Machine Learning in R

Lecture 3 – Model evaluation, Regularization

Jean Feng – Epidemiology and Biostatistics

Last lecture

1. Linear regression as an optimization procedure
2. Classification
    1. Loss functions
    2. Logistic Regression
    3. Multinomial Regression
    4. Linear Discriminant Analysis
    5. k-Nearest Neighbors
    6. Receiver Operating Characteristic (ROC)

Today's lecture

1. Model selection: Cross-validation
2. Low versus high-dimensional data
3. Regularization methods
    1. Ridge Regression
    2. Variable selection methods
        1. Variable subset selection
        2. Lasso
    3. Extensions

Today's lecture

1. **Model selection: Cross-validation**
2. Low versus high-dimensional data
3. Regularization methods
    1. Ridge Regression
    2. Variable selection methods
        1. Variable subset selection
        2. Lasso
4. Extensions

Model selection

- So far, we've seen a lot of different methods with different degrees of model complexity:
  - Linear regression
  - Polynomial regression

  } **Parametric models**
  Low model complexity

  - Kernel smoothing
  - K-nearest neighbors

  } **Nonparametric models**
  High model complexity

Which model do we pick?

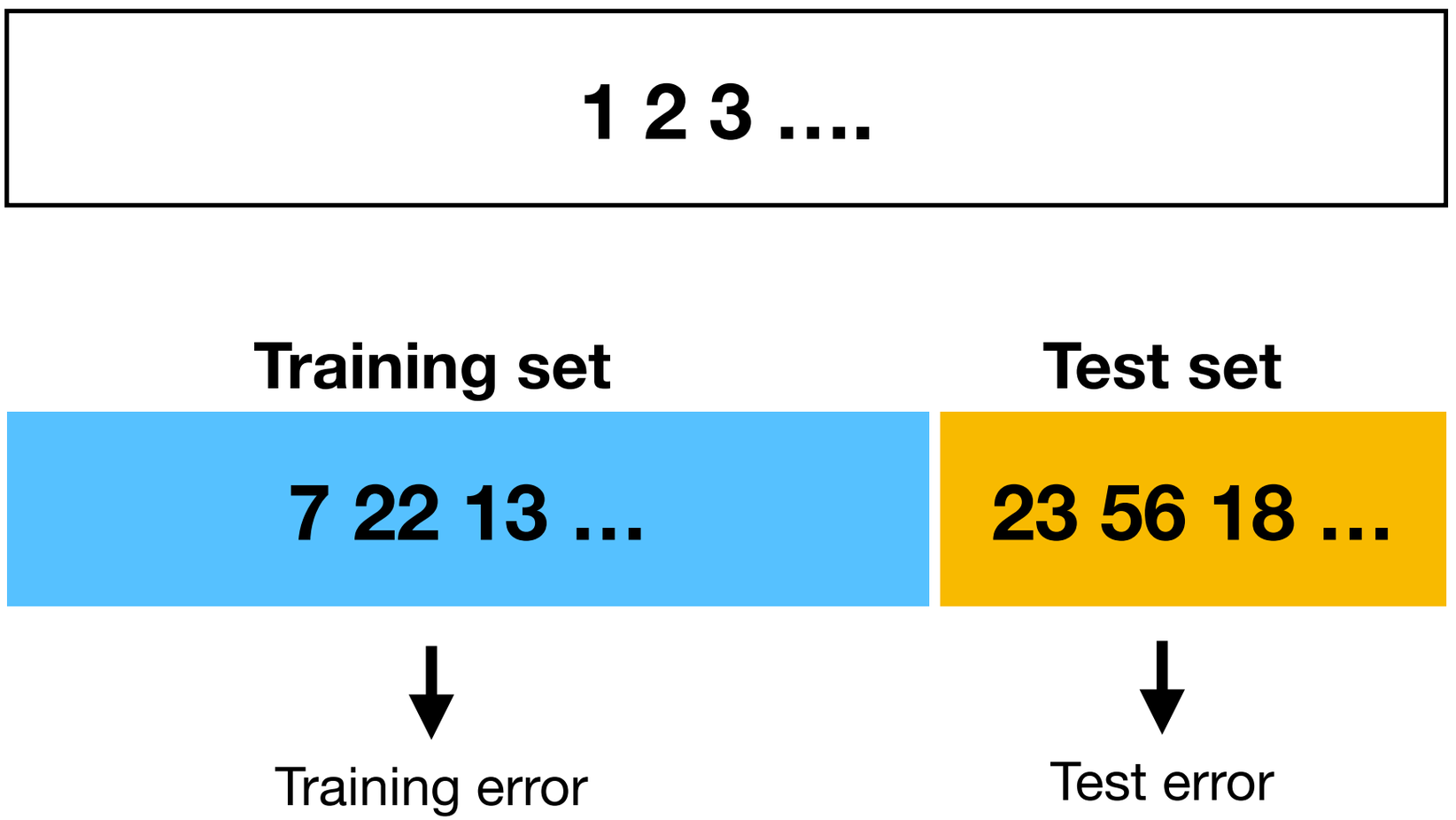
Tuning hyperparameters

- We've also seen many methods that require choosing a hyperparameter:
  - Kernel smoothing
    - Large bandwidth **Low model complexity**
    - Small bandwidth **High model complexity**
  - K-nearest neighbors
    - Large K **Low model complexity**
    - Small K **High model complexity**

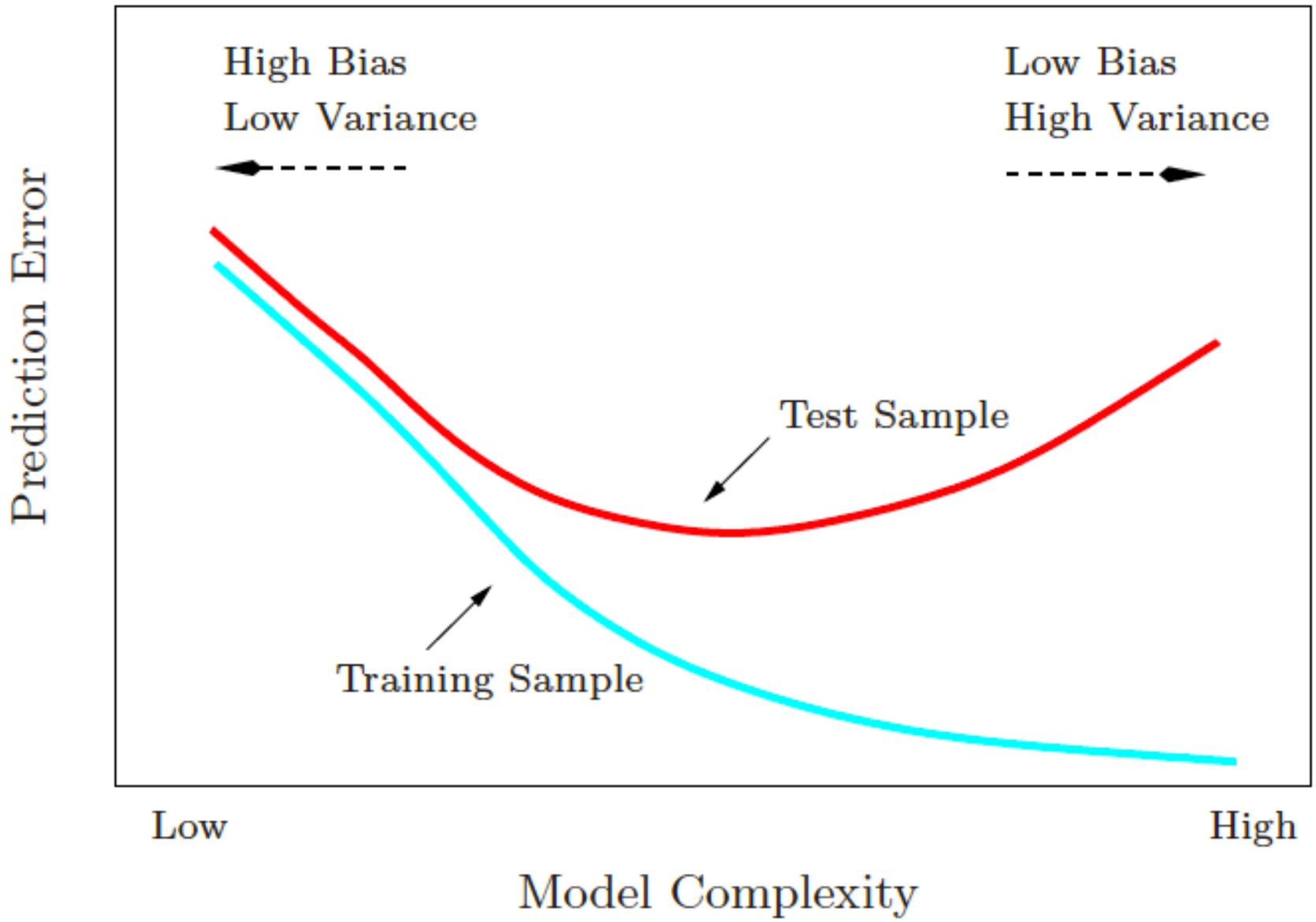
How do we select the value of the hyperparameter?

Training-test split

- Up to now, we've talked about doing model selection and hyperparameter tuning using a training-test split.



Bias-Variance Tradeoff



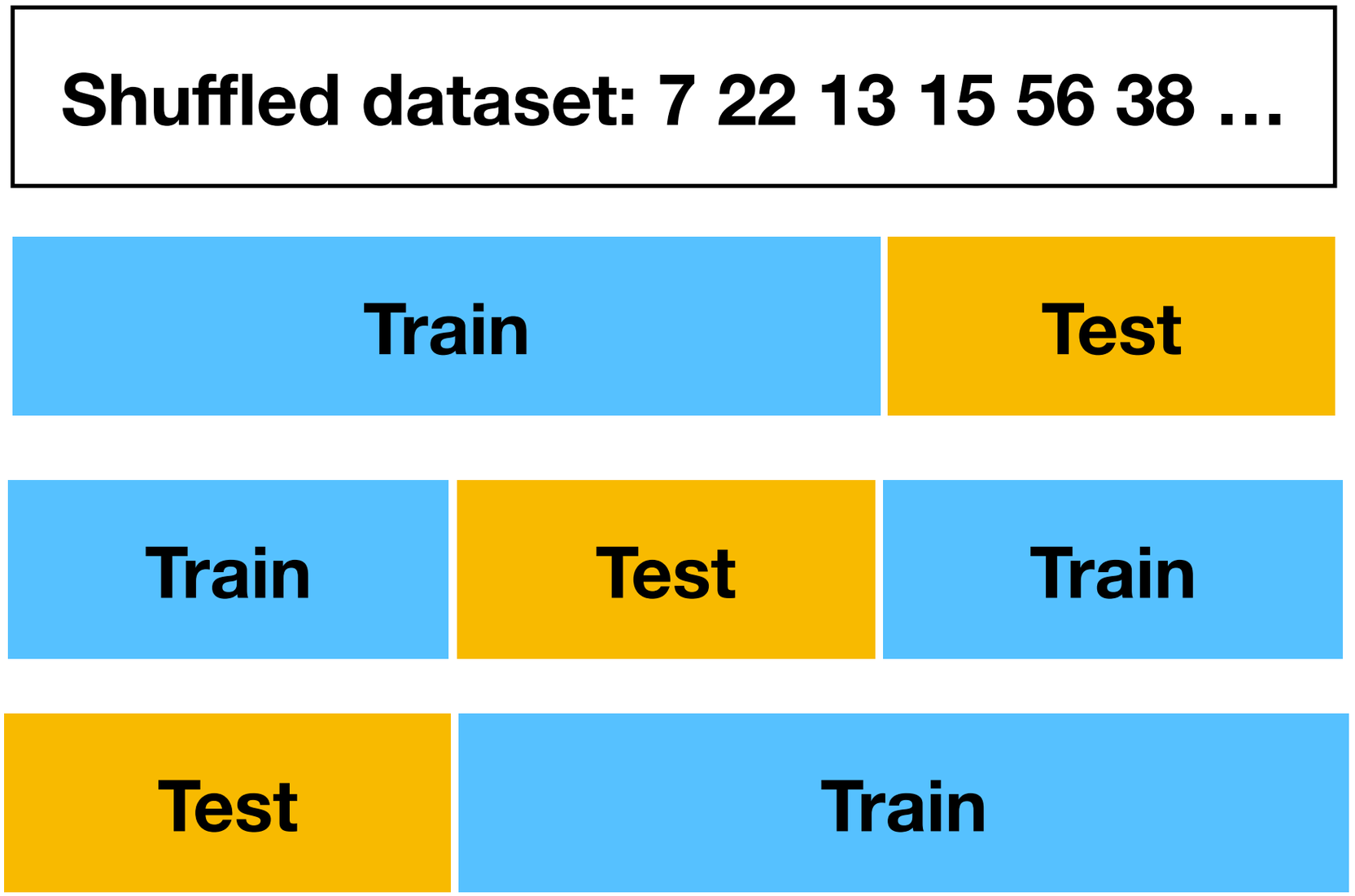
Drawbacks of training-test split

- The test error calculated using a training-test split can be noisy because it only performs a single split.

- It may not be representative of the target population by chance. This is particularly true if our dataset is small to begin with.

Can we use all the data somehow to do model selection?

3-Fold Cross-validation



The training and testing data are always disjoint!

K-Fold Cross-validation (CV)

1. Randomly split n observations into K equally-sized folds.

2. For $k = 1,...,K$:

    A. Fit the model using observations not in the kth fold.

    B. Calculate the test error e_k of this model on the kth fold.

3. Calculate $\frac{1}{K}\sum_{k=1}^{K} e_k$ the average cross-validated (CV) error.

4. Pick the model with the smallest CV error.

You picked your model. Now what?

- After you pick the model by training-test split or cross-validation, refit the selected model with the selected hyperparameters on all the data.

- Example:

1. Decide the best bandwidth for kernel smoothing using 3-Fold cross-validation. Suppose we find that bandwidth=10 has the smallest average CV error.

2. For our final model, perform kernel smoothing on all the data with bandwidth=10.

Today's lecture

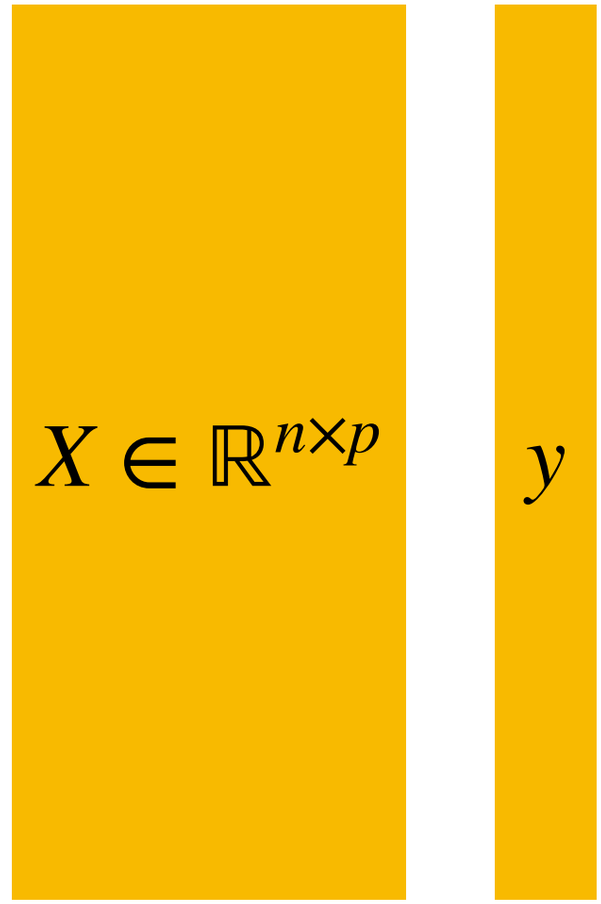
1. Model selection: Cross-validation
2. **Low versus high-dimensional data**
3. Regularization methods
    1. Ridge Regression
    2. Variable selection methods
        1. Variable subset selection
        2. Lasso
    3. Extensions

Example on the Heart dataset

- Suppose we have $n = 303$ people who we have measured three variables:
  - Age
  - Calcium levels
  - Cholesterol levels
- We wish to predict whether or not the patient has heart disease

Notation

- n is the number of observations
- p is the number of variables/ features
- y is the vector of outcomes for n observations
- X is a $n \times p$ matrix



Logistic regression in our simple example

- Classical results in statistics characterize the behavior of the estimated linear model as $n \to \infty$ and p stays fixed.

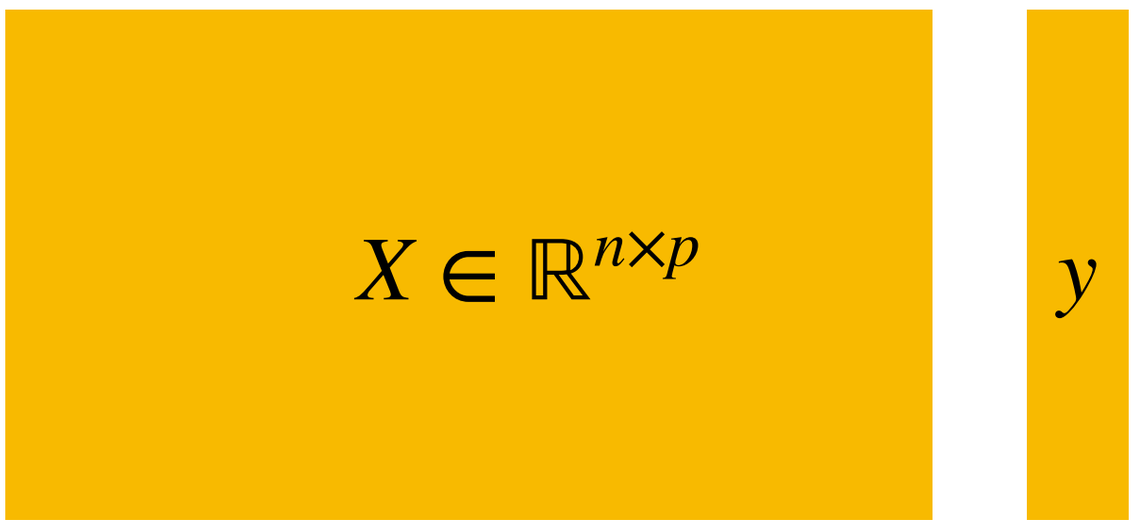
- The estimated coefficients, p-values, and t-statistics are all (approximately) valid because $n \gg p$ in our example.

```
glm(formula = AHD ~ . - AHD, family = binomial, data = heart)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.9451858  0.9722081  -2.001   0.0454 *
Age          0.0157108  0.0159460   0.985   0.3245
Ca           1.1757984  0.1846480   6.368 1.92e-10 ***
Chol         0.0008275  0.0025729   0.322   0.7477
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

High-dimensional data

- However, lots of datasets generated by modern biotechnologies generate **high-dimensional data** where $n \approx p$ or $n \ll p$. For example:
  - Gene expression data: $n = 300,\ p = 20000$
  - DNA sequence data: $n = 1000,\ p = 3000000$
  - DNA hypersensitivity data: $n = 200,\ p = 1000000000$



What goes wrong in high dimensions?

- Suppose that we included many variables in our model:
  - Age
  - Zodiac symbol
  - Favorite color
  - Mother's birthday
  - …
- Just by chance, one or more of these variables will be associated with the outcome in that particular dataset.

(This is basically the same problem as multiple hypothesis testing)

- If most of the variables are useless, classical statistical methods will **overfit**.

What goes wrong in high dimensions?

- Let's add 100 measurements that are completely randomly generated.

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.9111597  1.6080831  -2.432  0.01501 *
Age         -0.0003906  0.0262035  -0.015  0.98811
Ca           1.7470575  0.3220096   5.425 5.78e-08 ***
Chol         0.0087580  0.0042463   2.063  0.03916 *
`1`         -0.0270659  0.2352734  -0.115  0.90841
`2`          0.0849529  0.2279971   0.373  0.70944

`64`        -0.6776496  0.2342924  -2.892
`65`        -0.6792382  0.2428280  -2.797
`66`         0.1005103  0.2183149   0.460
`67`        -0.4714011  0.2500740  -1.885
`68`         0.5500208  0.2679982   2.052
`69`         0.0365903  0.2331883   0.157
`70`        -0.0098266  0.2089813  -0.047  0.96250
`71`         0.0538026  0.2281101   0.236  0.81354
`72`        -0.1605246  0.2409415  -0.666  0.50526
`73`        -0.5100719  0.2402887  -2.123  0.03378 *
`74`         0.7216039  0.2657294   2.716  0.00662 **
```

Oh dear!

```
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.9451858  0.9722081  -2.001   0.0454
Age          0.0157108  0.0159460   0.985   0.3245
Ca           1.1757984  0.1846480   6.368 1.92e-10
Chol         0.0008275  0.0025729   0.322   0.7477
```

What goes wrong in high dimensions?

- Let's add 200 measurements that are completely randomly generated.

```
glm(formula = AHD ~ . - AHD, family = binomial, data = heart_big_x)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-1.082e-05  -3.730e-06  -2.110e-08   2.847e-06   1.133e-05

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.172e+01  2.000e+06       0        1
Age         -9.621e-01  1.659e+04       0        1
Ca           3.243e+01  1.890e+05       0        1
Chol        -1.194e-01  7.446e+03       0        1
`1`          3.207e+00  3.013e+05       0        1
`2`         -8.298e+00  2.500e+05       0        1
`3`         -1.673e+01  1.479e+05       0        1
`4`          1.881e+00  2.598e+05       0        1
`5`          2.898e+00  1.956e+05       0        1
`6`          3.430e+00  3.083e+05       0        1
`7`          2.516e+00  4.082e+05       0        1
```

Oh my!

What goes wrong in high dimensions?

- Crazy coefficients
- The model fitting program might just crash
- The p-values and confidence intervals aren’t meaningful anymore
- Overfitting
- …

Wait, but shouldn't more variables be better?

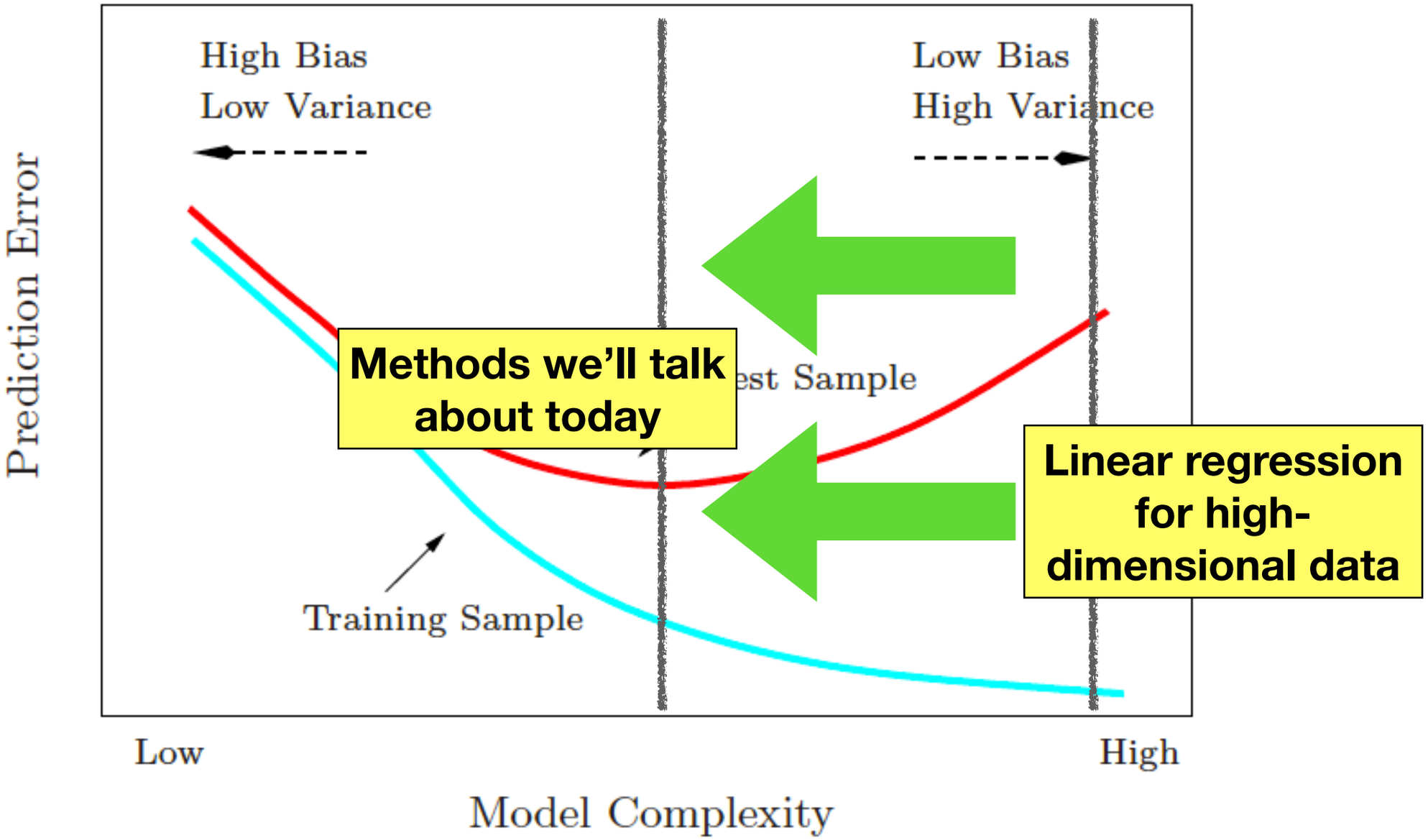
- **Q**: A dataset with more variables is better than a dataset with fewer variables, right?

- **A**: You don't necessarily want them all in your model!

- **Noise variables** (e.g. genes whose expression levels are not truly associated with the outcome) will simply increase the risk of overfitting, and increase the difficulty of developing an effective model that performs well on future observations.

- **Signal variables** (i.e. variables truly associated with the response) are always useful!

Bias-Variance Tradeoff in high-dimensions



Today's lecture

1. Model selection: Cross-validation
2. Low versus high-dimensional data
3. **Regularization methods**
    1. Ridge Regression
    2. Variable selection methods
        1. Variable subset selection
        2. Lasso
4. Extensions

Regularization methods

- Very broadly speaking, **regularization** is the process of **adding prior information** to improve the generalizability of the fitted model.

- There are many ways to perform regularization:
  - Adding a function to penalize high complexity models
  - Bayesian methods that place a prior on the model parameters
  - Early stopping in deep learning
  - …

We'll focus on this today

Linear regression + regularization

What prior knowledge can we add that can help us better estimate the model?

- Most variables have a small effect on the outcome. Shrink coefficients toward zero. **Ridge regression**

- Most variables have no effect on the outcome. Set most coefficients to zero. **Variable selection methods, Lasso**

Today's lecture

1. Model selection: Cross-validation
2. Low versus high-dimensional data
3. Regularization methods
    1. **Ridge Regression**
    2. Variable selection methods
        1. Variable subset selection
        2. Lasso
    3. Extensions

Ridge regression

Recall: Ordinary least squares fits a model by solving the optimization problem

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \left(y_i - (x_i^\top \beta + \beta_0) \right)^2}_{\text{Mean squared error}}$$

Ridge regression fits a model by adding a ridge penalty to the objective function:

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \left(y_i - (x_i^\top \beta + \beta_0) \right)^2}_{\text{Mean squared error}} + \lambda \underbrace{\sum_{j=1}^{p} \beta_j^2}_{\textbf{Ridge penalty}}$$

Ridge regression

Recall: Ordinary least squares fits a model by solving the optimization problem

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \left(y_i - (x_i^\top \beta + \beta_0)\right)^2}_{\text{Mean squared error}}$$

Ridge regression fits a model by adding a ridge penalty to the objective function:

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \left(y_i - (x_i^\top \beta + \beta_0)\right)^2}_{\text{Mean squared error}} + \lambda \underbrace{\|\beta\|_2^2}_{\substack{\textbf{Ridge} \\ \textbf{penalty}}}$$

Ridge penalty parameter

$$\hat{\beta}_\lambda = \underset{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - (x_i^\top \beta + \beta_0) \right)^2 + \lambda \|\beta\|_2^2$$

Here $\lambda \geq 0$ is a penalty parameter that controls the strength of the ridge penalty:

λ is a hyperparameter!

- When $\lambda = 0$, ridge regression is exactly the same as ordinary least squares.

- When $\lambda = \infty$, all the model coefficients are exactly zero.

- When λ is in between, we balance the priorities between two tasks:
  A. Find a model with a small mean squared error
  B. Find a model with small coefficients.

Bias-**variance** tradeoff in ridge regression

- Different values of λ correspond to different bias-variance trade-offs.

$$\|\hat{\beta}_\lambda - \beta^*\|_2^2 = \sum_{j=1}^{p} Var\left(\hat{\beta}_{\lambda,j}\right) + \sum_{j=1}^{p} Bias\left(\hat{\beta}_{\lambda,j}\right)^2$$

Large variance (possibly infinity) — **Zero variance**

Zero bias — **Big bias**

$\lambda = 0$ ***What is the right value for λ?*** $\lambda = \infty$

Question

- How are you going to select λ?

A. Whichever one has the smallest training error

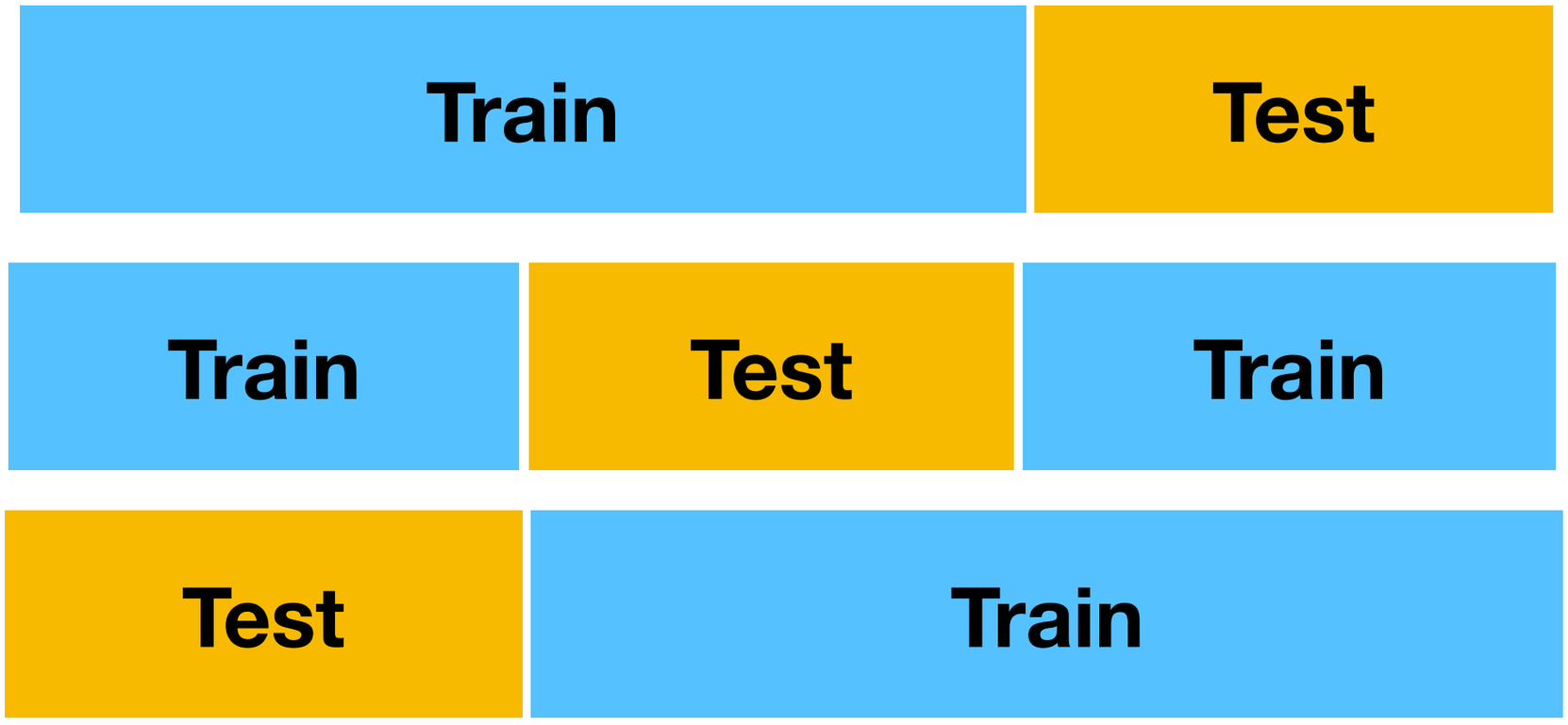
B. I'll decide on its value before looking at the data

C. Cross-validation

D. Training/validation split

Example: Tuning λ by 3-fold CV

Shuffled dataset: 7 22 13 15 56 38 ...



	Error $\lambda = 1$	Error $\lambda = 10$
Fold 1	0.8	0.9
Fold 2	0.75	0.8
Fold 3	0.85	0.85
Average	0.75	0.85

Choose $\lambda = 1$ because its CV error is smaller.

Logistic regression + Ridge penalty

- If you have a classification problem at hand, you can do the same thing.

Logistic regression:

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \ell \left(y_i, f_{\beta_0, \beta}(x_i) \right)}_{\text{Logistic loss}}$$

Logistic regression + **Ridge**:

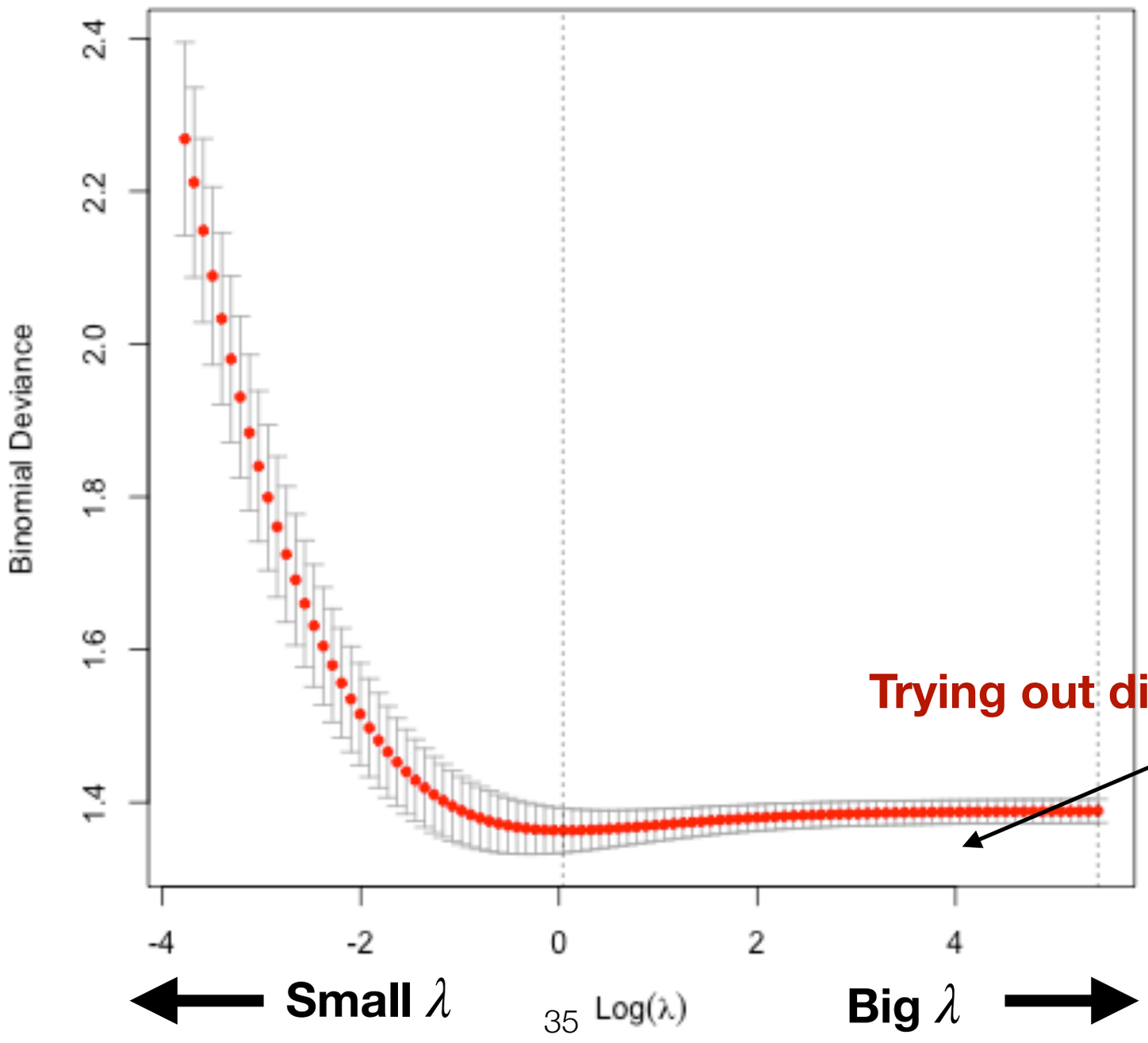
$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \ell \left(y_i, f_{\beta_0, \beta}(x_i) \right)}_{\text{Logistic loss}} + \lambda \underbrace{\|\beta\|_2^2}_{\substack{\textbf{Ridge} \\ \textbf{penalty}}}$$

Let's try out ridge on our Heart data

```
glm_ridge_model <- cv.glmnet(my_x, heart_big_x$AHD, family="binomial", alpha=0)
plot(glm_ridge_model)
```

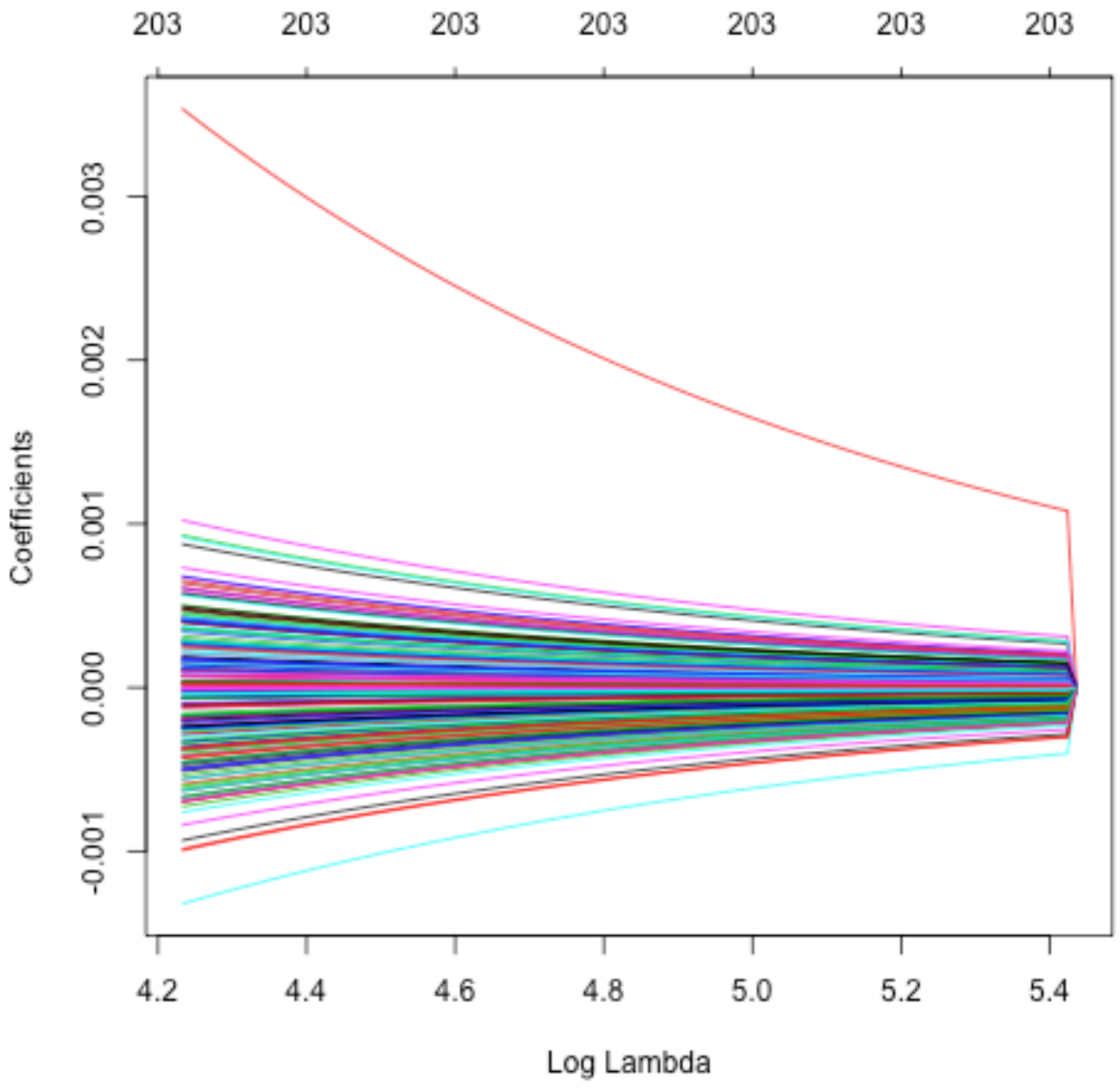
10-fold cross-validation

Deviance is a linear function of the negative log likelihood



Ridge “path”

- As λ increases, the coefficients converge towards zero.
- Notice how none of them are exactly zero, but are very close to zero!



Coefficients in ridge regression

- Our coefficients are no longer crazy.

Coefficients from ordinary least squares with no noise variables

```
Coefficients:
             Estimate Std. Error
(Intercept) -1.9451858  0.9722081
Age          0.0157108  0.0159460
Ca           1.1757984  0.1846480
Chol         0.0008275  0.0025729
```

Coefficients from ridge

```
Age           7.637208e-03
Ca            1.702367e-01
Chol          4.420894e-04
`1`           2.269135e-02
`2`           8.979332e-03
`3`          -2.481102e-02
`4`          -6.868805e-03
`5`           4.176497e-03
`6`          -1.752706e-02
`7`           3.468486e-02
`8`           7.693619e-03
`9`           5.392193e-03
`10`          1.232666e-02
`11`         -9.245543e-03
`12`         -1.241365e-02
`13`          1.895792e-02
`14`          3.479706e-03
`15`          4.013174e-03
`16`          1.672264e-02
`17`         -1.172106e-02
`18`         -1.538664e-02
```

Ridge regression

- Pros:
    - Simple procedure.
    - Ridge regression will shrink the model coefficients appropriately to prevent overfitting. All model coefficients will be non-zero.
- Cons:
    - If you believe that the outcome only depends on a few variables, you probably want a method that does feature selection.

Today's lecture

1. Model selection: Cross-validation
2. Low versus high-dimensional data
3. Regularization methods
    1. Ridge Regression
    2. **Variable selection methods**
        1. **Variable subset selection**
        2. Lasso
    3. Extensions

Linear regression + regularization

What prior knowledge can we add that can help us better estimate the model?

- Most variables do not have big effects on the outcome. Shrink coefficients toward zero.

Ridge regression

- Most variables have no effect on the outcome. Set most coefficients to zero.

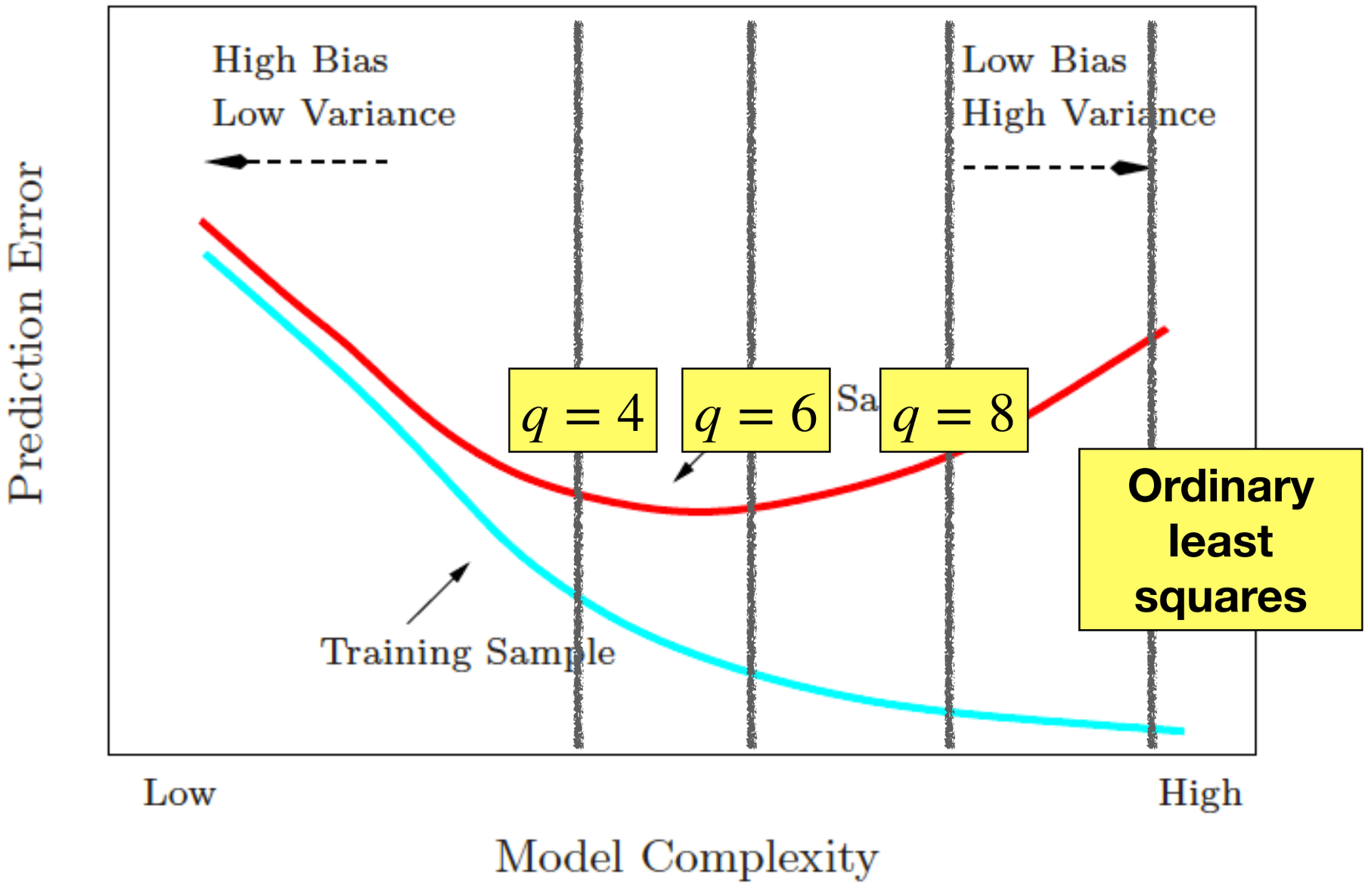
Variable selection methods, Lasso

Variable pre-selection

- A simple approach:
    - Choose a small set of variables, say the q variables most correlated with the response, where $q < n$.
    - Use least squares to fit a model predicting y using only these q variables.

Variable Pre-selection: Bias-variance Tradeoff

- Different values of q correspond to different model complexities.



Which of the procedures below is a valid approach to selecting the number of variables q?

Option 1:

1. For each possible value of q:
    A. Identify the q variables most associated with the outcome on the full dataset.
    B. Split the observations into a training set and a test set.
    C. Use the training set to fit a linear model.
    D. Evaluate the fitted model on the test set.
2. Select the q that led to the smallest test error.

Option 2:

1. Split the observations into a training set and a test set.
2. For each possible value of q:
    A. Identify the q variables most associated with the outcome on the training data.
    B. Use the training set to fit a linear model.
    C. Evaluate the fitted model on the test set.
3. Select the q that led to the smallest test error.

Finding the best subset of variables

- Variable pre-selection is simple and easy to implement, but it doesn't always work well.

- Variable pre-selection finds the variables that are most correlated with the outcome using **univariate** analyses.

- We might not have found the **best combination** for predicting the outcome.

Subset selection approaches

- **Best subset selection**: Consider all subsets of predictors
  - *Computationally intractable*
- **Stepwise regression**:
  - **Forward selection**: Greedily add predictors
  - **Backward selection**: Greedily remove predictors
  - *Heuristic procedure*
- **Modern penalized methods**

Today's lecture

1. Model selection: Cross-validation
2. Low versus high-dimensional data
3. Regularization methods
    1. Ridge Regression
    2. Variable selection methods
        1. Variable subset selection
        2. **Lasso**
    3. Extensions

The Lasso

- **The lasso** is different from ridge regression in that it encourages many model coefficients to be **exactly zero**.

- We say that the resulting model is **sparse** and that the lasso performs **feature selection.**

- The lasso tries to select the optimal combination of features for predicting the response.

The Lasso

Recall: Ordinary least squares fits a model by solving the optimization problem

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \left(y_i - (x_i^\top \beta + \beta_0) \right)^2}_{\text{Mean squared error}}$$

Lasso fits a model by adding a lasso penalty to the objective function:

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \left(y_i - (x_i^\top \beta + \beta_0) \right)^2}_{\text{Mean squared error}} + \lambda \underbrace{\sum_{j=1}^{p} |\beta_j|}_{\textbf{Lasso penalty}}$$

The ridge penalizes β_j^2.

The lasso penalizes $|\beta_j|$.

The Lasso

Recall: Ordinary least squares fits a model by solving the optimization problem

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \left(y_i - (x_i^\top \beta + \beta_0) \right)^2}_{\text{Mean squared error}}$$

Lasso fits a model by adding a lasso penalty to the objective function:

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \left(y_i - (x_i^\top \beta + \beta_0) \right)^2}_{\text{Mean squared error}} + \lambda \underbrace{\|\beta\|_1}_{\substack{\textbf{Lasso} \\ \textbf{penalty}}}$$

The Lasso + Logistic Regression

- If you have a classification problem at hand, you can do the same thing.

Logistic regression:

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \ell \left(y_i, f_{\beta_0, \beta}(x_i) \right)}_{\text{Logistic loss}}$$

Logistic regression + **Lasso**:

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} \ell \left(y_i, f_{\beta_0, \beta}(x_i) \right)}_{\text{Logistic loss}} + \lambda \underbrace{\|\beta\|_1}_{\textbf{Lasso penalty}}$$

Lasso penalty parameter

$$\hat{\beta}_\lambda = \min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - (x_i^\top \beta + \beta_0) \right)^2 + \lambda \|\beta\|_1$$

Here $\lambda \geq 0$ is a penalty parameter that controls the strength of the lasso penalty:

λ is a hyperparameter!

- Lasso is a lot like ridge!
  - When $\lambda = 0$, the lasso is the same as ordinary least squares.
  - When $\lambda = \infty$, all model coefficients are exactly zero.
- But unlike ridge, when λ is in between, **the lasso will set some coefficients to exactly zero.**

One standard error rule

- The "**One standard error rule**" is a heuristic rule for balancing model performance versus model simplicity/ interpretability.

- Recall that the validation error varies across the K folds. So we can calculate the standard error and confidence interval for the cross-validated error.

- Rather than picking the model with the smallest prediction error, we **pick the simplest model** within one standard error of the smallest prediction error.

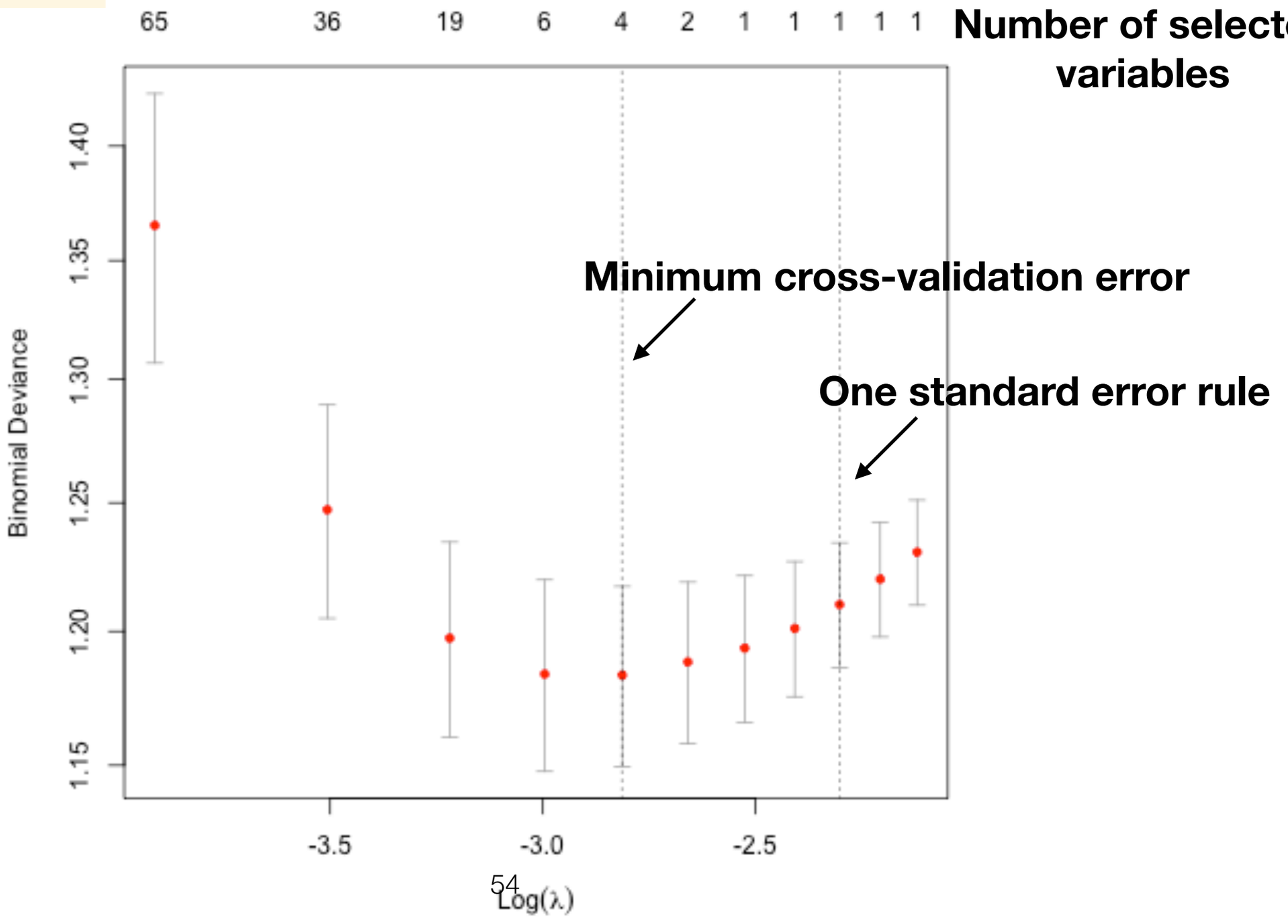
Prediction versus interpretation

- When you choose λ to minimize the cross-validated error, you are trying to find the model with the smallest prediction error. By focusing solely on model performance, this procedure tends to select a few variables that are not relevant.

- When you choose λ using the one standard error rule, you are trying to obtain a more interpretable model that only selects the truly relevant variables.

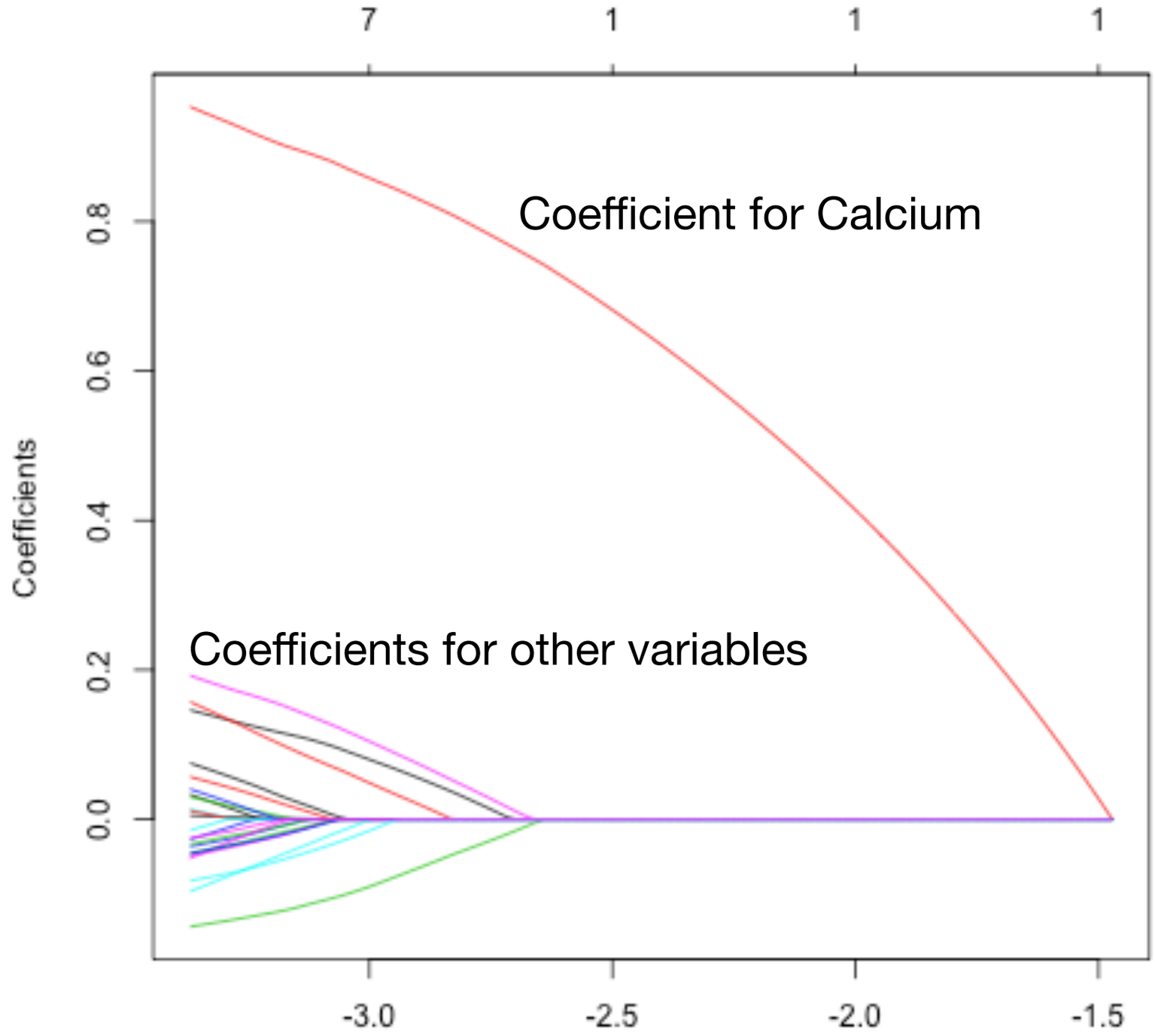
Let's try out the Lasso on our Heart data

```
lasso_model <- cv.glmnet(my_x, heart_big_x$AHD, family="binomial", alpha=1)
plot(lasso_model)
```



Lasso path

- As λ increases, the coefficients shrink towards zero. Some coefficients will be set to exactly zero.



Coefficients from the Lasso

- Let's compare using the one standard error rule to the minimum CV error rule:

```
> extract.coef(lasso_model, lambda="lambda.min")
                    Value SE Coefficient
X.Intercept. -0.72358021 NA (Intercept)
Ca            0.85676815 NA          Ca
X.14.        -0.08919939 NA        `14`
X.90.         0.07897948 NA        `90`
X.93.         0.04741359 NA        `93`
X.140.        0.10300000 NA       `140`
X.161.       -0.01422725 NA       `161`
> extract.coef(lasso_model, lambda="lambda.1se")
                  Value SE Coefficient
X.Intercept. -0.5776327 NA (Intercept)
Ca            0.6388395 NA          Ca
```

What about confidence intervals and p-values?

- Statistical theory for generating confidence intervals and p-values traditionally depends on $n \gg p$. When this no longer holds, the confidence intervals and p-values are no longer valid.

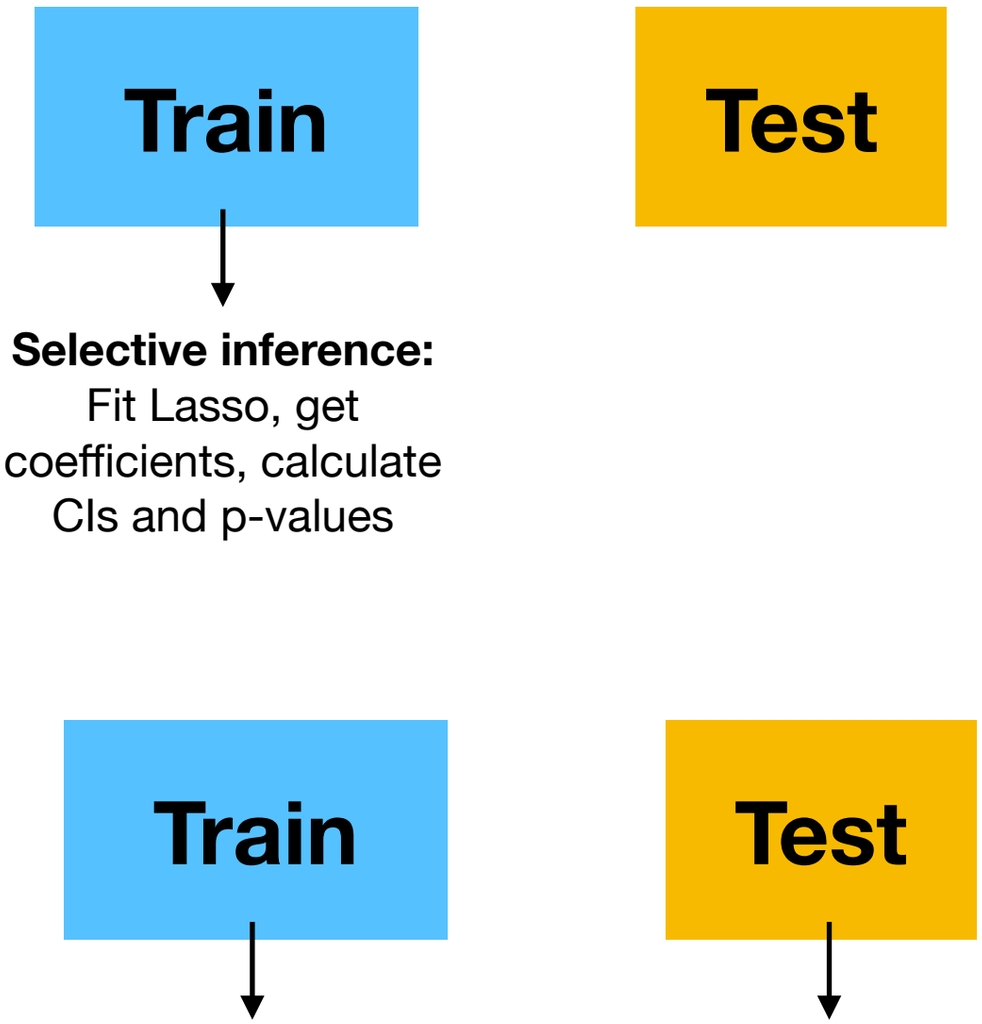
- The research field of "*selective inference*" tries to answer the question: After we mine a dataset to find potential associations (e.g. Lasso or variable subset selection), how do we assess the strength of these associations and appropriately quantify our uncertainty?

    - Existing methods rely on very strong assumptions like the noise is homoskedastic and Gaussian (this is not true for ordinary least squares!).

- Typically, the reason one fits the Lasso is to maximize prediction accuracy. The primary goal was not to assess and interpret associations between the outcome and the predictors. Therefore, I would shy away from reporting CIs and p-values.

    - However, if you insist on getting CIs and p-values, you can try a simple sample-splitting approach. The main caveat is that you need to allocate enough samples in the test data to get valid CIs and p-values.

    - Regardless, always report CI for the model's performance (e.g. CI for AUC)!



The Lasso

- Pros:
    - Simple and fast procedure
    - Performs variable selection. Has good prediction accuracy when the number of truly relevant variables is small.
- Cons:
    - Suppose a dataset with small n has a group of variables that are highly correlated. It is impossible to determine which of these variables are truly relevant. The Lasso will basically select one of these at random.

Today's lecture

1. Model selection: Cross-validation
2. Low versus high-dimensional data
3. Regularization methods
    1. Ridge Regression
    2. Variable selection methods
        1. Variable subset selection
        2. Lasso
4. **Extensions**

The Elastic Net

- The Elastic Net penalty is a mixture of the ridge and lasso penalties:

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} \ell \left(y_i, f_{\beta,\beta_0}(x_i) \right) + \lambda \left(\frac{1-\alpha}{2} \|\beta\|_2^2 + \alpha \|\beta\|_1 \right)$$

- Behaves like ridge regression and the Lasso: Performs variable selection. May select variables that are highly correlated with each other, whereas the Lasso will pick only one of these.

How many Elastic Net models did we fit?

Suppose we want to fit the elastic net. We are going to consider the possible hyperparameter values $\alpha = \{0,0.5,1\}$ and $\lambda = \{0.01,0.1,1,10\}$. We will tune the hyperparameters using 3-fold cross validation. After that, we refit the model on the full data. In total, how many Elastic Net models have we fit throughout this procedure?

A: 3 + 4 + 1 = 8

B: 3 * 4 + 1 = 13

C: (3 + 4) * 3 + 1 = 22

D: 3 * 4 * 3 + 1 = 37

Computation times can get pretty long when we tune multiple hyperparameters using cross-validation

Group lasso

- **Group lasso**: Penalty specifies how variables are grouped (e.g. gene pathways). For each group, the group lasso will either set all coefficients to exactly zero or all coefficients to some non-zero value.

Y ~ BRCA1_expr + BRCA2_expr + Favorite_Color

Variable	Coefficient
BRCA1 expr	0.5
BRCA2 expr	0
Favorite color = Blue	0.01
Favorite color = Red	0

Example result from the lasso

Variable	Coefficient
BRCA1 expr	0.5
BRCA2 expr	0.01
Favorite color = Blue	0
Favorite color = Red	0

Example result from the group lasso

Today's lecture

1. Model selection: Cross-validation
2. Low versus high-dimensional data
3. Regularization methods
    1. Ridge Regression
    2. Variable selection methods
        1. Variable subset selection
        2. Lasso
    3. Extensions