Prediction

Week 13

AEM 2850: R for Business Analytics

Cornell Dyson

Spring 2022

Acknowledgements: Ed Rubin

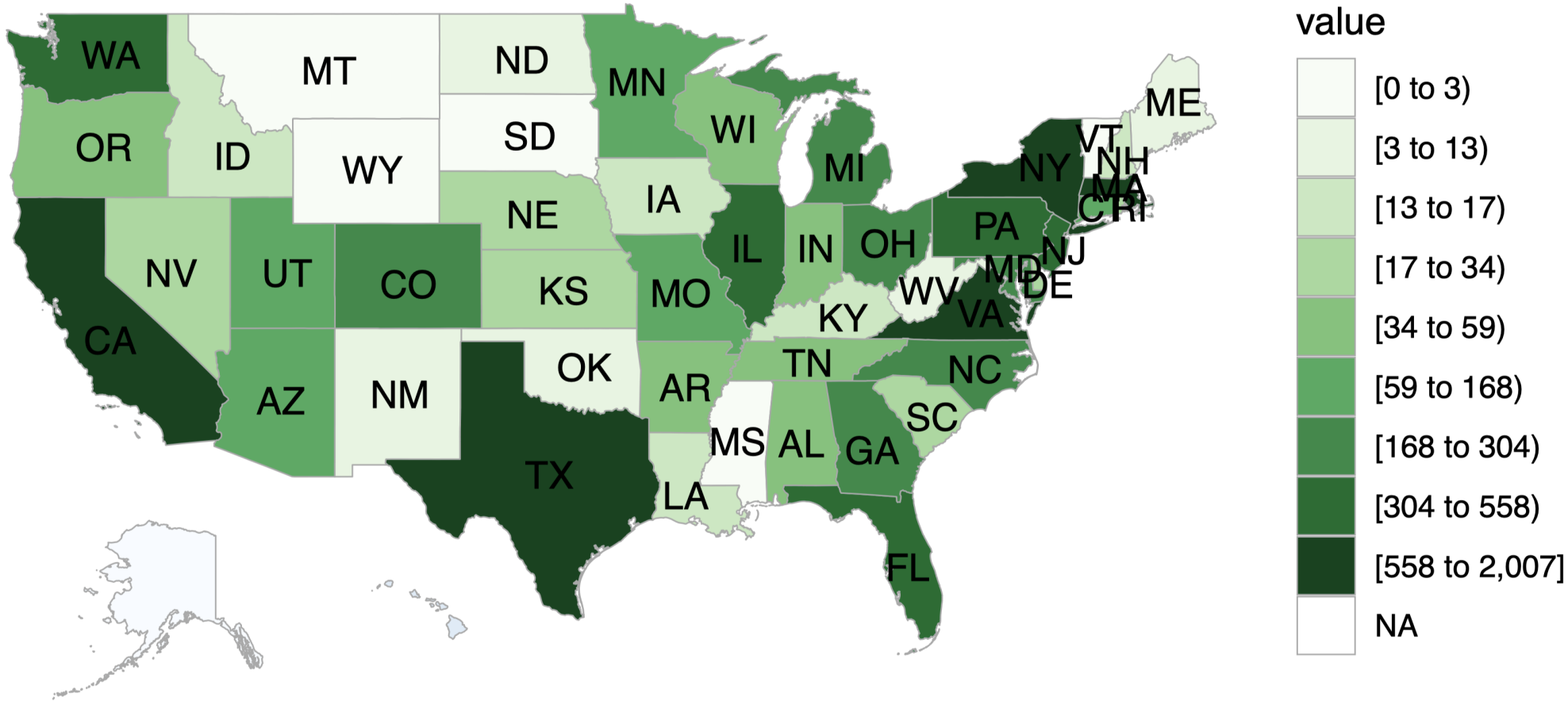
Announcements

Final project:

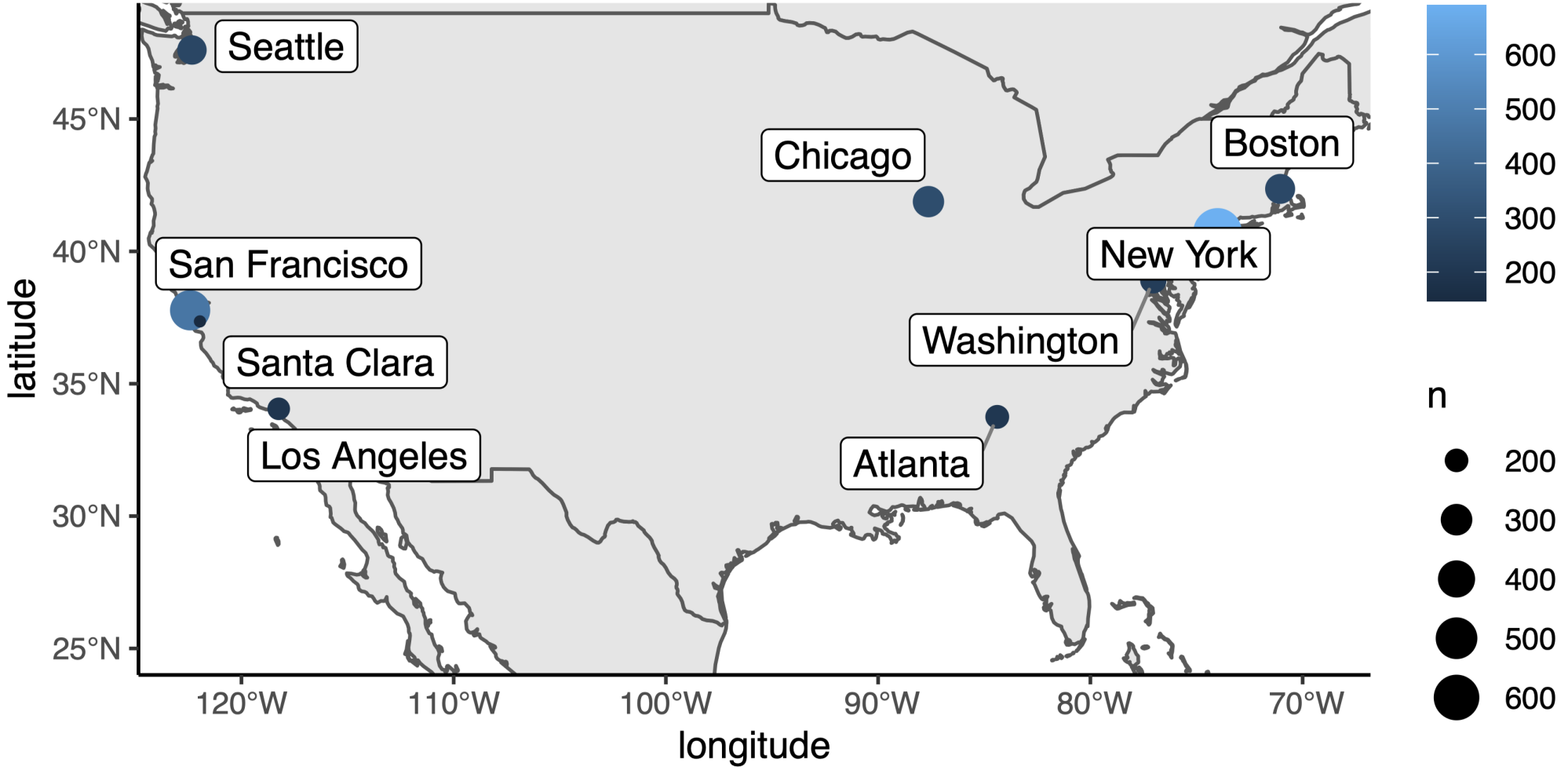
- Open-ended
- Choose your own groups of 3 by this Friday
- Choose your own data (soon!)
- We will post more guidance this week
- Due May 19 at 4:30pm

Questions before we get started?

Creative answers to Lab 12, Question 7



Creative answers to Lab 12, Question 7



Plan for today

Prologue

Linear regression

Shrinkage methods

Ridge regression

Lasso

Classification on Thursday, time permitting

Prologue

Prediction vs causal inference

What is prediction?

Why is prediction useful in business?

What is causal inference?

Why is causal inference useful?

Prediction vs causal inference

What do prediction and causal inference mean in this setting?

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

In our week on **relationships**, we estimated $\hat{\beta}$ to quantify *associations*

This week: *prediction* of Y_i, *i.e.*, constructing $\hat{\mathbf{Y}}_i$?

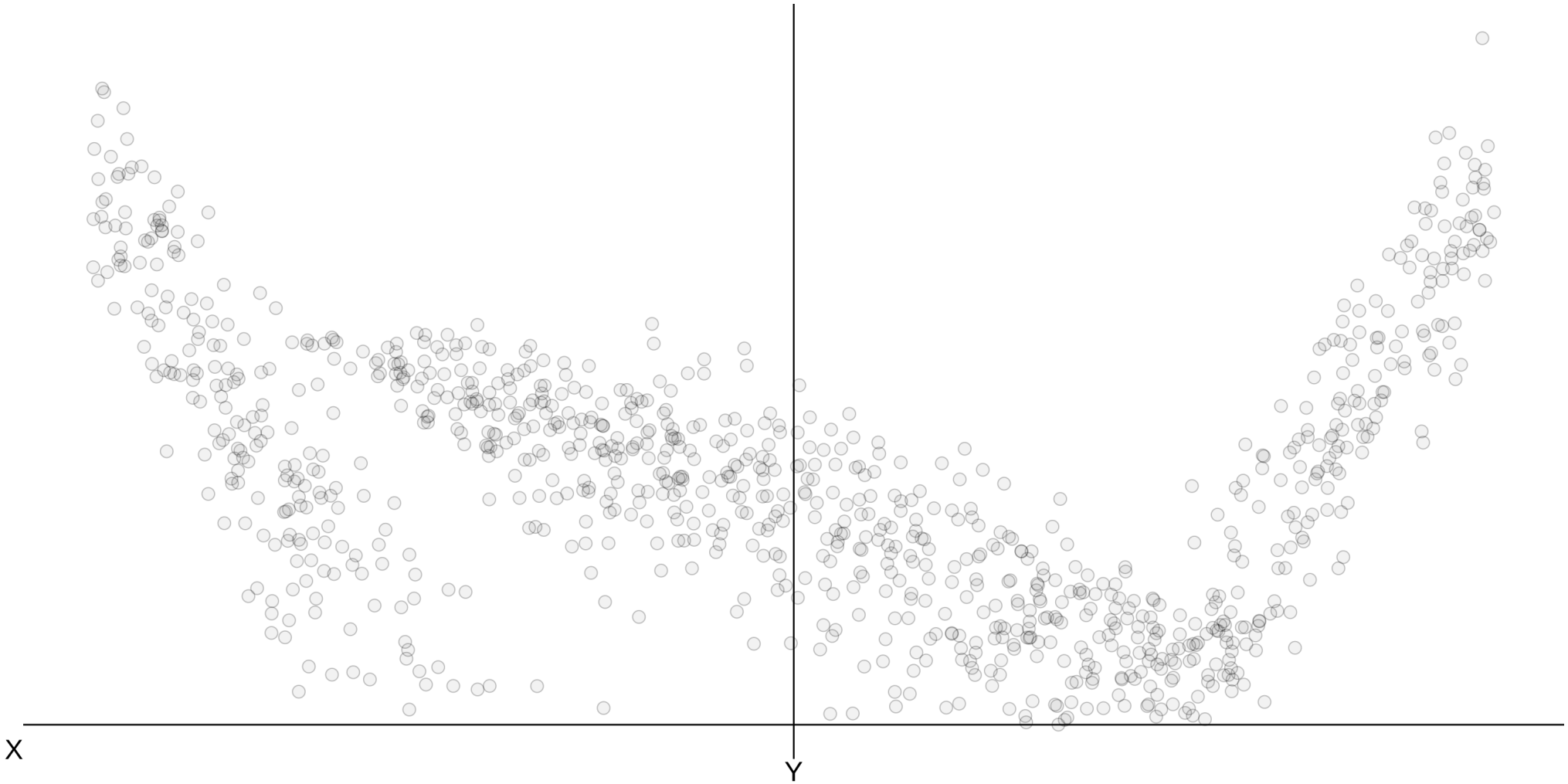
Next week: *causal inference* on β

This is not a methods class!

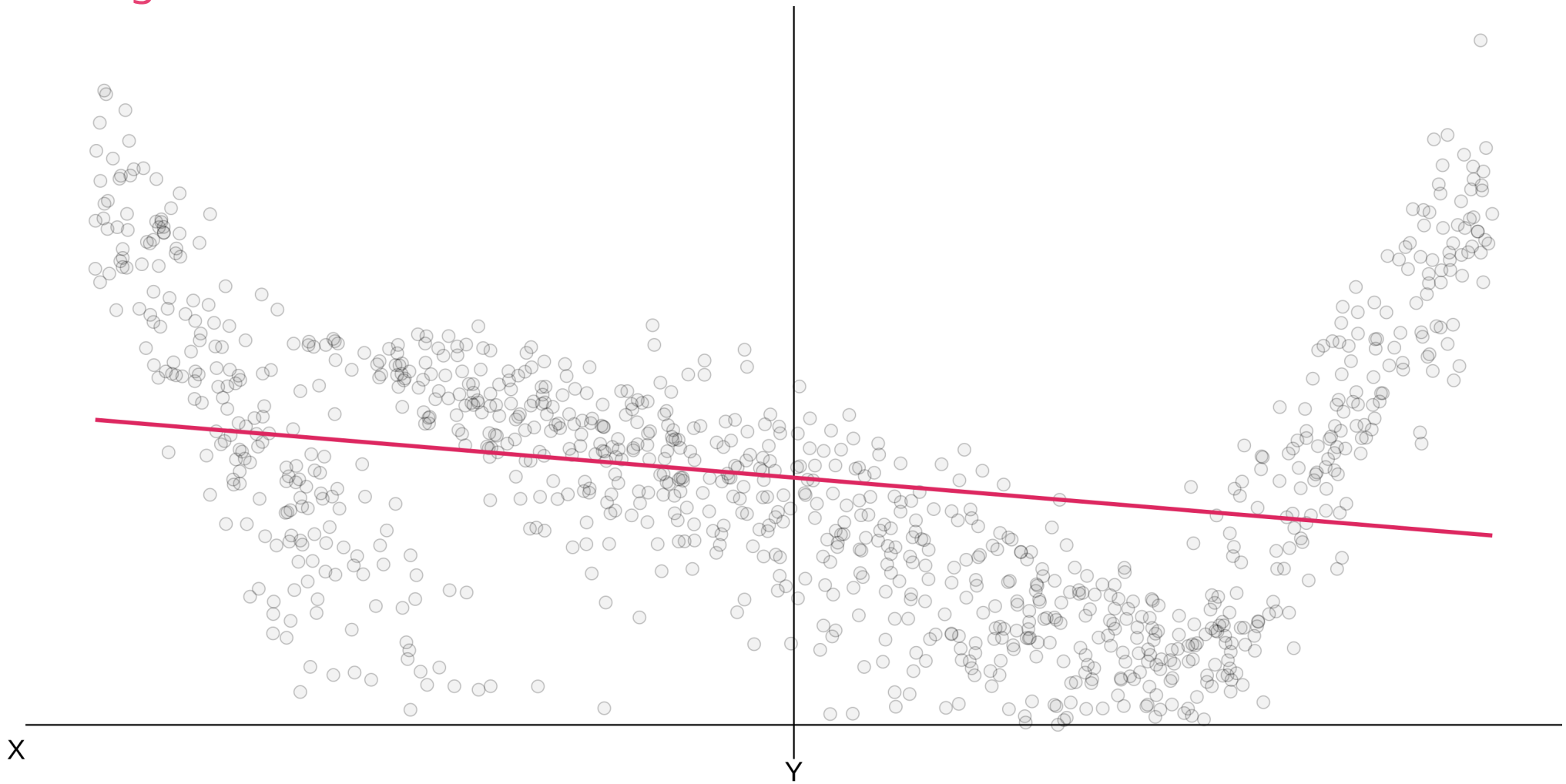
Prediction

Can we just use OLS like we did before?

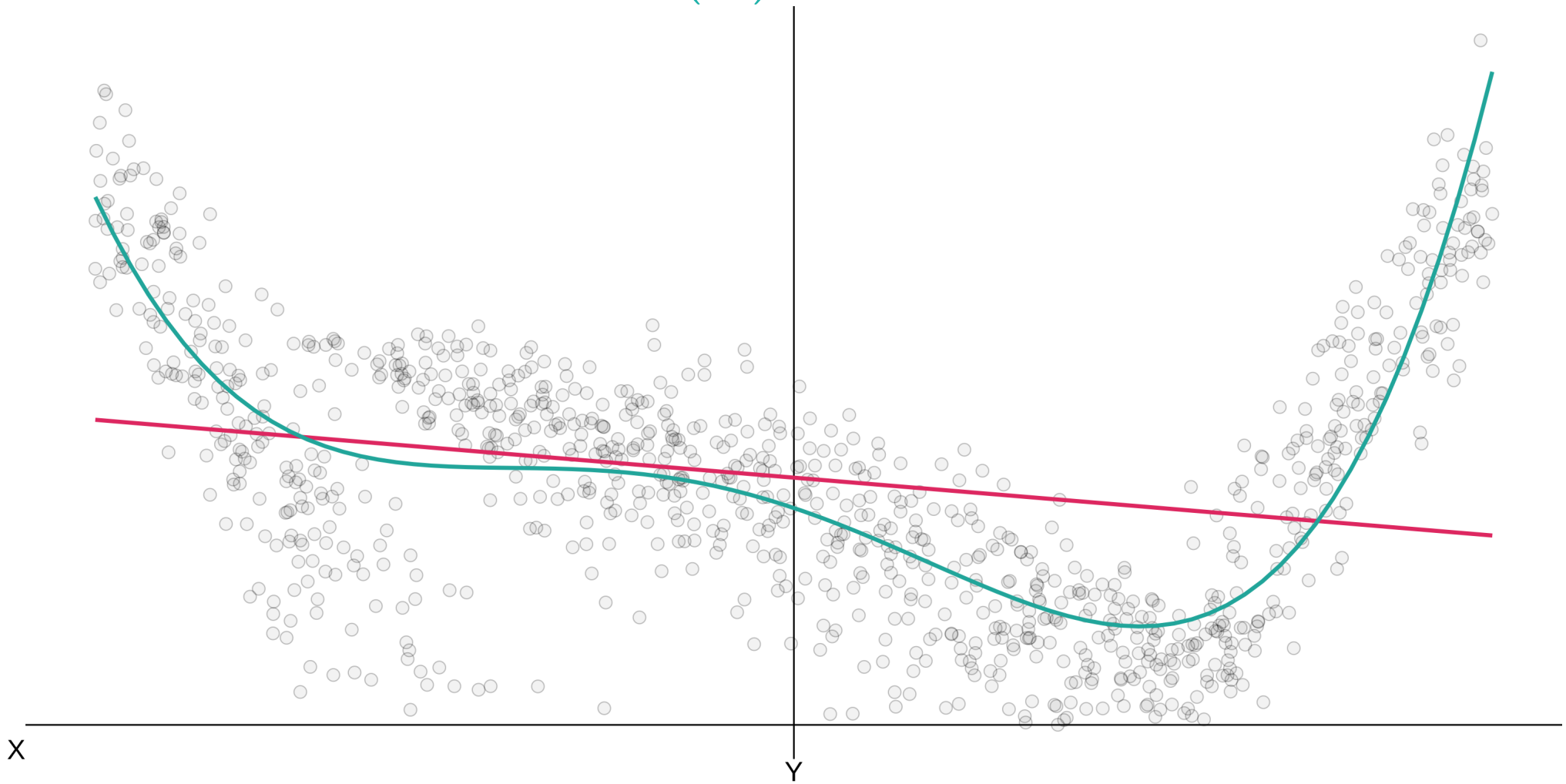
It depends


X
Y

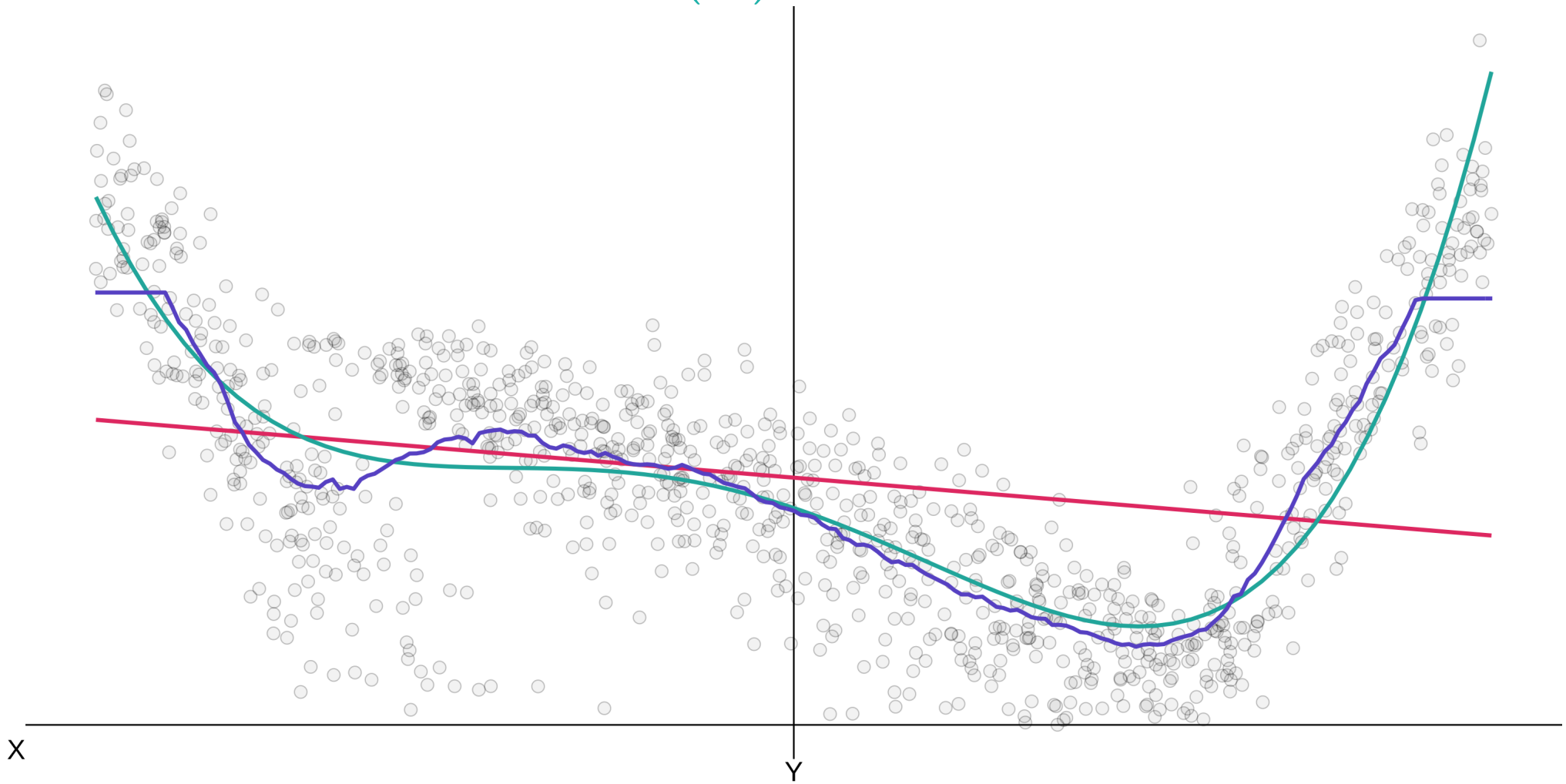
Linear regression



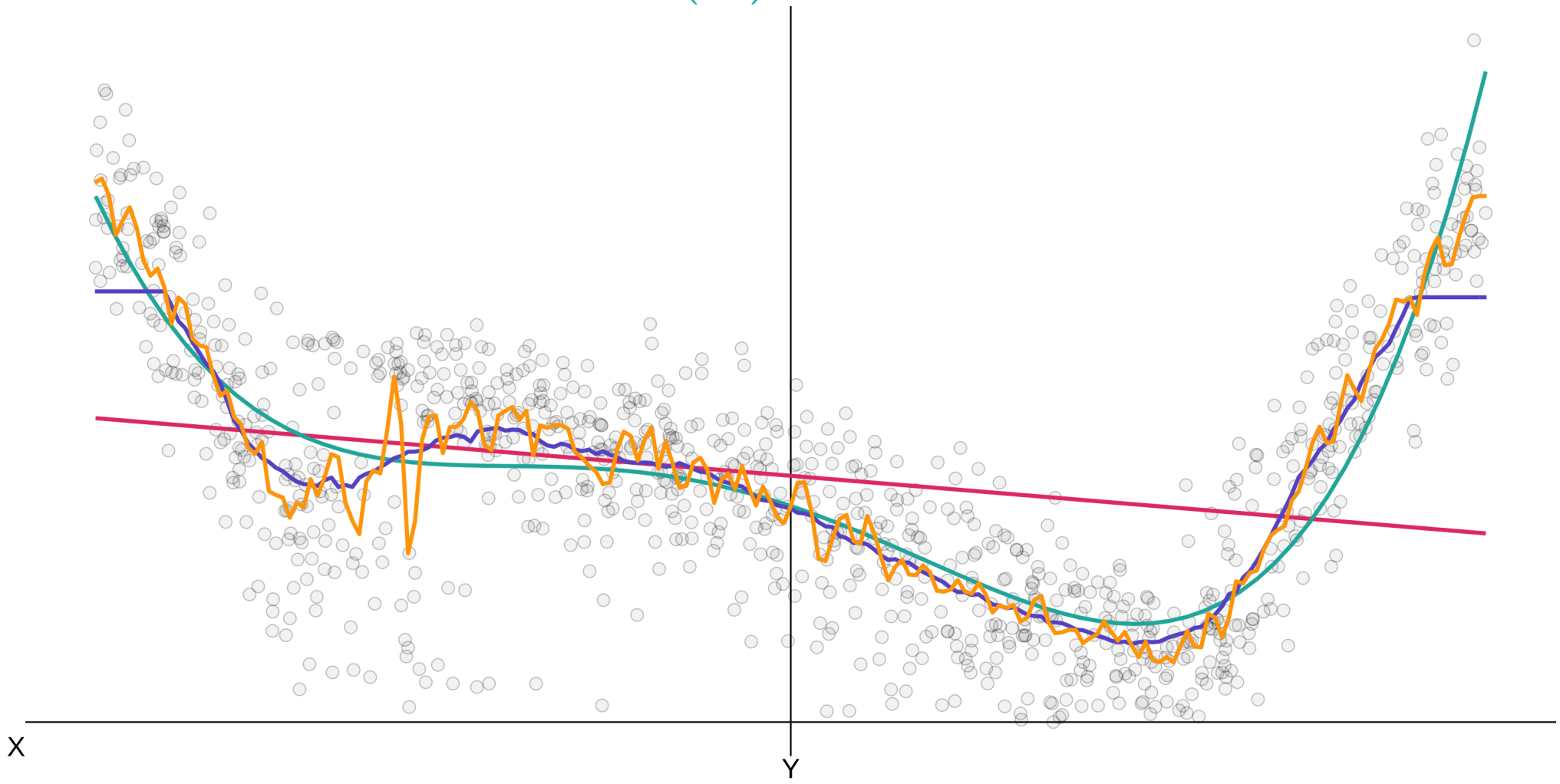
Linear regression, linear regression $\left(x^4\right)$



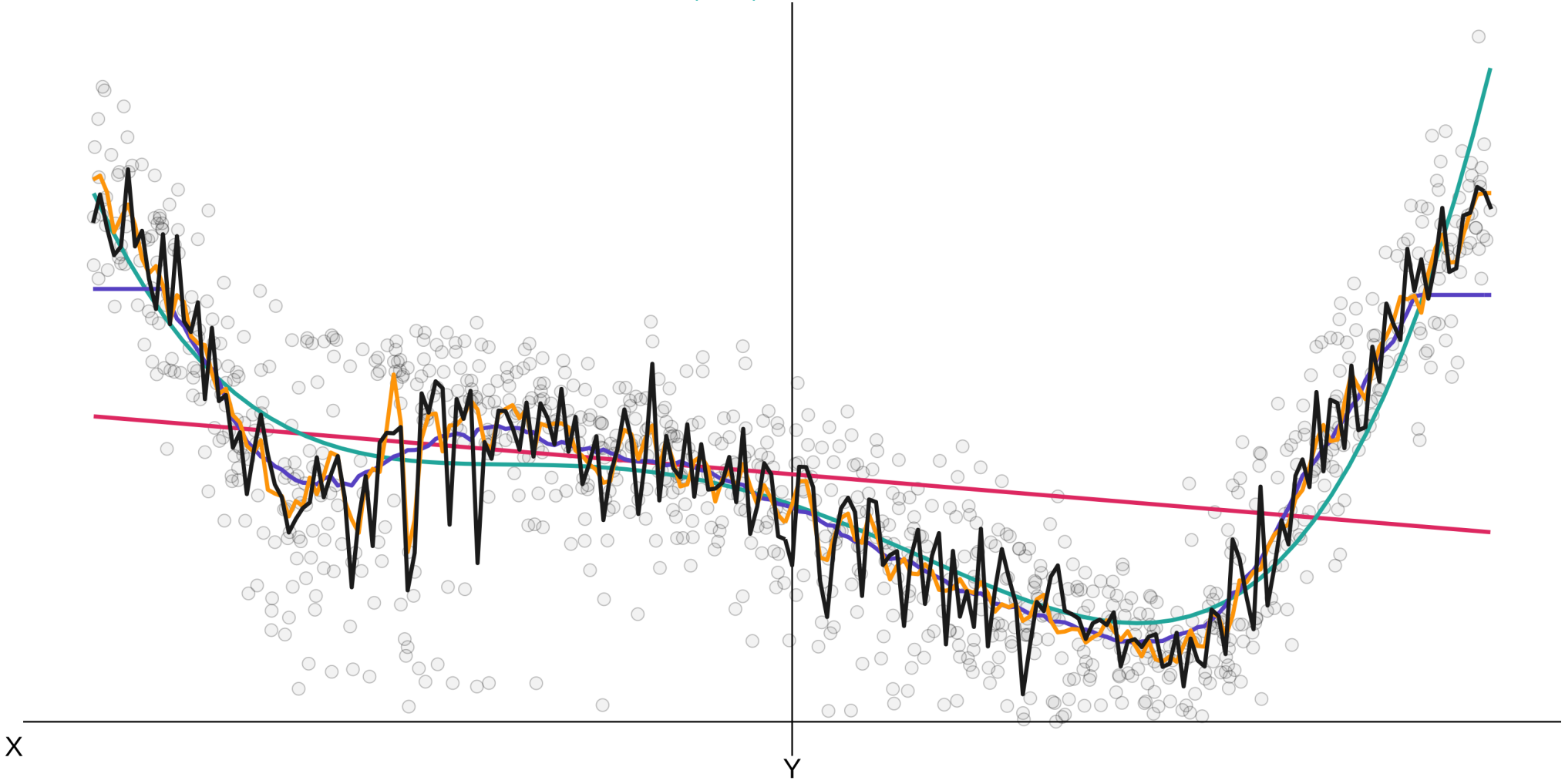
Linear regression, linear regression $\left(x^4\right)$, KNN (100)



Linear regression, linear regression $\left(x^4\right)$, KNN (100), KNN (10)



Linear regression, linear regression $\left(x^4\right)$, KNN (100), KNN (10), random forest



Tradeoffs

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

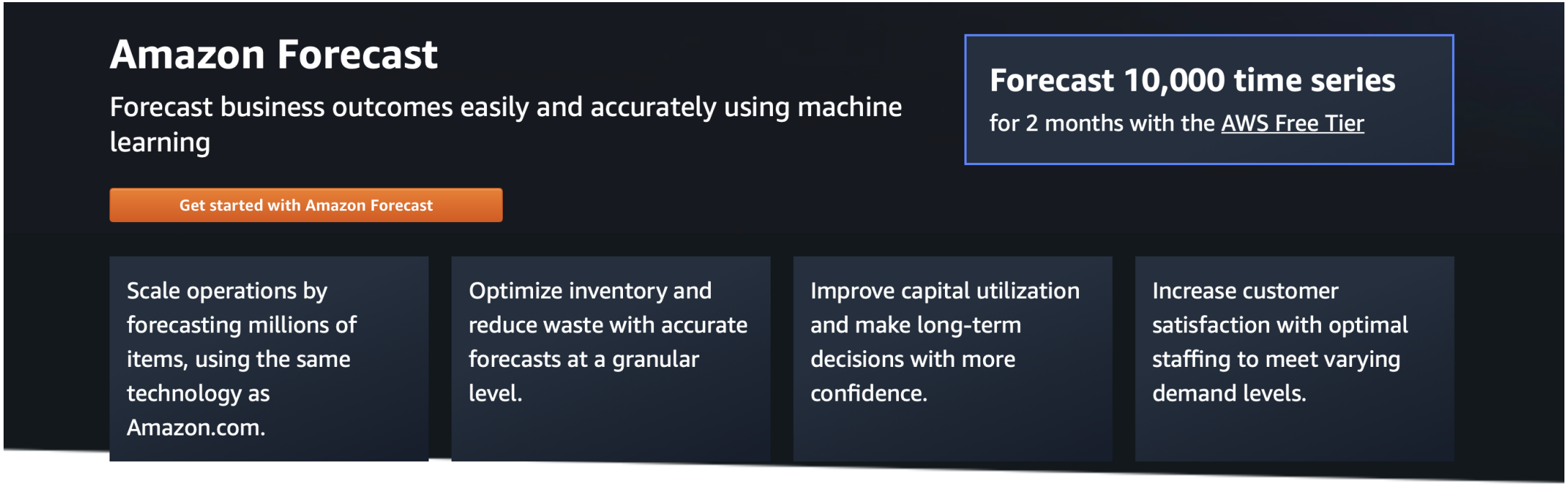
- **flexibility** and **parametric structure** (and interpretability)
- performance in **training** and **test** samples
- **variance** and **bias**

Many machine-learning (ML) techniques/algorithms optimize these tradeoffs

But it's important to understand them, and sometimes tailor to your use case

Applications: Amazon machine learning

Applications: Demand forecasting



Applications: Uber

Applications: UberEATS delivery times

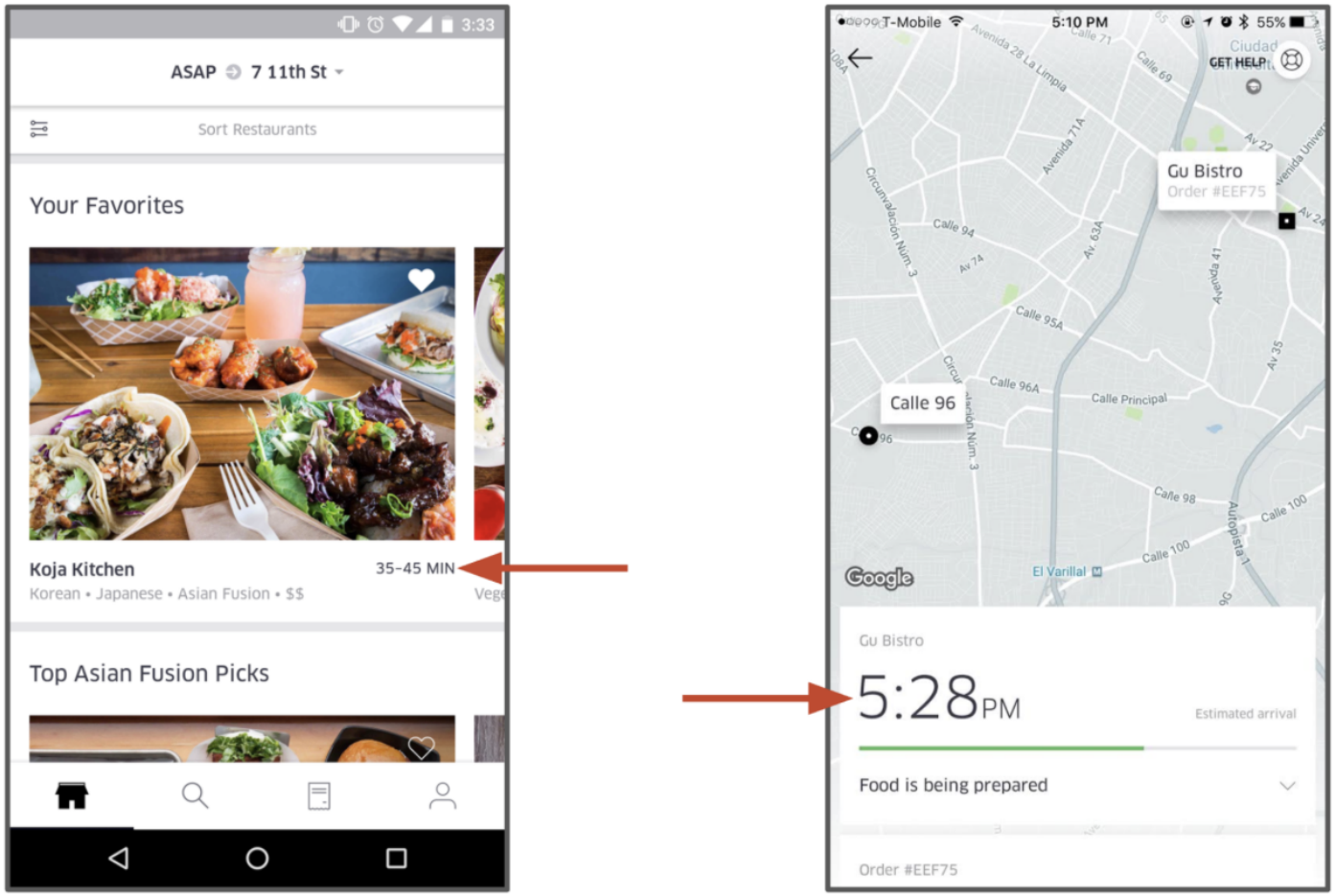
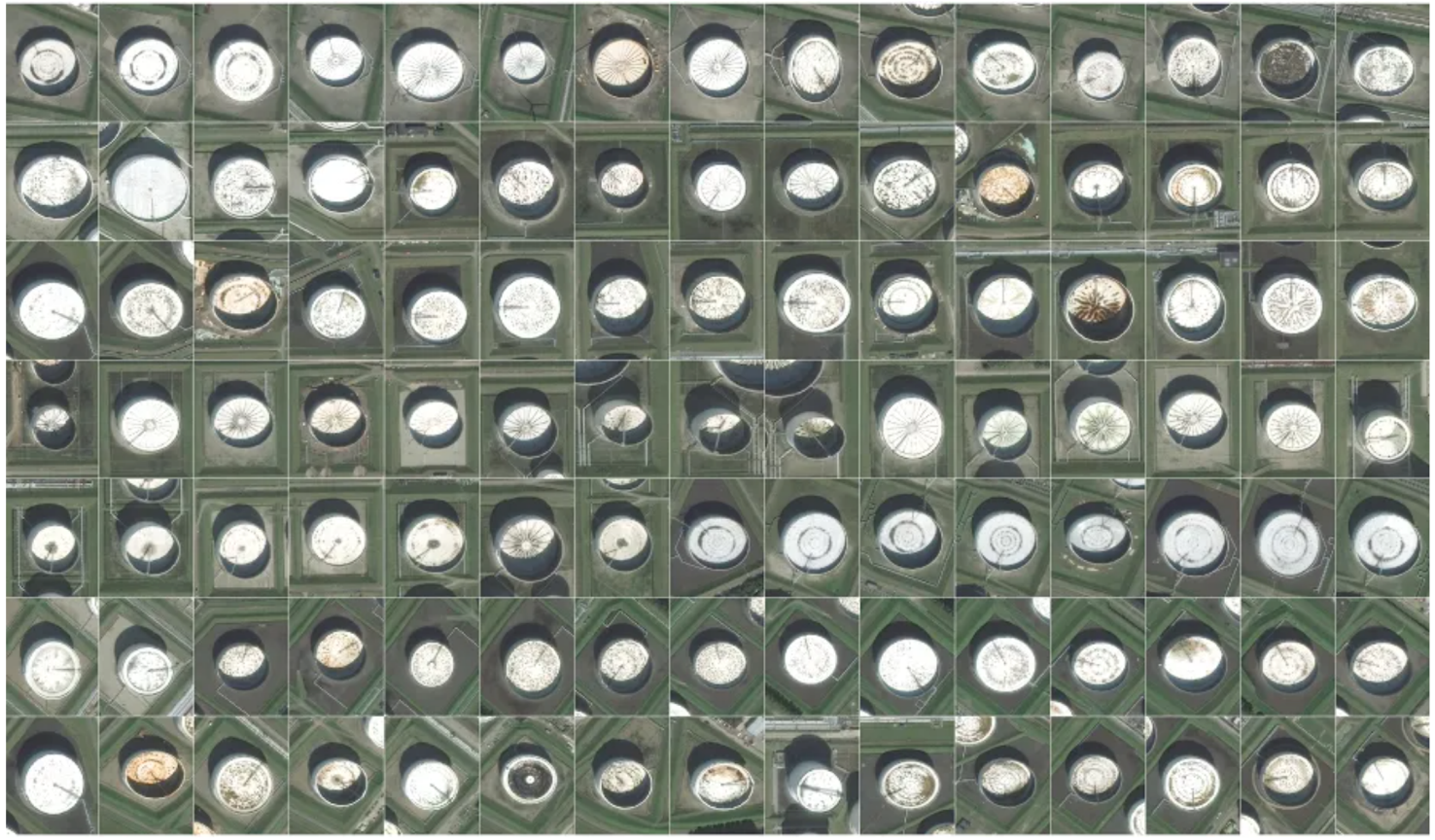


Figure 1: The UberEATS app hosts an estimated delivery time feature powered by machine learning models built on Michelangelo.

Applications: Economic activity

How AI Can Calculate Our Oil Surplus...From Space



ORBITAL INSIGHT/DIGITALGLOBE

Linear regression

Why revisit linear regression?

New perspective: **predicting** rather than estimating relationships

Many fancy statistical learning approaches are just linear regression with some tweaks

Linear regression

OLS for short

$$y_i = x_i\beta + e_i$$

OLS minimizes errors. What are errors?

$$e_i = y_i - \hat{y}_i$$

OLS minimizes errors in a specific sense. What sense?

$$\text{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2 = \sum_{i=1}^{n} e_i^2$$

$\hat{\beta}$ minimizes **sum of squared errors** (SSE), aka **residual sum of squares** (RSS)

Choosing prediction models

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + e_i$$

How to choose?

1. Define a measure of performance/fit
2. Choose the model that performs better

Sounds easy, right?

Performance

R-squared is a common way to assess fit of linear regression:

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}} \quad \text{where} \quad \text{TSS} = \sum_{i=1}^{n} \left(y_i - \bar{y}\right)^2$$

Which model will have a better R-squared? Why?

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + e_i$$

Add a new variable: RSS $\downarrow$ and TSS is unchanged. Thus, R-squared increases

Performance

Does anyone see a potential problem with this?

- What if we use n predictors for n data points?

Overfitting

Wait, what is overfitting?

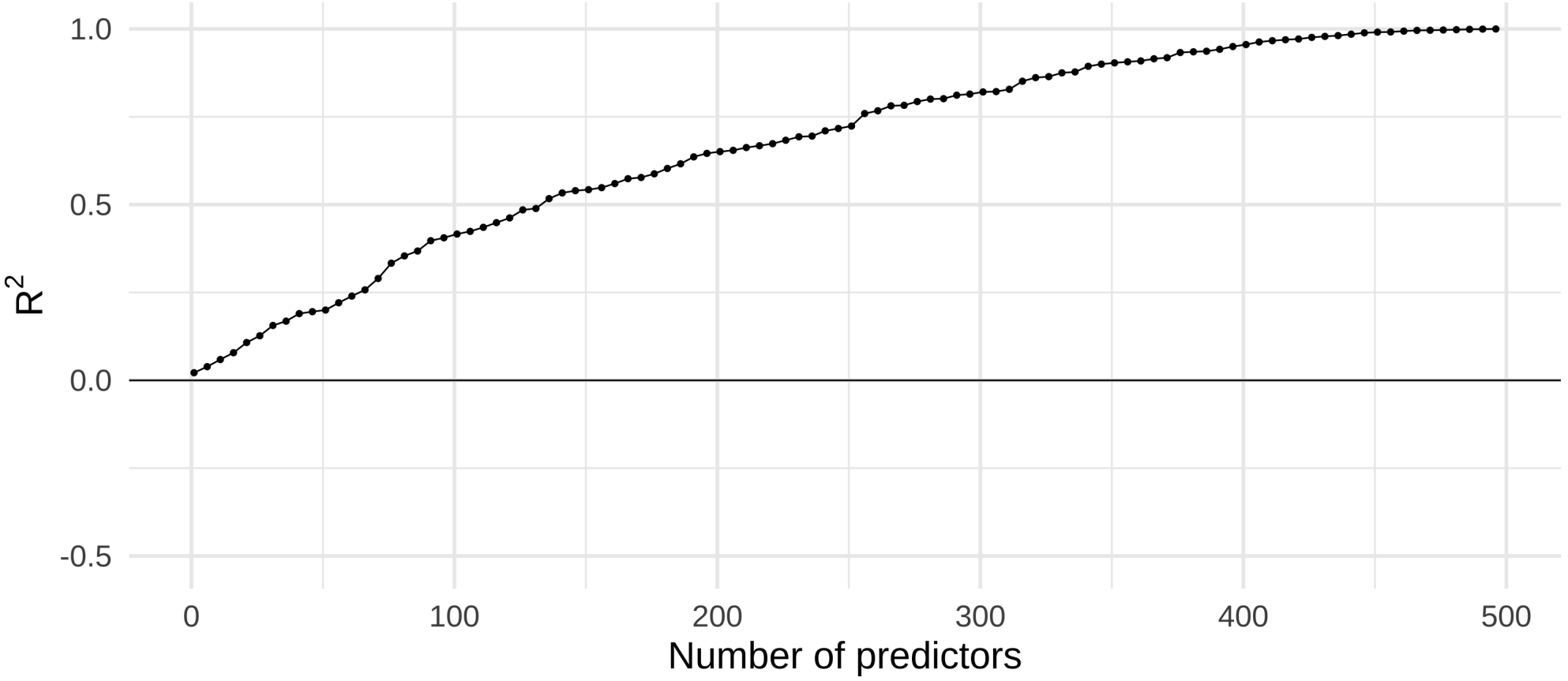
Overfitting is when a model is too good at predicting the data it was estimated on, but predictions fail on other data

Example

Let's see how R-squared does with simulated data on 500 very weak predictors

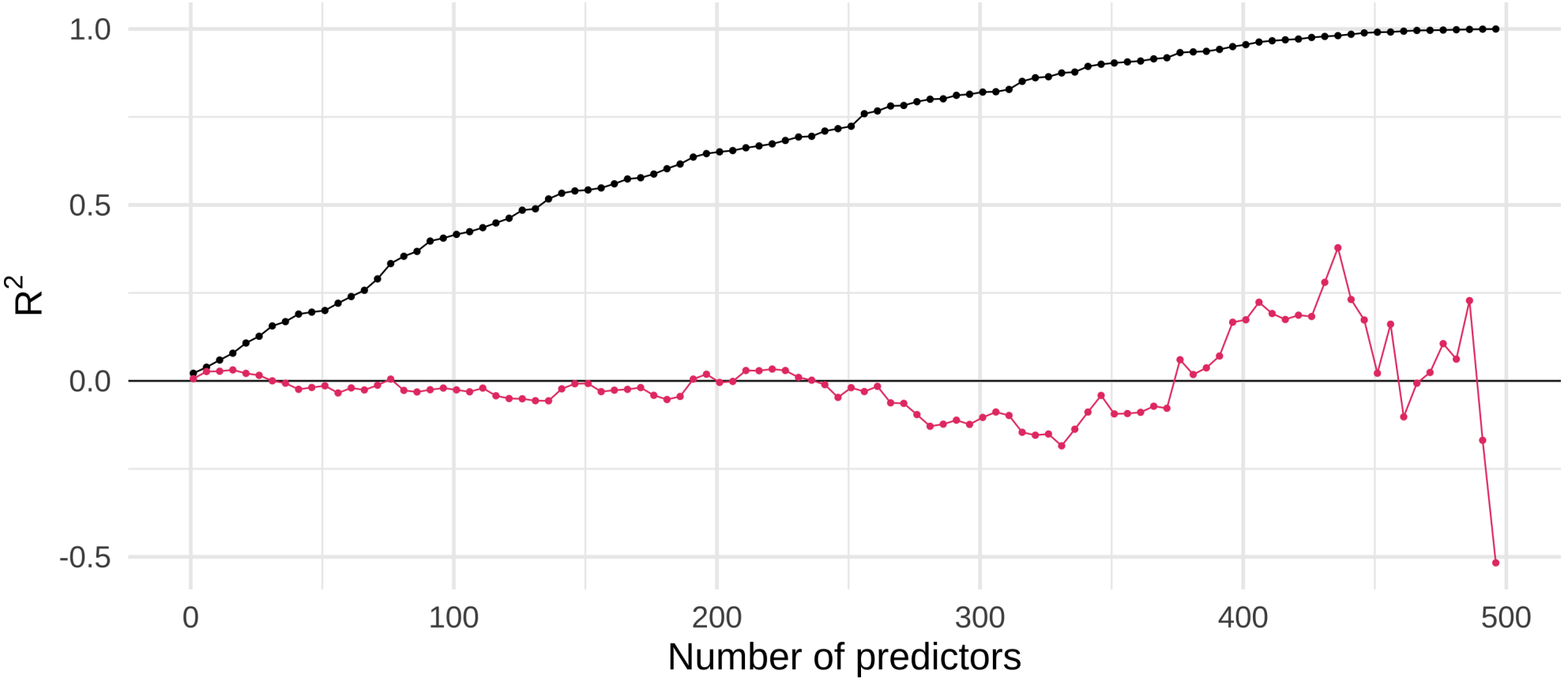
To address overfitting, we can compare **in-** vs. **out-of-sample** performance

In-sample R-squared mechanically increases as we add predictors



In-sample R-squared mechanically increases as we add predictors

Out-of-sample R-squared does not



Adjusted R-squared, residual standard error, AIC, BIC all suffer similarly

Model selection

We want a method to optimally select a (linear) model while balancing variance and bias and avoiding overfitting

Two options:

1. **Subset selection** chooses a (sub)set of p potential predictors by brute force search (and more sophisticated variants)

2. **Shrinkage** fits a model using all p variables but "shrinks" its coefficients

We will focus on shrinkage methods today

Shrinkage methods

Shrinkage methods

1. Fit a model that contains all p predictors
2. Simultaneously: shrink coefficients toward zero

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

Shrinkage methods

How could shrinking coefficients toward zero help our predictions?

Remember we face 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**

Ridge regression

Ridge regression

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: minimize the (weighted) sum of RSS plus a 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

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

$\lambda \ (\geq 0)$ 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

λ and penalization

λ determines how much ridge "cares about" RSS vs coefficients

Choosing a *good* value for λ is key.

- If λ is too small, we might as well use OLS
- If λ is too large, we might as well not have any predictors

How do we choose λ?

Cross validation

Cross validation in one slide

Cross validation is a resampling method used to test error, evaluate performance, or select a model

Basic idea:

1. split data up into two subsets
2. use one subset of data to **train** the model
3. use the other subset to **test** the model

By validating model out of sample in the test data, cross-validation prevents overfitting in the training data

Example data: Credit card balance data

Let's explore shrinkage methods using data from *ISL*'s R package `ISLR`

ID	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	Balance
1	14.9	3606	283	2	34	11	Male	No	Yes	Caucasian	333
2	106.0	6645	483	3	82	15	Female	Yes	Yes	Asian	903
3	104.6	7075	514	4	71	11	Male	No	No	Asian	580
4	148.9	9504	681	3	36	11	Female	No	No	Asian	964
5	55.9	4897	357	2	68	16	Male	No	Yes	Caucasian	331
6	80.2	8047	569	4	77	10	Male	No	No	Caucasian	1151
7	21.0	3388	259	2	37	12	Female	No	No	African American	203

The `Credit` dataset has 400 observations on 11 predictors

Ridge regression with glmnet

For ridge and lasso regression, we will use `glmnet::glmnet()`

The **key arguments** for `glmnet()` are

- `x` a matrix of predictors
- `y` outcome variable as a vector
- `standardize` (`T` or `F`)
- `alpha` elasticnet parameter
  - **`alpha=0` gives ridge**
  - `alpha=1` gives lasso
- `lambda` tuning parameter (sequence of numbers)
- `nlambda` alternatively, R picks a sequence of values for λ

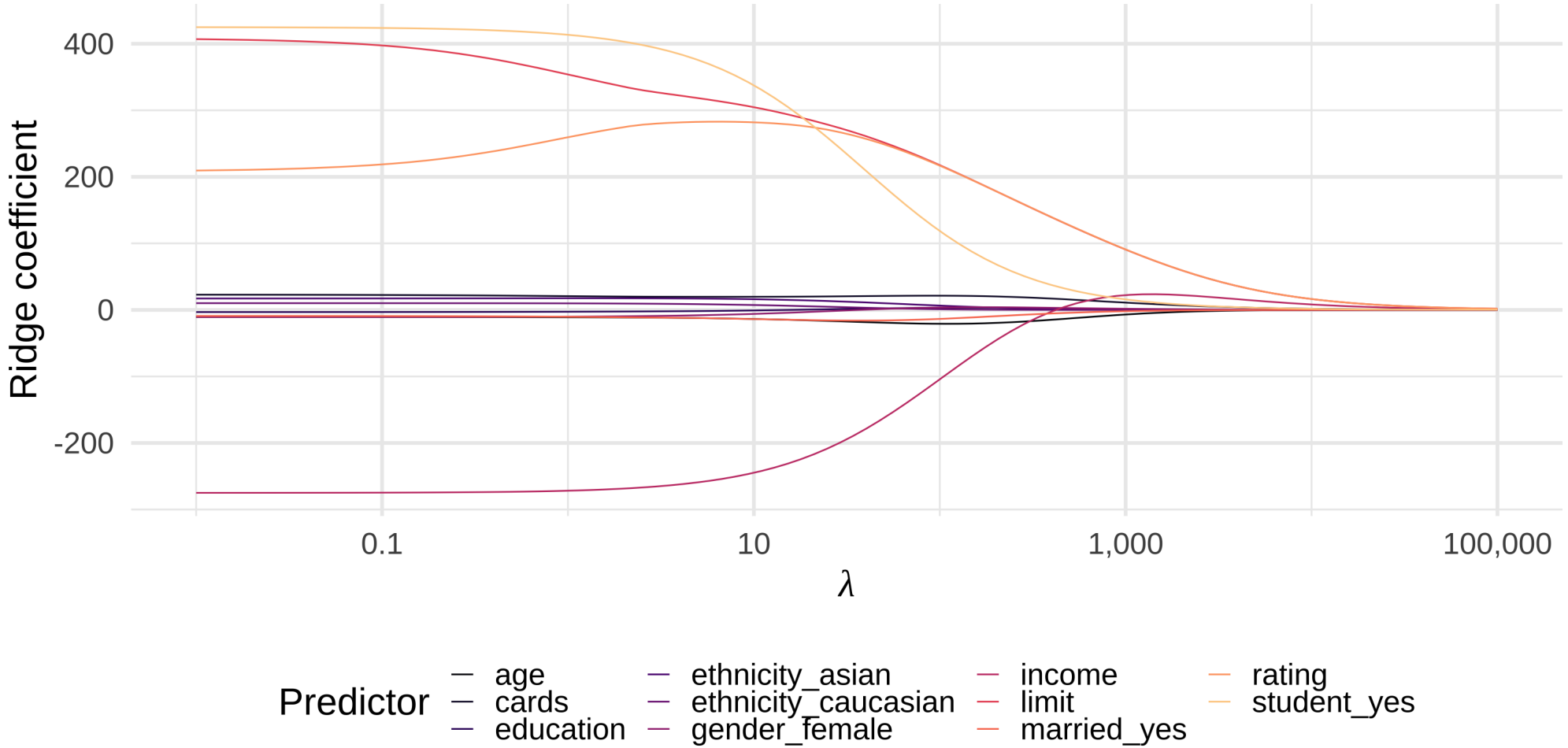
Ridge regression with glmnet

We just need to define a decreasing sequence for λ, and then we're set

```
# define our range of lambdas (glmnet wants decreasing range)
lambdas <- 10^seq(from = 5, to = -2, length = 100)
# fit ridge regression
est_ridge <- glmnet(
  x = credit_clean %>% dplyr::select(-balance, -id) %>% as.matrix(),
  y = credit_clean$balance,
  standardize = F, # data are pre-standardized
  alpha = 0,       # ridge regression
  lambda = lambdas
  )
```

The `glmnet` output contains estimated coefficients for each λ

Ridge regression coefficents



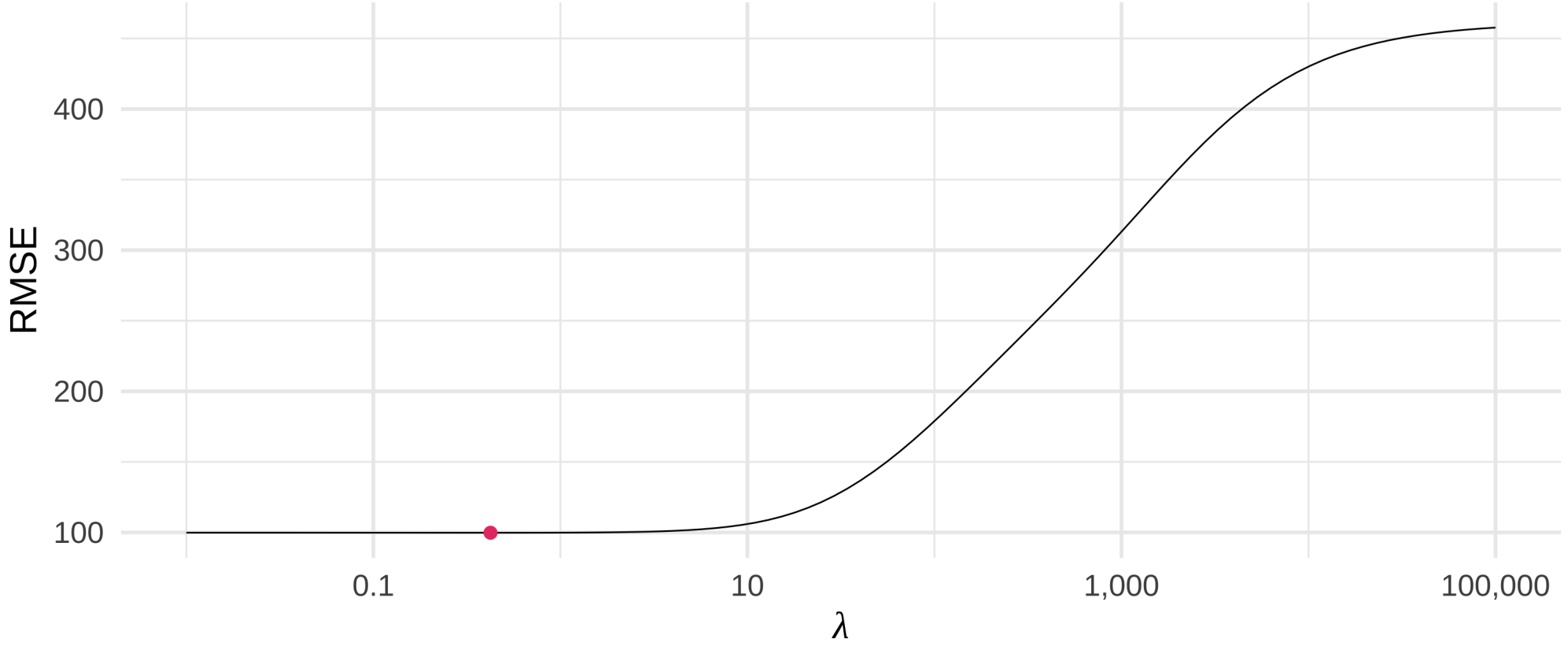
Ridge regression with cross-validation

`glmnet` provides a convenient cross-validation function: `cv.glmnet()`

```
# define our lambdas
lambdas <- 10^seq(from = 5, to = -2, length = 100)
# cross validation
ridge_cv <- cv.glmnet(
  x = credit_clean %>% dplyr::select(-balance, -id) %>% as.matrix(),
  y = credit_clean$balance,
  standardize = F, # data are pre-standardized
  alpha = 0,       # ridge regression
  lambda = lambdas,
  # new: how we make decisions and number of folds
  type.measure = "mse",
  nfolds = 5 # number of folds for k-fold cv
  )
```

Cross-validated RMSE and λ

Which λ minimizes cross-validated RMSE?

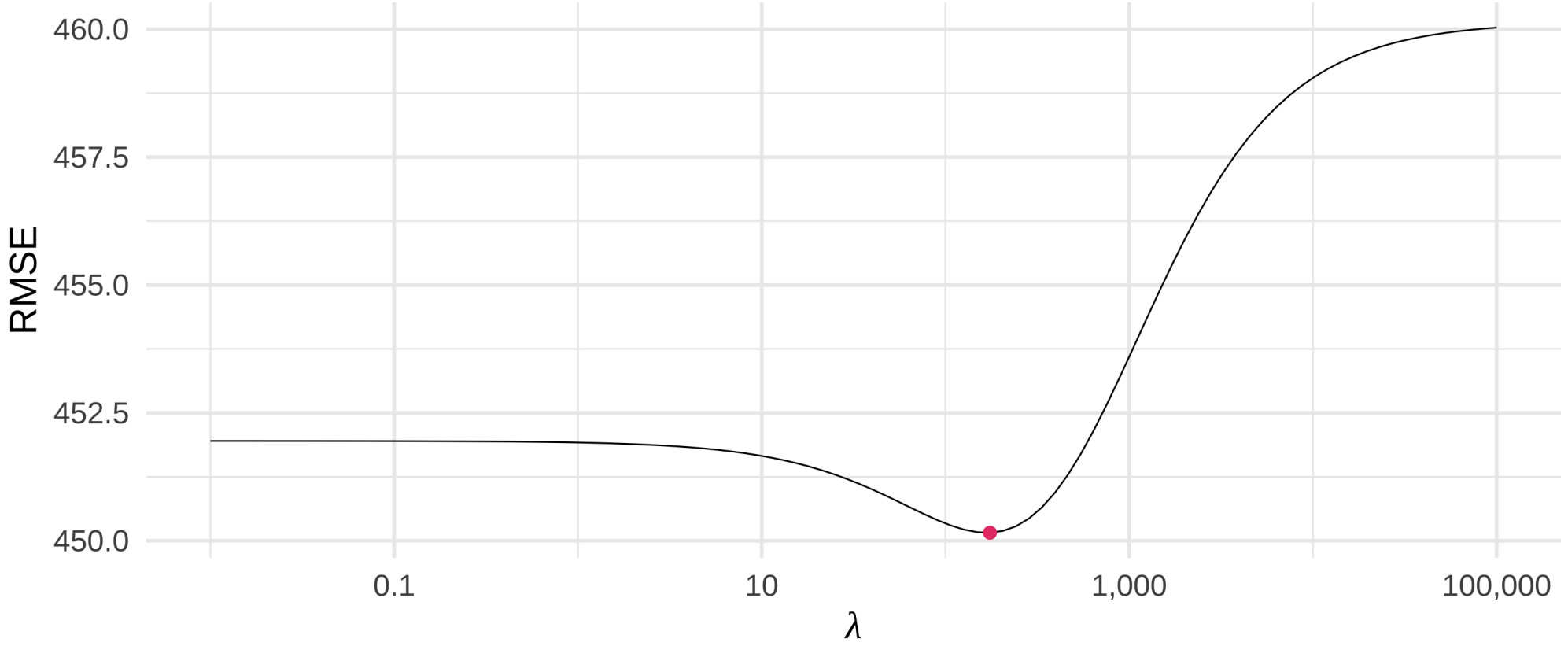


A model with fewer, weaker predictors

```
# Define our lambdas
lambdas <- 10^seq(from = 5, to = -2, length = 100)
# Cross validation
ridge_cv2 <- cv.glmnet(
  x = credit_clean %>% dplyr::select(-balance, -rating, -limit, -income) %>% as.matrix(),
  y = credit_clean$balance,
  alpha = 0,
  standardize = T,
  lambda = lambdas,
  # how we make decisions and number of folds
  type.measure = "mse",
  nfolds = 5
)
```

Cross-validated RMSE and λ

How does RMSE compare? How does optimal λ compare? Why?



Ridge regression predictions in R

Once you find λ via cross validation,

1. Fit your model on the full dataset using the optimal λ

```
# fit final model
final_ridge <- glmnet(
  x = credit_clean %>% dplyr::select(-balance, -id) %>% as.matrix(),
  y = credit_clean$balance,
  standardize = T,
  alpha = 0,
  lambda = ridge_cv$lambda.min # optimal lambda $<<
)
```

Ridge regression predictions in R

Once you find λ via cross validation,

1. Fit your model on the full dataset using the optimal λ

2. Make predictions

```
predict(
  final_ridge, # our final model
  type = "response", # predict y
  s = ridge_cv$lambda.min, # optimal lambda
  # x's to use for predicting y:
  newx = credit_clean %>% dplyr::select(-balance, -id) %>% as.matrix()
)
```

Lasso

Lasso

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

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

Shrinkage with lasso

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

You always pay λ to increase $\left|\beta_j\right|$ 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 still need to select λ carefully

Lasso with glmnet

We can also use `glmnet()` for lasso.

Recall The key arguments for `glmnet()` are

- `x` a matrix of predictors
- `y` outcome variable as a vector
- `standardize` (`T` or `F`)
- `alpha` elasticnet parameter
  - `alpha=0` gives ridge
  - **`alpha=1` gives lasso**
- `lambda` tuning parameter (sequence of numbers)
- `nlambda` alternatively, R picks a sequence of values for λ

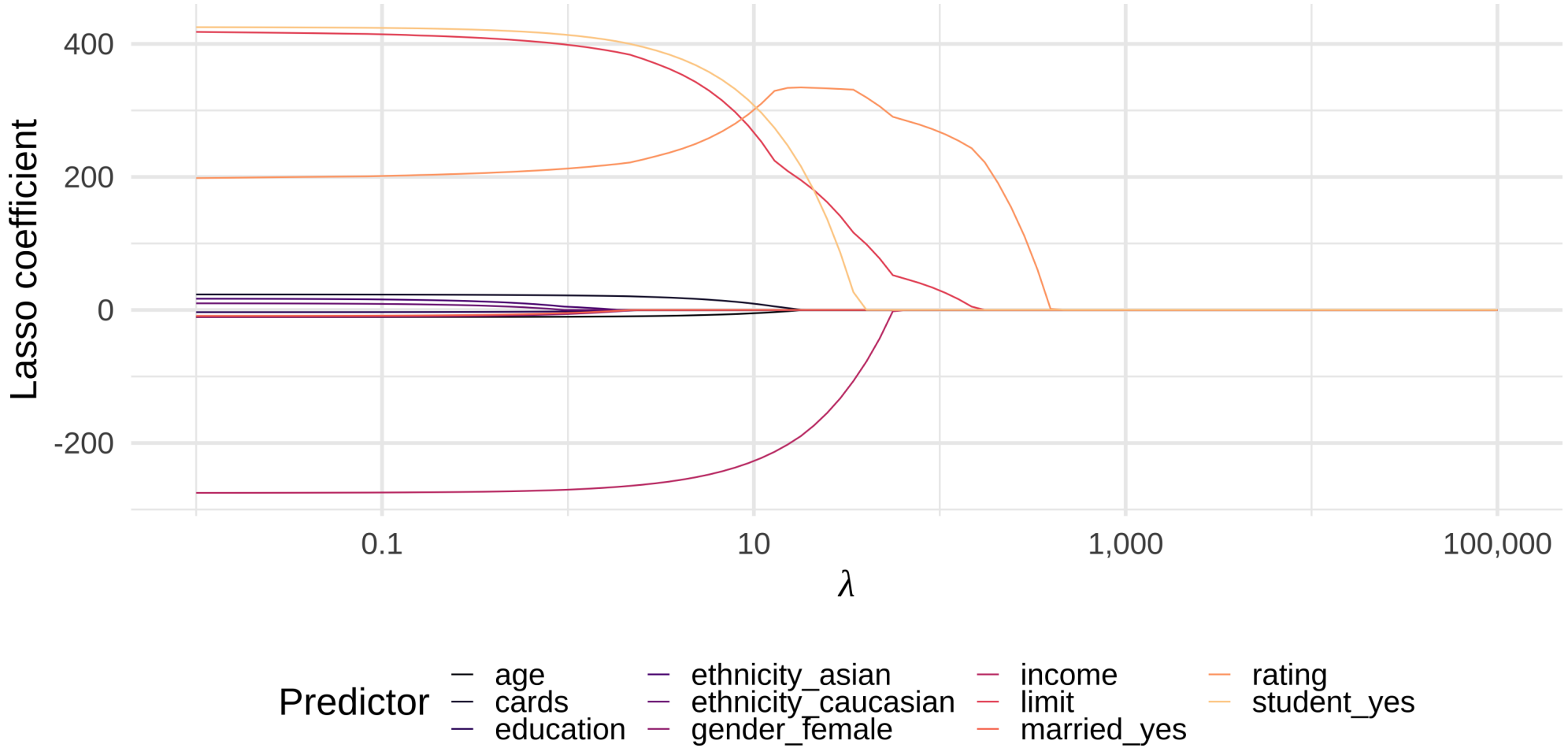
Lasso with glmnet

Again, we define a decreasing sequence for λ, and we're set

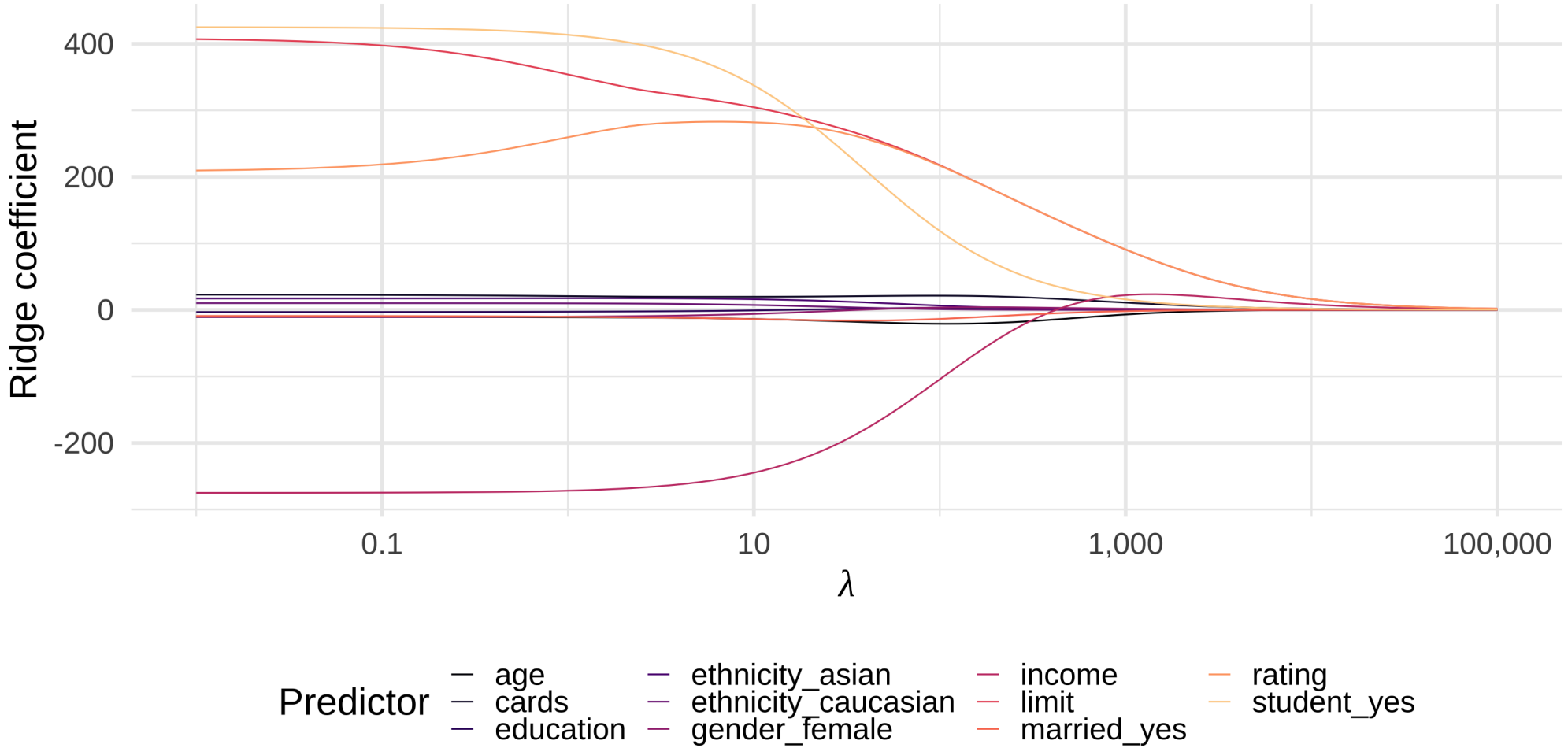
```
# define our range of lambdas (glmnet wants decreasing range)
lambdas <- 10^seq(from = 5, to = -2, length = 100)
# fit lasso regression
est_lasso <- glmnet(
  x = credit_clean %>% dplyr::select(-balance, -id) %>% as.matrix(),
  y = credit_clean$balance,
  standardize = F,
  alpha = 1,
  lambda = lambdas
  )
```

The `glmnet` output (`est_lasso` here) contains estimated coefficients for λ. You can use `predict()` to get coefficients for additional values of λ.

Lasso coefficents



Before: Ridge regression coefficents



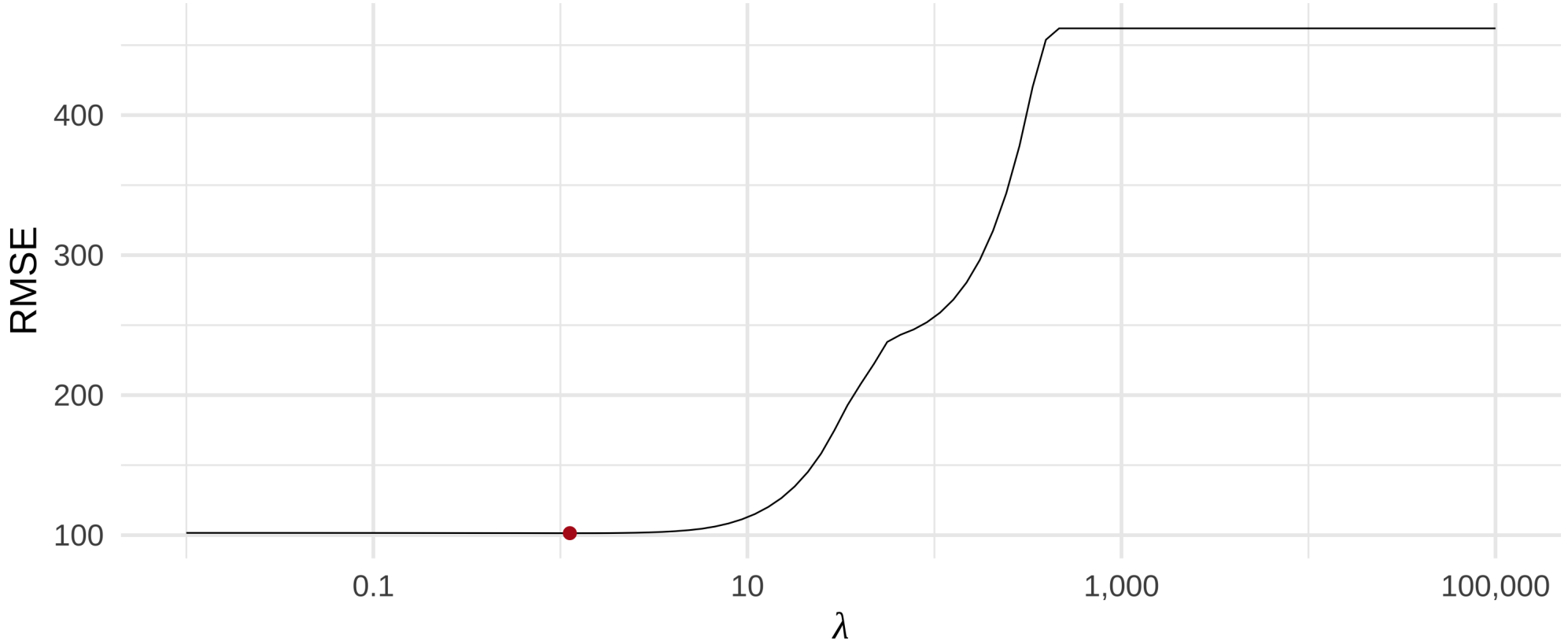
Lasso with cross-validation

We can also cross validate λ with `cv.glmnet()`

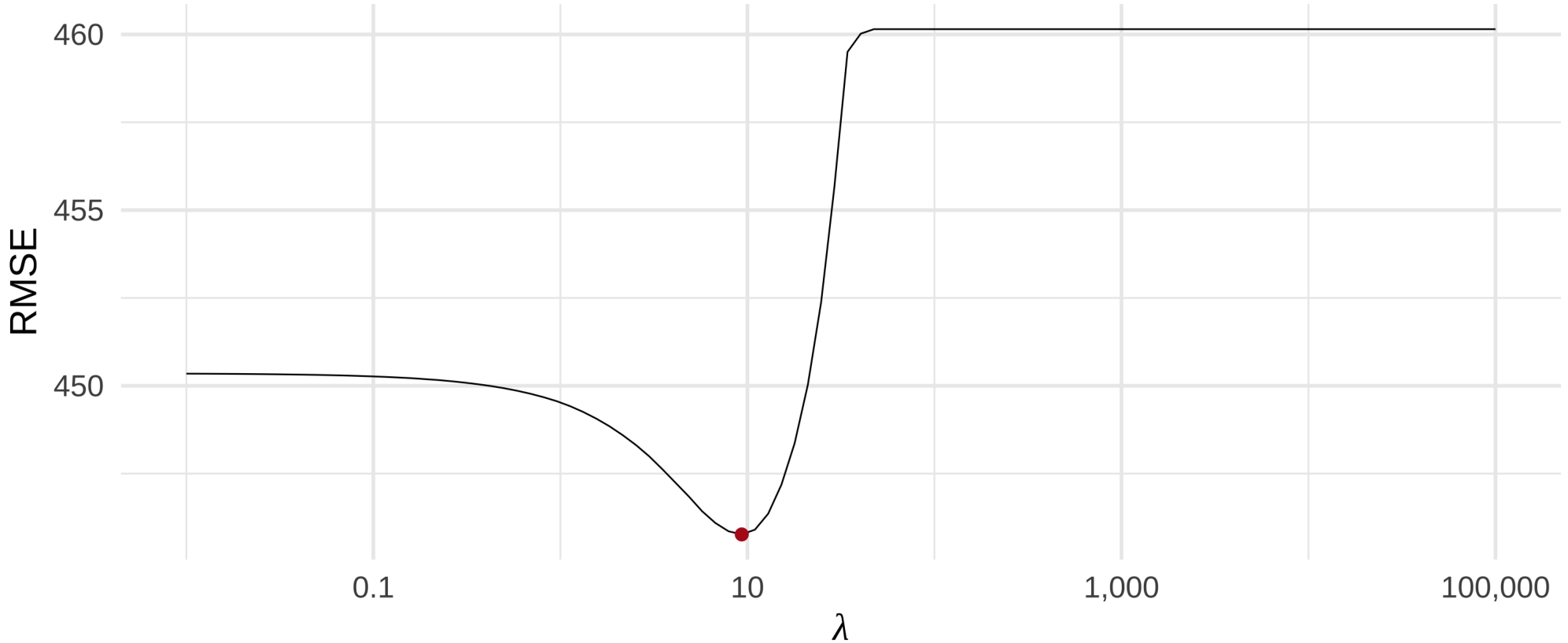
```
# Define our lambdas
lambdas = 10^seq(from = 5, to = -2, length = 100)
# Cross validation
lasso_cv = cv.glmnet(
  x = credit_clean %>% dplyr::select(-balance, -id) %>% as.matrix(),
  y = credit_clean$balance,
  alpha = 1,
  standardize = F,
  lambda = lambdas,
  type.measure = "mse",
  nfolds = 5
)
```

Cross-validated RMSE and λ

Which λ minimizes cross-validated RMSE?



A model with fewer, weaker predictors



How does RMSE compare? How does optimal λ compare? Why?

Ridge or lasso?

Ridge regression

+ shrinks $\hat{\beta}_j$ *near* 0
- many small $\hat{\beta}_j$
- doesn't work for selection
- difficult to interpret output
+ better when all $\beta_j \neq 0$

Best: p is large & $\beta_j \approx \beta_k$

Lasso

+ shrinks $\hat{\beta}_j$ *to* 0
+ many $\hat{\beta}_j = 0$
+ great for selection
+ sparse models easier to interpret
- implicitly assumes some $\beta = 0$

Best: p is large & many $\beta_j \approx 0$

> [N]either ridge... nor the lasso will universally dominate the other.

ISL, p. 224

Why not both?

Elasticnet combines ridge regression and lasso.

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

We now have two tuning parameters: λ (penalty) and α (mixture).

Remember the `alpha` argument in `glmnet()`?

- $\alpha = 0$ specifies ridge
- $\alpha = 1$ specifies lasso

Classification

Classification

Thursday, time permitting!