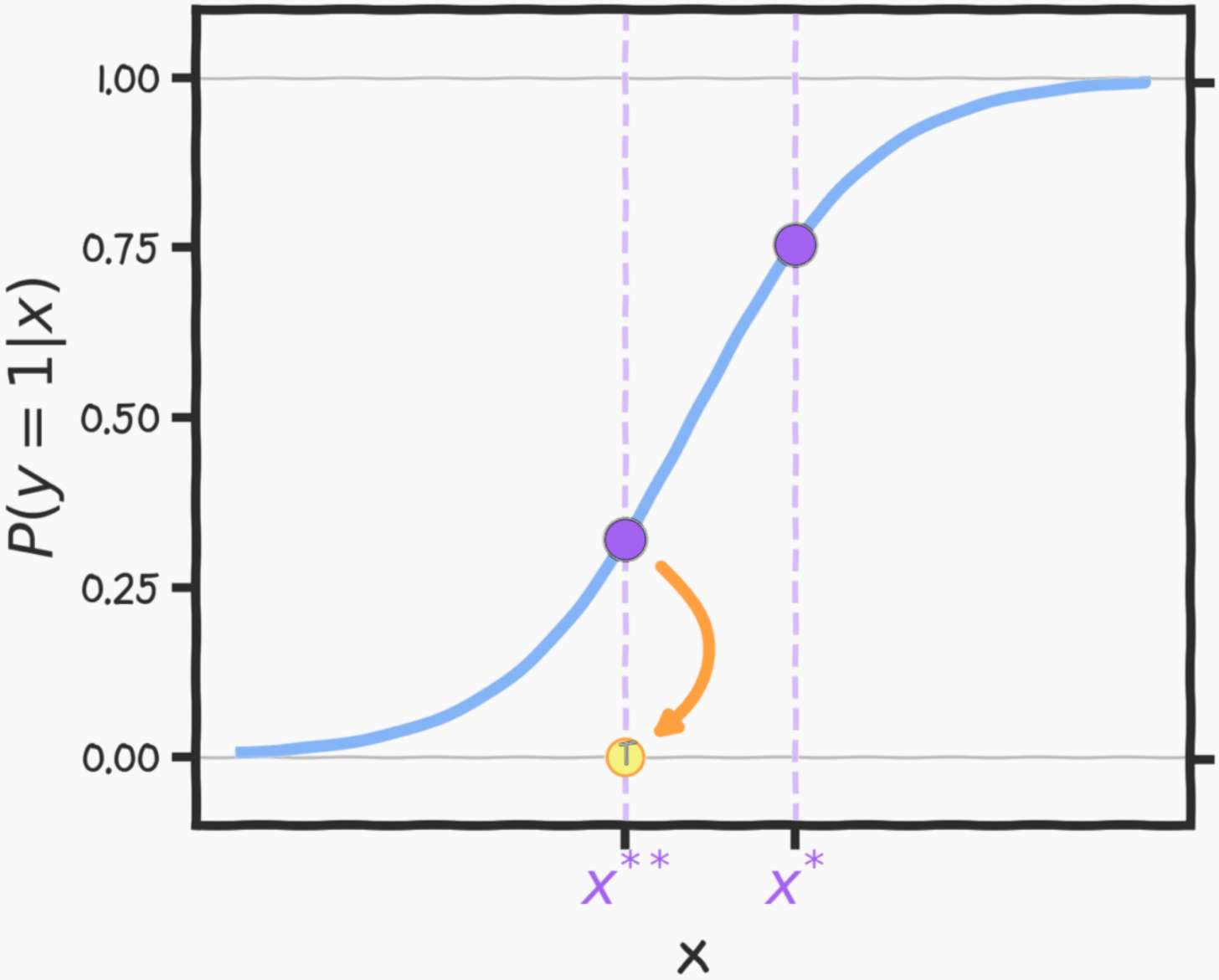


Flip a Coin with a
$\hat{P}$ Probability of Heads

$\hat{p} = P(y=1|x^{**})$

0.32

Logistic Regression

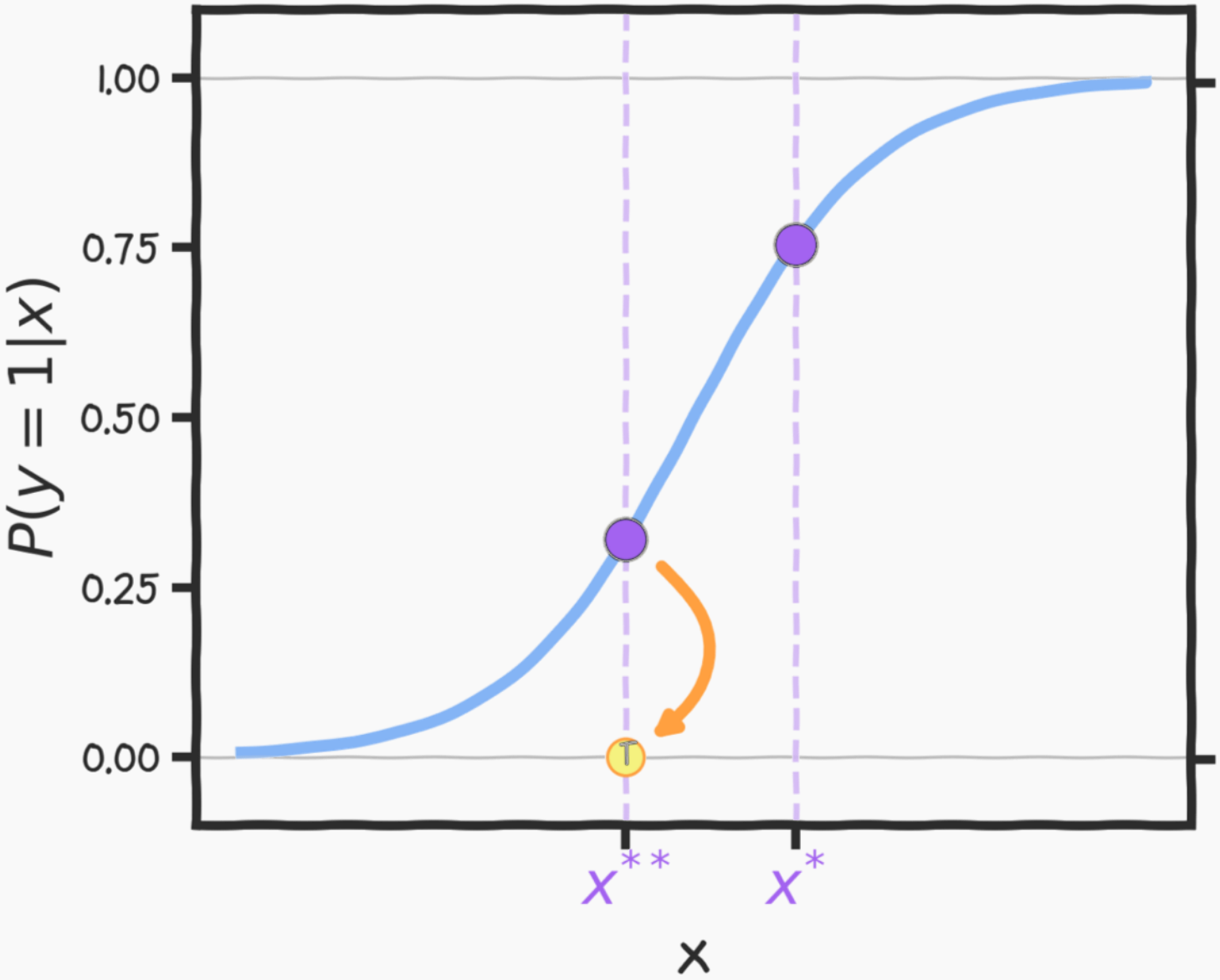


Flip a Coin with a $\hat{P}$ Probability of Heads

$\hat{p} = P(y=1|x^{**})$

0.32

Logistic Regression

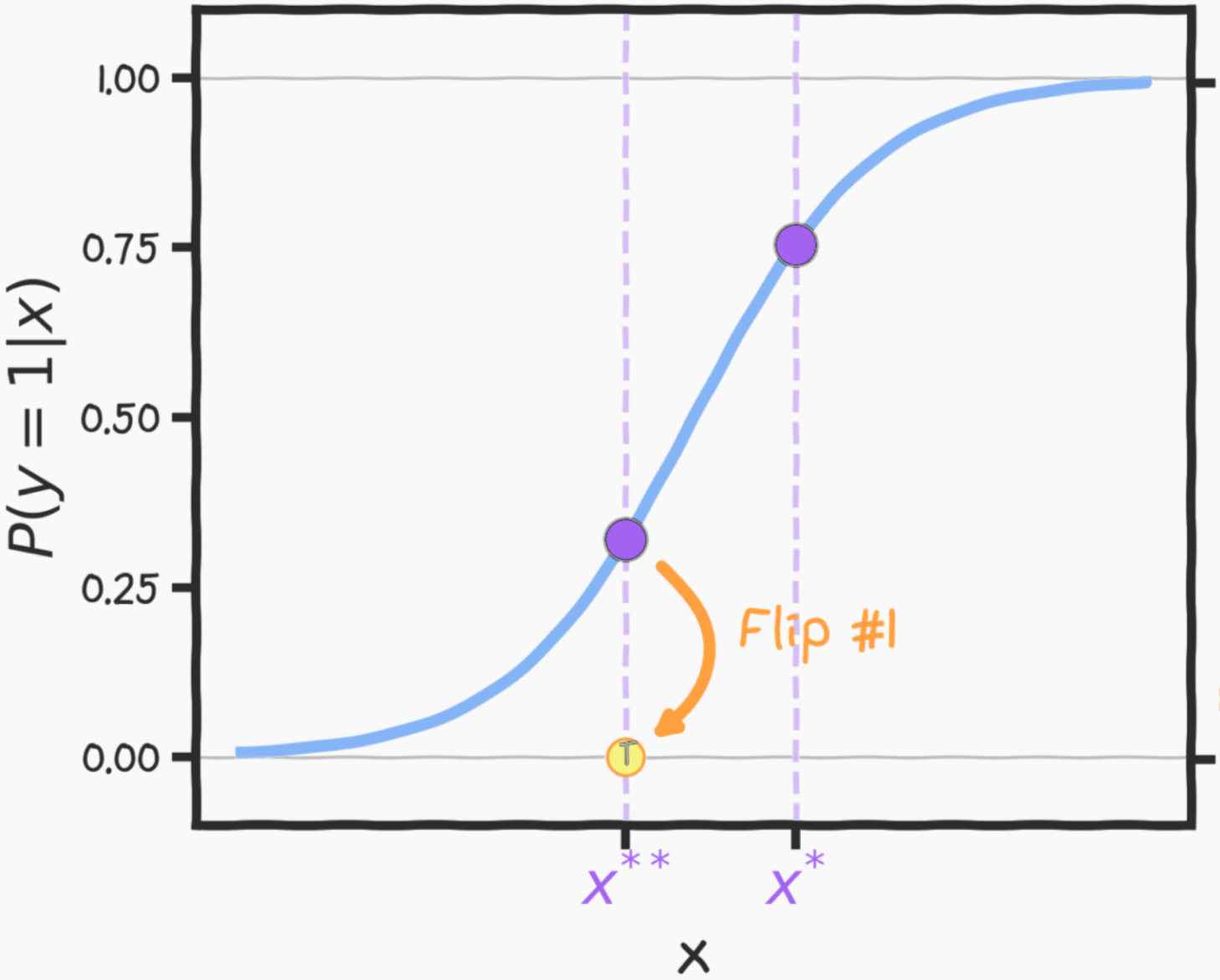


Record Outcome

$\hat{p} = P(y=1|x^{**})$

0.32

Logistic Regression



After 1 Outcome

$y = 1$

Record Outcome

$\hat{p} = P(y = 1|x^{**})$

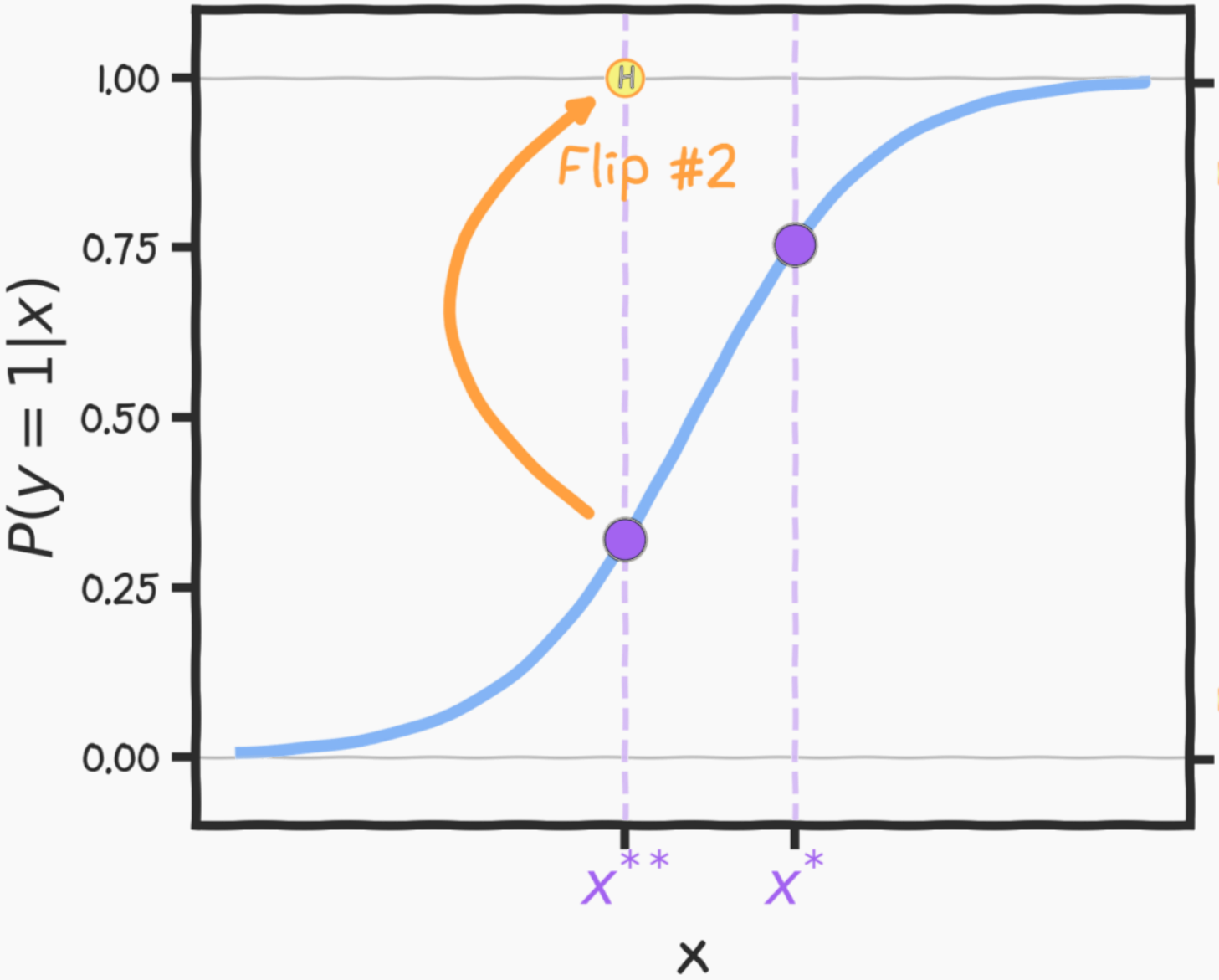
0.32

Flip #1

T

$y = 0$

Logistic Regression



After 2 Outcomes

$y = 1$

H

Observed Proportion

0.50

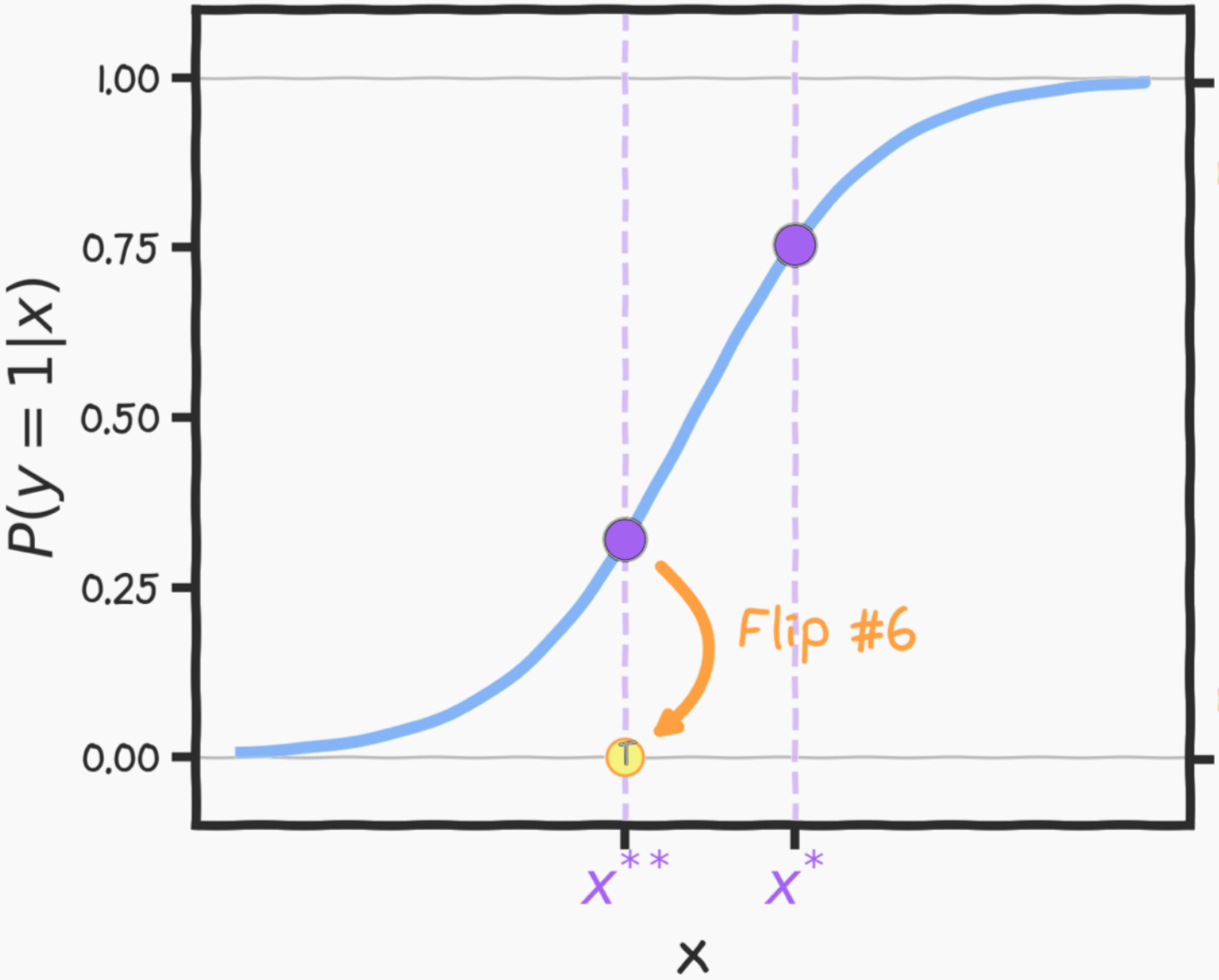
$\hat{p} = P(y = 1|x^{**})$

0.32

T

$y = 0$

Logistic Regression



After 6 Outcomes

$y = 1$

H

Observed Proportion

0.17

$\hat{p} = P(y = 1|x^{**})$

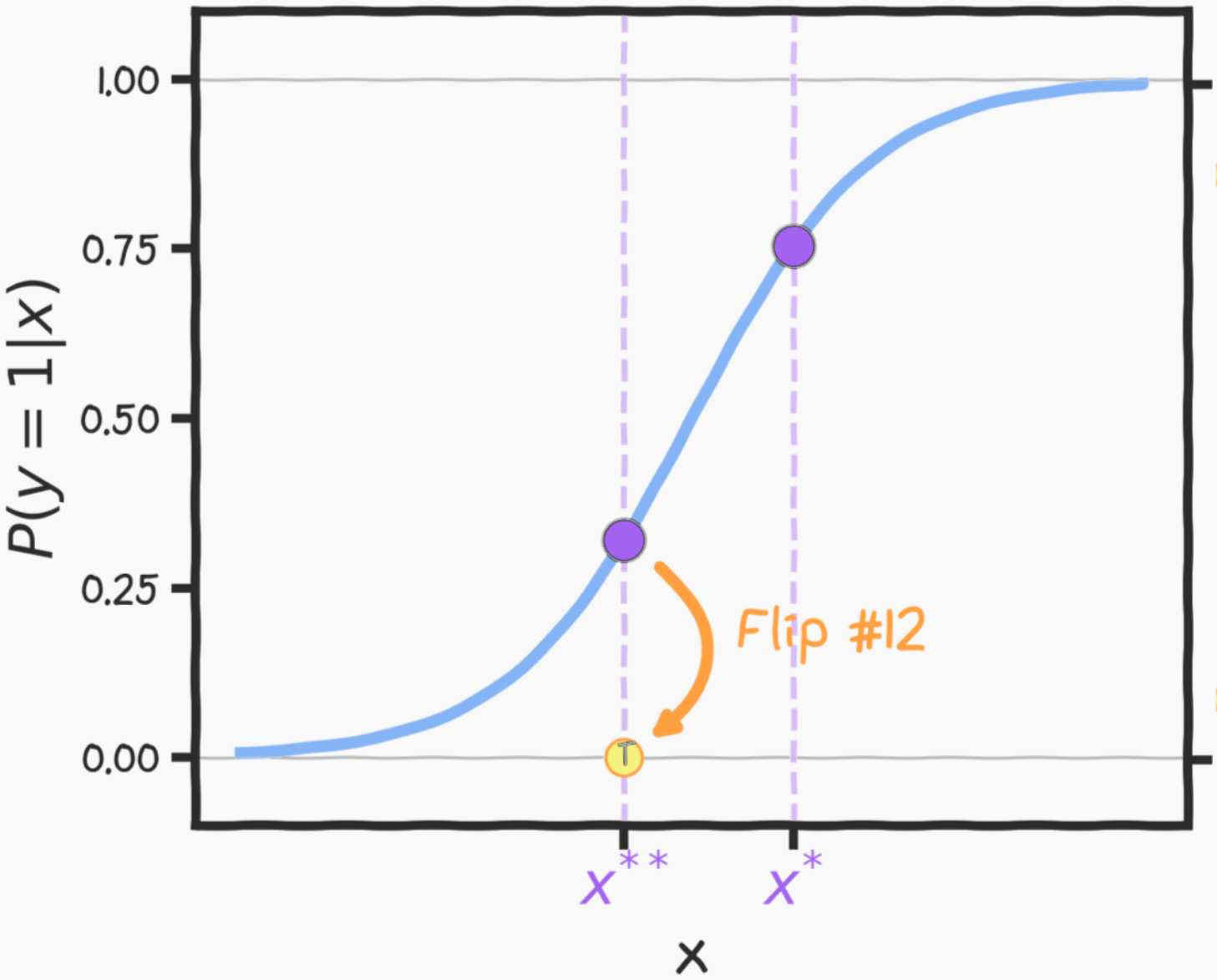
0.32

Flip #6

T T T T T

$y = 0$

Logistic Regression



After 12 Outcomes

$y = 1$

H H H H

Observed Proportion

0.33

$\hat{p} = P(y = 1|x^{**})$

0.32

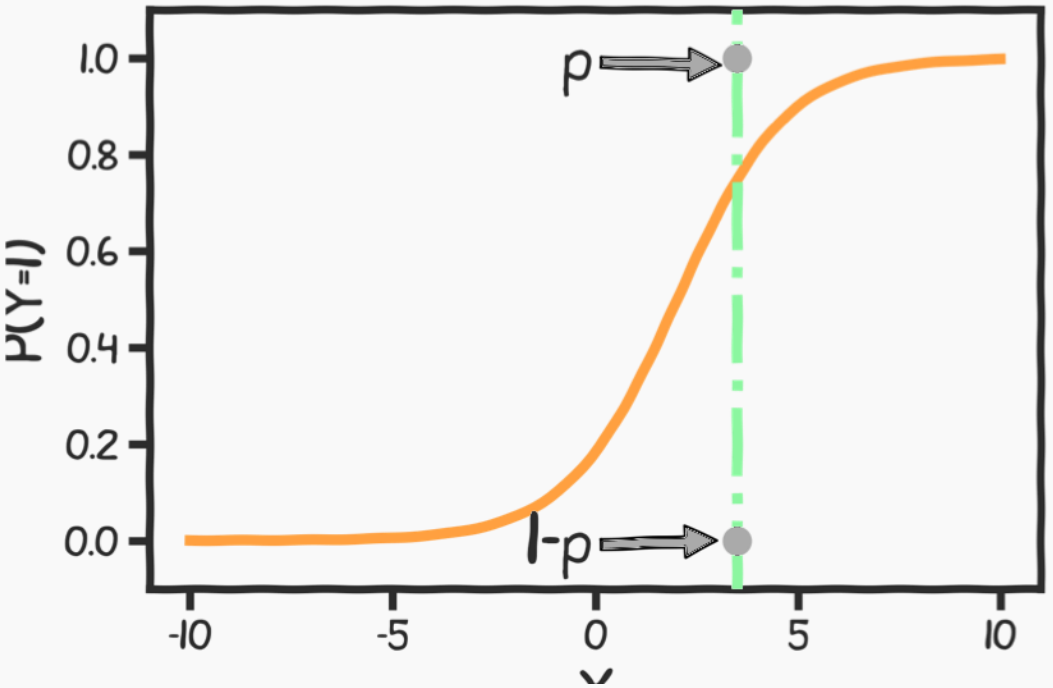
Flip #12

T T T T T T T T

$y = 0$

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.



Prob. $Y = 1$: $P(Y = 1) = p$
Prob. $Y = 0$: $P(Y = 0) = 1 - p$

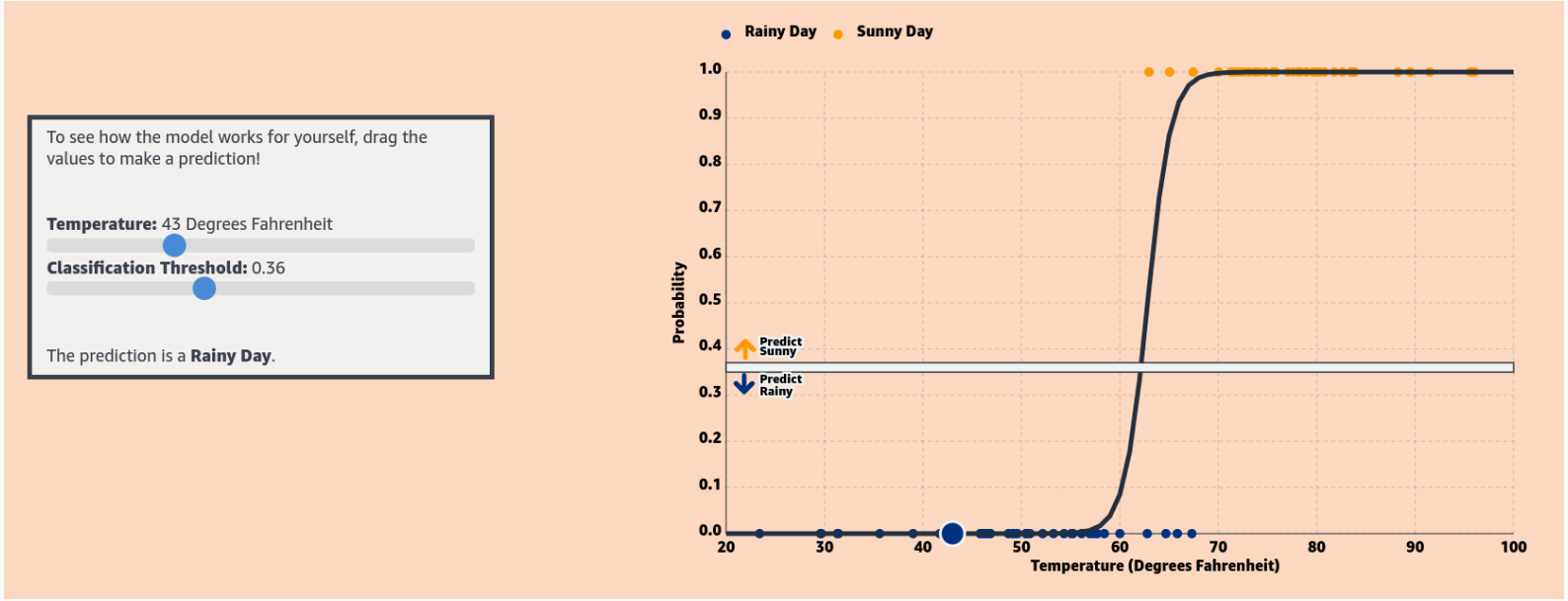
$$P(Y = y) = p^y(1 - p)^{(1-y)}$$

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 INDENTICAL 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

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

Using the formula for the log-likelihood, we get

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

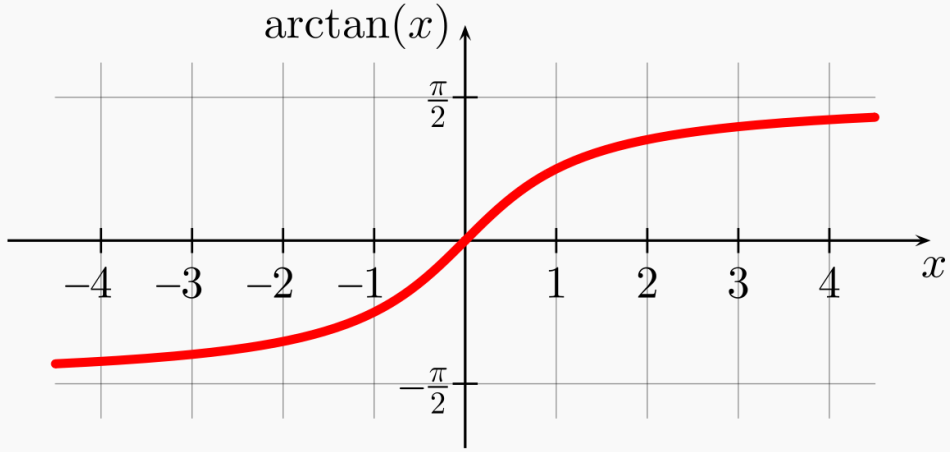
The final loss, called the **Binary Cross Entropy,** is then:

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

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 $\Phi(\beta_0 + \beta_1 X_1 + \cdots)$.

Outline

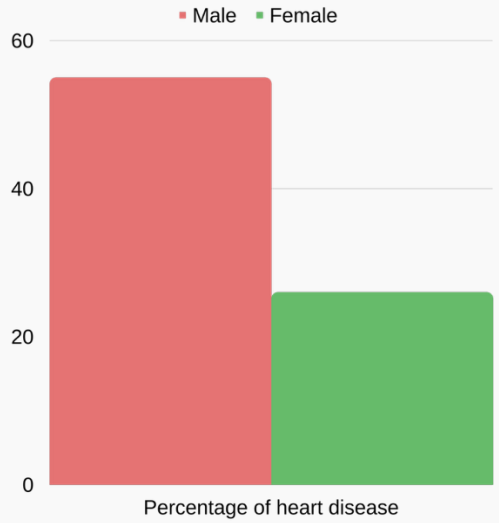
- Review
- What is Classification?
- Why not Linear Regression?
- Estimating the Simple Logistic Model
- **Inference in Logistic Regression**
- Multiple Logistic Regression
- Classification Decision Boundaries

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 = AHD from X = 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?

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

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

- Review
- 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:

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

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 $\beta_0, \beta_1, \ldots, \beta_p$ 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 $e^{\widehat{\beta}_j}$ 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

- Review
- 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

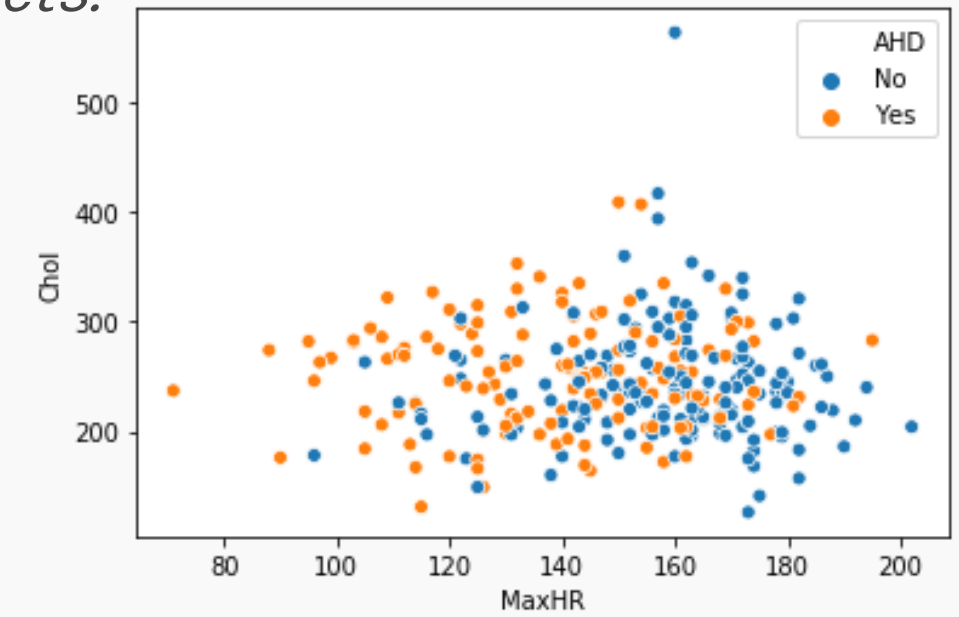
Recall that we could attempt to purely classify each observation based on whether the estimated $P(Y = 1)$ from the model is at least 0.5:

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

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

```
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]
```

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

What is the estimated model?

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

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?

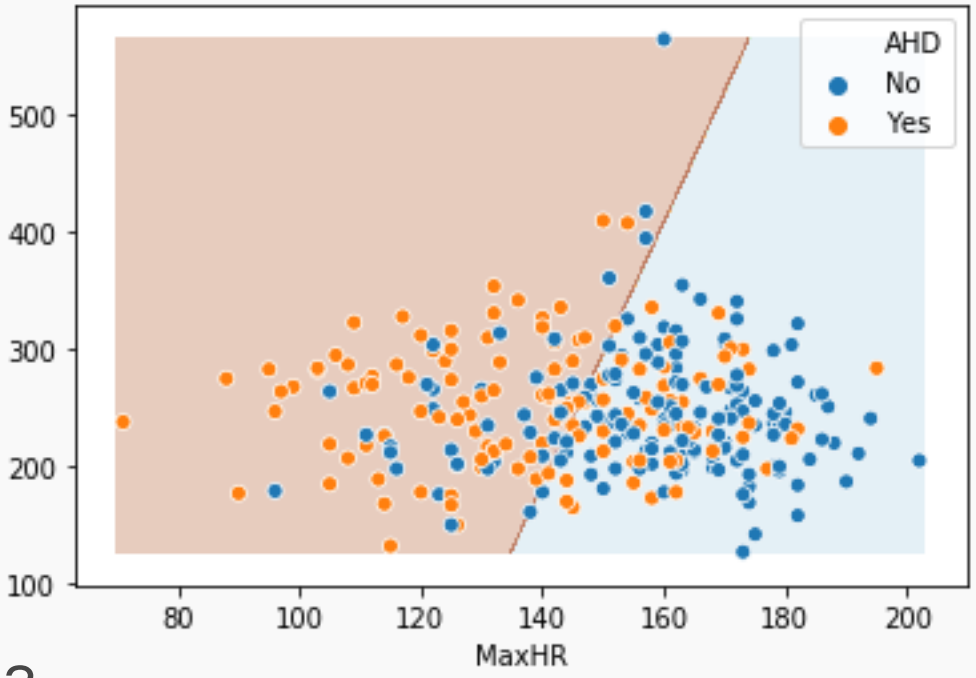
In logistic regression, the decision boundary is defined when $X\beta = 0$.

2D Classification in Logistic Regression: Example #1

A logistic regression model was fit to predict Y = AHD from X_1 = MaxHR and X_2 = 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?

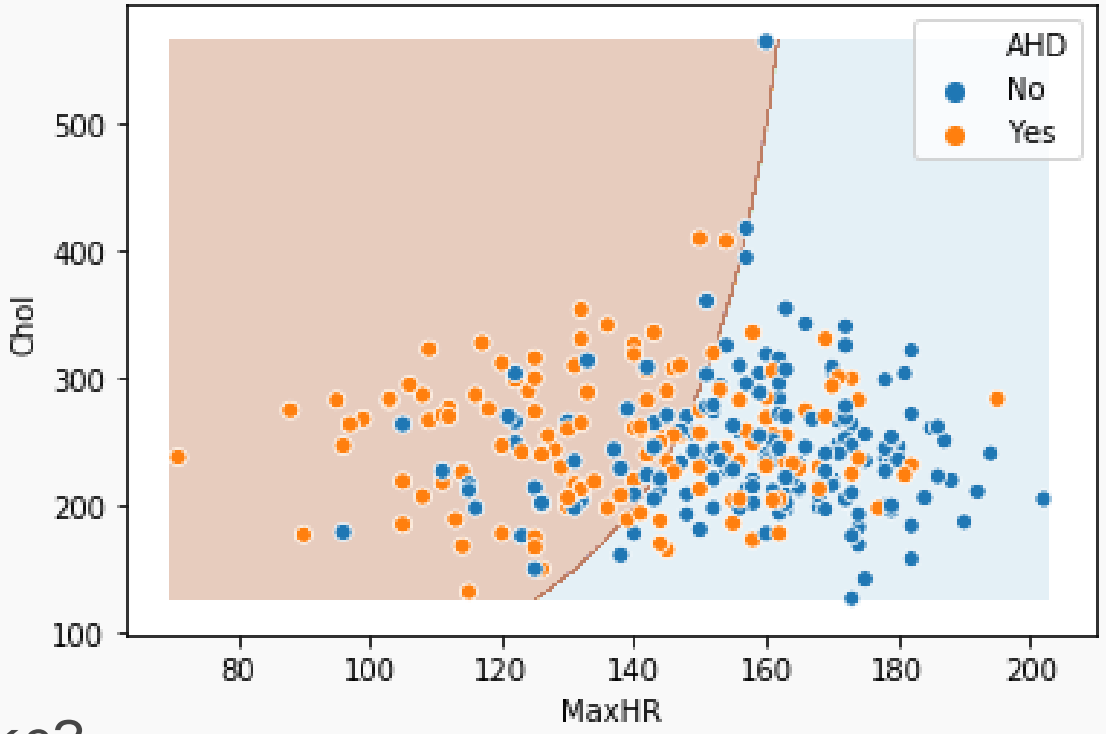
In logistic regression, decision boundaries are structured to be linear!

2D Classification in Logistic Regression: Example #2

A logistic regression model was fit to predict Y = AHD from X_1 = MaxHR and X_2 = Chol, and X_3 = 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?

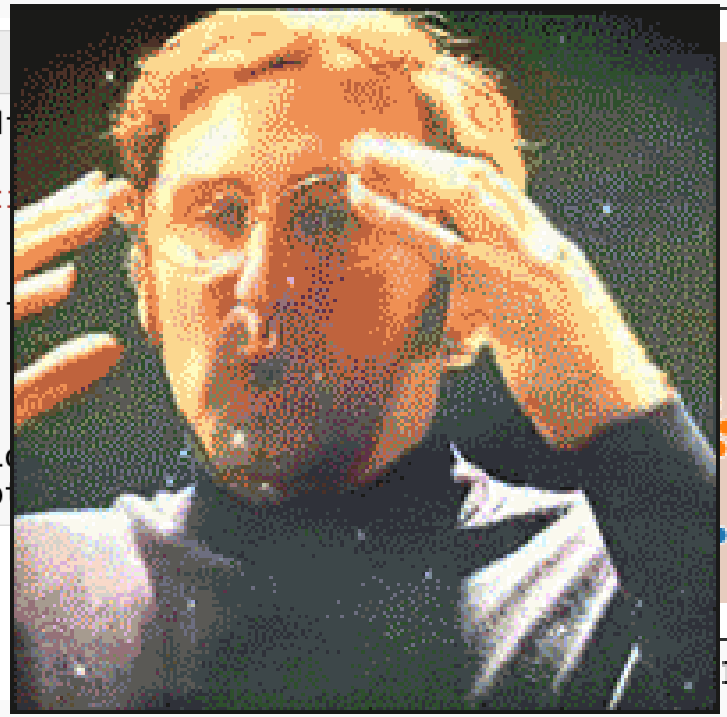
In logistic regression, decision boundaries are not always structured to be linear!

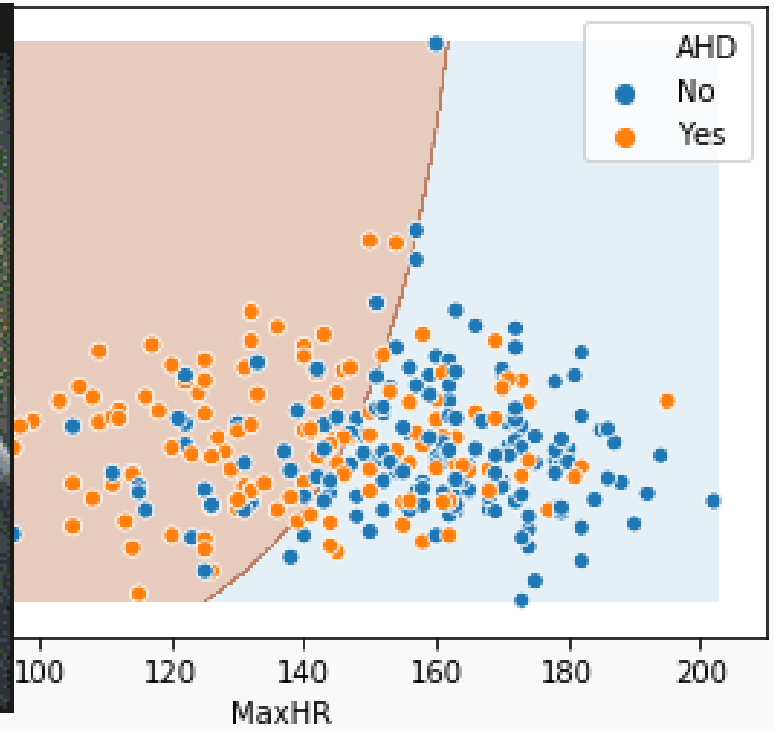
2D Classification in Logistic Regression: Example #2

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

```
df_heart['Interaction'] = df_heart.MaxHR * df

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

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


print('Estimated beta1, beta2, beta3: \n', lo
print('Estimated beta0: \n', logreg.intercep
```

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



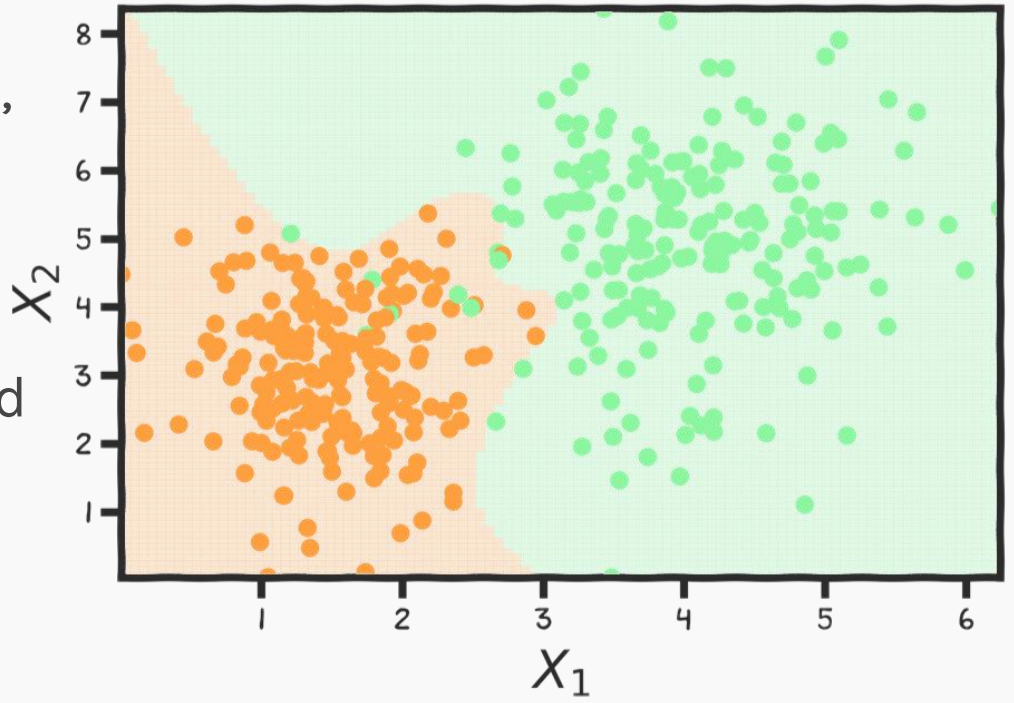


What will the decision boundary look like?

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?

$$X = \{X_1, X_1^2, X_2, X_2^2, X_1X_2\}$$

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