Subset\_Selection\_Jan30\_HW1

Hamed

1/30/2020

### SUBSET SELECTION: Forward, Backward & Stepwise selection methods using the “cpus” dataset

##### 1. Load the required packages and the cpus dataset from the MASS package

library(MASS)  
library(tidyverse)

## -- Attaching packages -------------------------------------- tidyverse 1.3.0 --

## <U+2713> ggplot2 3.2.1 <U+2713> purrr 0.3.3  
## <U+2713> tibble 2.1.3 <U+2713> dplyr 0.8.3  
## <U+2713> tidyr 1.0.0 <U+2713> stringr 1.4.0  
## <U+2713> readr 1.3.1 <U+2713> forcats 0.4.0

## -- Conflicts ----------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x dplyr::select() masks MASS::select()

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(leaps)  
head(cpus)

## name syct mmin mmax cach chmin chmax perf estperf  
## 1 ADVISOR 32/60 125 256 6000 256 16 128 198 199  
## 2 AMDAHL 470V/7 29 8000 32000 32 8 32 269 253  
## 3 AMDAHL 470/7A 29 8000 32000 32 8 32 220 253  
## 4 AMDAHL 470V/7B 29 8000 32000 32 8 32 172 253  
## 5 AMDAHL 470V/7C 29 8000 16000 32 8 16 132 132  
## 6 AMDAHL 470V/8 26 8000 32000 64 8 32 318 290

##### 2. Use syct, mmin , mmax , cach , chmin, chmax as the predictors (independent variables) to predict performance (perf).

From the model output we use the p-value to check the best predictors in the model, condition: (p-value < 0.05).

my\_model<-lm(perf~syct + mmin + mmax + cach + chmin + chmax, data = cpus, nvmax = 6)

## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :  
## extra argument 'nvmax' will be disregarded

summary(my\_model)

##   
## Call:  
## lm(formula = perf ~ syct + mmin + mmax + cach + chmin + chmax,   
## data = cpus, nvmax = 6)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -195.84 -25.17 5.41 26.53 385.75   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -5.590e+01 8.045e+00 -6.948 4.99e-11 \*\*\*  
## syct 4.886e-02 1.752e-02 2.789 0.00579 \*\*   
## mmin 1.529e-02 1.827e-03 8.371 9.42e-15 \*\*\*  
## mmax 5.571e-03 6.418e-04 8.680 1.33e-15 \*\*\*  
## cach 6.412e-01 1.396e-01 4.594 7.64e-06 \*\*\*  
## chmin -2.701e-01 8.557e-01 -0.316 0.75263   
## chmax 1.483e+00 2.201e-01 6.738 1.64e-10 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 59.99 on 202 degrees of freedom  
## Multiple R-squared: 0.8649, Adjusted R-squared: 0.8609   
## F-statistic: 215.5 on 6 and 202 DF, p-value: < 2.2e-16

##### 3. Perform best subset selection in order to choose the best predictors from the above predictors. What is the best model obtained according to Cp, BIC, and adjusted R2?

models <- regsubsets(perf~syct + mmin + mmax + cach + chmin + chmax,  
 data = cpus, nvmax = 6)  
summary(models)

## Subset selection object  
## Call: regsubsets.formula(perf ~ syct + mmin + mmax + cach + chmin +   
## chmax, data = cpus, nvmax = 6)  
## 6 Variables (and intercept)  
## Forced in Forced out  
## syct FALSE FALSE  
## mmin FALSE FALSE  
## mmax FALSE FALSE  
## cach FALSE FALSE  
## chmin FALSE FALSE  
## chmax FALSE FALSE  
## 1 subsets of each size up to 6  
## Selection Algorithm: exhaustive  
## syct mmin mmax cach chmin chmax  
## 1 ( 1 ) " " " " "\*" " " " " " "   
## 2 ( 1 ) " " " " "\*" "\*" " " " "   
## 3 ( 1 ) " " "\*" "\*" " " " " "\*"   
## 4 ( 1 ) " " "\*" "\*" "\*" " " "\*"   
## 5 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"   
## 6 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"

res.sum <- summary(models)

Displays the CP values at each predictor number from 1-predictor to 6-predictor

res.sum$cp

## [1] 176.563616 95.808585 28.225948 10.977588 5.099604 7.000000

Displays the BIC values at each predictor number from 1-predictor to 6-predictor

res.sum$bic

## [1] -274.7146 -320.4675 -370.5300 -383.5185 -386.1684 -380.9290

This shows the number of predictors that BIC, CP, ADJ.R2 support as best subset is the one with lowest BIC, CP, ADJ.R2 values.

data.frame(  
 Adj.R2 = which.max(res.sum$adjr2),  
 CP = which.min(res.sum$cp),  
 BIC = which.min(res.sum$bic)  
)

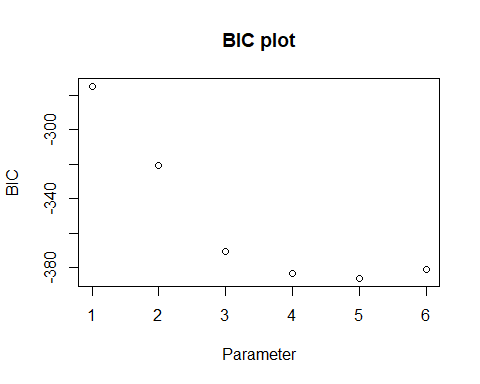
## Adj.R2 CP BIC  
## 1 5 5 5

* The function summary() reports the best set of variables for each model size. From the output, an asterisk specifies that a given variable is included in the corresponding model.
* For example, it can be seen that the best 2-variables model contains only mmax and cach variables (perf ~ mmax + cach). The best three-variable model is (perf ~ mmax + cach + mmin), and so forth.
* As shown above, adjusted R2,BIC and Cp criteria, tells us that the best model is the one with 5 predictor variables.
* A natural question is: which of these best models should we finally choose for our predictive analytics?

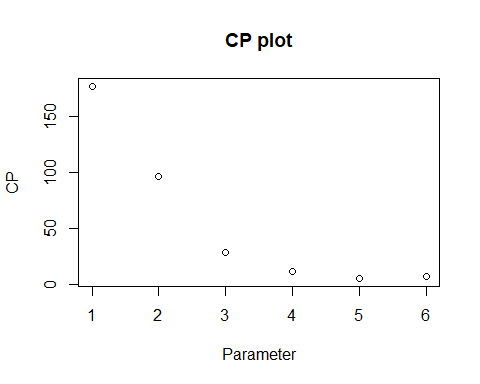
##### 4. Show some plots to provide evidence for your answer, and report the coefficients of the best model obtained for each criteria.

* The plots show that the BIC reduces as the number of parameters increase upto 5 then it becomes constant. The same goes for CP.

plot(res.sum$bic, xlab="Parameter", ylab="BIC",main="BIC plot")



plot(res.sum$cp, xlab="Parameter", ylab="CP",main = "CP plot")



##### 5. Repeat using forward stepwise selection and also using backwards stepwise selection.How does your answer compare to the best subset results?

* nvmax: the number of variable in the model. For example nvmax = 2, specify the best 2-variables model
* RMSE and MAE are two different metrics measuring the prediction error of each model. The lower the RMSE and MAE, the better the model.
* Rsquared indicates the correlation between the observed outcome values and the values predicted by the model. The higher the R squared, the better the model.

Fit the full model to show the performance before subset selection

library(MASS)  
full.model <- lm(perf ~syct + mmin + mmax + cach + chmin + chmax, data = cpus)  
full.model

##   
## Call:  
## lm(formula = perf ~ syct + mmin + mmax + cach + chmin + chmax,   
## data = cpus)  
##   
## Coefficients:  
## (Intercept) syct mmin mmax cach chmin   
## -55.900116 0.048863 0.015294 0.005571 0.641207 -0.270065   
## chmax   
## 1.482694

###### Model information

* Specify the tuning parameter nvmax, which corresponds to the maximum number of predictors to be incorporated in the model.
* For example, you can vary nvmax from 1 to 5. In this case, the function starts by searching different best models of different size, up to the best 5-variables model.
* That is, it searches the best 1-variable model, the best 2-variables model, …, the best 5-variables models.
* As the data set contains only 6 predictors, we’ll vary nvmax from 1 to 6 resulting to the identification of the 6 best models with different sizes: the best 1-variable model, the best 2-variables model, …, the best 6-variables model.
* We’ll use 10-fold cross-validation to estimate the average prediction error (RMSE) of each of the 6 models
* The output of the final model has predictors selected based on the number of asterics , the more the better.

###### Forward selection

# Set seed for reproducibility  
set.seed(123)  
# Set up repeated k-fold cross-validation  
train.control <- trainControl(method = "cv", number = 10)  
# Train the model  
  
step.model <- train(perf ~syct + mmin + mmax + cach + chmin + chmax, data = cpus,  
 method = "leapForward",   
 tuneGrid = data.frame(nvmax = 1:6),  
 trControl = train.control  
)  
step.model$results

## nvmax RMSE Rsquared MAE RMSESD RsquaredSD MAESD  
## 1 1 93.46400 0.6488552 56.59877 60.38499 0.1779760 22.54073  
## 2 2 89.08646 0.7168064 53.65864 59.70345 0.1501266 21.52435  
## 3 3 83.01186 0.7670026 48.96523 59.63821 0.1347547 21.51615  
## 4 4 74.41985 0.8004094 45.31268 49.53468 0.1247854 17.23585  
## 5 5 72.35948 0.8086084 44.53950 48.65457 0.1051621 17.19711  
## 6 6 72.78049 0.8063649 44.82444 47.12075 0.1081818 16.66283

Here we show the number of predictors the best subset will have after forward selection

step.model$bestTune

## nvmax  
## 5 5

The final model after forward selection showing the importance of each predictor.

summary(step.model$finalModel)

## Subset selection object  
## 6 Variables (and intercept)  
## Forced in Forced out  
## syct FALSE FALSE  
## mmin FALSE FALSE  
## mmax FALSE FALSE  
## cach FALSE FALSE  
## chmin FALSE FALSE  
## chmax FALSE FALSE  
## 1 subsets of each size up to 5  
## Selection Algorithm: forward  
## syct mmin mmax cach chmin chmax  
## 1 ( 1 ) " " " " "\*" " " " " " "   
## 2 ( 1 ) " " " " "\*" "\*" " " " "   
## 3 ( 1 ) " " "\*" "\*" "\*" " " " "   
## 4 ( 1 ) " " "\*" "\*" "\*" " " "\*"   
## 5 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"

###### Backward selection

set.seed(123)  
# Set up repeated k-fold cross-validation  
train.control <- trainControl(method = "cv", number = 10)  
# Train the model  
  
# Train the model  
step.model2 <- train(perf ~syct + mmin + mmax + cach + chmin + chmax, data = cpus,  
 method = "leapBackward",   
 tuneGrid = data.frame(nvmax = 1:6),  
 trControl = train.control)  
step.model2$results

## nvmax RMSE Rsquared MAE RMSESD RsquaredSD MAESD  
## 1 1 90.67875 0.6109622 53.41392 62.55296 0.2355191 24.00383  
## 2 2 88.89009 0.7007681 52.00267 62.55203 0.1235651 23.38322  
## 3 3 76.09071 0.7765757 46.44309 60.69159 0.1381791 23.17784  
## 4 4 74.41985 0.8004094 45.31268 49.53468 0.1247854 17.23585  
## 5 5 72.35948 0.8086084 44.53950 48.65457 0.1051621 17.19711  
## 6 6 72.78049 0.8063649 44.82444 47.12075 0.1081818 16.66283

Here we show the number of predictors the best subset will have after Backward selection.

step.model2$bestTune

## nvmax  
## 5 5

The final model after after Backward selection showing the importance of each predictor.

summary(step.model2$finalModel)

## Subset selection object  
## 6 Variables (and intercept)  
## Forced in Forced out  
## syct FALSE FALSE  
## mmin FALSE FALSE  
## mmax FALSE FALSE  
## cach FALSE FALSE  
## chmin FALSE FALSE  
## chmax FALSE FALSE  
## 1 subsets of each size up to 5  
## Selection Algorithm: backward  
## syct mmin mmax cach chmin chmax  
## 1 ( 1 ) " " "\*" " " " " " " " "   
## 2 ( 1 ) " " "\*" " " " " " " "\*"   
## 3 ( 1 ) " " "\*" "\*" " " " " "\*"   
## 4 ( 1 ) " " "\*" "\*" "\*" " " "\*"   
## 5 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"

###### Stepwise selection

set.seed(123)  
# Set up repeated k-fold cross-validation  
train.control <- trainControl(method = "cv", number = 10)  
# Train the model  
  
step.model3 <- train(perf ~syct + mmin + mmax + cach + chmin + chmax, data = cpus,  
 method = "leapSeq",   
 tuneGrid = data.frame(nvmax = 1:6),  
 trControl = train.control)  
step.model3$results

## nvmax RMSE Rsquared MAE RMSESD RsquaredSD MAESD  
## 1 1 93.46400 0.6488552 56.59877 60.38499 0.17797603 22.54073  
## 2 2 88.10355 0.7235298 52.90480 60.02006 0.14462287 21.58357  
## 3 3 76.85737 0.7660263 46.67915 60.69824 0.15230524 23.38628  
## 4 4 73.30065 0.8123798 45.88578 48.41967 0.08509873 16.35415  
## 5 5 72.00203 0.8281705 44.90645 47.86131 0.08199148 16.33195  
## 6 6 72.78049 0.8063649 44.82444 47.12075 0.10818179 16.66283

Here we show the number of predictors the best subset will have

step.model3$bestTune

## nvmax  
## 5 5

The final model after stepwise selection showing the importance of each predictor.

summary(step.model3$finalModel)

## Subset selection object  
## 6 Variables (and intercept)  
## Forced in Forced out  
## syct FALSE FALSE  
## mmin FALSE FALSE  
## mmax FALSE FALSE  
## cach FALSE FALSE  
## chmin FALSE FALSE  
## chmax FALSE FALSE  
## 1 subsets of each size up to 5  
## Selection Algorithm: 'sequential replacement'  
## syct mmin mmax cach chmin chmax  
## 1 ( 1 ) " " " " "\*" " " " " " "   
## 2 ( 1 ) " " " " "\*" "\*" " " " "   
## 3 ( 1 ) " " "\*" "\*" " " " " "\*"   
## 4 ( 1 ) "\*" "\*" "\*" "\*" " " " "   
## 5 ( 1 ) "\*" "\*" "\*" "\*" "\*" " "

### REGRESSION METHODS: OLS, RIDGE, LASSO, PCR, & PLS USING “College” dataset

#### Predict the number of applications received using the other variables in the College data set in library ISLR

###### (a) Split the data set into a training set and a test set using caret library and fit each of the following models using caret and ten fold cross validation.

library(ISLR)  
library(glmnet)

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

## Loaded glmnet 3.0-2

attach(College)  
head(College)

## Private Apps Accept Enroll Top10perc Top25perc  
## Abilene Christian University Yes 1660 1232 721 23 52  
## Adelphi University Yes 2186 1924 512 16 29  
## Adrian College Yes 1428 1097 336 22 50  
## Agnes Scott College Yes 417 349 137 60 89  
## Alaska Pacific University Yes 193 146 55 16 44  
## Albertson College Yes 587 479 158 38 62  
## F.Undergrad P.Undergrad Outstate Room.Board Books  
## Abilene Christian University 2885 537 7440 3300 450  
## Adelphi University 2683 1227 12280 6450 750  
## Adrian College 1036 99 11250 3750 400  
## Agnes Scott College 510 63 12960 5450 450  
## Alaska Pacific University 249 869 7560 4120 800  
## Albertson College 678 41 13500 3335 500  
## Personal PhD Terminal S.F.Ratio perc.alumni Expend  
## Abilene Christian University 2200 70 78 18.1 12 7041  
## Adelphi University 1500 29 30 12.2 16 10527  
## Adrian College 1165 53 66 12.9 30 8735  
## Agnes Scott College 875 92 97 7.7 37 19016  
## Alaska Pacific University 1500 76 72 11.9 2 10922  
## Albertson College 675 67 73 9.4 11 9727  
## Grad.Rate  
## Abilene Christian University 60  
## Adelphi University 56  
## Adrian College 54  
## Agnes Scott College 59  
## Alaska Pacific University 15  
## Albertson College 55

x <- model.matrix(Apps~., College)[,-1]  
y <- College$Apps  
lambda <- 10^seq(10, -2, length = 100)  
  
  
# Train test split  
set.seed(489)  
train = sample(1:nrow(x), nrow(x)/2)  
test = (-train)  
ytest = y[test]

###### (b) Fit a linear model using ordinary least squares on the training set, and report the test mean squared error obtained.

OLS\_lm <- lm(Apps~., data = College, subset = train)  
OLS\_lm

##   
## Call:  
## lm(formula = Apps ~ ., data = College, subset = train)  
##   
## Coefficients:  
## (Intercept) PrivateYes Accept Enroll Top10perc Top25perc   
## -544.41744 -170.52279 1.74160 -1.41087 38.28257 -6.06587   
## F.Undergrad P.Undergrad Outstate Room.Board Books Personal   
## 0.07306 0.08748 -0.08632 0.16650 0.06319 0.09351   
## PhD Terminal S.F.Ratio perc.alumni Expend Grad.Rate   
## -11.10782 2.19668 4.12585 3.56206 0.05095 1.92934

#Find the best lambda from our list via cross-validation  
cv.out <- cv.glmnet(x[train,], y[train], alpha = 0)  
cv.out

##   
## Call: cv.glmnet(x = x[train, ], y = y[train], alpha = 0)   
##   
## Measure: Mean-Squared Error   
##   
## Lambda Measure SE Nonzero  
## min 397.4 2103455 1270039 17  
## 1se 2554.6 3360297 2169940 17

#Best lambda  
bestlam <- cv.out$lambda.min  
bestlam

## [1] 397.4201

#Make predictions  
OLS.pred <- predict(OLS\_lm, newdata = College[test,])  
head(OLS.pred)

## Adelphi University Adrian College Albertson College   
## 3350.61158 1397.93516 608.67123   
## Albertus Magnus College Alderson-Broaddus College Allegheny College   
## 54.98646 686.22811 2922.74735

* MEAN SQUARED ERROR FOR OLS

#check Mean Squared Error  
mean((OLS.pred-ytest)^2)

## [1] 1403054

###### (c) Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test mean squared error obtained. Report the value of λ used in the model

ridge.mod <- glmnet(x[train,], y[train], alpha = 0, lambda = lambda)  
summary(ridge.mod)

## Length Class Mode   
## a0 100 -none- numeric  
## beta 1700 dgCMatrix S4   
## df 100 -none- numeric  
## dim 2 -none- numeric  
## lambda 100 -none- numeric  
## dev.ratio 100 -none- numeric  
## nulldev 1 -none- numeric  
## npasses 1 -none- numeric  
## jerr 1 -none- numeric  
## offset 1 -none- logical  
## call 5 -none- call   
## nobs 1 -none- numeric

#Find the best lambda from our list via cross-validation  
cv.out <- cv.glmnet(x[train,], y[train], alpha = 0)  
cv.out

##   
## Call: cv.glmnet(x = x[train, ], y = y[train], alpha = 0)   
##   
## Measure: Mean-Squared Error   
##   
## Lambda Measure SE Nonzero  
## min 397.4 2352967 1646036 17  
## 1se 3077.1 3903384 2937840 17

#Best lambda  
bestlam <- cv.out$lambda.min  
bestlam

## [1] 397.4201

#make predictions  
ridge.pred <- predict(ridge.mod, s = bestlam, newx = x[test,])  
head(ridge.pred)

## 1  
## Adelphi University 3000.9738  
## Adrian College 1164.0138  
## Albertson College 595.0114  
## Albertus Magnus College 317.8752  
## Alderson-Broaddus College 549.4096  
## Allegheny College 2677.7668

* MEAN SQUARED ERROR FOR RIDGE REGSRESSION

#Mean squared error  
mean((ridge.pred-ytest)^2)

## [1] 1298095

###### (d) Fit a lasso model on the training set, with fraction chosen by cross validation. Report the test mean squared error obtained, along with the number of non-zero coefficient estimates and the fraction.

lasso.mod <- glmnet(x[train,], y[train], alpha = 1, lambda = lambda)  
summary(lasso.mod)

## Length Class Mode   
## a0 100 -none- numeric  
## beta 1700 dgCMatrix S4   
## df 100 -none- numeric  
## dim 2 -none- numeric  
## lambda 100 -none- numeric  
## dev.ratio 100 -none- numeric  
## nulldev 1 -none- numeric  
## npasses 1 -none- numeric  
## jerr 1 -none- numeric  
## offset 1 -none- logical  
## call 5 -none- call   
## nobs 1 -none- numeric

lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test,])  
head(lasso.pred)

## 1  
## Adelphi University 2741.3266  
## Adrian College 1686.0656  
## Albertson College 998.9299  
## Albertus Magnus College 629.1303  
## Alderson-Broaddus College 875.6115  
## Allegheny College 2954.1927

* MEAN SQUARED ERROR FOR LASSO REGRESSION

mean((lasso.pred-ytest)^2)

## [1] 1798354

###### (e) Fit a PCR model on the training set, with no. of principal components M chosen by cross-validation. Report the test mean squared error obtained, along with the value of M selected by cross-validation.

set.seed(123)  
smp\_size <- floor(0.75 \* nrow(mtcars))  
train\_ind <- sample(seq\_len(nrow(College)), size = smp\_size)  
train\_p <- College[train\_ind, ]  
test\_p <- College[-train\_ind,c(1,3:18) ]  
y\_test=College[-train\_ind,2]  
  
require(pls)

## Loading required package: pls

##   
## Attaching package: 'pls'

## The following object is masked from 'package:caret':  
##   
## R2

## The following object is masked from 'package:stats':  
##   
## loadings

pcr\_model <- pcr(Apps~., data = train\_p,scale =TRUE, validation = "CV")  
summary(pcr\_model)

## Data: X dimension: 24 17   
## Y dimension: 24 1  
## Fit method: svdpc  
## Number of components considered: 17  
##   
## VALIDATION: RMSEP  
## Cross-validated using 10 random segments.  
## (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps  
## CV 2426 2779 1375 1389 1371 1509 1612  
## adjCV 2426 2749 1351 1365 1342 1477 1568  
## 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps  
## CV 1605 1625 1842 1786 1664 1373 1346  
## adjCV 1559 1574 1779 1714 1606 1305 1282  
## 14 comps 15 comps 16 comps 17 comps  
## CV 1180 936.9 1293 2503  
## adjCV 1126 890.5 1224 2386  
##   
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps  
## X 37.40 62.89 73.63 81.16 87.87 91.82 93.96 95.65  
## Apps 23.12 81.60 82.53 84.39 86.72 89.36 90.93 91.92  
## 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps 15 comps  
## X 97.14 97.93 98.65 99.09 99.50 99.79 99.96  
## Apps 92.87 95.00 95.49 98.20 98.27 98.44 98.99  
## 16 comps 17 comps  
## X 99.98 100.00  
## Apps 98.99 99.03

pcr\_pred <- predict(pcr\_model, test\_p, ncomp = 3)  
head(pcr\_pred)

## [1] 1930.6961 1451.1950 704.8014 2322.1893 815.2842 1231.9749

* MEAN SQUARED ERROR FOR Principal component regression (PCR)

mean((pcr\_pred - y\_test)^2)

## [1] 3664827

###### (f) Fit a PLS model on the training set, with M chosen by cross validation. Report the test error obtained, along with the value of M selected by cross-validation.

library(caret)  
# Compile cross-validation settings  
set.seed(100)  
myfolds <- createMultiFolds(train\_p$Apps, k = 5, times = 10)  
control <- trainControl("repeatedcv", index = myfolds, selectionFunction = "oneSE")  
  
# Train PLS model  
mod1 <- train(Apps ~ ., data = train\_p,  
 method = "pls",  
 metric = "RMSE",  
 tuneLength = 20,  
 trControl = control,  
 preProc = c("zv","center","scale"))  
  
summary(mod1)

## Data: X dimension: 24 17   
## Y dimension: 24 1  
## Fit method: oscorespls  
## Number of components considered: 8  
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
## X 30.84 61.11 69.40 75.50 81.94 85.84 89.89  
## .outcome 82.50 87.84 92.86 95.44 96.89 97.95 98.46  
## 8 comps  
## X 94.22  
## .outcome 98.71

The model results display the metrics in the model including: ncom (number of predictors in a subset,also is the value of M), root mean squared error (RMSE), R-squared, mean absolute error (MAE) etc. The lowest RMSE indicates the best subset size and inturn the best model.

mod1$results

## ncomp RMSE Rsquared MAE RMSESD RsquaredSD MAESD  
## 1 1 1084.9854 0.7133798 861.7848 521.4875 0.3403221 332.2876  
## 2 2 1180.7783 0.7798500 875.6709 564.5886 0.3116584 354.9473  
## 3 3 1230.5371 0.7154510 897.1544 561.0249 0.3305893 389.9395  
## 4 4 1191.8641 0.7283764 905.3304 475.5401 0.2813624 375.5198  
## 5 5 1132.9057 0.7626434 881.3255 443.3958 0.2641401 354.1532  
## 6 6 1074.9639 0.7841007 846.2498 384.8527 0.2348362 305.2852  
## 7 7 1030.9962 0.8037494 823.7655 344.5840 0.2158111 280.5208  
## 8 8 977.8485 0.8270842 799.8415 320.7414 0.1946075 256.7419  
## 9 9 961.4021 0.8474581 796.9018 369.8159 0.1860991 279.0550  
## 10 10 1003.3505 0.8466150 833.7763 403.9196 0.1928643 310.0666  
## 11 11 1066.8502 0.8373313 887.4662 421.3553 0.1989530 320.6783  
## 12 12 1137.5497 0.8239245 944.5730 463.5097 0.2021272 351.6782  
## 13 13 1229.7565 0.7731685 1007.7498 525.0526 0.2575168 399.2797  
## 14 14 1472.4609 0.7342725 1172.4776 702.0939 0.2797465 505.0709  
## 15 15 1936.8054 0.6771353 1474.1501 1218.4259 0.3113862 797.1762  
## 16 16 2309.6684 0.6659932 1770.6816 1720.6389 0.3131435 1165.2617

###### (g) Comment on the results obtained. Is there much difference among the test errors resulting from these five approaches?

* There is a noticeable difference between OLS, Ridge, PCR and PLS regression in terms of mean squared error whereby Ridge regression had the lowest mean squared error followed by PLS, OLS,Lasso and then Principal Component Regression had the highest mean squared error.