# Model Validation

## Miguel Fernandez

### BAN-502

# Import libraries  
library(tidyverse)  
library(MASS)  
library(caret)

# Read in data  
bike <- read\_csv("hour.csv")  
  
# Recode variables and levels  
bike <- bike %>%  
 mutate(season = as\_factor(as.character(season))) %>%  
 mutate(season = fct\_recode(season,  
 "Spring" = "1",  
 "Summer" = "2",  
 "Fall" = "3",  
 "Winter" = "4")) %>%  
 mutate(yr = as\_factor(as.character(yr))) %>%  
 mutate(mnth = as\_factor(as.character(mnth))) %>%  
 mutate(hr = as\_factor(as.character(hr))) %>%  
 mutate(holiday = as\_factor(as.character(holiday))) %>%  
 mutate(holiday = fct\_recode(holiday,  
 "NotHoliday" = "0",  
 "Holiday" = "1")) %>%  
 mutate(workingday = as\_factor(as.character(workingday))) %>%  
 mutate(workingday = fct\_recode(workingday,  
 "NotWorkingDay" = "0",  
 "WorkingDay" = "1")) %>%  
 mutate(weathersit = as\_factor(as.character(weathersit))) %>%  
 mutate(weathersit = fct\_recode(weathersit,  
 "NoPrecip" = "1",  
 "Misty" = "2",  
 "LightPrecip" = "3",  
 "HeavyPrecip" = "4")) %>%  
 mutate(weekday = as\_factor(as.character(weekday))) %>%  
 mutate(weekday = fct\_recode(weekday,  
 "Sunday" = "0",  
 "Monday" = "1",  
 "Tuesday" = "2",  
 "Wednesday" = "3",  
 "Thursday" = "4",  
 "Friday" = "5",  
 "Saturday" = "6"))

#### Task 1

# Set random seed  
set.seed(1234)  
# Split data for training and testing  
train.rows <- createDataPartition(y=bike$count,  
 p=0.7,  
 list = FALSE)  
train <- slice(bike, train.rows)  
test <- slice(bike, -train.rows)

#### Task 2

The function createDataPartition has separated the data for modeling. There are 12,167 observations in the train data set and 5,212 observations in the test data set.

#### Task 3

# Construct linear model  
mod1 = lm(count ~ season + mnth + hr + holiday + weekday +  
 temp + weathersit, train)  
summary(mod1)

##   
## Call:  
## lm(formula = count ~ season + mnth + hr + holiday + weekday +   
## temp + weathersit, data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -419.31 -61.93 -9.98 52.57 504.24   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -81.2946 6.9356 -11.721 < 2e-16 \*\*\*  
## seasonSummer 28.8486 6.4074 4.502 6.78e-06 \*\*\*  
## seasonFall 19.7865 7.6029 2.602 0.009266 \*\*   
## seasonWinter 62.0339 6.4333 9.643 < 2e-16 \*\*\*  
## mnth2 -0.8013 5.1396 -0.156 0.876114   
## mnth3 2.5584 5.7973 0.441 0.659003   
## mnth4 -1.2250 8.6334 -0.142 0.887166   
## mnth5 -1.5879 9.2279 -0.172 0.863382   
## mnth6 -15.3992 9.4846 -1.624 0.104485   
## mnth7 -38.8277 10.6085 -3.660 0.000253 \*\*\*  
## mnth8 -16.8557 10.3542 -1.628 0.103569   
## mnth9 5.4060 9.2152 0.587 0.557459   
## mnth10 -2.7341 8.5079 -0.321 0.747943   
## mnth11 -12.8043 8.2169 -1.558 0.119193   
## mnth12 -15.3615 6.5409 -2.349 0.018864 \*   
## hr1 -19.7855 6.9722 -2.838 0.004550 \*\*   
## hr2 -28.2440 6.9696 -4.052 5.10e-05 \*\*\*  
## hr3 -40.3146 7.0910 -5.685 1.34e-08 \*\*\*  
## hr4 -40.5469 7.0249 -5.772 8.03e-09 \*\*\*  
## hr5 -26.7454 6.9592 -3.843 0.000122 \*\*\*  
## hr6 32.8518 7.0435 4.664 3.13e-06 \*\*\*  
## hr7 161.3872 6.9925 23.080 < 2e-16 \*\*\*  
## hr8 312.2263 6.9502 44.923 < 2e-16 \*\*\*  
## hr9 164.2556 7.0163 23.411 < 2e-16 \*\*\*  
## hr10 107.1856 6.9552 15.411 < 2e-16 \*\*\*  
## hr11 139.6256 7.0057 19.930 < 2e-16 \*\*\*  
## hr12 179.7448 6.9778 25.760 < 2e-16 \*\*\*  
## hr13 178.6812 7.0201 25.453 < 2e-16 \*\*\*  
## hr14 156.2811 7.0628 22.127 < 2e-16 \*\*\*  
## hr15 168.7543 7.0939 23.788 < 2e-16 \*\*\*  
## hr16 228.1106 7.0881 32.182 < 2e-16 \*\*\*  
## hr17 377.6085 7.0185 53.802 < 2e-16 \*\*\*  
## hr18 347.7287 6.9806 49.813 < 2e-16 \*\*\*  
## hr19 238.7339 7.0128 34.043 < 2e-16 \*\*\*  
## hr20 159.7394 7.0231 22.745 < 2e-16 \*\*\*  
## hr21 108.1070 6.9494 15.556 < 2e-16 \*\*\*  
## hr22 72.3808 6.9874 10.359 < 2e-16 \*\*\*  
## hr23 32.5734 6.9996 4.654 3.30e-06 \*\*\*  
## holidayHoliday -29.0249 6.4088 -4.529 5.98e-06 \*\*\*  
## weekdaySunday -14.0349 3.7638 -3.729 0.000193 \*\*\*  
## weekdayMonday -6.5302 3.8944 -1.677 0.093604 .   
## weekdayTuesday -7.2790 3.8319 -1.900 0.057509 .   
## weekdayWednesday -3.2707 3.7984 -0.861 0.389212   
## weekdayThursday -1.7267 3.8053 -0.454 0.650004   
## weekdayFriday 1.3251 3.7744 0.351 0.725539   
## temp 288.1743 12.1860 23.648 < 2e-16 \*\*\*  
## weathersitMisty -19.6696 2.3717 -8.293 < 2e-16 \*\*\*  
## weathersitLightPrecip -94.1331 3.8166 -24.664 < 2e-16 \*\*\*  
## weathersitHeavyPrecip -80.2490 64.7672 -1.239 0.215356   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 111.9 on 12118 degrees of freedom  
## Multiple R-squared: 0.6217, Adjusted R-squared: 0.6202   
## F-statistic: 414.8 on 48 and 12118 DF, p-value: < 2.2e-16

We can see that the coefficient for seasonWinter is the strongest of all the seasons. This seems counterintuitive as one would expect summer to have more of an impact on the model. Also, looking at the month variable there are surprising results. The summer months all have negative coefficients. This is likely caused by collinearity as season is a predictor of month. Bike riding and commuting patterns overall tend to be seasonal and removing month might improve the model. We can also see that time of day, particularly the morning and evening commute times have the largest impact on the model which is expected. The model has a strong R-squared value of 0.63. However, removing features that show signs of collinearity would improve the R-squared value.

#### Task 4

# Examine training predictions  
train.pred <- predict(mod1, newdata = train)  
head(train.pred)

## 1 2 3 4 5 6   
## -37.68169 -46.14026 -52.44730 -52.67962 -58.54772 14.95557

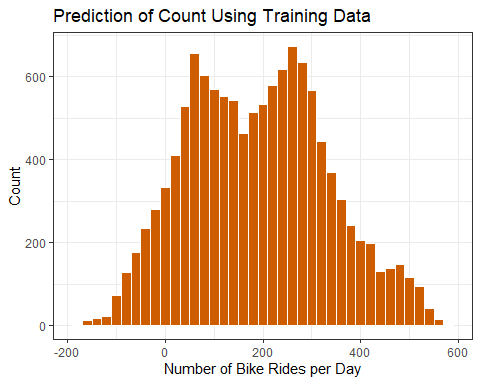
# Generate summaries  
summary(train.pred)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -183.90 75.04 188.25 189.33 289.19 584.44

summary(train$count)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.0 40.0 142.0 189.3 281.0 977.0

# Create histogram of predictions and actual counts  
x1 <- as.data.frame(train.pred)  
ggplot(x1, aes(train.pred)) +  
 geom\_histogram(binwidth = 20,  
 fill = "#CE5C00",  
 color = "white") +  
 ggtitle("Prediction of Count Using Training Data") +  
 xlab("Number of Bike Rides per Day") +  
 ylab("Count") +  
 theme\_bw()



Looking at the summary of the model’s prediction for count, we can see that there are negative values. This an unexpected result as there cannot be negative bike ridership. The model has a mean prediction of 189.33 rides per day and a maximum value of 584.44. The above summary of the acutal bike rideship data from the training data set gives us a sense of how the model performed. We see that the lowest daily count was 1 while the model predicted negative values. However, the mean values are very similar between the predictions and the actual counts. The maximum value in the actual data is nearly 1,000 which is a large gap between the two models. This is likely the result of some significant outliers in the training data set. The histogram reveals a bimodal distribution in the predictions with peaks around 75 and 300 rides per day. We also see a small percentage of predictions with negative values.

#### Task 5

# Examine testing predictions  
test.pred <- predict(mod1, newdata = test)  
head(test.pred)

## 1 2 3 4 5 6   
## -12.13272 137.72755 174.04493 17.56108 -22.20993 168.48847

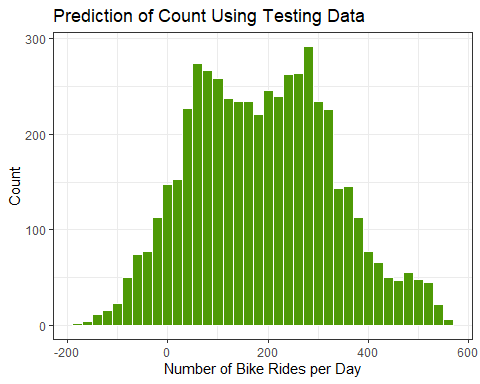
# Generate summaries  
summary(test.pred)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -187.00 78.66 188.34 189.41 288.72 567.39

summary(test$count)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.0 40.0 142.0 189.8 281.0 963.0

# Create histogram of predictions and actual counts  
x2 <- as.data.frame(test.pred)  
ggplot(x2, aes(test.pred)) +  
 geom\_histogram(binwidth = 20,  
 fill = "#4E9A06",  
 color = "white") +  
 ggtitle("Prediction of Count Using Testing Data") +  
 xlab("Number of Bike Rides per Day") +  
 ylab("Count") +  
 theme\_bw()



The summaries and distributions between the training data set and testing data set are almost identical. While the results do have negative values of bike rides, the model did perform well on the unseen data and predicted values similar to those of the training data.

#### Task 6

# Calculate R-squared value  
sse <- sum((test$count - test.pred)^2)  
sst <- sum((test$count - mean(test$count))^2)  
print(1 - sse/sst)

## [1] 0.6289223

The model has a strong R-squared value of 0.63. While the model did predict negative values, they only represent a small subset of the data. While 63 percent of the data can be explained by the model, it is likely that those negative values make up a portion of the 37 percent that is not explained by the model.

#### Task 7

The k-fold cross-validation technique and the train test split method are two ways to validate a data model. The latter splits the data set into two sections, a training portion and a testing portion. The modeler is able to specify what percentage of the data is used for training and fine tuning of the model. The testing set remains unseen by the model until final validation. This is also a disadvantage of the train test split approach. Data is intentionally withheld from the model while training and any patterns that exist in that data will be missed in training. This is where the k-folds cross-validation method shines. Using this approach, the modeler specifies how many segments to split the data into. This is the k in k-folds and is often 5 or 10 folds. The model will then be trained on k - 1 segments of the data while the last segment is saved for testing. This is repeated until all of the segments have been tested. This approach requires more processing time but yields more accurate results.