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)
```

```r
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")
```

```r
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.

```r
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.

```r
table(weather2$events)
```

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

```r
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
```

```r
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.

```r
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
```

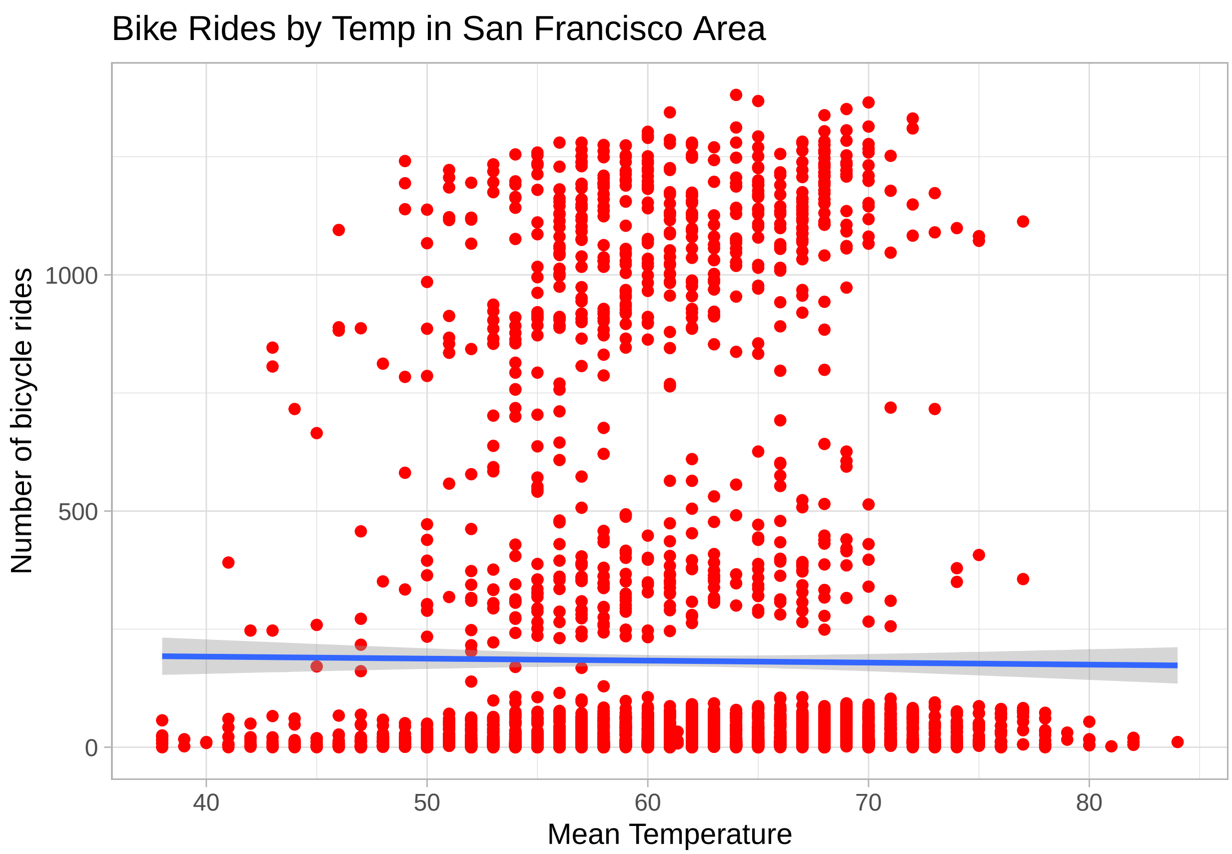
```r
dailyrides <- dailyrides %>%
  mutate(mean_temperature_f = ifelse(is.na(mean_temperature_f),
                                     mean(mean_temperature_f, na.rm = TRUE), mean_temperature_f),
         mean_humidity = ifelse(is.na(mean_humidity),
                                mean(mean_humidity, 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
```

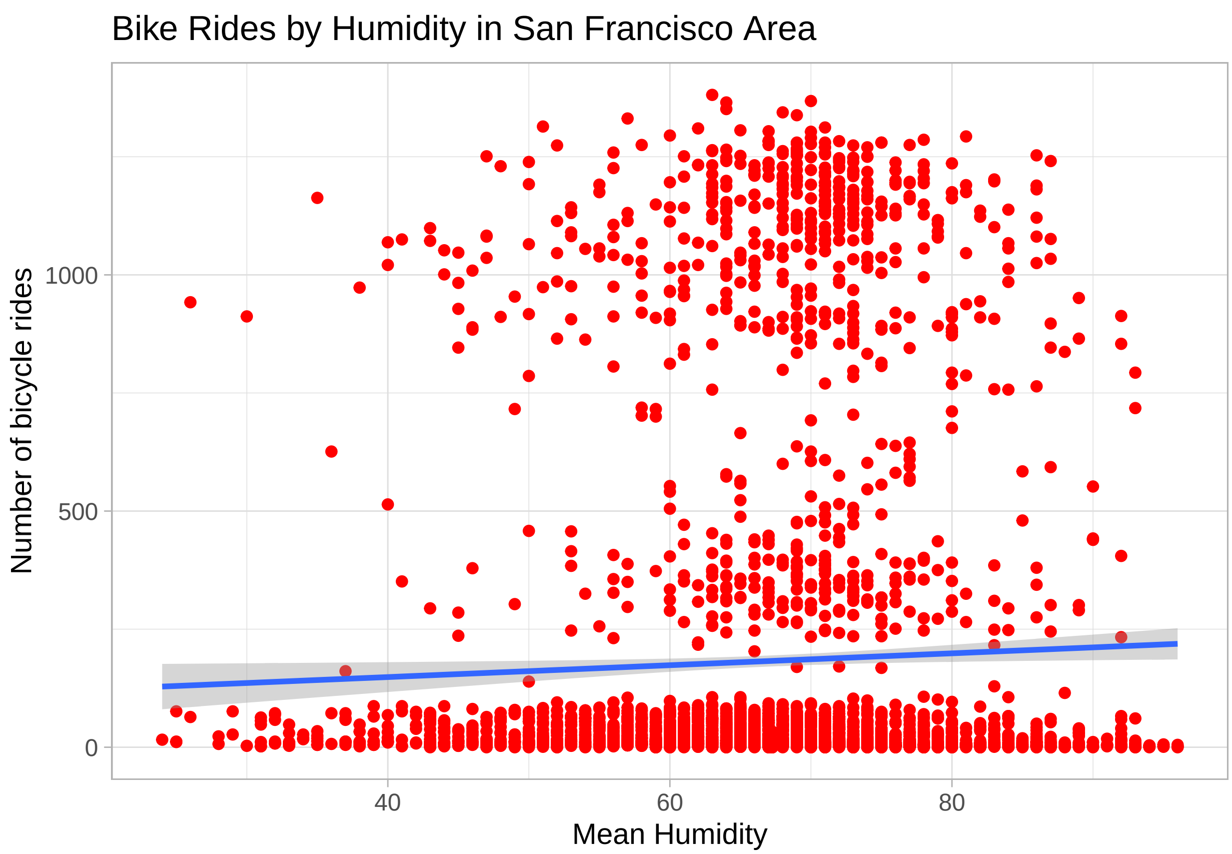
Plots

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.

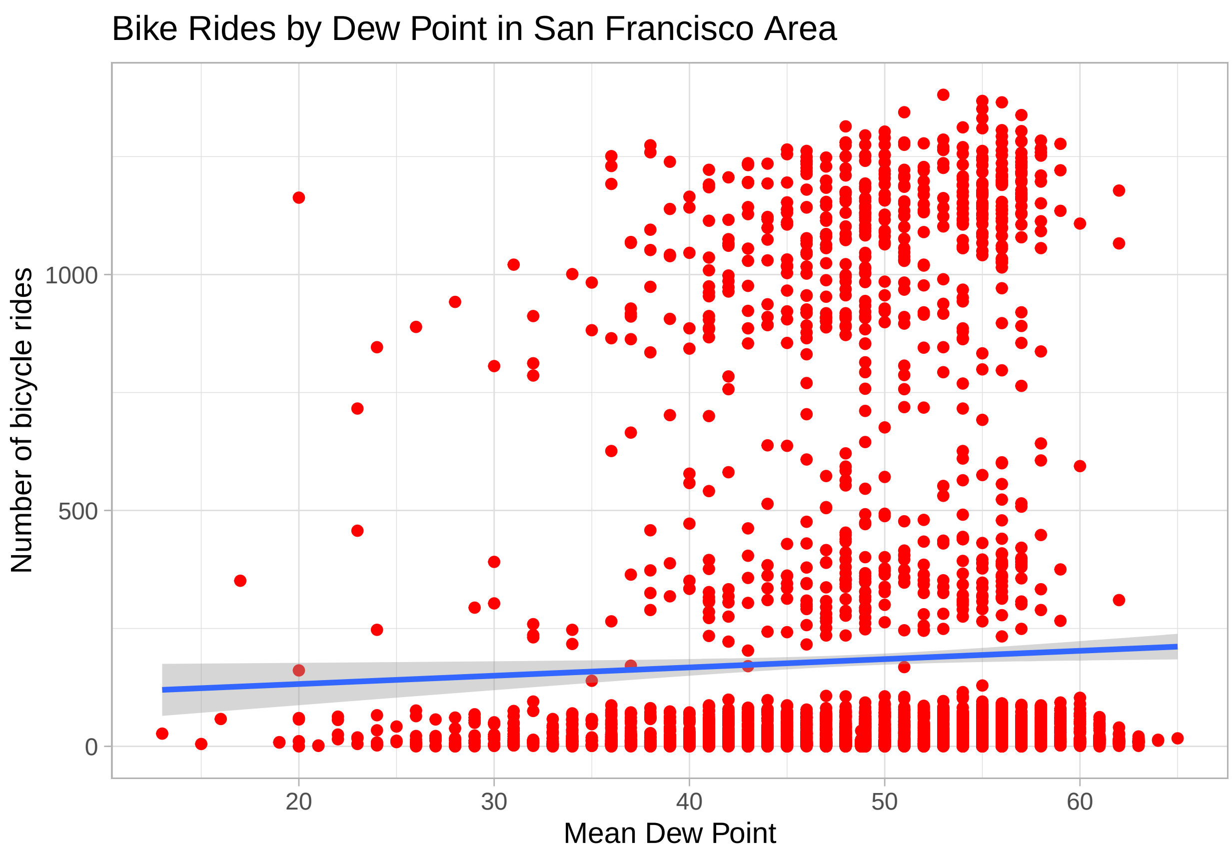
```
## `geom_smooth()` using formula 'y ~ x'
```



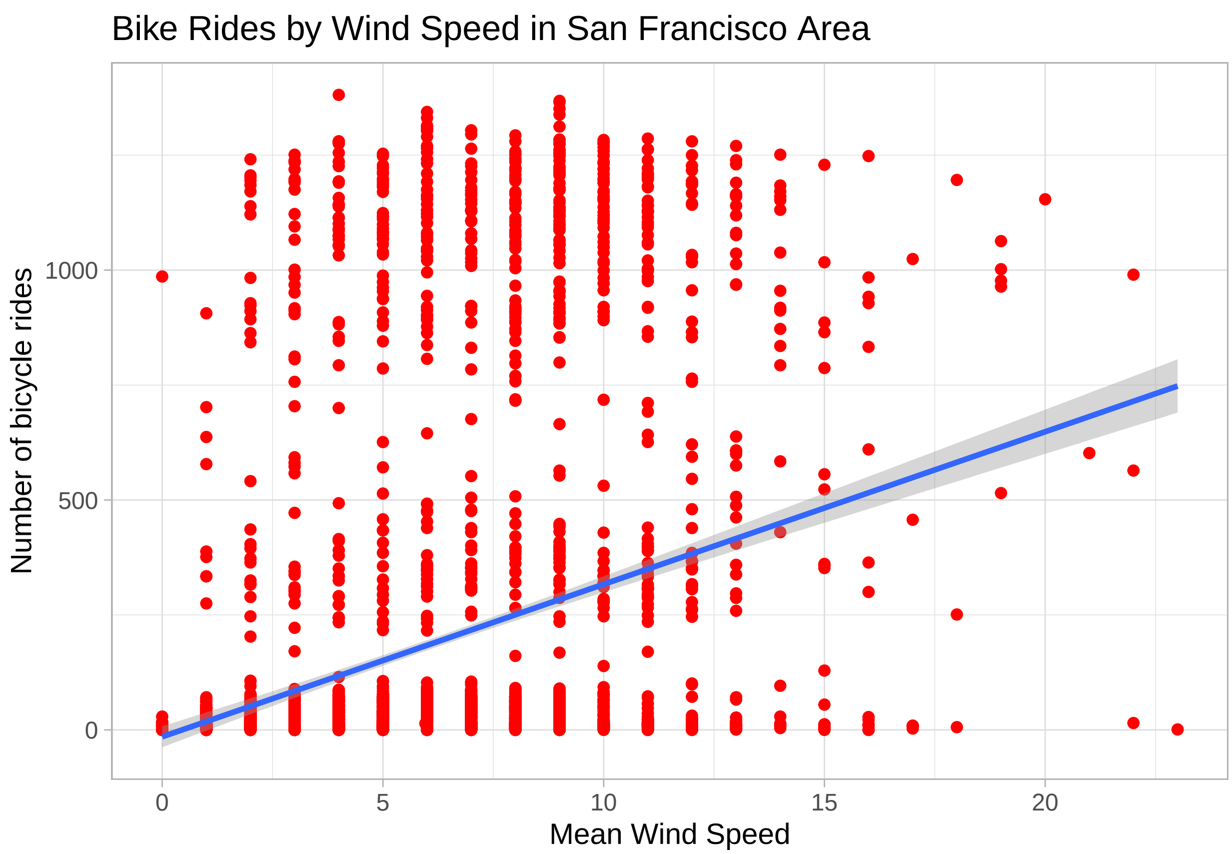

```
## `geom_smooth()` using formula 'y ~ x'
```

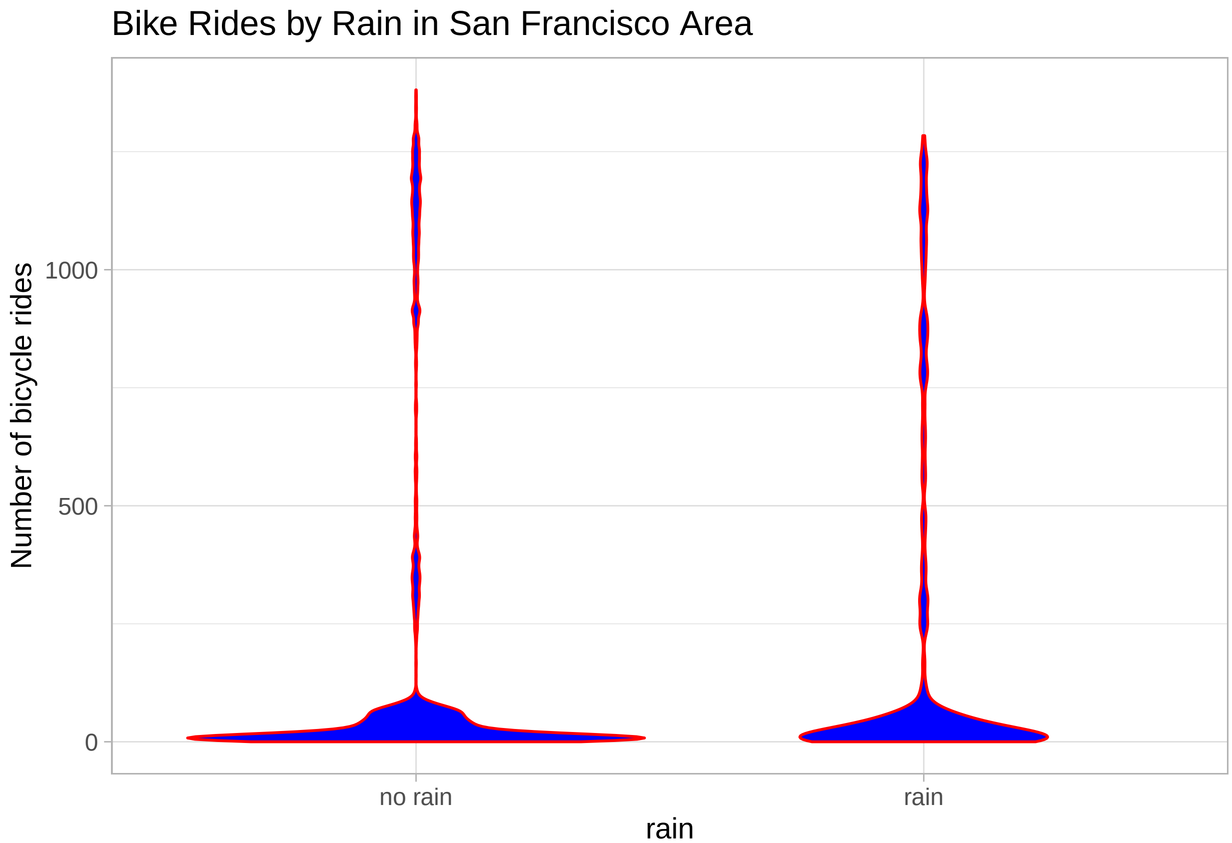


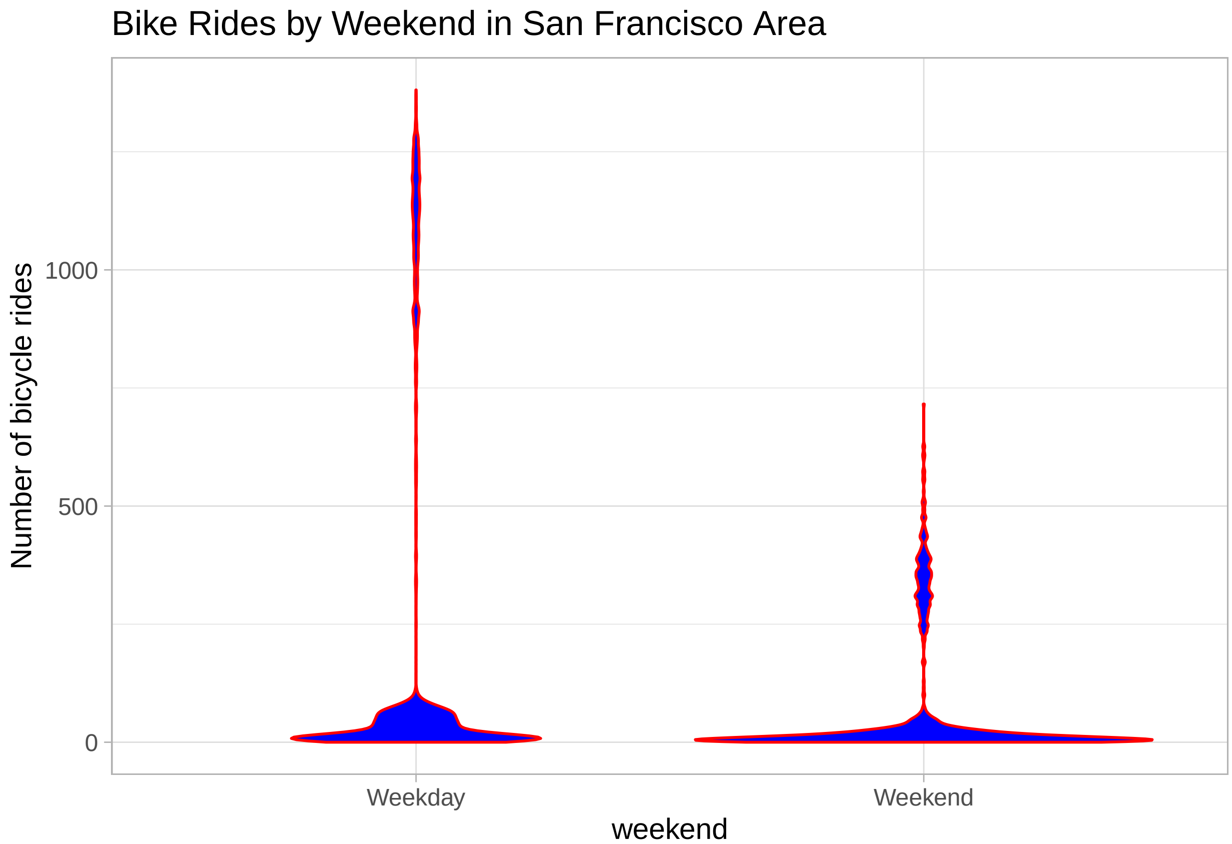

```
## `geom_smooth()` using formula 'y ~ x'
```



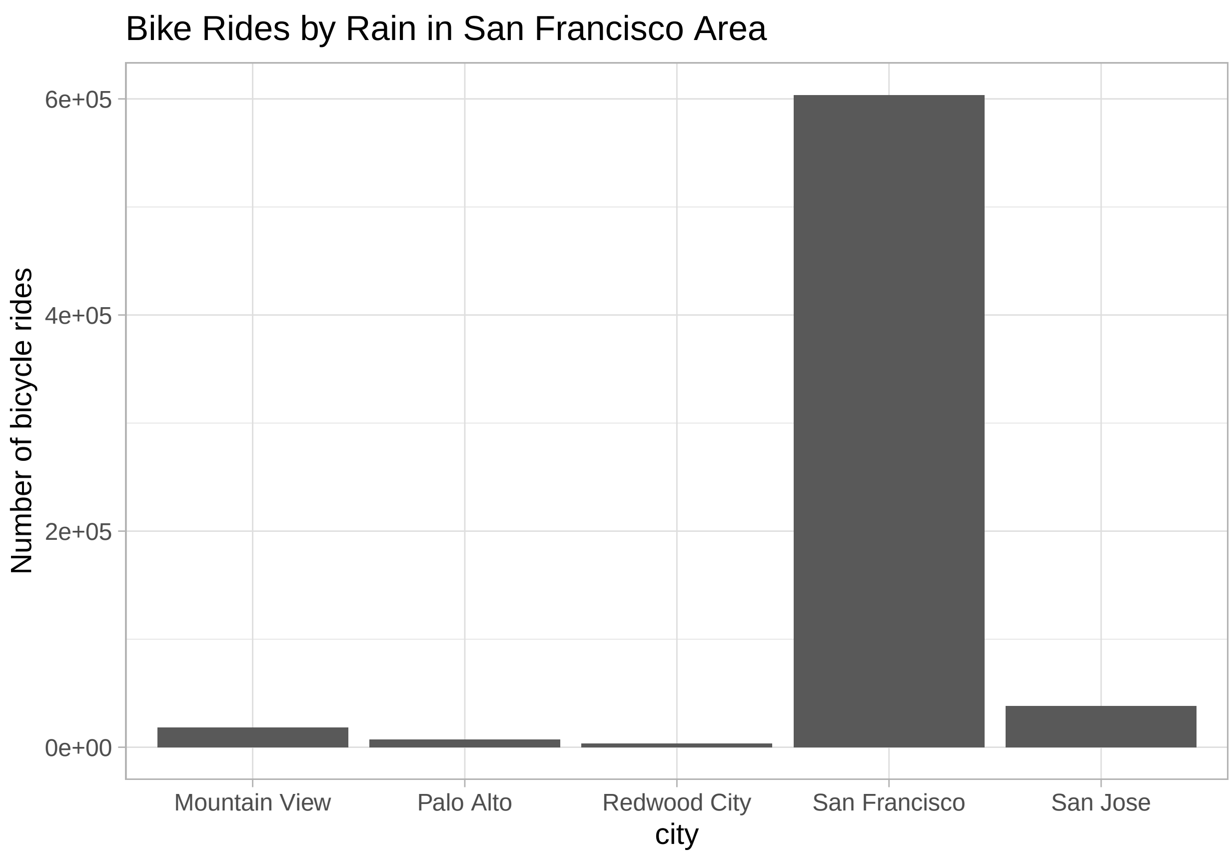

```
## `geom_smooth()` using formula 'y ~ x'
```


Bike Rides by Wind Speed in San Francisco Area
Number of bicycle rides
1000
500
0
0
5
10
15
20
Mean Wind Speed


Bike Rides by Rain in San Francisco Area
Number of bicycle rides
1000
500
0
no rain
rain
rain



Bike Rides by Weekend in San Francisco Area
Number of bicycle rides
1000
500
0
Weekday
Weekend
weekend



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)
```

```r
summary(lm.fit)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -739.64  -43.37  -23.67   81.90  449.46
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -23.45302   76.47064  -0.307   0.7591
## rainrain              -10.63694   11.84198  -0.898   0.3692
## mean_temperature_f      1.15349    1.63735   0.704   0.4812
## mean_humidity          -0.64074    0.90495  -0.708   0.4790
## mean_wind_speed_mph     0.07602    1.24582   0.061   0.9513
## mean_dew_point_f        1.19933    1.77281   0.677   0.4988
## weekendWeekend       -139.33824    7.66833 -18.171   <2e-16 ***
## `cityPalo Alto`       -19.49482   11.20121  -1.740   0.0820 .
## `cityRedwood City`    -12.38490   12.06521  -1.026   0.3048
## `citySan Francisco`   831.40856   11.81893  70.346   <2e-16 ***
## `citySan Jose`         25.39801   10.96397   2.316   0.0206 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 145.9 on 1823 degrees of freedom
## Multiple R-squared:  0.8434, Adjusted R-squared:  0.8425
## F-statistic: 981.5 on 10 and 1823 DF,  p-value: < 2.2e-16
```

```r
pred.lm <- predict(lm.fit, test_data)
Lm.RMSE <- RMSE(pred.lm, test_data$ridecount) #149.50
Lm.RMSE
```

```
## [1] 156.4445
```

Partial Least Squares

```r
library(pls)
```

```
##
## Attaching package: 'pls'

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

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

```r
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
##   146.1167  0.8424221  93.72185
##
## Tuning parameter 'ncomp' was held constant at a value of 9
```

```r
pred.pls <- predict(pls.fit, test_data)
Pls.RMSE <- RMSE(pred.pls, test_data$ridecount) #149.503
Pls.RMSE
```

```
## [1] 156.4622
```

Elastic Net

```r
library(elasticnet)
```

```
## Loading required package: lars
```

```
## Loaded lars 1.2
```

```r
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      347.1218  0.8119049  245.76499
##   0.00    0.10      327.5314  0.8119049  228.65239
##   0.00    0.15      308.3046  0.8119049  212.37366
##   0.00    0.20      289.5141  0.8119049  197.29371
##   0.00    0.25      271.2512  0.8119049  183.17379
##   0.00    0.30      253.6301  0.8119049  169.66434
##   0.00    0.35      236.7942  0.8119049  156.68633
##   0.00    0.40      220.9222  0.8119049  143.94259
##   0.00    0.45      206.2346  0.8119049  131.44346
##   0.00    0.50      192.9976  0.8119049  119.52156
##   0.00    0.55      181.5215  0.8119049  108.39168
##   0.00    0.60      172.1854  0.8123674   98.52233
##   0.00    0.65      164.6836  0.8220803   91.03793
##   0.00    0.70      158.2899  0.8297938   85.94785
##   0.00    0.75      153.2714  0.8349581   87.72227
##   0.00    0.80      149.7708  0.8380565   90.65410
##   0.00    0.85      147.7340  0.8404536   92.06413
##   0.00    0.90      146.5041  0.8423273   92.29966
##   0.00    0.95      146.0029  0.8429956   93.18637
##   0.00    1.00      146.0947  0.8427120   93.88207
##   0.01    0.05      347.1681  0.8119049  245.80572
##   0.01    0.10      327.6228  0.8119049  228.73197
##   0.01    0.15      308.4392  0.8119049  212.48625
##   0.01    0.20      289.6897  0.8119049  197.43443
##   0.01    0.25      271.4647  0.8119049  183.33789
##   0.01    0.30      253.8776  0.8119049  169.85451
##   0.01    0.35      237.0703  0.8119049  156.90434
##   0.01    0.40      221.2198  0.8119049  144.19073
##   0.01    0.45      206.5445  0.8119049  131.70574
##   0.01    0.50      193.3077  0.8119049  119.79632
##   0.01    0.55      181.8162  0.8119049  108.67054
##   0.01    0.60      172.4341  0.8121844   98.76520
##   0.01    0.65      164.9078  0.8218602   91.26772
##   0.01    0.70      158.4859  0.8296450   85.99696
##   0.01    0.75      153.4270  0.8348742   87.62245
##   0.01    0.80      149.8683  0.8380284   90.54834
##   0.01    0.85      147.8370  0.8402603   92.27559
##   0.01    0.90      146.6179  0.8421379   92.40300
##   0.01    0.95      146.1011  0.8428116   93.20814
##   0.01    1.00      146.0966  0.8427059   93.93881
##   0.10    0.05      345.9268  0.8119049  244.71063
##   0.10    0.10      325.1839  0.8119049  226.62667
##   0.10    0.15      304.8597  0.8119049  209.55698
```

```
##   0.10    0.20      285.0445  0.8119049  193.82501
##   0.10    0.25      265.8527  0.8119049  179.03024
##   0.10    0.30      247.4297  0.8119049  164.92578
##   0.10    0.35      229.9603  0.8119049  151.29584
##   0.10    0.40      213.6768  0.8119049  137.85772
##   0.10    0.45      198.8668  0.8119049  124.88136
##   0.10    0.50      185.8751  0.8119049  112.68434
##   0.10    0.55      175.0947  0.8119049  101.63343
##   0.10    0.60      166.6659  0.8197366   93.03437
##   0.10    0.65      159.5510  0.8286341   86.62156
##   0.10    0.70      153.9528  0.8345198   87.37238
##   0.10    0.75      150.3679  0.8375987   90.08827
##   0.10    0.80      148.4905  0.8396311   90.71211
##   0.10    0.85      147.4286  0.8406888   91.59185
##   0.10    0.90      146.9448  0.8410605   92.66693
##   0.10    0.95      147.1986  0.8403437   93.73106
##   0.10    1.00      147.7472  0.8392035   95.03012
##
## 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.
```

```r
pred.enet <- predict(enet.fit, test_data)
Enet.RMSE <- RMSE(pred.enet, test_data$ridecount) #149.615
Enet.RMSE
```

```
## [1] 156.4995
```

Neural Network

```r
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.
```

```r
pred.nn <- predict(nn.fit, test_data)
NN.RMSE <- RMSE(pred.nn, test_data$ridecount) #401.664
NN.RMSE
```


```r
pred.nn <- predict(nn.fit, test_data)
NN.RMSE <- RMSE(pred.nn, test_data$ridecount) #401.664
NN.RMSE
```


```
## [1] 396.1724
```

Mars

The second best model.

```r
library(earth)
```

```
## Loading required package: Formula

## Loading required package: plotmo

## Loading required package: plotrix

## Loading required package: TeachingDemos
```

```r
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      160.02993  0.8130321  77.76364
##   1        3      147.42240  0.8410077  95.87971
##   1        4      146.71637  0.8425420  95.53048
##   1        5      146.02298  0.8440185  94.14146
##   1        6      146.02298  0.8440185  94.14146
##   1        7      146.02298  0.8440185  94.14146
##   1        8      146.02298  0.8440185  94.14146
##   1        9      146.02298  0.8440185  94.14146
##   1       10      146.02298  0.8440185  94.14146
##   1       11      146.02298  0.8440185  94.14146
##   1       12      146.02298  0.8440185  94.14146
##   1       13      146.02298  0.8440185  94.14146
##   1       14      146.02298  0.8440185  94.14146
##   1       15      146.02298  0.8440185  94.14146
##   1       16      146.02298  0.8440185  94.14146
##   1       17      146.02298  0.8440185  94.14146
```

```
##    1        18       146.02298  0.8440185  94.14146
##    1        19       146.02298  0.8440185  94.14146
##    1        20       146.02298  0.8440185  94.14146
##    1        21       146.02298  0.8440185  94.14146
##    1        22       146.02298  0.8440185  94.14146
##    1        23       146.02298  0.8440185  94.14146
##    1        24       146.02298  0.8440185  94.14146
##    1        25       146.02298  0.8440185  94.14146
##    1        26       146.02298  0.8440185  94.14146
##    1        27       146.02298  0.8440185  94.14146
##    1        28       146.02298  0.8440185  94.14146
##    1        29       146.02298  0.8440185  94.14146
##    1        30       146.02298  0.8440185  94.14146
##    1        31       146.02298  0.8440185  94.14146
##    1        32       146.02298  0.8440185  94.14146
##    1        33       146.02298  0.8440185  94.14146
##    1        34       146.02298  0.8440185  94.14146
##    2         2       160.02993  0.8130321  77.76364
##    2         3        91.80955  0.9374554  44.67754
##    2         4        85.68126  0.9456848  42.70427
##    2         5        85.66301  0.9455825  39.41147
##    2         6        84.40577  0.9471151  36.90580
##    2         7        83.88839  0.9477040  36.53782
##    2         8        83.85073  0.9477474  36.53023
##    2         9        83.85073  0.9477474  36.53023
##    2        10        83.85073  0.9477474  36.53023
##    2        11        83.85073  0.9477474  36.53023
##    2        12        83.85073  0.9477474  36.53023
##    2        13        83.85073  0.9477474  36.53023
##    2        14        83.85073  0.9477474  36.53023
##    2        15        83.85073  0.9477474  36.53023
##    2        16        83.85073  0.9477474  36.53023
##    2        17        83.85073  0.9477474  36.53023
##    2        18        83.85073  0.9477474  36.53023
##    2        19        83.85073  0.9477474  36.53023
##    2        20        83.85073  0.9477474  36.53023
##    2        21        83.85073  0.9477474  36.53023
##    2        22        83.85073  0.9477474  36.53023
##    2        23        83.85073  0.9477474  36.53023
##    2        24        83.85073  0.9477474  36.53023
##    2        25        83.85073  0.9477474  36.53023
##    2        26        83.85073  0.9477474  36.53023
##    2        27        83.85073  0.9477474  36.53023
##    2        28        83.85073  0.9477474  36.53023
##    2        29        83.85073  0.9477474  36.53023
##    2        30        83.85073  0.9477474  36.53023
##    2        31        83.85073  0.9477474  36.53023
##    2        32        83.85073  0.9477474  36.53023
##    2        33        83.85073  0.9477474  36.53023
##    2        34        83.85073  0.9477474  36.53023
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nprune = 8 and degree = 2.
```

```r
pred.mars <- predict(mars.fit, test_data)
Mars.RMSE <- RMSE(pred.mars, test_data$ridecount) #84.910
Mars.RMSE
```

```
## [1] 89.33572
```

Knn

Best Model. Had the lowest RMSE

```r
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
```

```
## 7-nearest neighbor regression model
```

```r
pred.knn <- predict(knn.fit, test_data)
Knn.RMSE <- RMSE(pred.knn, test_data$ridecount) #79.565
Knn.RMSE
```

```
## [1] 90.81753
```