DS740: Midterm

Eli (Ilya) Bolotin

6/23/2019

Project Summary

The purpose of this project is to model a dataset using appropriate machine learning methods in order to produce maximally accurate predictions that have relevant non-technical applications. Said differently, the primary technical goals are to:

- Make reasonably accurate predictions of the response variable given a set of predictors.
- Identify the most influential predictor variables.

The primary *non*-technical goals are to:

- Compare two response variables in the dataset.
- Briefly explain how the predictive capabilities of the model may benefit individuals with some relationship to the context of the data or field from which it originated.

Project Structure and Results

This project is structured into two parts that commence directly below. There is an explanation of results (conclusion) at the end of Part 2. To summarize the conclusion:

- The final predictive model achieves ~96% predictive accuracy on the full dataset.
- 5 predictors (out of 14) reduce 80% of the total variation in the model.
- RandomForest regression appears to be well-suited to further modeling test data.

Part 1

The dataset I selected for this project was the **2004 New Car and Truck** data (http://ww2.amstat.org/publications/jse/datasets/04cars.txt). This dataset has the following characteristics (as addressed in the assignment):

- 14 predictors, multiple missing values (consider variables and/or observations)
- Factor response: Type (AWD is all-wheel drive, RWD is read-wheel drive, and Other)
- Quantitative response: Retailprice

Part 1a. Response variable selection

I selected *Retailprice* as the (quantitative) response variable for this exercise. Regressing *Retailprice* on other vehicle attributes (variables) is useful for the purpose of understanding which attributes are most influential (on retail price). And this is ultimately the question that I seek to answer: of the given predictors in this dataset, which of them contribute most to the retail price of a vehicle?

Part 1b. Practical purposes to data analysis and model creation

There are numerous practical purposes to this data analysis. For example, by understanding the relationship between the response variable and predictors in this dataset, it is possible for:

- Car dealership salespeople to optimally price vehicles.
- Auto manufacturers to design vehicles with specific features to maximize retail value.
- Consumers to estimate the true value of a vehicle they are considering purchasing.

The primary audience for this data analysis are individuals that are concerned with the retail price of vehicles. Such individuals may work in the automotive manufacturing industry, in the auto sales industry, or are simply consumers of vehicles that are looking to maximize the return on their investment.

Part 2

Prior to answering Part 2a, it is necessary to first analyze the data.

Read in the data

```r
cars <- read.csv("04cars.csv")
```

Remove missing values and create x and y groups.

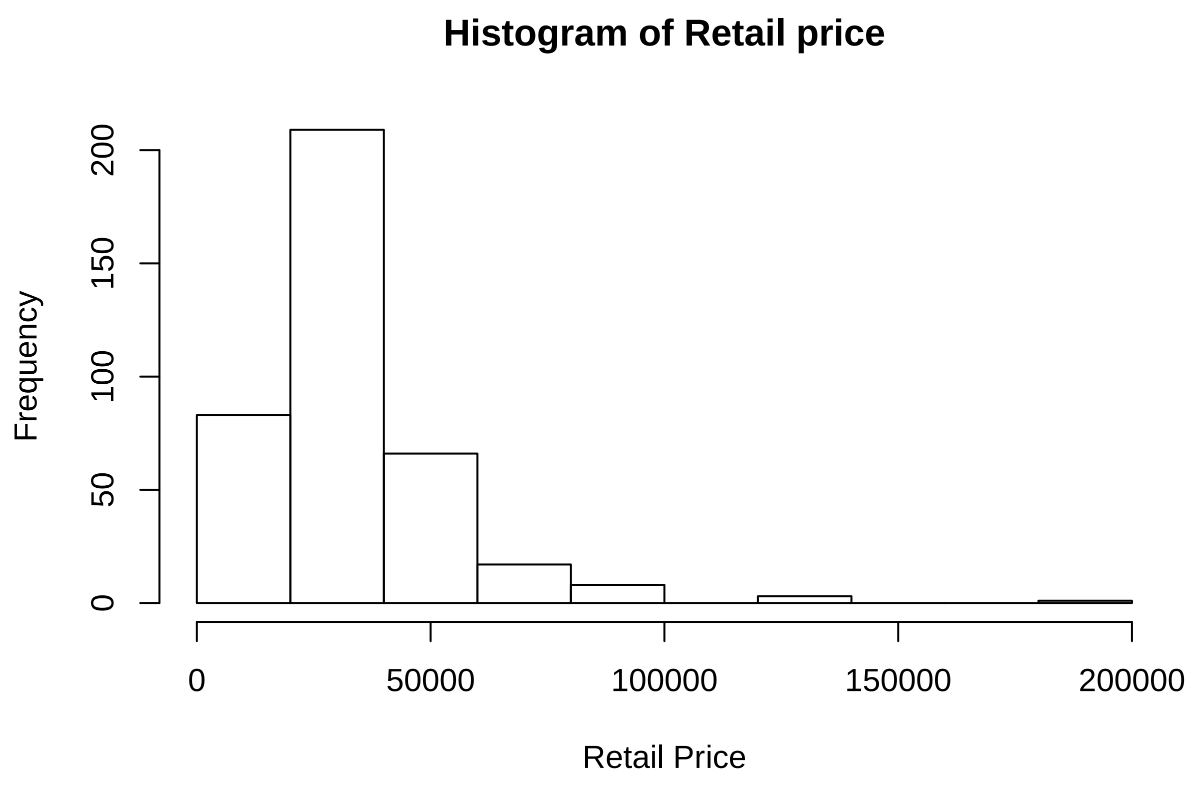
```r
# Get count of observations
n = dim(cars)[1]

# Get count observations without missing values
n.complete <- length(which(complete.cases(cars)))

# Return cleaned cars dataset without Type
cars <- cars[complete.cases(cars),-1]

# Convert height variable to integer
cars$Height <- as.integer(cars$Height)

# define x, y
y = cars$Retailprice
x = model.matrix(Retailprice~., data=cars)[,c(-1)]
```

Summary of operations above:

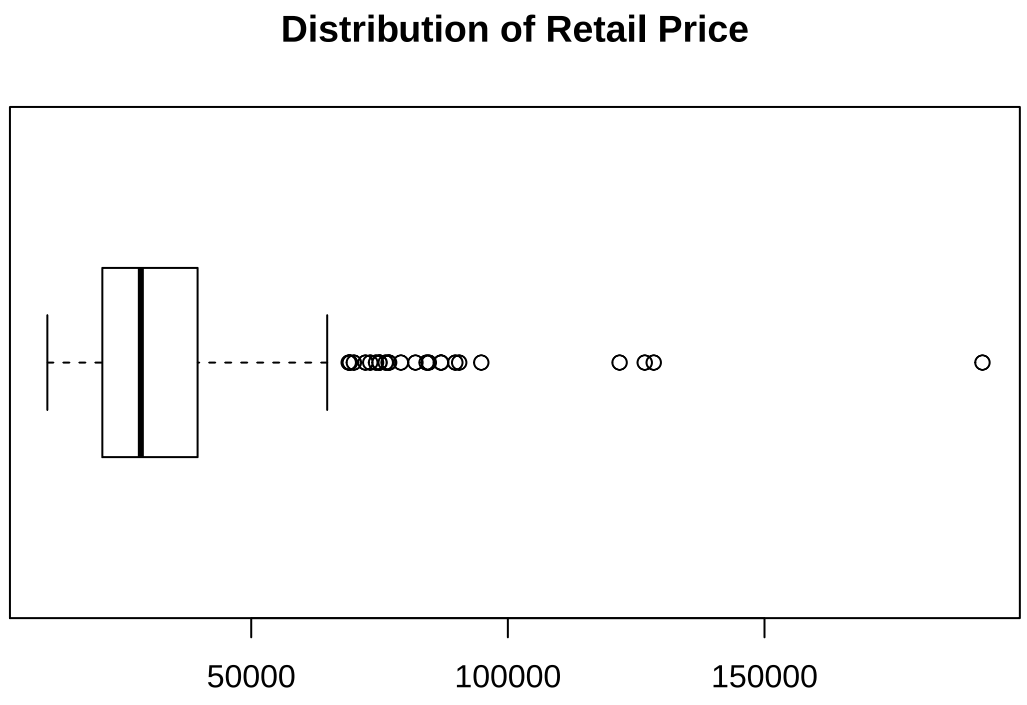
- The original dataset has 428 observations (vehicles).
- Only 387 vehicles have complete (non-missing) information.
- For the ensuing analysis, we retain 387 observations with complete, non-missing data for each variable.

Next, analyze the distribution of the response variable, test for normality, and plot observations

```r
# create histogram of retail prices
hist(cars$Retailprice, main = "Histogram of Retail price", xlab = "Retail Price")
```



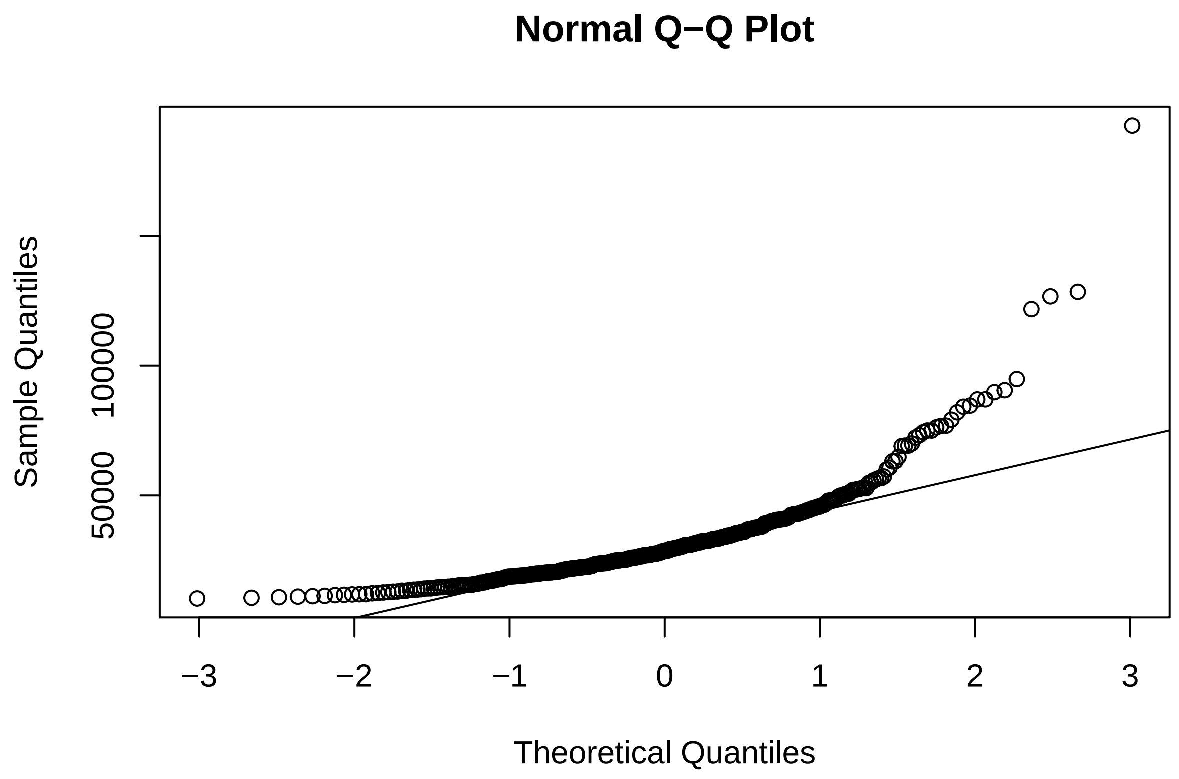
```r
# Plot box plot, get number of outliers
bp <- boxplot(cars$Retailprice, main = "Distribution of Retail Price", horizontal = T)
```



```
cat("num outliers: ", length(bp$out), "\n")
```

```
## num outliers:  25
```

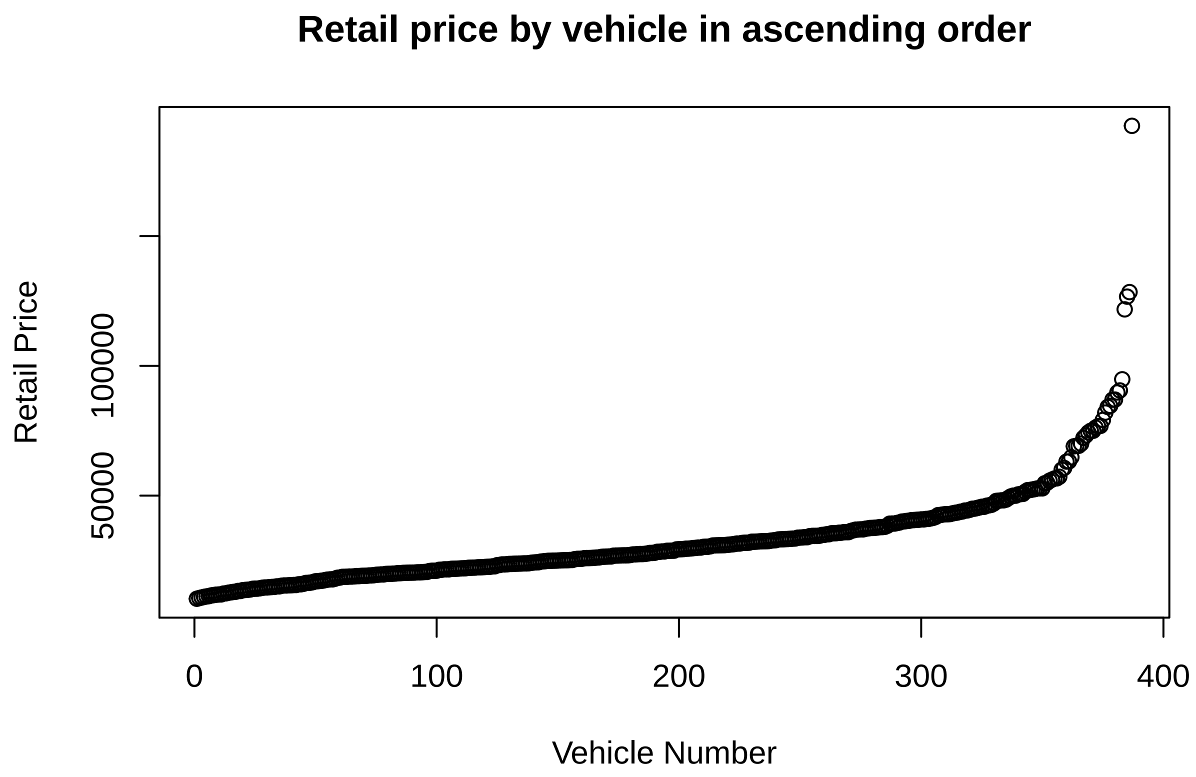
```
# render Q-Q plot
qqnorm(cars$Retailprice)
qqline(cars$Retailprice)
```



```
# Perform Shapiro-Wilk test of normality
shapiro.test(cars$Retailprice)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  cars$Retailprice
## W = 0.77616, p-value < 2.2e-16
```

```
# Plot retail price of cars by index (since retail price is already in
# ascending order)
plot(1:n.complete, cars$Retailprice, xlab = "Vehicle Number", ylab = "Retail Price",
    main = "Retail price by vehicle in ascending order")
```



Summary of operations above:

- *Retailprice* is highly right-skewed.
- There are 25 outliers out of 387 observations (~7.0%)
- The Shapiro-Wilk tests confirms (at the 5% significance level) that *Retailprice* is non-normally distributed (P = ~0)
- The Q-Q Plot indicates that the sample distribution does not derive from a hypothetically normal distribution.
- And finally, the plot of *Retailprice* over index indicates that *Retailprice* belongs to a non-linear distribution at the population level of cars and trucks from 2004.

(Part 2a.) Identify two possible methods (from two different lessons) for predicting the response and provide justification for why these methods are appropriate for the data.*

Out of the 7+ regression methods that we have covered thus far, 2 are among the most preferred for this exercise, in my opinion. They are regularizied regression/shrinkage methods (Ridge, LASSO, Elastic Net) and RandomForest regression:

- Regularized regression is a suitable choice because it enables us to predict the response variable on the basis of *multiple* predictor variables while decreasing test variance through the use of a penalty parameter (lambda).
- RandomForest regression is based on training decision trees on bootstrapped samples of data. It excels in reducing test variance by limiting the random subset of predictors that are considered at each split

of the tree, effectively decorrelating trees that are fit to samples [1]. Furthermore, RandomForest is an applicable approach to modeling this data because it is well-suited to handle qualitative variables [2] and also offers high model interpretability, which is a benefit in cases when the response variable is financial.

Note: The following additional methods were also trained on this data and are available in the appendix (and will not run automatically):

- Bagging
- Boosting
- Multiple Linear regression (GLM)
- Best subset selection
- KNN

(Part 2b.) Use cross-validation techniques to select between the two methods (and among any parameters needed for those methods).

Set up sample groups for CV10

```r
# create CV groups
n <- dim(cars)[1]
k = 10
groups = c(rep(1:k,floor(n/k)),1:(n-floor(n/k)*k))
set.seed(3)
cvgroups = sample(groups, n)
```

Perform 10-fold CV for RandomForest

```r
set.seed(3)

# store predictions for randomForest
predict.rf = rep(-1,n)
mse_mtry <- rep(NA, 10)

# loop through every fold, train models, and make predictions
for(i in 1:k) {
  train = (cvgroups != i)
  test = (cvgroups == i)
  rf.cv <- randomForest(Retailprice~., data=cars[train,], importance=TRUE)
  predict.rf[test] = predict(rf.cv, newdata=cars[test,], type="response")
}
```

Calculate CV10 and R2 scores:

```
## Randomforest CV10 error:  84332155
```

```
## Randomforest R2:  0.7826804
```

Perform 10-fold CV for Shrinkage models (Ridge, Lasso, E-Net):

```r
# create hyperparameters
n.models = 10
lambda_list = 10^seq(10,-2, length=1000)
alpha_list = c(0,.1,.2,.3,.4,.5,.6,.7,.8,.9,1)

# store best lambda and cvm for Ridge, LASSO, ENET
best_lambdas = rep(NA, n.models)
best_cvm = rep(NA, n.models)

# perform CV10 for each model: Ridge, LASSO, ENET
for(m in 1:n.models) {
  set.seed(3)
  cv.fit = cv.glmnet(x, y, lambda = lambda_list, alpha = alpha_list[m], nfolds = k)
  best_lambdas[m] <- cv.fit$lambda.min
  best_cvm[m] <- cv.fit$cvm[which(cv.fit$lambda == cv.fit$lambda.min)]
}

# compute best alpha and best lambda for lowest CV model
lowest_cv_index = order(best_cvm)[1]
best_alpha = alpha_list[lowest_cv_index]
best_lambda = best_lambdas[lowest_cv_index]

# store predictions for best model
cv.fit.predictions <- rep(NA, n)

# CV10: loop through every fold, train models, and make predictions
for(i in 1:k) {
  train = (cvgroups != i)
  test = (cvgroups == i)
  optimal.fit = glmnet(x[train,], y[train], alpha = best_alpha, lambda = best_lambda)
  cv.fit.predictions[test] = predict(optimal.fit, newx = x[test, ], s = best_lambda)
}
```

Calculate CV10 and R2 scores:

```
## E-net CV10 error:  111741019
```

```
## E-net R2:  0.7120492
```

Results of operations above:

- The CV10 RandomForest model is superior to the best regularization model above (alpha = 0.2, lambda = 174.75). RandomForest results in an R2 of ~0.78 and CV10 of 84332155, while E-Net produces an R2 of ~0.71 and a CV10 of 111741019 Therefore, we will test double 10-fold cross-validation with RandomForest regression.

(Part 2c.) Add an outer level of cross-validation to further assess the predictive ability of the model selected.

Perform double 10-fold CV with RandomForest (Using rfcv function from RandomForest package)

```r
# create sampling groups
p = dim(cars)[2]-1
folds = 10
groups.out = c(rep(1:folds,floor(n/folds)), 1:(n%%folds))
set.seed(3)
cvgroups.out = sample(groups.out, n)
rf.doublecv2.predictions2 = rep(NA, n)

# perform outer cross-validation loop
for (i in 1:folds)  {
  train = (cvgroups.out != i)
  test = (cvgroups.out == i)
  num.train.obs = dim(x[train,])[1]

  # create groups for train/test observations
  if ((num.train.obs%%folds) == 0) {
      groups.in= rep(1:folds,floor(num.train.obs/folds))
  } else {
      groups.in=c(rep(1:folds,floor(num.train.obs/folds)),(1:(num.train.obs%%folds)))
  }

  # sample training observations
  cvgroups.in = sample(groups.in, num.train.obs)

  # perform randomforest 10-fold cross validation with step function
  rf.cv.fit2 <- rfcv(x[train,], y[train], cv.fold=10, scale="log", step=0.5,
↪  mtry=function(p) max(1, floor(sqrt(p))), recursive=FALSE, subset=cvgroups.in)

  # get lowest_mse and optimal subset of predictors for reach split (mtry)
  lowest_mse <- min(rf.cv.fit2$error.cv)
  best_m <- rf.cv.fit2$n.var[which(rf.cv.fit2$error.cv == min(rf.cv.fit2$error.cv))]
  rf.doublecv10.fit2 <- randomForest(Retailprice~., data=cars[train,], mtry=best_m,
↪  importance=TRUE)
  rf.doublecv2.predictions2[test] = predict(rf.doublecv10.fit2, newdata=cars[test,],
↪  type="response")
}
```

Calculate CV10 and R2 scores:

```
## RF double CV10 MSE:  80089867
```

```
## RF double CV10 R2:  0.7936125
```

(Part 2d.) Fit the final selected model to the data and summarize and discuss the outcome of the fit. Include estimates of any model parameters.*

Fit final model

```r
final.model <- randomForest(Retailprice~., data=cars, mtry=best_m, importance=TRUE)
predict.fm = predict(final.model, newdata=x, type="response")
```

Calculate out-of-bag (OOB) MSE and R2

A bagged tree is fit to ~2/3 of the observations of a bootstrapped sample [1]. Predictions are made on the other 1/3 (out-of-bag) observations. MSE and R2 is calculated on these OOB predictions for every sample. Below, we take the mean MSE and R2 of OOB predictions across all bootstrapped samples. The results are:

```
## Final model OOB MSE:  74368100
```

```
## Final model OOB R2:  0.8083572
```

Calculate CV10 and R2 scores:

```
## Final model MSE:  16820857
```

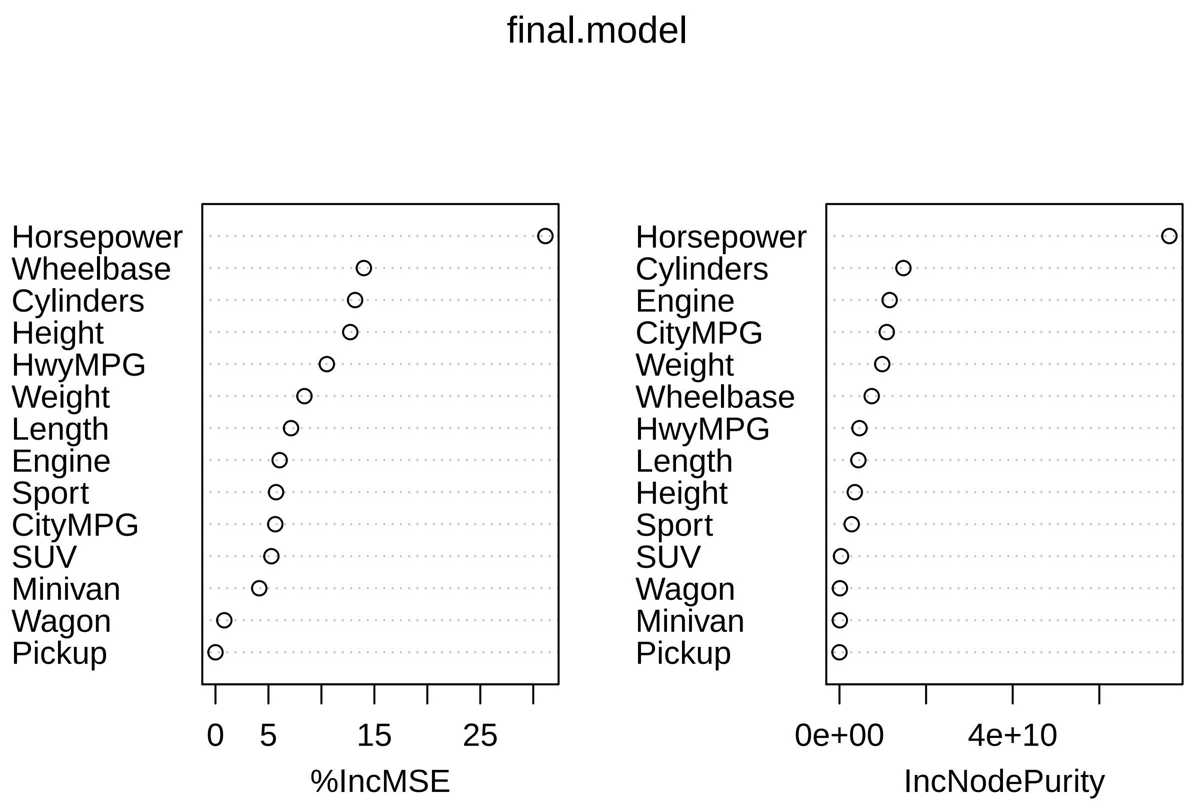
```
## Final model R2:  0.9566535
```

Calculate and plot variable importance

```r
importance(final.model)
```

```
##                %IncMSE IncNodePurity
## Sport        5.7189422    2827226376
## SUV          5.2792588     357770785
## Wagon        0.8340518     101980506
## Minivan      4.1323707      79839857
## Pickup       0.0000000             0
## Engine       6.0591260   11586009527
## Cylinders   13.1815042   14767829128
## Horsepower  31.1476046   76186338649
## CityMPG      5.6458500   10917273614
## HwyMPG      10.5128086    4618764981
## Weight       8.3995061    9869551199
## Wheelbase   14.0100303    7446217175
## Length       7.1331615    4364484076
## Height      12.7246414    3546628888
```

```r
varImpPlot(final.model)
```



Conclusion

The final model (RandomForest with mtry=7), produces the following CV and R2 scores:

Double CV10:

- RF double CV10 MSE: 80089867
- RF double CV10 R2: 0.7936125

OOB Predictions:

- Final model OOB MSE: 74368100
- Final model OOB R2: 0.8083572

Full data predictions:

- Final model MSE: 16820857
- Final model R2: 0.9566535

Given the small size of this dataset, the MSE and R2 scores achieved through RandomForest regression with double CV10 and OOB predictions indicate that this model shows promise of delivering accurate predictions on test data. The final RandomForest model, when fitted to the full dataset, produces an R2 of nearly 0.96.

The most important (positively correlated) predictors on retail price were: horsepower, wheelbase, cylinders, height, and highway mpg. Furthermore, the plot of variable importance indicates that the final model could benefit in accuracy by excluding certain predictors such as "pickup".

Appendix

Below you will find various regression models performed with 10-fold cross validation. Please find the table at the bottom for a comparison of MSE and R2 for all models tested in this exercise.

Perform double 10-fold CV with RandomForest regression (using methodology taught in class):

```r
# create sampling groups
p = dim(cars)[2]-1
m = p - 1 # m is the predictor subset size
folds = 10
groups.out = c(rep(1:folds,floor(n/folds)), 1:(n%%folds))
set.seed(3)
cvgroups.out = sample(groups.out, n)
rf.doublecv10.predictions = rep(NA, n)

# perform outer cross-validation loop
for (i in 1:folds)  {
  train = (cvgroups.out != i)
  test = (cvgroups.out == i)
  num.train.obs = dim(x[train,])[1]

  # create groups for train/test observations
  if ((num.train.obs%%folds) == 0) {
      groups.in= rep(1:folds,floor(num.train.obs/folds))
  } else {
      groups.in=c(rep(1:folds,floor(num.train.obs/folds)),(1:(num.train.obs%%folds)))
  }

  cvgroups.in = sample(groups.in, num.train.obs)

  # store MSE for every m
  mtry_matrix = matrix(NA, nr=m, nc=folds)

  # loop through every subset of predictors 'm'
  for(j in 1:m) {
    # for every 'm', perform CV10 with resampled cvgroups
    for(k in 1:folds) {
      rf.cv10 <- randomForest(Retailprice~., data=cars[train,], subset=cvgroups.in,
↪   mtry=j, importance=TRUE)
      mtry_matrix[j,k] <- mean(rf.cv10$mse) # take mean MSE of out-of-bag predictions for
↪   this fold and model
    }
  }

  lowest_mse <- apply(mtry_matrix, 1, mean)
  best_m <- order(lowest_mse)[1]
  rf.doublecv10 <- randomForest(Retailprice~., data=cars[train,], mtry=best_m,
↪   importance=TRUE)
  rf.doublecv10.predictions[test] = predict(rf.doublecv10, newdata=cars[test,],
↪   type="response")
}
```

Calculate CV10 and R2 scores:

```r
rf.doublecv10.mse = mean((rf.doublecv10.predictions-y)^2)
rf.doublecv10.R2 = 1-sum((rf.doublecv10.predictions-y)^2)/sum((y-mean(y))^2)
cat("RF double CV10 MSE: ", rf.doublecv10.mse, "\n")
```

```
## RF double CV10 MSE:  82355819
```

```r
cat("RF double CV10 R2: ", rf.doublecv10.R2, "\n")
```

```
## RF double CV10 R2:  0.7877733
```

```r
models_cv[4] <- rf.doublecv10.mse
models_r2[4] <- rf.doublecv10.R2
```

Bagging: CV10

```r
set.seed(3)

# store predictions for randomForest
predict.rf = rep(-1,n)
predict.bagging = rep(-1,n)

# loop through every fold, train models, and make predictions
for(i in 1:k) {
  train = (cvgroups != i)
  test = (cvgroups == i)
  bagging.cv <- randomForest(Retailprice~., data=cars[train,], mtry=14, importance=TRUE)
  predict.bagging[test] = predict(bagging.cv, newdata=cars[test,], type="response")
}
```

Calculate CV10 and R2 scores:

```r
# Calculate CV error and R2 for both randomforest and bagging
bagging_cv_error <- mean((predict.bagging-y)^2)
bagging_R2 = 1-sum((predict.bagging-y)^2)/sum((y-mean(y))^2)
cat("Bagging CV10 error: ", bagging_cv_error, "\n")
```

```
## Bagging CV10 error:  82852234
```

```r
cat("Bagging R2: ", bagging_R2, "\n")
```

```
## Bagging R2:  0.7864941
```

```r
models_cv[5] <- bagging_cv_error
models_r2[5] <- bagging_R2
```

Boosting: CV10

```
set.seed(3)
# store predictions for gbm (boosting)
predict.gbm = rep(-1,n)

# loop through every fold, train models, and make predictions
for(i in 1:k) {
  train = (cvgroups != i)
  test = (cvgroups == i)
  cv.gbm <- gbm(Retailprice~., data=cars[train,], distribution="gaussian", n.trees=5000,
↪  interaction.depth=4, shrinkage=.001)
  predict.gbm[test] = predict(cv.gbm, newdata=cars[test,], n.trees=5000, type="response")
}
```

Calculate CV10 and R2 scores:

```
# Calculate CV error and R2 for boosting
gbm_cv_error <- mean((predict.gbm-y)^2)
gbm_r2 = 1-sum((predict.gbm-y)^2)/sum((y-mean(y))^2)
cat("Boosting CV10 error: ", gbm_cv_error, "\n")
```

```
## Boosting CV10 error:  116210660
```

```
cat("Boosting R2: ", gbm_r2, "\n")
```

```
## Boosting R2:  0.7005311
```

```
models_cv[6] <- gbm_cv_error
models_r2[6] <- gbm_r2
```

Multiple Linear Regression: CV10

```
set.seed(3)
# store predictions for lm cv
predict.lm = rep(-1,n)

# loop through every fold, train models, and make predictions
for(i in 1:k) {
  train = (cvgroups != i)
  test = (cvgroups == i)
  glm.fit = glm(Retailprice~., data=cars[train,])
  predict.lm[test] <- predict(glm.fit, cars[test,])
}
```

Calculate CV10 and R2 scores:

```
# Calculate MSE on full data predictions
lm_cv <- mean((predict.lm-y)^2)
```

```r
lm_r2 = 1-sum((predict.lm-y)^2)/sum((y-mean(y))^2)
cat("LM CV error: ", lm_cv, "\n")
```

```
## LM CV error:  112241241
```

```r
cat("LM R2: ", lm_r2, "\n")
```

```
## LM R2:  0.7107601
```

```r
models_cv[7] <- lm_cv
models_r2[7] <- lm_r2
```

Regsubsets: CV10

```r
# Define predict function for regsubsets
predict.regsubsets <- function(object, newdata, id, ...) {
  form = as.formula(object$call[[2]])
  mat = model.matrix(form, newdata)
  coefi = coef(object, id=id)
  xvars = names(coefi)
  mat[ , xvars] %*% coefi
}
```

```r
# specify num observations, num max predictors, and groups to designate each observation
↪  to
p = dim(cars)[2]-2

# set seed, create sample from groups, create matrix to store MSE error for every model
↪  of each fold
group.error = matrix(NA, nr=p, nc=k)
regsub.preds = matrix(NA, nr=length(y), nc=p)

# loop through every fold, generate regsubsets models
for(i in 1:k) {
  set.seed(3)
  train = (cvgroups != i)
  test = (cvgroups == i)
  regsub.cv = regsubsets(Retailprice~., data = cars[train,], nvmax=p)

  # for every regsubsets model, predict Y on the current fold and then calculate MSE
  ↪  using actual Y
  for(j in 1:p) {
    regsub.preds[test,j] = predict(regsub.cv, newdata = cars[test,], id=j)
    group.error[j,i] = mean((regsub.preds[test,j] - y[test])^2)
  }
}
```

```
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
```

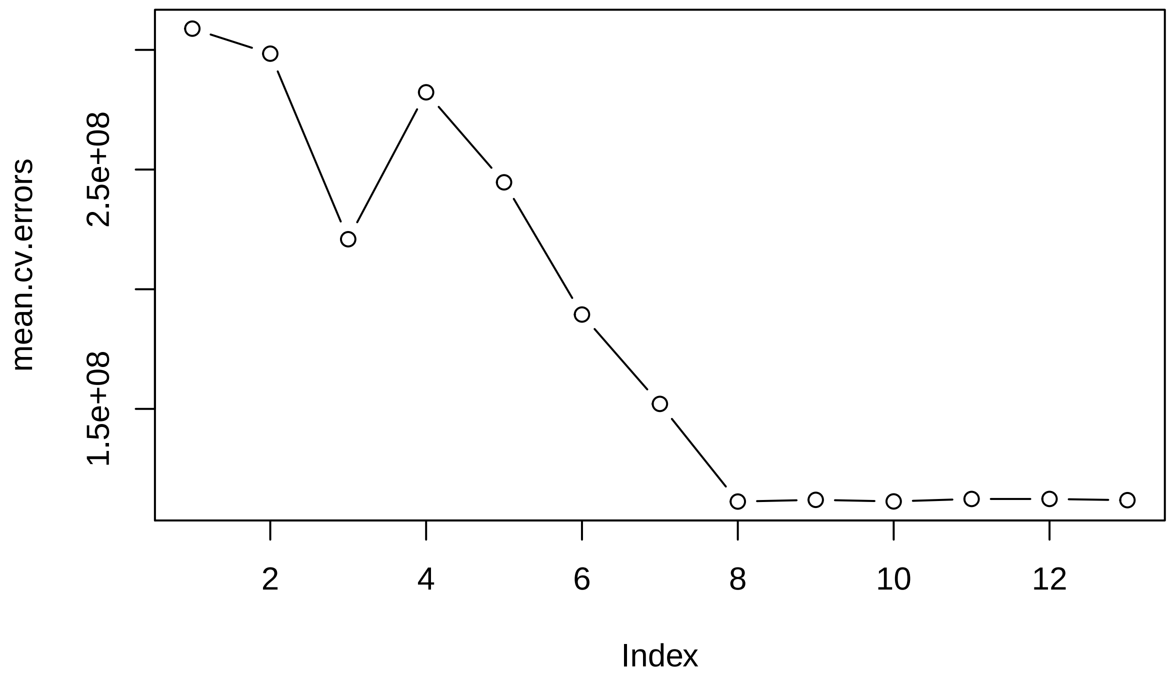
```
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
## Reordering variables and trying again:
```

Calculate mean MSE per model across all folds

```
# calculate mean MSE for every model
mean.cv.errors = apply(group.error, 1, mean)

# select best model by lowest mean MSE
best.model = order(mean.cv.errors)[1]

# get best model coefs
coef(regsub.cv, id=best.model)
```

```
## (Intercept)       Sport   Cylinders  Horsepower     CityMPG      Weight
## 3816.096527 3715.470374 1292.497412  234.695793  593.954072    3.973504
##   Wheelbase      Length      Pickup
## -544.780926   19.882777    0.000000
```

```
# plot MSE vs model
par(mfrow=c(1,1))
plot(mean.cv.errors, type="b")
```



Calculate CV10 and R2 scores:

```
regsubsets_cv <- mean((regsub.preds[,best.model]-y)^2)
regsubsets_r2 = 1-sum((regsub.preds[,best.model]-y)^2)/sum((y-mean(y))^2)
cat("Regsubsets best model (8) CV10 error: ", regsubsets_cv, "\n")
```

```
## Regsubsets best model (8) CV10 error:  111623629
```

```
cat("Regsubsets best model R2: ", regsubsets_r2, "\n")
```

```
## Regsubsets best model R2:  0.7123517
```

```
models_cv[8] <- regsubsets_cv
models_r2[8] <- regsubsets_r2
```

KNN: CV10

```
set.seed(3)
# store predictions for knn cv
predict.knn = rep(-1,n)

# loop through every fold, train models, and make predictions
for(i in 1:k) {
  train = (cvgroups != i)
  test = (cvgroups == i)
  knn.fit <- knn.reg(x[train,], x[test,], y[train], k = 3, algorithm=c("kd_tree"))
  predict.knn[test] <- knn.fit$pred
}
```

Calculate CV10 and R2 scores:

```
knn_cv <- mean((predict.knn-y)^2)
knn_r2 = 1-sum((predict.knn-y)^2)/sum((y-mean(y))^2)
cat("KNN CV error: ", knn_cv, "\n")
```

```
## KNN CV error:  139869667
```

```
cat("KNN R2: ", knn_r2, "\n")
```

```
## KNN R2:  0.6395631
```

```
models_cv[9] <- knn_cv
models_r2[9] <- knn_r2
```

Comparison of all models

```
model_assessment <- cbind(model_names, models_cv, models_r2)
model_assessment
```

```
##       model_names                         models_cv models_r2
## [1,] "RandomForest CV10"             84332155  0.7826804
## [2,] "Shrinkage Models CV10"         111741019 0.7120492
## [3,] "RandomForest Double CV10 - v2" 80089867  0.7936125
## [4,] "RandomForest Double CV10 - v1" 82355819  0.7877733
## [5,] "Bagging CV10"                  82852234  0.7864941
## [6,] "Boosting CV10"                 116210660 0.7005311
## [7,] "MLR CV10"                      112241241 0.7107601
## [8,] "Regsubsets CV10"               111623629 0.7123517
## [9,] "KNN CV10"                      139869667 0.6395631
```

Sources:

- [1] Introduction to Statistical Learning, page 319-320.
- [2] Introduction to Statistical Learning, page 315.