Module 3 Assignment 1 Model Validation

Laura Parsons

May 30, 2019

Load the necessary libraries.

#install.packages("caret")  
library(tidyverse, quietly = TRUE)

## -- Attaching packages ------------------------------------------ tidyverse 1.2.1 --

## v ggplot2 3.1.0 v purrr 0.3.2   
## v tibble 2.1.1 v dplyr 0.8.0.1  
## v tidyr 0.8.3 v stringr 1.4.0   
## v readr 1.3.1 v forcats 0.4.0

## -- Conflicts --------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(MASS)

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

Read in the dataset and convert variables season, yr, mnth, hr, holiday, workingday, weathersit, and weekday.

bike= read\_csv("hour.csv")

## Parsed with column specification:  
## cols(  
## instant = col\_double(),  
## dteday = col\_date(format = ""),  
## season = col\_double(),  
## yr = col\_double(),  
## mnth = col\_double(),  
## hr = col\_double(),  
## holiday = col\_double(),  
## weekday = col\_double(),  
## workingday = col\_double(),  
## weathersit = col\_double(),  
## temp = col\_double(),  
## atemp = col\_double(),  
## hum = col\_double(),  
## windspeed = col\_double(),  
## casual = col\_double(),  
## registered = col\_double(),  
## count = col\_double()  
## )

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: Split the data into training and testing data sets.

set.seed(1234)  
train.rows = createDataPartition(y = bike$count, p=0.7, list = FALSE)   
train = bike[train.rows,]   
test = bike[-train.rows,]

Task 2: Within the training dataset there are 12167 rows of data. Within the testing dataset there are 5212 rows of data.

Task 3: Build a linear regression model using the training data.

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   
## -411.57 -62.29 -9.66 51.54 494.52   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -87.1390 6.9960 -12.456 < 2e-16 \*\*\*  
## seasonSummer 34.0014 6.3399 5.363 8.33e-08 \*\*\*  
## seasonFall 27.1663 7.4964 3.624 0.000291 \*\*\*  
## seasonWinter 60.2453 6.3962 9.419 < 2e-16 \*\*\*  
## mnth2 0.6289 5.1046 0.123 0.901951   
## mnth3 7.4480 5.7452 1.296 0.194867   
## mnth4 -6.6612 8.5213 -0.782 0.434401   
## mnth5 -6.2329 9.1424 -0.682 0.495407   
## mnth6 -15.8184 9.3673 -1.689 0.091306 .   
## mnth7 -39.2578 10.4561 -3.755 0.000174 \*\*\*  
## mnth8 -21.7608 10.2226 -2.129 0.033300 \*   
## mnth9 1.3338 9.0877 0.147 0.883319   
## mnth10 0.9570 8.4836 0.113 0.910185   
## mnth11 -15.1008 8.1639 -1.850 0.064382 .   
## mnth12 -12.2448 6.4726 -1.892 0.058542 .   
## hr1 -13.3293 6.9652 -1.914 0.055682 .   
## hr2 -27.4480 7.0006 -3.921 8.87e-05 \*\*\*  
## hr3 -33.8591 7.0797 -4.783 1.75e-06 \*\*\*  
## hr4 -37.7544 7.1298 -5.295 1.21e-07 \*\*\*  
## hr5 -20.8072 7.0678 -2.944 0.003247 \*\*   
## hr6 37.4750 7.0673 5.303 1.16e-07 \*\*\*  
## hr7 174.5062 6.9408 25.142 < 2e-16 \*\*\*  
## hr8 310.6002 7.0497 44.059 < 2e-16 \*\*\*  
## hr9 172.3560 7.0135 24.575 < 2e-16 \*\*\*  
## hr10 112.8882 7.0375 16.041 < 2e-16 \*\*\*  
## hr11 139.8538 7.0762 19.764 < 2e-16 \*\*\*  
## hr12 182.1016 7.0797 25.722 < 2e-16 \*\*\*  
## hr13 177.8863 7.0168 25.351 < 2e-16 \*\*\*  
## hr14 163.2828 7.1329 22.891 < 2e-16 \*\*\*  
## hr15 178.1201 7.0976 25.096 < 2e-16 \*\*\*  
## hr16 231.1350 7.1679 32.246 < 2e-16 \*\*\*  
## hr17 382.4767 7.0346 54.371 < 2e-16 \*\*\*  
## hr18 361.1422 7.1736 50.343 < 2e-16 \*\*\*  
## hr19 237.1363 7.0249 33.757 < 2e-16 \*\*\*  
## hr20 166.4963 6.9865 23.831 < 2e-16 \*\*\*  
## hr21 114.6982 6.9704 16.455 < 2e-16 \*\*\*  
## hr22 75.1763 7.0002 10.739 < 2e-16 \*\*\*  
## hr23 35.4147 6.9890 5.067 4.10e-07 \*\*\*  
## holidayHoliday -21.8882 6.4894 -3.373 0.000746 \*\*\*  
## weekdaySunday -16.5691 3.7640 -4.402 1.08e-05 \*\*\*  
## weekdayMonday -7.9035 3.8915 -2.031 0.042277 \*   
## weekdayTuesday -7.1190 3.7953 -1.876 0.060717 .   
## weekdayWednesday -7.4042 3.7927 -1.952 0.050938 .   
## weekdayThursday -0.9102 3.7787 -0.241 0.809662   
## weekdayFriday -0.3409 3.7732 -0.090 0.928011   
## temp 288.5138 12.1631 23.721 < 2e-16 \*\*\*  
## weathersitMisty -19.1163 2.3603 -8.099 6.06e-16 \*\*\*  
## weathersitLightPrecip -90.5259 3.7350 -24.237 < 2e-16 \*\*\*  
## weathersitHeavyPrecip 83.0764 111.2351 0.747 0.455166   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 111 on 12118 degrees of freedom  
## Multiple R-squared: 0.6229, Adjusted R-squared: 0.6214   
## F-statistic: 417.1 on 48 and 12118 DF, p-value: < 2.2e-16

This appears to be a high quality model. The adjusted R-squared value is 0.6214 which is a good adjusted R-squared value. However, when observing the variables in the model it is odd that the estimate associated with heavy precipitation is positive while the values for mist and light precipitation are negative. This means that the variable weathersit might be highly correlated with another variable and might indicate multicollinearity.

Task 4: Make predictions on the training set.

predict\_train = predict(mod1, newdata = train)  
head(predict\_train)

## 1 2 3 4 5 6   
## -36.99526 -51.11404 -51.75482 -55.65016 -57.81925 13.80902

Most of the predictions are negative and in the fifties. However there is also a prediction of -36 and 13. Since this model is made to predict future counts it is odd that it is predicting several negative counts for bike rides. Since these predictions are negative which is unexpected it might indicate an under-fitted model.

Task 5: Make predictions on the testing set.

predict\_test = predict(mod1, newdata = test)  
head(predict\_test)

## 1 2 3 4 5 6   
## -17.895722 177.541411 156.579769 216.138357 204.347307 9.891889

Most of these predictions are positive and between 150 and 205. However there is also a prediction of -17 and 9. Even though these predictions seem more likely then the predictions for the training dataset most of the values are close together and it would have been better to see more variability within the predictions. Since there are still negative predictions it might indicate an under-fitted model.

Task 6: Manually calculate the R-squared value on the testing dataset.

SSE = sum((test$count - predict\_test)^2)   
SST = sum((test$count - mean(test$count))^2)   
1 - SSE/SST

## [1] 0.6250483

The R-squared value for the testing dataset is 0.625. The R-squared value for the training dataset was 0.621. Since the R-squared value is slightly higher for the testing dataset than the training dataset it indicates that the model created with the training dataset is likely to preform similarly on new data.

Task 7: Describe how kfold cross validation differs from model validation via a training testing split.  
K-fold cross validation splits the dataset into K data sets. You then find the optimal model K times using all data sets except 1 each time and alternating which one is excluded until all data sets have been excluded once. You then average all of the models together to create the best possible model. K-fold cross validation is preferable because all of the data points are represented in the final model however, since they were excluded from one of the previous models it helps reduce the possibility for over-fitting. The training/testing method requires you to split the data into a training dataset (between 70-80% of the original data) and a testing dataset (between 20-30% of the original data). The testing dataset is then used to build the best possible model. After the model is produced it is tested with the testing dataset. Since the testing dataset was not used to help build the model it could potentially affect the model and the model could be under-fit.