Machine Learning

POLI SCI 210

Introduction to Empirical Methods in Political Science

AI Prompts

- Supervised vs. unsupervised learning
- Overfitting problem [in machine learning]
- Explain how your (AI chatbot) algorithm works
- Explain how [fancy algorithm] works
- How can a [political scientist] use [machine learning/AI] for [application of interest]

Roadmap

- **Tuesday:** Big picture, simple models
- **Thursday:** Fancier models, generative AI

Summary of the course

- Focus on **inference** since it is how political scientists test theories
- **Statistical inference:** summarize data, quantify uncertainty
- **Univariate:** Mean, confidence intervals, standard errors
- **Bivariate:** Difference in means (experiments, potential outcomes)
- **Multivariate:** OLS regression

Summary of the course

- **Subplot:** *Bivariate* and *multivariate* only make sense if we want to make **causal statements**
- **Causal inference:** Impose some structure to justify assumptions
- **Small N:** Necessary and sufficient as logic of inference

But *inference* is not the only thing we do with *data*

These are not statistical inference

But they all mean (kinda) the same

Data science

Machine learning

Statistical learning

Artificial intelligence

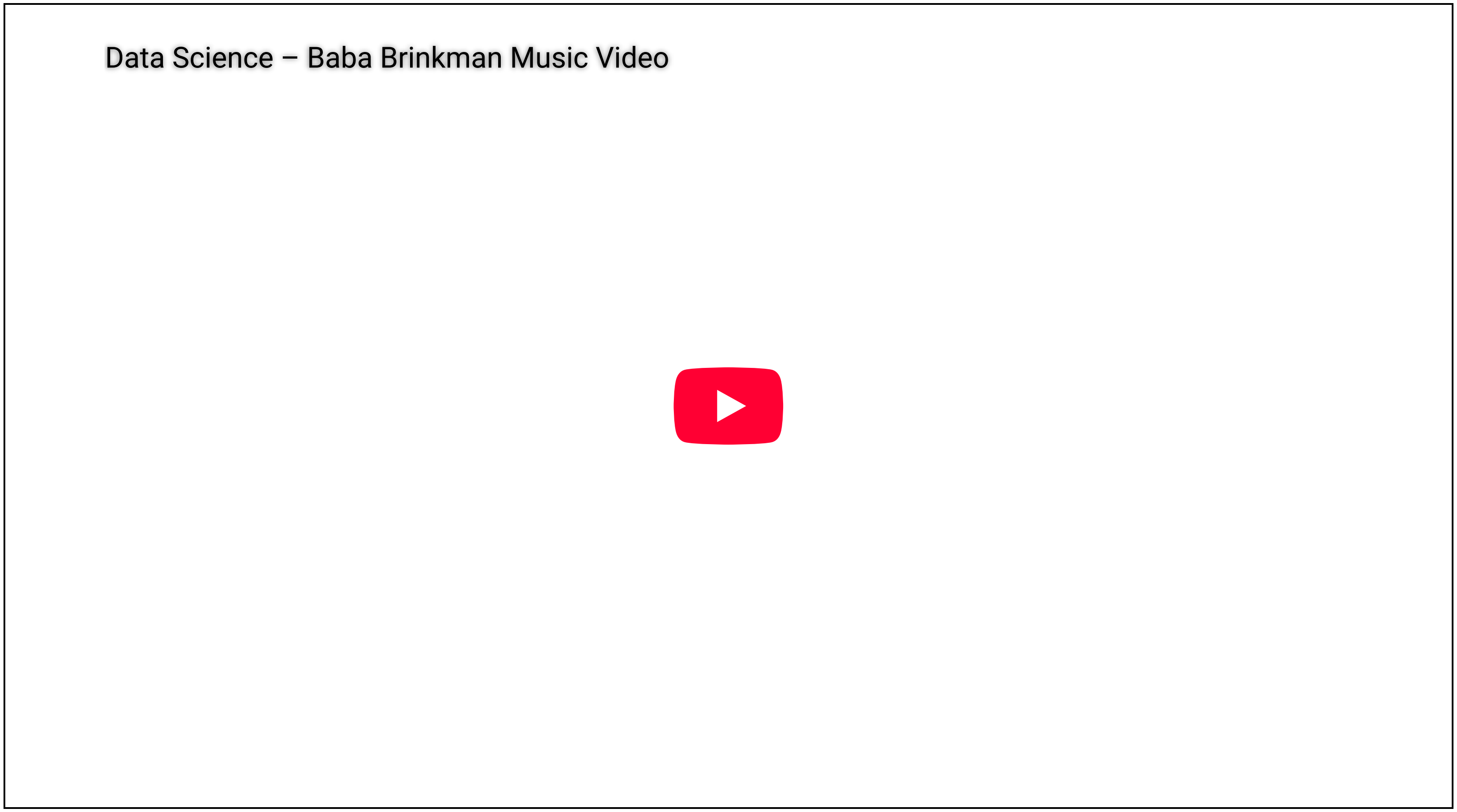
Predictive modeling

Deep learning

Big data (?)

Different flavors depending on the field, but methods are the same

How are they different?



https://youtu.be/uHGlCi9jOWY?si=wgfvS9IiV5_FQ2aT

Ok, but how are they different?

Statistical inference

- Use data we have to learn about a target population
- Or measure a quantify of interest
- *Main product:* Estimates, uncertainty

Statistical learning

- Use data we have to *predict* how new data will look like
- Minimize *prediction error*
- *Main product:* Rules, error metrics

Same stuff, different language

Statistical inference	Statistical learning

Same stuff, different language

Statistical inference	Statistical learning
Outcome variable	Response, output

Same stuff, different language

Statistical inference	Statistical learning
Outcome variable	Response, output
Explanatory variable	Predictor, input, feature

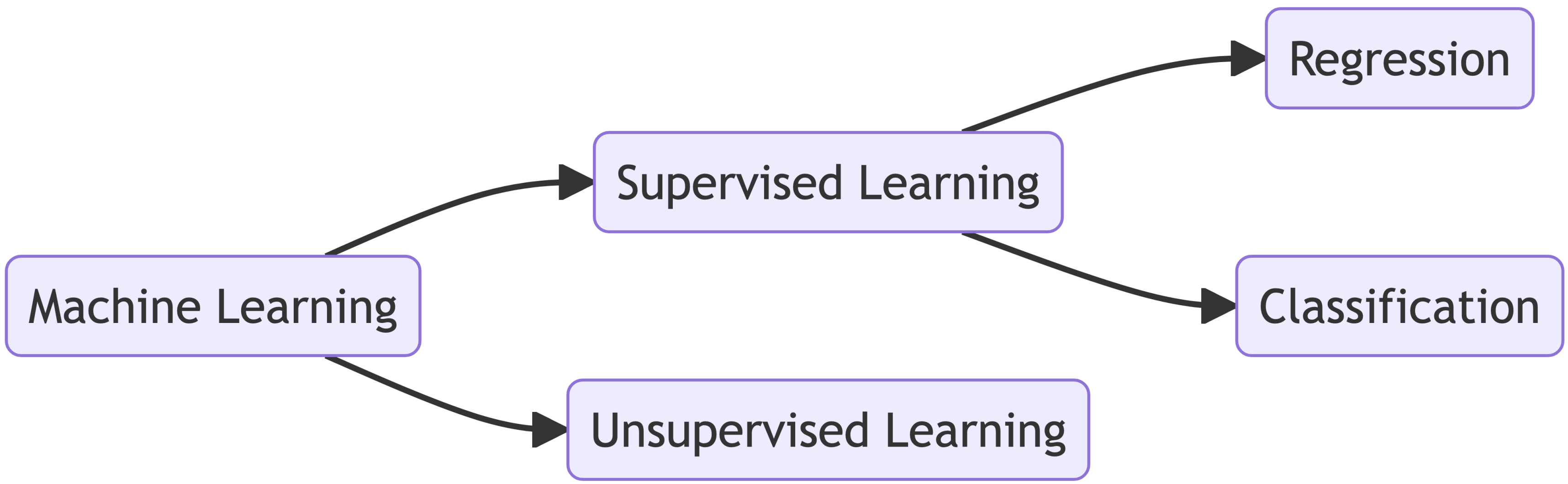
Same stuff, different language

Statistical inference	Statistical learning
Outcome variable	Response, output
Explanatory variable	Predictor, input, feature
Model	Algorithm

Same stuff, different language

Statistical inference	Statistical learning
Outcome variable	Response, output
Explanatory variable	Predictor, input, feature
Model	Algorithm
Uncertainty	Error

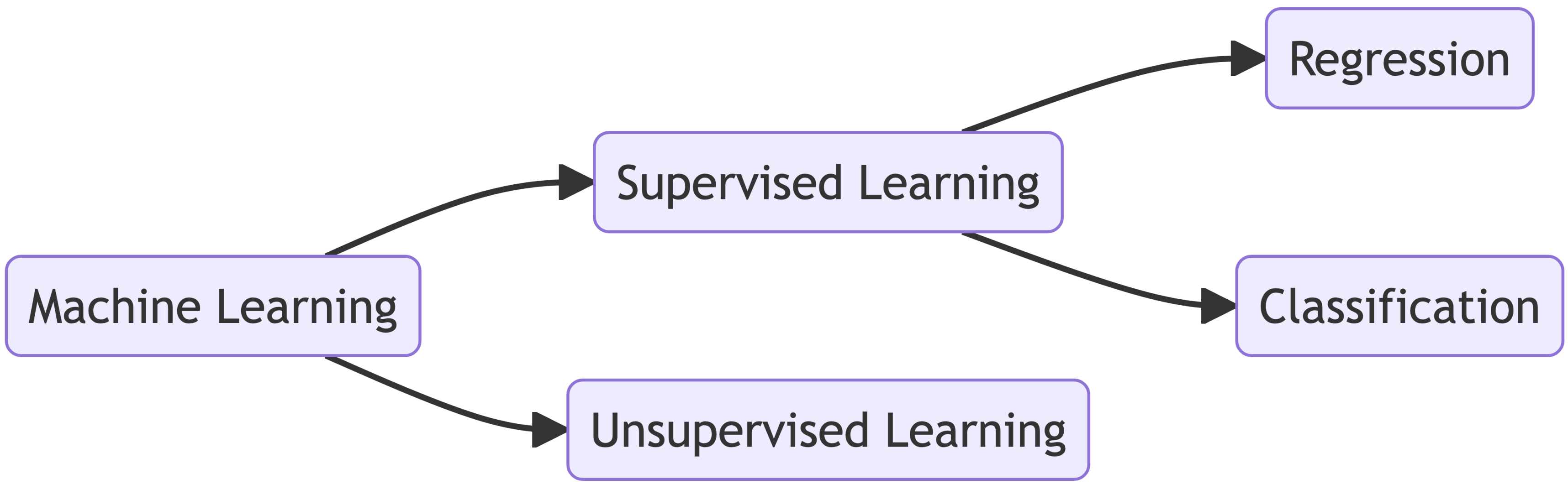
Flavors of machine learning



Supervised: Predict “correct” answer

Example: Was this text written by AI? (yes/no)

Flavors of machine learning

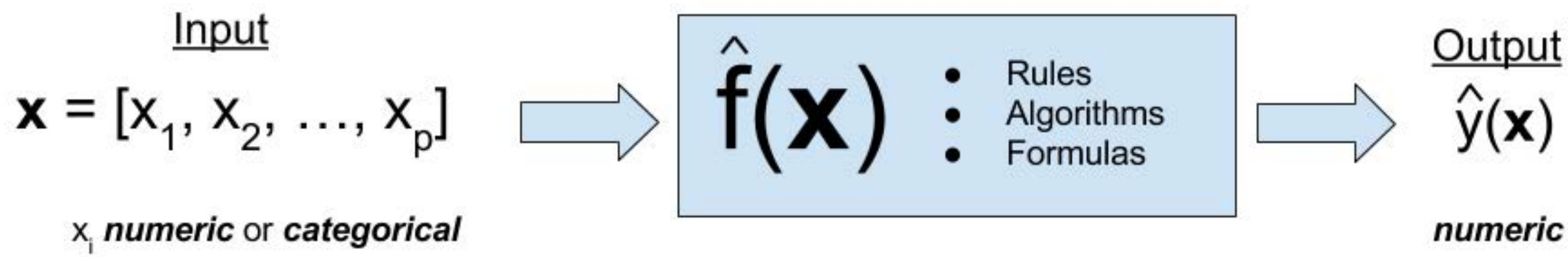


Unsupervised: No “correct” answer

Learn underlying structure of data (dimensions, clusters)

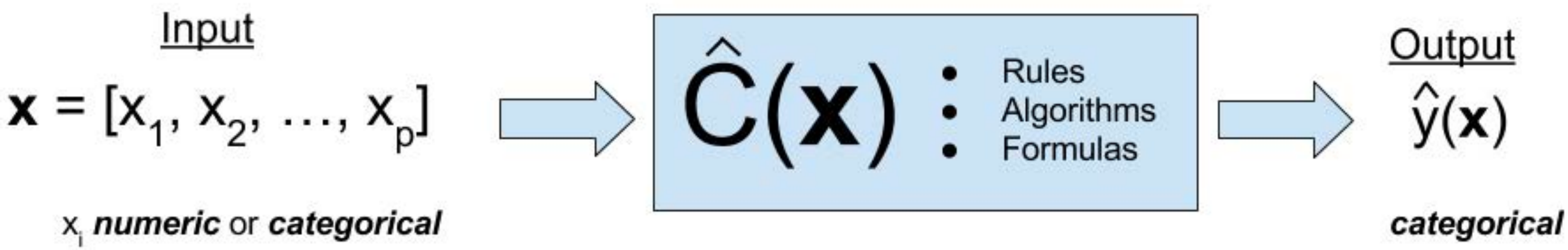
Supervised learning

Regression

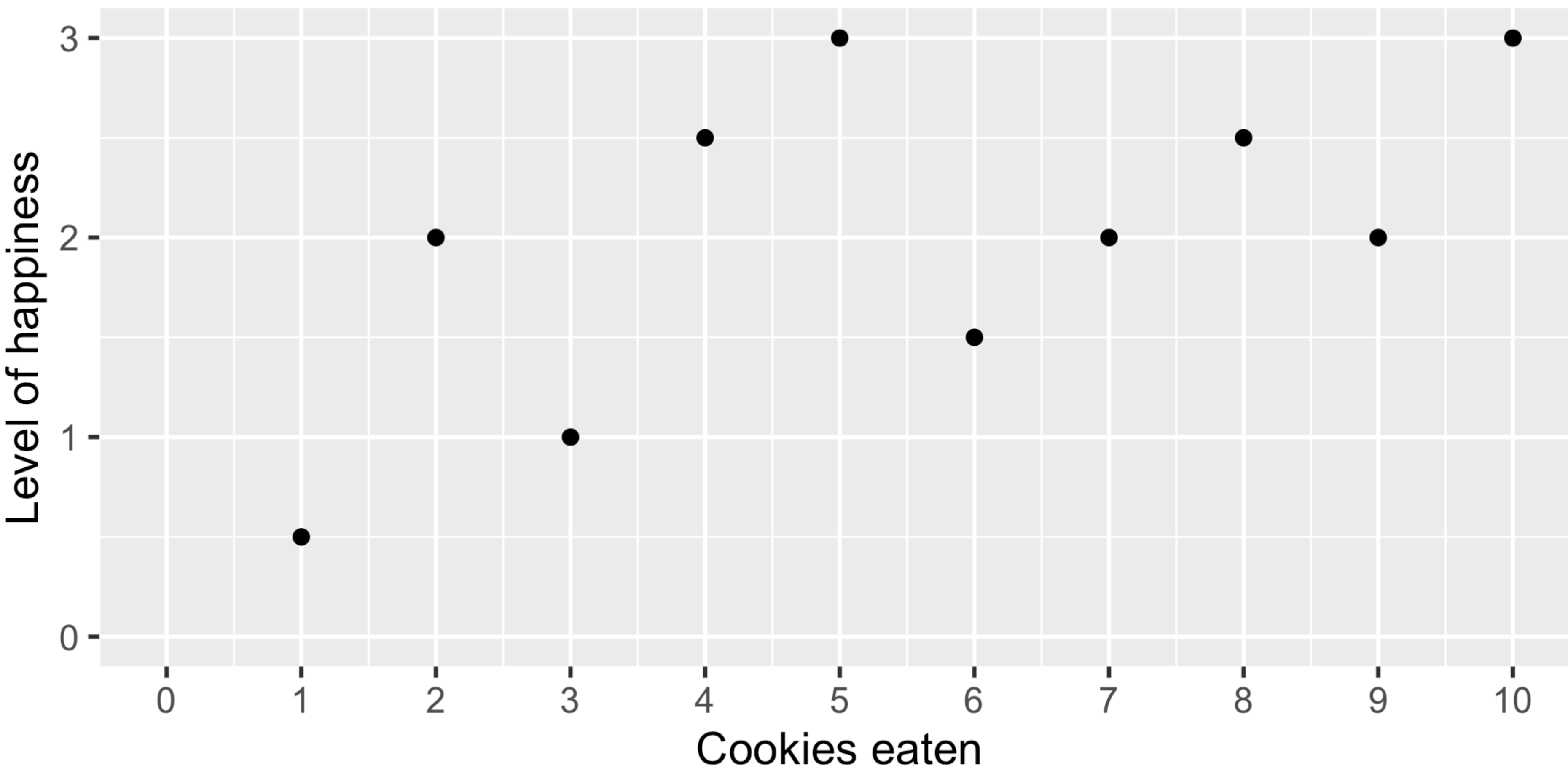


Supervised learning

Classification

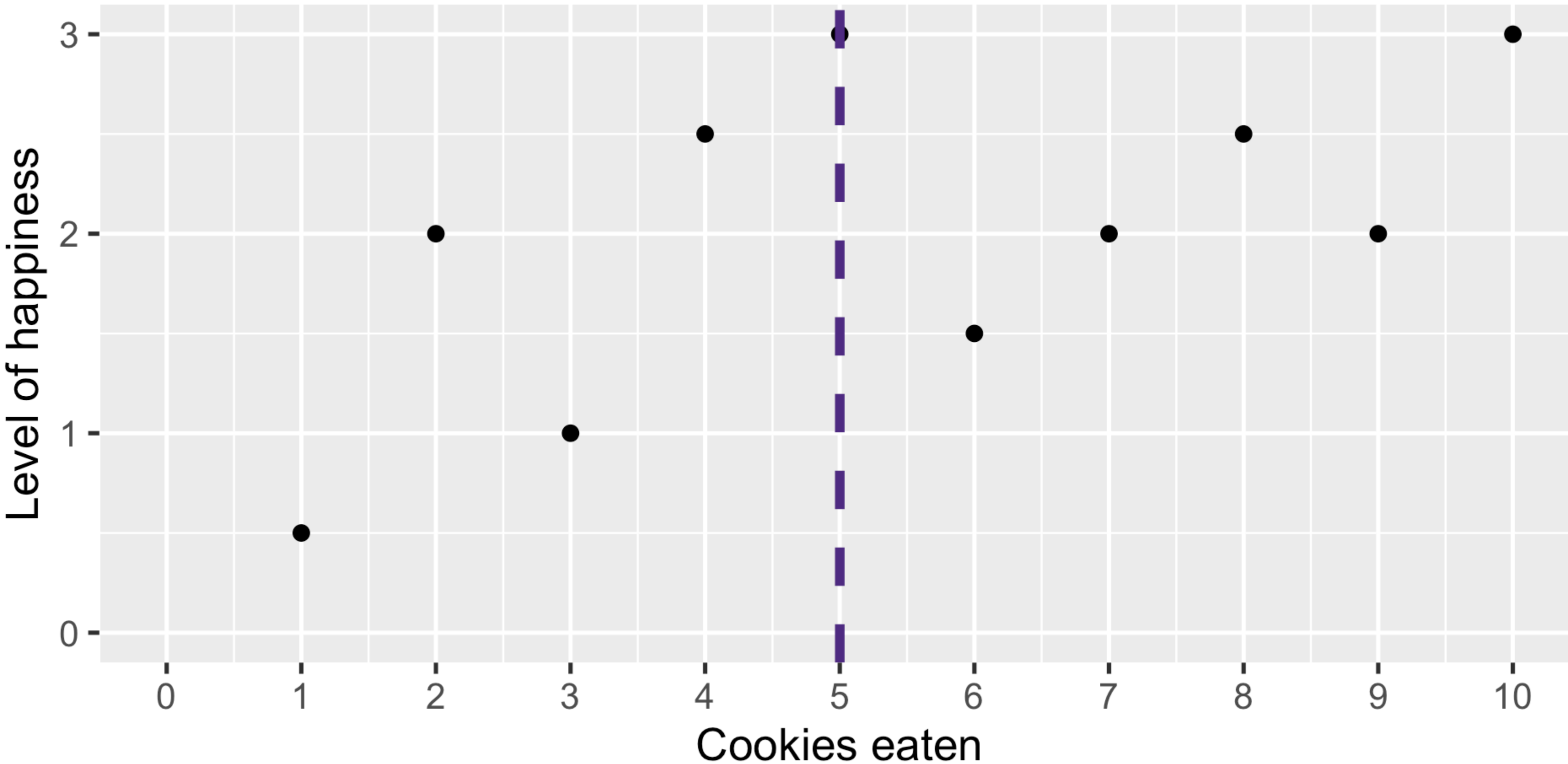


Toy example



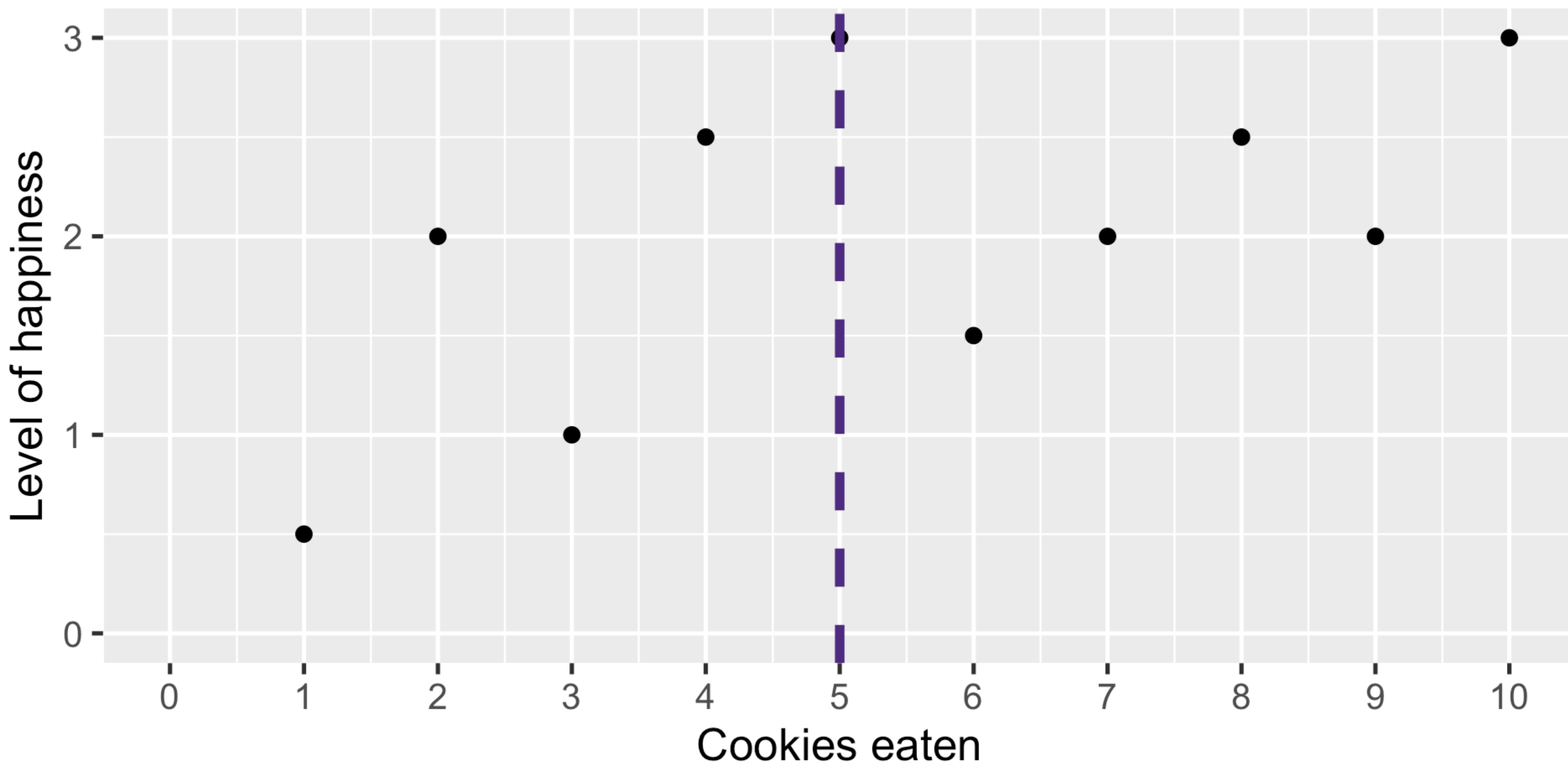
How happy will the next person be?

They eat 5 cookies



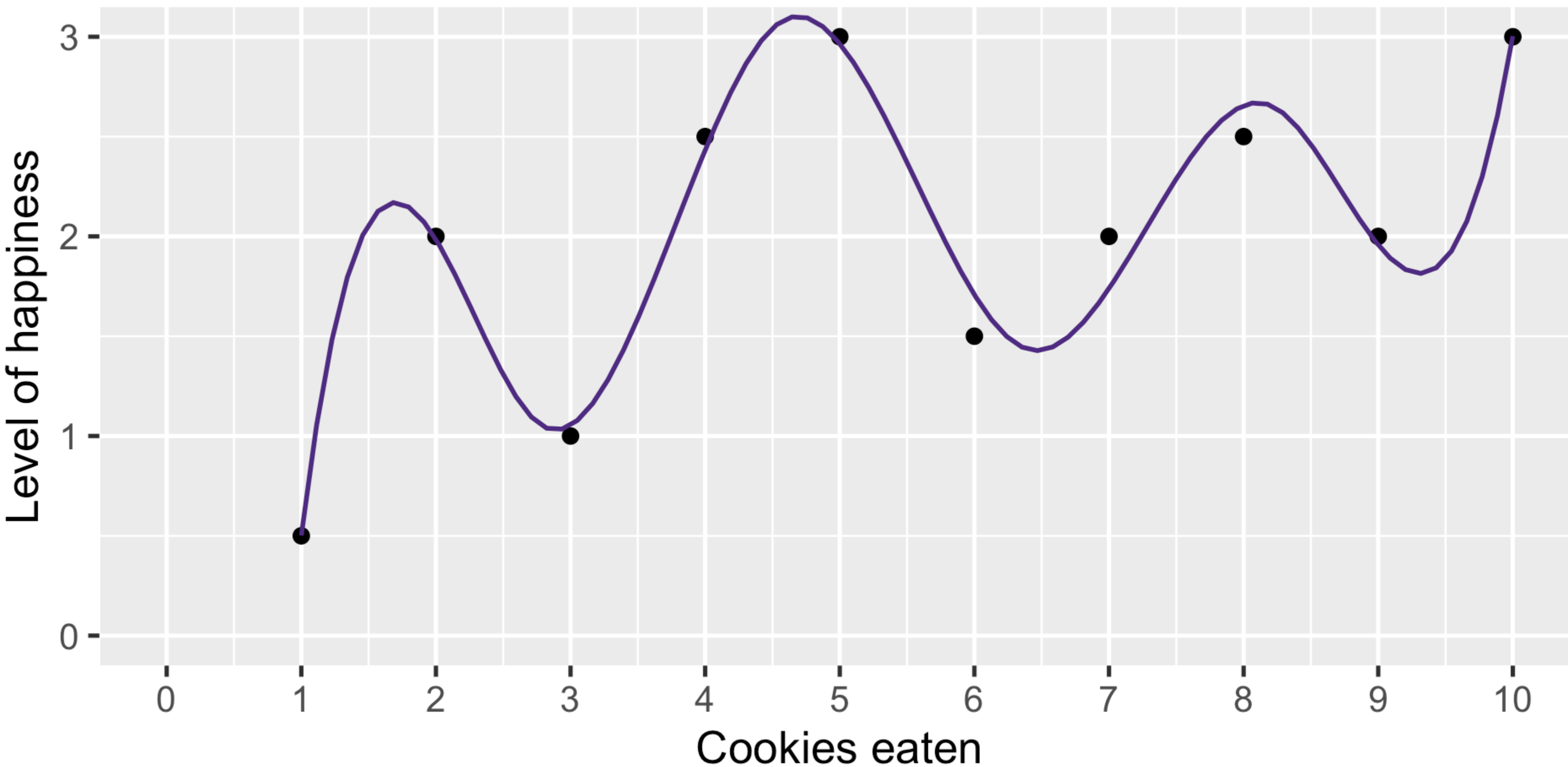
How happy will the next person be?

They eat 5 cookies

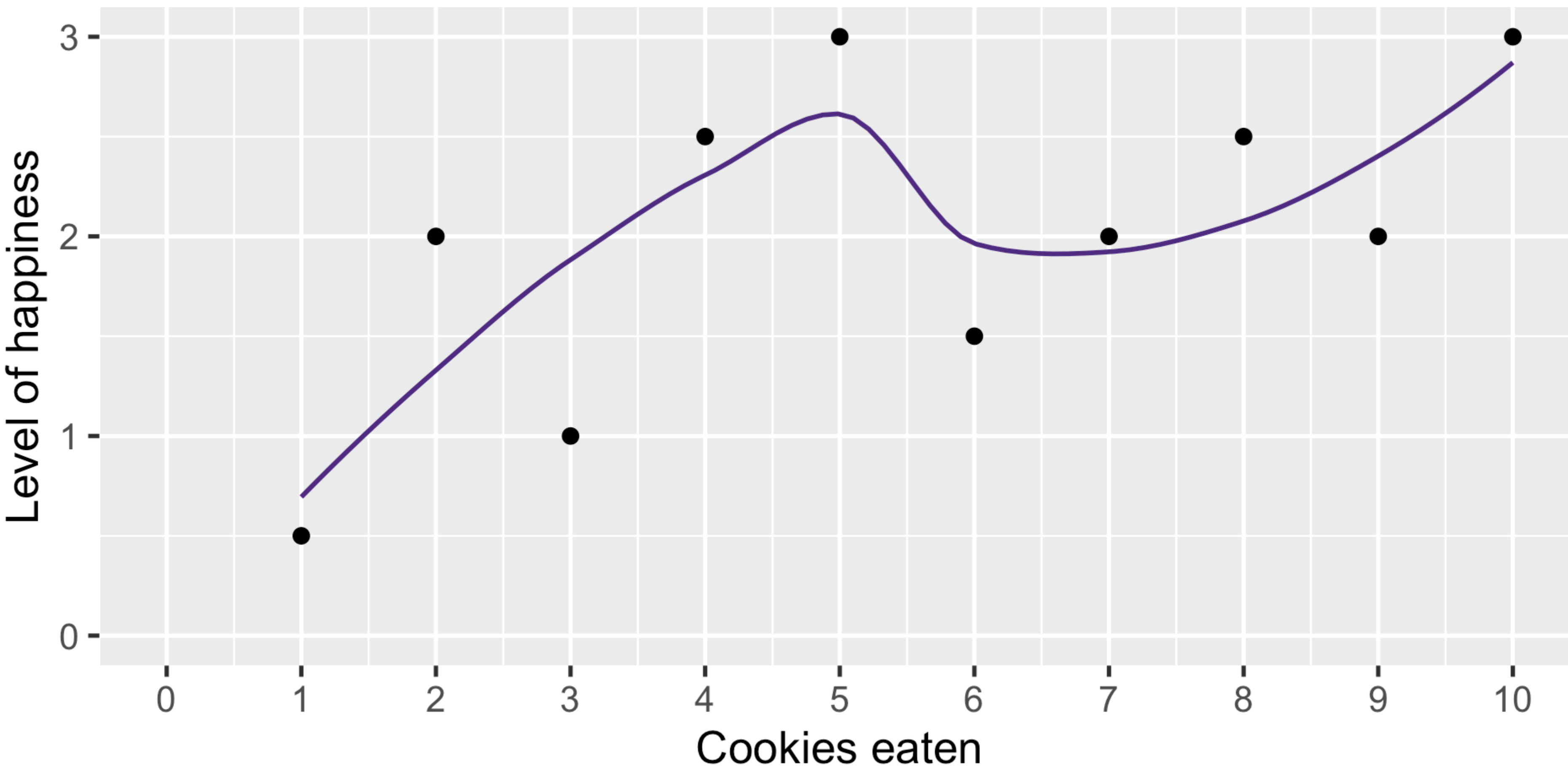


We already know one way

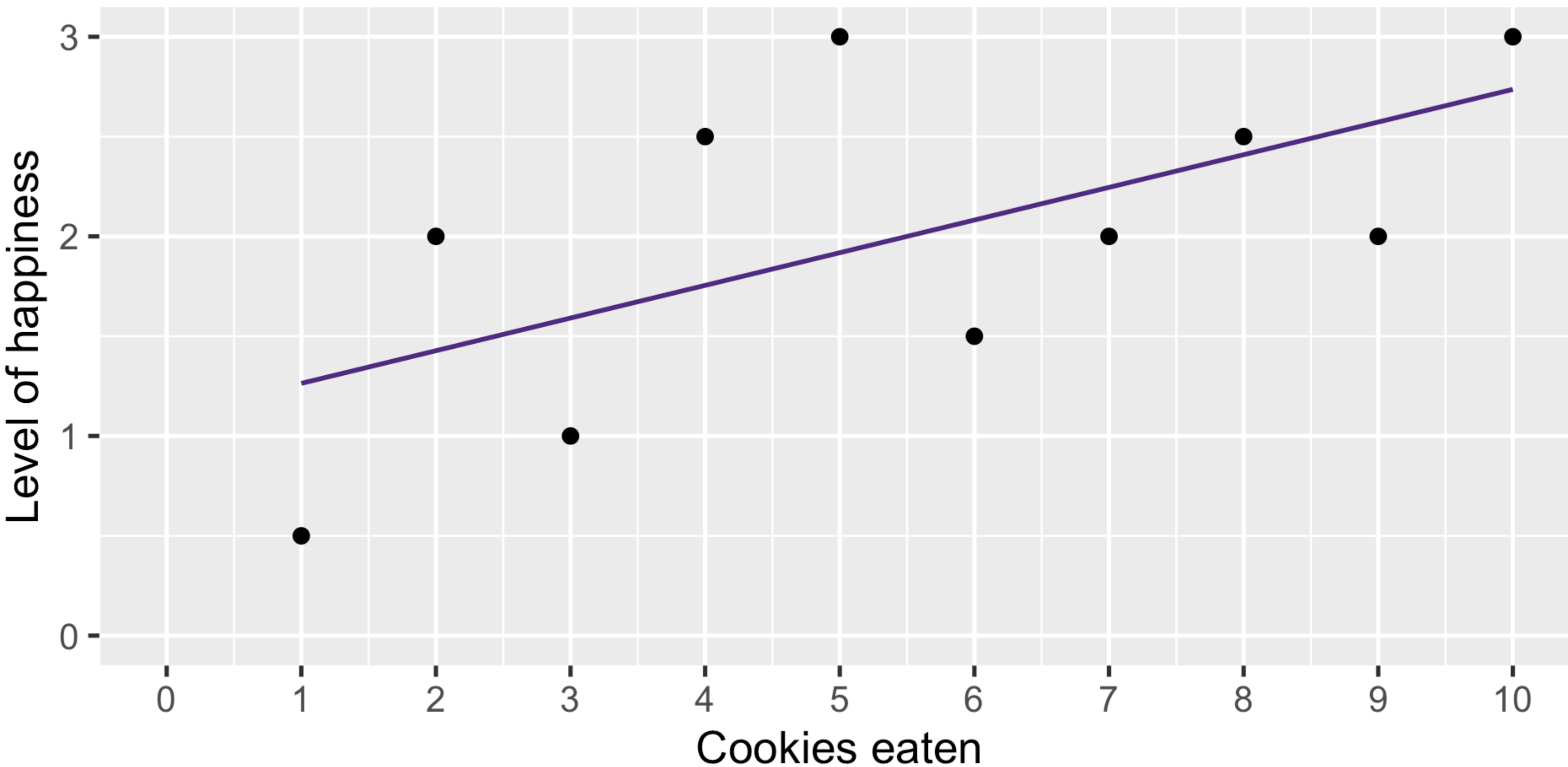
Drawing lines!



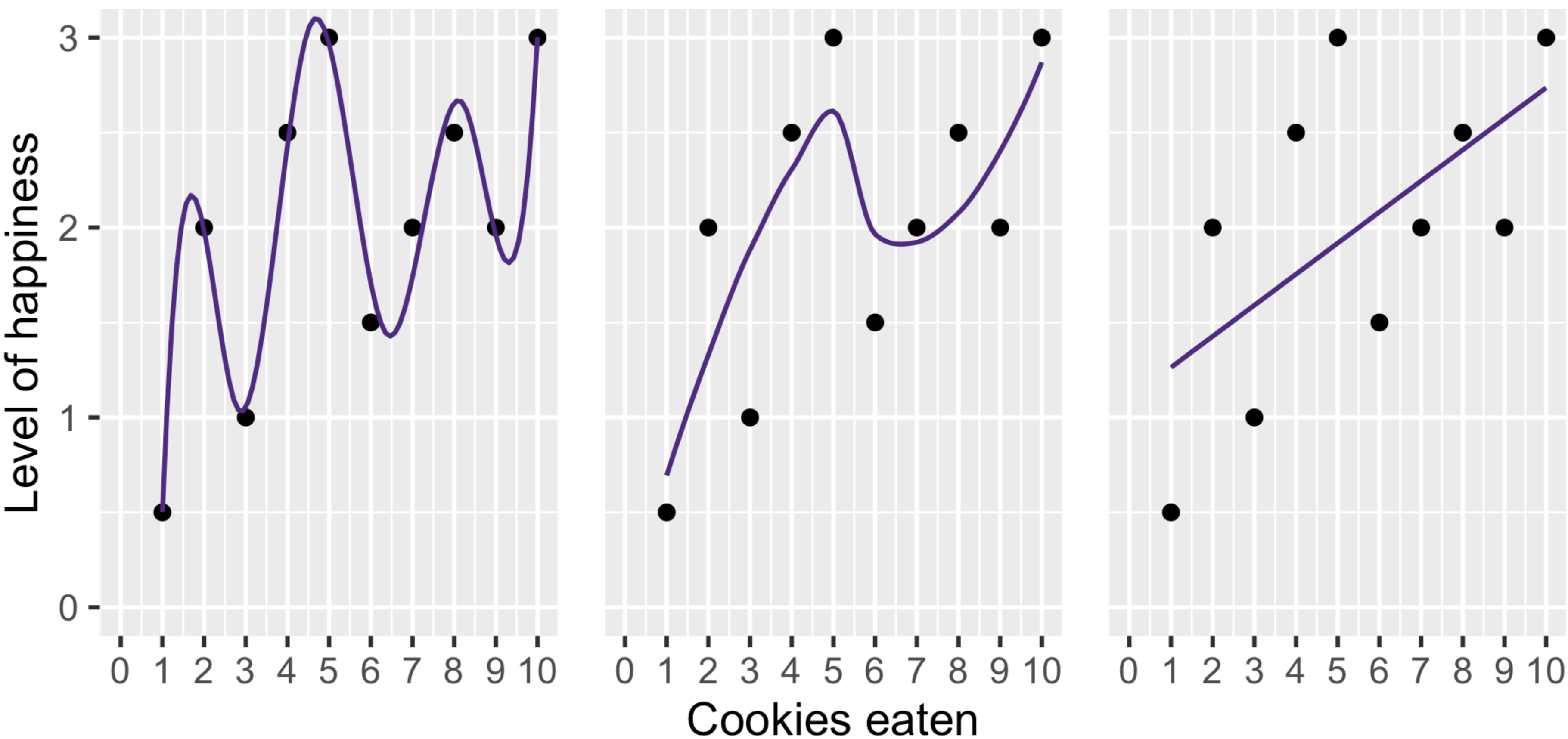
Drawing lines!



Drawing lines!



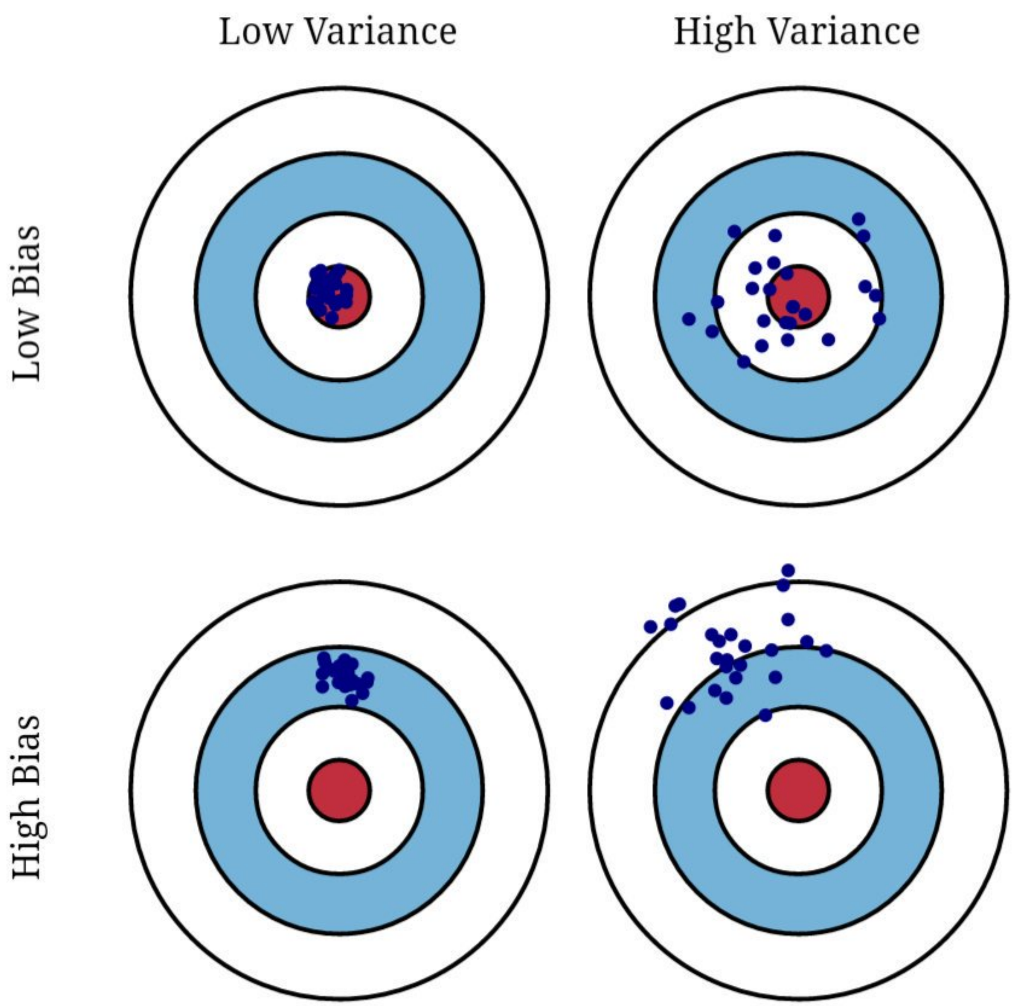
Which one seems better?

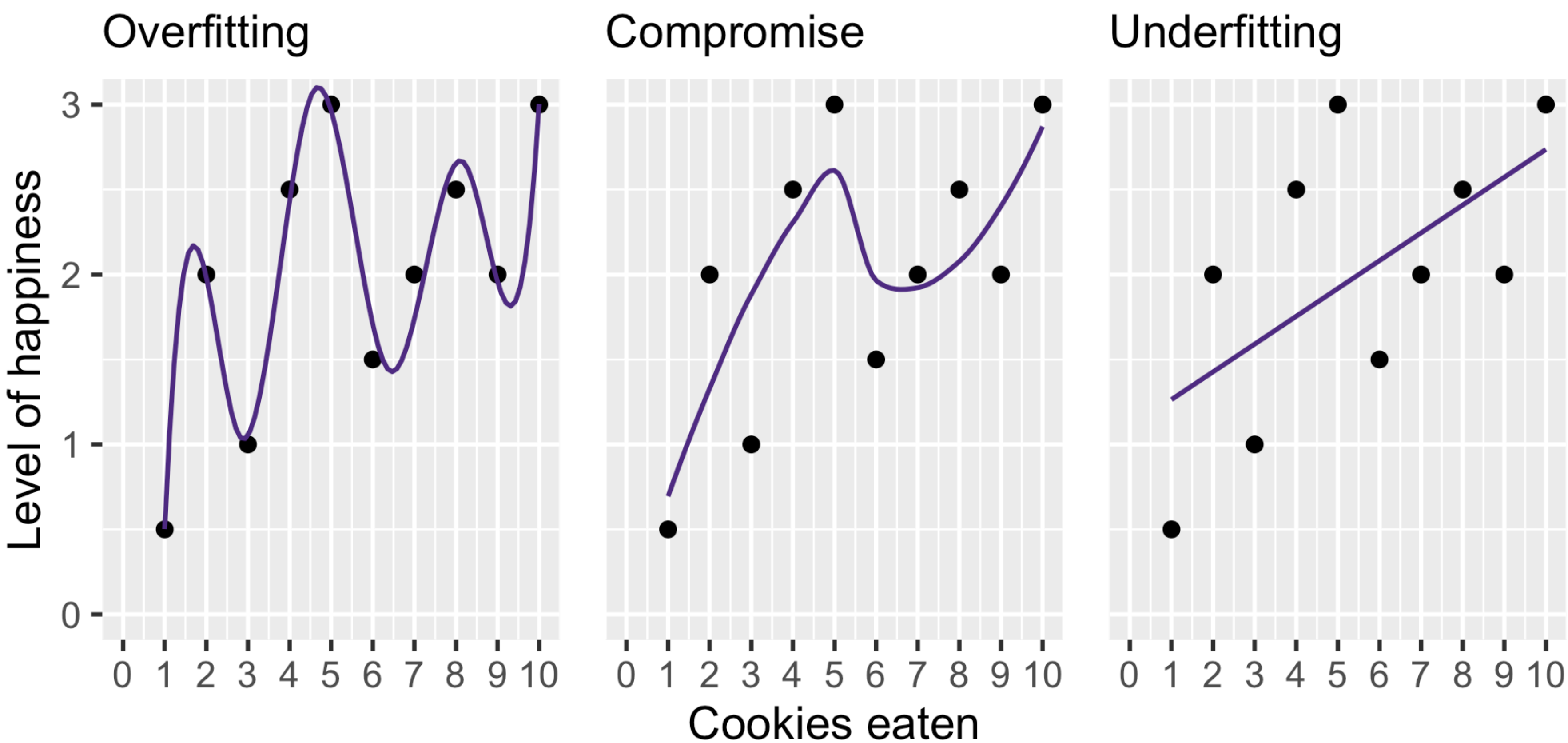


Need to balance

1. Being as close as possible
2. Avoid relying on specific observations

We already have language for this





Overfitting: Low bias, high variance

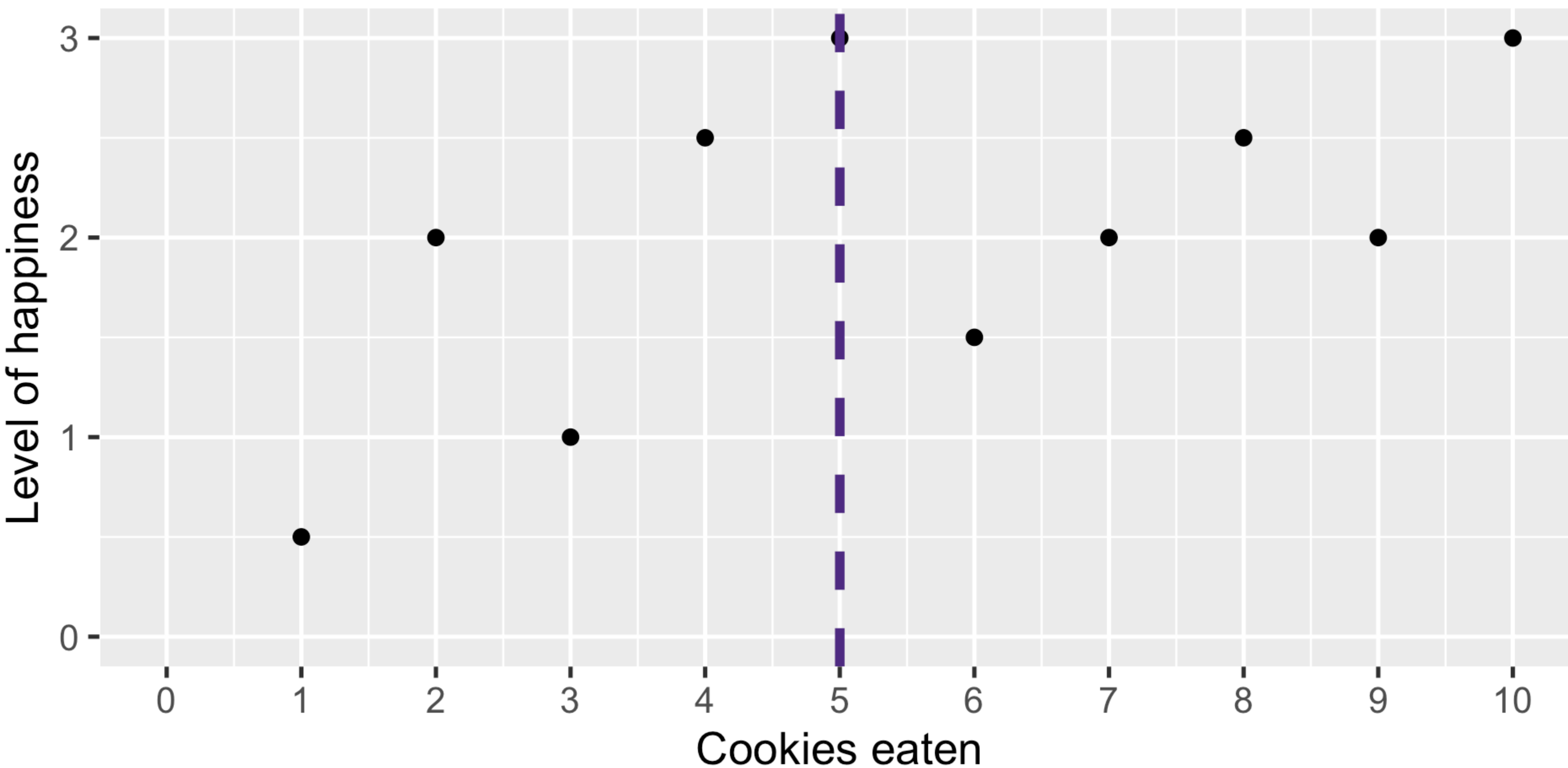
Underfitting: High bias, low variance

Supervised learning methods

Parametric: Functional form can be written as an equation (e.g. OLS)

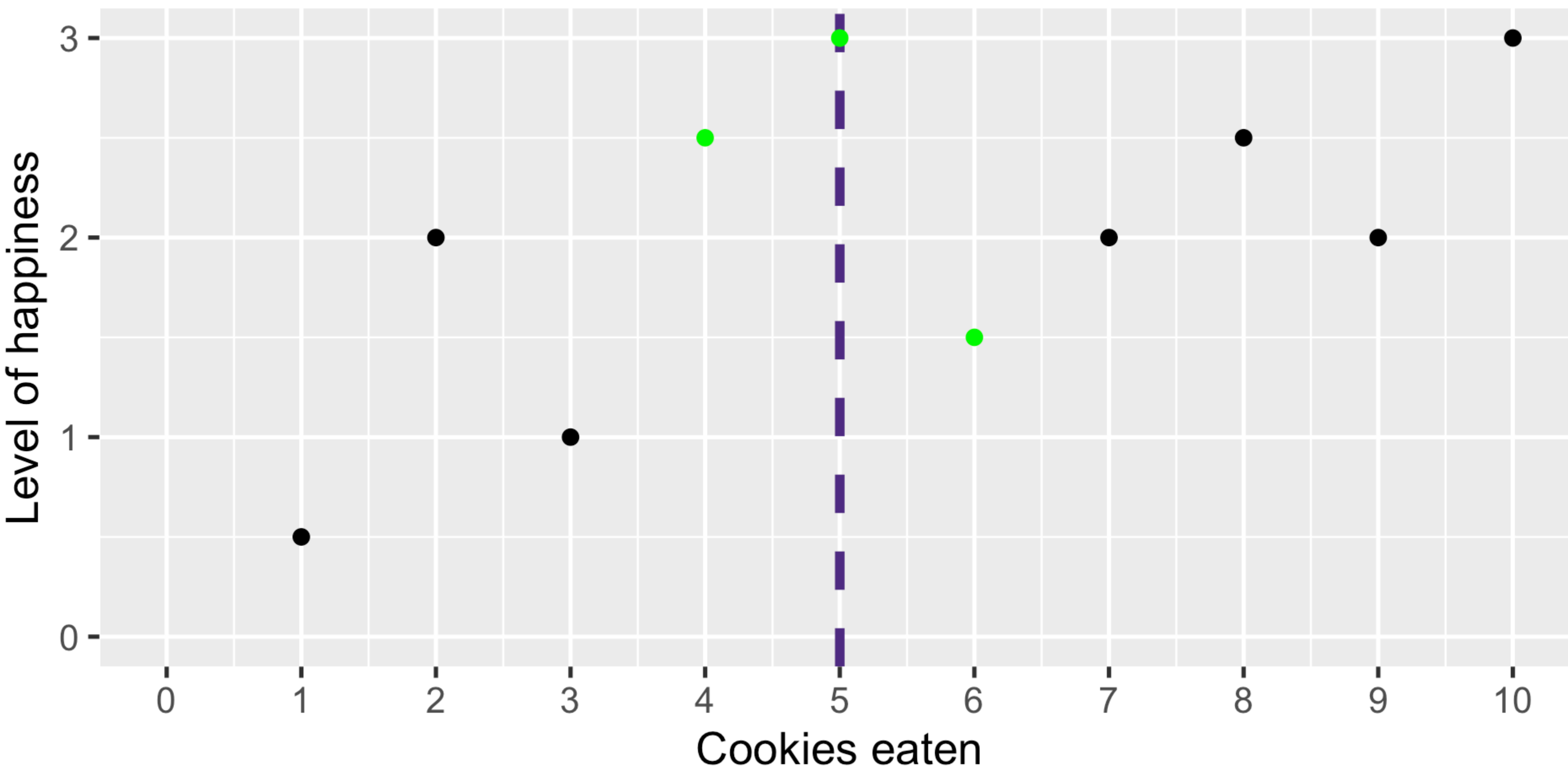
Nonparametric: Cannot be written as an equation

Nonparametric cookies



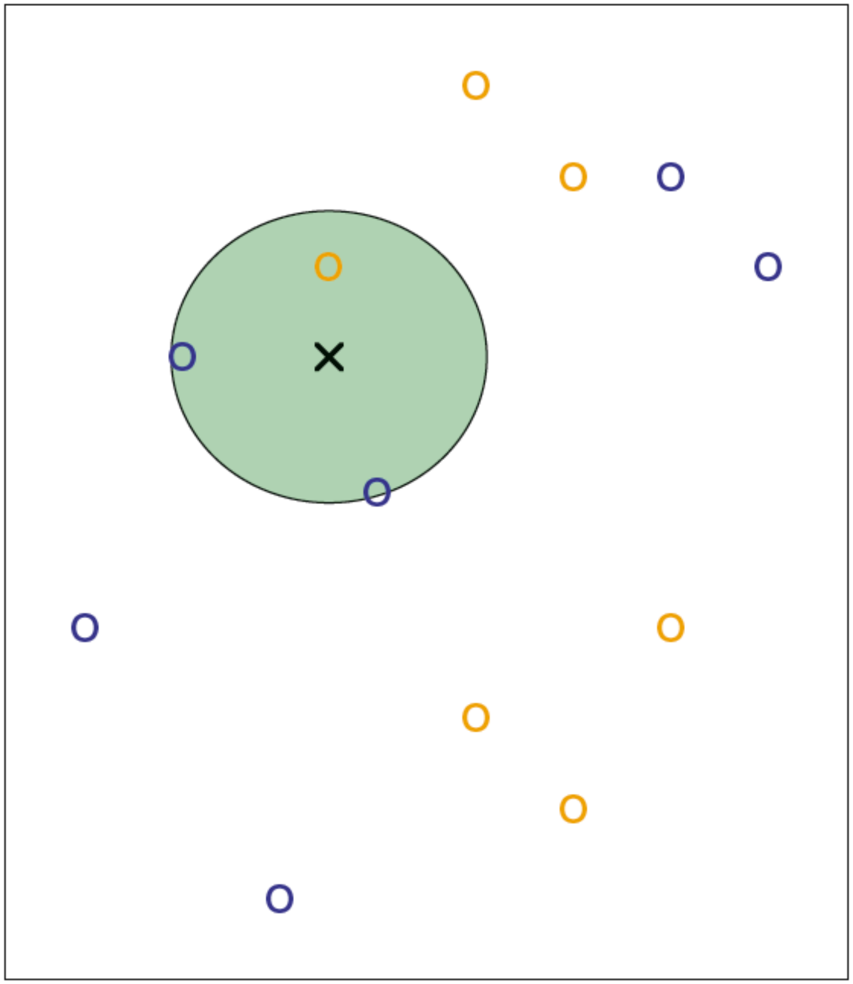
What would be a good guess for the new person?

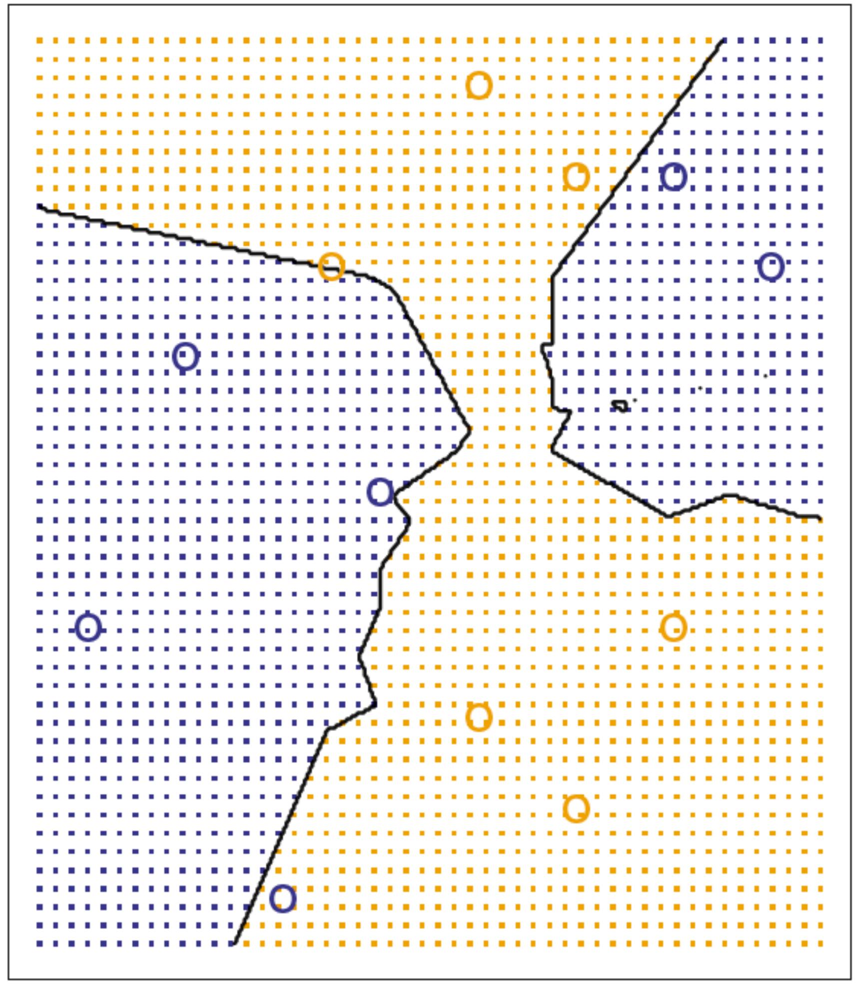
Nonparametric cookies



Observations nearby!

K-Nearest Neighbors (KNN)





KNN algorithm

For each new observation:

- Find K closest observations based on observed features
- **Regression:** Predict new value taking the average of all neighbors
- **Classification:** Predict category with highest probability among neighbors

Pros: Flexible. Works better than what you would think

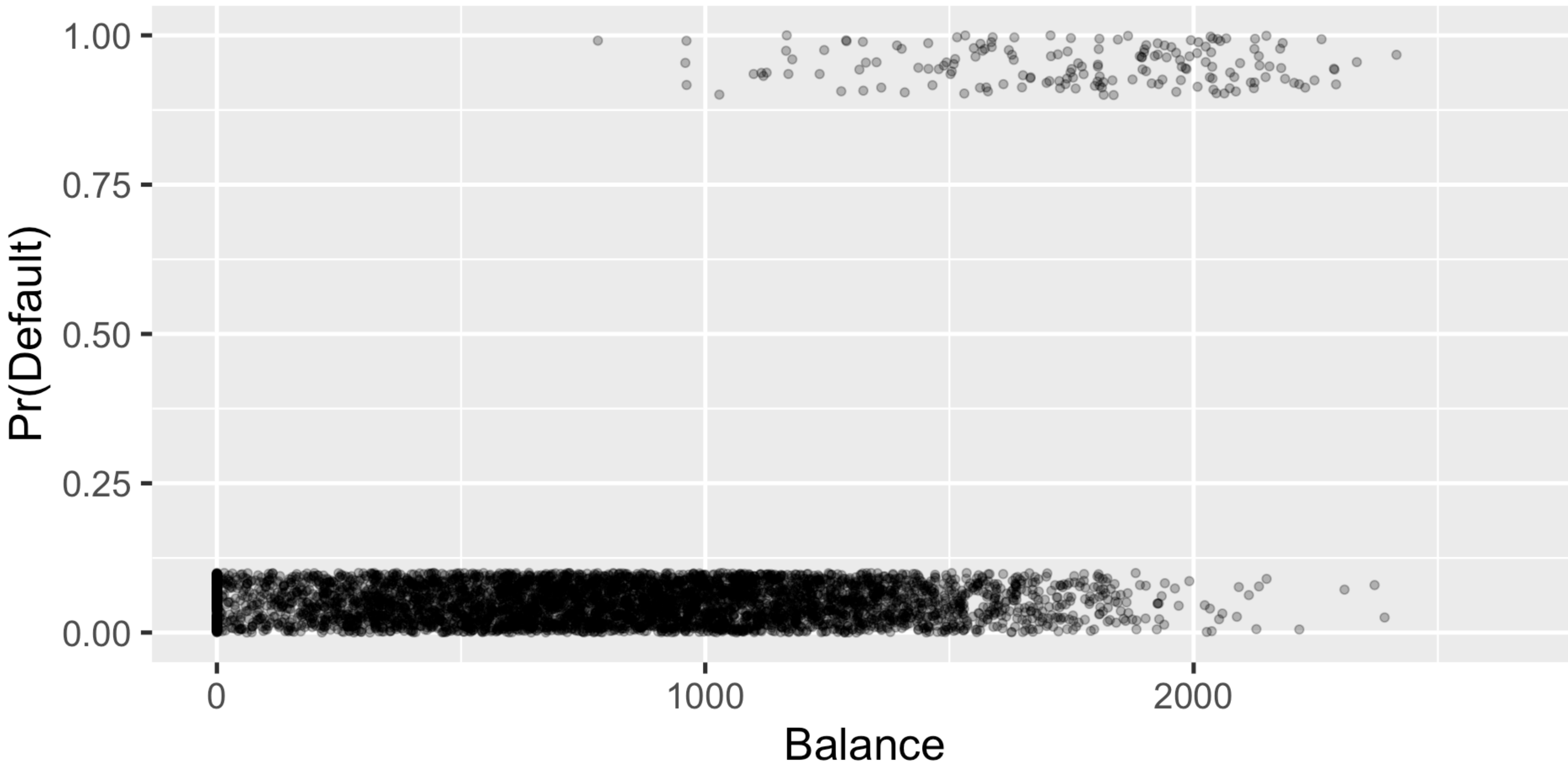
Cons: Computationally inefficient, struggles with complex data structures

Classification example

	default	student	balance	income
1	No	No	729.5264952	44361.6251
2	No	Yes	817.1804066	12106.1347
3	No	No	1073.5491640	31767.1389
4	No	No	529.2506047	35704.4939
5	No	No	785.6558829	38463.4959
6	No	Yes	919.5885305	7491.5586
7	No	No	825.5133305	24905.2266
8	No	Yes	808.6675043	17600.4513
9	No	No	1161.0578540	37468.5293
10	No	No	0.0000000	29275.2683
11	No	Yes	0.0000000	21871.0731
12	No	Yes	1220.5837530	13268.5622
13	No	No	237.0451140	28251.6953
14	No	No	606.7423433	44994.5558
15	No	No	1112.9684006	23810.1741

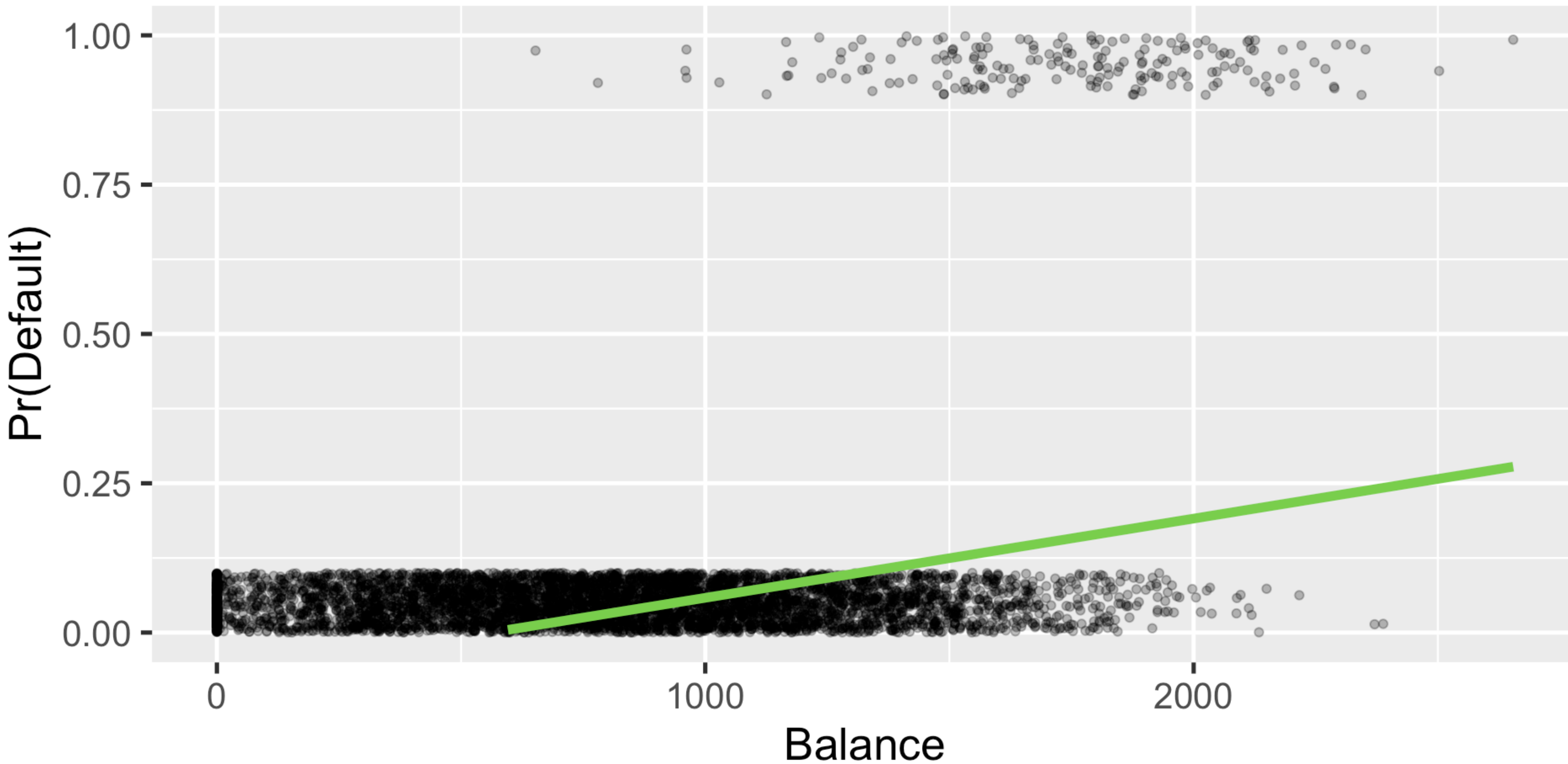
Goal: *predict* whether a customer's credit card will go on default

Visualize



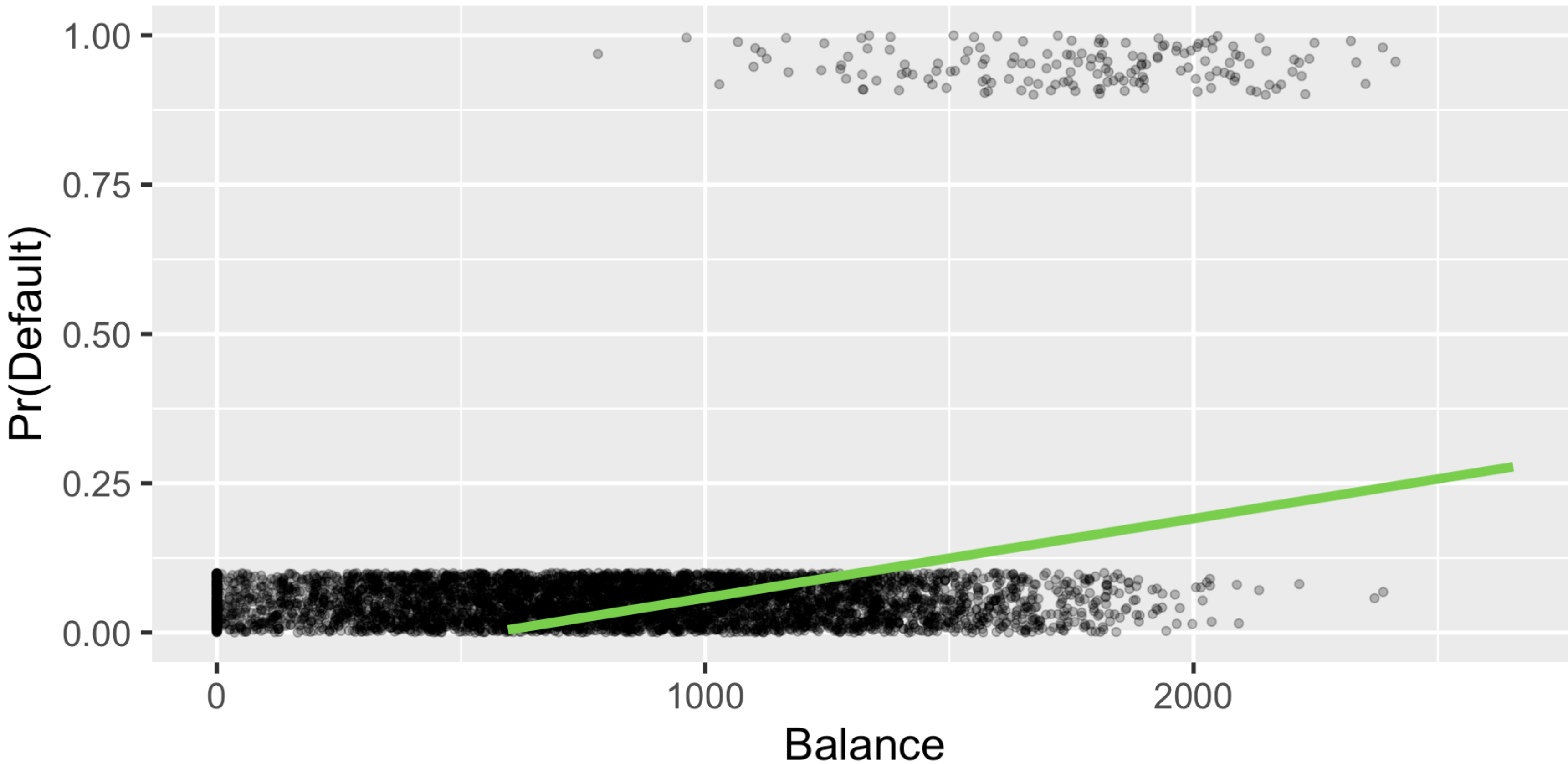
Tricky to find neighbors

Visualize



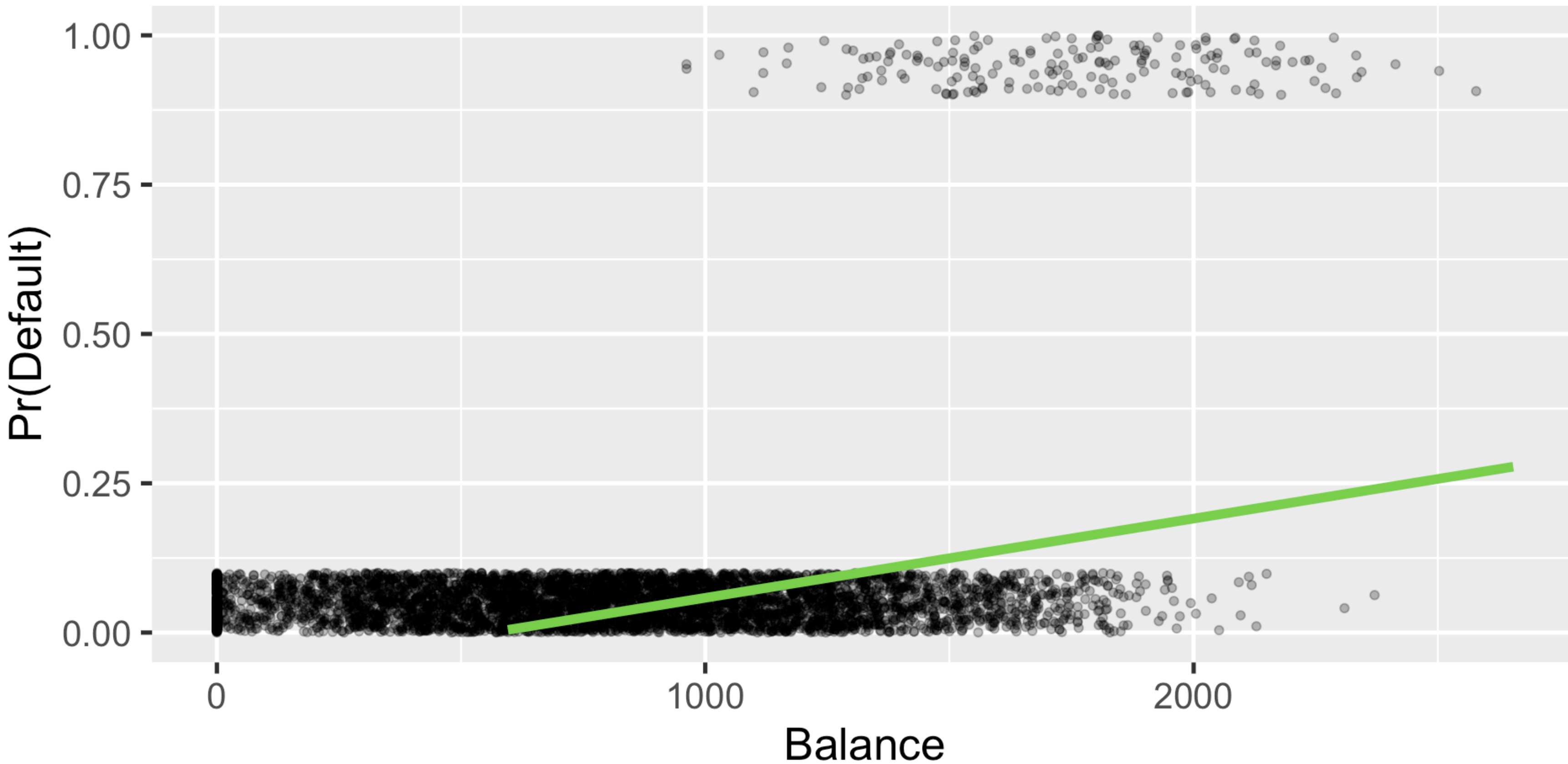
What about OLS regression?

Predicted probabilities



Not that good at catching those who may default

Predicted probabilities



Also it can (technically) exceed the 0-1 range!

What is the problem?

Before: We wanted a single number summary that characterizes the relationship and has good statistical properties

Now: We want a model that predicts new data well, we don’t care about producing precise or interpretable estimates

We also want a model that produces **valid** classifications!

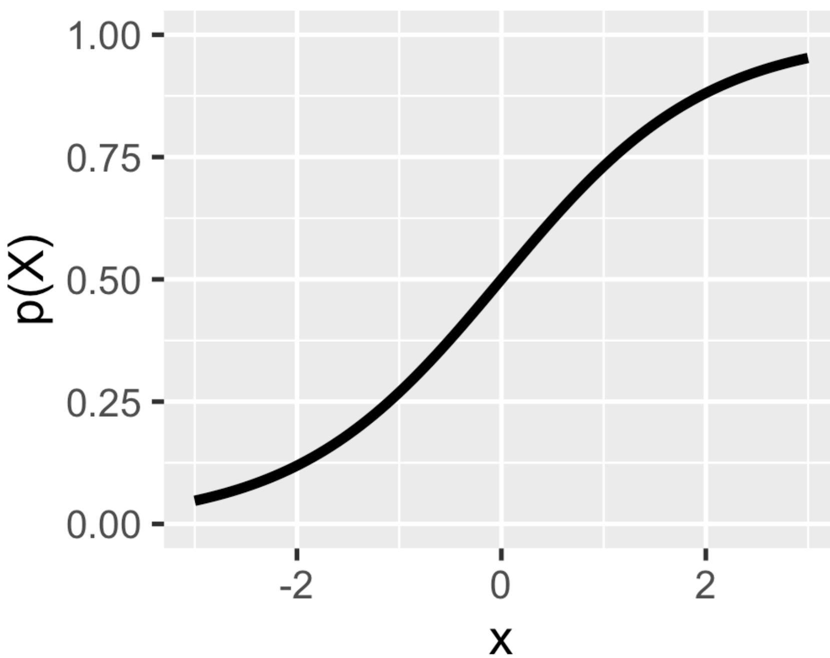
Logistic regression

- A variant of regression that respects the laws of probability
- Uses an intermediate step called a **link function**

Logistic regression

For the logit model, the link is the *logistic function*

$$p(X) = \frac{e^{X\beta}}{1 + e^{X\beta}}$$



Logistic regression

For the logit model, the link is the *logistic function*

$$p(X) = \frac{e^{X\beta}}{1 + e^{X\beta}}$$

Rearrange to get the *odds ratio*

$$\frac{p(X)}{1 - p(X)} = e^{X\beta}$$

Logistic regression

Taking the natural logarithm gives the *log odds*

$$log\left(\frac{p(X)}{1-p(X)}\right) = X\beta$$

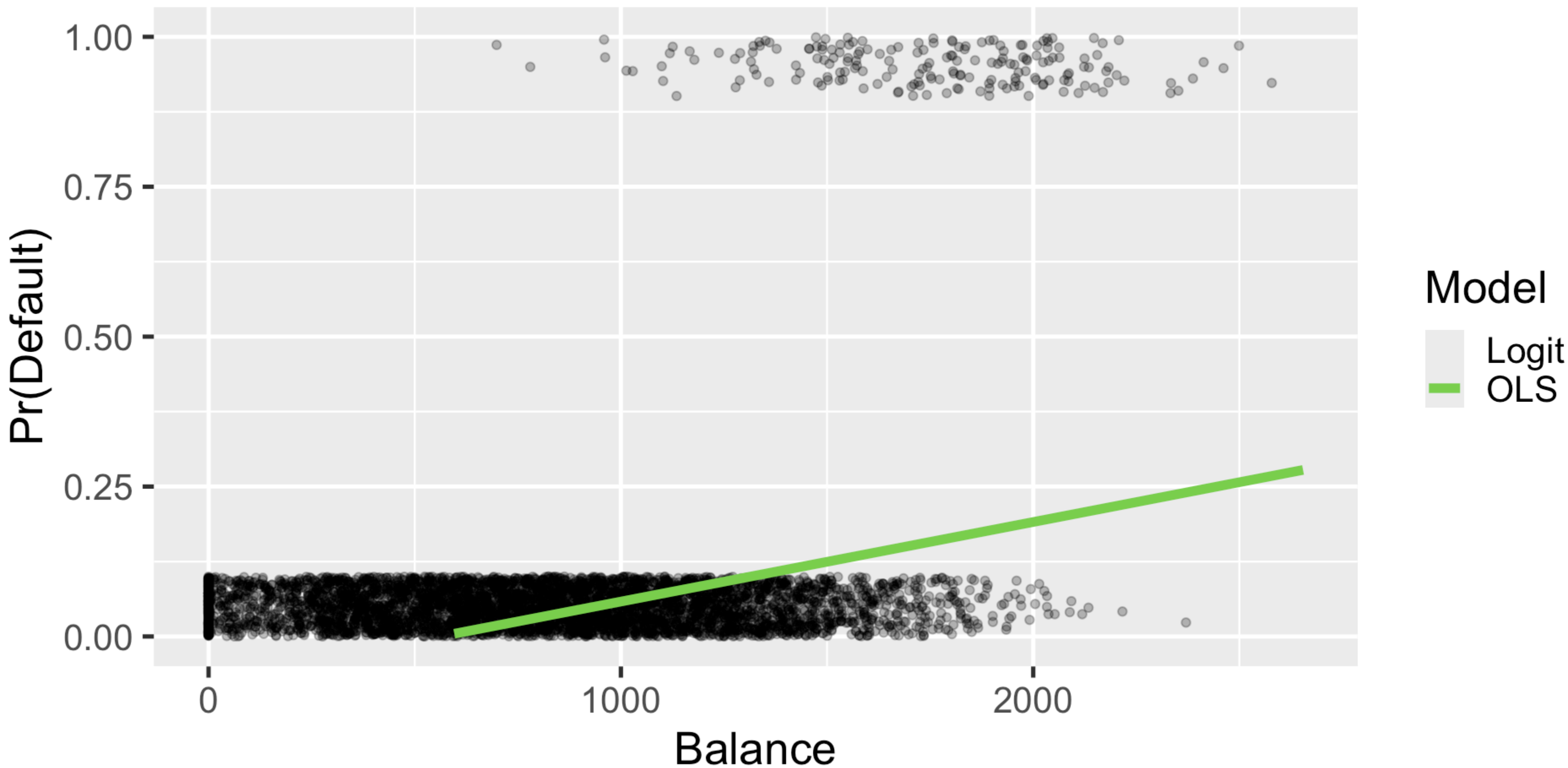
Logistic regression

Taking the natural logarithm gives the *log odds*

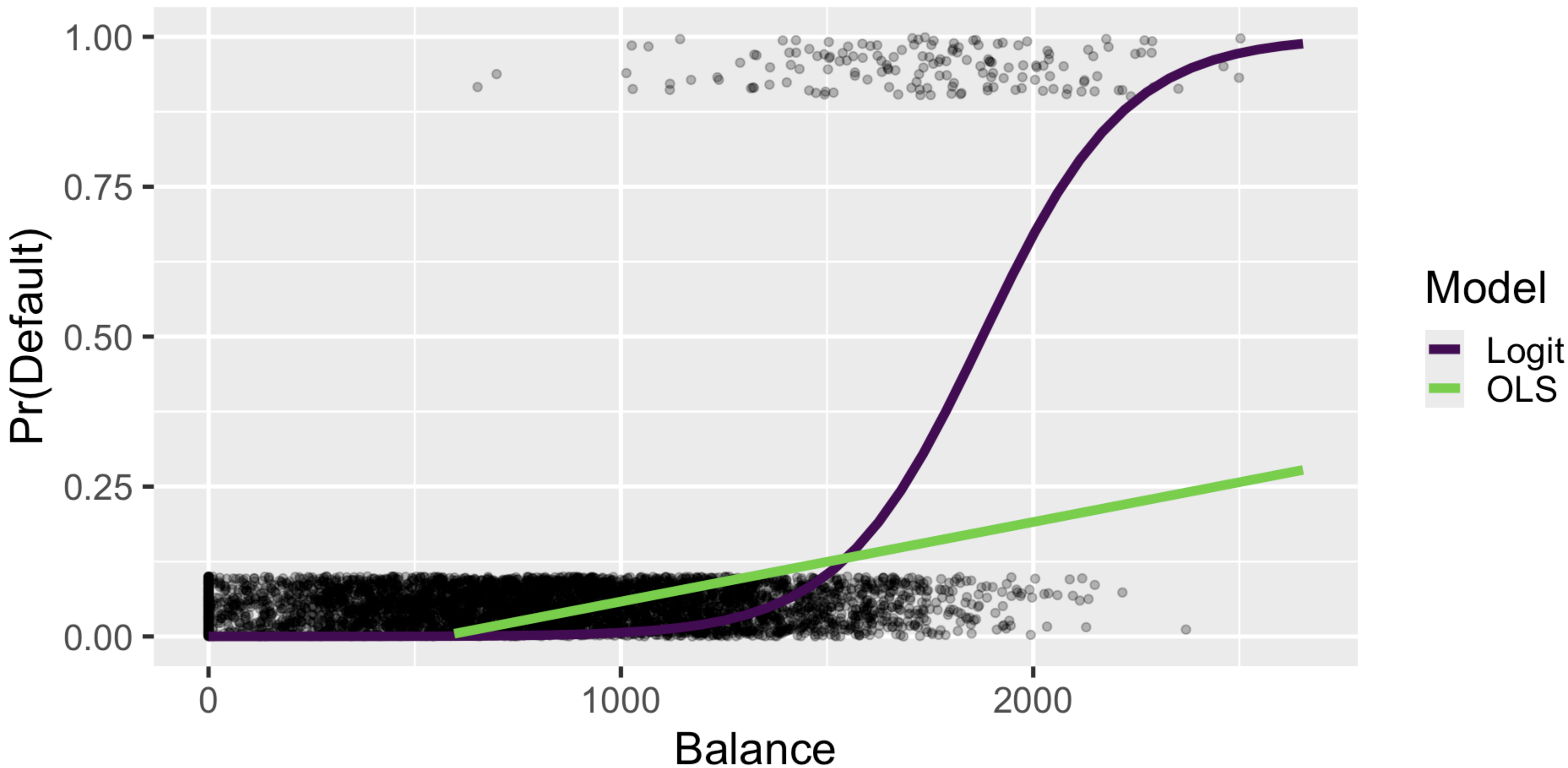
$$log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k + \varepsilon$$

- It's called *logit* because you need to *log it* to compute
- Estimate with **maximum likelihood estimation**
- Weird to interpret, but we do not care as long as we get good *classification*

What changes?



What changes?



Great! What next?

We have baby’s first machine learning models

How do we now if these (or any other fancy model) performs well?

We need a way to *quantify* **prediction error**

Error metrics

Regression

Remember this?

$$SSR = \sum_{i=1}^{n} e_i^2$$

Error metrics

Regression

Remember this?

$$SSR = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

It's our friend the **Sum of Squared Residuals**!

You used to be a *criterion* to minimize so that we could draw good lines

Now you are an **error metric**

Error metrics

Regression

$$SSR = \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$$

But not with those clothes!

Error metrics

Regression

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{N}$$

But not with those clothes!

Now you are a **Mean Squared Error**

But you could look prettier!

Error metrics

Regression

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}{N}}$$

You are a **Root Mean Squared Error**

You are now expressed on *response variable* units ❤

↓ RMSE ⇒ Better prediction

Error metrics

Classification

	Actual	
Predicted	**False (0)**	**True (1)**
False (0)	True Negative (**TN**)	False Negative (**FN**)
True (1)	False Positive (**FP**)	True Positive (**TP**)

We can use these to calculate several metrics

Error metrics

Classification

Name	Measurement	Note

Error metrics

Classification

Name	Measurement	Note
Error rate	$Avg(I(y_i \neq \widehat{y}_i))$	Proportion actual $\neq$ predicted

Error metrics

Classification

Name	Measurement	Note
Error rate	$Avg(I(y_i \neq \widehat{y}_i))$	Proportion actual $\neq$ predicted
Accuracy	$1 - \text{error rate}$	Proportion correct

Error metrics

Classification

Name	Measurement	Note
Error rate	$Avg(I(y_i \neq \widehat{y}_i))$	Proportion actual $\neq$ predicted
Accuracy	$1 - \text{error rate}$	Proportion correct
Accuracy	$(TN + TP)/n$	Proportion correct

Error metrics

Classification

Name	Measurement	Note
Error rate	$Avg(I(y_i \neq \widehat{y}_i))$	Proportion actual $\neq$ predicted
Accuracy	$1 - \text{error rate}$	Proportion correct
Accuracy	$(TN + TP)/n$	Proportion correct
Sensitivity	$TP/(TP+FN)$	Proportion correct positives

Error metrics

Classification

Name	Measurement	Note
Error rate	$Avg(I(y_i \neq \widehat{y}_i))$	Proportion actual $\neq$ predicted
Accuracy	$1 - \text{error rate}$	Proportion correct
Accuracy	$(TN + TP)/n$	Proportion correct
Sensitivity	$TP/(TP+FN)$	Proportion correct positives
Specificity	$TN/(TN+FP)$	Proportion correct negatives

Hold on

Aren’t these metrics assuming that we **know** true positives/negatives?

How do we calculate if we don’t know?

More next time!

Machine Learning 🤖

POLI SCI 210

Introduction to Empirical Methods in Political Science

Last time

Error metrics in machine learning

Regression: Root Mean Squared Error (RMSE)

Classification: Error rate, accuracy, sensitivity, specificity

These require *actual* and *predicted* values

But why predict if you know *actual* values?

Remember: We are doing this to learn about *new data*

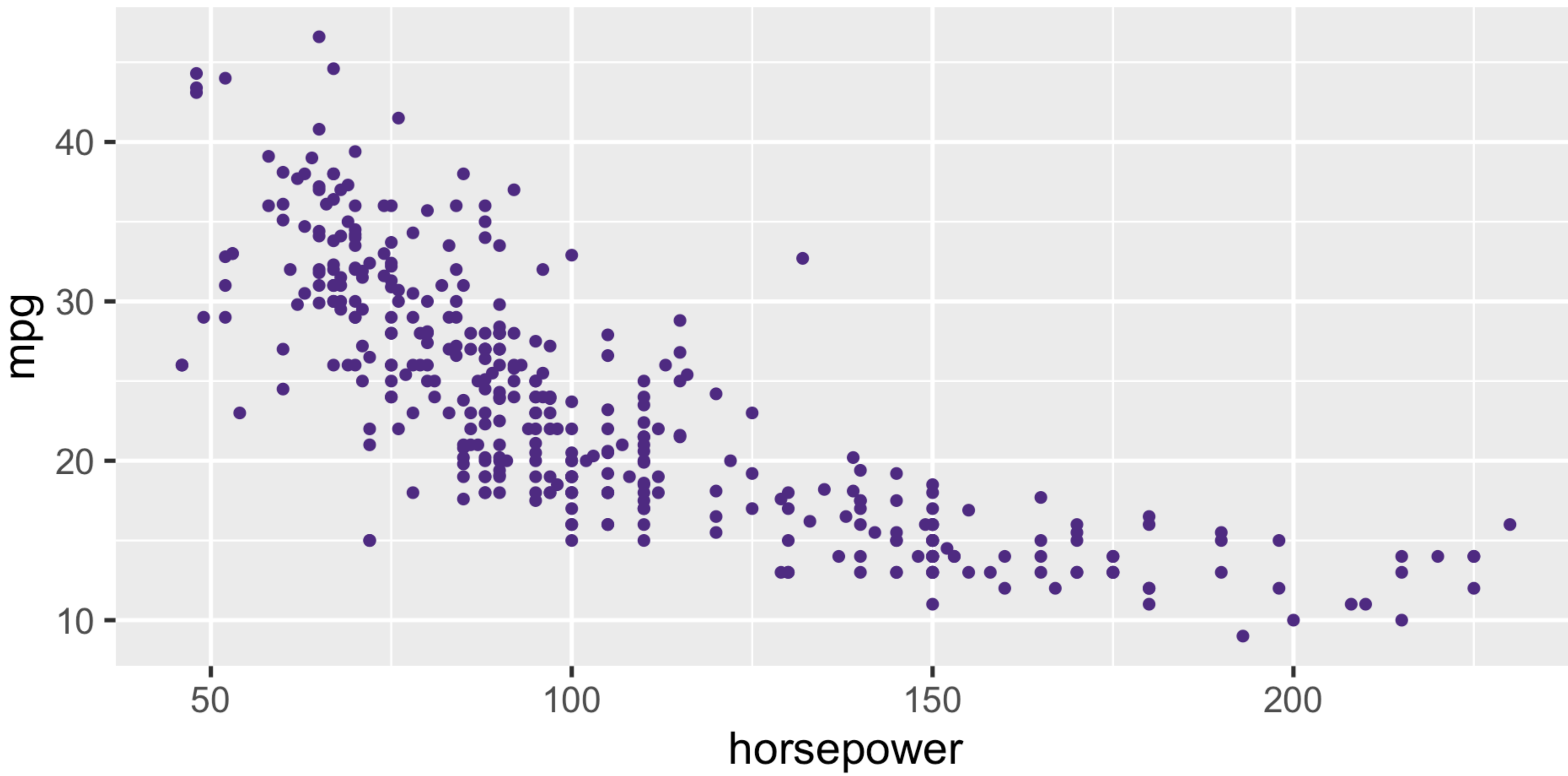
Resampling methods

General idea: Use existing data to mimic predicting new data

Easiest: Validation set approach

- Split data into **training** and **validation** set
- Normally via random sampling
- Usually larger training set
- Generate predictions on training set
- Evaluate performance on validation set

Example: Auto data



OLS models

Linear: $\widehat{\texttt{mpg}} = \beta_0 + \beta_1 \texttt{horsepower}$

Quadratic:

$\widehat{\texttt{mpg}} = \beta_0 + \beta_1 \texttt{horsepower} + \beta_2 \texttt{horsepower}^2$

Cubic:

$\widehat{\texttt{mpg}} = \beta_0 + \beta_1 \texttt{horsepower} + \beta_2 \texttt{horsepower}^2 + \beta_3 \texttt{horsep}$

More polynomial terms → more curvy

50/50 train/validation split at random

Choose model that would predict new data better

Results

Fit	RMSE
Linear	4.82
Quadratic	4.33
Cubic	4.34

Notice how results change based on train/validation split

Results

	RMSE	
Fit	**Split 1**	**Split 2**
Linear	4.82	5.03
Quadratic	4.33	4.47
Cubic	4.34	4.47

Fancier *resampling methods* take advantage of this to provide more robust performance

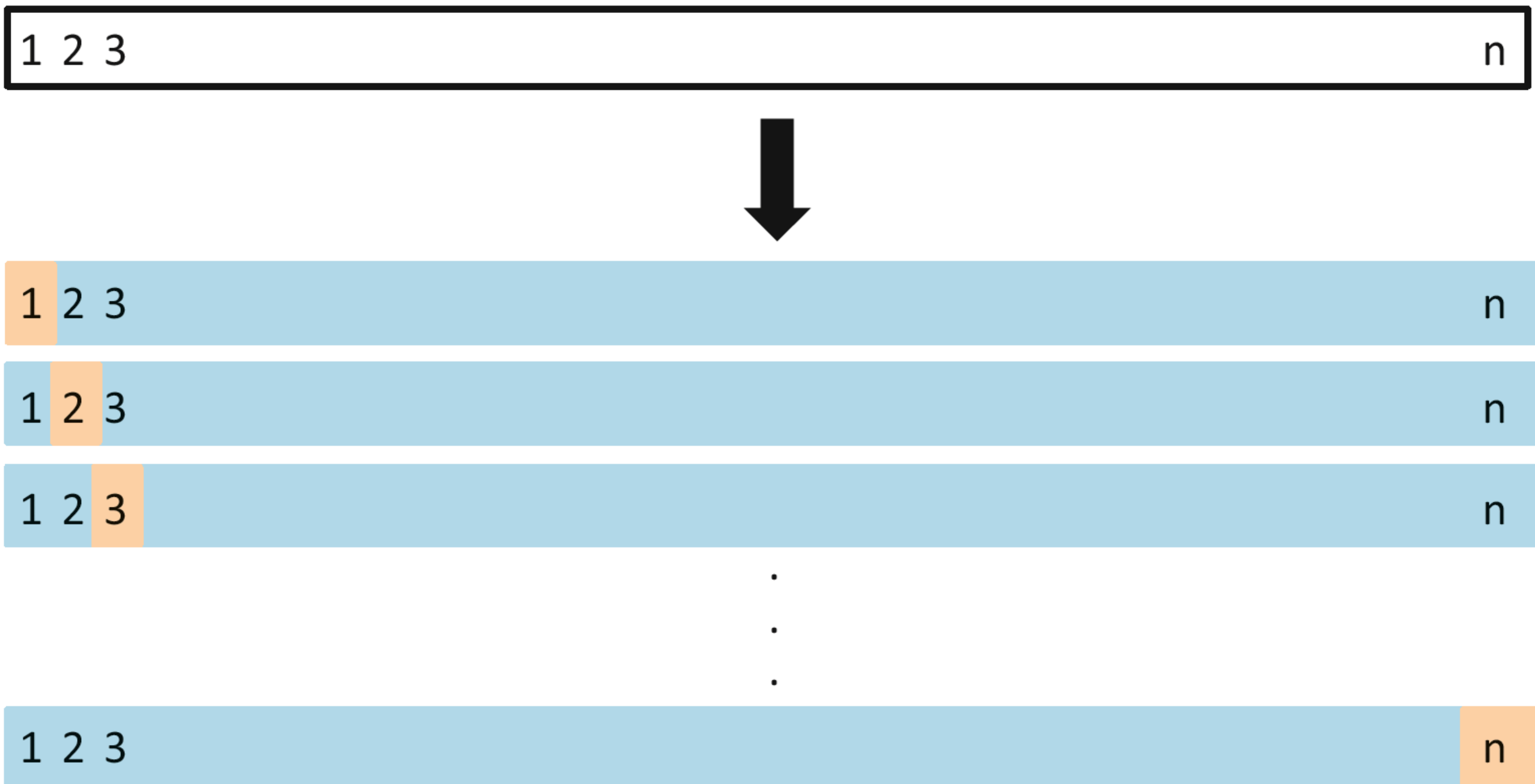
Cost: Increased computing times (but trivial for consumer-level tasks)

Example: Cross-validation

Idea: Do many train-validation splits and then average over their performance

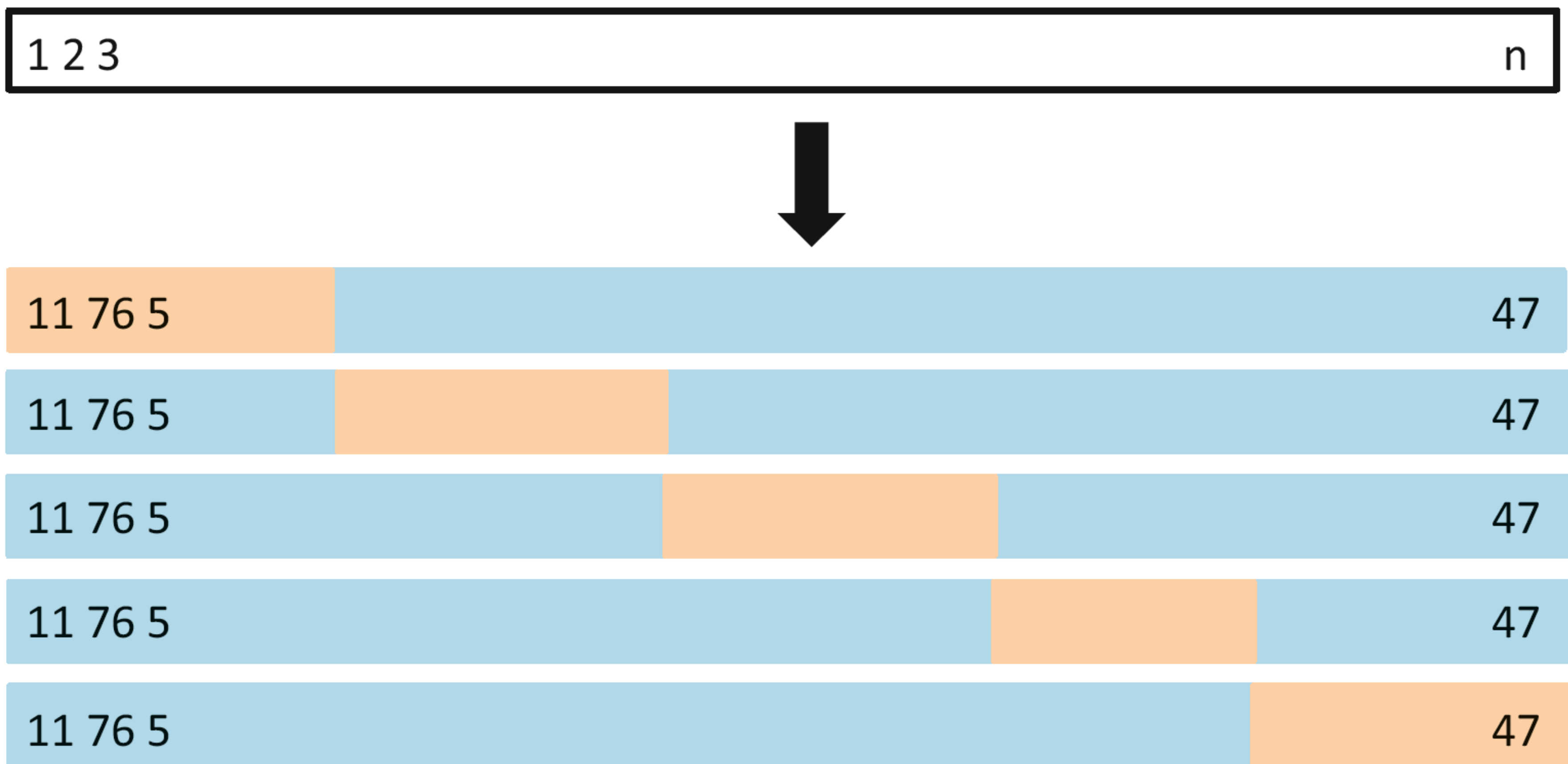
Example: Cross-validation

Leave-**O**ne-**O**ut **C**ross-**V**alidation (LOOCV)



Example: Cross-validation

K-fold cross-validation



Application: credit cards data

	default	student	balance	income
1	No	No	729.5264952	44361.6251
2	No	Yes	817.1804066	12106.1347
3	No	No	1073.5491640	31767.1389
4	No	No	529.2506047	35704.4939
5	No	No	785.6558829	38463.4959
6	No	Yes	919.5885305	7491.5586
7	No	No	825.5133305	24905.2266
8	No	Yes	808.6675043	17600.4513
9	No	No	1161.0578540	37468.5293
10	No	No	0.0000000	29275.2683
11	No	Yes	0.0000000	21871.0731
12	No	Yes	1220.5837530	13268.5622
13	No	No	237.0451140	28251.6953
14	No	No	606.7423433	44994.5558
15	No	No	1112.9684006	23810.1741

Goal: Predict who will default on their credit card

Algorithms

Logistic regression:

$$\widehat{p}(\texttt{default}) = \beta_0 + \beta_1 \texttt{income} + \beta_2 \texttt{balance} + \beta_3 \texttt{student}$$

Compare with **KNN** (5, 10, 20)

All tuned with *5-fold CV*

Results

Algorithm	k
Logit	
KNN	5
KNN	10
KNN	20

Results

Algorithm	k	Accuracy
Logit		0.97
KNN	5	0.97
KNN	10	0.97
KNN	20	0.97

Results

Algorithm	k	Accuracy	Sensitivity
Logit		0.97	0.312
KNN	5	0.97	0.159
KNN	10	0.97	0.069
KNN	20	0.97	0.012

Results

Algorithm	k	Accuracy	Sensitivity	Specificity
Logit		0.97	0.312	1
KNN	5	0.97	0.159	1
KNN	10	0.97	0.069	1
KNN	20	0.97	0.012	1

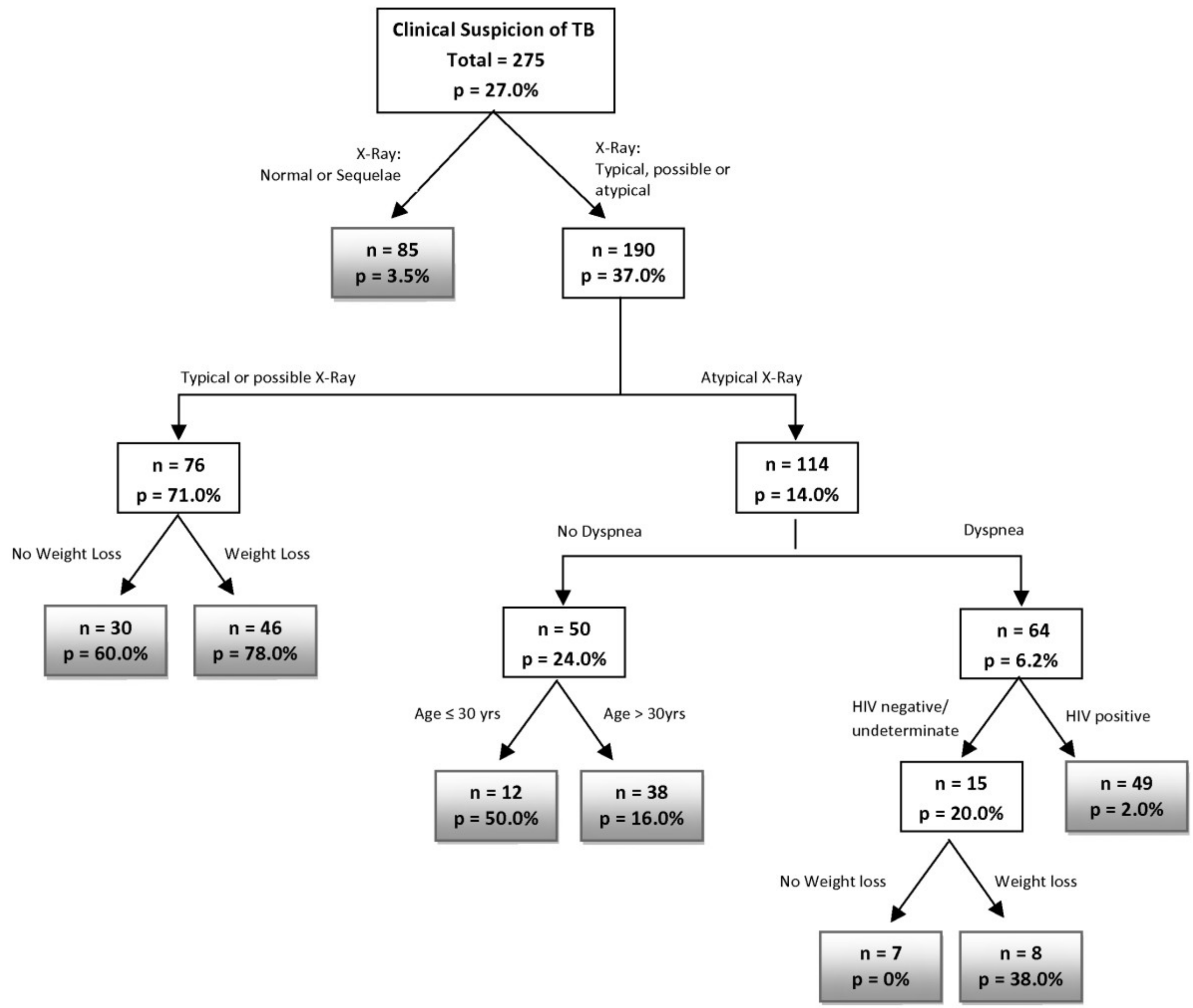
Which one seems more appropriate?

Fancier models

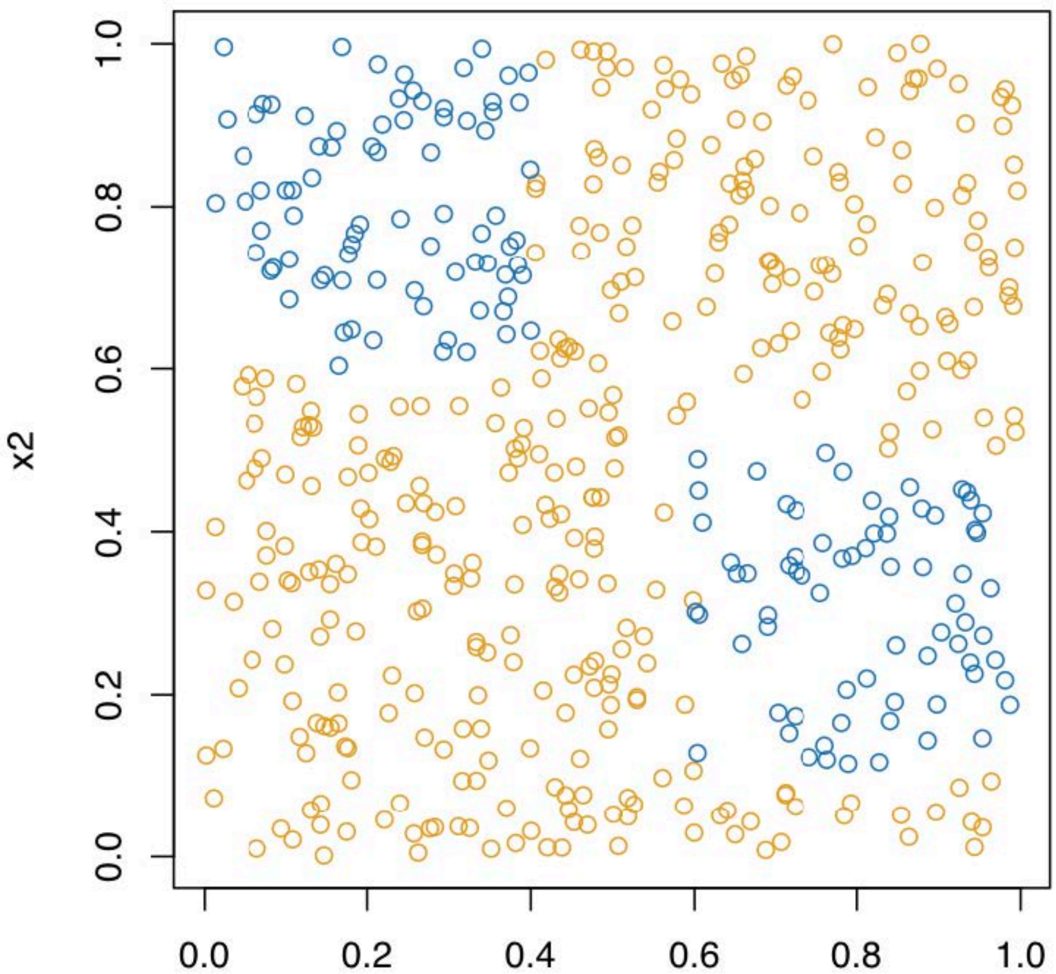
- Achieve even higher performance at the expense of even more computing power
- Can be **parametric** or **nonparametric**
- Technically, they all have *tuning parameters*
- **Difference:** Functional form assumptions

These are a few examples at the limits of consumer-level computing power

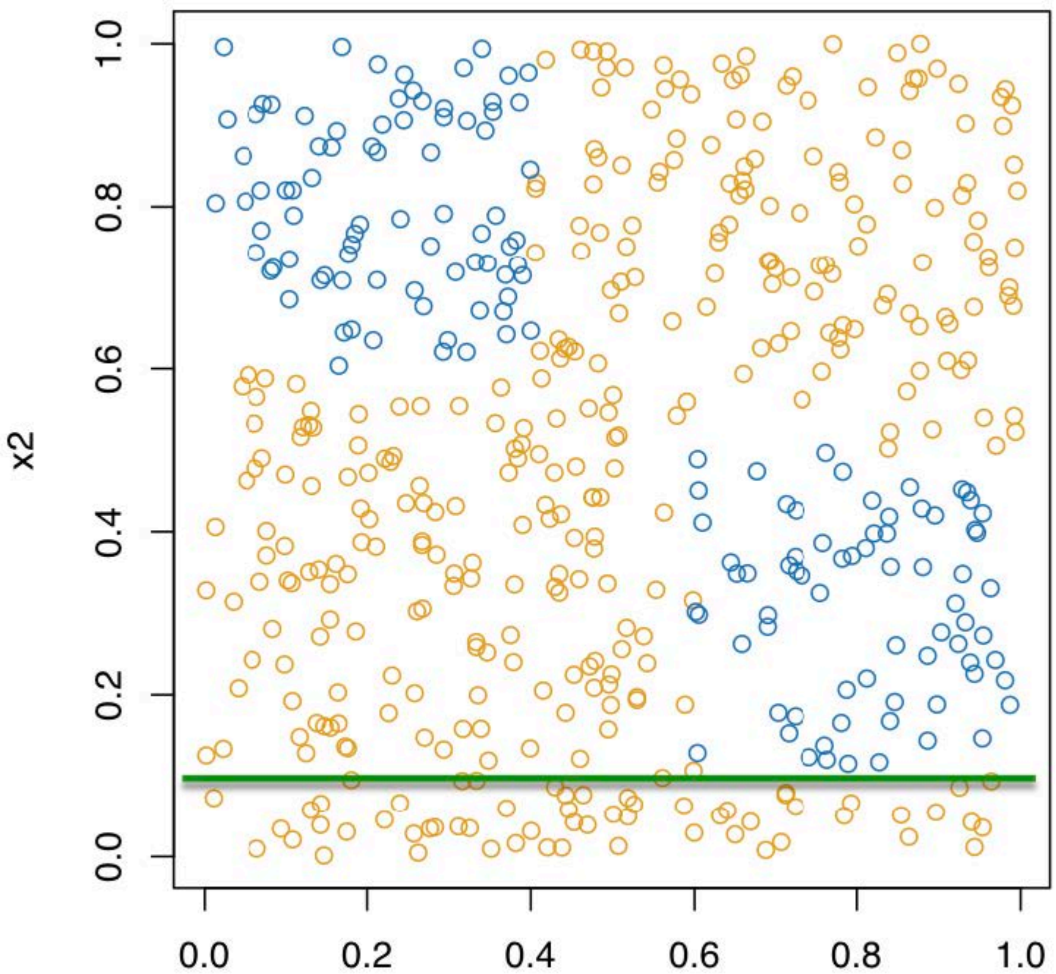
Regression/classification trees



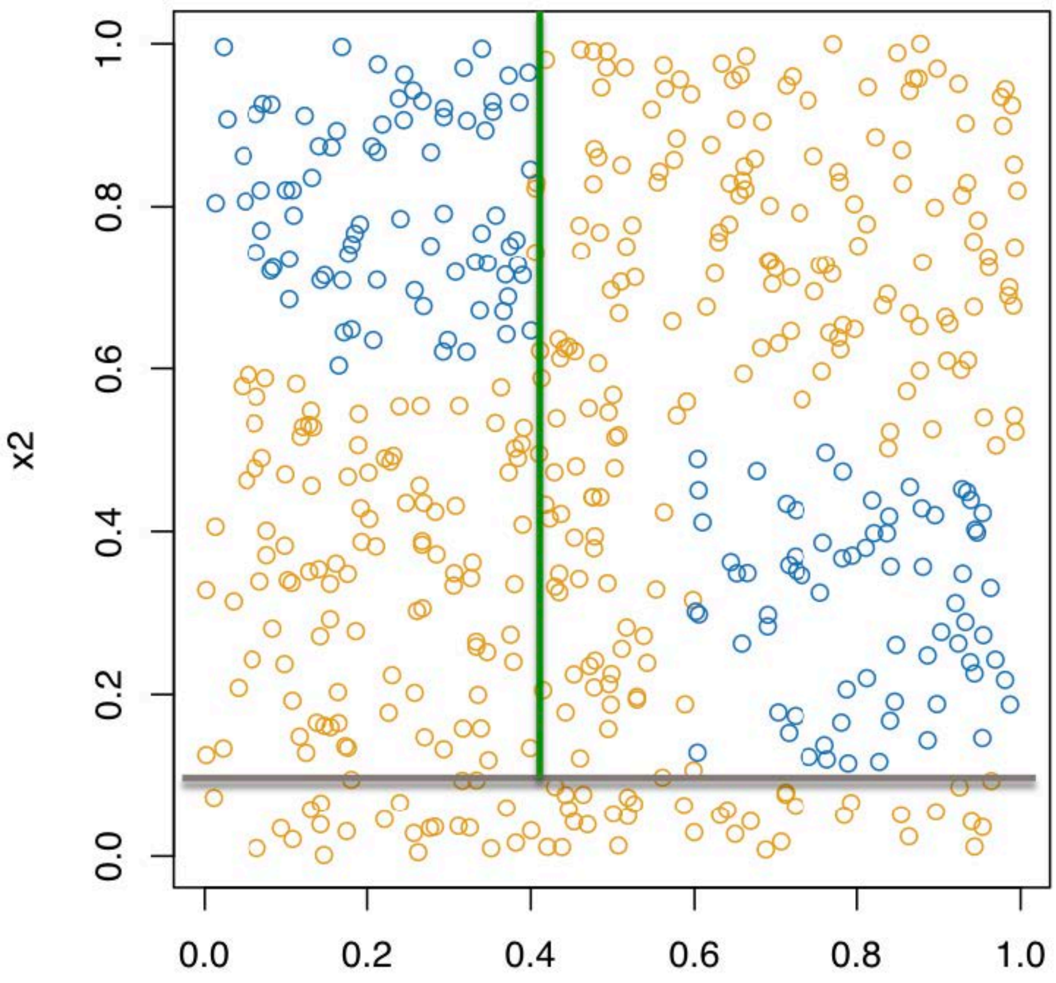
Algorithm: Recursive binary partitioning



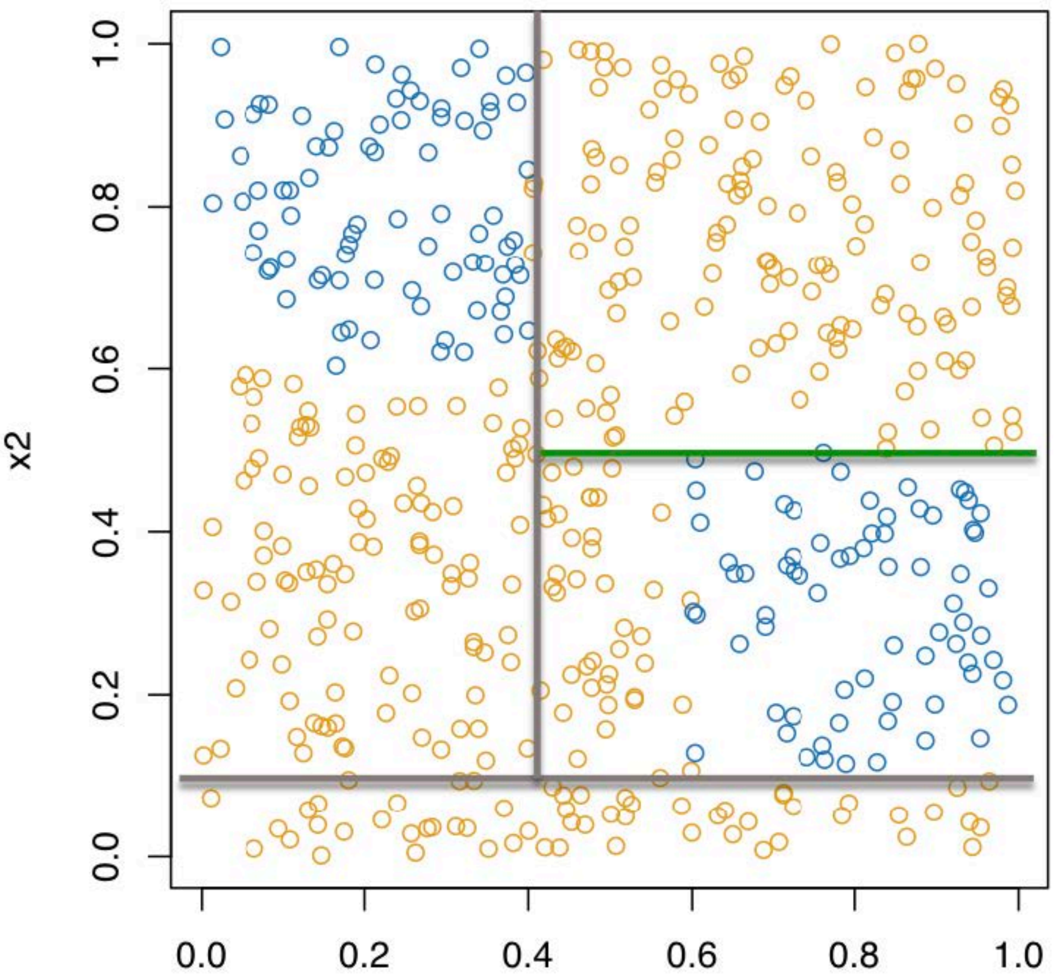
Algorithm: Recursive binary partitioning



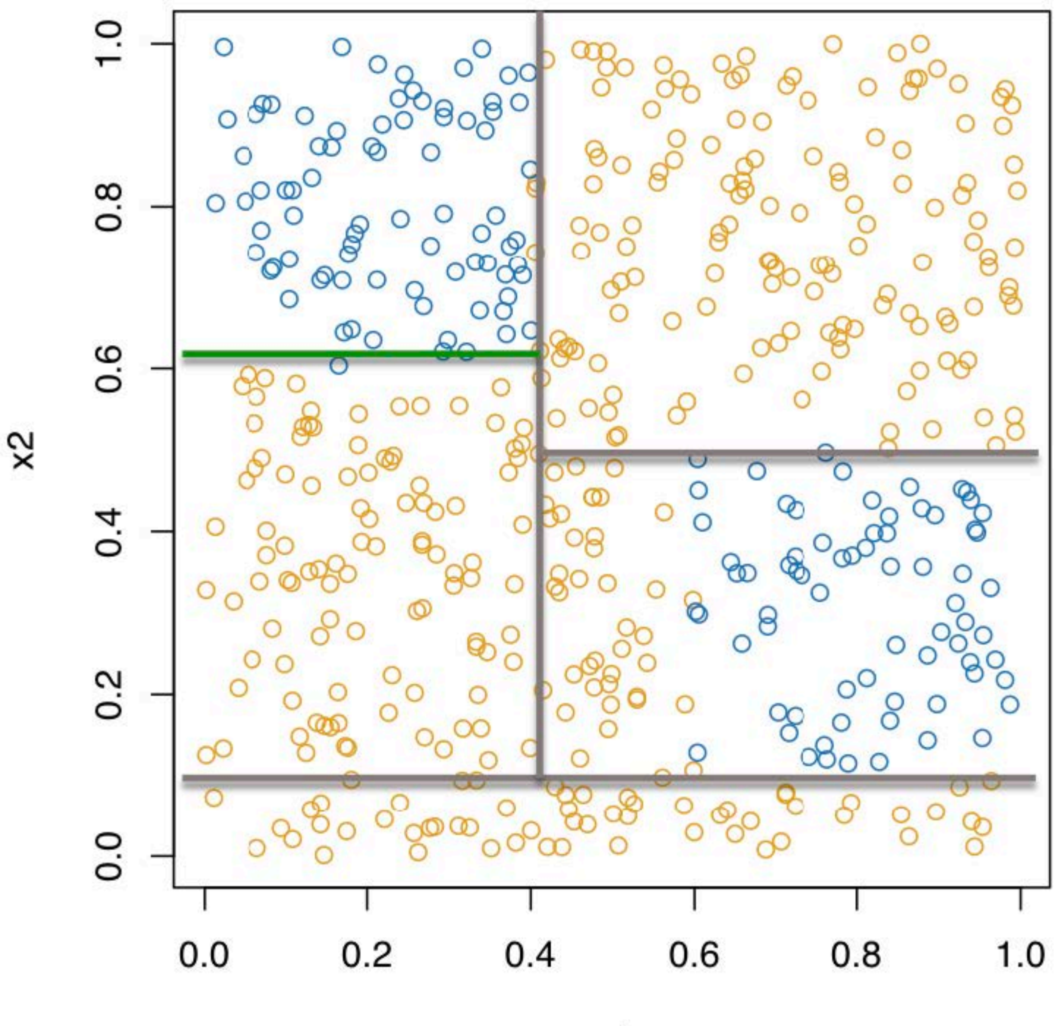
Algorithm: Recursive binary partitioning



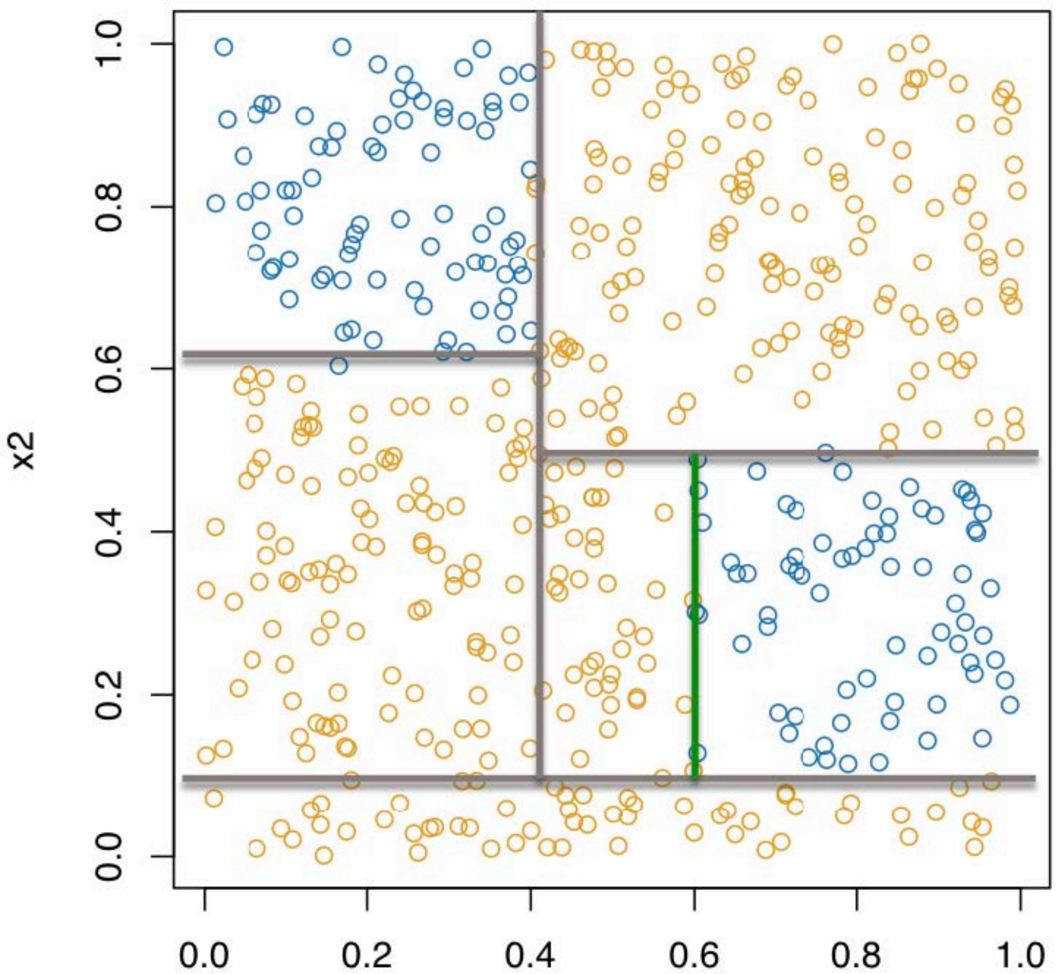
Algorithm: Recursive binary partitioning



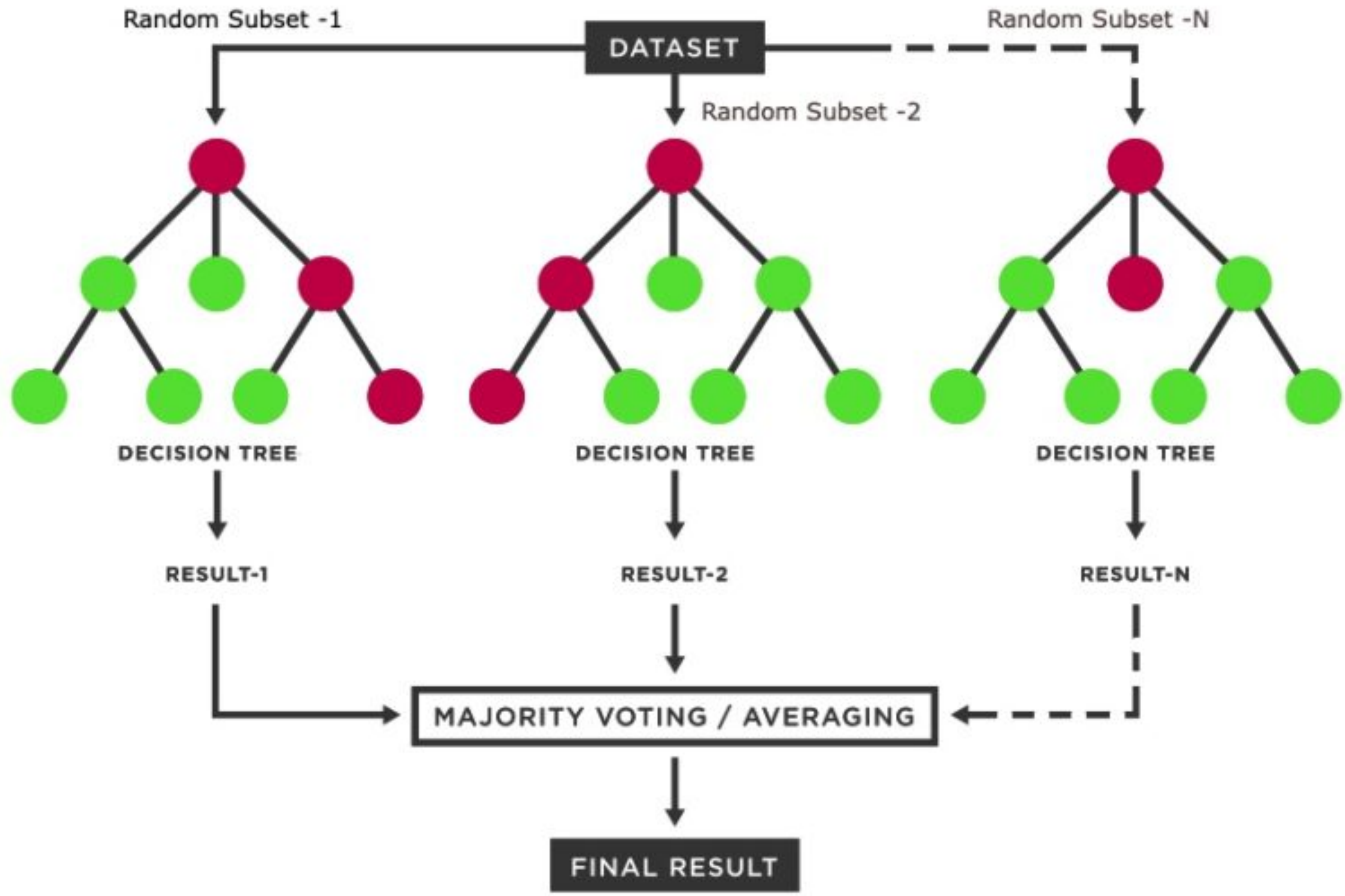
Algorithm: Recursive binary partitioning



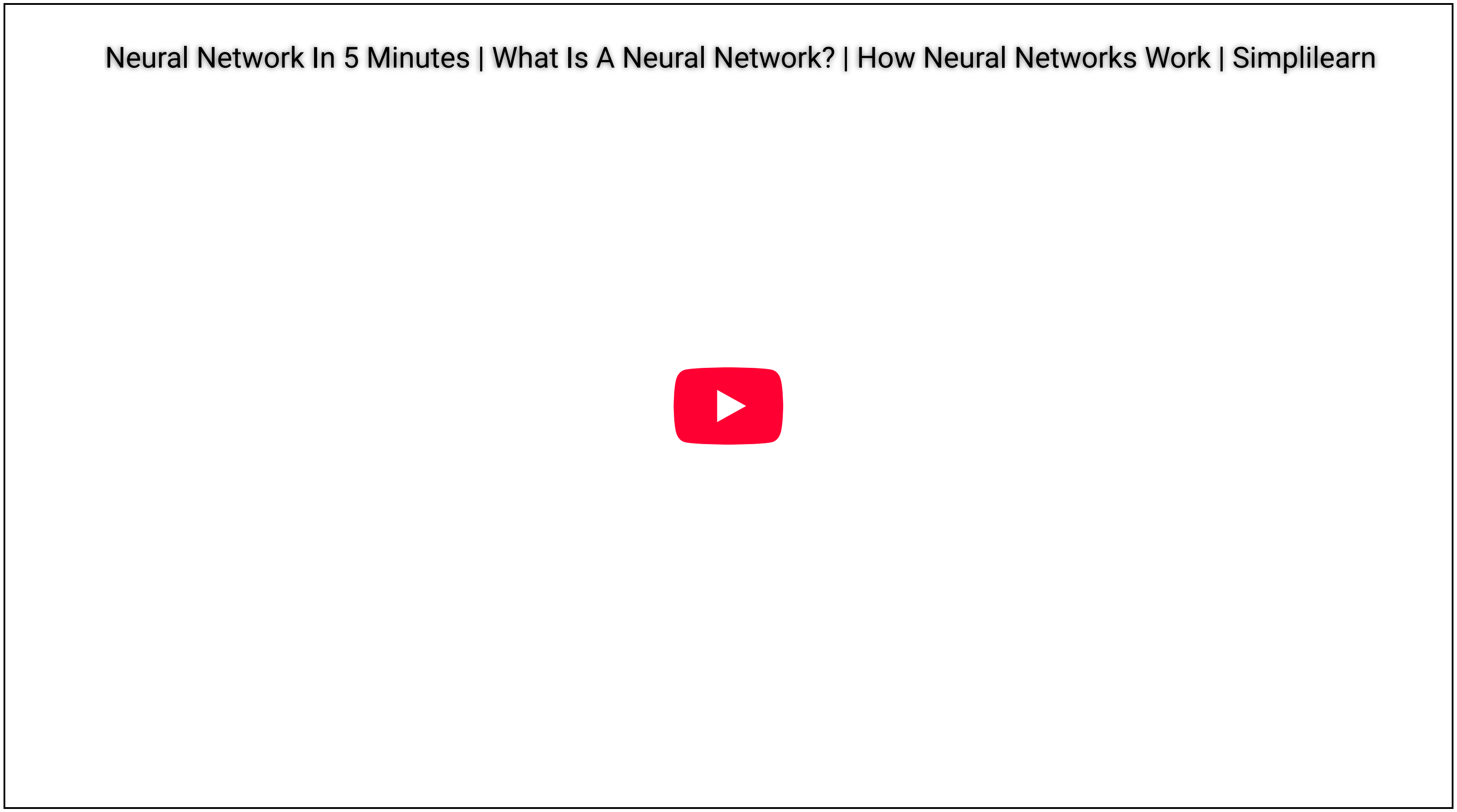
Algorithm: Recursive binary partitioning



Random forests



Neural networks



https://youtu.be/bfmFfD2RIcg?si=xAEVyJ3BKr2JCQzF

Beyond consumer-level

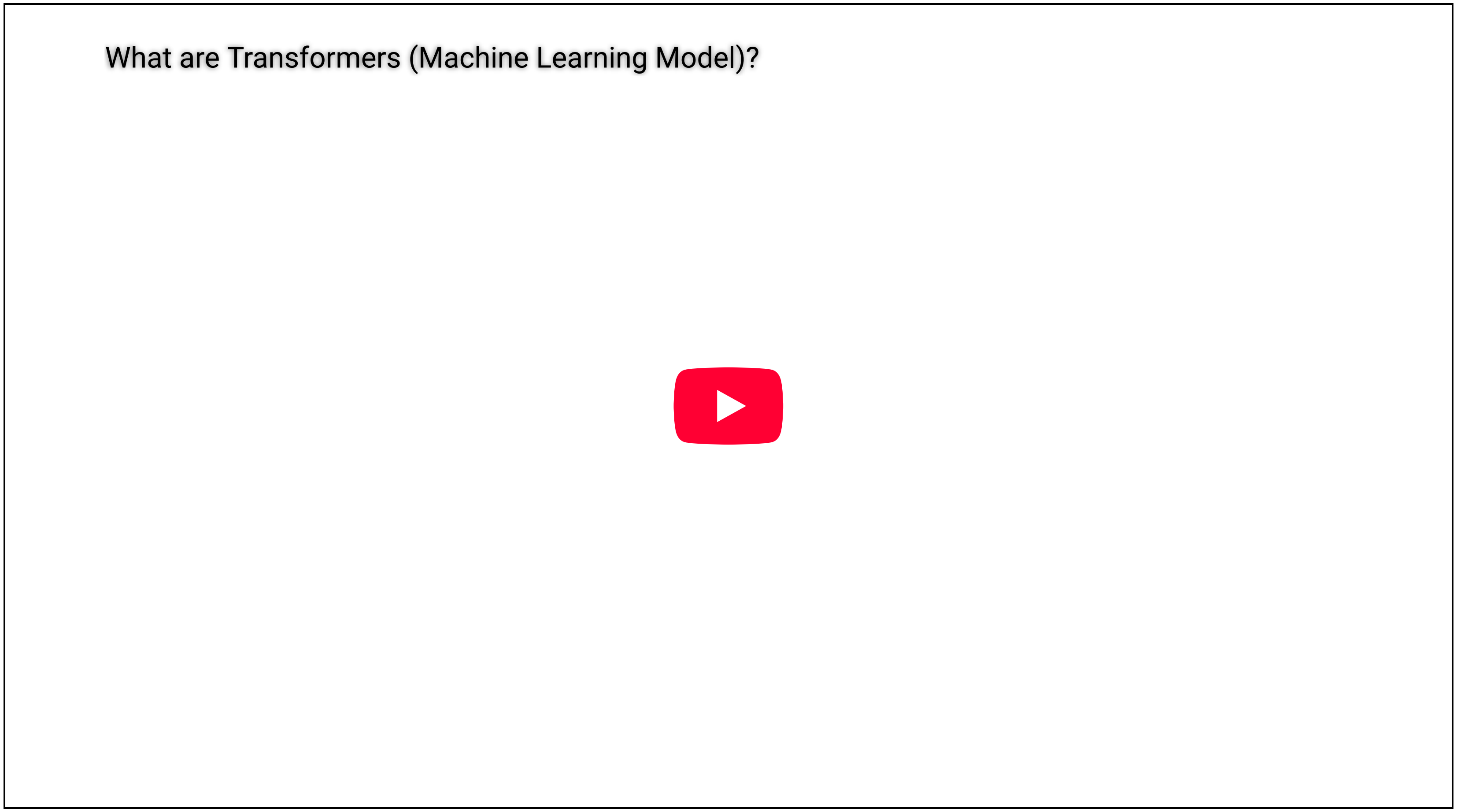
Mostly **deep learning** models trained on vast amounts of **unstructured** data, then used to create *new* data

Combination of **extractive** and **generative** AI

- **Extractive:** Learns patterns, gives structure (supervised/unsupervised learning)
- **Generative:** Creates new information

The magic behind *generative AI* as it exists today is the **transformer architecture**

Transformer architecture



https://youtu.be/ZXiruGOCn9s?si=jtJsAzNs8O5UDlwO

Questions?

Bye!