Machine Learning (in One Lecture)

EC 607, Set 12

Edward Rubin
Spring 2020

Prologue

Schedule

Last time

Resampling methods

Today

A one-lecture introduction to machine-learning methods

Upcoming

The end is near. As is the final.

Prediction: What's the goal?

Prediction: What's the goal?

What's different?

Machine-learning methods focus on **prediction**. What's different?

Prediction: What's the goal?

What's different?

Machine-learning methods focus on **prediction**. What's different?

Up to this point, we've focused on causal **identification/inference** of β, *i.e.*,

$$\text{Y}_i = \mathbf{X}_i \beta + u_i$$

meaning we want an unbiased (consistent) and precise estimate $\hat{\beta}$.

Prediction: What's the goal?

What's different?

Machine-learning methods focus on **prediction**. What's different?

Up to this point, we've focused on causal **identification/inference** of β, *i.e.*,

$$\text{Y}_i = \mathbf{X}_i \beta + \boldsymbol{u}_i$$

meaning we want an unbiased (consistent) and precise estimate $\hat{\beta}$.

With **prediction**, we shift our focus to accurately estimating outcomes.

In other words, how can we best construct $\hat{\text{Y}}_i$?

Prediction: What's the goal?

... so?

So we want "nice"-performing estimates $\hat{y}$ instead of $\hat{\beta}$.

Q Can't we just use the same methods (*i.e.*, OLS)?

Prediction: What's the goal?

... so?

So we want "nice"-performing estimates $\hat{y}$ instead of $\hat{\beta}$.

Q Can't we just use the same methods (*i.e.*, OLS)?

A It depends.

Prediction: What's the goal?

... so?

So we want "nice"-performing estimates $\hat{y}$ instead of $\hat{\beta}$.

Q Can't we just use the same methods (*i.e.*, OLS)?

A It depends. How well does your **linear**-regression model approximate the underlying data? (And how do you plan to select your model?)

Prediction: What's the goal?

... so?

So we want "nice"-performing estimates $\hat{y}$ instead of $\hat{\beta}$.

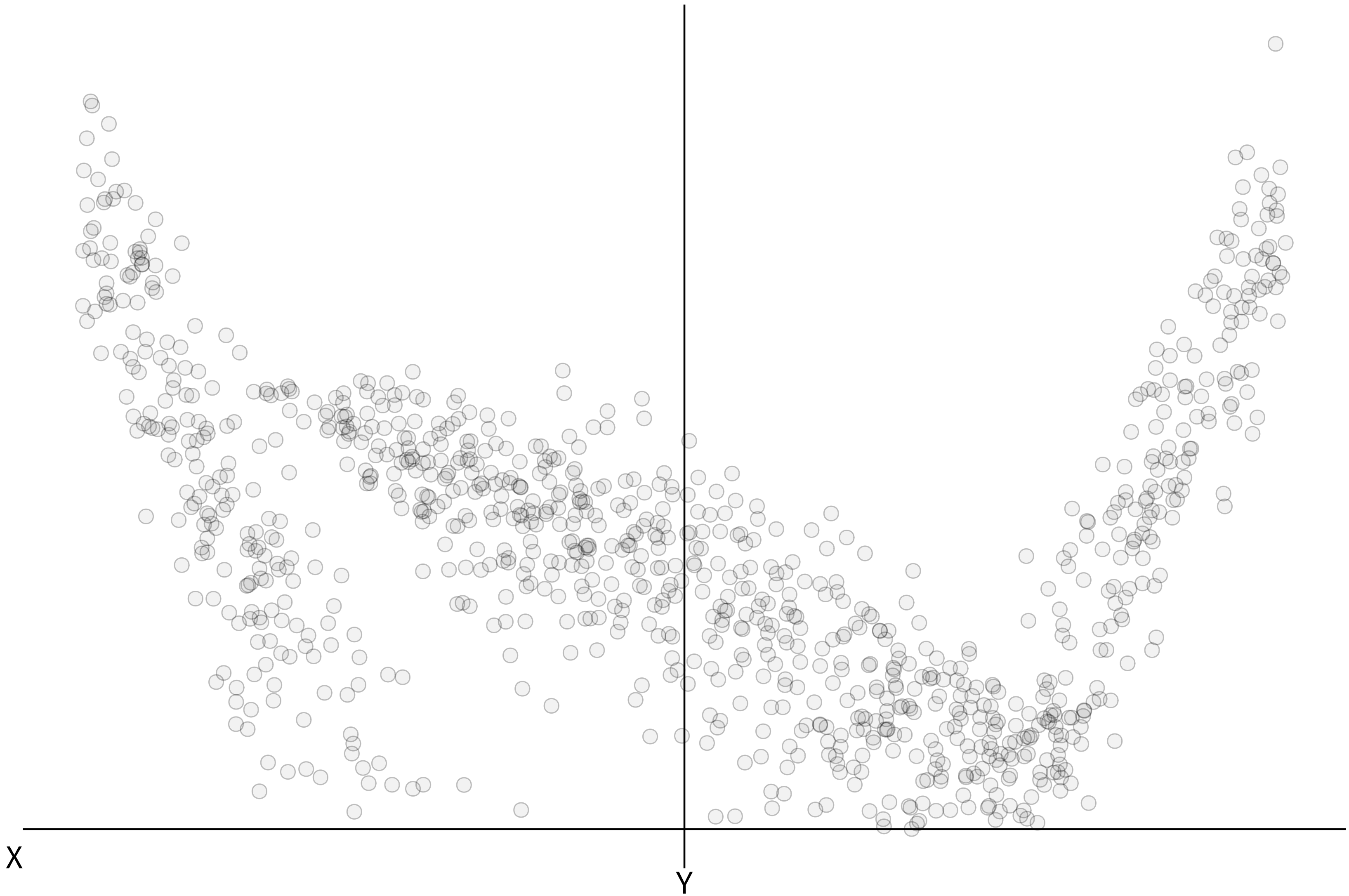
Q Can't we just use the same methods (*i.e.*, OLS)?

A It depends. How well does your **linear**-regression model approximate the underlying data? (And how do you plan to select your model?)

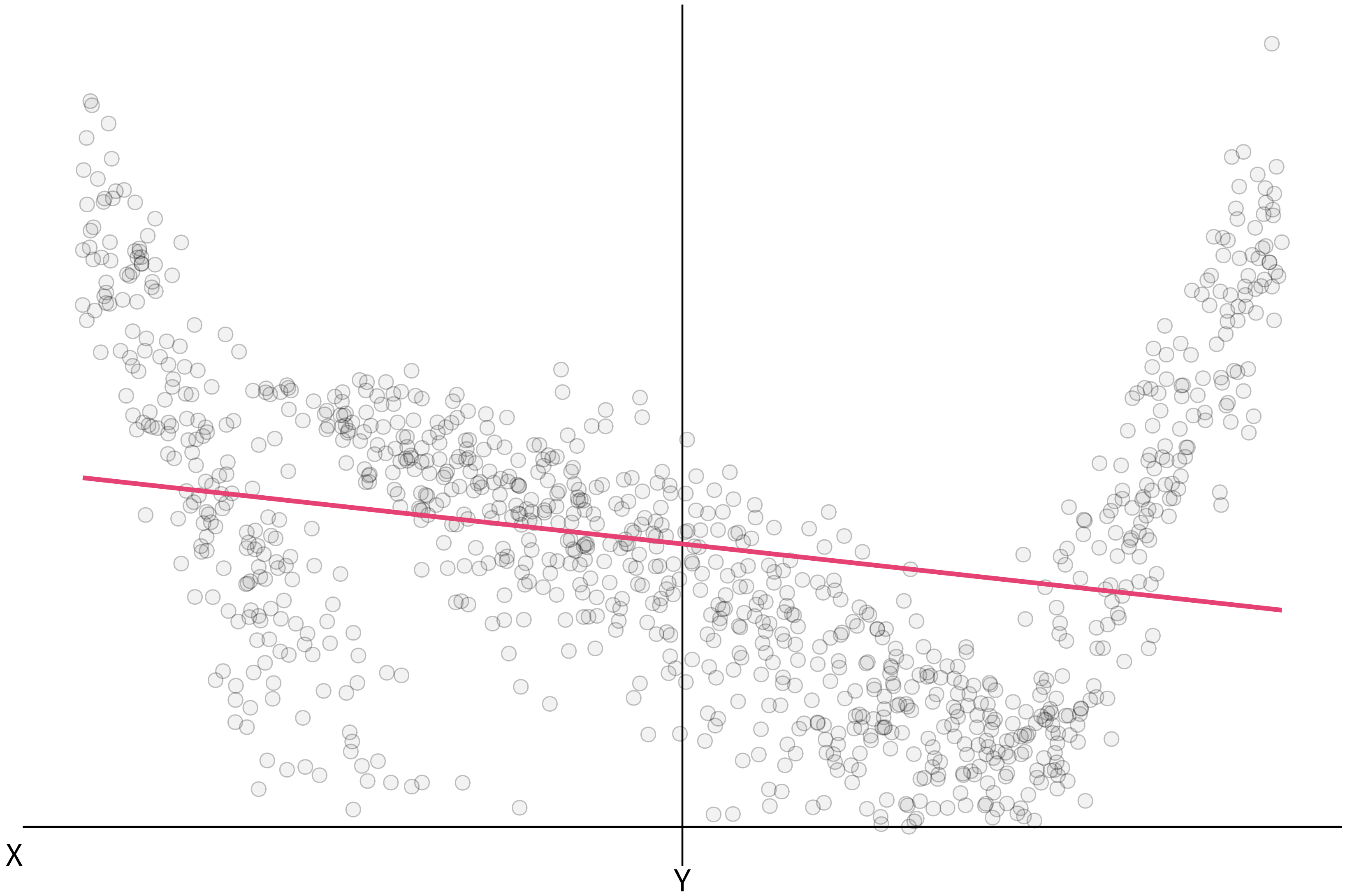
Recall Least-squares regression is a great **linear** estimator.

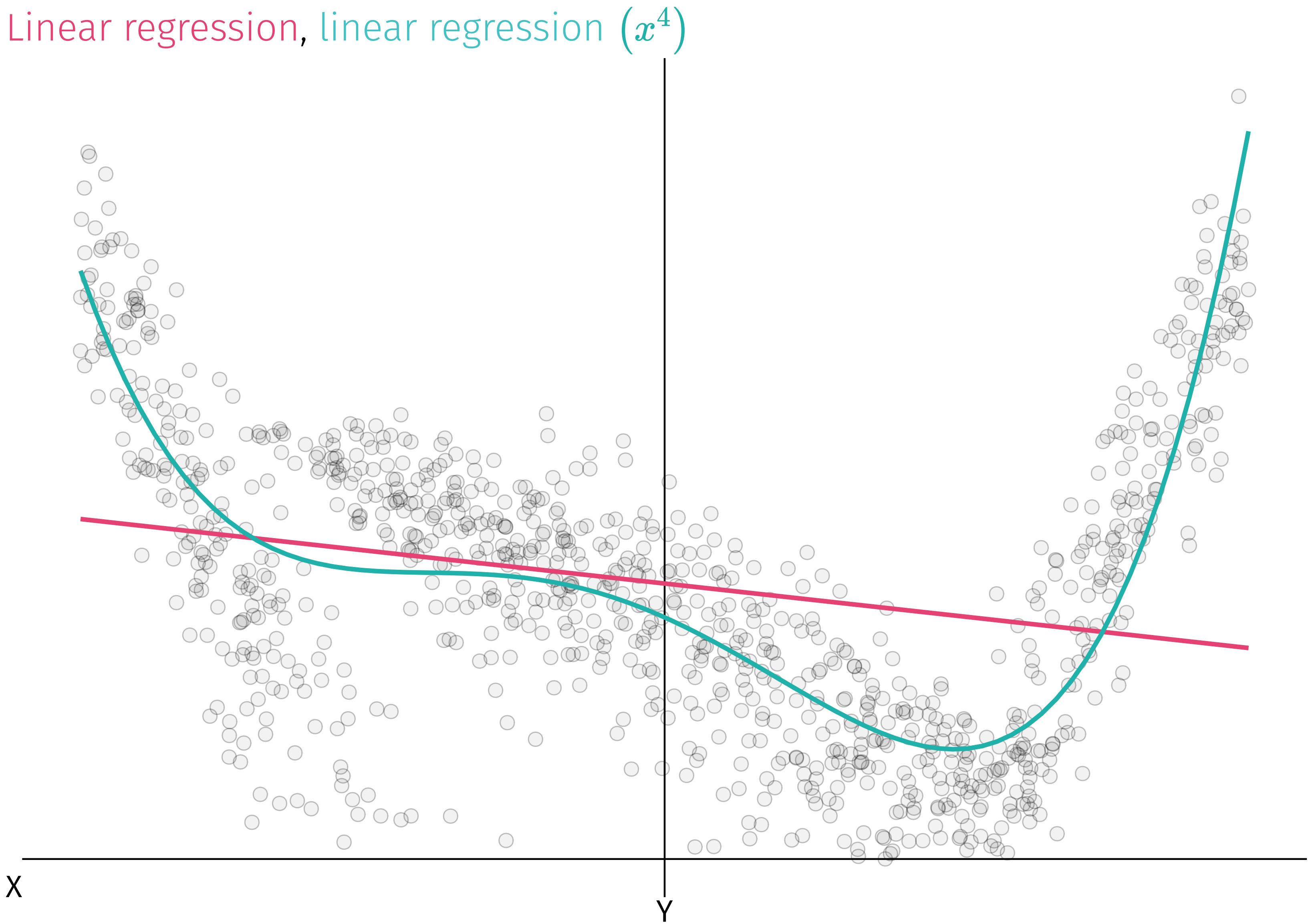
Data data be tricky[†]—as can understanding many relationships.

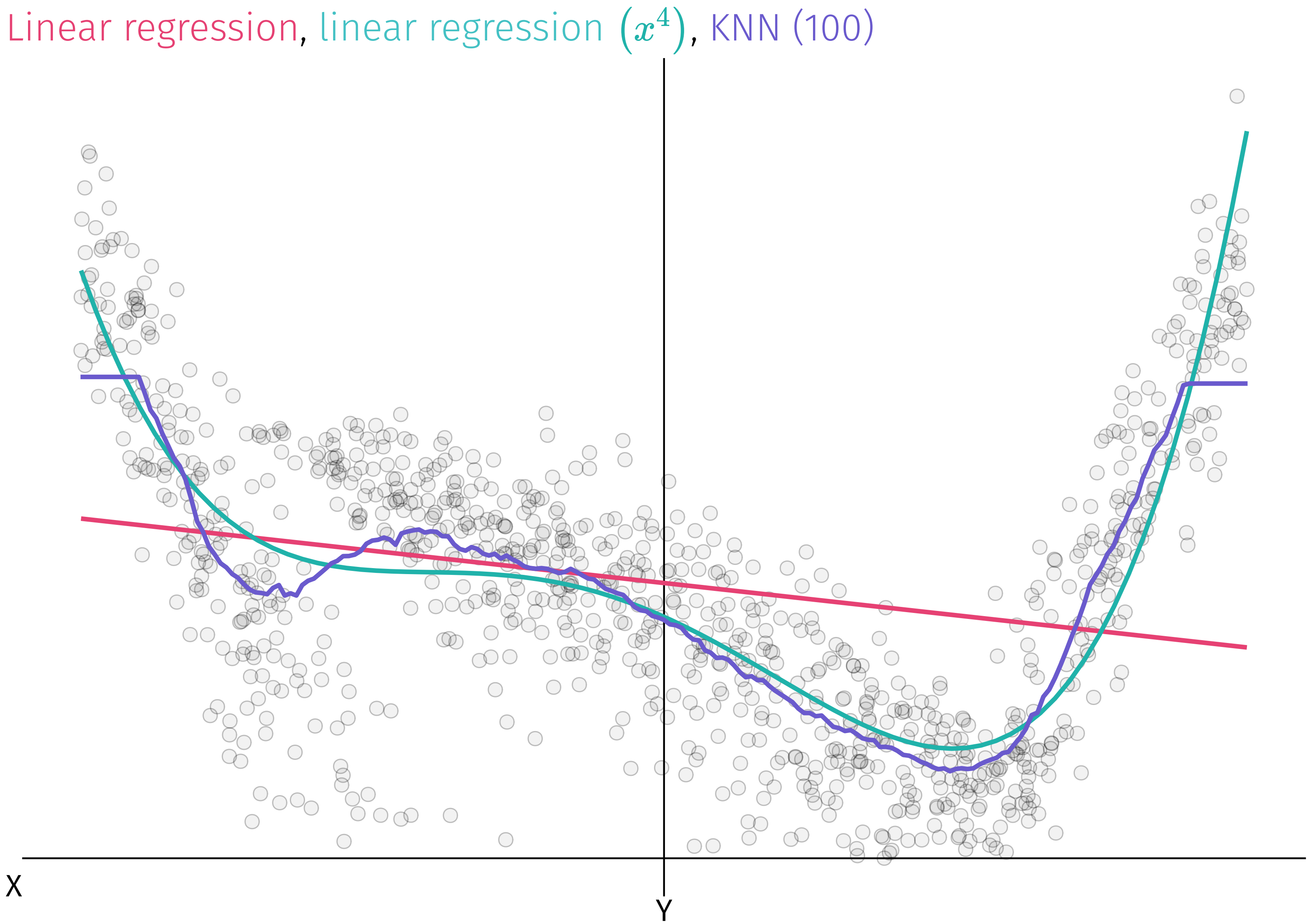
† "Tricky" might mean nonlinear... or many other things...

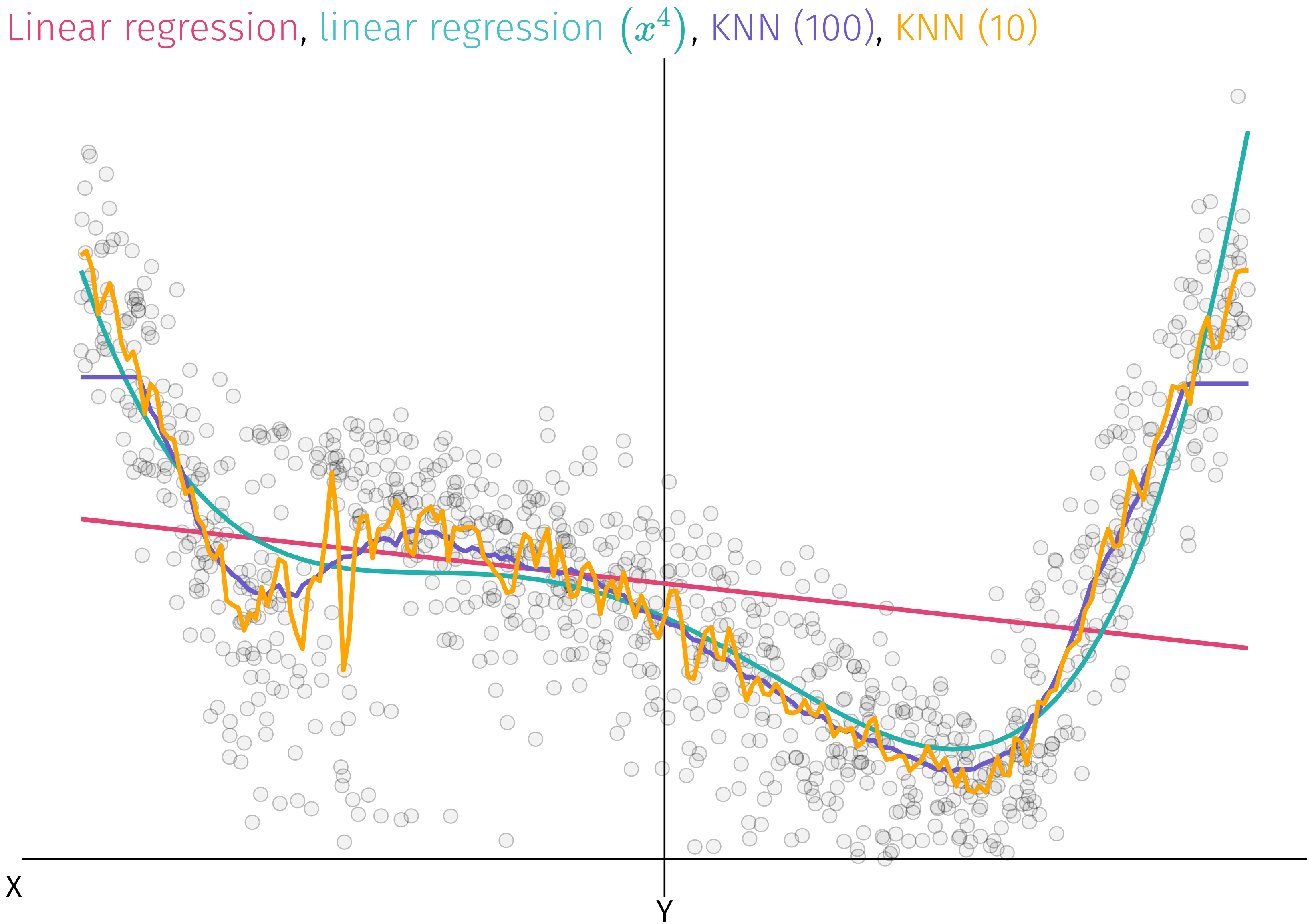

X
Y

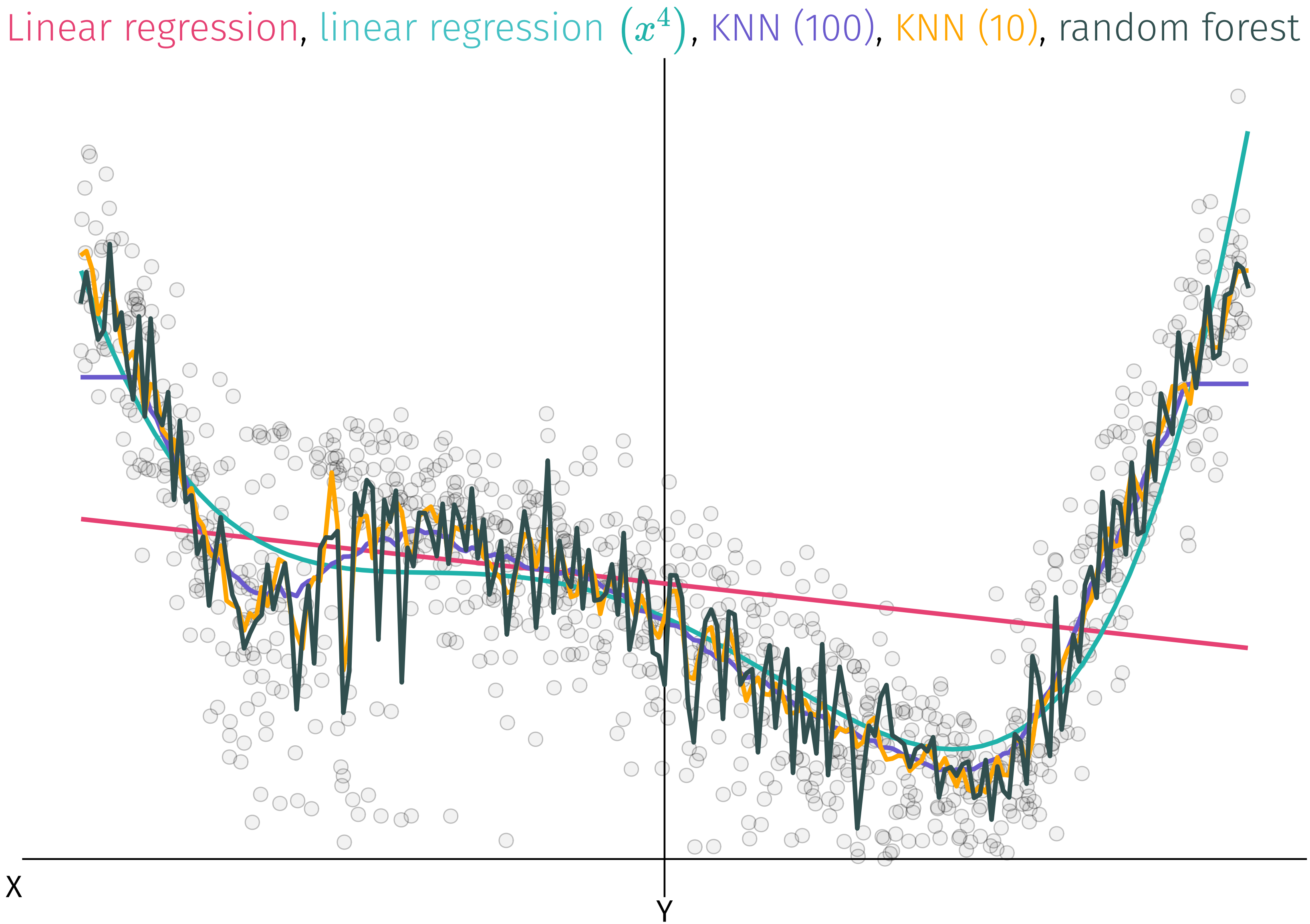
Linear regression




Linear regression, linear regression (x^4)
X
Y


Linear regression, linear regression (x^4), KNN (100)
X
Y


Linear regression, linear regression (x^4), KNN (100), KNN (10)
X
Y



Linear regression, linear regression (x^4), KNN (100), KNN (10), random forest
X
Y

Note That example only had one predictor…

What's the goal?

Tradeoffs

In prediction, we constantly face many tradeoffs, *e.g.*,

- **flexibility** and **parametric structure** (and interpretability)
- performance in **training** and **test** samples
- **variance** and **bias**

What's the goal?

Tradeoffs

In prediction, we constantly face many tradeoffs, *e.g.*,

- **flexibility** and **parametric structure** (and interpretability)
- performance in **training** and **test** samples
- **variance** and **bias**

As your economic training should have predicted, in each setting, we need to **balance the additional benefits and costs** of adjusting these tradeoffs.

What's the goal?

Tradeoffs

In prediction, we constantly face many tradeoffs, *e.g.*,

- **flexibility** and **parametric structure** (and interpretability)
- performance in **training** and **test** samples
- **variance** and **bias**

As your economic training should have predicted, in each setting, we need to **balance the additional benefits and costs** of adjusting these tradeoffs.

Many machine-learning (ML) techniques/algorithms are crafted to optimize with these tradeoffs, but the practitioner (you) still needs to be careful.

What's the goal?

There are many reasons to step outside the world of linear regression...

What's the goal?

There are many reasons to step outside the world of linear regression...

Multi-class classification problems

- Rather than {0,1}, we need to classify y_i into 1 of K classes
- *E.g.*, ER patients: {heart attack, drug overdose, stroke, nothing}

What's the goal?

There are many reasons to step outside the world of linear regression...

Multi-class classification problems

- Rather than {0,1}, we need to classify y_i into 1 of K classes
- *E.g.*, ER patients: {heart attack, drug overdose, stroke, nothing}

Text analysis and **image recognition**

- Comb though sentences (pixels) to glean insights from relationships
- *E.g.*, detect sentiments in tweets or roof-top solar in satellite imagery

What's the goal?

There are many reasons to step outside the world of linear regression...

Multi-class classification problems

- Rather than {0,1}, we need to classify y_i into 1 of K classes
- *E.g.*, ER patients: {heart attack, drug overdose, stroke, nothing}

Text analysis and **image recognition**

- Comb though sentences (pixels) to glean insights from relationships
- *E.g.*, detect sentiments in tweets or roof-top solar in satellite imagery

Unsupervised learning

- You don't know groupings, but you think there are relevant groups
- *E.g.*, classify spatial data into groups

Stanford University (Stanford, CA) researchers have developed a deep-learning algorithm that can evaluate chest X-ray images for signs of disease at a level exceeding practicing radiologists.

Parking Lot Vehicle Detection Using Deep Learning

THE NEW YORKER

A REPORTER AT LARGE OCTOBER 14, 2019 ISSUE

The Next Word

Where will predictive text take us?

Text by John Seabrook

Gender Classifier	Darker Male	Darker Female	Lighter Male	Lighter Female	Largest Gap
Microsoft	94.0%	79.2%	100%	98.3%	20.8%
FACE++	99.3%	65.5%	99.2%	94.0%	33.8%
IBM	88.0%	65.3%	99.7%	92.9%	34.4%

Flexibility is huge, but we still want to avoid overfitting.

Statistical learning

What is it good for?

Statistical learning

What is it good for?

A lot of things.

Statistical learning

What is it good for?

A lot of things. We tend to break statistical-learning into two(-ish) classes:

1. **Supervised learning** builds ("learns") a statistical model for predicting an **output** ($\mathbf{y}$) given a set of **inputs** ($\mathbf{x}_1, \ldots, \mathbf{x}_p$),

Statistical learning

What is it good for?

A lot of things. We tend to break statistical-learning into two(-ish) classes:

1. **Supervised learning** builds ("learns") a statistical model for predicting an **output** ($\mathbf{y}$) given a set of **inputs** ($\mathbf{x}_1, \ldots, \mathbf{x}_p$), *i.e.*, we want to build a model/function f

$$\mathbf{y} = f(\mathbf{x}_1, \ldots, \mathbf{x}_p)$$

   that accurately describes $\mathbf{y}$ given some values of $\mathbf{x}_1, \ldots, x_p$.

Statistical learning

What is it good for?

A lot of things. We tend to break statistical-learning into two(-ish) classes:

1. **Supervised learning** builds ("learns") a statistical model for predicting an **output** ($\mathbf{y}$) given a set of **inputs** ($\mathbf{x}_1, \ldots, \mathbf{x}_p$), *i.e.*, we want to build a model/function f

$$\mathbf{y} = f(\mathbf{x}_1, \ldots, \mathbf{x}_p)$$

   that accurately describes $\mathbf{y}$ given some values of $\mathbf{x}_1, \ldots, x_p$.

2. **Unsupervised learning** learns relationships and structure using only **inputs** ($x_1, \ldots, x_p$) without any *supervising* output

Statistical learning

What is it good for?

A lot of things. We tend to break statistical-learning into two(-ish) classes:

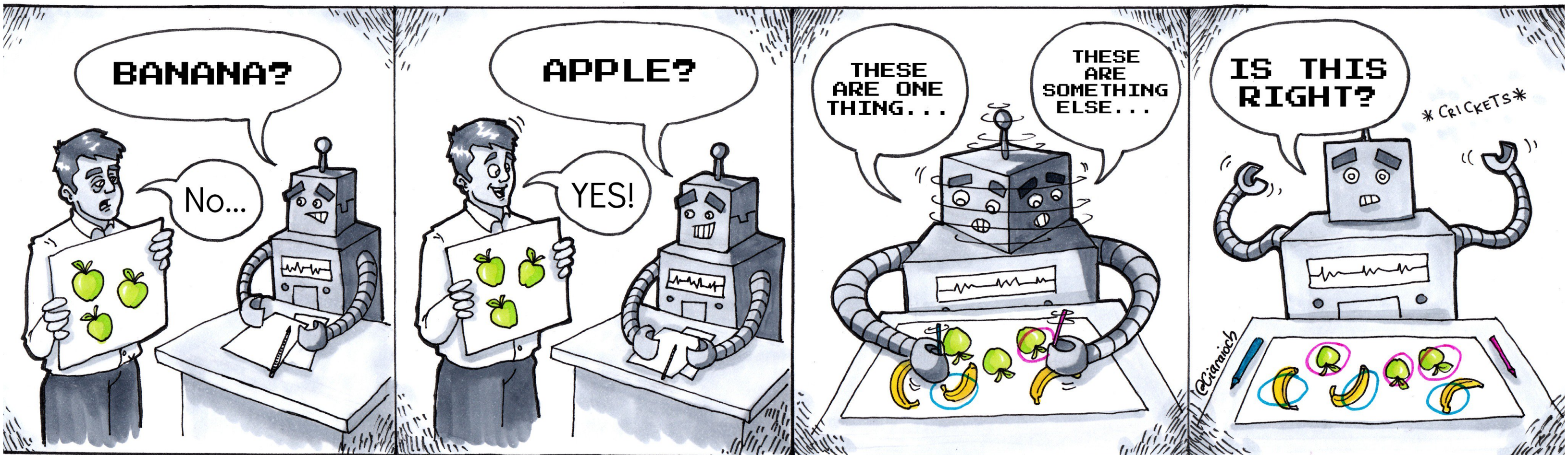
1. **Supervised learning** builds ("learns") a statistical model for predicting an **output** ($\mathbf{y}$) given a set of **inputs** ($\mathbf{x}_1, \ldots, \mathbf{x}_p$), *i.e.*, we want to build a model/function f

$$\mathbf{y} = f(\mathbf{x}_1, \ldots, \mathbf{x}_p)$$

   that accurately describes $\mathbf{y}$ given some values of $\mathbf{x}_1, \ldots, x_p$.

2. **Unsupervised learning** learns relationships and structure using only **inputs** ($x_1, \ldots, x_p$) without any *supervising* output—letting the data "speak for itself."

Semi-supervised learning falls somewhere between these supervised and unsupervised learning—generally applied to supervised tasks when labeled **outputs** are incomplete.



Supervised Learning | **Unsupervised Learning**

Source

Statistical learning

Output

We tend to further break **supervised learning** into two groups, based upon the **output** (the outcome we want to predict):

Statistical learning

Output

We tend to further break **supervised learning** into two groups, based upon the **output** (the outcome we want to predict):

1. **Classification tasks** for which the values of $\mathbf{y}$ are discrete categories
   E.g., race, sex, loan default, hazard, disease, flight status

2. **Regression tasks** in which $\mathbf{y}$ takes on continuous, numeric values.
   E.g., price, arrival time, number of emails, temperature

Note$_1$ The use of *regression* differs from our use of *linear regression*.

Statistical learning

Output

We tend to further break **supervised learning** into two groups, based upon the **output** (the outcome we want to predict):

1. **Classification tasks** for which the values of $\mathbf{y}$ are discrete categories
   E.g., race, sex, loan default, hazard, disease, flight status

2. **Regression tasks** in which $\mathbf{y}$ takes on continuous, numeric values.
   E.g., price, arrival time, number of emails, temperature

Note$_1$ The use of *regression* differs from our use of *linear regression*.

Note$_2$ Don't get tricked: Not all numbers represent continuous, numerical values—*e.g.*, zip codes, industry codes, social security numbers.†

† **Q** Where would you put responses to 5-item Likert scales?

Statistical learning

The goal

As defined before, we want to *learn* a model to understand our data.

The goal

As defined before, we want to *learn* a model to understand our data.

1. Take our (numeric) output $\mathbf{y}$.
2. Imagine there is a function f that takes inputs $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_p$ and maps them, plus a random, mean-zero error term ε, to the output.

$$\mathbf{y} = f(\mathbf{X}) + \varepsilon$$

Learning from $\hat{f}$

There are two main reasons we want to learn about f

1. ***Causal* inference settings** How do changes in $\mathbf{X}$ affect $\mathbf{y}$?
   What we've done all quarter.

Learning from $\hat{f}$

There are two main reasons we want to learn about f

1. ***Causal* inference settings** How do changes in $\mathbf{X}$ affect $\mathbf{y}$?
   What we've done all quarter.

2. **Prediction problems** Predict $\mathbf{y}$ using our estimated f, *i.e.*,

$$\hat{\mathbf{y}} = \hat{f}(\mathbf{X})$$

   our *black-box setting* where we care less about f than $\hat{\mathbf{y}}$.[†]

† You shouldn't actually treat your prediction methods as total black boxes.

Statistical learning

Learning from $\hat{f}$

There are two main reasons we want to learn about f

1. ***Causal* inference settings** How do changes in $\mathbf{X}$ affect $\mathbf{y}$?
   What we've done all quarter.

2. **Prediction problems** Predict $\mathbf{y}$ using our estimated f, *i.e.*,

$$\hat{\mathbf{y}} = \hat{f}(\mathbf{X})$$

   our *black-box setting* where we care less about f than $\hat{\mathbf{y}}$.[†]

Similarly, in causal-inference settings, we don't particulary care about $\hat{\mathbf{y}}$.

† You shouldn't actually treat your prediction methods as total black boxes.

Prediction errors

As tends to be the case in life, you will make errors in predicting $\mathbf{y}$.

The accuracy of $\hat{\mathbf{y}}$ depends upon **two errors**:

Prediction errors

As tends to be the case in life, you will make errors in predicting $\mathbf{y}$.

The accuracy of $\hat{\mathbf{y}}$ depends upon **two errors**:

1. **Reducible error** The error due to $\hat{f}$ imperfectly estimating f.
   Reducible in the sense that we could improve $\hat{f}$.

Prediction errors

As tends to be the case in life, you will make errors in predicting $\mathbf{y}$.

The accuracy of $\hat{\mathbf{y}}$ depends upon **two errors**:

1. **Reducible error** The error due to $\hat{f}$ imperfectly estimating f.
   Reducible in the sense that we could improve $\hat{f}$.

2. **Irreducible error** The error component that is outside of the model f.
   Irreducible because we defined an error term ε unexplained by f.

Prediction errors

As tends to be the case in life, you will make errors in predicting $\mathbf{y}$.

The accuracy of $\hat{\mathbf{y}}$ depends upon **two errors**:

1. **Reducible error** The error due to $\hat{f}$ imperfectly estimating f.
   Reducible in the sense that we could improve $\hat{f}$.

2. **Irreducible error** The error component that is outside of the model f.
   Irreducible because we defined an error term ε unexplained by f.

Note As its name implies, you can't get rid of *irreducible* error—but we can try to get rid of *reducible* errors.

Statistical learning

Prediction errors

Why we're stuck with *irreducible* error

$$E\left[\left\{\mathbf{y} - \hat{\mathbf{y}}\right\}^2\right] = E\left[\left\{f(\mathbf{X}) + \varepsilon + \hat{f}(\mathbf{X})\right\}^2\right]$$
$$= \underbrace{\left[f(\mathbf{X}) - \hat{f}(\mathbf{X})\right]^2}_{\text{Reducible}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{Irreducible}}$$

In less math:

- If ε exists, then $\mathbf{X}$ cannot perfectly explain $\mathbf{y}$.
- So even if $\hat{f} = f$, we still have irreducible error.

Statistical learning

Prediction errors

Why we're stuck with *irreducible* error

$$E\left[\{\mathbf{y} - \hat{\mathbf{y}}\}^2\right] = E\left[\left\{f(\mathbf{X}) + \varepsilon + \hat{f}(\mathbf{X})\right\}^2\right]$$
$$= \underbrace{\left[f(\mathbf{X}) - \hat{f}(\mathbf{X})\right]^2}_{\text{Reducible}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{Irreducible}}$$

In less math:

- If ε exists, then $\mathbf{X}$ cannot perfectly explain $\mathbf{y}$.
- So even if $\hat{f} = f$, we still have irreducible error.

Thus, to form our **best predictors**, we will **minimize reducible error**.

Model accuracy

MSE

Mean squared error (MSE) is the most common† way to measure model performance in a regression setting.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left[y_i - \hat{f}\left(x_i \right) \right]^2$$

Recall: $y_i - \hat{f}\left(x_i \right) = y_i - \hat{y}_i$ is our prediction error.

† *Most common* does not mean best—it just means lots of people use it.

MSE

Mean squared error (MSE) is the most common[†] way to measure model performance in a regression setting.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left[y_i - \hat{f}(x_i) \right]^2$$

Recall: $y_i - \hat{f}(x_i) = y_i - \hat{y}_i$ is our prediction error.

Two notes about MSE

1. MSE will be (relatively) very small when **prediction error** is nearly zero.
2. MSE **penalizes** big errors more than little errors (the squared part).

† *Most common* does not mean best—it just means lots of people use it.

Model accuracy

Training or testing?

Low MSE (accurate performance) on the data that trained the model isn't actually impressive—maybe the model is just overfitting our data.[†]

What we want: How well does the model perform **on data it has never seen**?

† Recall the kNN performance for k=1.

Model accuracy

Training or testing?

Low MSE (accurate performance) on the data that trained the model isn't actually impressive—maybe the model is just overfitting our data.[†]

What we want: How well does the model perform **on data it has never seen**?

This introduces an important distinction:

1. **Training data**: The observations (y_i, x_i) used to **train** our model $\hat{f}$.
2. **Testing data**: The observations (y_0, x_0) that our model has yet to see—and which we can use to evaluate the performance of $\hat{f}$.

† Recall the kNN performance for k=1.

Model accuracy

Training or testing?

Low MSE (accurate performance) on the data that trained the model isn't actually impressive—maybe the model is just overfitting our data.[†]

What we want: How well does the model perform **on data it has never seen**?

This introduces an important distinction:

1. **Training data**: The observations (y_i, x_i) used to **train** our model $\hat{f}$.
2. **Testing data**: The observations (y_0, x_0) that our model has yet to see—and which we can use to evaluate the performance of $\hat{f}$.

Real goal: Low test-sample MSE (not the training MSE from before).

† Recall the kNN performance for k=1.

Model accuracy

Regression and loss

For **regression settings**, the loss is our prediction's distance from truth, *i.e.*,

$$\text{error}_i = y_i - \hat{y}_i \qquad \text{loss}_i = \left|y_i - \hat{y}_i\right| = \left|\text{error}_i\right|$$

Depending upon our ultimate goal, we choose **loss/objective functions**.

$$\text{L1 loss} = \sum_i \left|y_i - \hat{y}_i\right| \qquad \text{MAE} = \frac{1}{n}\sum_i \left|y_i - \hat{y}_i\right|$$

$$\text{L2 loss} = \sum_i \left(y_i - \hat{y}_i\right)^2 \qquad \text{MSE} = \frac{1}{n}\sum_i \left(y_i - \hat{y}_i\right)^2$$

Whatever we're using, we care about **test performance** (*e.g.*, test MSE), rather than training performance.

Model accuracy

Classification

For **classification problems**, we often use the **test error rate**.

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(y_i \neq \hat{y}_i)$$

The **Bayes classifier**

1. predicts class j when $\mathbf{Pr}\big(y_0 = j|\mathbf{X} = \mathbf{x}_0\big)$ exceeds all other classes.
2. produces the **Bayes decision boundary**—the decision boundary with the lowest test error rate.
3. is unknown: we must predict $\mathbf{Pr}\big(y_0 = j|\mathbf{X} = \mathbf{x}_0\big)$.

Flexibility

The bias-variance tradeoff

Finding the optimal level of flexibility highlights the **bias-variance tradeoff**.

Bias The error that comes from inaccurately estimating f.

- More flexible models are better equipped to recover complex relationships (f), reducing bias. (Real life is seldom linear.)
- Simpler (less flexible) models typically increase bias.

Variance The amount $\hat{f}$ would change with a different **training sample**

- If new **training sets** drastically change $\hat{f}$, then we have a lot of uncertainty about f (and, in general, $\hat{f} \not\approx f$).
- More flexible models generally add variance to f.

Flexibility

The bias-variance tradeoff

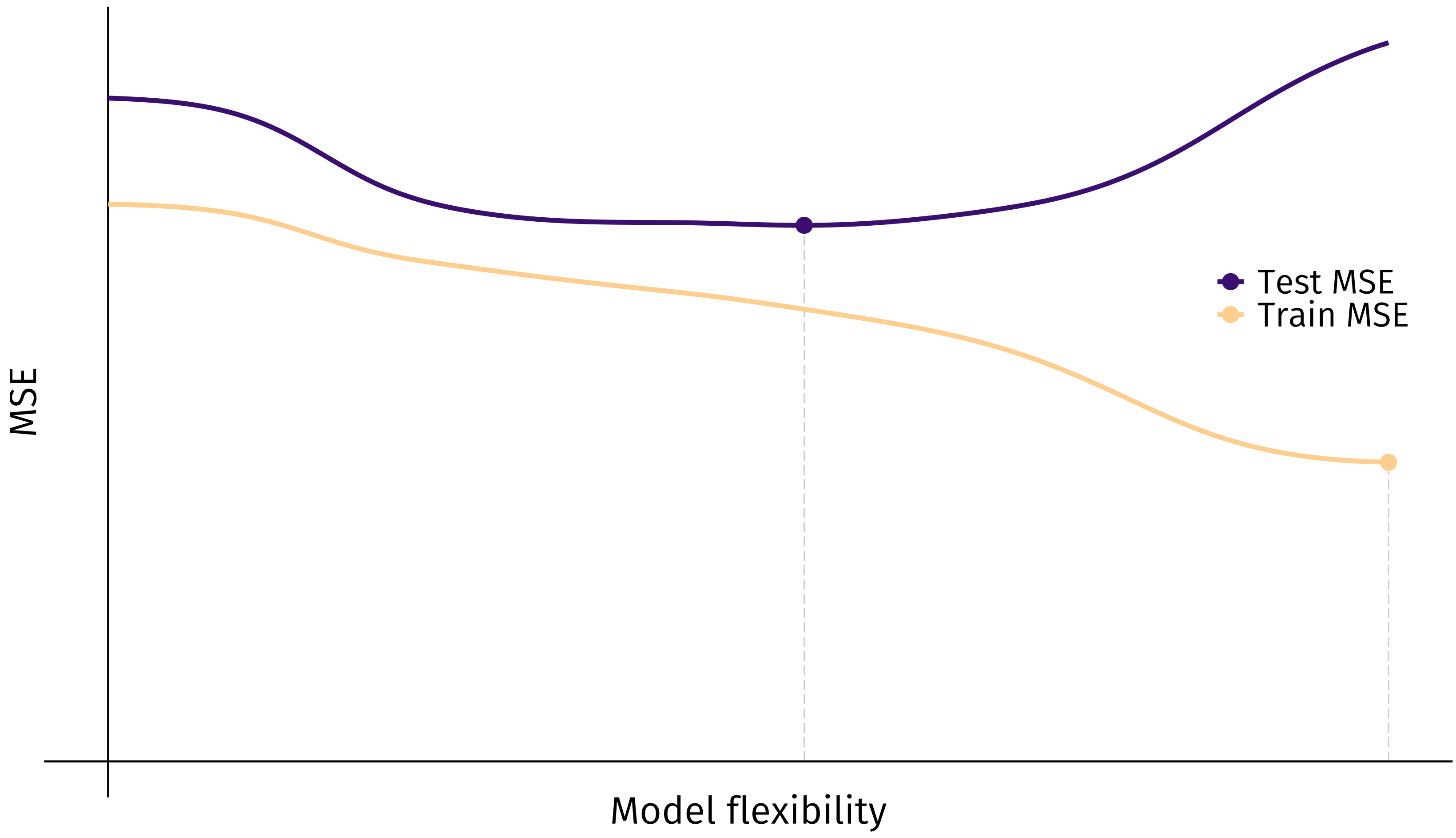
The expected value[†] of the **test MSE** can be written

$$E\left[\left(\mathbf{y_0} - \hat{f}\left(\mathbf{X}_0\right)\right)^2\right] = \underbrace{\text{Var}\left(\hat{f}\left(\mathbf{X}_0\right)\right)}_{\text{Variance}} + \underbrace{\left[\text{Bias}\left(\hat{f}\left(\mathbf{X}_0\right)\right)\right]^2}_{\text{Bias}} + \underbrace{\text{Var}(\varepsilon)}_{\text{Irr. error}}$$

The tradeoff in terms of model flexibility

- Increasing flexibility *from total inflexibility* generally **reduces bias more** than it increases variance (reducing test MSE).
- At some point, the marginal benefits of flexibility **equal** marginal costs.
- Past this point (optimal flexibility), we **increase variance more** than we reduce bias (increasing test MSE).

U-shaped test MSE with respect to model flexibility (KNN here). Increases in variance eventually overcome reductions in (squared) bias.



Resampling refresher

Resampling methods help understand uncertainty in statistical modeling.

The process behind the magic of resampling methods:

1. **Repeatedly draw samples** from the **training data**.
2. **Fit your model**(s) on each random sample.
3. **Compare** model performance (or estimates) **across samples**.
4. Infer the **variability/uncertainty in your model** from (3).

Resampling

Hold out

Recall: We want to find the model that **minimizes out-of-sample test error**.

If we have a large test dataset, we can use it (once).

Q_1 What if we don't have a test set?

Q_2 What if we need to select and train a model?

Q_3 How can we avoid overfitting our training[†] data during model selection?

† Also relevant for *testing* data.

Hold out

Recall: We want to find the model that **minimizes out-of-sample test error**.

If we have a large test dataset, we can use it (once).

$\mathbf{Q}_1$ What if we don't have a test set?

$\mathbf{Q}_2$ What if we need to select and train a model?

$\mathbf{Q}_3$ How can we avoid overfitting our training[†] data during model selection?

$\mathbf{A}_{1,2,3}$ **Hold-out methods** (*e.g.*, cross validation) use training data to estimate test performance—**holding out** a mini "test" sample of the training data that we use to estimate the test error.

† Also relevant for *testing* data.

Hold-out methods

Option 1: The *validation set* approach

To estimate the **test error**, we can *hold out* a subset of our **training data** and then **validate** (evaluate) our model on this held out **validation set**.

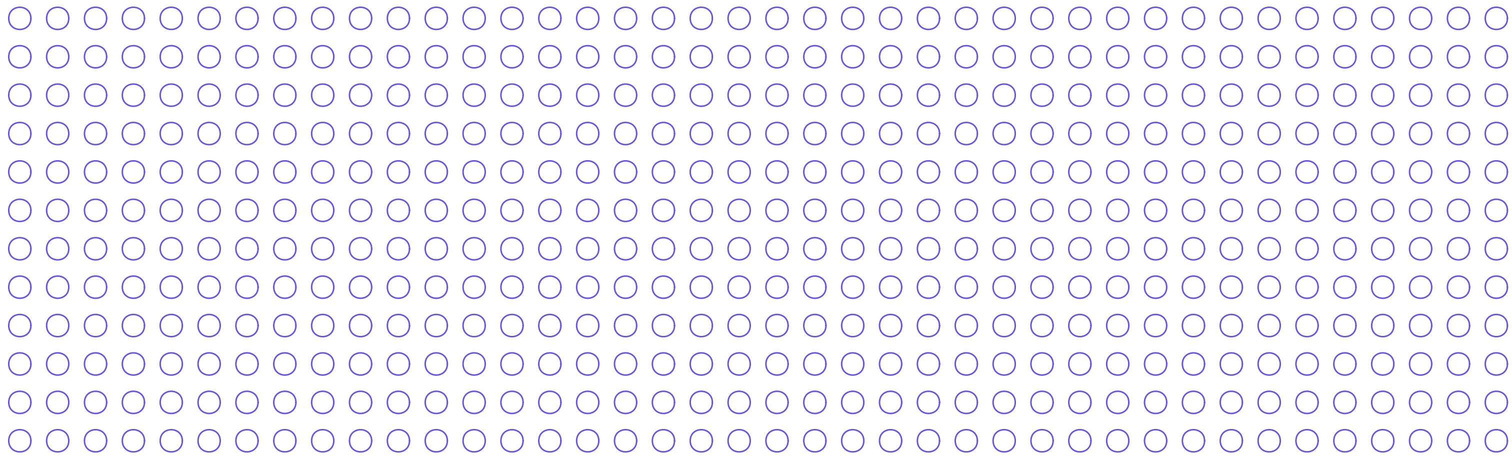
- The **validation error rate** estimates the **test error rate**
- The model only "sees" the non-validation subset of the **training data**.

Hold-out methods

Option 1: The *validation set* approach

To estimate the **test error**, we can *hold out* a subset of our **training data** and then **validate** (evaluate) our model on this held out **validation set**.

- The **validation error rate** estimates the **test error rate**
- The model only "sees" the non-validation subset of the **training data**.



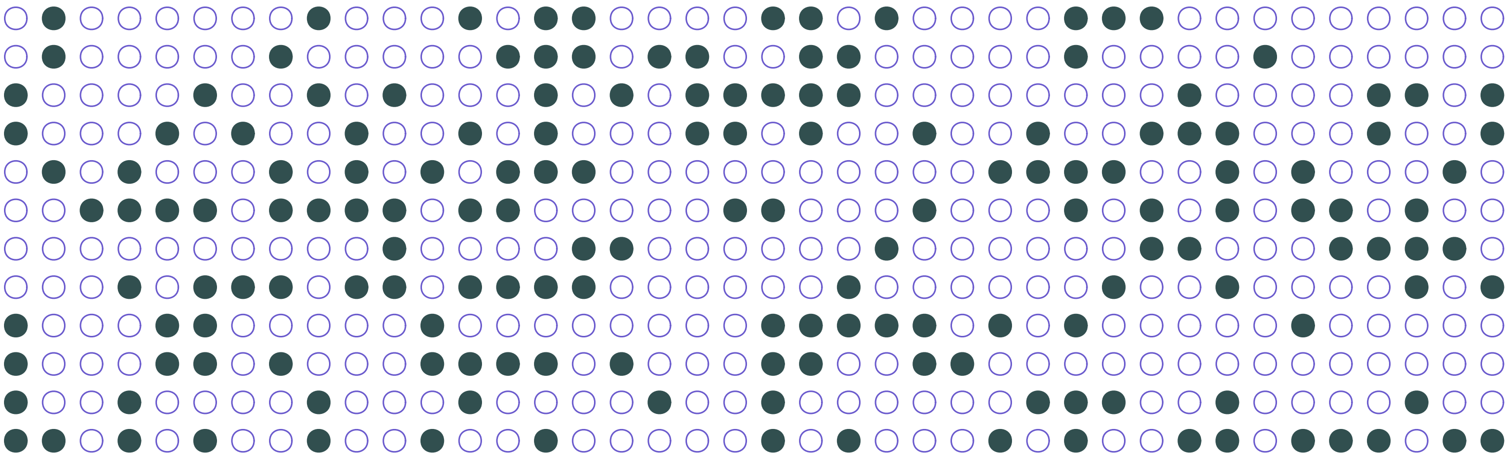
Initial training set

Hold-out methods

Option 1: The *validation set* approach

To estimate the **test error**, we can *hold out* a subset of our **training data** and then **validate** (evaluate) our model on this held out **validation set**.

- The **validation error rate** estimates the **test error rate**
- The model only "sees" the non-validation subset of the **training data**.



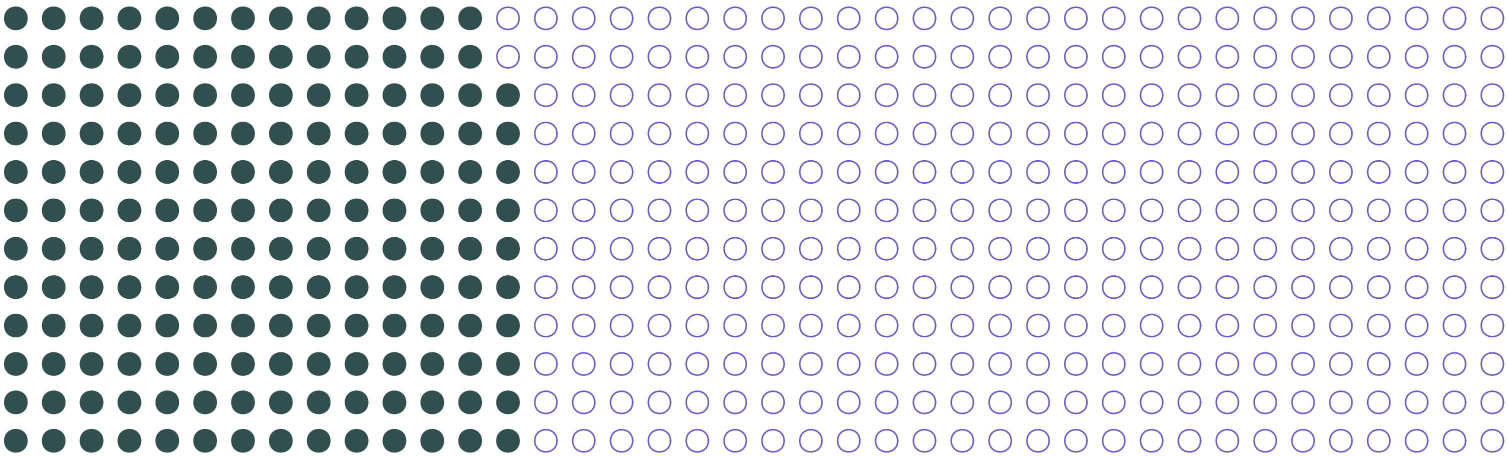
Validation (sub)set **Training set:** Model training

Hold-out methods

Option 1: The *validation set* approach

To estimate the **test error**, we can *hold out* a subset of our **training data** and then **validate** (evaluate) our model on this held out **validation set**.

- The **validation error rate** estimates the **test error rate**
- The model only "sees" the non-validation subset of the **training data**.



Validation (sub)set

Training set: Model training

Hold-out methods

Option 1: The *validation set* approach

Example We could use the validation-set approach to help select the degree of a polynomial for a linear-regression model.

Hold-out methods

Option 1: The *validation set* approach

Example We could use the validation-set approach to help select the degree of a polynomial for a linear-regression model.

The goal of the validation set is to ***estimate* out-of-sample (test) error.**

Q So what?

Hold-out methods

Option 1: The *validation set* approach

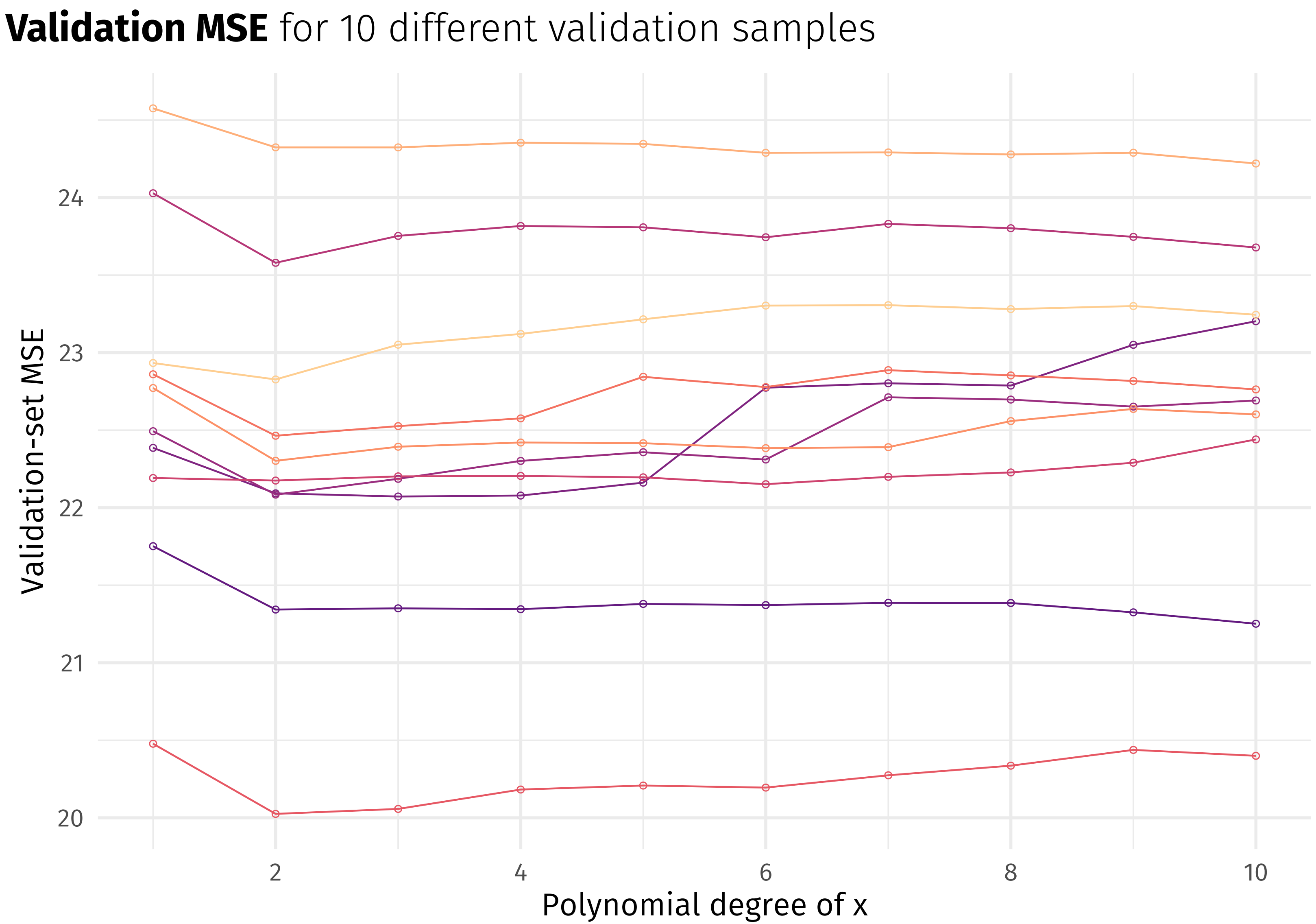
Example We could use the validation-set approach to help select the degree of a polynomial for a linear-regression model.

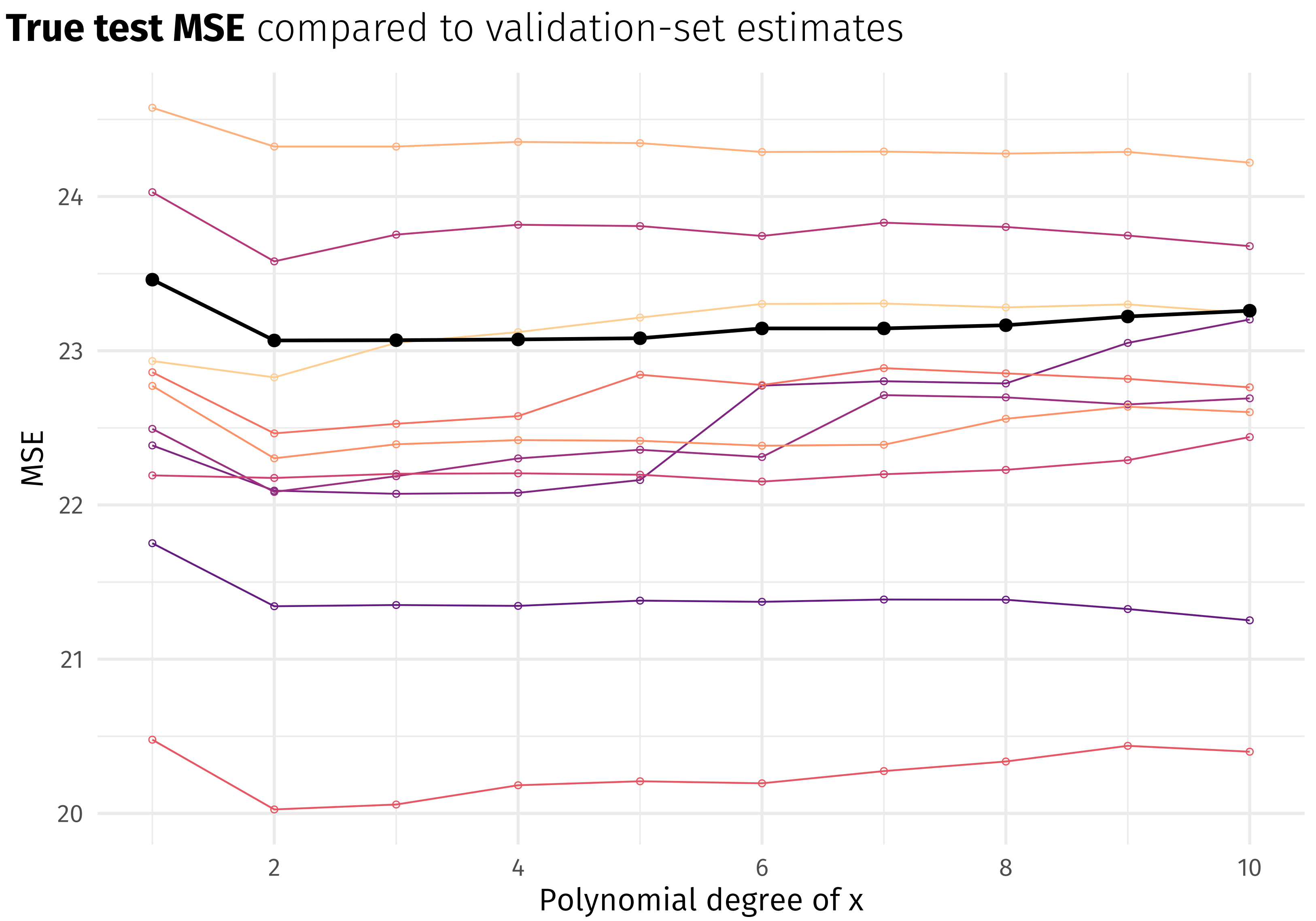
The goal of the validation set is to ***estimate* out-of-sample (test) error.**

Q So what?

- Estimates come with **uncertainty**—varying from sample to sample.
- Variability (standard errors) is larger with **smaller samples**.

Problem This estimated error is often based upon a fairly small sample (<30% of our training data). So its variance can be large.


Validation MSE for 10 different validation samples
Validation-set MSE
24
23
22
21
20
2
4
6
8
10
Polynomial degree of x



True test MSE compared to validation-set estimates
24
23
22
21
20
MSE
2
4
6
8
10
Polynomial degree of x

Hold-out methods

Option 1: The *validation set* approach

Put differently: The validation-set approach has (≥) two major drawbacks:

1. **High variability** Which observations are included in the validation set can greatly affect the validation MSE.
2. **Inefficiency in training our model** We're essentially throwing away the validation data when training the model—"wasting" observations.

Hold-out methods

Option 1: The *validation set* approach

Put differently: The validation-set approach has (≥) two major drawbacks:

1. **High variability** Which observations are included in the validation set can greatly affect the validation MSE.
2. **Inefficiency in training our model** We're essentially throwing away the validation data when training the model—"wasting" observations.

(2) $\implies$ validation MSE may overestimate test MSE.

Even if the validation-set approach provides an unbiased estimator for test error, it is likely a pretty noisy estimator.

Hold-out methods

Option 2: Leave-one-out cross validation

Cross validation solves the validation-set method's main problems.

- Use more (= all) of the data for training (lower variability; less bias).
- Still maintains separation between training and validation subsets.

Hold-out methods

Option 2: Leave-one-out cross validation

Cross validation solves the validation-set method's main problems.

- Use more (= all) of the data for training (lower variability; less bias).
- Still maintains separation between training and validation subsets.

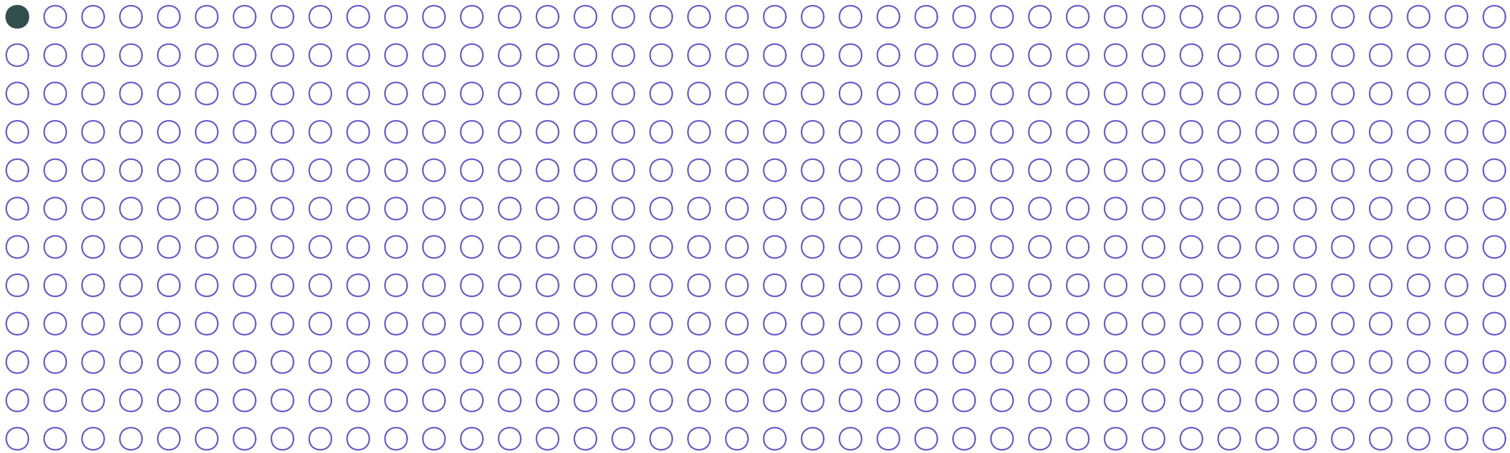
Leave-one-out cross validation (LOOCV) is perhaps the cross-validation method most similar to the validation-set approach.

- Your validation set is exactly one observation.
- *New* You repeat the validation exercise for every observation.
- *New* Estimate MSE as the mean across all observations.

Hold-out methods

Option 2: Leave-one-out cross validation

Each observation takes a turn as the **validation set**,
while the other n-1 observations get to **train the model**.

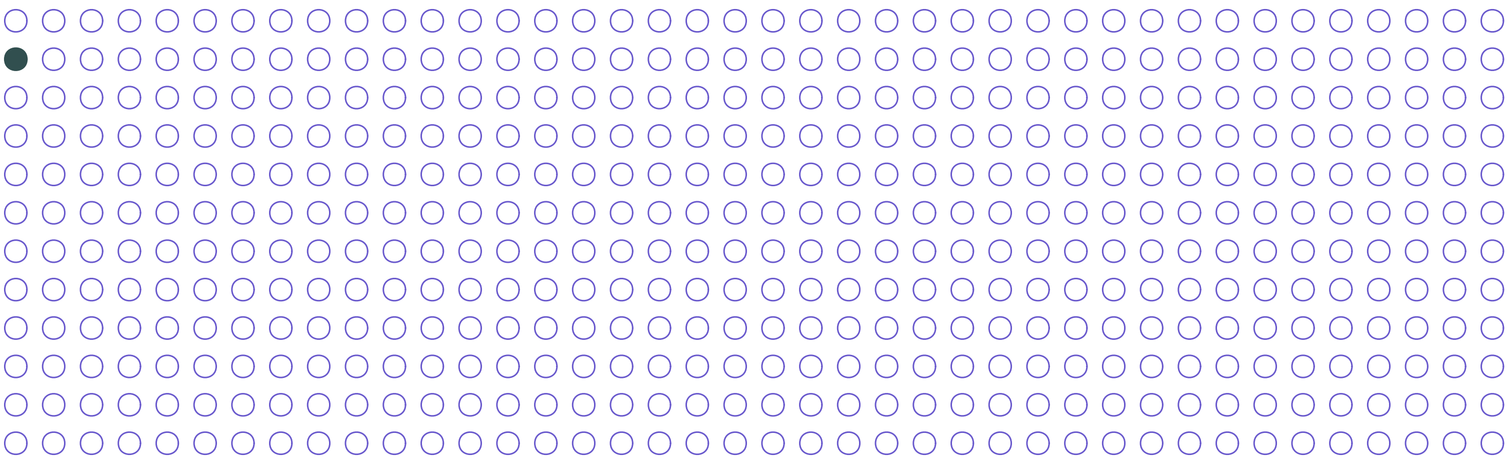


Observation 1's turn for validation produces MSE_1.

Hold-out methods

Option 2: Leave-one-out cross validation

Each observation takes a turn as the **validation set**,
while the other n-1 observations get to **train the model**.

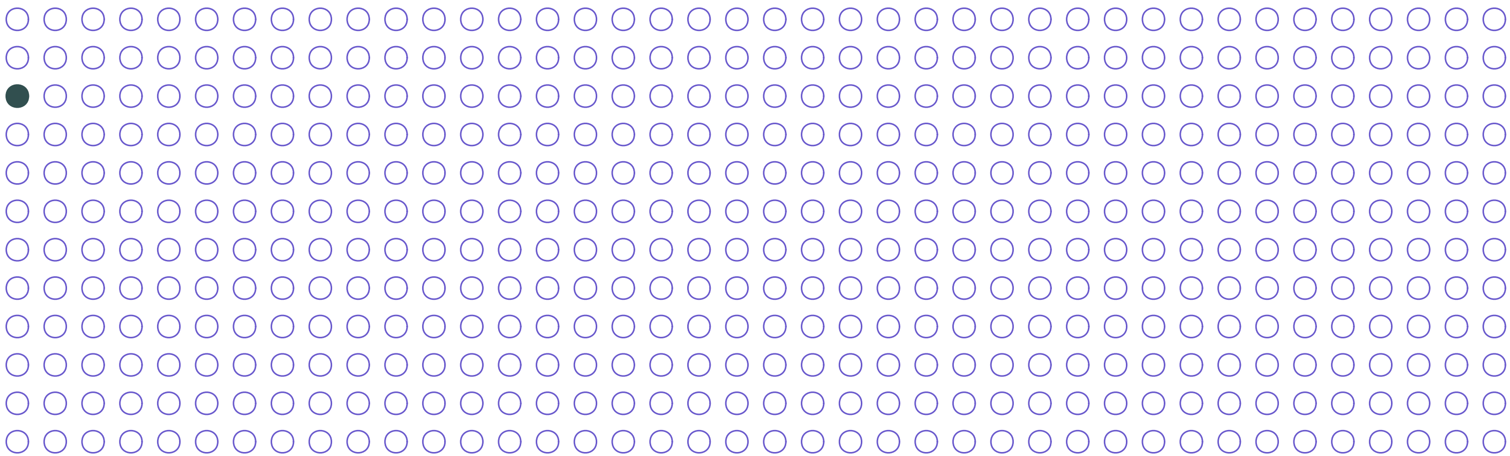


Observation 2's turn for validation produces MSE_2.

Hold-out methods

Option 2: Leave-one-out cross validation

Each observation takes a turn as the **validation set**,
while the other n-1 observations get to **train the model**.

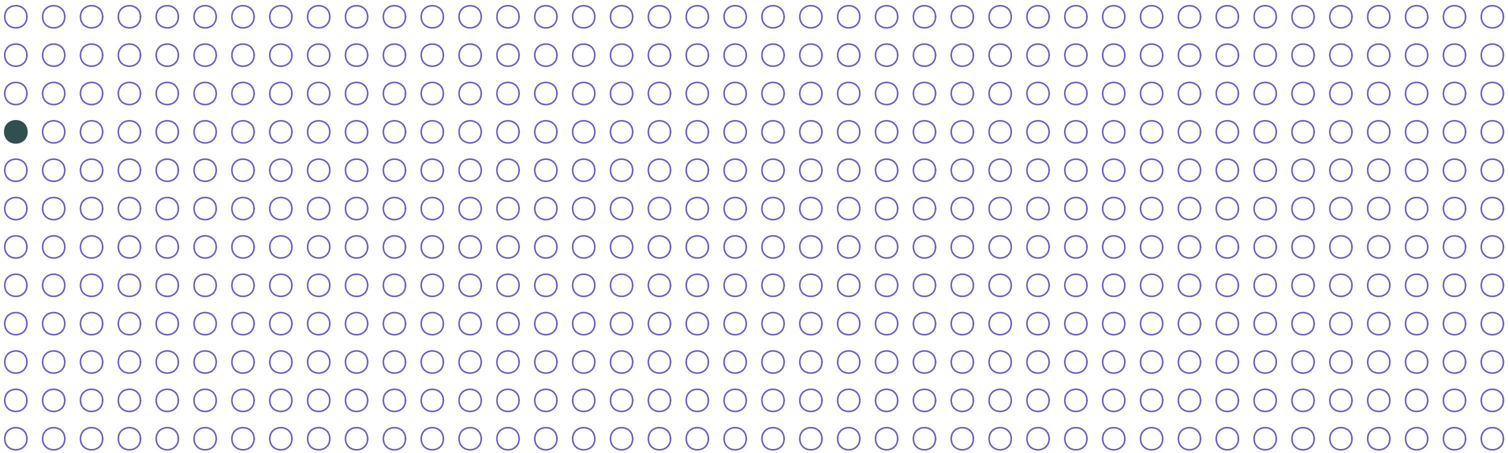


Observation 3's turn for validation produces MSE_3.

Hold-out methods

Option 2: Leave-one-out cross validation

Each observation takes a turn as the **validation set**,
while the other n-1 observations get to **train the model**.

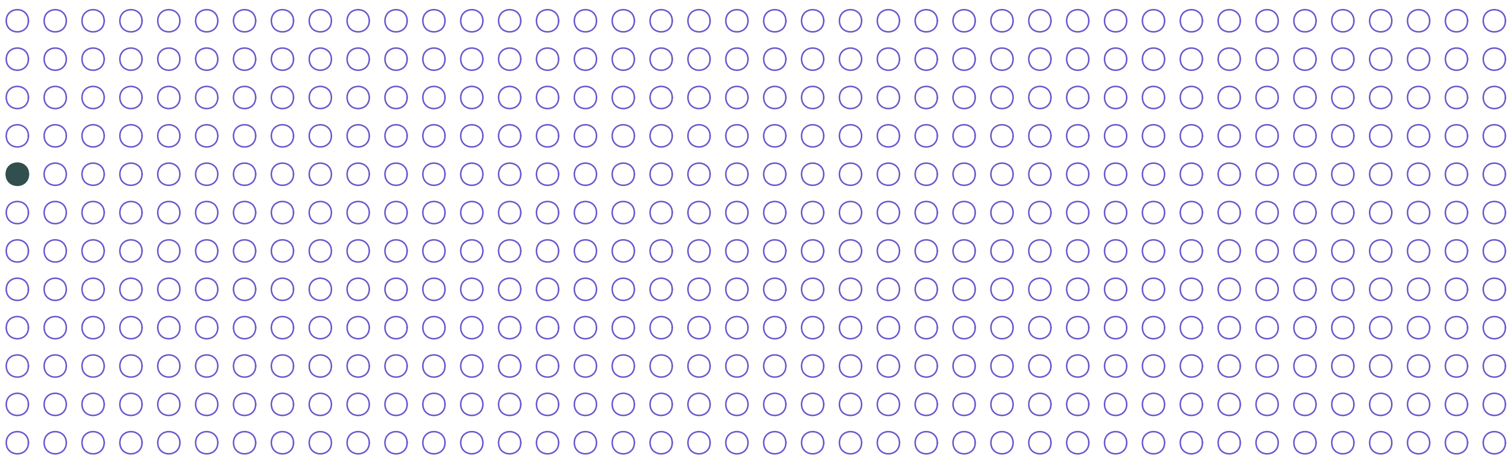


Observation 4's turn for validation produces MSE_4.

Hold-out methods

Option 2: Leave-one-out cross validation

Each observation takes a turn as the **validation set**,
while the other n-1 observations get to **train the model**.

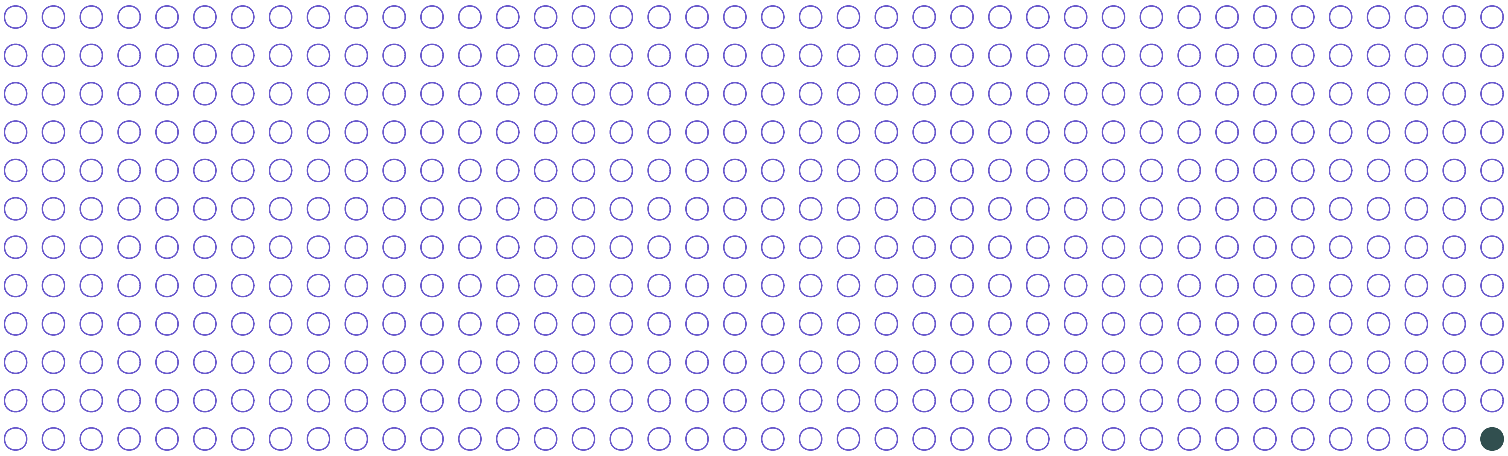


Observation 5's turn for validation produces MSE_5.

Hold-out methods

Option 2: Leave-one-out cross validation

Each observation takes a turn as the **validation set**,
while the other n-1 observations get to **train the model**.



Observation n's turn for validation produces MSE_n.

Hold-out methods

Option 2: Leave-one-out cross validation

Because **LOOCV uses n-1 observations** to train the model,[†] MSE_i (validation MSE from observation i) is approximately unbiased for test MSE.

Problem MSE_i is a terribly noisy estimator for test MSE (albeit ≈unbiased).

† And because often n-1 ≈ n.

Hold-out methods

Option 2: Leave-one-out cross validation

Because **LOOCV uses n-1 observations** to train the model,[†] MSE_i (validation MSE from observation i) is approximately unbiased for test MSE.

Problem MSE_i is a terribly noisy estimator for test MSE (albeit ≈unbiased).

Solution Take the mean!

$$\text{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\text{MSE}_i$$

† And because often n-1 ≈ n.

Hold-out methods

Option 2: Leave-one-out cross validation

Because **LOOCV uses n-1 observations** to train the model,[†] MSE_i (validation MSE from observation i) is approximately unbiased for test MSE.

Problem MSE_i is a terribly noisy estimator for test MSE (albeit ≈unbiased).

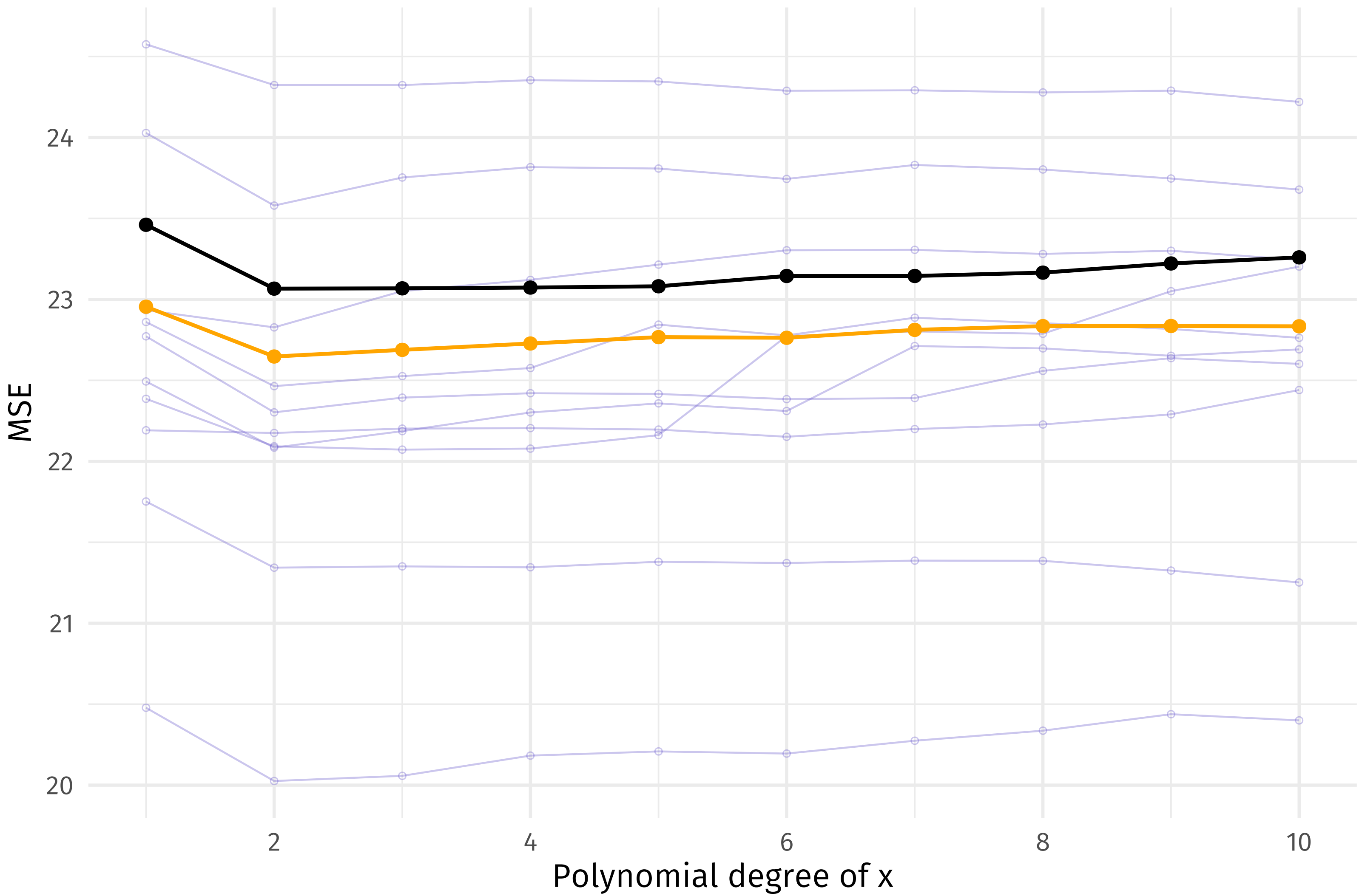
Solution Take the mean!

$$\text{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \text{MSE}_i$$

1. LOOCV **reduces bias** by using n-1 (almost all) observations for training.
2. LOOCV **resolves variance**: it makes all possible comparison (no dependence upon which validation-test split you make).

† And because often n-1 ≈ n.

True test MSE and **LOOCV MSE** compared to **validation-set estimates**



Option 3: k-fold cross validation

Leave-one-out cross validation is a special case of a broader strategy: **k-fold cross validation**.

1. **Divide** the training data into k equally sized groups (folds).
2. **Iterate** over the k folds, treating each as a validation set once (training the model on the other $k-1$ folds).
3. **Average** the folds' MSEs to estimate test MSE.

Hold-out methods

Option 3: k-fold cross validation

Leave-one-out cross validation is a special case of a broader strategy: **k-fold cross validation**.

1. **Divide** the training data into k equally sized groups (folds).
2. **Iterate** over the k folds, treating each as a validation set once (training the model on the other $k-1$ folds).
3. **Average** the folds' MSEs to estimate test MSE.

Benefits?

Option 3: k-fold cross validation

Leave-one-out cross validation is a special case of a broader strategy: **k-fold cross validation**.

1. **Divide** the training data into k equally sized groups (folds).
2. **Iterate** over the k folds, treating each as a validation set once (training the model on the other $k - 1$ folds).
3. **Average** the folds' MSEs to estimate test MSE.

Benefits?

1. **Less computationally demanding** (fit model $k = 5$ or 10 times; not n).

Hold-out methods

Option 3: k-fold cross validation

Leave-one-out cross validation is a special case of a broader strategy: **k-fold cross validation**.

1. **Divide** the training data into k equally sized groups (folds).
2. **Iterate** over the k folds, treating each as a validation set once (training the model on the other $k-1$ folds).
3. **Average** the folds' MSEs to estimate test MSE.

Benefits?

1. **Less computationally demanding** (fit model $k = 5$ or 10 times; not n).
2. **Greater accuracy** (in general) due to bias-variance tradeoff!

Hold-out methods

Option 3: k-fold cross validation

Leave-one-out cross validation is a special case of a broader strategy: **k-fold cross validation**.

1. **Divide** the training data into k equally sized groups (folds).
2. **Iterate** over the k folds, treating each as a validation set once (training the model on the other $k-1$ folds).
3. **Average** the folds' MSEs to estimate test MSE.

Benefits?

1. **Less computationally demanding** (fit model $k =$ 5 or 10 times; not n).
2. **Greater accuracy** (in general) due to bias-variance tradeoff!
    - Somewhat higher bias, relative to LOOCV: $n-1$ vs. $(k-1)/k$.

Hold-out methods

Option 3: k-fold cross validation

Leave-one-out cross validation is a special case of a broader strategy: **k-fold cross validation**.

1. **Divide** the training data into k equally sized groups (folds).
2. **Iterate** over the k folds, treating each as a validation set once (training the model on the other $k-1$ folds).
3. **Average** the folds' MSEs to estimate test MSE.

Benefits?

1. **Less computationally demanding** (fit model $k=5$ or 10 times; not n).
2. **Greater accuracy** (in general) due to bias-variance tradeoff!
   - Somewhat higher bias, relative to LOOCV: $n-1$ vs. $(k-1)/k$.
   - Lower variance due to high-degree of correlation in LOOCV MSE_i.

Hold-out methods

Option 3: k-fold cross validation

Leave-one-out cross validation is a special case of a broader strategy:
k-fold cross validation.

1. **Divide** the training data into k equally sized groups (folds).
2. **Iterate** over the k folds, treating each as a validation set once (training the model on the other $k-1$ folds).
3. **Average** the folds' MSEs to estimate test MSE.

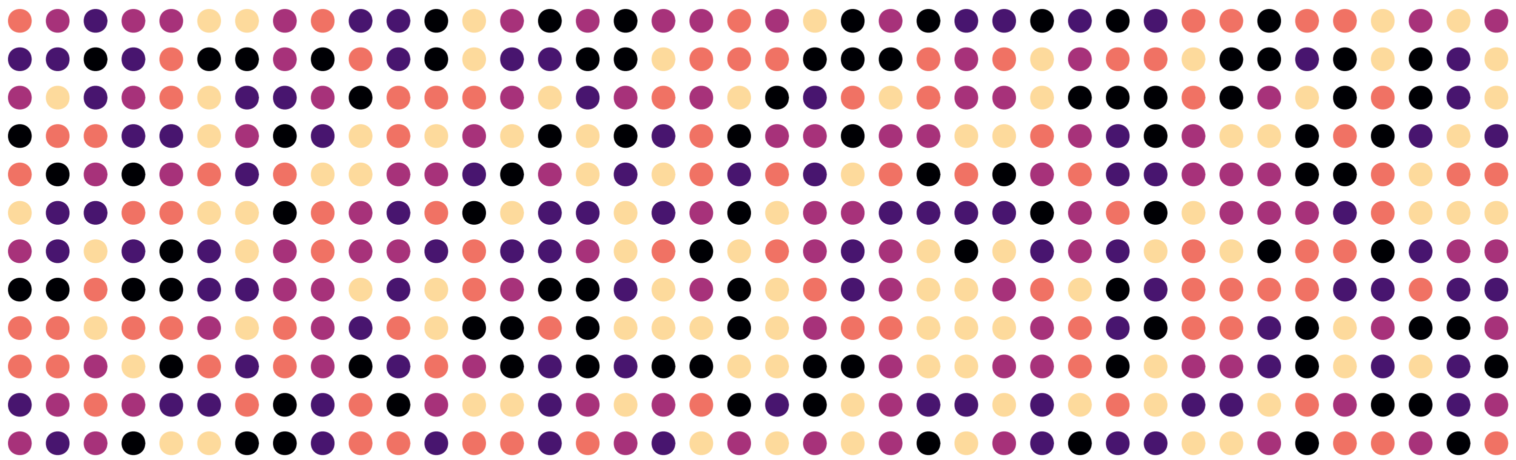
Benefits?

1. **Less computationally demanding** (fit model $k =$ 5 or 10 times; not n).
2. **Greater accuracy** (in general) due to bias-variance tradeoff!
    - Somewhat higher bias, relative to LOOCV: $n-1$ vs. $(k-1)/k$.
    - Lower variance due to high-degree of correlation in LOOCV MSE_i.🤯

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\text{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i$$



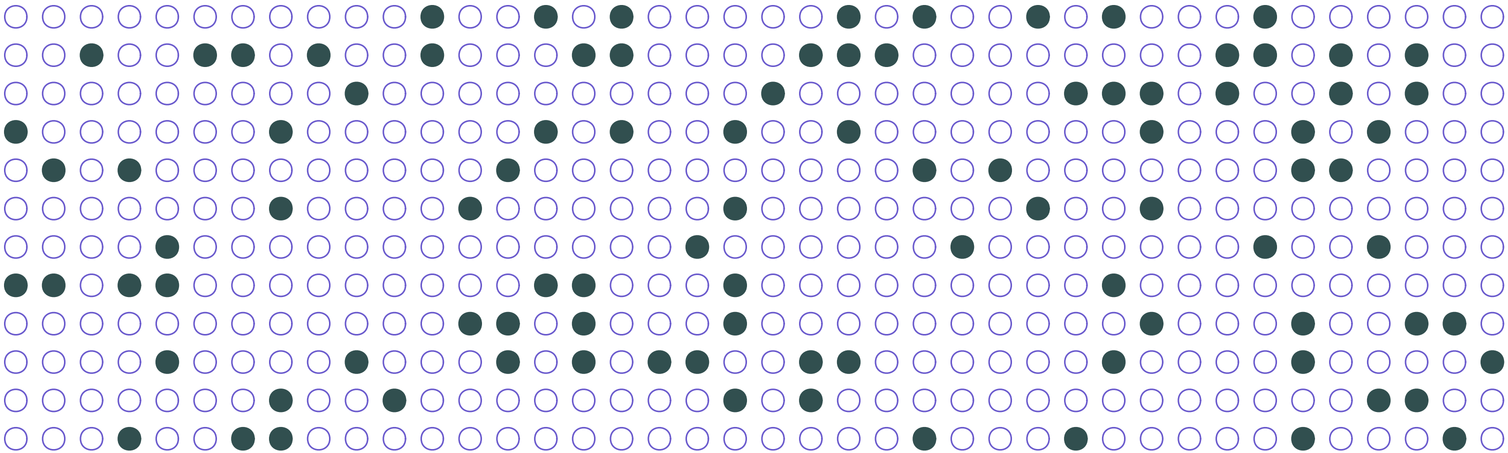
Our $k = 5$ folds.

Hold-out methods

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\text{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i$$



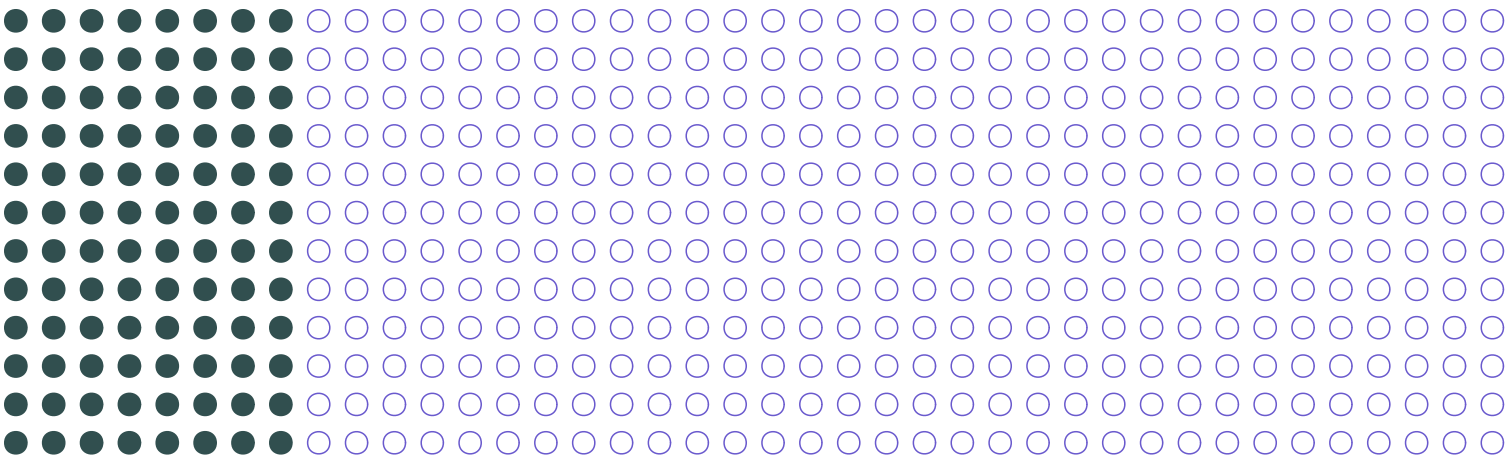
Each fold takes a turn at **validation**. The other $k - 1$ folds **train**.

Hold-out methods

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\text{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i$$



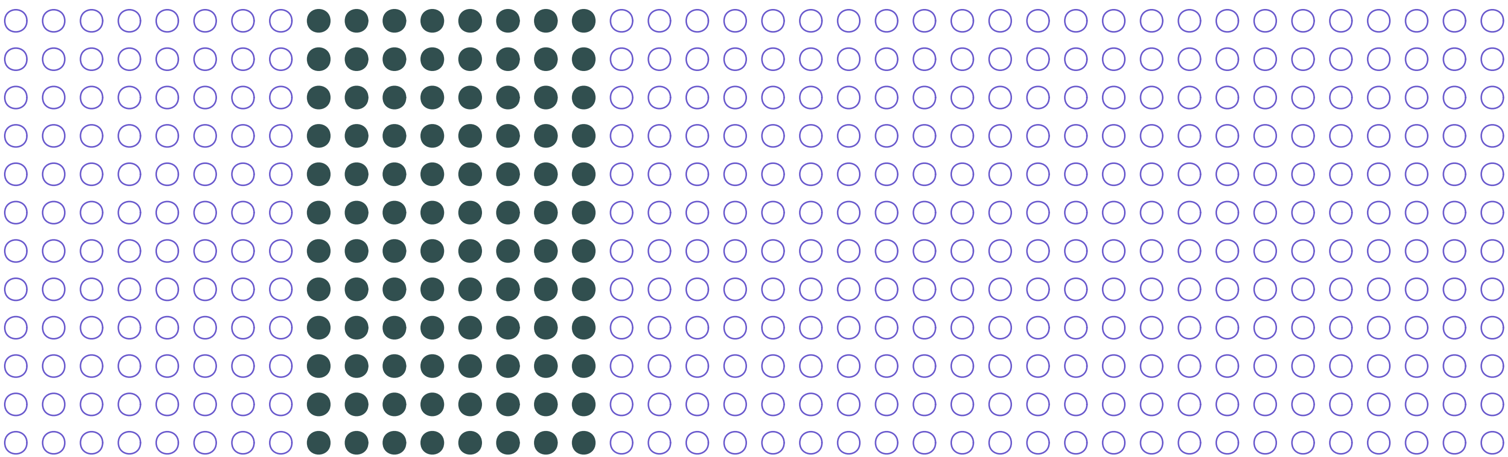
For $k = 5$, fold number 1 as the **validation set** produces $\text{MSE}_{k=1}$.

Hold-out methods

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\text{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i$$



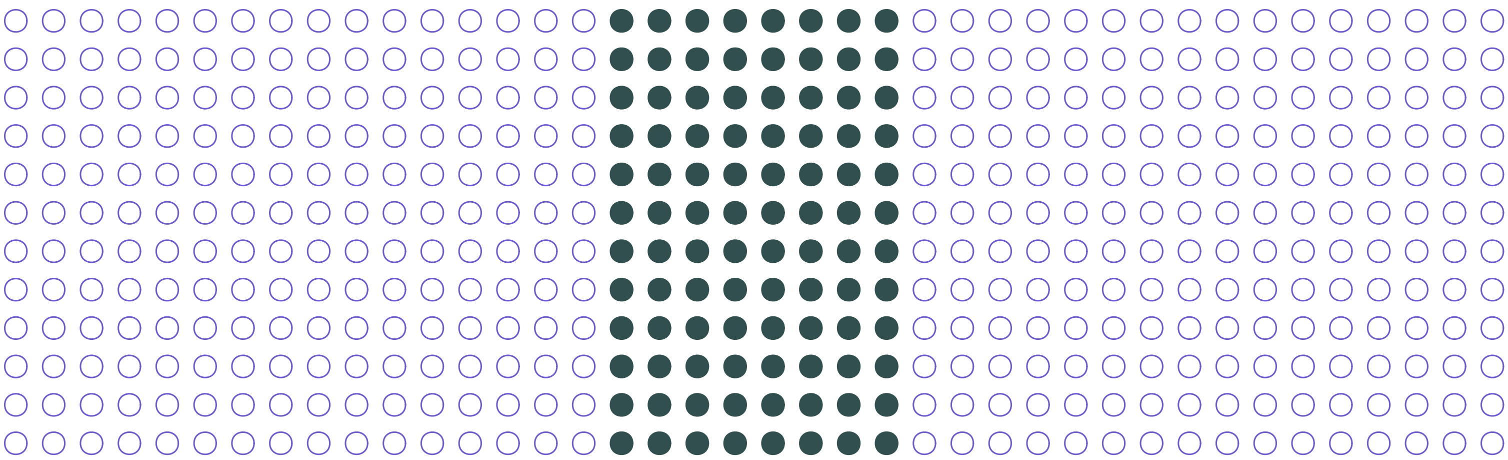
For $k = 5$, fold number 2 as the **validation set** produces $\text{MSE}_{k=2}$.

Hold-out methods

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\text{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\text{MSE}_i$$



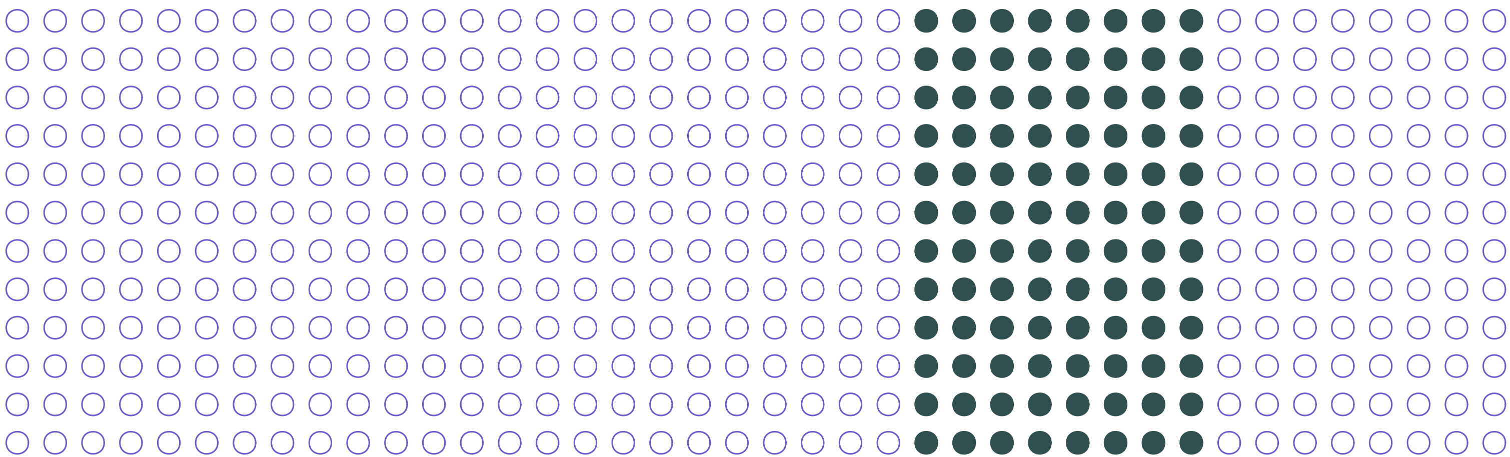
For $k = 5$, fold number 3 as the **validation set** produces $\text{MSE}_{k=3}$.

Hold-out methods

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\text{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \text{MSE}_i$$



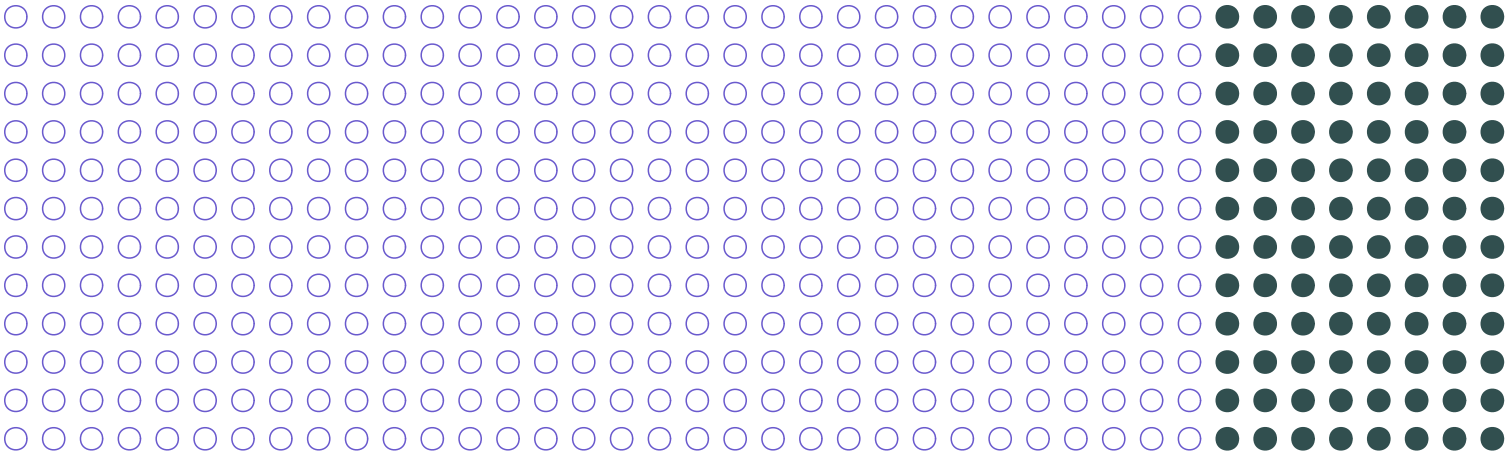
For $k = 5$, fold number 4 as the **validation set** produces $\text{MSE}_{k=4}$.

Hold-out methods

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\text{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\text{MSE}_i$$

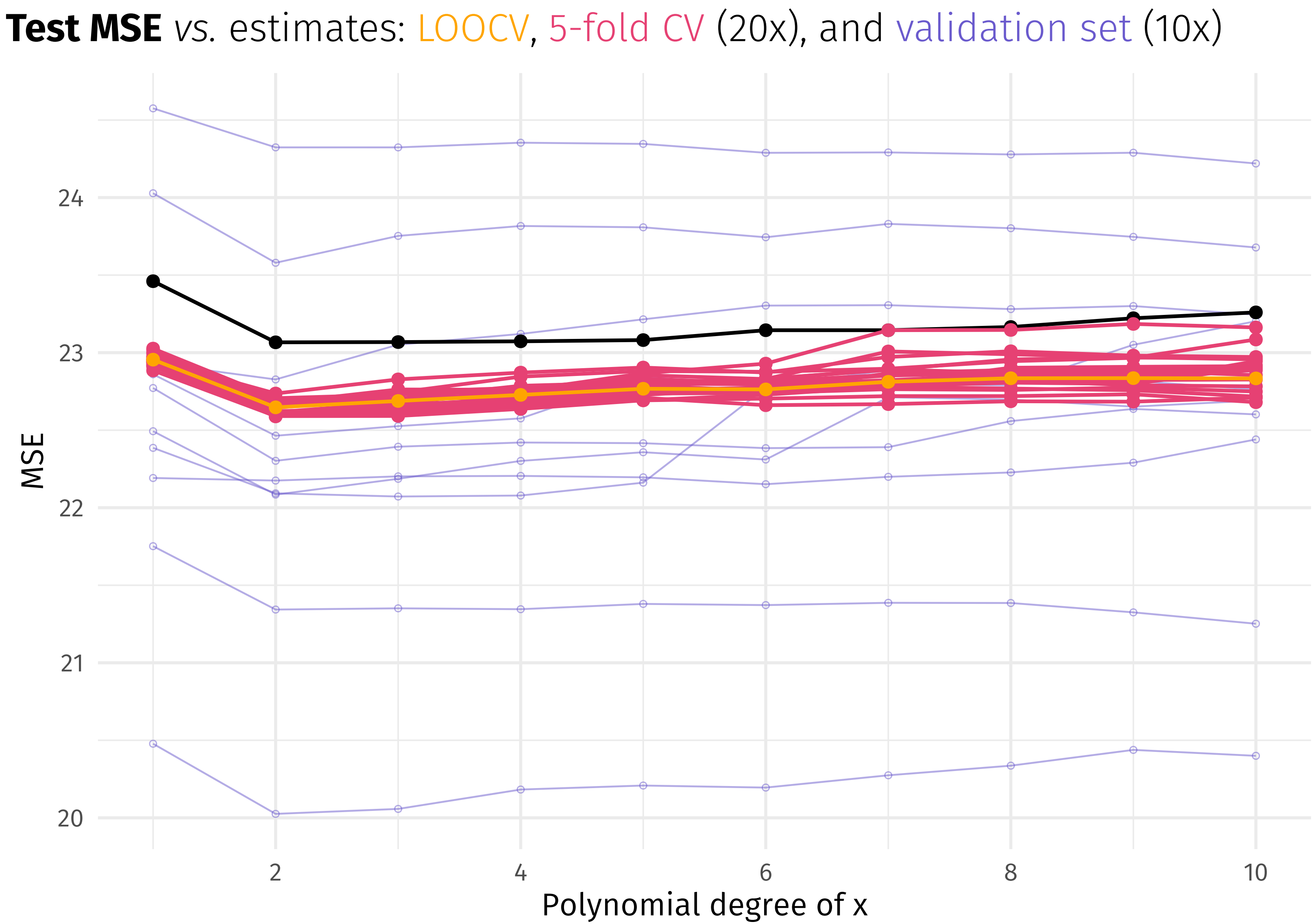


For $k = 5$, fold number 5 as the **validation set** produces $\text{MSE}_{k=5}$.

Option 3: k-fold cross validation

With k-fold cross validation, we estimate test MSE as

$$\mathrm{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \mathrm{MSE}_i$$



Test MSE vs. estimates: LOOCV, 5-fold CV (20x), and validation set (10x)
24
23
22
21
20
MSE
2
4
6
8
10
Polynomial degree of x

Note: Each of these methods extends to classification settings, *e.g.*, LOOCV

$$\text{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(y_i \neq \hat{y}_i)$$

Hold-out methods

Caveat

So far, we've treated each observation as separate/independent from each other observation.

The methods that we've defined so far actually need this independence.

Hold-out methods

Goals and alternatives

You can use CV for either of two important **modeling tasks:**

- **Model selection** Choosing and tuning a model
- **Model assessment** Evaluating a model's accuracy

Hold-out methods

Goals and alternatives

You can use CV for either of two important **modeling tasks:**

- **Model selection** Choosing and tuning a model

- **Model assessment** Evaluating a model's accuracy

Alternative approach: **Shrinkage methods**

- fit a model that contains all p predictors
- simultaneously: shrink† coefficients toward zero

† Synonyms for *shrink*: constrain or regularize

Goals and alternatives

You can use CV for either of two important **modeling tasks:**

- **Model selection** Choosing and tuning a model
- **Model assessment** Evaluating a model's accuracy

Alternative approach: **Shrinkage methods**

- fit a model that contains all p predictors
- simultaneously: shrink† coefficients toward zero

Idea: Penalize the model for coefficients as they move away from zero.

† Synonyms for *shrink*: constrain or regularize

Shrinkage

Why?

Q How could shrinking coefficients twoard zero help or predictions?

Shrinkage

Why?

Q How could shrinking coefficients twoard zero help or predictions?

A Remember we're generally facing a tradeoff between bias and variance.

Shrinkage

Why?

Q How could shrinking coefficients twoard zero help or predictions?

A Remember we're generally facing a tradeoff between bias and variance.

- Shrinking our coefficients toward zero **reduces the model's variance**.[†]
- **Penalizing** our model for **larger coefficients** shrinks them toward zero.
- The **optimal penalty** will balance reduced variance with increased bias.

† Imagine the extreme case: a model whose coefficients are all zeros has no variance.

Shrinkage

Why?

Q How could shrinking coefficients twoard zero help or predictions?

A Remember we're generally facing a tradeoff between bias and variance.

- Shrinking our coefficients toward zero **reduces the model's variance**.[†]
- **Penalizing** our model for **larger coefficients** shrinks them toward zero.
- The **optimal penalty** will balance reduced variance with increased bias.

Now you understand shrinkage methods.

- Ridge regression
- Lasso
- Elasticnet

† Imagine the extreme case: a model whose coefficients are all zeros has no variance.

Ridge regression

Ridge regression

Back to least squares (again)

Recall Least-squares regression gets $\hat{\beta}_j$'s by minimizing RSS, *i.e.*,

$$\min_{\hat{\beta}} \text{RSS} = \min_{\hat{\beta}} \sum_{i=1}^{n} e_i^2 = \min_{\hat{\beta}} \sum_{i=1}^{n} \left(y_i - \underbrace{\left[\hat{\beta}_0 + \hat{\beta}_1 x_{i,1} + \cdots + \hat{\beta}_p x_{i,p} \right]}_{=\hat{y}_i} \right)^2$$

Back to least squares (again)

Recall Least-squares regression gets $\hat\beta_j$'s by minimizing RSS, *i.e.*,

$$\min_{\hat\beta} \text{RSS} = \min_{\hat\beta} \sum_{i=1}^{n} e_i^2 = \min_{\hat\beta} \sum_{i=1}^{n} \left(y_i - \underbrace{\left[\hat\beta_0 + \hat\beta_1 x_{i,1} + \cdots + \hat\beta_p x_{i,p} \right]}_{=\hat{y}_i} \right)^2$$

Ridge regression makes a small change

- adds a shrinkage penalty = the sum of squared coefficents $\left(\lambda \sum_j \beta_j^2\right)$
- minimizes the (weighted) sum of RSS and the shrinkage penalty

Ridge regression

Back to least squares (again)

Recall Least-squares regression gets $\hat{\beta}_j$'s by minimizing RSS, *i.e.*,

$$\min_{\hat{\beta}} \text{RSS} = \min_{\hat{\beta}} \sum_{i=1}^{n} e_i^2 = \min_{\hat{\beta}} \sum_{i=1}^{n} \left(y_i - \underbrace{\left[\hat{\beta}_0 + \hat{\beta}_1 x_{i,1} + \cdots + \hat{\beta}_p x_{i,p} \right]}_{=\hat{y}_i} \right)^2$$

Ridge regression makes a small change

- adds a shrinkage penalty = the sum of squared coefficents $\left(\lambda \sum_j \beta_j^2 \right)$
- minimizes the (weighted) sum of RSS and the shrinkage penalty

$$\min_{\hat{\beta}^R} \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Ridge regression

Ridge regression

$$\min_{\hat{\beta}^R} \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Least squares

$$\min_{\hat{\beta}} \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2$$

λ (≥ 0) is a tuning parameter for the harshness of the penalty.

$\lambda = 0$ implies no penalty: we are back to least squares.

Ridge regression

Ridge regression

$$\min_{\hat{\beta}^R} \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Least squares

$$\min_{\hat{\beta}} \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2$$

λ (≥ 0) is a tuning parameter for the harshness of the penalty.

$\lambda = 0$ implies no penalty: we are back to least squares.

Each value of λ produces a new set of coefficents.

Ridge regression

Ridge regression

$$\min_{\hat{\beta}^R} \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Least squares

$$\min_{\hat{\beta}} \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2$$

λ (≥ 0) is a tuning parameter for the harshness of the penalty.

$\lambda = 0$ implies no penalty: we are back to least squares.

Each value of λ produces a new set of coefficents.

Ridge's approach to the bias-variance tradeoff: Balance

- reducing **RSS**, *i.e.*, $\sum_i \left(y_i - \hat{y}_i \right)^2$
- reducing **coefficients** (ignoring the intercept)

λ determines how much ridge "cares about" these two quantities.†

† With $\lambda = 0$, least-squares regression only "cares about" RSS.

Ridge regression

λ and penalization

Choosing a *good* value for λ is key.

- If λ is too small, then our model is essentially back to OLS.
- If λ is too large, then we shrink all of our coefficients too close to zero.

λ and penalization

Choosing a *good* value for λ is key.

- If λ is too small, then our model is essentially back to OLS.
- If λ is too large, then we shrink all of our coefficients too close to zero.

Q So what do we do?

Ridge regression

λ and penalization

Choosing a *good* value for λ is key.

- If λ is too small, then our model is essentially back to OLS.
- If λ is too large, then we shrink all of our coefficients too close to zero.

Q So what do we do?

A Cross validate!

(You saw that coming, right?)

Ridge regression

Penalization and standardization

Important Predictors' **units** can drastically **affect ridge regression results**.

Why?

Ridge regression

Penalization and standardization

Important Predictors' **units** can drastically **affect ridge regression results**.

Why? Because $\mathbf{x}_j$'s units affect β_j, and ridge is very sensitive to β_j.

Penalization and standardization

Important Predictors' **units** can drastically **affect ridge regression results**.

Why? Because $\mathbf{x}_j$'s units affect β_j, and ridge is very sensitive to β_j.

Example Let x_1 denote distance.

Least-squares regression

If x_1 is *meters* and $\beta_1 = \mathbf{3}$, then when x_1 is *km*, $\beta_1 = \mathbf{3,000}$.

The scale/units of predictors do not affect least squares' estimates.

Penalization and standardization

Important Predictors' **units** can drastically **affect ridge regression results**.

Why? Because $\mathbf{x}_j$'s units affect β_j, and ridge is very sensitive to β_j.

Example Let x_1 denote distance.

Least-squares regression

If x_1 is *meters* and $\beta_1 = \mathbf{3}$, then when x_1 is *km*, $\beta_1 = \mathbf{3,000}$.

The scale/units of predictors do not affect least squares' estimates.

Ridge regression pays a much larger penalty for $\beta_1 = \mathbf{3,000}$ than $\beta_1 = \mathbf{3}$.

You will not get the same (scaled) estimates when you change units.

Ridge regression

Penalization and standardization

Important Predictors' **units** can drastically **affect ridge regression results**.

Why? Because $\mathbf{x}_j$'s units affect β_j, and ridge is very sensitive to β_j.

Example Let x_1 denote distance.

Least-squares regression

If x_1 is *meters* and $\beta_1 = \mathbf{3}$, then when x_1 is *km*, $\beta_1 = \mathbf{3,000}$.

The scale/units of predictors do not affect least squares' estimates.

Ridge regression pays a much larger penalty for $\beta_1 = \mathbf{3,000}$ than $\beta_1 = \mathbf{3}$.

You will not get the same (scaled) estimates when you change units.

Solution Standardize your variables, *i.e.*, `x_stnd = (x - mean(x))/sd(x)`.

Lasso

Lasso

Intro

Lasso simply replaces ridge's *squared* coefficients with absolute values.

Lasso

Intro

Lasso simply replaces ridge's *squared* coefficients with absolute values.

Ridge regression

$$\min_{\hat{\beta}^R} \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Lasso

$$\min_{\hat{\beta}^L} \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2 + \lambda \sum_{j=1}^{p} \left| \beta_j \right|$$

Everything else will be the same—except one aspect...

Lasso

Shrinkage

Unlike ridge, lasso's penalty does not increase with the size of β_j.

You always pay λ to increase $|\beta_j|$ by one unit.

Lasso

Shrinkage

Unlike ridge, lasso's penalty does not increase with the size of β_j.

You always pay λ to increase $|\beta_j|$ by one unit.

The only way to avoid lasso's penalty is to **set coefficents to zero**.

Lasso

Shrinkage

Unlike ridge, lasso's penalty does not increase with the size of β_j.

You always pay λ to increase $|\beta_j|$ by one unit.

The only way to avoid lasso's penalty is to **set coefficents to zero**.

This feature has two **benefits**

1. Some coefficients will be **set to zero**—we get "sparse" models.
2. Lasso can be used for subset/feature **selection**.

Lasso

Shrinkage

Unlike ridge, lasso's penalty does not increase with the size of β_j.

You always pay λ to increase $|\beta_j|$ by one unit.

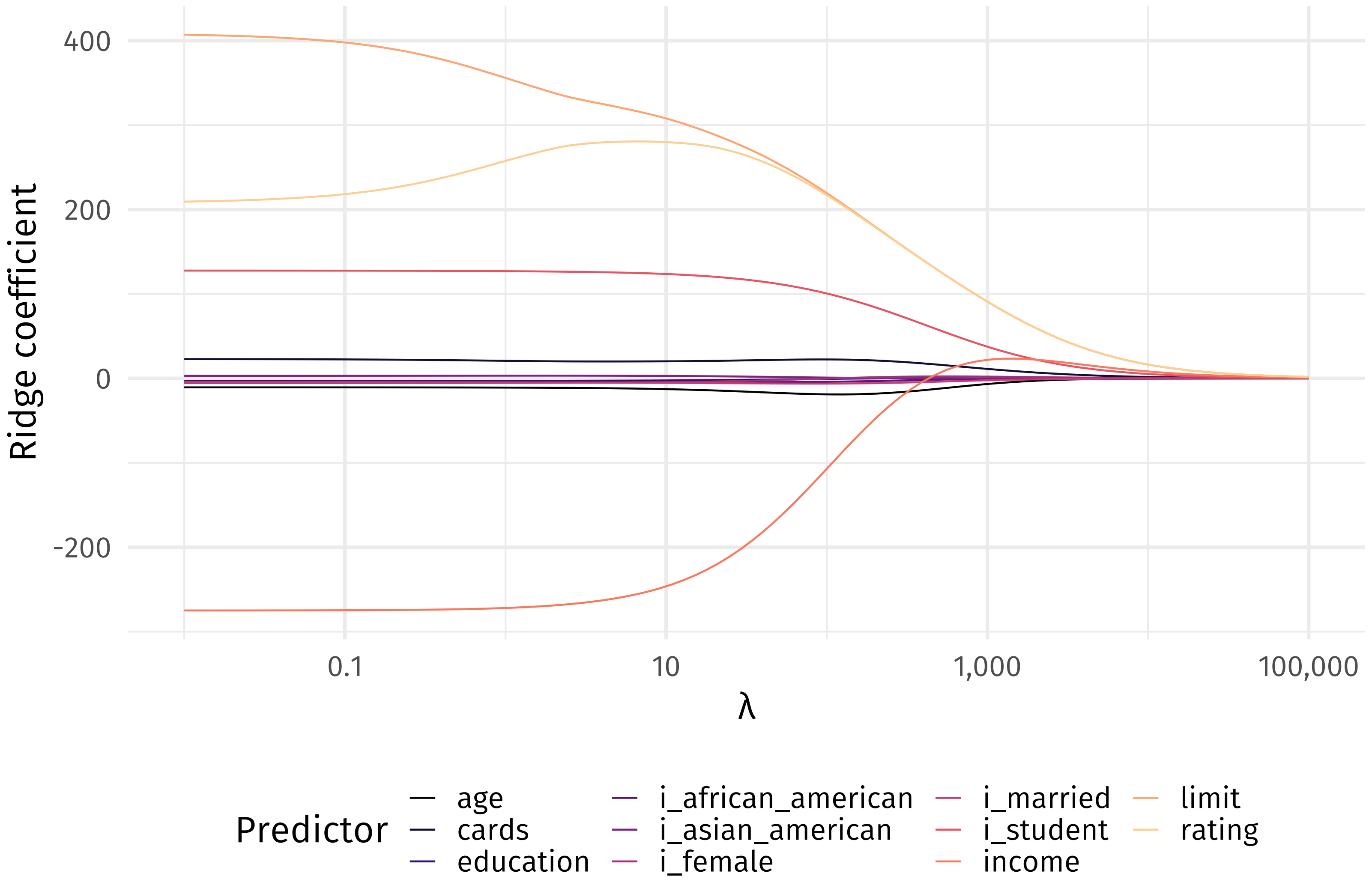
The only way to avoid lasso's penalty is to **set coefficents to zero**.

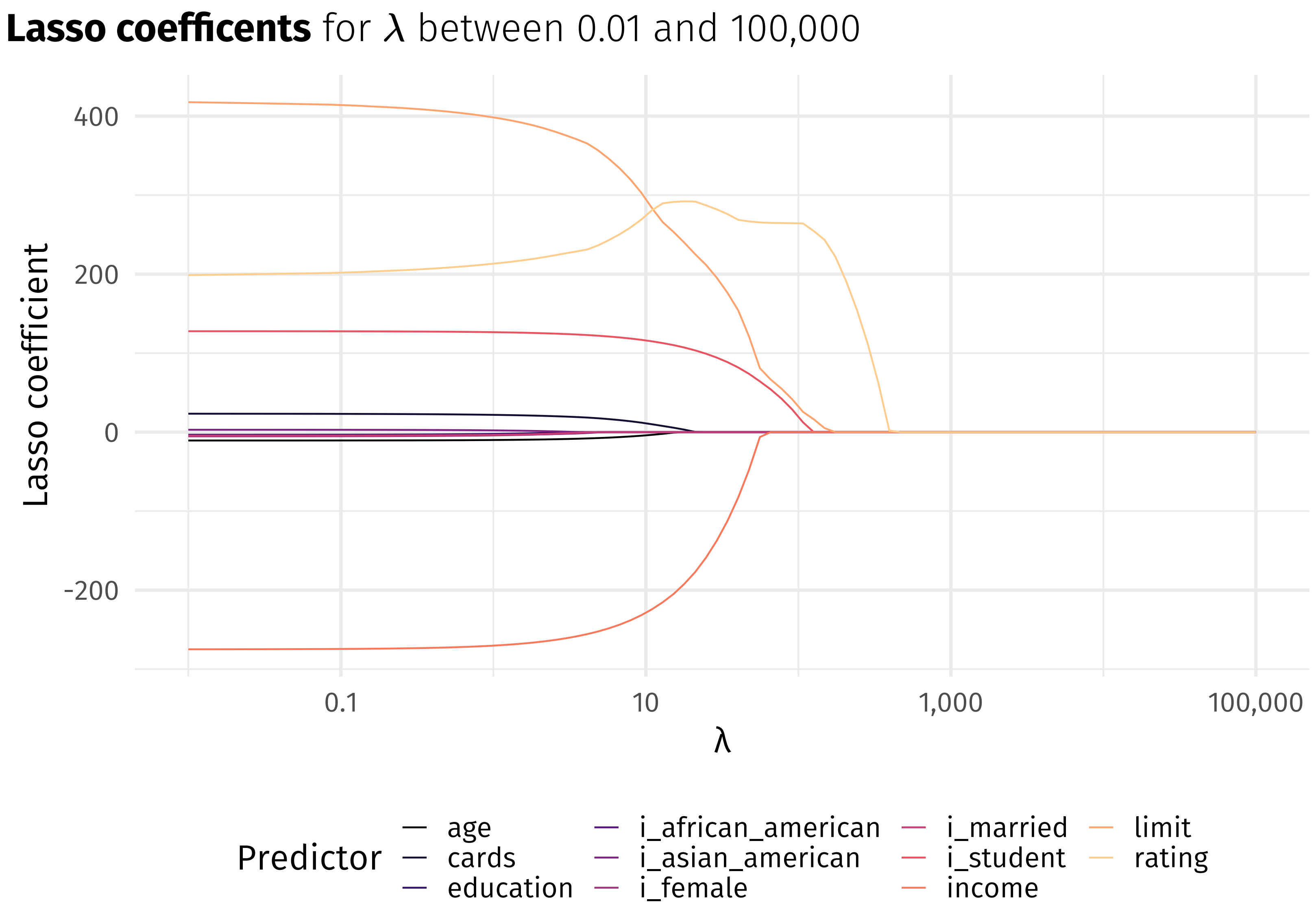
This feature has two **benefits**

1. Some coefficients will be **set to zero**—we get "sparse" models.
2. Lasso can be used for subset/feature **selection**.

We will still need to carefully select λ.

Ridge regression coefficents for λ between 0.01 and 100,000




Lasso coefficents for λ between 0.01 and 100,000
400
200
0
-200
0.1
10
1,000
100,000
λ
Lasso coefficient
Predictor
age
cards
education
i_african_american
i_asian_american
i_female
i_married
i_student
income
limit
rating

Wrap up

Now you understand the basic tenants of machine learning:

- How **prediction** differs from causal inference
- **Bias-variance tradeoff** (the benefits and costs of flexibility)
- **Cross validation**: Performance and tuning
- In- *vs*. out-of-sample **performance**

Sources

Sources (articles) of images

- Deep learning and radiology
- Parking lot detection
- *New Yorker* writing
- Gender Shades

I pulled the comic from Twitter.