# oj_solution.R

*user*

*2020-03-18*

```r
###############################################################################
## Course: Machine Learning for Economists and Business Analysts
## Topic: Self-Study - Exercise 2 - Orange Juice
###############################################################################
rm(list = ls())

library(glmnet)
```

```
## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-18
```

```r
library(grf)
```

```
## Warning: package 'grf' was built under R version 3.6.1
```

```r
library(rpart)
library(rpart.plot)
```

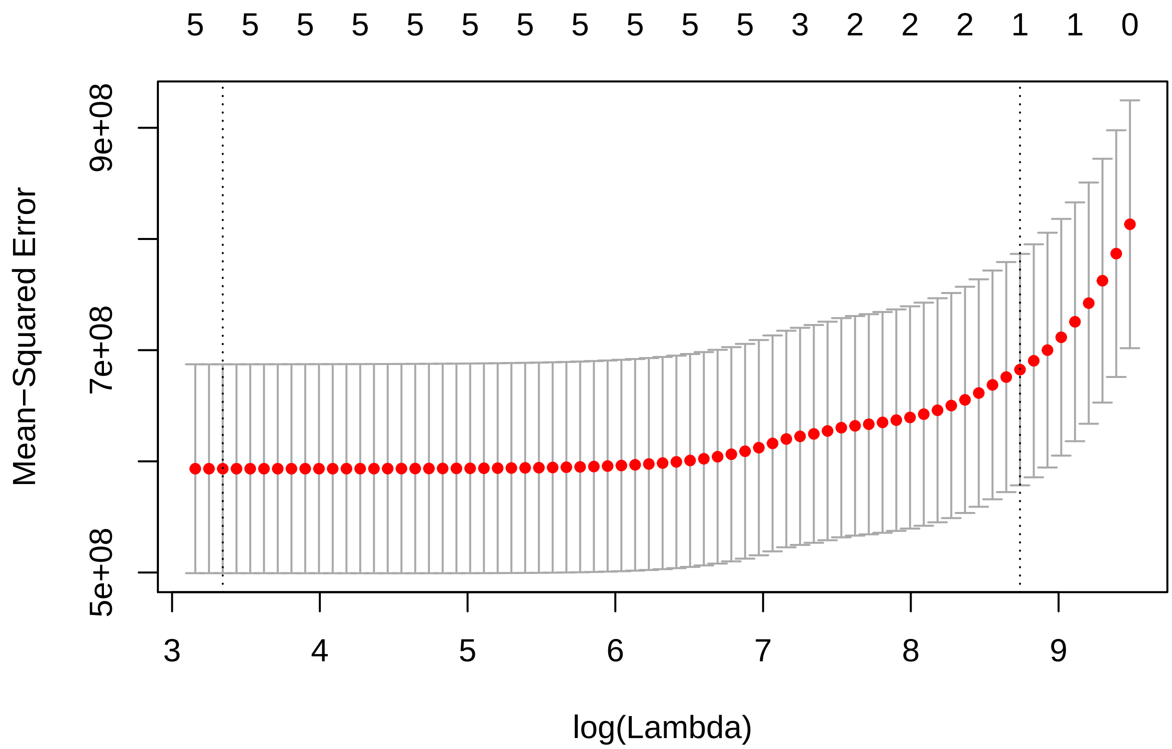
```
## Warning: package 'rpart.plot' was built under R version 3.6.1
```

```r
#getwd()
#setwd("")

juice <- read.csv("juice.csv", sep = ",")
new_grocery <- read.csv("new_grocery.csv", sep = ",")

###############################################################################
# Task 1
###############################################################################

# Generate a missing dummy
missing <- (is.na(juice$price) == TRUE)
new_missing <- (is.na(new_grocery$price) == TRUE)

# Replace missing prices with zero
juice$price[is.na(juice$price)] <-0
new_grocery$price[is.na(new_grocery$price)] <-0

# Generate Dummies for Brands
brand_1 <- (juice$brand == "minute.maid")
brand_2 <- (juice$brand == "dominicks")
brand_3 <- (juice$brand == "tropicana")
new_brand_1 <- (new_grocery$brand == "minute.maid")
new_brand_2 <- (new_grocery$brand == "dominicks")
new_brand_3 <- (new_grocery$brand == "tropicana")

# Generate outcome and control variables
```

```r
y <- as.matrix(juice$sales)
x <- as.matrix(cbind(juice$price, missing, brand_1, brand_2,
                     brand_3, juice$feat))
colnames(x) <- c("price", "missing", "minute.maid", "dominicks",
                 "tropicana", "featured")
new_x <- as.matrix(cbind(new_grocery$price, new_missing, new_brand_1,
                         new_brand_2, new_brand_3, new_grocery$feat))
colnames(new_x) <- c("price", "missing", "minute.maid", "dominicks",
                     "tropicana", "featured")

###############################################################################
# Task 2
###############################################################################

set.seed(123456789)
# Generate variable with the rows in training data
size <- floor(0.5 * nrow(juice))
training_set <- sample(seq_len(nrow(juice)), size = size)

###############################################################################
# Task 3
###############################################################################

## Lasso ##
set.seed(27112019)
lasso.cv <- cv.glmnet(x[training_set,],y[training_set],
                      type.measure = "mse", family = "gaussian",
                      nfolds = 10, alpha = 1)
coef_lasso1 <- coef(lasso.cv, s = "lambda.min")
print(coef_lasso1) # plot Lasso coefficients
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                     1
## (Intercept)  46870.28
## price       -15697.63
## missing     -35957.33
## minute.maid      .
## dominicks    -6891.85
## tropicana     7302.81
## featured     25376.33
```

```r
plot(lasso.cv) # plot CV-MSE
```




```r
# Fitted values
predlasso <- predict(lasso.cv, newx = x, s = lasso.cv$lambda.min)

# Calculate the MSE
MSElasso <- mean((y[-training_set] - predlasso[-training_set])^2)
R2lasso <- round(1- MSElasso/var(y[-training_set]), digits = 3)
print(paste0("R-squared Lasso: ", R2lasso))
```

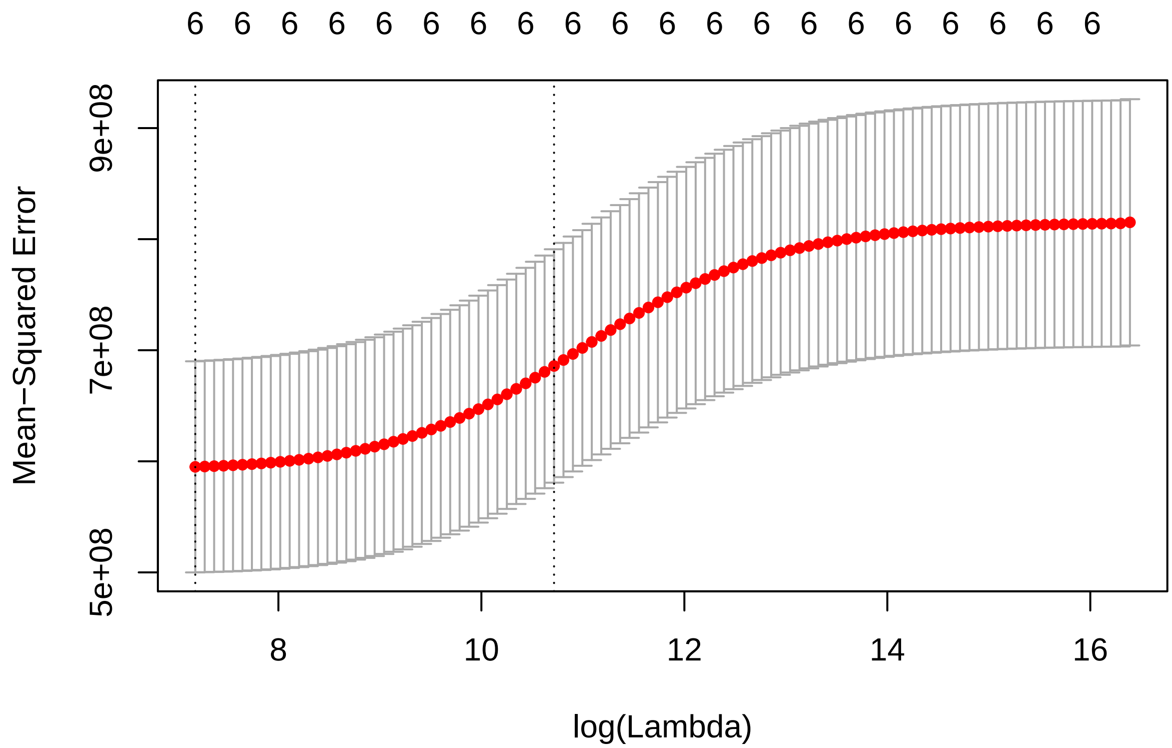
```
## [1] "R-squared Lasso: 0.282"
```

```r
#####################################

## Ridge ##
set.seed(27112019)
ridge.cv <- cv.glmnet(x[training_set,],y[training_set],
                      type.measure = "mse",family = "gaussian",
                      nfolds = 10, alpha = 0)
coef_ridge <- coef(ridge.cv, s = "lambda.min")
print(coef_ridge) # plot Lasso coefficients
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)   41510.4420
## price        -13297.7058
## missing      -29324.5839
## minute.maid     -66.5484
## dominicks     -5640.8366
```

```
## tropicana     5770.3250
## featured     25078.6543
```

```r
plot(ridge.cv) # plot CV-MSE
```




```r
# Fitted values
predridge <- predict(ridge.cv, newx = x, s = ridge.cv$lambda.min)

# Calculate the MSE
MSEridge <- mean((y[-training_set] - predridge[-training_set])^2)
R2ridge <- round(1- MSEridge/var(y[-training_set]), digits = 3)

print(paste0("R-squared Lasso: ", R2lasso))
```

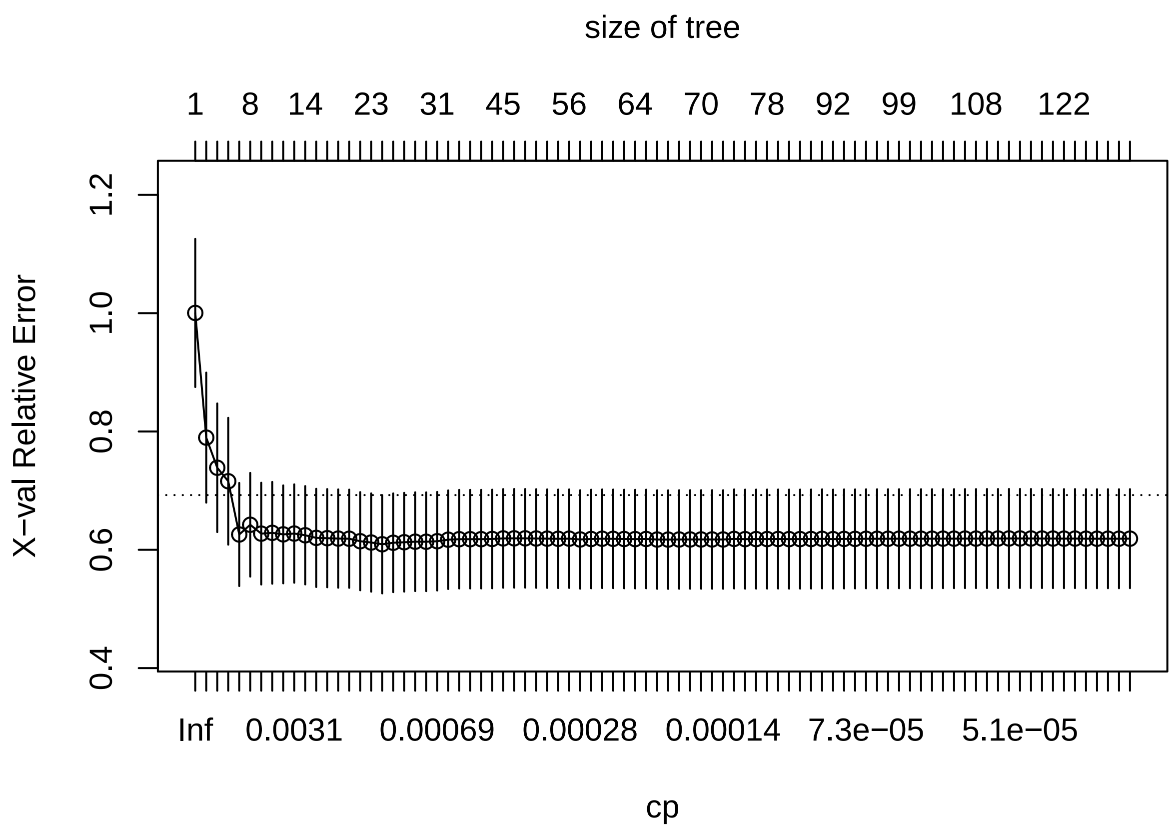
```
## [1] "R-squared Lasso: 0.282"
```

```r
print(paste0("R-squared Ridge: ", R2ridge))
```

```
## [1] "R-squared Ridge: 0.281"
```

```r
#####################################

## Tree ##
set.seed(27112019)

# Prepare data for tree estimator
outcome <- y[training_set]
tree_data <-  data.frame(outcome, x[training_set,])
```

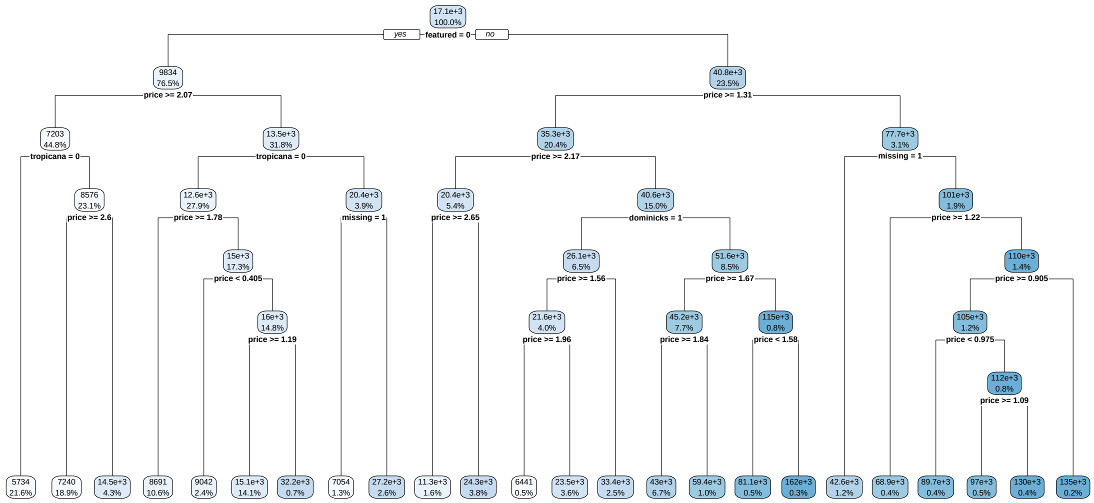
```r
deep_tree <- rpart(formula = outcome ~., data = tree_data, method = "anova",
                   xval = 10, y = TRUE,
                   control = rpart.control(cp = 0.00002, minbucket=10))

# Plot CV-MSE
plotcp(deep_tree)
```




```r
# Optimal tree size
op.index <- which.min(deep_tree$cptable[, "xerror"])
print(paste0("Optimal number of splits: ", deep_tree$cptable[op.index,2]))
```

```
## [1] "Optimal number of splits: 23"
```

```r
## Select the Tree that Minimises CV-MSE
# Get cp-value that corresponds to optimal tree size
cp.vals <- deep_tree$cptable[op.index, "CP"]

# Prune the deep tree
pruned_tree <- prune(deep_tree, cp = cp.vals)

## Plot tree structure
rpart.plot(pruned_tree,digits=3)
```




```r
# Fitted values
predtree <- predict(pruned_tree, newdata= as.data.frame(x))

# Calculate the MSE
MSEtree <- mean((y[-training_set] - predtree[-training_set])^2)
R2tree <- round(1- MSEtree/var(y[-training_set]), digits = 3)

print(paste0("R-squared Lasso: ", R2lasso))
```

```
## [1] "R-squared Lasso: 0.282"
```

```r
print(paste0("R-squared Ridge: ", R2ridge))
```

```
## [1] "R-squared Ridge: 0.281"
```

```r
print(paste0("R-squared Tree: ", R2tree))
```

```
## [1] "R-squared Tree: 0.413"
```

```r
######################################

## Random Forest ##
set.seed(27112019)

rep <- 1000 # number of trees
cov <- 2/3 # share of covariates
frac <- 1/2 # fraction of subsample
min_obs <- 10 # max. size of terminal leaves in trees
```

```r
# Build Forest
forest <- regression_forest(x[training_set,],y[training_set,],
              mtry = floor(cov*ncol(x)), sample.fraction = frac,
              num.trees = rep,min.node.size = min_obs, honesty=FALSE)

# Fitted values
predforest <- predict(forest, newdata=x)$predictions

# Calculate MSE
MSEforest <- mean((y[-training_set] - predforest[-training_set])^2)
R2forest <- round(1- MSEforest/var(y[-training_set]), digits = 3)

print(paste0("R-squared Lasso: ", R2lasso))
```

```
## [1] "R-squared Lasso: 0.282"
```

```r
print(paste0("R-squared Ridge: ", R2ridge))
```

```
## [1] "R-squared Ridge: 0.281"
```

```r
print(paste0("R-squared Tree: ", R2tree))
```

```
## [1] "R-squared Tree: 0.413"
```

```r
print(paste0("R-squared Forest: ", R2forest))
```

```
## [1] "R-squared Forest: 0.431"
```

```r
###############################################################################
# Task 4
###############################################################################

# Fitted values new data
pred_new <- predict(forest, newdata=new_x)$predictions
id_new <- as.matrix(new_grocery$id)
write.csv(cbind(id_new,pred_new),"strittmatter.csv")
```