Final Exam

Tyler Reed

August 11, 2018

## Import Data

build\_data <- read.csv('https://raw.githubusercontent.com/da6813/summer18/master/final\_exam/final\_exam\_train.csv')  
results\_data <- read.csv('https://raw.githubusercontent.com/da6813/summer18/master/final\_exam/final\_exam\_test.csv')  
#summary(build\_data)  
#str(build\_data)

## Data Preprocessing

# Looking for Near Zero Variance

set.seed(123)  
  
nzv = nearZeroVar(build\_data)   
  
nonZeroVar = build\_data[, -nzv] # Removed 18 columns due to Near Zero Variance  
  
colnames(nonZeroVar)

## [1] "property\_id" "AG" "DCK" "LA" "LA2"   
## [6] "OP" "PA" "year\_built" "acres" "eff\_front"   
## [11] "eff\_depth" "value"

# Looking for Correlation

correlation = cor(nonZeroVar)  
  
correlation

## property\_id AG DCK LA  
## property\_id 1.0000000000 -0.014159670 0.067490599 0.062403441  
## AG -0.0141596700 1.000000000 0.015883094 0.242838631  
## DCK 0.0674905989 0.015883094 1.000000000 0.035545301  
## LA 0.0624034405 0.242838631 0.035545301 1.000000000  
## LA2 0.1764692303 -0.091464111 0.040575464 -0.567488976  
## OP 0.0010732078 0.002506967 -0.005923192 0.135189523  
## PA 0.0006070867 0.052644873 -0.221474737 0.040517521  
## year\_built 0.6458959574 -0.018336827 0.057707694 0.058558778  
## acres -0.0209590505 0.117923410 0.166904306 0.255596661  
## eff\_front 0.0588911103 -0.032500848 -0.009762960 0.003675632  
## eff\_depth 0.0463217577 -0.046882500 -0.046069928 -0.020984766  
## value 0.3351179266 0.160913673 0.130969370 0.358936896  
## LA2 OP PA year\_built acres  
## property\_id 0.17646923 0.001073208 0.0006070867 0.64589596 -0.02095905  
## AG -0.09146411 0.002506967 0.0526448729 -0.01833683 0.11792341  
## DCK 0.04057546 -0.005923192 -0.2214747371 0.05770769 0.16690431  
## LA -0.56748898 0.135189523 0.0405175214 0.05855878 0.25559666  
## LA2 1.00000000 -0.019512397 0.0960291733 0.16711068 0.23776431  
## OP -0.01951240 1.000000000 -0.1503321308 0.09299420 0.16008571  
## PA 0.09602917 -0.150332131 1.0000000000 0.02287640 0.11131042  
## year\_built 0.16711068 0.092994196 0.0228764045 1.00000000 0.04468973  
## acres 0.23776431 0.160085708 0.1113104155 0.04468973 1.00000000  
## eff\_front -0.14272994 -0.109845981 -0.0337652161 -0.24634461 -0.37626747  
## eff\_depth -0.19540534 -0.129766235 -0.0338855785 -0.25039679 -0.42244967  
## value 0.49419325 0.185266421 0.1147090630 0.38305941 0.57947808  
## eff\_front eff\_depth value  
## property\_id 0.058891110 0.04632176 0.3351179  
## AG -0.032500848 -0.04688250 0.1609137  
## DCK -0.009762960 -0.04606993 0.1309694  
## LA 0.003675632 -0.02098477 0.3589369  
## LA2 -0.142729940 -0.19540534 0.4941933  
## OP -0.109845981 -0.12976623 0.1852664  
## PA -0.033765216 -0.03388558 0.1147091  
## year\_built -0.246344615 -0.25039679 0.3830594  
## acres -0.376267472 -0.42244967 0.5794781  
## eff\_front 1.000000000 0.95487964 -0.1980661  
## eff\_depth 0.954879644 1.00000000 -0.3065391  
## value -0.198066078 -0.30653911 1.0000000

nonCorrVar = nonZeroVar[,-11] # High correlation exists between eff\_front and eff\_depth predictors. I removed eff\_depth.  
colnames(nonCorrVar)

## [1] "property\_id" "AG" "DCK" "LA" "LA2"   
## [6] "OP" "PA" "year\_built" "acres" "eff\_front"   
## [11] "value"

# Looking for Skewness

skewValues = apply(nonCorrVar, 2, skewness)  
skewValues

## property\_id AG DCK LA LA2 OP   
## 0.00000000 -2.52699552 2.53540781 0.21524681 0.91690484 3.16944531   
## PA year\_built acres eff\_front value   
## 2.78578083 0.44408992 1.50644990 -0.37014122 0.09029818

# Settling on a final set of predictors

finalData = nonCorrVar[-1] # Removed the property\_ID predictor because it adds no value to my model.  
colnames(finalData)

## [1] "AG" "DCK" "LA" "LA2" "OP"   
## [6] "PA" "year\_built" "acres" "eff\_front" "value"

featurePlot(x = finalData[, 1:9], y = finalData$value,  
 plot = "scatter",  
 type = c("p", "smooth"),  
 layout = c(3,1))

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : at -4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : radius 19.892

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : all data on boundary of neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : pseudoinverse used at -4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : neighborhood radius 4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : reciprocal condition number 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : zero-width neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : at -4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : radius 19.892

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : all data on boundary of neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : pseudoinverse used at -4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : neighborhood radius 4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : reciprocal condition number 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : zero-width neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : at -4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : radius 19.892

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : all data on boundary of neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : pseudoinverse used at -4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : neighborhood radius 4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : reciprocal condition number 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : zero-width neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : at -4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : radius 19.892

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : all data on boundary of neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : pseudoinverse used at -4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : neighborhood radius 4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : reciprocal condition number 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : zero-width neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : at -4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : radius 19.892

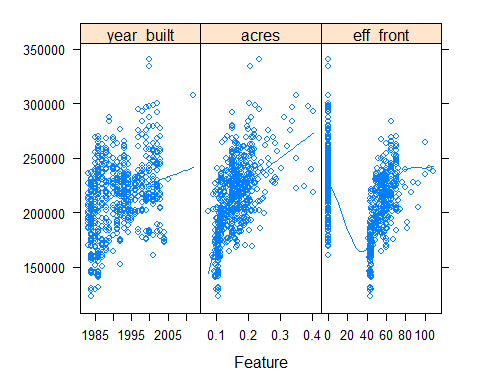
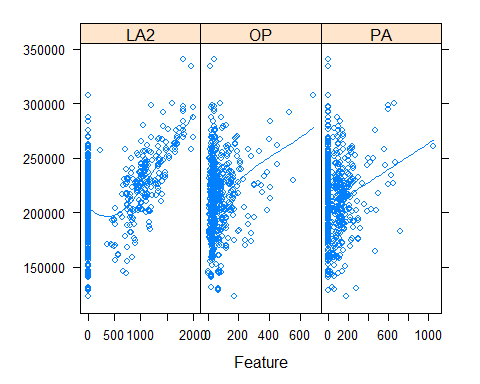
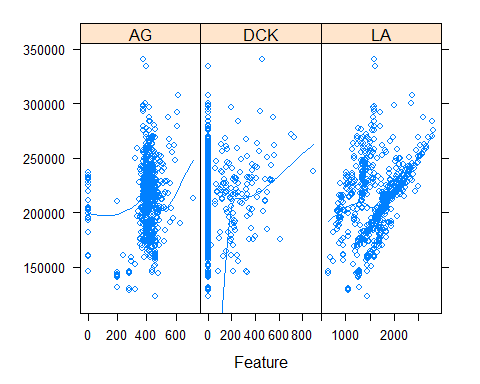
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : all data on boundary of neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : pseudoinverse used at -4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : neighborhood radius 4.46

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : reciprocal condition number 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## FALSE, : zero-width neighborhood. make span bigger



## Sample Selection - Generate a Training and Test Set

set.seed(123)  
train = createDataPartition(finalData[,1], p = .75, list = FALSE)  
  
propTrain = finalData[train,]  
propTest = finalData[-train,]  
  
finalResults = sqldf("select AG, DCK, LA, LA2,  
 OP, PA, year\_built, acres,  
 eff\_front from results\_data") # Used SQL to match predictors in the results\_data dataset.

# Model input - I tried different predictor combinations, but settled on the ones below.

model = (value ~ AG + DCK + LA + LA2 + OP + PA + year\_built + acres + eff\_front)

Random Forest - training

set.seed(123)  
control = trainControl(method = "repeatedcv", number = 10, repeats = 10)  
metric = "RMSE"  
tunegrid = expand.grid(.mtry=c(1:9))  
  
rfTune = train(model, data=propTrain,  
 method="rf",  
 tuneGrid=tunegrid,  
 trControl=control)  
  
rfTune

## Random Forest   
##   
## 458 samples  
## 9 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 413, 411, 414, 410, 411, 412, ...   
## Resampling results across tuning parameters:  
##   
## mtry RMSE Rsquared MAE   
## 1 16152.44 0.8144597 12001.757  
## 2 13201.77 0.8620751 9447.993  
## 3 12150.21 0.8789429 8551.700  
## 4 11596.74 0.8874695 8034.290  
## 5 11314.69 0.8914981 7726.561  
## 6 11224.54 0.8920818 7570.421  
## 7 11220.48 0.8913210 7531.272  
## 8 11273.27 0.8896532 7560.400  
## 9 11461.92 0.8855339 7682.983  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was mtry = 7.

Random Forest - testing

rfTest = predict(rfTune, newdata = propTest)  
  
RMSE(rfTest, propTest$value)

## [1] 11423.86

rfValidate = predict(rfTune, newdata = finalResults)  
  
rfValidate

## 1 2 3 4 5   
## 196068.1 215678.7 209124.5 249260.2 223816.5

Logistic Regression - training

set.seed(123)  
glmTune = train(model, data = propTrain,  
 method = "glm",  
 preProc = c("center", "scale"),  
 trControl = trainControl(method = "repeatedcv", repeats = 10))  
  
glmTune

## Generalized Linear Model   
##   
## 458 samples  
## 9 predictor  
##   
## Pre-processing: centered (9), scaled (9)   
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 413, 411, 414, 410, 411, 412, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 10996.49 0.8945986 8402.936

Logistic Regression - testing

glmTest = predict(glmTune, newdata = propTest)  
  
RMSE(glmTest, propTest$value)

## [1] 11369.63

glmValidate = predict(glmTune, newdata = finalResults)  
  
glmValidate

## 1 2 3 4 5   
## 196043.2 209562.8 204767.7 264434.9 221929.6

PCA - training

pcaTune = train(model, data = propTrain,  
 method = "glm",  
 preProc = c("center", "scale", "pca"),  
 trControl = trainControl(method = "repeatedcv", number = 10, repeats = 10))   
pcaTune

## Generalized Linear Model   
##   
## 458 samples  
## 9 predictor  
##   
## Pre-processing: centered (9), scaled (9), principal component  
## signal extraction (9)   
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 412, 412, 411, 414, 412, 413, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 18659.54 0.6968293 14479.56

PCA - testing

pcaTest = predict(pcaTune, newdata = propTest)  
  
RMSE(pcaTest, propTest$value)

## [1] 16941.63

PLS - training

plsTune = train(model, data = propTrain,  
 method = "pls",  
 preProc = c("center", "scale"),  
 trControl = trainControl(method = "repeatedcv", number = 10, repeats = 10))  
plsTune

## Partial Least Squares   
##   
## 458 samples  
## 9 predictor  
##   
## Pre-processing: centered (9), scaled (9)   
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 414, 412, 411, 411, 412, 412, ...   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared MAE   
## 1 19543.16 0.6636189 15354.235  
## 2 13789.21 0.8332899 10948.827  
## 3 11905.65 0.8761530 9266.574  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 3.

PLS - testing

plsTest = predict(plsTune, newdata = propTest)  
  
RMSE(plsTest, propTest$value)

## [1] 11755.7

Ridge - training

ridgeGrid = data.frame(.lambda = seq(0, .1, length = 15))  
ridgeTune = train(model, data = propTrain,   
 method = "ridge",   
 preProc = c("center", "scale"),   
 trControl = trainControl(method = "repeatedcv", number = 10, repeats = 10),  
 tuneGrid = ridgeGrid)  
ridgeTune

## Ridge Regression   
##   
## 458 samples  
## 9 predictor  
##   
## Pre-processing: centered (9), scaled (9)   
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 412, 412, 413, 411, 412, 413, ...   
## Resampling results across tuning parameters:  
##   
## lambda RMSE Rsquared MAE   
## 0.000000000 10973.36 0.8954940 8398.332  
## 0.007142857 10974.77 0.8953332 8423.580  
## 0.014285714 10987.11 0.8949986 8454.682  
## 0.021428571 11008.62 0.8945133 8490.120  
## 0.028571429 11037.81 0.8938979 8528.861  
## 0.035714286 11073.40 0.8931700 8570.374  
## 0.042857143 11114.32 0.8923450 8613.590  
## 0.050000000 11159.64 0.8914364 8657.235  
## 0.057142857 11208.57 0.8904560 8700.667  
## 0.064285714 11260.46 0.8894141 8743.768  
## 0.071428571 11314.74 0.8883197 8787.126  
## 0.078571429 11370.91 0.8871808 8830.688  
## 0.085714286 11428.57 0.8860043 8873.459  
## 0.092857143 11487.37 0.8847962 8915.696  
## 0.100000000 11547.00 0.8835620 8958.301  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was lambda = 0.

Ridge - testing

ridgeTest = predict(ridgeTune, newdata = propTest)  
  
RMSE(ridgeTest, propTest$value)

## [1] 11369.63

ridgeValidate = predict(ridgeTune, newdata = finalResults)  
  
ridgeValidate

## 1 2 3 4 5   
## 196043.2 209562.8 204767.7 264434.9 221929.6

SVM - training

set.seed(123)  
svmTune = train(model, data = propTrain,  
 method = "svmRadial",  
 preProc = c("center", "scale"),  
 tuneLength = 10,  
 trControl = trainControl(method = "repeatedcv", number = 10, repeats = 10))  
svmTune

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 458 samples  
## 9 predictor  
##   
## Pre-processing: centered (9), scaled (9)   
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 413, 411, 414, 410, 411, 412, ...   
## Resampling results across tuning parameters:  
##   
## C RMSE Rsquared MAE   
## 0.25 15123.01 0.8160357 10033.951  
## 0.50 13690.38 0.8443962 8916.296  
## 1.00 12854.90 0.8599272 8444.636  
## 2.00 12445.07 0.8668573 8196.721  
## 4.00 12450.02 0.8656256 8282.900  
## 8.00 12532.14 0.8629631 8388.324  
## 16.00 12769.56 0.8569199 8631.844  
## 32.00 13273.53 0.8448296 8947.079  
## 64.00 13907.71 0.8295335 9414.639  
## 128.00 14681.67 0.8111800 9994.120  
##   
## Tuning parameter 'sigma' was held constant at a value of 0.1229649  
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were sigma = 0.1229649 and C = 2.

SVM - testing

svmTest = predict(svmTune, newdata = propTest)  
  
RMSE(svmTest, propTest$value)

## [1] 10369.85

svmValidate = predict(svmTune, newdata = finalResults)  
  
svmValidate

## [1] 219352.0 213867.9 213964.2 249300.4 222608.2

## Model Selection

rfTest

## 2 3 5 7 8 9 11 15   
## 214891.6 192498.2 185644.9 205423.5 216541.3 241892.1 275180.1 247675.7   
## 17 18 19 20 34 36 38 40   
## 244780.9 250592.2 270507.7 252194.2 249718.6 215439.9 225127.2 198453.6   
## 58 59 61 62 68 71 72 76   
## 221757.5 192192.4 216089.1 232651.1 213785.7 226362.9 220532.5 191165.0   
## 83 84 86 88 91 97 98 99   
## 205682.1 214831.2 188459.5 207261.3 225138.3 261012.2 226426.9 243503.8   
## 101 102 104 107 108 118 124 126   
## 235432.7 210747.9 219592.6 227318.7 192758.3 236599.2 196079.6 228704.2   
## 128 139 140 143 147 148 157 161   
## 237103.3 207018.3 215672.3 255933.4 175825.8 177151.5 242281.0 183188.3   
## 167 168 169 171 173 175 185 193   
## 145129.6 135552.9 179938.9 159038.2 149507.9 178074.5 158123.4 162190.1   
## 198 208 211 216 222 225 231 241   
## 144355.9 144589.7 179797.3 135349.7 161906.4 181378.9 177085.7 182762.1   
## 243 244 248 252 258 262 264 277   
## 143634.3 160272.4 175568.0 229283.3 183867.5 196237.6 233045.1 194704.2   
## 278 283 287 288 291 297 301 303   
## 220316.4 197388.5 202117.8 189319.6 208237.3 225443.1 194346.8 245595.9   
## 308 313 318 319 321 322 338 339   
## 200305.3 222734.7 208910.2 198550.5 210452.1 193588.9 216348.3 178143.1   
## 341 342 344 350 353 356 359 360   
## 190272.3 164043.5 173488.3 220770.3 224633.9 201020.2 208794.4 243626.9   
## 362 373 374 394 396 402 409 415   
## 200409.8 212765.1 245975.9 223563.4 223978.1 259281.8 240061.7 207655.5   
## 419 420 425 427 429 431 432 439   
## 216232.2 234774.4 226759.0 213142.0 224649.2 244803.5 266517.6 236735.5   
## 443 444 447 452 456 460 468 477   
## 223764.0 219904.1 202698.3 218420.4 200135.6 251231.9 195846.1 228440.8   
## 486 498 503 504 509 512 516 519   
## 212652.4 226748.3 245393.7 212251.1 251158.3 223255.2 265353.4 217085.9   
## 523 524 525 528 535 538 541 544   
## 211052.1 225993.3 225411.5 212450.2 226834.7 227863.9 237884.5 229889.5   
## 547 551 558 560 561 565 566 573   
## 217468.3 226765.9 267549.4 306335.3 276900.6 310692.1 292998.1 312384.2   
## 574 577 588 594 599 606   
## 266158.2 266676.8 238454.7 226170.7 245488.7 230210.3

RMSE(rfTest, propTest$value)

## [1] 11423.86

glmTune

## Generalized Linear Model   
##   
## 458 samples  
## 9 predictor  
##   
## Pre-processing: centered (9), scaled (9)   
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 413, 411, 414, 410, 411, 412, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 10996.49 0.8945986 8402.936

RMSE(glmTest, propTest$value)

## [1] 11369.63

pcaTune

## Generalized Linear Model   
##   
## 458 samples  
## 9 predictor  
##   
## Pre-processing: centered (9), scaled (9), principal component  
## signal extraction (9)   
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 412, 412, 411, 414, 412, 413, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 18659.54 0.6968293 14479.56

RMSE(pcaTest, propTest$value)

## [1] 16941.63

plsTune

## Partial Least Squares   
##   
## 458 samples  
## 9 predictor  
##   
## Pre-processing: centered (9), scaled (9)   
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 414, 412, 411, 411, 412, 412, ...   
## Resampling results across tuning parameters:  
##   
## ncomp RMSE Rsquared MAE   
## 1 19543.16 0.6636189 15354.235  
## 2 13789.21 0.8332899 10948.827  
## 3 11905.65 0.8761530 9266.574  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was ncomp = 3.

RMSE(plsTest, propTest$value)

## [1] 11755.7

ridgeTune

## Ridge Regression   
##   
## 458 samples  
## 9 predictor  
##   
## Pre-processing: centered (9), scaled (9)   
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 412, 412, 413, 411, 412, 413, ...   
## Resampling results across tuning parameters:  
##   
## lambda RMSE Rsquared MAE   
## 0.000000000 10973.36 0.8954940 8398.332  
## 0.007142857 10974.77 0.8953332 8423.580  
## 0.014285714 10987.11 0.8949986 8454.682  
## 0.021428571 11008.62 0.8945133 8490.120  
## 0.028571429 11037.81 0.8938979 8528.861  
## 0.035714286 11073.40 0.8931700 8570.374  
## 0.042857143 11114.32 0.8923450 8613.590  
## 0.050000000 11159.64 0.8914364 8657.235  
## 0.057142857 11208.57 0.8904560 8700.667  
## 0.064285714 11260.46 0.8894141 8743.768  
## 0.071428571 11314.74 0.8883197 8787.126  
## 0.078571429 11370.91 0.8871808 8830.688  
## 0.085714286 11428.57 0.8860043 8873.459  
## 0.092857143 11487.37 0.8847962 8915.696  
## 0.100000000 11547.00 0.8835620 8958.301  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was lambda = 0.

RMSE(ridgeTest, propTest$value)

## [1] 11369.63

svmTune

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 458 samples  
## 9 predictor  
##   
## Pre-processing: centered (9), scaled (9)   
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 413, 411, 414, 410, 411, 412, ...   
## Resampling results across tuning parameters:  
##   
## C RMSE Rsquared MAE   
## 0.25 15123.01 0.8160357 10033.951  
## 0.50 13690.38 0.8443962 8916.296  
## 1.00 12854.90 0.8599272 8444.636  
## 2.00 12445.07 0.8668573 8196.721  
## 4.00 12450.02 0.8656256 8282.900  
## 8.00 12532.14 0.8629631 8388.324  
## 16.00 12769.56 0.8569199 8631.844  
## 32.00 13273.53 0.8448296 8947.079  
## 64.00 13907.71 0.8295335 9414.639  
## 128.00 14681.67 0.8111800 9994.120  
##   
## Tuning parameter 'sigma' was held constant at a value of 0.1229649  
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were sigma = 0.1229649 and C = 2.

RMSE(svmTest, propTest$value)

## [1] 10369.85

## What is the expected accuracy of your approach (R^2 / RMSE / MAE etc.)?

For this exercise, I choose the GLM model due to its simplicity and relatively good RMSE of 10996.49, R^2 of 89.46%, and MAE of 8402.936.

Using RMSE, the expected accuracy is 11369.63.

results <- resamples(list(RF = rfTune, GLM = glmTune, PCA = pcaTune, PLS = plsTune, RIDGE = ridgeTune, SVM = svmTune))  
summary(results)

##   
## Call:  
## summary.resamples(object = results)  
##   
## Models: RF, GLM, PCA, PLS, RIDGE, SVM   
## Number of resamples: 100   
##   
## MAE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## RF 5146.191 6762.091 7484.927 7531.272 8171.744 10775.60 0  
## GLM 6107.136 7687.613 8440.467 8402.936 9035.424 10868.09 0  
## PCA 10996.348 13474.701 14126.633 14479.562 15306.096 18952.73 0  
## PLS 6581.646 8428.338 9190.642 9266.574 10075.436 14137.49 0  
## RIDGE 6096.424 7786.145 8394.100 8398.332 8982.524 12066.84 0  
## SVM 5308.002 7231.735 8187.521 8196.721 9026.644 11446.29 0  
##   
## RMSE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## RF 7301.484 9857.177 11238.24 11220.48 12213.58 16952.44 0  
## GLM 7296.168 10070.111 10833.16 10996.49 12002.27 14727.07 0  
## PCA 14392.285 17072.847 18177.04 18659.54 20095.41 25218.05 0  
## PLS 8081.122 10779.137 11836.98 11905.65 12883.30 18118.02 0  
## RIDGE 7591.634 9919.364 11038.50 10973.36 11812.93 16370.59 0  
## SVM 7450.032 10382.702 12179.53 12445.07 14038.09 19163.57 0  
##   
## Rsquared   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## RF 0.7747851 0.8754179 0.8936323 0.8913210 0.9110306 0.9534170 0  
## GLM 0.8163243 0.8762111 0.8949614 0.8945986 0.9158498 0.9479579 0  
## PCA 0.4942572 0.6570215 0.7152091 0.6968293 0.7459998 0.8202867 0  
## PLS 0.7369879 0.8543220 0.8832600 0.8761530 0.9016438 0.9535589 0  
## RIDGE 0.7964237 0.8758976 0.8985219 0.8954940 0.9155490 0.9566683 0  
## SVM 0.7389312 0.8351027 0.8681635 0.8668573 0.9016733 0.9549596 0

RMSE(glmTest, propTest$value) #RMSE for my glmTest prediction.

## [1] 11369.63

## Predict the House Values for results\_data

glmValidate = predict(glmTune, newdata = finalResults)  
  
glmValidate

## 1 2 3 4 5   
## 196043.2 209562.8 204767.7 264434.9 221929.6