San Francisco Bike Share

Ryan Porter

5/5/2020

#Predicting Daily Bike ride numbers in the San Francisco Area

The goal of this project is to create a model that can predict the amount of bike rides per day. I have three different datasets one with the trips, another is the stations that the bike was taken and returned to, and the final dataset is the weather for that day.

## Loading Packages & Files

library(tidyverse)  
library(lubridate)  
library(caret)  
  
#Load Data  
trip <- read.csv("trip.csv")  
weather <- read.csv("weather.csv")  
station <- read.csv("station.csv")

The following code is to get the dates into the proper format.

weather$date <- mdy(weather$date)  
trip$start\_date <- mdy\_hm(trip$start\_date)  
trip$end\_date <- mdy\_hm(trip$end\_date)  
trip$date <- trip$start\_date  
trip$date <- as.Date(trip$date)  
  
trip$id2 <- trip$id  
trip$id <- trip$start\_station\_id   
trip <- left\_join(trip, station, by = c ("id")) #join the station datset to the trip dataset

##Daily Ride Counts

I ran two different models one with Weekends filtered out but the RMSE error doubled so I decide to keep it. It makes sense because it is a strong predictor in the linear Regression model.

dailyrides <- as.data.frame(table(trip$date, trip$city))  
colnames(dailyrides) <- c("date","city", "ridecount")  
dailyrides$date <- as.Date(dailyrides$date)  
  
dailyrides$weekend <- as.factor(wday(dailyrides$date))  
dailyrides$weekend <- (dailyrides$weekend == 1 | dailyrides$weekend == 7) #Sunday = 1 and Saturday = 7  
dailyrides$weekend <- factor(dailyrides$weekend, labels = c("Weekday", "Weekend"))  
  
#dailyrides <- filter(dailyrides, weekend == "Weekday")

table(dailyrides$city) #the distribution of rides by city

##   
## Mountain View Palo Alto Redwood City San Francisco San Jose   
## 733 733 733 733 733

##Add Weather data

The last dataset is weather data for each of the days. It has a lot of variables so I only took the variables that were averages and the events that happened each day.

zip\_code <- unique(weather$zip\_code)  
city <- c ("San Francisco", "Redwood City", "Palo Alto", "Mountain View", "San Jose")  
index <- cbind(city, zip\_code)   
weather <- merge(weather, index, by = "zip\_code")  
  
weather2 <- weather %>%  
 select(zip\_code, date ,mean\_temperature\_f,   
 mean\_humidity, mean\_dew\_point\_f, mean\_wind\_speed\_mph,  
 events, city)

##Events of the weather

As you can see from the table we have a lot of missing values for events. Since four of the five events include rain then I made a new variable that states wether is rained that day or not. Events that were just fog were classified as no rain days.

table(weather2$events)

##   
## Fog Fog-Rain rain   
## 3143 112 17 2   
## Rain Rain-Thunderstorm   
## 388 3

weather2$events <- factor(weather2$events)  
weather2$rain <- ifelse(unclass(weather2$events) > 2  
 , c("rain"), c("no rain"))  
  
table(weather2$rain) # 410 days it rained

##   
## no rain rain   
## 3255 410

weather2$rain <- factor(weather2$rain)  
  
#Merge to dailyrides dataframe  
dailyrides <- left\_join(dailyrides, weather2, by = c("date", "city"))

##Dealing with missing data

I just used the average. I thought about using a Linear model for predicting the missing values but the variance wasn’t very large so it would do the job in this instance. Plus the amount of missing values wasn’t very large like mean\_wind\_speed was missing one value. Even though they give you min and max for each of them the ones missing the means values were also missing those values so it was a lack of data.

sapply(dailyrides, function(x) {sum(is.na(x))})

## date city ridecount weekend   
## 0 0 0 0   
## zip\_code mean\_temperature\_f mean\_humidity mean\_dew\_point\_f   
## 0 4 54 54   
## mean\_wind\_speed\_mph events rain   
## 1 0 0

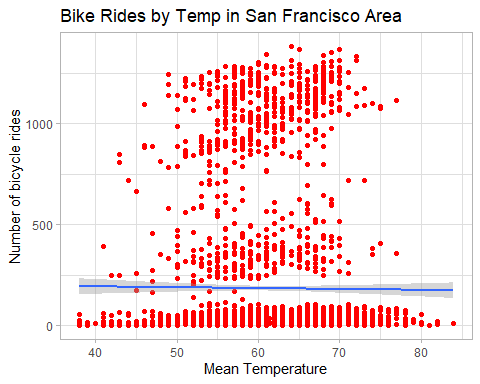
dailyrides <- dailyrides %>%  
 mutate(mean\_temperature\_f = ifelse(is.na(mean\_temperature\_f),   
 mean(mean\_temperature\_f, na.rm = TRUE), mean\_temperature\_f),  
 mean\_humidity = ifelse(is.na(mean\_humidity),   
 mean(mean\_humidity, na.rm = TRUE), mean\_humidity),  
 mean\_dew\_point\_f = ifelse(is.na(mean\_dew\_point\_f),   
 mean(mean\_dew\_point\_f, na.rm = TRUE), mean\_dew\_point\_f),  
 mean\_wind\_speed\_mph = ifelse(is.na(mean\_wind\_speed\_mph),   
 mean(mean\_wind\_speed\_mph, na.rm = TRUE), mean\_wind\_speed\_mph))  
  
sapply(dailyrides, function(x) {sum(is.na(x))})

## date city ridecount weekend   
## 0 0 0 0   
## zip\_code mean\_temperature\_f mean\_humidity mean\_dew\_point\_f   
## 0 0 0 0   
## mean\_wind\_speed\_mph events rain   
## 0 0 0

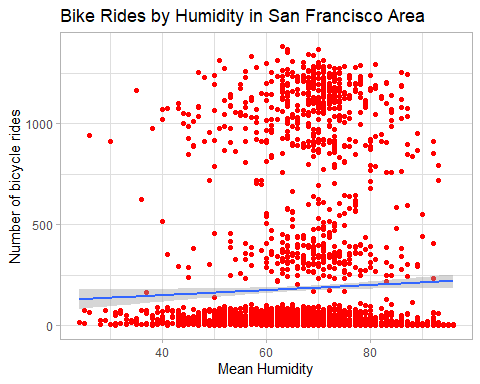
## Plots

Daily ride counts by the predictor variables that will be used in the machine learning. The last plot makes it clear that a lot of the bike rides started in San Francisco and only a small portion are from the surrounding areas.

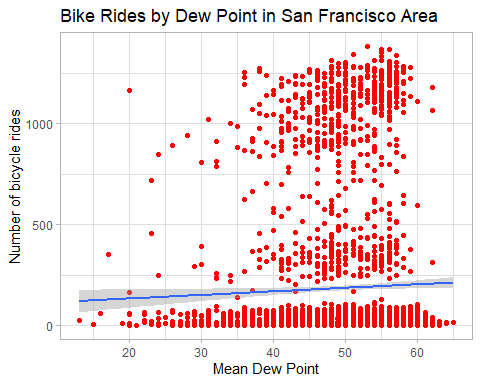
## `geom\_smooth()` using formula 'y ~ x'



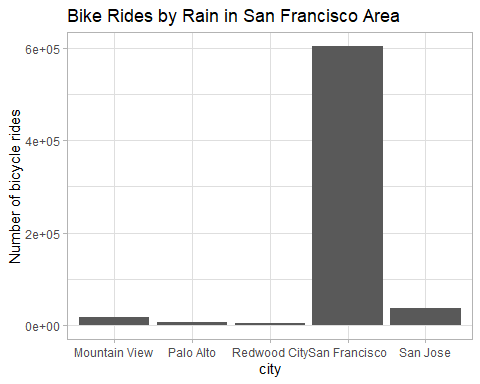
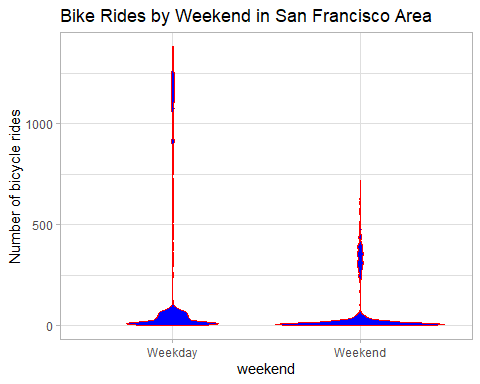
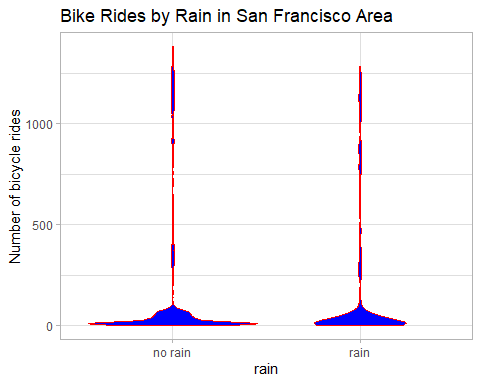
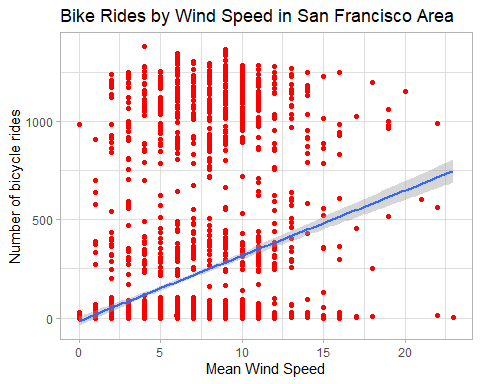
## `geom\_smooth()` using formula 'y ~ x'



## `geom\_smooth()` using formula 'y ~ x'



## `geom\_smooth()` using formula 'y ~ x'



##Make train and test sets for Machine Learning

index <- createDataPartition(dailyrides$ridecount, times = 1, p = 0.5, list = FALSE)  
train\_data <- dailyrides[index,]  
test\_data <- dailyrides[-index,]  
  
set.seed(1234) #set the seed even though it seems not to matter  
ctrl <- trainControl(method = "repeatedcv", repeats = 3) #cross validation

##Linear Regression

Rain was significant predictor and averaged about 31 less bike rides a day if it were raining that day. San Francisco was a big predictor which makes sense because a lot of our data was from there. I did run a model that filtered for just San Francisco but the RMSE got a lot bigger so I decided to leave it in. The most suprising thing was that weekends was signifcant but in the opposite way. There were 142 less bike rides a day if it was a weekend which I thought would be the opposite. It would be interesting to look at the locations of where the bike rides are occuring and see if they are using them to get to work.

lm.fit <- train(ridecount ~ rain + mean\_temperature\_f + mean\_humidity +  
 mean\_wind\_speed\_mph + mean\_dew\_point\_f + weekend + city,  
 data = train\_data,  
 method = "lm",  
 trControl = ctrl)  
  
summary(lm.fit)

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -751.85 -49.78 -25.02 80.90 486.78   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -42.4329 76.8672 -0.552 0.58100   
## rainrain -44.6929 12.0721 -3.702 0.00022 \*\*\*  
## mean\_temperature\_f 0.7185 1.6463 0.436 0.66257   
## mean\_humidity -0.3247 0.9262 -0.351 0.72592   
## mean\_wind\_speed\_mph -0.4941 1.2733 -0.388 0.69803   
## mean\_dew\_point\_f 1.9157 1.8065 1.060 0.28910   
## weekendWeekend -144.2715 7.8417 -18.398 < 2e-16 \*\*\*  
## `cityPalo Alto` -20.0864 11.7254 -1.713 0.08687 .   
## `cityRedwood City` -18.5958 12.7661 -1.457 0.14538   
## `citySan Francisco` 798.7695 12.0338 66.377 < 2e-16 \*\*\*  
## `citySan Jose` 29.1814 11.4388 2.551 0.01082 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 151.7 on 1823 degrees of freedom  
## Multiple R-squared: 0.8255, Adjusted R-squared: 0.8245   
## F-statistic: 862.3 on 10 and 1823 DF, p-value: < 2.2e-16

pred.lm <- predict(lm.fit, test\_data)  
Lm.RMSE <- RMSE(pred.lm, test\_data$ridecount) #149.50  
Lm.RMSE

## [1] 149.5682

##Partial Least Squares

library(pls)

##   
## Attaching package: 'pls'

## The following object is masked from 'package:caret':  
##   
## R2

## The following object is masked from 'package:stats':  
##   
## loadings

pls.fit <- train(ridecount ~ rain + mean\_temperature\_f + mean\_humidity +  
 mean\_wind\_speed\_mph + mean\_dew\_point\_f + weekend + city,   
 data = train\_data,   
 method = "pls",  
 trControl = ctrl,   
 preProc = c("center", "scale"),  
 #tuneLength = 30)  
 tuneGrid = data.frame(ncomp=9))  
  
pls.fit

## Partial Least Squares   
##   
## 1834 samples  
## 7 predictor  
##   
## Pre-processing: centered (10), scaled (10)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 1650, 1651, 1651, 1651, 1650, 1650, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 152.1035 0.824423 100.3175  
##   
## Tuning parameter 'ncomp' was held constant at a value of 9

pred.pls <- predict(pls.fit, test\_data)  
Pls.RMSE <- RMSE(pred.pls, test\_data$ridecount) #149.503  
Pls.RMSE

## [1] 149.5679

##Elastic Net

library(elasticnet)

## Loading required package: lars

## Loaded lars 1.2

enetGrid <- expand.grid(.lambda = c(0,0.01, .1), .fraction = seq(.05, 1, length = 20))  
  
enet.fit <- train(ridecount ~ rain + mean\_temperature\_f + mean\_humidity +  
 mean\_wind\_speed\_mph + mean\_dew\_point\_f + weekend + city,   
 data = train\_data,   
 method = "enet",   
 trControl = ctrl,   
 preProc = c("center", "scale"),   
 tuneGrid = enetGrid)   
  
enet.fit

## Elasticnet   
##   
## 1834 samples  
## 7 predictor  
##   
## Pre-processing: centered (10), scaled (10)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 1650, 1651, 1650, 1650, 1652, 1650, ...   
## Resampling results across tuning parameters:  
##   
## lambda fraction RMSE Rsquared MAE   
## 0.00 0.05 341.3955 0.7890963 241.57481  
## 0.00 0.10 321.7538 0.7890963 223.91920  
## 0.00 0.15 302.5504 0.7890963 207.02873  
## 0.00 0.20 283.8750 0.7890963 191.51642  
## 0.00 0.25 265.8389 0.7890963 177.42288  
## 0.00 0.30 248.5813 0.7890963 164.30987  
## 0.00 0.35 232.2745 0.7890963 151.66271  
## 0.00 0.40 217.1296 0.7890963 139.37479  
## 0.00 0.45 203.4003 0.7890963 127.43519  
## 0.00 0.50 191.3813 0.7890963 116.09908  
## 0.00 0.55 181.4126 0.7892356 105.80088  
## 0.00 0.60 173.5541 0.7978880 97.72330  
## 0.00 0.65 166.7731 0.8071064 91.68505  
## 0.00 0.70 161.3118 0.8133778 91.87174  
## 0.00 0.75 157.3223 0.8172685 94.98602  
## 0.00 0.80 154.9217 0.8198731 97.27598  
## 0.00 0.85 153.4186 0.8222862 97.80154  
## 0.00 0.90 152.4312 0.8239475 98.42786  
## 0.00 0.95 151.8704 0.8248628 99.37115  
## 0.00 1.00 151.9246 0.8245804 100.43789  
## 0.01 0.05 341.5082 0.7890963 241.67661  
## 0.01 0.10 321.9746 0.7890963 224.11750  
## 0.01 0.15 302.8735 0.7890963 207.30782  
## 0.01 0.20 284.2926 0.7890963 191.85428  
## 0.01 0.25 266.3408 0.7890963 177.80564  
## 0.01 0.30 249.1541 0.7890963 164.74416  
## 0.01 0.35 232.9005 0.7890963 152.15116  
## 0.01 0.40 217.7859 0.7890963 139.92646  
## 0.01 0.45 204.0577 0.7890963 128.03216  
## 0.01 0.50 192.0034 0.7890963 116.71346  
## 0.01 0.55 181.9419 0.7890963 106.39125  
## 0.01 0.60 174.0430 0.7971861 98.21548  
## 0.01 0.65 167.2059 0.8065867 91.98030  
## 0.01 0.70 161.6621 0.8130263 91.61445  
## 0.01 0.75 157.5514 0.8170757 94.70111  
## 0.01 0.80 155.0669 0.8196373 97.18108  
## 0.01 0.85 153.5416 0.8220569 97.87298  
## 0.01 0.90 152.5072 0.8238192 98.34712  
## 0.01 0.95 151.9157 0.8247706 99.20845  
## 0.01 1.00 151.9337 0.8245721 100.43215  
## 0.10 0.05 340.3807 0.7890963 240.65713  
## 0.10 0.10 319.7670 0.7890963 222.14207  
## 0.10 0.15 299.6477 0.7890963 204.55759  
## 0.10 0.20 280.1299 0.7890963 188.51642  
## 0.10 0.25 261.3487 0.7890963 174.01950  
## 0.10 0.30 243.4740 0.7890963 160.36620  
## 0.10 0.35 226.7183 0.7890963 147.24182  
## 0.10 0.40 211.3428 0.7890963 134.42376  
## 0.10 0.45 197.6603 0.7890963 122.11729  
## 0.10 0.50 186.0296 0.7890963 110.65863  
## 0.10 0.55 176.8859 0.7929561 101.00698  
## 0.10 0.60 169.2210 0.8041293 93.67530  
## 0.10 0.65 162.9485 0.8117637 90.83501  
## 0.10 0.70 158.2333 0.8165704 94.07375  
## 0.10 0.75 155.5464 0.8193623 96.03251  
## 0.10 0.80 154.2470 0.8209885 96.80584  
## 0.10 0.85 153.2511 0.8224442 97.48293  
## 0.10 0.90 152.7018 0.8231423 98.36497  
## 0.10 0.95 152.8354 0.8226122 99.55229  
## 0.10 1.00 153.3877 0.8213675 101.00228  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were fraction = 0.95 and lambda = 0.

pred.enet <- predict(enet.fit, test\_data)  
Enet.RMSE <- RMSE(pred.enet, test\_data$ridecount) #149.615  
Enet.RMSE

## [1] 149.5803

##Neural Network

nnGrid <- expand.grid(.decay = c(0,0.01,.1), .size = c(1:10))   
  
nn.fit <- train(ridecount ~ rain + mean\_temperature\_f + mean\_humidity +  
 mean\_wind\_speed\_mph + mean\_dew\_point\_f + weekend + city,   
 data = train\_data ,   
 method = "nnet",  
 trControl = ctrl,   
 preProc = c("center", "scale"),   
 tuneGrid = nnGrid)

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :  
## There were missing values in resampled performance measures.

pred.nn <- predict(nn.fit, test\_data)  
NN.RMSE <- RMSE(pred.nn, test\_data$ridecount) #401.664  
NN.RMSE

pred.nn <- predict(nn.fit, test\_data)  
NN.RMSE <- RMSE(pred.nn, test\_data$ridecount) #401.664  
NN.RMSE

## [1] 402.053

##Mars

The second best model.

library(earth)

## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos

marsGrid <- expand.grid(.degree = 1:2, .nprune = 2:34)  
  
mars.fit <- train(ridecount ~ rain + mean\_temperature\_f + mean\_humidity +  
 mean\_wind\_speed\_mph + mean\_dew\_point\_f + weekend + city,   
 data = train\_data ,   
 method = "earth",  
 tuneGrid = marsGrid,  
 trControl = ctrl)  
  
mars.fit

## Multivariate Adaptive Regression Spline   
##   
## 1834 samples  
## 7 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 1650, 1650, 1651, 1650, 1650, 1652, ...   
## Resampling results across tuning parameters:  
##   
## degree nprune RMSE Rsquared MAE   
## 1 2 167.09385 0.7877093 82.66460  
## 1 3 154.07219 0.8189265 101.46566  
## 1 4 152.83170 0.8217970 101.26766  
## 1 5 152.55714 0.8225083 100.35240  
## 1 6 152.42879 0.8228261 100.55953  
## 1 7 152.62759 0.8223932 100.77527  
## 1 8 152.71063 0.8221546 100.93364  
## 1 9 152.71063 0.8221546 100.93364  
## 1 10 152.71063 0.8221546 100.93364  
## 1 11 152.71063 0.8221546 100.93364  
## 1 12 152.71063 0.8221546 100.93364  
## 1 13 152.71063 0.8221546 100.93364  
## 1 14 152.71063 0.8221546 100.93364  
## 1 15 152.71063 0.8221546 100.93364  
## 1 16 152.71063 0.8221546 100.93364  
## 1 17 152.71063 0.8221546 100.93364  
## 1 18 152.71063 0.8221546 100.93364  
## 1 19 152.71063 0.8221546 100.93364  
## 1 20 152.71063 0.8221546 100.93364  
## 1 21 152.71063 0.8221546 100.93364  
## 1 22 152.71063 0.8221546 100.93364  
## 1 23 152.71063 0.8221546 100.93364  
## 1 24 152.71063 0.8221546 100.93364  
## 1 25 152.71063 0.8221546 100.93364  
## 1 26 152.71063 0.8221546 100.93364  
## 1 27 152.71063 0.8221546 100.93364  
## 1 28 152.71063 0.8221546 100.93364  
## 1 29 152.71063 0.8221546 100.93364  
## 1 30 152.71063 0.8221546 100.93364  
## 1 31 152.71063 0.8221546 100.93364  
## 1 32 152.71063 0.8221546 100.93364  
## 1 33 152.71063 0.8221546 100.93364  
## 1 34 152.71063 0.8221546 100.93364  
## 2 2 167.09385 0.7877093 82.66460  
## 2 3 100.46978 0.9221645 46.77106  
## 2 4 92.78318 0.9339661 44.14302  
## 2 5 91.43405 0.9352651 43.42748  
## 2 6 90.28874 0.9368519 42.38413  
## 2 7 90.02471 0.9373344 40.64050  
## 2 8 89.65201 0.9377791 40.07794  
## 2 9 87.33771 0.9408863 37.66716  
## 2 10 87.03807 0.9413358 37.38975  
## 2 11 87.59407 0.9405542 37.54820  
## 2 12 87.58611 0.9405537 37.57590  
## 2 13 87.59608 0.9405436 37.57173  
## 2 14 87.59608 0.9405436 37.57173  
## 2 15 87.59608 0.9405436 37.57173  
## 2 16 87.59608 0.9405436 37.57173  
## 2 17 87.59608 0.9405436 37.57173  
## 2 18 87.59608 0.9405436 37.57173  
## 2 19 87.59608 0.9405436 37.57173  
## 2 20 87.59608 0.9405436 37.57173  
## 2 21 87.59608 0.9405436 37.57173  
## 2 22 87.59608 0.9405436 37.57173  
## 2 23 87.59608 0.9405436 37.57173  
## 2 24 87.59608 0.9405436 37.57173  
## 2 25 87.59608 0.9405436 37.57173  
## 2 26 87.59608 0.9405436 37.57173  
## 2 27 87.59608 0.9405436 37.57173  
## 2 28 87.59608 0.9405436 37.57173  
## 2 29 87.59608 0.9405436 37.57173  
## 2 30 87.59608 0.9405436 37.57173  
## 2 31 87.59608 0.9405436 37.57173  
## 2 32 87.59608 0.9405436 37.57173  
## 2 33 87.59608 0.9405436 37.57173  
## 2 34 87.59608 0.9405436 37.57173  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final values used for the model were nprune = 10 and degree = 2.

pred.mars <- predict(mars.fit, test\_data)  
Mars.RMSE <- RMSE(pred.mars, test\_data$ridecount) #84.910  
Mars.RMSE

## [1] 83.57733

##Knn

Best Model. Had the lowest RMSE

knn.fit <- train(ridecount ~ rain + mean\_temperature\_f + mean\_humidity +  
 mean\_wind\_speed\_mph + mean\_dew\_point\_f + weekend + city,   
 data = train\_data ,   
 method = "knn",  
 preProc = c("center", "scale"),  
 tuneGrid = data.frame(k=seq(1,101,2)),  
 trControl = ctrl)  
  
knn.fit$finalModel #9

## 5-nearest neighbor regression model

pred.knn <- predict(knn.fit, test\_data)  
Knn.RMSE <- RMSE(pred.knn, test\_data$ridecount) #79.565  
Knn.RMSE

## [1] 85.44089