STA 478 Homework 5 Solutions

Dr. Robert Buscaglia

October 03, 2025

ISLR Chapter 4

Problem 3 (ISLR : Exercise 11)

In this problem, you will develop models that predict whether a given car gets above or below median gas mileage based on the `Auto` data set that is included in the package `ISLR`.

```
data(Auto, package='ISLR')
```

a

Create a binary variable, `mpg01`, that is assigned a 1 if mpg is a value above the sample median, and a 0 if mpg is a value below the median. You can compute the median using the `median()` function. Add this column to the `Auto` data set. *Hint: You also need to make sure this is a factor or else your modeling functions will complain. Use the `factor()` command to do so.*

Solution.

I create a new data frame with the factored `mpg01` variable, and then remove the numerical version of mpg.

```
Auto.new <- Auto %>%
  mutate(mpg01 = factor(ifelse(mpg>median(mpg), "Above", "Below"))) %>%
  dplyr::select(-mpg)
```

b

Explore the data graphically in order to investigate the association between mpg01 and the other features. Which of the other features seem most likely to be useful in predicting mpg01? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

Solution.

I start with an evaluation of the data frame structure.

```
str(Auto.new)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
```

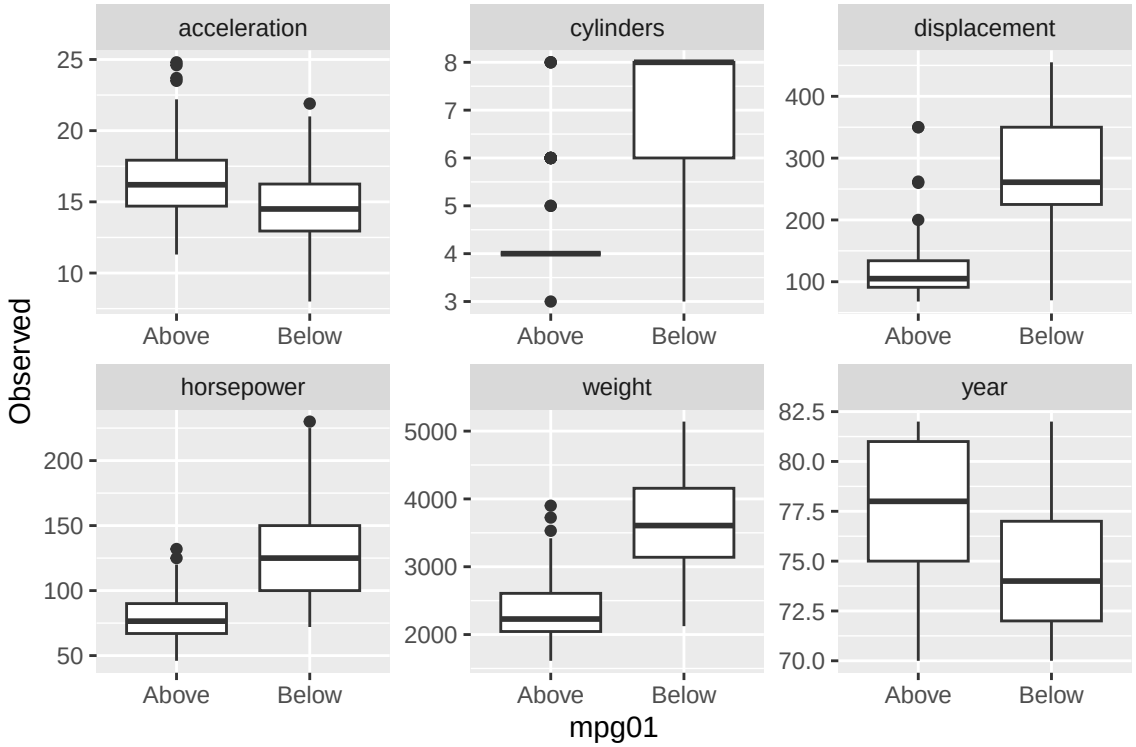
```
##  $ year         : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin       : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name         : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54
##  $ mpg01        : Factor w/ 2 levels "Above","Below": 2 2 2 2 2 2 2 2 2 2 ...
```

The variable `name` has 304 levels, and without some arbitrary groupings will over-parameterize my model, thus I will drop `name`. The variable `cylinders` also is strict count data, and may be modeled as factors, but for now I will leave it numerical. It seems treating cylinders this way works, as a "unit increase in cylinders" is a viable interpretation, although factors might improve model sensitivity to this variable. I also factor `origin` as a numerical representation is detrimental to modeling this correctly, as it indicates a location/space not a numerical value.

```
Auto.new <- Auto.new %>% dplyr::select(-name)
Auto.new$origin <- factor(Auto.new$origin)
```

With this parameter reduction and cleaning complete, I move on to visualization. I choose boxplots to see how well particular variables may discriminate between my two classes. I have to remove `origin` for this analysis.

```
Auto.new.gather <- Auto.new %>%
  dplyr::select(-origin) %>%
  pivot_longer(names_to = "Variable", values_to = "Observed", 1:6)
ggplot(Auto.new.gather, aes(x = mpg01)) +
  geom_boxplot(aes(y = Observed)) + facet_wrap(.~Variable, scales='free')
```

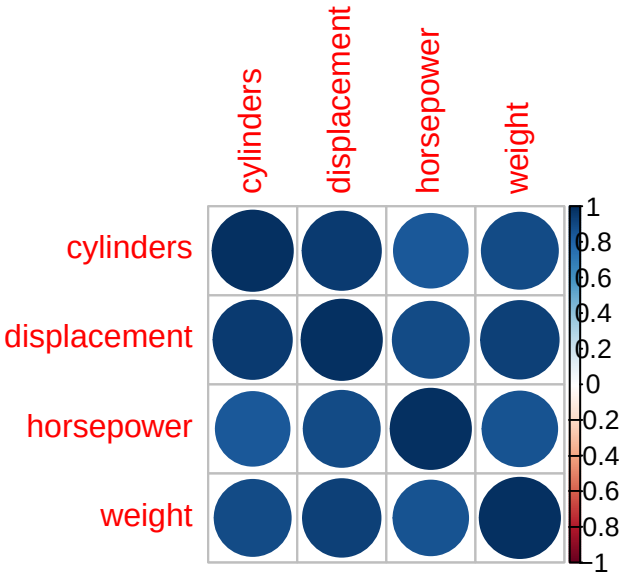


It is observed that most of the variables could have some use in discriminating between cars with above and below median MPG. We might be inclined to choose `cylinders`, `displacement`, `horsepower`, and `weight` as they seem to be the most distinct. If we were to move forward with these four choices, we should then also carefully scrutinize the predictor-predictor correlations.

```
cor(Auto.new[,1:4])
```

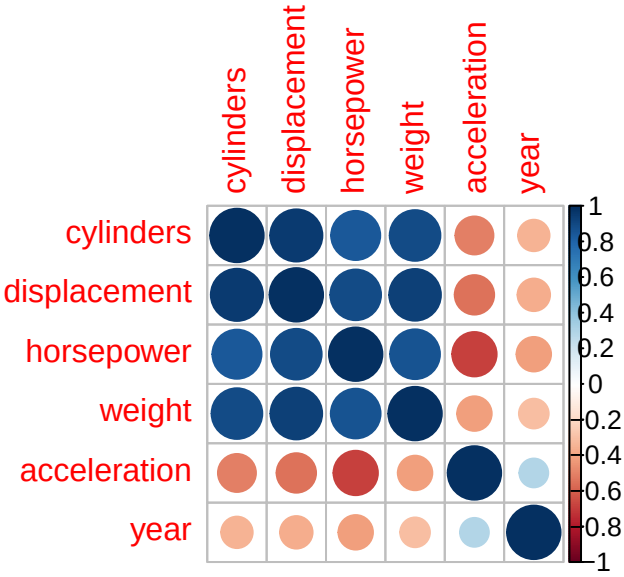
```
##              cylinders displacement horsepower    weight
## cylinders    1.0000000    0.9508233  0.8429834 0.8975273
## displacement 0.9508233    1.0000000  0.8972570 0.9329944
## horsepower   0.8429834    0.8972570  1.0000000 0.8645377
## weight       0.8975273    0.9329944  0.8645377 1.0000000
```

```
corrplot::corrplot(cor(Auto.new[,1:4]))
```



These predictors are all strongly correlated, and I really only want to choose one of them. If I wanted to minimize correlation, we could choose `cylinders` and `horsepower`. If we had not just discriminated based on boxplots, we might have also chosen `year` or `acceleration`, as these are much less correlated to the other predictors.

```
corrplot::corrplot(cor(Auto.new[,1:6]))
```



We could use both `acceleration` and `year` along with `cylinders` all three of which are only moderately correlated to one another. This is clearly a difficult choice and we will want methods for parameter selection. We also still have to decide if offsets for `origin` are influential to the model. We could also select a handful of potential models, and run cross-validation to compare them.

Let's think about a computational means to solving this issue of parameter selection. We will learn selection methods in later chapters, but for now we have all (mostly) seen the use of step-wise selection via significance/AIC/BIC or some other metric. Below, I incorporate the use of bootstrapping to evaluate parameter significance by using generalized linear models and stepwise building. I draw a bootstrap selection from my full data.frame, then run **backwards hybrid** selection using `step()` (the default). I repeat this for 1000 bootstrap iterations, each time recording the coefficients of the model. Note that I use **BIC** as I am interested in parsimonious models.

```
set.seed(10) ### same answer
Coef.Table <- NULL
trials = 1000
for(j in 1:trials)
{
  index <- sample(1:392, 392, replace=TRUE) ### resamples
  Boot.temp <- Auto.new %>% slice(index) ### bootstrap dataframe

  ### start with all parameter model and evaluate backward selection on bootstrap dataframe.
  step.test <- step(glm(mpg01 ~ ., data=Boot.temp,
                        family='binomial'), k = log(dim(Boot.temp)[1]), trace=FALSE)

  ### Save the coefficients
  ### The code accounts for the chance we may choose different sizes / parameters each time
  ifelse(is.null(Coef.Table),
         Coef.Table <- t(as.data.frame(step.test$coefficients)),
         Coef.Table <- merge(Coef.Table, t(as.data.frame(step.test$coefficients)), all=TRUE)
  )
}
```

We can then ask, how often were parameters selected for the glm? Below I calculate the fraction of time a parameter is included as significant based on the BIC selection criterion.

```
### How many NAs are present? subtract from total iterations, find frequency
Perc.Incl <- (trials - apply(apply(Coef.Table, 2, is.na),2,sum))/trials
Perc.Incl
```

```
##  (Intercept)         year       weight      origin2      origin3 displacement
##        1.000        1.000        0.980        0.300        0.300        0.185
## acceleration    cylinders   horsepower
##        0.151        0.067        0.551
```

We see that the `(intercept)` is always included, along with the variable `year`. A second parameter for the variable `weight` is included 98% of the time, and seems like an important predictor for the model. Finally, `horsepower` and `origin` appear infrequently, with `acceleration`, `displacement`, and `cylinders` all appearing in less than 20% of the estimated models.

You may choose models how you see fit for these exercises, but the point of the above is tha twe should be scrutinizing our variables prior to running models. I have a good idea of which variables I expect to be useful in predicting `mpg01`, and am starting to rationalize how I expect these predictors to behave in a model. To move forward, we will consider two models:

Model 1 : mpg01 ~ year + weight obtained from bootstrapping

Model 2 : mpg01 ~ cylinders + horsepower obtained from visualization

We could test many more during our cross-validation, but for now I will use these two. Maybe we should consider a model with the origin as a factor, or the use of `horsepower` included in Model 1. This is where CV can really help, along with an expert opinion on what types of relationships are sensible. I personally like the model with `year` and `weight`, as these seem like reasonable factors that can explain why a car may have reduced MPG. A similar argument can be made for many two or three parameter groupings, such as Model 2, as the correlation above indicates many of these variables have a similar ability to explain the variance in `mpg`. My current setup also includes only primary/main effects. There could also be consideration of interactions termss. Possibly polynomials could help also but we should explore more visually before attempting this.

For each of d - g, use subset cross-validation by splitting into 66% **training and** 33% **testing. Repeat at least** 200 **times and produce the estimated `mean test set MSE`.**

d

Perform LDA on the training data in order to predict mpg01 using the variables that seemed most associated with `mpg01` in (b).

Solution.

We start with modeling using LDA. I test both models within the loop. To ensure I test and train all models on the same splits, we will set a seed at the top of each chunk. We could also write one large loop to test all models or store the indexes in vectors/lists.

```
set.seed(10)
trials = 200
split = 0.66
Accuracy.lda <- matrix(nrow=trials, ncol=2)
for(j in 1:trials)
{
  index <- sample(1:nrow(Auto.new), floor(nrow(Auto.new)*split))
  Train <- Auto.new %>% slice(index)
  Test <- Auto.new %>% slice(-index)

  ### model 1 train and test

  lda.fit <- lda(mpg01 ~ year + weight, data=Train)
  preds.lda <- predict(lda.fit, Test)
  pred.class <- ifelse(preds.lda$posterior[,1] > 0.5, 'Above', 'Below')

  Accuracy.lda[j,1] <- mean(pred.class==Test$mpg01)

  ### model 2 train and test
```

```
  lda.fit <- lda(mpg01 ~ cylinders + horsepower, data=Train)
  preds.lda <- predict(lda.fit, Test)
  pred.class <- ifelse(preds.lda$posterior[,1] > 0.5, 'Above', 'Below')

  Accuracy.lda[j,2] <- mean(pred.class==Test$mpg01)
}
```

I can then ask, on average, how well each of these models performs using LDA.

```
colnames(Accuracy.lda) <- c('Model 1 (year + weight)', 'Model 2 (cylinders + horsepower)')
Error.lda = 1 - Accuracy.lda
summary(Error.lda)
```

```
##  Model 1 (year + weight) Model 2 (cylinders + horsepower)
##  Min.   :0.02985         Min.   :0.04478
##  1st Qu.:0.08209         1st Qu.:0.08209
##  Median :0.09701         Median :0.09701
##  Mean   :0.09716         Mean   :0.09825
##  3rd Qu.:0.11940         3rd Qu.:0.11194
##  Max.   :0.15672         Max.   :0.15672
```

It seems both models perform well, with mean test set errors of 10%; however, it seems as though Model 2 has less variation in the error (not much, but min and Q3 are closer to the center), indicating more stable predictions.

e

Perform QDA on the training data in order to predict mpg01 using the variables that seemed most associated with mpg01 in (b). What is the test error of the model obtained?

Solution.

Repeat the work with QDA estimation.

```
set.seed(10)
trials = 200
split = 0.66
Accuracy.qda <- matrix(nrow=trials, ncol=2)
for(j in 1:trials)
{
  index <- sample(1:nrow(Auto.new), floor(nrow(Auto.new)*split))
  Train <- Auto.new %>% slice(-index)
  Test <- Auto.new %>% slice(index)

  ### model 1 train and test

  qda.fit <- qda(mpg01 ~ year + weight, data=Train)
  preds.qda <- predict(qda.fit, Test)
  pred.class <- ifelse(preds.qda$posterior[,1] > 0.5, 'Above', 'Below')
```

```
  Accuracy.qda[j,1] <- mean(pred.class==Test$mpg01)

  ### model 2 train and test

  qda.fit <- qda(mpg01 ~ cylinders + horsepower, data=Train)
  preds.qda <- predict(qda.fit, Test)
  pred.class <- ifelse(preds.qda$posterior[,1] > 0.5, 'Above', 'Below')

  Accuracy.qda[j,2] <- mean(pred.class==Test$mpg01)
}
```

Compare the mean behaviors of the models.

```
colnames(Accuracy.qda) <- c('Model 1 (year + weight)', 'Model 2 (cylinders + horsepower)')
Error.qda = 1 - Accuracy.qda
summary(Error.qda)
```

```
##  Model 1 (year + weight) Model 2 (cylinders + horsepower)
##  Min.   :0.06589         Min.   :0.06589
##  1st Qu.:0.08527         1st Qu.:0.08915
##  Median :0.09302         Median :0.09690
##  Mean   :0.09341         Mean   :0.09705
##  3rd Qu.:0.10078         3rd Qu.:0.10465
##  Max.   :0.12403         Max.   :0.15116
```

We now see Model 1 having a slightly better Test set performance, with a mean error rate of 9.2%. These models are still likely statistical equivalent, and we might want to begin to evaluate the performance based on AUC. Model 1 with QDA does seem useful in comparison to the others so far, so there is indications of some nonlinearity here. Both are still performing very well, with QDA seemingly slightly improved over LDA when using Model 1.

f

Perform logistic regression on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in (b). What is the test error of the model obtained?

Solution.

Repeat the work using estimation based on logistic regression.

```
set.seed(10)
trials = 200
split = 0.66
Accuracy.glm <- matrix(nrow=trials, ncol=2)
for(j in 1:trials)
{
  index <- sample(1:nrow(Auto.new), floor(nrow(Auto.new)*split))
  Train <- Auto.new %>% slice(-index)
```

```
  Test <- Auto.new %>% slice(index)

  ### model 1 train and test

  glm.fit <- glm(mpg01 ~ year + weight, data=Train, family = 'binomial')
  preds.glm <- predict(glm.fit, Test, type='response')
  pred.class <- ifelse(preds.glm<0.5, 'Above', 'Below')

  Accuracy.glm[j,1] <- mean(pred.class==Test$mpg01)

  ### model 2 train and test

  glm.fit <- glm(mpg01 ~ cylinders + horsepower, data=Train, family = 'binomial')
  preds.glm <- predict(glm.fit, Test, type='response')
  pred.class <- ifelse(preds.glm<0.5, 'Above', 'Below')

  Accuracy.glm[j,2] <- mean(pred.class==Test$mpg01)
}
```

Comparing mean behaviors.

```
colnames(Accuracy.glm) <- c('Model 1 (year + weight)', 'Model 2 (cylinders + horsepower)')
Error.glm = 1 - Accuracy.glm
summary(Error.glm)
```

```
##  Model 1 (year + weight) Model 2 (cylinders + horsepower)
##  Min.   :0.06589         Min.   :0.06589
##  1st Qu.:0.08915         1st Qu.:0.09302
##  Median :0.09690         Median :0.10078
##  Mean   :0.09671         Mean   :0.10157
##  3rd Qu.:0.10465         3rd Qu.:0.10853
##  Max.   :0.12403         Max.   :0.15116
```

It is probably not too surprising that the way I built Model 1 leads to a slightly improved test set mean error rate. Again, evidence shows both models are performing well, now with the GLM on par with LDA. QDA still seems slightly improved.

g

Perform KNN on the training data, with several values of K, in order to predict `mpg01`. Use only the variables that seemed most associated with `mpg01` in (b). What test errors do you obtain? Which value of K seems to perform the best on this data set?

Solution.

I now perform the analysis using KNN. We are asked to consider several values for the neighborhood size. To keep things clear, I use a fine mesh of $K = 1, 3, 5, 10, 15, ..., 50$.

```
trials = 200
split = 0.66
K.grid = c(1,3,seq(5, 50, 5))
Accuracy.KNN.model.1 <- matrix(nrow=trials, ncol=length(K.grid))
Accuracy.KNN.model.2 <- matrix(nrow=trials, ncol=length(K.grid))
for(k in 1:length(K.grid))
{
  set.seed(10) ### resetting seed before evaluating validation.
  for(j in 1:trials)
  {
  index <- sample(1:nrow(Auto.new), floor(nrow(Auto.new)*split))
  Train <- Auto.new %>% slice(-index)
  Test <- Auto.new %>% slice(index)

  ### model 1 train and test

  KNN.fit <- kknn(mpg01 ~ year + weight, Train, Test, k=K.grid[k],
                  kernel = 'rectangular')
  Accuracy.KNN.model.1[j,k]<-mean(KNN.fit$fitted.values==Test$mpg01)

  ### model 2 train and test

  KNN.fit <- kknn(mpg01 ~ cylinders + horsepower, Train, Test, k=K.grid[k],
                  kernel = 'rectangular')
  Accuracy.KNN.model.2[j,k]<-mean(KNN.fit$fitted.values==Test$mpg01)
  }

}
```

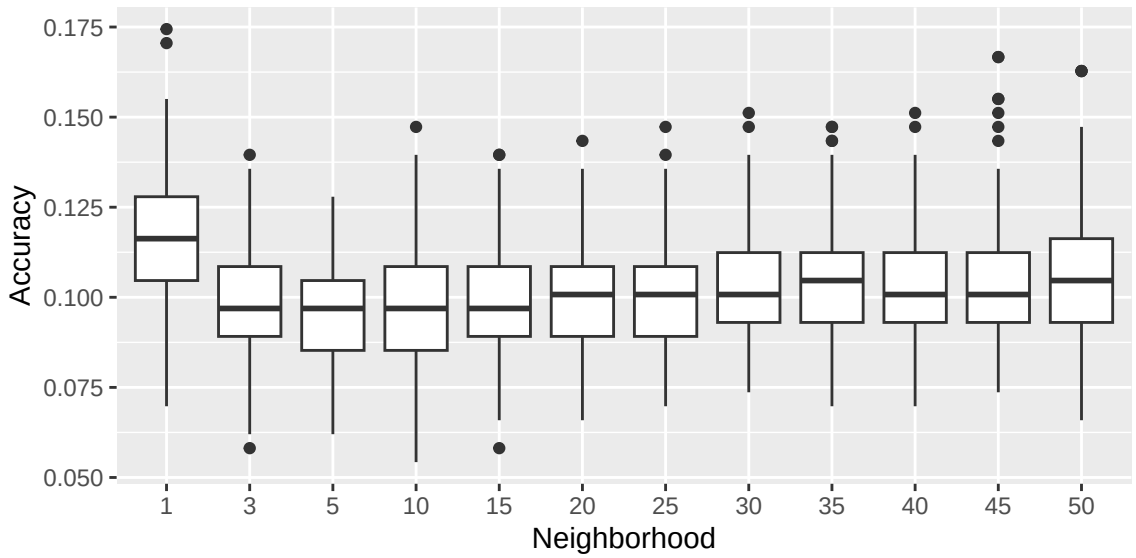
Compare the mean behaviors for model 1, across many neighborhood sizes.

```
Error.KNN.model.1 = 1 - Accuracy.KNN.model.1
colnames(Error.KNN.model.1) <- K.grid
Err.KNN.M.1.long <- Error.KNN.model.1 %>%
  as.data.frame() %>%
  pivot_longer(names_to = 'Neighborhood', values_to = 'Accuracy', 1:length(K.grid)) %>%
  mutate(Neighborhood = factor(Neighborhood, levels=K.grid))

Err.KNN.M.1.long %>% group_by(Neighborhood) %>%
  summarise(Mean = mean(Accuracy),
            Std = sd(Accuracy),
            Median = median(Accuracy),
            IQR = IQR(Accuracy)) %>%
  arrange(Mean) %>%
  kable(align='c', caption='Model 1 KNN Summary Error Rate Statistics arranged by Mean.')
```

Table 1: Model 1 KNN Summary Error Rate Statistics arranged by Mean.

Neighborhood	Mean	Std	Median	IQR
5	0.0951938	0.0142274	0.0968992	0.0193798
10	0.0969186	0.0155695	0.0968992	0.0232558
3	0.0986240	0.0140515	0.0968992	0.0193798
15	0.0989535	0.0139231	0.0968992	0.0193798
20	0.0991860	0.0138103	0.1007752	0.0193798
25	0.1004070	0.0141846	0.1007752	0.0193798
40	0.1031589	0.0141211	0.1007752	0.0193798
30	0.1033333	0.0138062	0.1007752	0.0193798
45	0.1042442	0.0168856	0.1007752	0.0193798
35	0.1043023	0.0135954	0.1046512	0.0193798
50	0.1052519	0.0173299	0.1046512	0.0232558
1	0.1173256	0.0186218	0.1162791	0.0232558

```
ggplot(Err.KNN.M.1.long, aes(x = Neighborhood, y = Accuracy)) + geom_boxplot()
```




```
colnames(Accuracy.KNN.model.1) <- paste('K = ', K.grid)
colnames(Error.KNN.model.1) <- paste('K = ', K.grid)
```

Same but now for model 2. We could certainly `tidy` these solutions more.

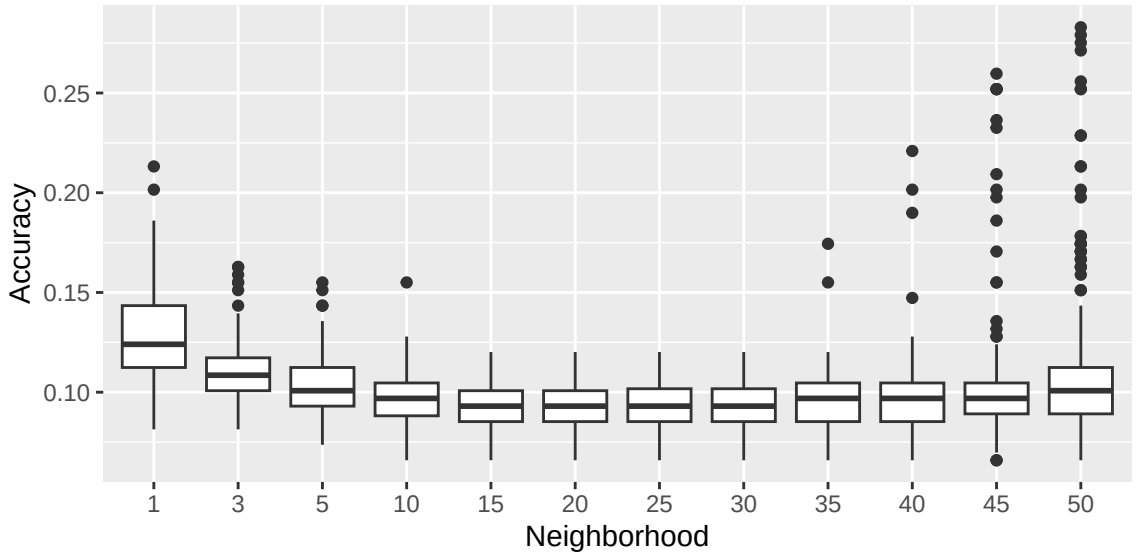
```
Error.KNN.model.2 = 1 - Accuracy.KNN.model.2
colnames(Error.KNN.model.2) <- K.grid
Err.KNN.M.2.long <- Error.KNN.model.2 %>%
  as.data.frame() %>%
  pivot_longer(names_to = 'Neighborhood', values_to = 'Accuracy', 1:length(K.grid)) %>%
  mutate(Neighborhood = factor(Neighborhood, levels=K.grid))

Err.KNN.M.2.long %>% group_by(Neighborhood) %>%
```

```
  summarise(Mean = mean(Accuracy),
            Std = sd(Accuracy),
            Median = median(Accuracy),
            IQR = IQR(Accuracy)) %>%
  arrange(Mean) %>%
  kable(align='c', caption='Model 2 KNN Summary Error Rate Statistics arranged by Mean.')
```

Table 2: Model 2 KNN Summary Error Rate Statistics arranged by Mean.

Neighborhood	Mean	Std	Median	IQR
15	0.0938566	0.0112476	0.0930233	0.0155039
20	0.0938760	0.0112159	0.0930233	0.0155039
30	0.0939922	0.0112668	0.0930233	0.0164729
25	0.0940116	0.0112350	0.0930233	0.0164729
35	0.0951744	0.0134587	0.0968992	0.0193798
10	0.0953876	0.0112201	0.0968992	0.0164729
40	0.0968023	0.0181900	0.0968992	0.0193798
5	0.1028876	0.0133013	0.1007752	0.0193798
45	0.1046899	0.0342889	0.0968992	0.0155039
3	0.1103295	0.0152480	0.1085271	0.0164729
50	0.1130233	0.0419267	0.1007752	0.0232558
1	0.1296318	0.0227759	0.1240310	0.0310078

```
ggplot(Err.KNN.M.2.long, aes(x = Neighborhood, y = Accuracy)) + geom_boxplot()
```




```
colnames(Accuracy.KNN.model.2) <- paste('K = ', K.grid)
colnames(Error.KNN.model.2) <- paste('K = ', K.grid)
```

Let's extract our best neighborhood sizes for these two models. I will store the best results for each model in a new object to prepare for my final comparisons below.

```r
Error.KNN <- matrix(ncol=2, nrow=trials)
which.min(colMeans(Error.KNN.model.1))
```

```
## K =  5
##      3
```

```r
colMeans(Error.KNN.model.1)[3]
```

```
##    K =  5
## 0.0951938
```

```r
Error.KNN[,1] <- Error.KNN.model.1[,3]
```

Model 1 using `Year + Weight` obtains a mean test error rate of 9.5%.

```r
which.min(colMeans(Error.KNN.model.2))
```

```
## K =  15
##       5
```

```r
colMeans(Error.KNN.model.2)[5]
```

```
##    K =  15
## 0.09385659
```

```r
Error.KNN[,2] <- Error.KNN.model.2[,5]
```

Model 2 using `Cylinders + Horsepower` obtains a similar mean test error rate of 9.4%.

Visualization and Conclusions.

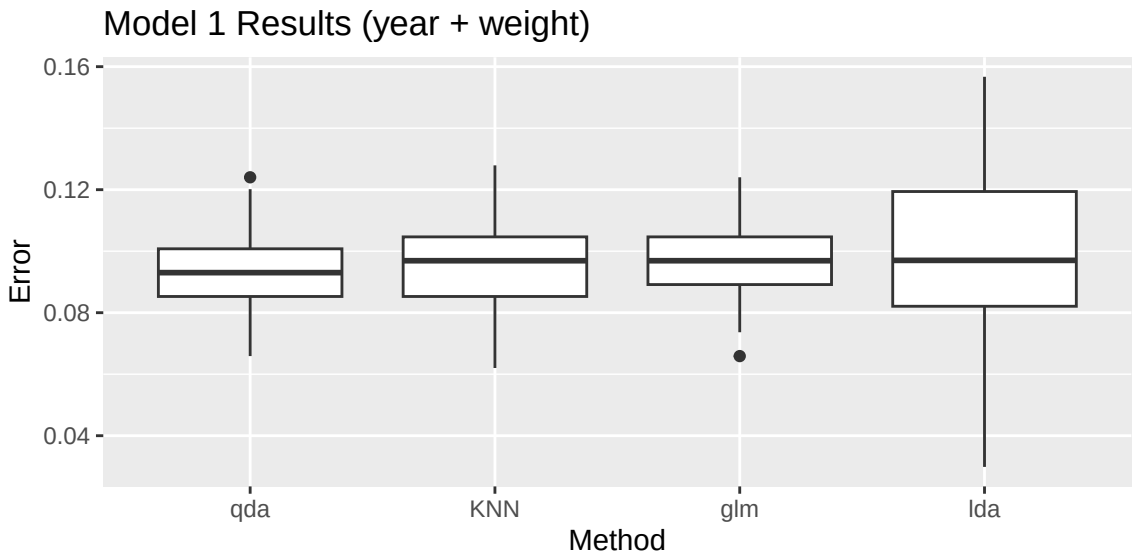
The evaluation of the Auto data set is finished by visualization of the two different models cross-validation performance for each of the techniques used above. Boxplots can be very informative in this situation. I do some data cleaning before visualizing.

```r
lda.df <- data.frame(Iteration = 1:trials, Error.lda) %>% mutate(Method = 'lda')
colnames(lda.df)[2:3] <- c('Model 1', 'Model 2')
lda.df <- lda.df %>% pivot_longer(names_to='Model', values_to = 'Error', 2:3)

qda.df <- data.frame(Iteration = 1:trials, Error.qda) %>% mutate(Method = 'qda')
colnames(qda.df)[2:3] <- c('Model 1', 'Model 2')
qda.df <- qda.df %>% pivot_longer(names_to='Model', values_to = 'Error', 2:3)

glm.df <- data.frame(Iteration = 1:trials, Error.glm) %>% mutate(Method = 'glm')
colnames(glm.df)[2:3] <- c('Model 1', 'Model 2')
```

```
glm.df <- glm.df %>% pivot_longer(names_to='Model', values_to = 'Error', 2:3)

knn.cv <- data.frame(Iteration = 1:trials, Error.KNN) %>% mutate(Method = 'KNN')
colnames(knn.cv)[2:3] <- c('Model 1', 'Model 2')
knn.cv <- knn.cv %>% pivot_longer(names_to='Model', values_to = 'Error', 2:3)

Full.Output <- rbind(lda.df, qda.df, glm.df, knn.cv)
```

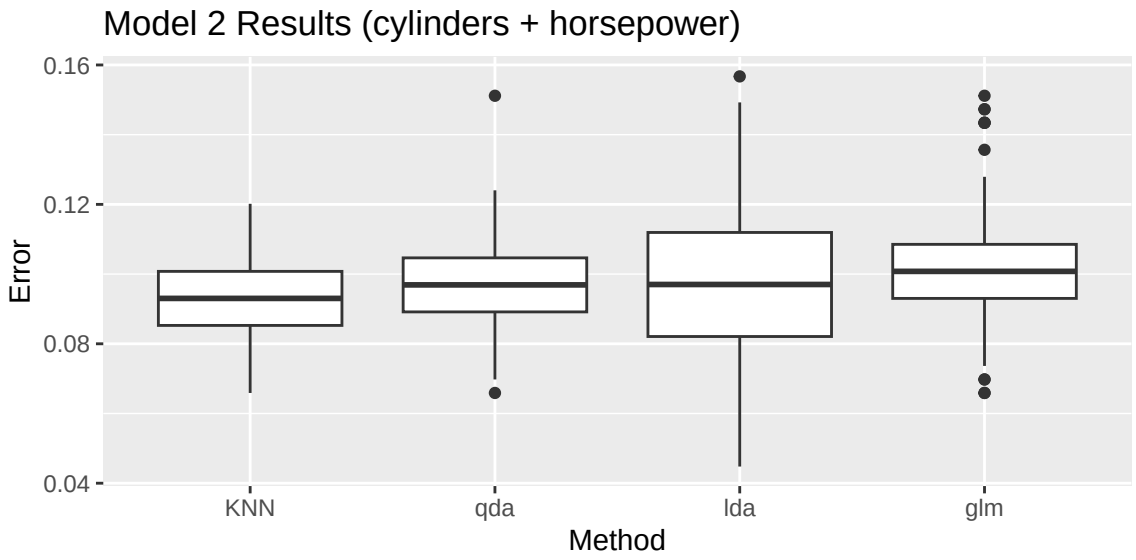
Now we can make some nice plots or tables from the Full.Output. Here is a comparison of Model 1 results sorted by mean test error rate.

```
ggplot(Full.Output %>% filter(Model=='Model 1'),
       aes(x=reorder(Method, Error, FUN = mean))) +
  geom_boxplot(aes(y=Error)) +
  ggtitle('Model 1 Results (year + weight)') + xlab('Method')
```



We observe that there are some differences between the methods used to estimate class, with QDA performing best for Model 1 formulation. We achieved competitive models with KNN and glm to our QDA result, but lda has extremely high variations. This is more evidence that this model form has a nonlinear boundary, and the success of QDA and KNN.

```
ggplot(Full.Output %>% filter(Model=='Model 2'),
       aes(x=reorder(Method, Error, FUN = mean))) +
  geom_boxplot(aes(y=Error)) +
  ggtitle('Model 2 Results (cylinders + horsepower)') + xlab('Method')
```



For Model 2, we achieved the lowest mean test set error rate with KNN using $K = 15$ neighbors, with nearly equivalent results but increasing error rates in the order of `qda`, `lda`, then `glm`.*

We can also calculate statistics from the CV replicates

Summary of results for Model 1

```
Full.Output %>% filter(Model=='Model 1') %>% group_by(Method) %>%
  summarise(Mean = mean(Error), SD = sd(Error),
            Median=median(Error), IQR = IQR(Error)) %>% arrange(Mean) %>%
  kable()
```

Method	Mean	SD	Median	IQR
qda	0.0934109	0.0110315	0.0930233	0.0155039
KNN	0.0951938	0.0142274	0.0968992	0.0193798
glm	0.0967054	0.0110503	0.0968992	0.0155039
lda	0.0971642	0.0236933	0.0970149	0.0373134

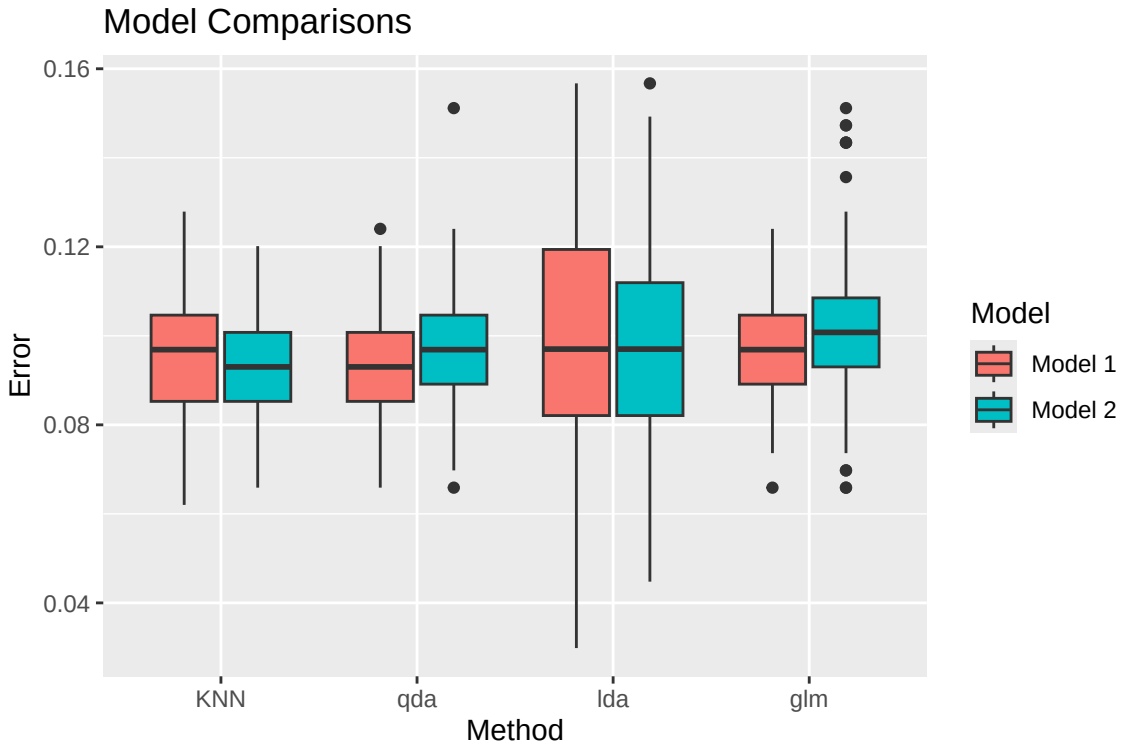
Summary of results for Model 2

```
Full.Output %>% filter(Model=='Model 2') %>% group_by(Method) %>%
  summarise(Mean = mean(Error), SD = sd(Error),
            Median=median(Error), IQR = IQR(Error)) %>% arrange(Mean) %>%
  kable()
```

Method	Mean	SD	Median	IQR
KNN	0.0938566	0.0112476	0.0930233	0.0155039
qda	0.0970543	0.0123846	0.0968992	0.0155039
lda	0.0982463	0.0221401	0.0970149	0.0298507
glm	0.1015698	0.0142521	0.1007752	0.0155039

Side-by-side comparisons of all models.

```
ggplot(Full.Output,
       aes(x=reorder(Method, Error, FUN = mean))) +
  geom_boxplot(aes(y=Error, fill=Model)) + ggtitle('Model Comparisons') +
  xlab('Method')
```



The comparison side-by-side provides interesting insight into the similarity of the models tested. LDA is consistently producing a larger variance with slightly increased mean test set error. QDA provides the lowest mean test set error with both models, showing consistently that Model 1 has slight improvements over Model 2. KNN provides a very sound solution for a nonparametric (non-interpretable) approach. Where I would go from here might be influenced by the final goals of this analysis, but QDA would provide both a highly interpretable and high accuracy predictive model, especially if Model 1 was used.

Exercise 2

Using the `Boston` data set from the `MASS` package, fit classification models in order to predict whether a given suburb has a crime rate above or below the median. Explore logistic regression, LDA, QDA, and KNN models using various subsets of the predictors. Describe your findings. Be sure to include the use of cross-validation.

Solution.

This solution is abbreviated and shows the evaluation of all 8192 possible models that could be built in this scenario (omitting the null model). We setup the data set for work.

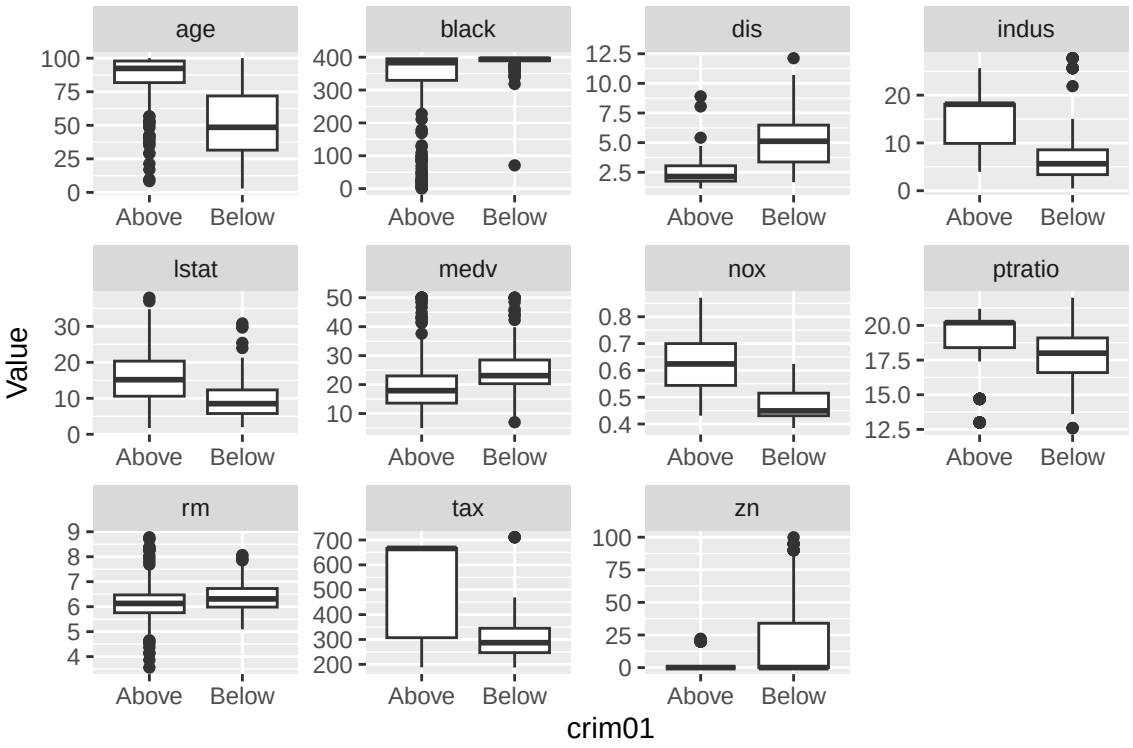
```
data(Boston, package='MASS')
Boston.new <- Boston %>%
  mutate(chas = factor(chas), rad = factor(rad),
```

```
        crim01 = factor(ifelse(crim > median(crim), 'Above', 'Below'))) %>%
  select(-crim)
str(Boston.new)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : Factor w/ 9 levels "1","2","3","4",..: 1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
##  $ crim01 : Factor w/ 2 levels "Above","Below": 2 2 2 2 2 2 2 2 2 2 ...
```

We can visualize the predictors again the categorical response to make decisions regarding which elements will be included in the model.

```
Boston.long <- Boston.new %>% dplyr::select(-chas, -rad) %>%
  pivot_longer(names_to = 'Variable', values_to = 'Value', 1:11)
ggplot(Boston.long, aes(x = crim01, y=Value)) +
  geom_boxplot() +
  facet_wrap(.~Variable, scales='free')
```



I then chose to write an algorithm to check all possible models. The code is below. Be aware, this checks over 8000 models, running cross-validation on all of them using `lda`, `qda`, and `logistic regression`. I choose to omit `KNN` as to not validate a neighborhood grid.

```
iter <- 200
m.counter <- 1

preds <- colnames(Boston.new %>% dplyr::select(-crim01))
total.preds <- length(preds)

lda.acc <- qda.acc <- glm.acc <- matrix(ncol = 2^total.preds-1, nrow = iter)
all.models <- list()

for(k in 1:total.preds)
{
  models.temp <- apply(combn(preds, k), 2, paste, collapse='+')
  total.models <- length(models.temp)
  for(j in 1:total.models)
  {
    m <- formula(paste(c('crim01 ~ ', models.temp[j]), collapse=''))
    all.models[[m.counter]] <- m
    set.seed(1010)
    for(i in 1:iter)
    {
      index <- sample(1:nrow(Boston.new), nrow(Boston.new)*0.66)
      train <- Boston.new %>% slice(index)
      test <- Boston.new %>% slice(-index)

      lda.fit <- lda(m, train)
      lda.pred <- predict(lda.fit, test)
      lda.class <- ifelse(lda.pred$posterior[,2]>0.5, 'Below', 'Above')
      lda.acc[i,m.counter] <- mean(lda.class == test$crim01)

      qda.fit <- try(qda(m, Boston.new), silent=TRUE)

      if(inherits(qda.fit, 'try-error'))
      {
        qda.acc[i,m.counter]<-NA
      } else {
          qda.pred <- predict(qda.fit, test)
          qda.class <- ifelse(qda.pred$posterior[,2]>0.5, 'Below', 'Above')
          qda.acc[i,m.counter] <- mean(qda.class == test$crim01)
      }

      glm.fit <- glm(m, data=train, family='binomial')
      glm.pred <- predict(glm.fit, test, type='response')
      glm.class <- ifelse(glm.pred>0.5, 'Below', 'Above')
      glm.acc[i, m.counter] <- mean(glm.class == test$crim01)
    }
    m.counter <- m.counter + 1}}
```

Now that is one heck of a simulation! This took about 5 hours to run on my office desktop. There are many ways we can improve it, but I wanted to give you a really neat tool and the code to be able to execute it. I have check 8191 models (I excluded the null model) using three different methods across 200 iterations of 66% subset cross-validation. There is a lot we could view and evaluate now that we have made it this far. Let us start by producing a summary of the top 20 models found from each method.

```
glm.bss <- data.frame(glm.acc)
colnames(glm.bss) <- all.models
glm.bss.long <- glm.bss %>%
  pivot_longer(names_to = 'Model', values_to = 'Accuracy', 1:8191)
glm.bss.long %>% group_by(Model) %>%
  summarise(Mean.Acc = mean(Accuracy),
            Std.Acc = sd(Accuracy),
            # Median.Acc = median(Accuracy),
            # IQR.Acc = IQR(Accuracy)
            ) %>%
  arrange(desc(Mean.Acc)) %>% slice(1:20) %>%
  kable(align='lcccc', caption='Best-subset results using logistic regression.')
```

Table 5: Best-subset results using logistic regression.

Model	Mean.Acc	Std.Acc
crim01 ~ indus + nox + rad + tax + black + medv	0.9505202	0.0139395
crim01 ~ indus + nox + rm + rad + tax + black + medv	0.9500578	0.0138234
crim01 ~ indus + nox + rad + tax + black + lstat + medv	0.9499711	0.0139991
crim01 ~ indus + nox + rm + rad + tax + black + lstat + medv	0.9498266	0.0141323
crim01 ~ indus + nox + rm + rad + tax + black	0.9497110	0.0140039
crim01 ~ indus + nox + dis + rad + tax + black + medv	0.9496821	0.0147831
crim01 ~ indus + nox + rad + tax + medv	0.9496532	0.0141659
crim01 ~ zn + indus + nox + rm + dis + rad + black + lstat + medv	0.9495954	0.0149678
crim01 ~ indus + nox + rm + rad + tax + medv	0.9494798	0.0140904
crim01 ~ zn + indus + nox + rm + age + dis + rad + tax + black + medv	0.9494509	0.0155600
crim01 ~ zn + indus + nox + dis + rad + tax + black + lstat + medv	0.9493642	0.0148335
crim01 ~ zn + indus + nox + rad + tax + black + medv	0.9493353	0.0148078
crim01 ~ zn + indus + nox + rm + dis + rad + tax + black + medv	0.9493353	0.0146826
crim01 ~ zn + indus + nox + dis + rad + ptratio + medv	0.9492775	0.0146993
crim01 ~ zn + indus + nox + rm + age + dis + rad + black + medv	0.9492197	0.0156555
crim01 ~ indus + chas + nox + rad + tax + black + medv	0.9491329	0.0140039
crim01 ~ indus + nox + rad + tax + black	0.9491329	0.0143941
crim01 ~ zn + indus + nox + dis + rad + tax + black + medv	0.9491329	0.0154082
crim01 ~ indus + chas + nox + rad + tax + medv	0.9490462	0.0142190
crim01 ~ indus + nox + rad + tax	0.9490173	0.0145653

```r
lda.bss <- data.frame(lda.acc)
colnames(lda.bss) <- all.models
lda.bss.long <- lda.bss %>%
  pivot_longer(names_to = 'Model', values_to = 'Accuracy', 1:ncol(lda.bss))
lda.bss.long %>% group_by(Model) %>%
  summarise(Mean.Acc = mean(Accuracy),
            Std.Acc = sd(Accuracy),
            # Median.Acc = median(Accuracy),
            # IQR.Acc = IQR(Accuracy)
            ) %>%
  arrange(desc(Mean.Acc)) %>% slice(1:20) %>%
  kable(align='lcccc', caption='Best-subset results using LDA.')
```

Table 6: Best-subset results using LDA.

Model	Mean.Acc	Std.Acc
crim01 ~ zn + nox + age + rad + ptratio + black + medv	0.9302023	0.0178509
crim01 ~ zn + nox + age + rad + tax + ptratio + black + medv	0.9300289	0.0190032
crim01 ~ zn + nox + age + dis + rad + tax + ptratio + black + medv	0.9297688	0.0191120
crim01 ~ zn + nox + age + dis + rad + ptratio + black + medv	0.9297110	0.0178086
crim01 ~ zn + nox + rm + age + rad + ptratio + black + medv	0.9295087	0.0178581
crim01 ~ zn + nox + rm + age + rad + tax + ptratio + black + medv	0.9293642	0.0187521
crim01 ~ zn + nox + rm + age + dis + rad + ptratio + black + medv	0.9292775	0.0177286
crim01 ~ zn + chas + nox + age + rad + ptratio + black + medv	0.9291618	0.0177288
crim01 ~ zn + nox + rm + age + dis + rad + tax + ptratio + black + medv	0.9291618	0.0189997
crim01 ~ zn + chas + nox + age + rad + black + medv	0.9291329	0.0184822
crim01 ~ zn + chas + nox + age + rad + tax + ptratio + black + medv	0.9291329	0.0179569
crim01 ~ zn + nox + age + rad + black + medv	0.9290462	0.0187142
crim01 ~ zn + nox + age + dis + rad + black + medv	0.9290173	0.0184269
crim01 ~ zn + indus + nox + age + dis + rad + tax + ptratio + black + medv	0.9288728	0.0195759
crim01 ~ zn + nox + age + rad + tax + black + medv	0.9288439	0.0193012
crim01 ~ zn + chas + nox + rm + age + rad + ptratio + black + medv	0.9288150	0.0173805
crim01 ~ zn + chas + nox + age + dis + rad + tax + ptratio + black + medv	0.9287283	0.0178526
crim01 ~ zn + indus + nox + age + dis + rad + ptratio + black + medv	0.9287283	0.0197375
crim01 ~ zn + chas + nox + age + dis + rad + ptratio + black + medv	0.9286705	0.0178651
crim01 ~ zn + chas + nox + rm + age + rad + black + medv	0.9286127	0.0184140

QDA had a severe problem because it could not estimate a model when including the rad variable due to inconsistent grouping (there are not enough observations of certain rad levels to fit the more flexible QDA model.) I was able to include in my simulation a way to set these models to have an accuracy of NA, and the code below deals with this. However, the results for QDA are not as good because there was no rad variable, which we see from the first two models types, seemed an important variable to include in the model.

```
qda.bss <- data.frame(qda.acc)
colnames(qda.bss) <- all.models
qda.bss <- qda.bss %>% select_if(function(x) !any(is.na(x)))
qda.bss.long <- qda.bss %>%
  pivot_longer(names_to = 'Model', values_to = 'Accuracy', 1:ncol(qda.bss))
qda.bss.long %>% group_by(Model) %>%
  summarise(Mean.Acc = mean(Accuracy),
            Std.Acc = sd(Accuracy),
            # Median.Acc = median(Accuracy),
            # IQR.Acc = IQR(Accuracy)
            ) %>%
  arrange(desc(Mean.Acc)) %>% slice(1:20) %>%
  kable(align='lcccc', caption='Best-subset results using QDA.')
```

Table 7: Best-subset results using QDA.

Model	Mean.Acc	Std.Acc
crim01 ~ zn + indus + chas + nox + rm + dis + ptratio + lstat	0.8938728	0.0179738
crim01 ~ zn + indus + chas + nox + rm + dis + tax + ptratio	0.8934104	0.0202956
crim01 ~ zn + indus + nox + rm + dis + tax + ptratio	0.8915318	0.0194795
crim01 ~ zn + indus + nox + rm + dis + tax + ptratio + black + lstat + medv	0.8909827	0.0195231
crim01 ~ zn + indus + nox + rm + dis + tax + ptratio + lstat	0.8907514	0.0193673
crim01 ~ zn + indus + nox + rm + dis + ptratio + black + lstat + medv	0.8894798	0.0194845
crim01 ~ chas + nox + age + dis + tax + black + lstat + medv	0.8886127	0.0190589
crim01 ~ chas + nox + rm + age + dis + tax + black + lstat	0.8884104	0.0196871
crim01 ~ zn + indus + chas + nox + rm + dis + tax + ptratio + lstat + medv	0.8884104	0.0193343
crim01 ~ zn + chas + nox + rm + age + dis + tax + ptratio + black + lstat	0.8871676	0.0190611
crim01 ~ zn + chas + nox + rm + tax + ptratio + black + lstat + medv	0.8871676	0.0196851
crim01 ~ zn + chas + nox + rm + age + tax + ptratio + black + lstat	0.8871098	0.0191661
crim01 ~ zn + indus + nox + rm + dis + tax + ptratio + lstat + medv	0.8862139	0.0189606
crim01 ~ zn + indus + chas + age + dis + tax + ptratio + lstat	0.8860116	0.0194379
crim01 ~ zn + indus + chas + nox + rm + dis + ptratio + black + lstat + medv	0.8860116	0.0197124
crim01 ~ zn + indus + chas + nox + rm + dis + tax + ptratio + lstat	0.8854624	0.0193506
crim01 ~ zn + chas + nox + rm + dis + tax + black + lstat + medv	0.8851734	0.0197848
crim01 ~ zn + indus + chas + nox + dis + tax + ptratio + medv	0.8850867	0.0203114
crim01 ~ zn + chas + nox + rm + dis + tax + ptratio + lstat + medv	0.8850578	0.0185089
crim01 ~ zn + indus + chas + nox + dis + tax + black + lstat + medv	0.8849711	0.0204222

```
glm.best <- glm.bss.long %>%
  filter(Model == 'crim01 ~ indus + nox + rad + tax + black + medv')
lda.best <- lda.bss.long %>%
  filter(Model == 'crim01 ~ zn + nox + age + rad + ptratio + black + medv')
qda.best <- qda.bss.long %>%
  filter(Model == 'crim01 ~ zn + indus + chas + nox + rm + dis + ptratio + lstat')
final.comparison <- rbind(glm.best, lda.best, qda.best) %>%
  mutate(Method = rep(c('GLM', 'LDA', 'QDA'), each=200))
ggplot(final.comparison, aes(x = Method, y = Accuracy)) +
  geom_boxplot(aes(fill=Method)) +
  labs(title = 'Final Comparison of crim01 Classifiers')
```

Best-subset analysis provides an extensive evaluation of all models and methods. The results show that a logistic regression model using 6 predictors can achieve validation accuracy of 95.1%. This is a fairly remarkable result as it is likely also very interpretable. We can evaluate other characteristics of the model including coefficients, significance testing, and model assumptions. We can discuss how to extract a `bagged` model from the cross-validation results, and we should never forget the importance of checking the summary information of a model. This is an important time to note that we MUST always confirm the assumptions. Here we are doing logistic regression and do not have a normality assumption, but when finalizing a model from computational methods such as presented here, it is also important to review the statistical 'basics' you have learned in other courses.

The methods describe in these solutions also show the ability to search complex model spaces. The methods included here only evaluated main-effect models. However, both the bootstrapping and best-subset approach could be used to search over models include interactions or polynomial terms. The breadth of which we can search in the model world is growing daily, and while we should use our statistical knowledge when possible to guide decisions, do not be afraid to explore these large spaces using computational means when possible. If you used correlation analysis and traditional model building, you likely ended up with a 2 or 3 predictor model and about 90 to 92% accuracy. Computational model building explored a much larger space, finding a 95.1% validated accuracy model, which we can now explore further or potentially put to practical use.

Discussion and Future

This assignment was meant to give you some hands-on experience both producing these models and performing the cross-validation. Judging from my work above, it should be clear there are several models that are all equivalent and competitive for these classification problems. We have rationalization for why some of these have slightly improved mean test set Error Rates. However, it should be stressed very few of these models are significantly different. We will learn tests for this in the near future.

A final note, notice I was able to study 2 models fairly easily. It should be pointed at, that all possible models could be checked here. This is known as `best-subset analysis`. There is no reason, with the speed of our modern computers, that we can't check all possible models when `p` is reasonable. IBM classified reasonable as about `p = 25-30`, which puts us at 2^{25} or about 33-million models to 2^{30} or about a billion models that can be checked in a relatively short amount of time (less than a day). So, keep in mind as we move into model selection, that it is possible to do `best-subset` searches when `p` is of a reasonable size.