Class 7

BUS 696

Prof. Jonathan Hersh

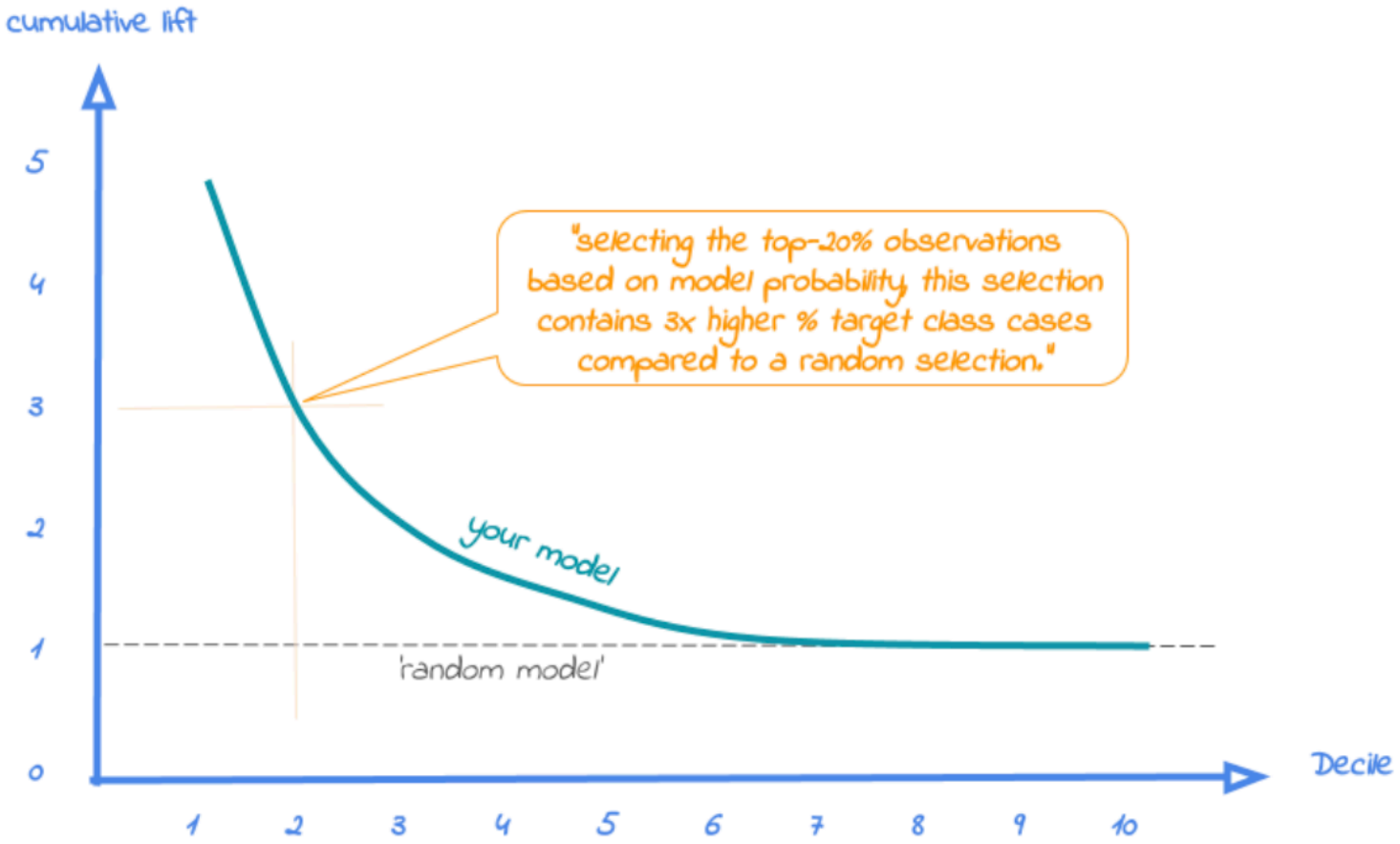
BUS 696: Class 7 Announcements

1. Pset 5 solutions
2. Problem Set 6 Posted
3. Final Project
    - Qs?
4. General Questions?

BUS 696: Class 7 Outline

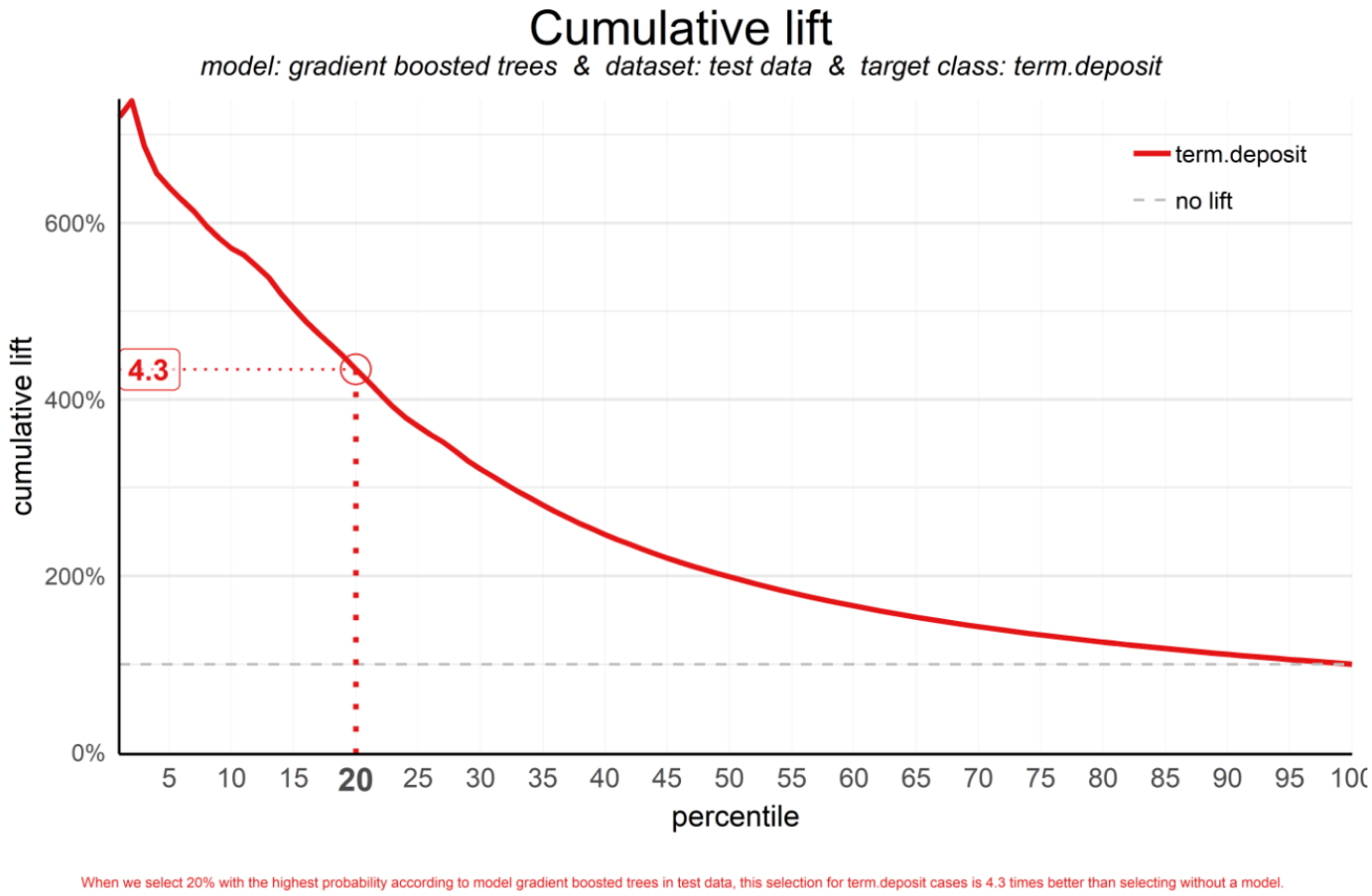
1. AI in the News
2. K-Fold Cross-Validation
3. Bootstrapping
4. Variable Selection
    - Best Subset Selection (Don’t use)
    - Forward and Backward Model Selection
5. Primer on Optimization
6. Ridge Regression

modelplotr R package: Cumulative Lift



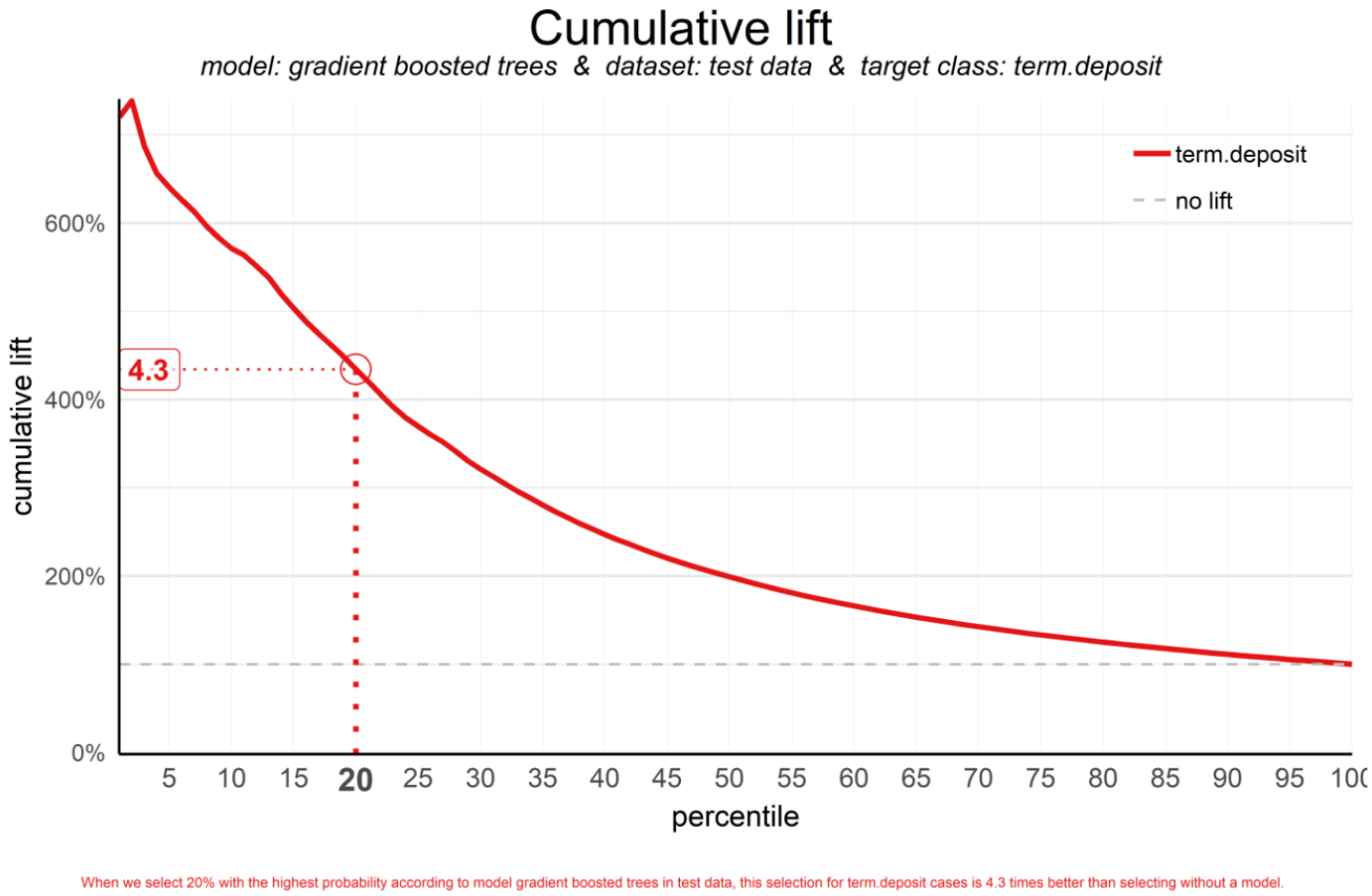
https://modelplot.github.io/intro_modelplotr.html

modelplotr R package: Cumulative Lift



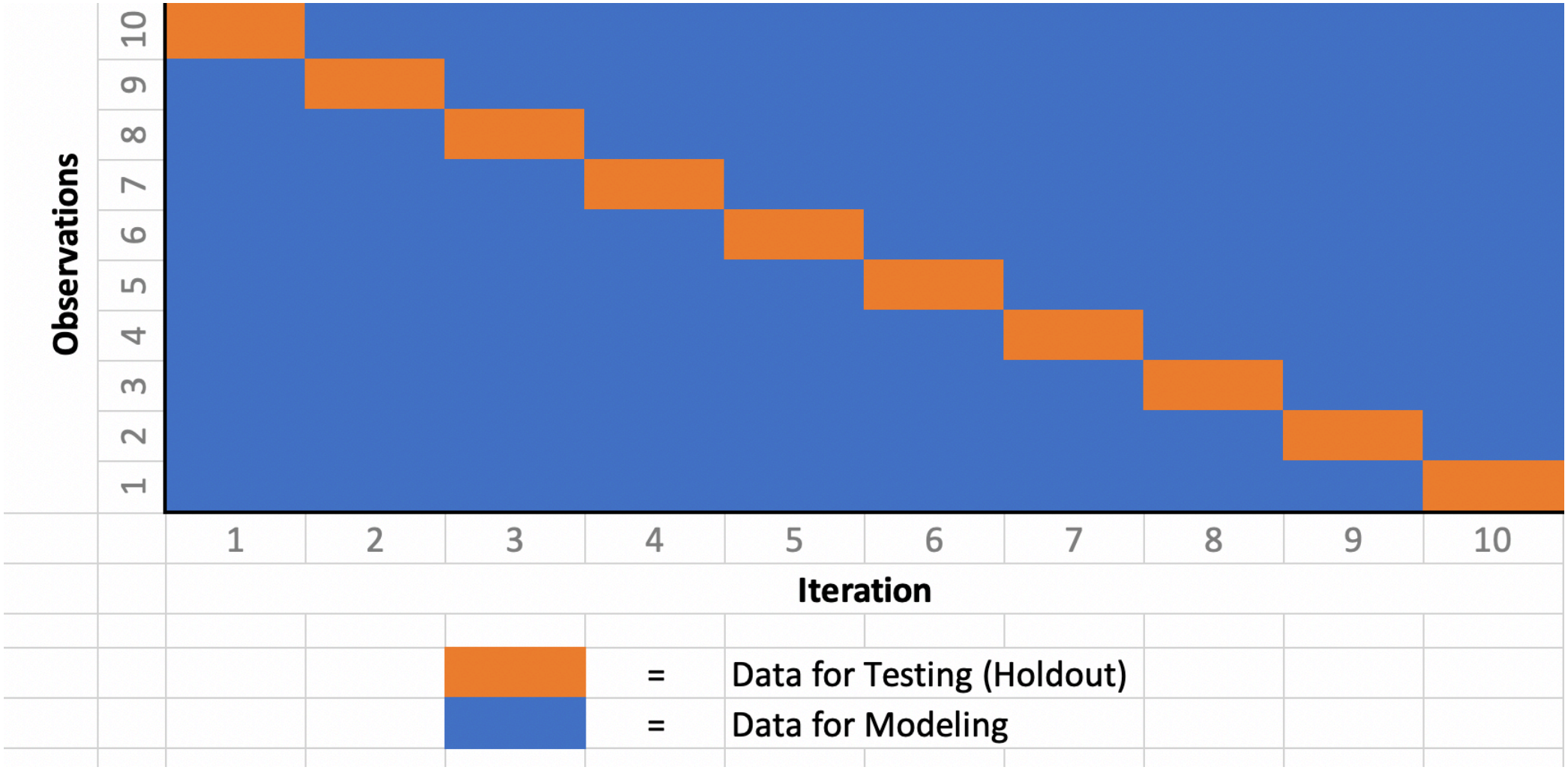
https://modelplot.github.io/intro_modelplotr.html

modelplotr R package: Cumulative Lift



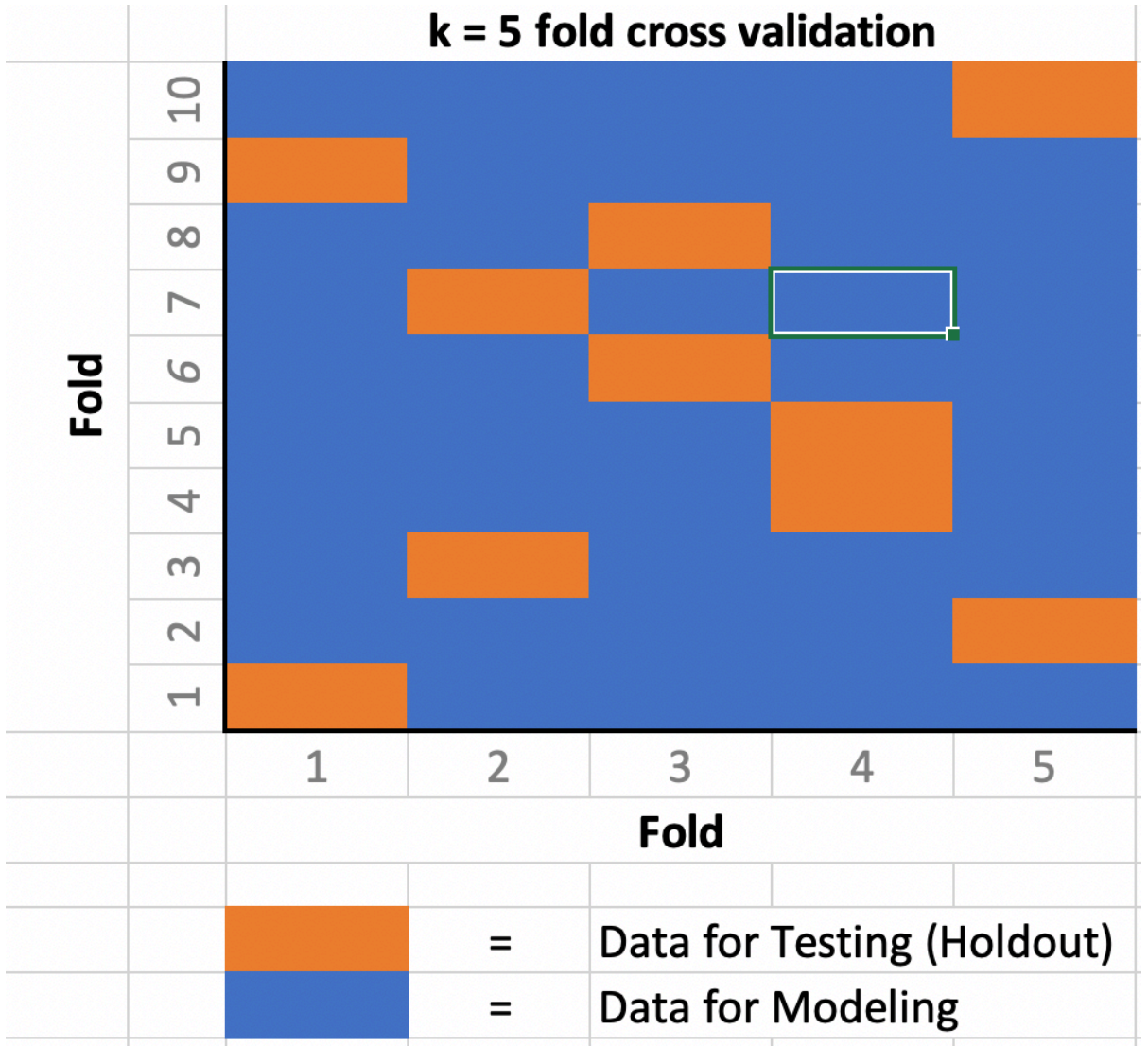
https://modelplot.github.io/intro_modelplotr.html

Leave-One-Out Cross-Validation



K-Fold Cross-Validation

- Similar to LOOCV, but we first partition (divide) data into K distinct groups
- Fit a model using data excluding group 1, use that model to predict into group 1.
- Fit a model using data excluding group 2, etc
- Proceed until we have yhats for every group



Resampling: K-Fold Cross-Validation

- We start by randomly assigning each data point to one of k folds
- Here we are setting k = 3
- We fit a model excluding data from fold 1
- That model is used to predict into fold 1

$\hat{f}_{X^{-\{1\}}}(X^{-\{1\}})$

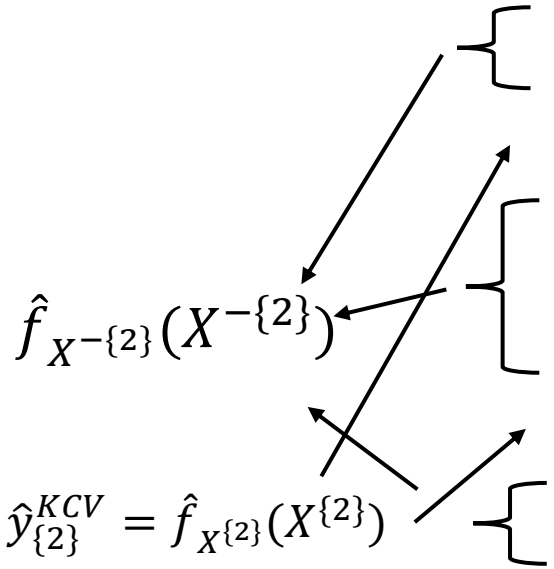
$\hat{y}_{\{1\}}^{KCV} = \hat{f}_{X^{\{1\}}}(X^{\{1\}})$

$\hat{y}^{KCV}$	Fold	mpg	cyl	Displ
	3	20	4	3
	2	15	6	5
	3	12	4	2.4
11	1	10	8	4.6
	2	14	6	3
22	1	25	4	2

$X^{-\{1\}}$: X excluding fold 1

Resampling: K-Fold Cross-Validation

- Here we are setting k = 3
- Next we fit a model excluding observations in fold 2
- That model is used to predict into fold 2

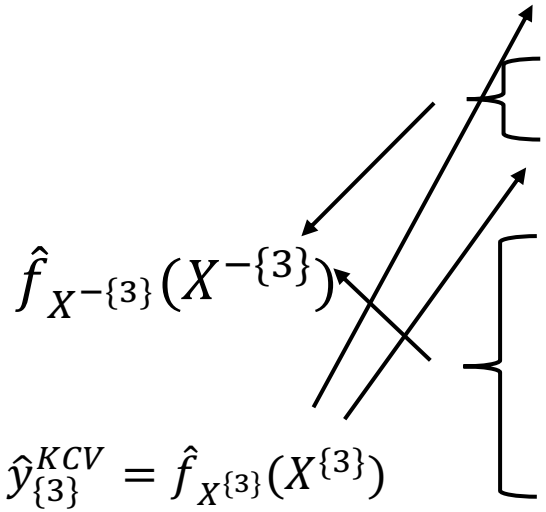


$\hat{y}^{KCV}$	Fold	mpg	cyl	Displ
	3	20	4	3
18	2	15	6	5
	3	12	4	2.4
11	1	10	8	4.6
15	2	14	6	3
22	1	25	4	2

$X^{-\{2\}}$: X excluding fold 2

Resampling: K-Fold Cross-Validation

- Here we are setting k = 3
- Next we fit a model excluding observations in fold 3
- That model is used to predict into fold 3



$\hat{y}^{KCV}$	Fold	mpg	cyl	Displ
22	3	20	4	3
18	2	15	6	5
12	3	12	4	2.4
11	1	10	8	4.6
15	2	14	6	3
22	1	25	4	2

$X^{-\{3\}}$: X excluding fold 3

K-Fold CV Versus LOOCV

- Advantages of K-Fold CV over LOOCV
  - Only need to estimate K models
- Disadvantages
  - Higher variance (more uncertainty in y_hats)
- Because of computational cost K-Fold CV more commonly used

$\hat{y}^{KCV}$	mpg	cyl	Displ
18	20	4	3
16	15	6	5
12	12	4	2.4
11	10	8	4.6
11	14	6	3
22	25	4	2

$$MSE_{KCV} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i^{KCV}\right)^2$$

K-Fold Cross-Validation in R

```
#-----------------------------------------------
### k-fold Cross validation
#-----------------------------------------------
Auto_sub <-
  mutate(Auto_sub,
         folds = createFolds(Auto_sub$mpg,
                             k = 10, list = FALSE)
  )
```

```
### K-Fold Cross Validation
nfolds <- 10
preds_10FoldCV_DF <- data.frame(
  folds = Auto_sub$folds,
  preds_10FoldCV = rep(NA,nrow(Auto_sub))
)
```

```
> Auto_sub$folds
  [1]  1  9  3  6  3  3  1  9  9  1  3 10  9  7  8  5  6  4  4  7 10  8  5
 [24]  3  9  5  4  7  7  1  2  3  7  1  2  7  5  7  1  5  1  9 10  2  2 10
 [47]  3  1  9 10  4  1  5  9  6  1  6  1  1  7  2  8  8  3 10 10 10  5  2
 [70]  1  1  9  1  2  8  3  1  2  5  2  2  8  7  9 10 10  5  4  4  7  3  3
 [93]  2  5  6  7 10  3  8  9 10 10  8 10  2  9  4  8  8  5 10  9  4  9  9
[116]  8  3  5  6  7  1  7  9  6  6  5  5  3 10  2  9  7  4  4  2  6  8  2
```

```
> preds_10FoldCV_DF
    folds preds_10FoldCV
1       1             NA
2       9             NA
3       3             NA
4       6             NA
5       3             NA
6       3             NA
7       1             NA
8       9             NA
9       9             NA
10      1             NA
11      3             NA
12     10             NA
```

K-Fold Cross-Validation in R

```r
for(i in 1:nfolds){
  mod <- lm(mpg ~ .,
            data = Auto_sub %>%
              filter(folds != i))
  preds <- predict(mod,
                   newdata = filter(Auto_sub,
                                    folds == i))
  preds_10FoldCV_DF[preds_10FoldCV_DF$folds == i,"preds_10FoldCV"]  <- preds
}
```

	RMSE (pred vs true)	R2 (pred vs true)
In-Sample	3.293	0.8215
LOOCV	3.382	0.8118
10 Fold CV	3.357	0.8147

- On average 10-Fold CV diagnostic measures will be in-between in-sample measures and LOOCV measures.

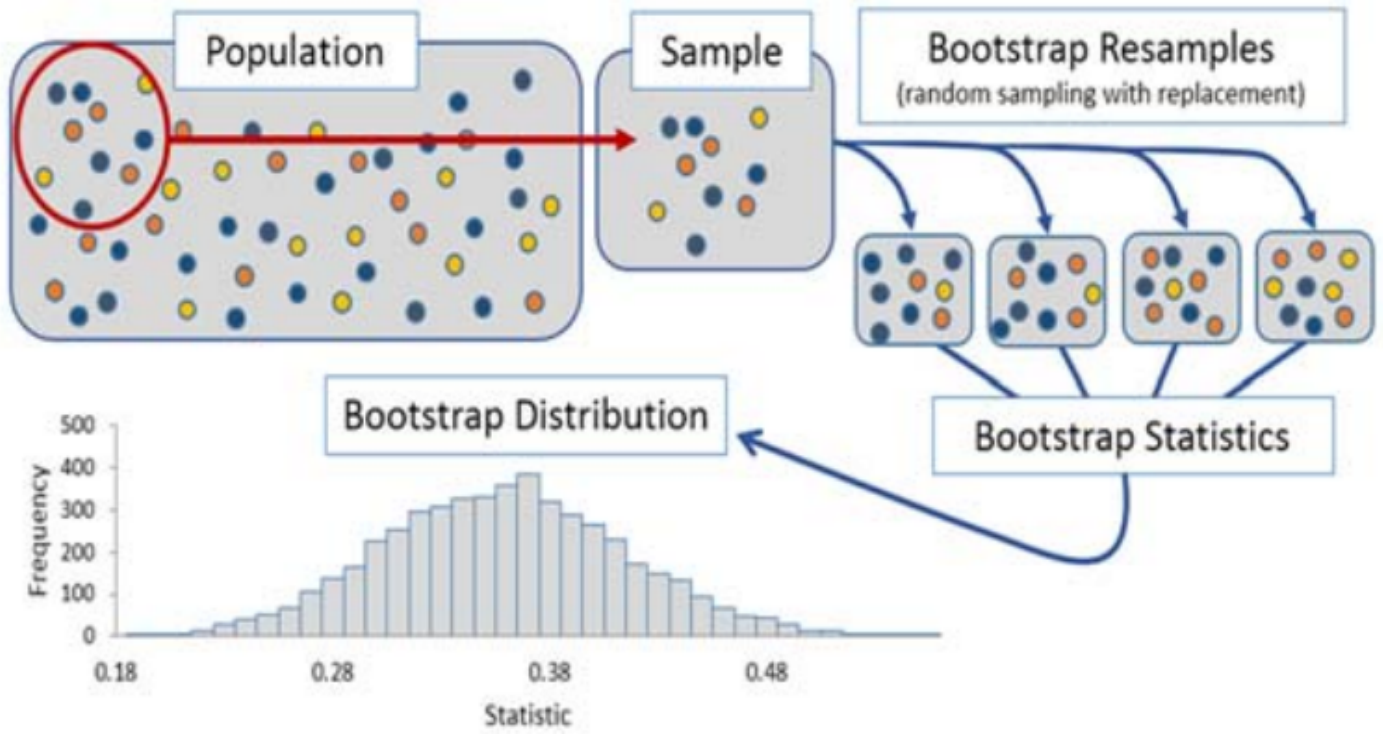
BUS 696: Class 7 Outline

1. AI in the News
2. K-Fold Cross-Validation
3. **Bootstrapping**
4. Variable Selection
    - Best Subset Selection (Don’t use)
    - Forward and Backward Model Selection
5. Primer on Optimization
6. Ridge Regression

Bootstrap

- The idea of the bootstrap is we take the original data (which is itself a sample from some population of possible data) and generate B bootstrap resamples.

- To do that we sample with replacement the original dataset until we have B bootstrap datasets, each of size n_b

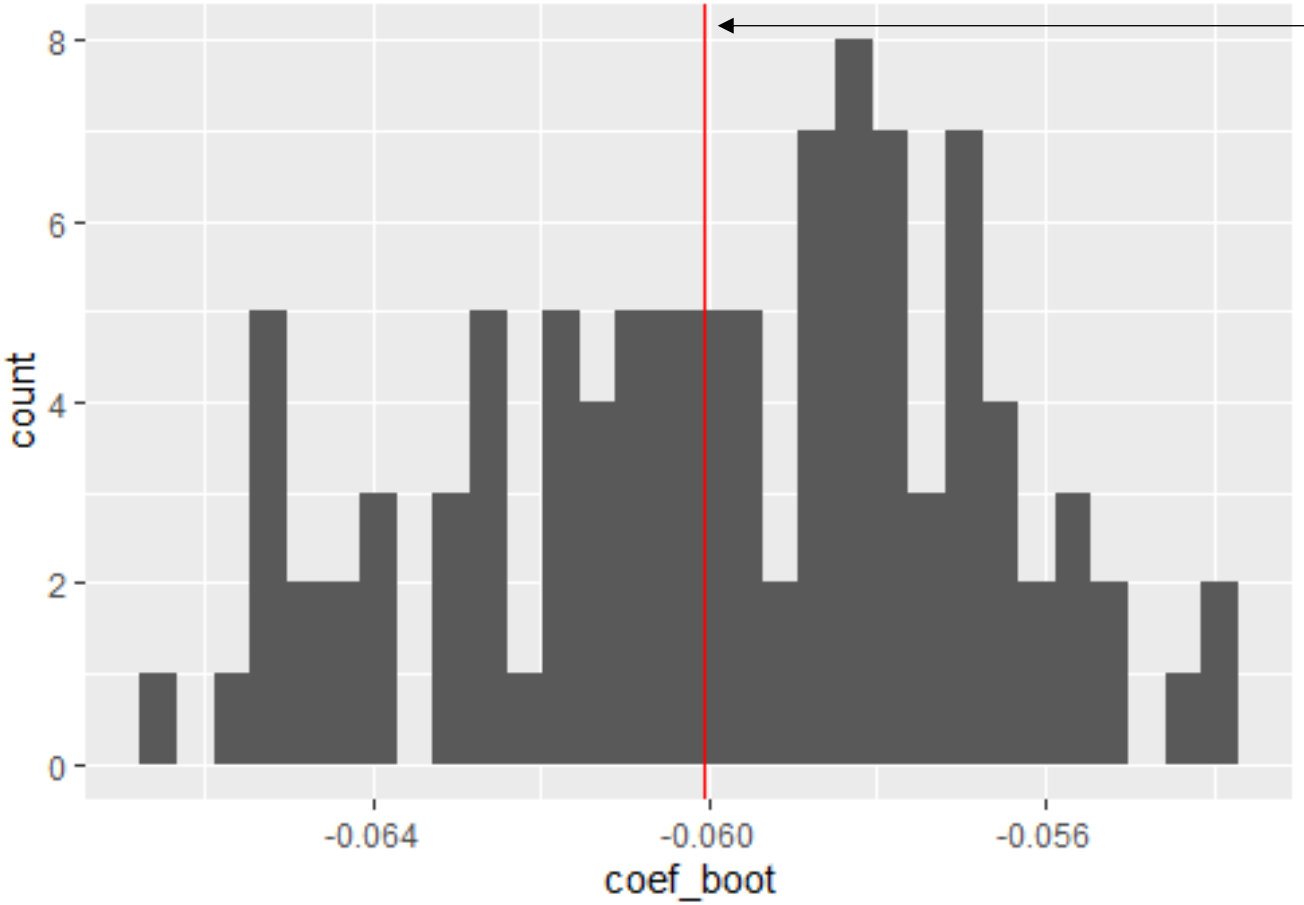


Bootstrapping in R

```r
B = 100 # number of bootstraped datasets
n_boot = 200 # size of each bootstrapped sample
coef_boot = NULL
for(b in 1:B){
  idx <- sample(1:nrow(Auto_sub),
                size = n_boot, replace = TRUE)
  mod <- lm(mpg ~ displacement,
            data = Auto_sub %>% slice(idx))
  coef_boot[b] <- mod$coefficients[2]
}
```

- Again, many ways to do it. First we do it by hand.

Bootstrapping in R



Linear model coefficient on original sample

Each point shows a coefficient from a different bootstrapped sample

```
mod_lm <- lm(mpg ~ displacement,
             data = Auto_sub)

coef_boot <- data.frame(coef_boot =
                          coef_boot)

ggplot(coef_boot, aes(x = coef_boot)) +
  geom_histogram() +
  geom_vline(xintercept = mod_lm$coefficients[2],
             color = "red")
```

BUS 696: Class 7 Outline

1. AI in the News
2. K-Fold Cross-Validation
3. Bootstrapping
4. **Variable Selection**
    - Best Subset Selection (Don’t use)
    - Forward and Backward Model Selection
5. Primer on Optimization
6. Ridge Regression

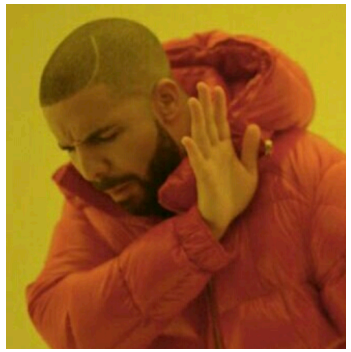
Variable Selection (Sometimes Called Model Selection)

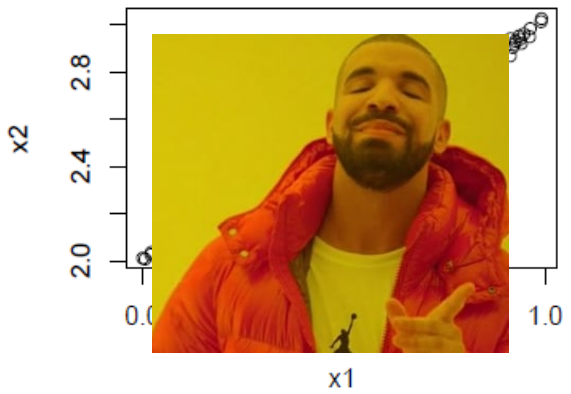
- Which Xs to include in our models?
- Including too many variables -> high variance –> model overfit
- Three solutions
    1. Subset selection
    2. Shrinkage (Lasso, Ridge, and Elastic-Net)
    3. Dimension Reduction

Solutions to feature selection problem

1. **Subset selection**
   - Identify a subset of p predictors that explain y the best
2. **Shrinkage**
   - Add a "penalty" or "cost" to additional parameters.

3. **Dimension reduction**
   - "Project" the p predictors into an M-dimensional space where $M < p$.





Best subset selection algorithm

1. For $k = 1,2, \dots, p$:

   a) Fit all $\binom{p}{k}$ models that contain exactly k variables

   b) Pick the best model (e.g. highest R^2, lowest MSE) among $\binom{p}{k}$ models, call it M_k

2. Select single best model among $M_0, \mathrm{M}_1, \dots, M_p$

A combinatorics note on $\binom{p}{k}$

- $\binom{p}{k}$ "$p\ choose\ k$"
  - "how many ways are there to **choose k elements from p objects** without caring about order"
- $\binom{p}{k} = \frac{p!}{p!(p-k)!}$
  - $p! = 1 \cdot 2 \cdot \ldots \cdot p$
- E.g. choose 3 cards from my deck of 10
  - $\binom{10}{3} = \frac{10!}{3!(10-3)!} = \frac{10!}{3! \cdot 7!} = \frac{10\times9\times8\times7\times\cdots\times1}{3\times2\times1 \cdot 7\times\cdots\times1} = \frac{10\times9\times8}{3\times2\times1} = 120$

$\binom{p}{k}$ in the context of variable selection

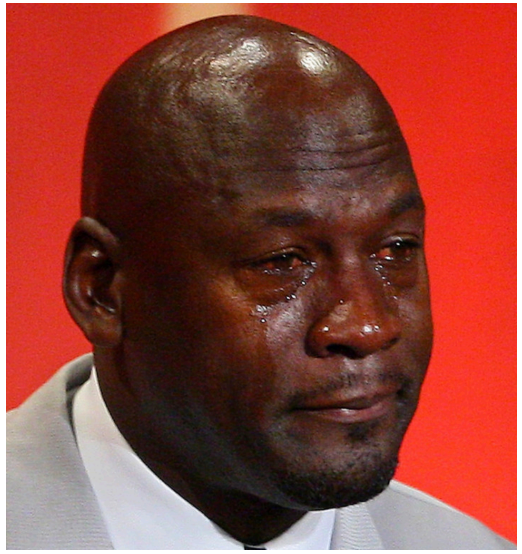
- $p =$ number of total X variables in model
- $\mathrm{k} =$ variables used to fit a given model

- $\binom{p}{k}$ = fit a model, choosing k variables out of a total set of p

Why is best subset selection not a good idea?

- Example: model with 100 variables
- You have to estimate $2^p = 2^{100} \approx 1.26 \times 10^{30}$ models for best subset selection

Name	Power	Number	SI symbol	SI prefix
One	0	1	*(none)*	*(none)*
Ten	1	10	da*(D)*	deca
Hundred	2	100	h*(H)*	hecto
Thousand	3	1,000	k*(K)*	kilo
Ten Thousand (Myriad)	4	10,000		
Hundred Thousand (Lakh)	5	100,000		
Million	6	1,000,000	M	mega
Ten Million (Crore)	7	10,000,000		
Hundred Million	8	100,000,000		
Billion (Milliard)	9	1,000,000,000	G	giga
Trillion (Billion)	12	1,000,000,000,000	T	tera
Quadrillion (Billiard)	15	1,000,000,000,000,000	P	peta
Quintillion (Trillion)	18	1,000,000,000,000,000,000	E	exa
Sextillion (Trilliard)	21	1,000,000,000,000,000,000,000	Z	zetta
Septillion (Quadrillion)	24	1,000,000,000,000,000,000,000,000	Y	yotta
Octillion (Quadrilliard)	27	1,000,000,000,000,000,000,000,000,000	X	xona
Nonillion (Quintillion)	30	1,000,000,000,000,000,000,000,000,000,000		
Decillion (Quintilliard)	33	1,000,000,000,000,000,000,000,000,000,000,000		



BUS 696: Class 7 Outline

1. AI in the News
2. K-Fold Cross-Validation
3. Bootstrapping
4. Variable Selection
    - Best Subset Selection (Don’t use)
    - **Forward and Backward Model Selection**
5. Primer on Optimization
6. Ridge Regression

Forward stepwise algorithm

1. Start with M_0, a model with just an intercept

1. For $k = 0,1, \ldots, p-1$:

    a) Add one variables in set $p - \mathrm{k}$ that best improves model M_k

    b) Choice of variable to add may be lowest p-value, highest increase in adjusted R^2 (implementation can vary)

2. Select single best model among $M_0, \mathrm{M}_1, \ldots, M_p$

leaps package – regsubsets() function

regsubsets {leaps}

functions for model selection

Description

Model selection by exhaustive search, forward or backward stepwise, or sequential replacement

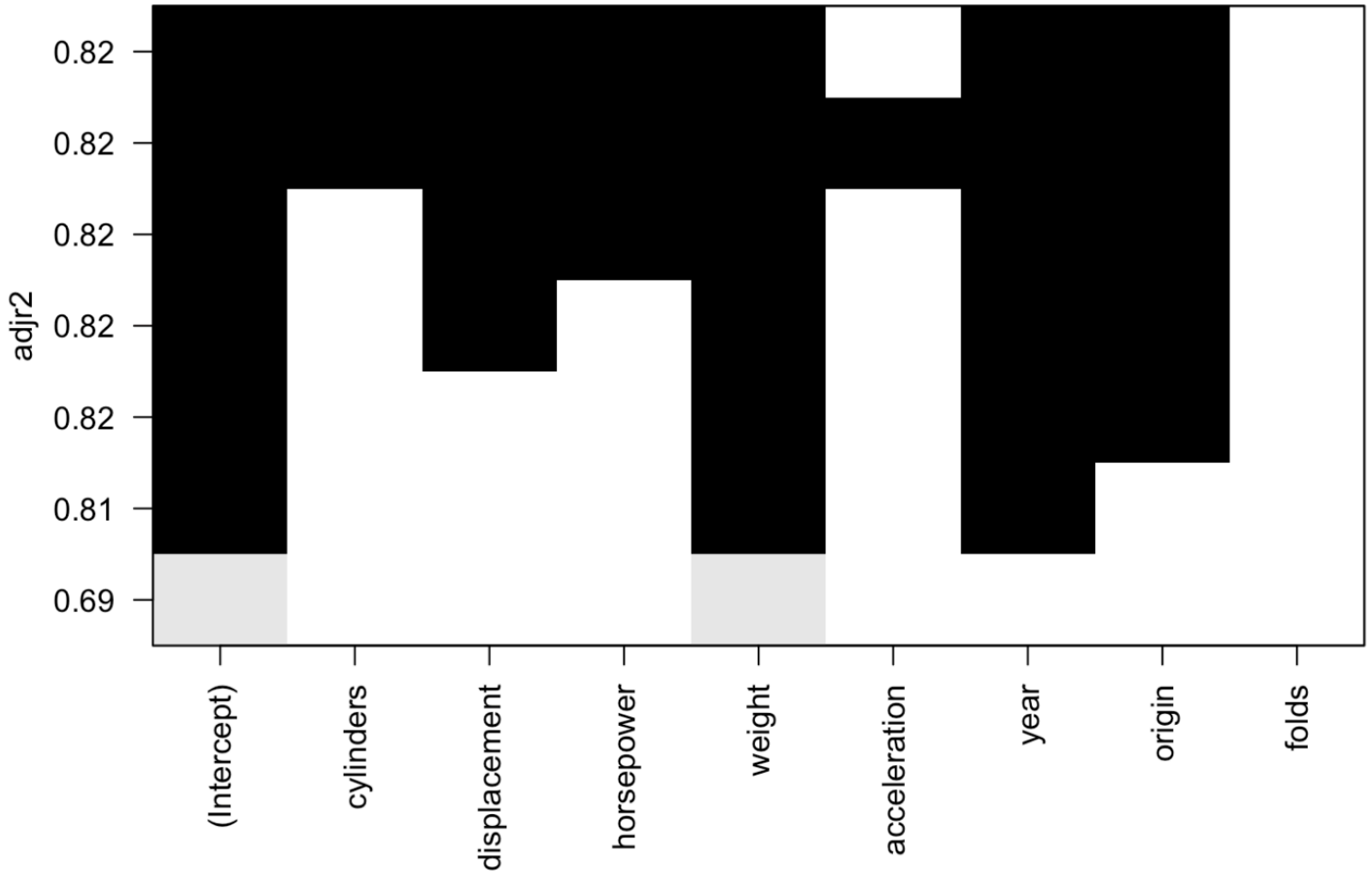
Usage

```
regsubsets(x=, ...)

## S3 method for class 'formula'
regsubsets(x=, data=, weights=NULL, nbest=1, nvmax=8,
 force.in=NULL, force.out=NULL, intercept=TRUE,
 method=c("exhaustive", "backward", "forward", "seqrep"),
 really.big=FALSE,
 nested=(nbest==1),...)

## Default S3 method:
regsubsets(x=, y=, weights=rep(1, length(y)), nbest=1, nvmax=8,
force.in=NULL, force.out=NULL, intercept=TRUE,
 method=c("exhaustive","backward", "forward", "seqrep"),
really.big=FALSE,nested=(nbest==1),...)
```

Regsubsets() forward selection

```
Subset selection object
Call: regsubsets.formula(mpg ~ ., data = Auto_sub, nvmax = 7, method = "forward")
8 Variables  (and intercept)
             Forced in Forced out
cylinders        FALSE      FALSE
displacement     FALSE      FALSE
horsepower       FALSE      FALSE
weight           FALSE      FALSE
acceleration     FALSE      FALSE
year             FALSE      FALSE
origin           FALSE      FALSE
folds            FALSE      FALSE
1 subsets of each size up to 7
Selection Algorithm: forward
         cylinders displacement horsepower weight acceleration year origin folds
1  ( 1 ) " "       " "          " "        "*"    " "          " "  " "    " "
2  ( 1 ) " "       " "          " "        "*"    " "          "*"  " "    " "
3  ( 1 ) " "       " "          " "        "*"    " "          "*"  "*"    " "
4  ( 1 ) " "       "*"          " "        "*"    " "          "*"  "*"    " "
5  ( 1 ) " "       "*"          "*"        "*"    " "          "*"  "*"    " "
6  ( 1 ) "*"       "*"          "*"        "*"    " "          "*"  "*"    " "
7  ( 1 ) "*"       "*"          "*"        "*"    "*"          "*"  "*"    " "
```

Plot Summary of Forward Model Selection



Backwards stepwise algorithm

1. Start with M_p, a full model with all variables

1. For $k = p, p-1, \ldots, 1$:

    a) Remove one variables in set k that best improves model M_k

    b) Choice of variable to remove may be one that has lowest p-value, highest increase in adjusted R^2 (implementation can vary)

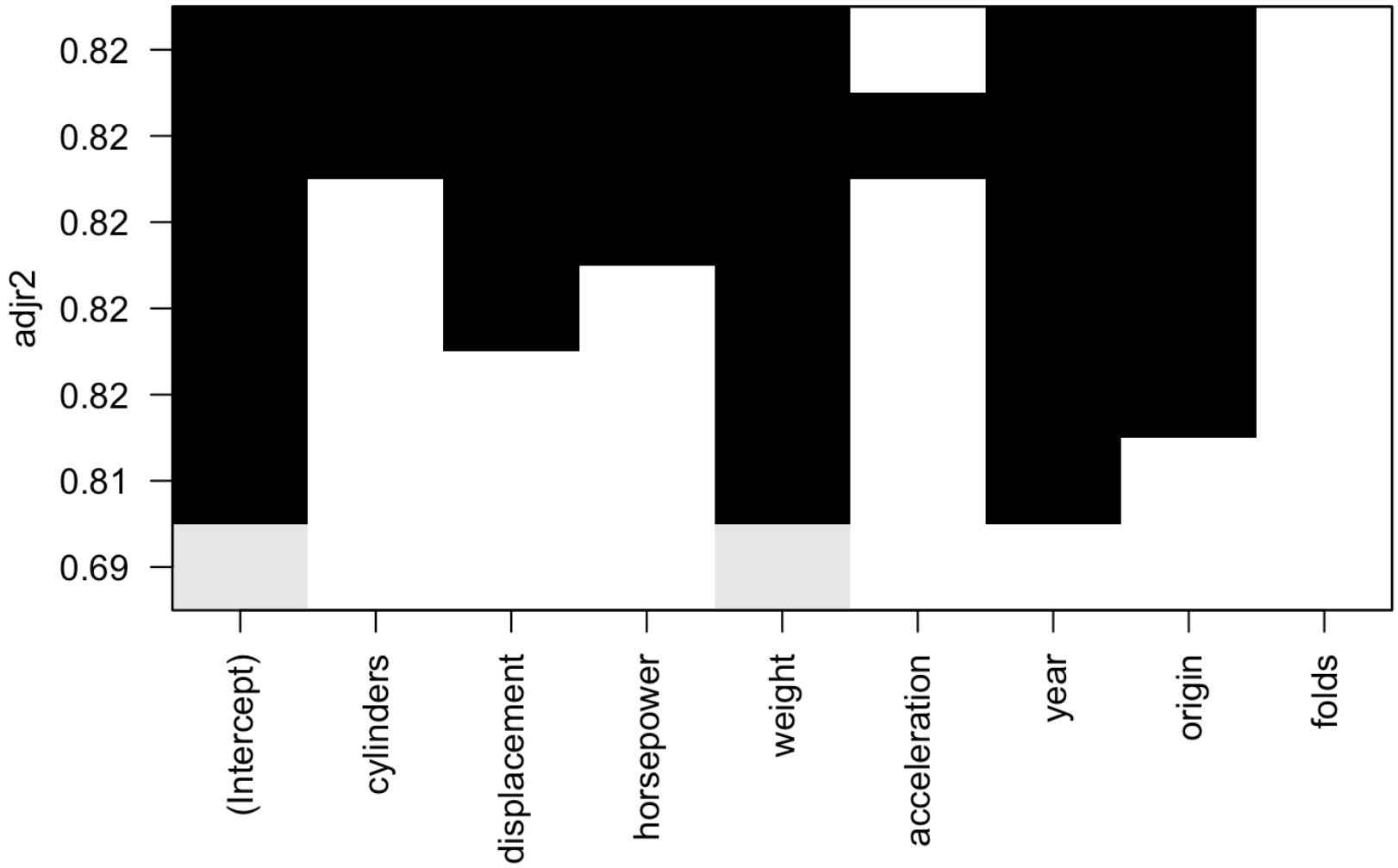
2. Select single best model among $M_0, \mathrm{M}_1, \ldots, M_p$

Regsubsets backward selection

```r
auto_fit_bkwd <-
  regsubsets(mpg ~ . ,
             data = Auto_sub,
             nvmax = 7,
             method = "backward")
summary(auto_fit_bkwd)
plot(auto_fit_bkwd, scale = "adjr2")
```

```
Selection Algorithm: backward
         cylinders displacement horsepower weight acceleration year origin folds
1  ( 1 ) " "       " "          " "        "*"    " "          " "  " "    " "
2  ( 1 ) " "       " "          " "        "*"    " "          "*"  " "    " "
3  ( 1 ) " "       " "          " "        "*"    " "          "*"  "*"    " "
4  ( 1 ) " "       "*"          " "        "*"    " "          "*"  "*"    " "
5  ( 1 ) " "       "*"          "*"        "*"    " "          "*"  "*"    " "
6  ( 1 ) "*"       "*"          "*"        "*"    " "          "*"  "*"    " "
7  ( 1 ) "*"       "*"          "*"        "*"    "*"          "*"  "*"    " "
```

Plot Summary of Backward Model Selection



Pros/cons of forward stepwise

Pros

- Quick for small datasets with small p
- Can overfit if don't cross validate over some selection mechanism (i.e. # of max variables to use in model, p-value selection criteria

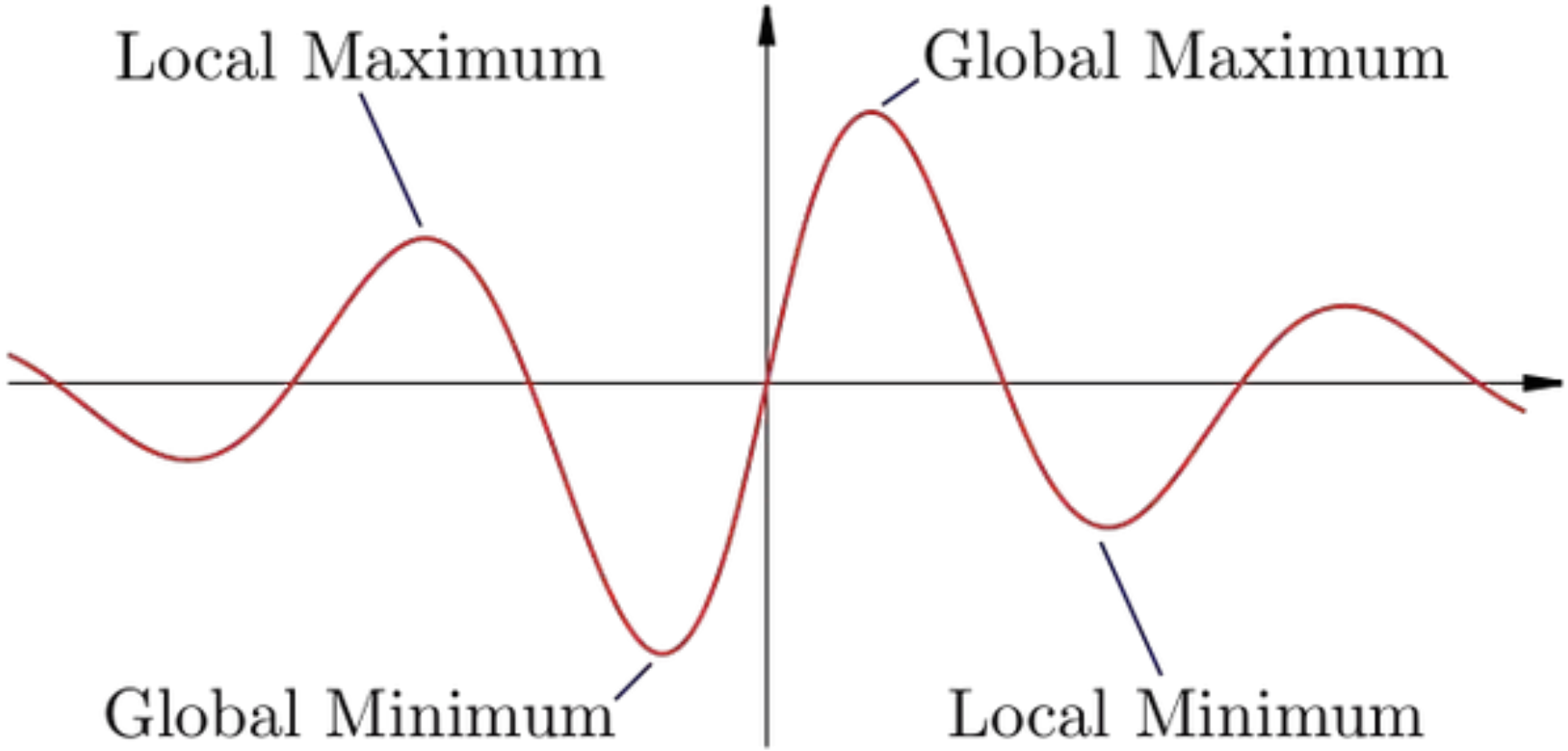
Cons

- Doesn't work well in the presence of multicollinearity
- Not guaranteed best solution (aka global maximum)

BUS 696: Class 7 Outline

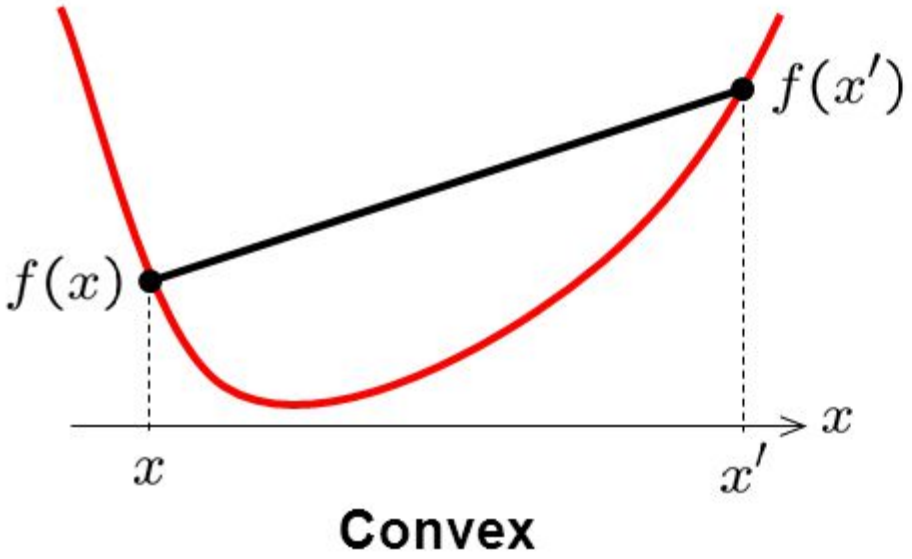
1. AI in the News
2. K-Fold Cross-Validation
3. Bootstrapping
4. Variable Selection
    - Best Subset Selection (Don’t use)
    - Forward and Backward Model Selection
5. **Primer on Optimization**
6. Ridge Regression

Global versus local maximum

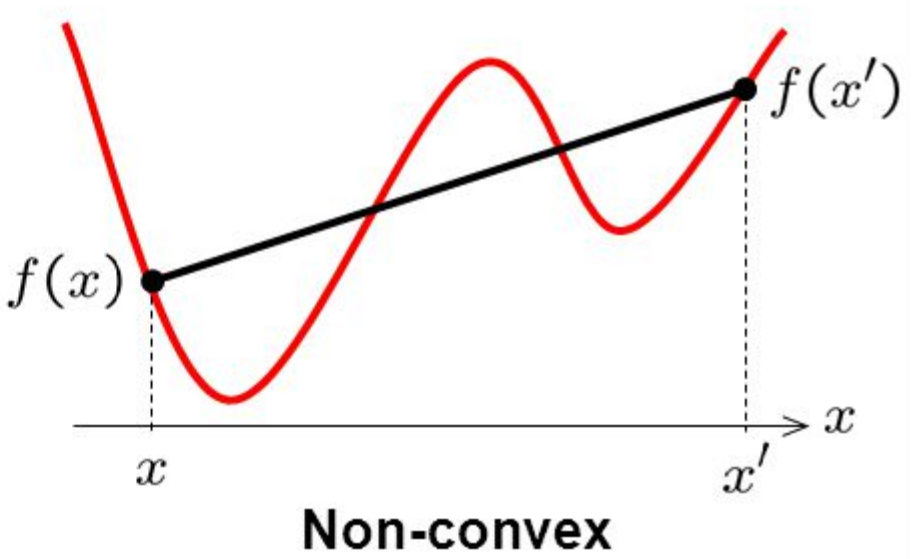


Convex versus non-convex function

- A function f is convex if $f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda)f(x')$ for any x and x'
- (That is the chord connecting any two points in the function is completely contained in the function)



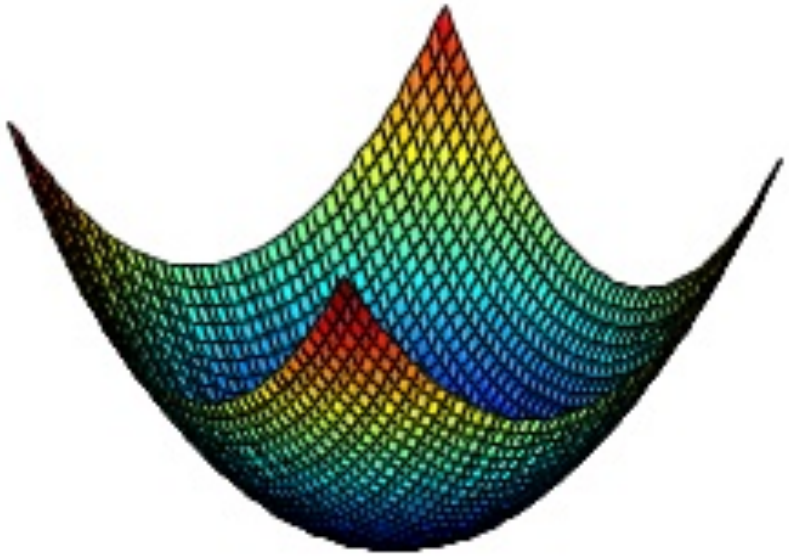
Convex

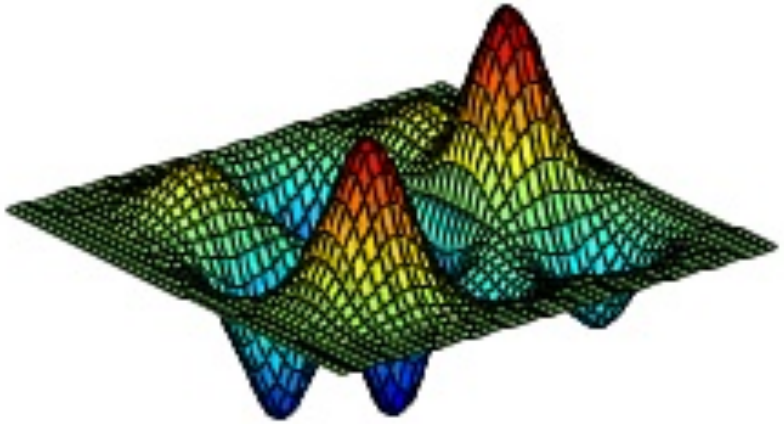


Non-convex

Sufficiency for global versus local minimum

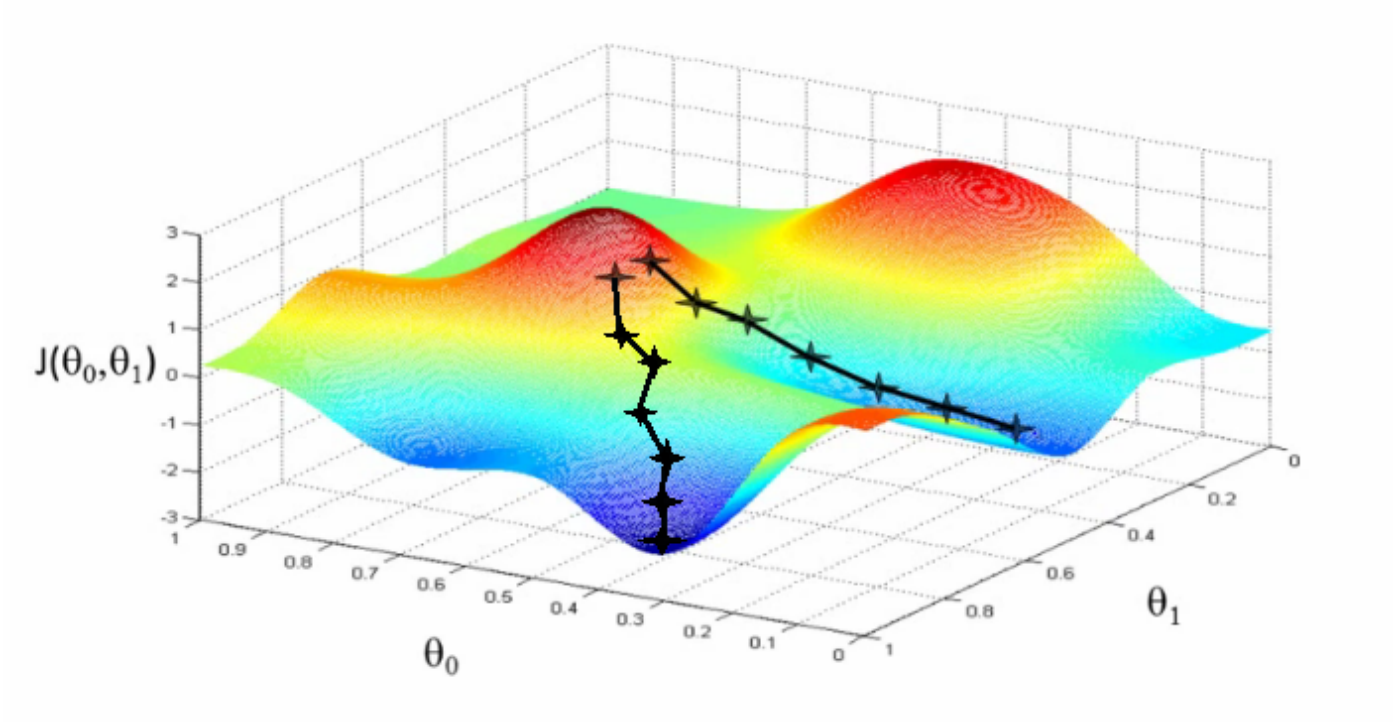
- If x^* is a local minimum for $f(x)$ and f is convex then x^* is the **global minimum**





What does this mean for stepwise selection?

- Stepwise is a non-convex optimization problem
- Therefore, we are never guaranteed a global minimum (i.e. that β's will minimize residual sum of squares $\sum(y_i - \beta_0 - \sum x_i \beta_j)$



BUS 696: Class 7 Outline

1. AI in the News
2. K-Fold Cross-Validation
3. Bootstrapping
4. Variable Selection
    - Best Subset Selection (Don’t use)
    - Forward and Backward Model Selection
5. Primer on Optimization
6. **Ridge Regression**

Least Squares (OLS Estimator)

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{K} x_{ij}\, \beta_j \right)^2 \right\}$$

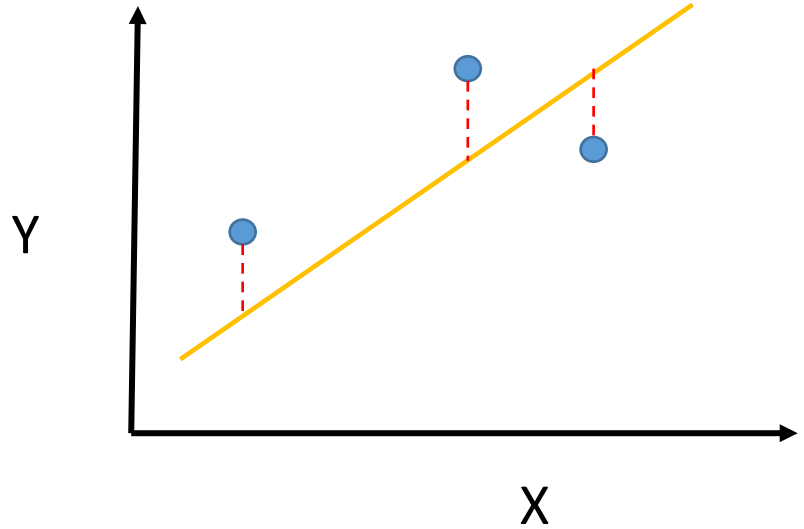
Beta_hat “solves” (e.g. is the argument that minimizes) the objective function

$$\hat{\beta}\ minimizes: \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{K=1} x_{ij}\, \beta_j \right)^2 \right\}$$

Least Squares (OLS Estimator)

$$\hat{\beta}\ minimizes: \sum_{i=1}^{N}(y_i - \beta_0 - x_{i1}\beta_1)^2$$

Least squares minimizes the sum of squared residuals (e.g. $y_i - \hat{y}$)

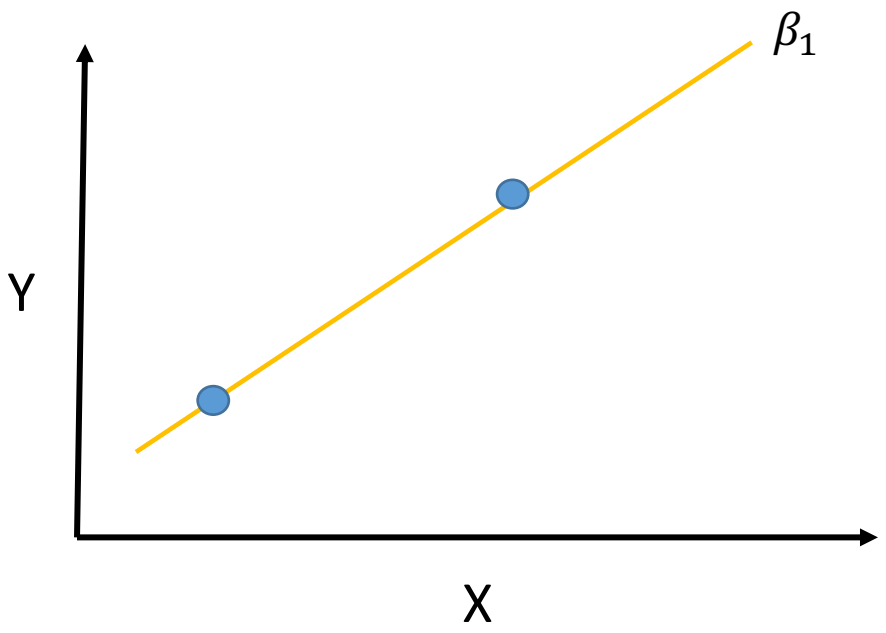


Let's set K = 1 (i.e. one explanatory variable to make this easier)

Visually, the slope (β_1) minimizes the difference between the points and the yellow line (red lines)

Ridge Regression Idea

$\hat{\beta}_{ridge}\, minimizes\colon\ residuals + \lambda \cdot (\beta_1)^2$

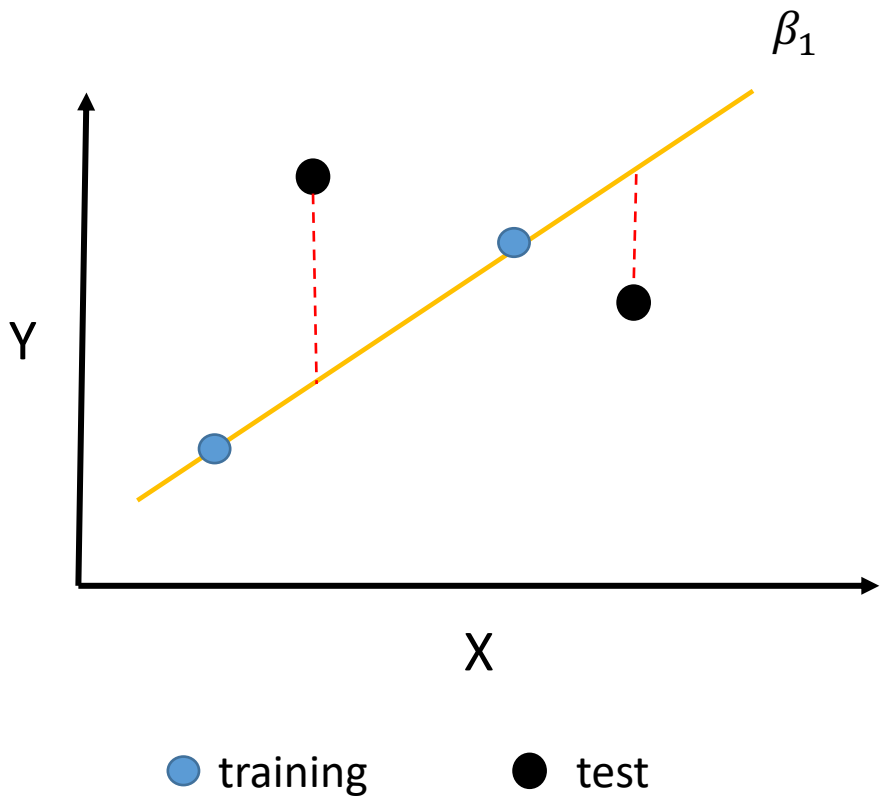


Ridge regression in contrast minimizes the OLS residuals plus the squared slope of beta times λ

Why is this smart? Let's take the example where we only have two data points.

Our best fit line describes the points perfectly and our residuals = 0. Our bias is zero!

OLS Bias and Variance



Bias = 0, which is good

What about our variance?

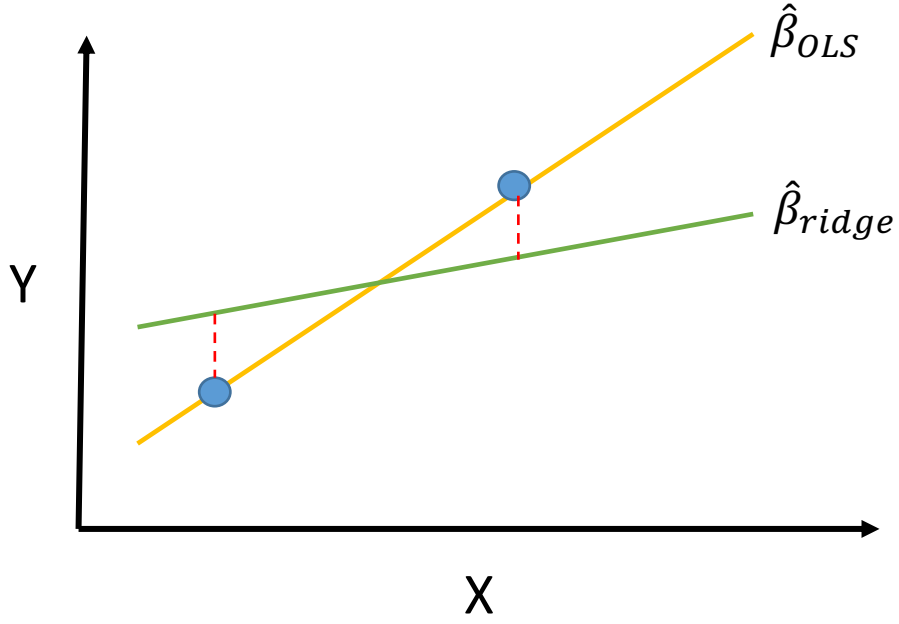
Imaging we had the following data in our test set

Then our test error would be the lines in red, and our variance would be high

Intuition: increasing bias can often reduce variance

Ridge Regression Idea

$\hat{\beta}_{ridge}\ minimizes\colon\ residuals + \lambda \cdot (slope)^2$



Now let $\lambda = 1$. $\hat{\beta}_{ridge}$ minimizes:

residuals + $1 \cdot (\beta_1)^2$

= 0 + $1 \cdot (\beta_1)^2$

Suppose the OLS slope = 2

then $\hat{\beta}_{ridge}$ would choose to have some positive residuals to because

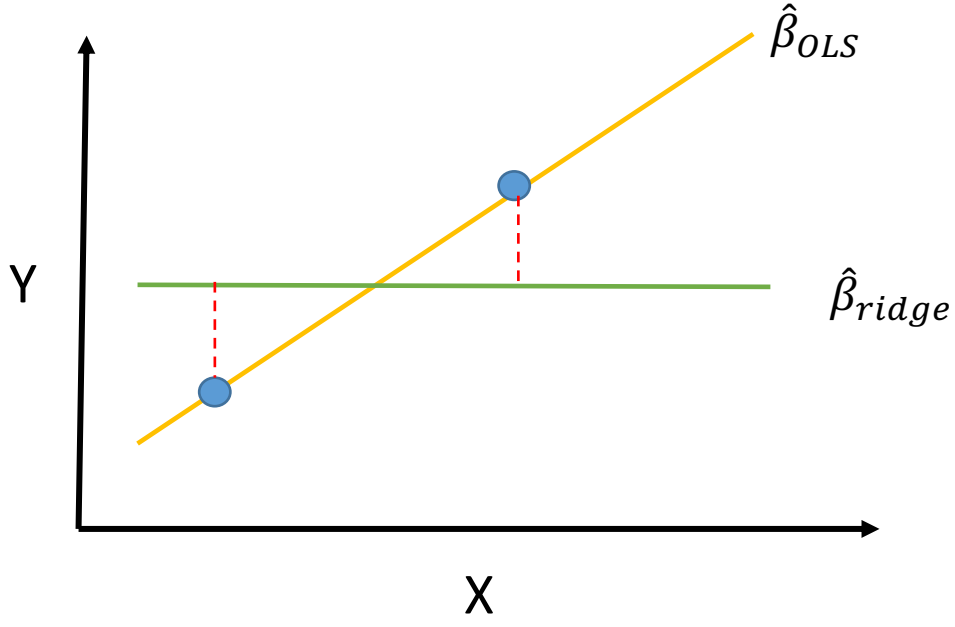
-residuals + $1 \cdot (1)^2$

is likely less than $0 + (2)^2 = 4$

Aka: we accept a little bias (higher residuals) for less variance (better test performance)

Ridge Regression and Lambda

$\hat{\beta}_{ridge}\ minimizes\colon\ residuals + \lambda \cdot (slope)^2$



What if we set $\lambda = 1000$?

$\hat{\beta}_{ridge}$ minimizes:

residuals + $1000 \cdot (\beta_1)^2$

Here ridge will have to accept very high residuals in order to avoid high slope penalty

$\hat{\beta}_{ridge}$ will set slope = 0 and we get:

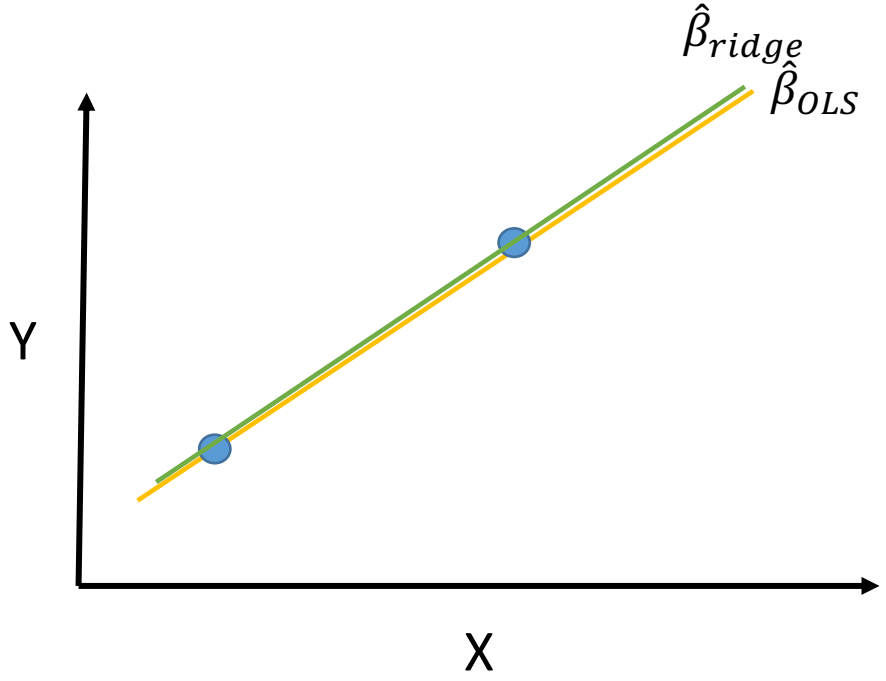
residuals + $1000 \cdot (0)^2$

= residuals + 0

E.g. we accept a lot of bias but low variance

Ridge Regression and Lambda

$\hat{\beta}_{ridge}\, minimizes\colon\ residuals + \lambda \cdot (slope)^2$



What if we set $\lambda = 0$?

$\hat{\beta}_{ridge}$ minimizes:

residuals + $0 \cdot (\beta_1)^2$

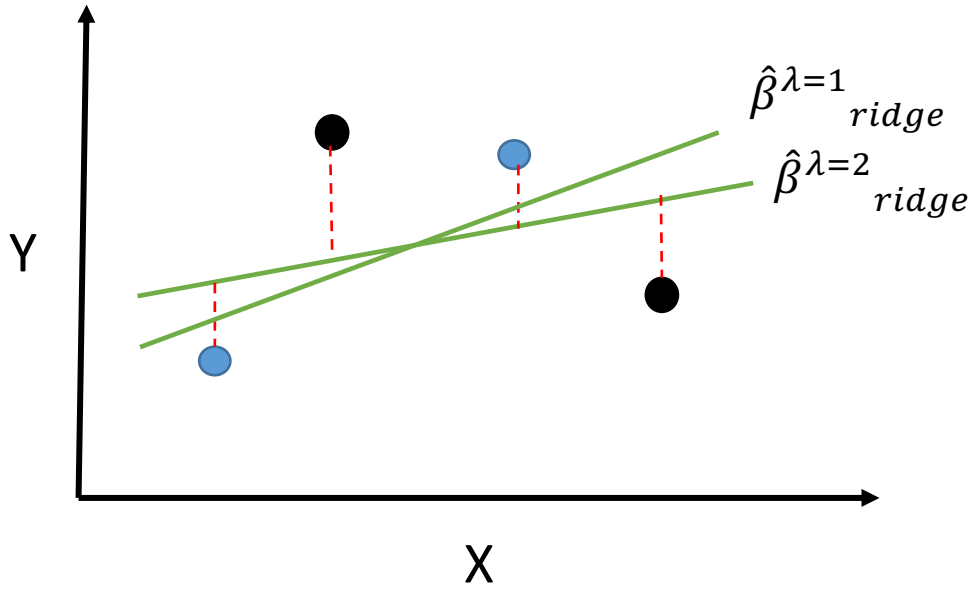
= residuals

That's just the OLS estimator and $\hat{\beta}_{ridge} = \hat{\beta}_{OLS}$

E.g. lower the penalty on lambda the closer ridge is to OLS.

Choosing λ for Ridge?

$\hat{\beta}_{ridge}\, minimizes\colon\ residuals + \lambda \cdot (slope)^2$



So how do we choose λ?

In practice we estimate a many models with many different values of λ

We pick a min and max lambda (say 0 and 10), then choose some points in-between

Our **optimal** λ^* is the lambda that minimizes cross-validated error

glmnet package

glmnet

From glmnet v2.0-18 by Trevor Hastie — 99.99th Percentile

Fit A GLM With Lasso Or Elasticnet Regularization

Fit a generalized linear model via penalized maximum likelihood. The regularization path is computed for the lasso or elasticnet penalty at a grid of values for the regularization parameter lambda. Can deal with all shapes of data, including very large sparse data matrices. Fits linear, logistic and multinomial, poisson, and Cox regression models.

Keywords models, regression

Usage

```
glmnet(x, y, family=c("gaussian","binomial","poisson","multinomial","cox","mgaussian"),
    weights, offset=NULL, alpha = 1, nlambda = 100,
    lambda.min.ratio = ifelse(nobs
```

Arguments

x input matrix, of dimension nobs x nvars; each row is an observation vector. Can be in sparse matrix format (inherit from class `"sparseMatrix"` as in package `Matrix` ; not yet available for `family="cox"`)

y response variable. Quantitative for `family="gaussian"` , or `family="poisson"` (non-negative counts). For `family="binomial"` should be either a factor with two levels, or a two-column matrix of counts or proportions (the second column is treated as the target class; for a factor, the last level in alphabetical order is the target class). For `family="multinomial"` , can be a `nc>=2` level factor, or a matrix with `nc` columns of counts or proportions. For either `"binomial"` or `"multinomial"` , if `y` is presented as a vector, it will be coerced into a factor. For `family="cox"` , `y` should be a two-column matrix with columns named 'time' and 'status'. The latter is a binary variable, with '1' indicating death, and '0' indicating right censored. The function `Surv()` in package survival produces such a matrix. For `family="mgaussian"` , `y` is a matrix of quantitative responses.

glmnetUtils: Necessary Additional Package

Introduction to glmnetUtils

The glmnetUtils package provides a collection of tools to streamline the process of fitting elastic net models with glmnet. I wrote the package after a couple of projects where I found myself writing the same boilerplate code to convert a data frame into a predictor matrix and a response vector. In addition to providing a formula interface, it also features a function `cva.glmnet` to do crossvalidation for both α and λ, as well as some utility functions.

The formula interface

The interface that glmnetUtils provides is very much the same as for most modelling functions in R. To fit a model, you provide a formula and data frame. You can also provide any arguments that glmnet will accept. Here are some simple examples for different types of data:

```
# least squares regression
(mtcarsMod <- glmnet(mpg ~ cyl + disp + hp, data=mtcars))
```

```
## Call:
## glmnet.formula(formula = mpg ~ cyl + disp + hp, data = mtcars)
##
## Model fitting options:
##     Sparse model matrix: FALSE
##     Use model.frame: FALSE
##     Alpha: 1
##     Lambda summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.03326 0.11690 0.41003 1.02839 1.44125 5.05505
```

glmnetUtils: Necessary Additional Package

```
# estimate ridge mod
Ridge_mod <- cv.glmnet(profitM ~ .,
                       data = movies_train %>%
                         select(-c(director_name,actor_1_name,
                                   actor_2_name,actor_3_name,
                                   plot_keywords,movie_imdb_link,
                                   country,budgetM,grossM, genres,
                                   language, movie_title)),
                       alpha = 0)

coef(Ridge_mod)
```

Very important:

Set alpha = 0 to get ridge

alpha = 1 to estimate Lasso model (more on this later)

coef() function

```
> coef(Ridge_mod)
31 x 1 sparse Matrix of class "dgCMatrix"
                                          1
(Intercept)                   375.8728467164299
color                           3.3709229184702
color Black and White          -1.4710509793354
colorColor                      1.5332871913922
num_critic_for_reviews          0.0000661387495
duration                       -0.0560651087860
director_facebook_likes        -0.0001810292387
actor_3_facebook_likes          0.0002041101931
actor_1_facebook_likes         -0.0000430491628
gross                           0.0000008036590
num_voted_users                 0.0000290618332
cast_total_facebook_likes       0.0000061427897
facenumber_in_poster            0.0473720680504
num_user_for_reviews            0.0006816099097
content_ratingPG                1.1396240741168
content_ratingPG-13             0.2819483822036
content_ratingR                -0.9505434689214
content_ratingOther             0.5102102078375
budget                         -0.0000007517667
title_year                     -0.1874436349418
actor_2_facebook_likes         -0.0000069751333
imdb_score                      0.7207706122038
aspect_ratio                   -1.4278392481220
movie_facebook_likes            0.0000078108292
genre_mainAction               -2.6892740660307
genre_mainAdventure            -1.3700422837322
genre_mainComedy                2.0182528348528
genre_mainCrime                -1.0237554171846
genre_mainDrama                 0.5432805069308
genre_mainOther                 2.0246784070736
cast_total_facebook_likes000s   0.0147232247410
```