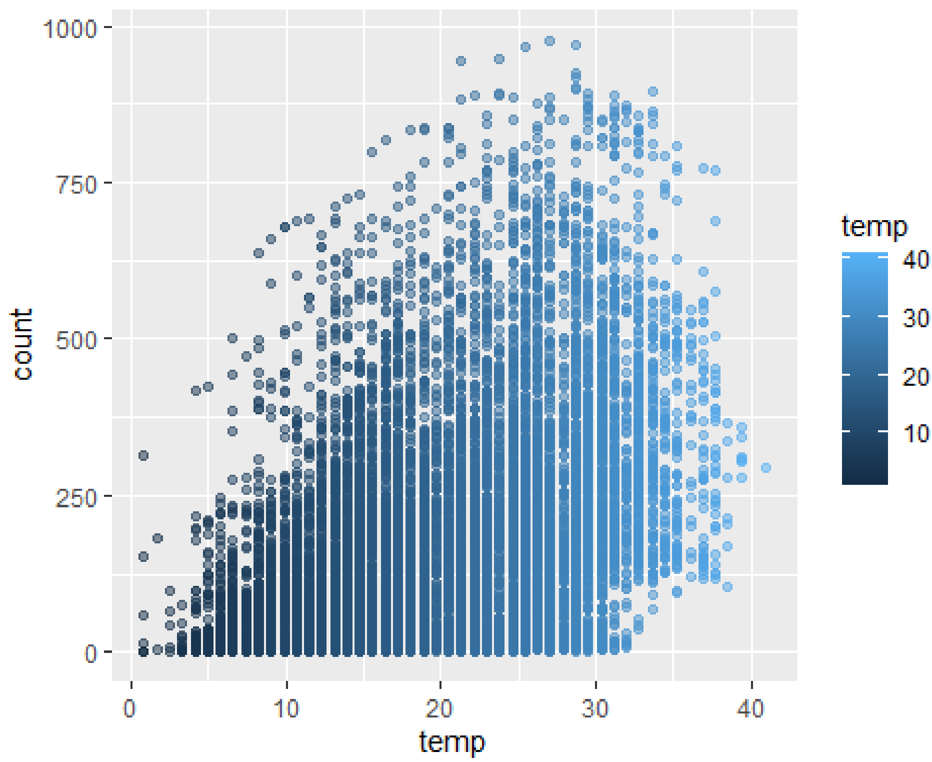
Bike_Rent_Model_Prediction_Updated

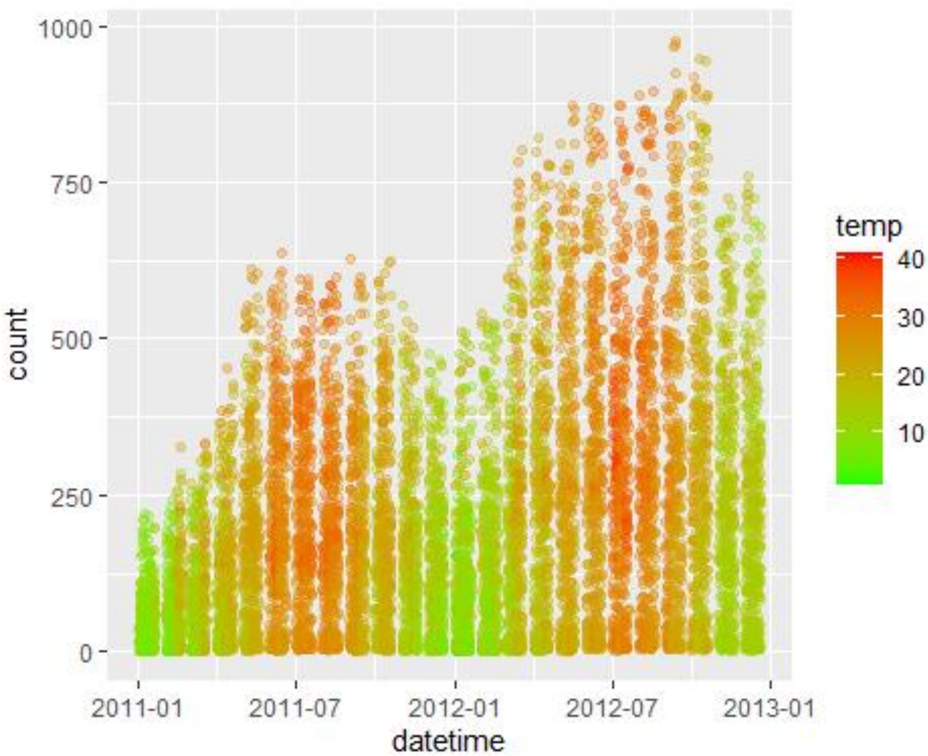
Nabil Momin

2024-06-16

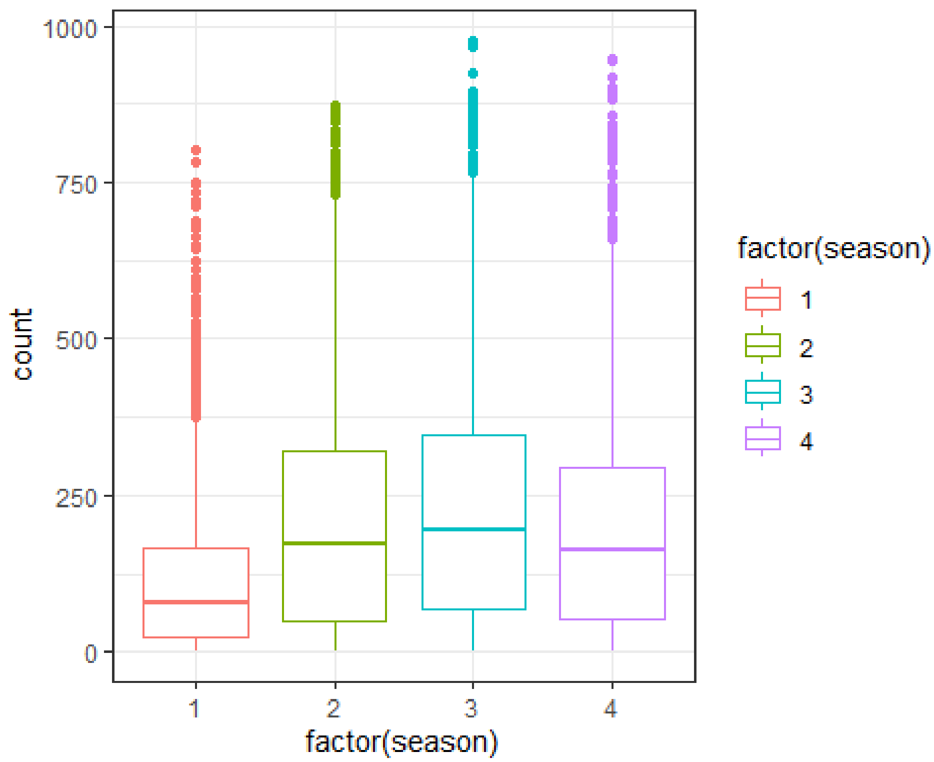
```
library(corrgram)
library(corrplot)

## corrplot 0.92 loaded

library(caTools)
library(Amelia)

## Loading required package: Rcpp

## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.8.2, built: 2024-04-10)
## ## Copyright (C) 2005-2024 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##

library(ggplot2)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(corrgram)
library(corrplot)
library(caTools)


# making bike as the data frame for containing the bike.csv file

bike <- read.csv('bikeshare.csv')

# making the ggplot to really see how is count related to other factors

ggplot(bike,aes(temp,count)) + geom_point(aes(color=temp),alpha=0.5)
```



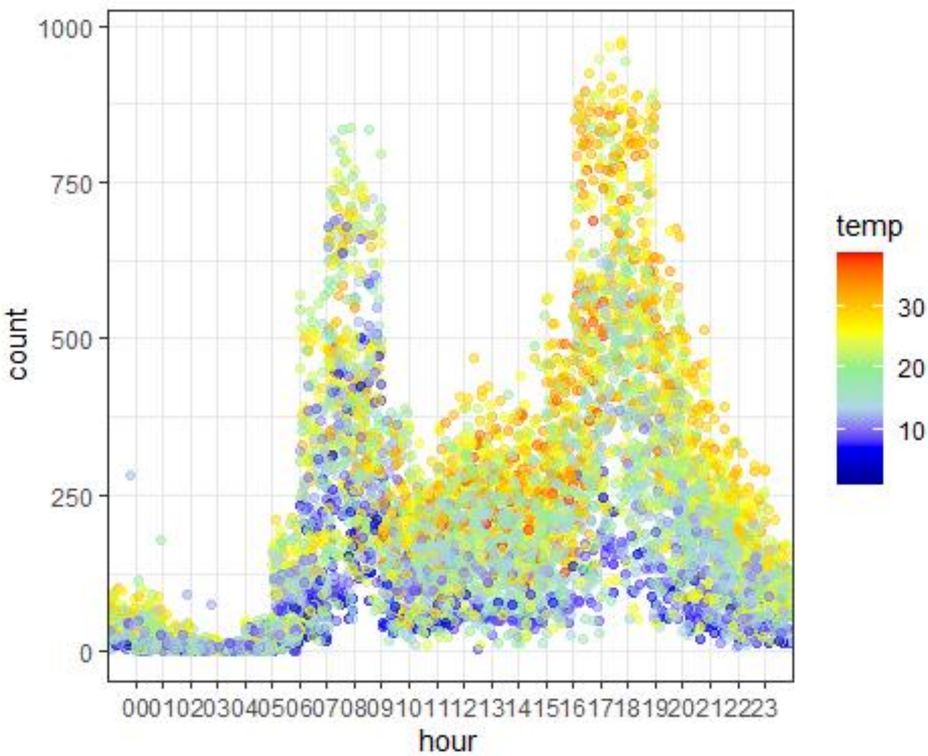

```r
# we see that the count of bike rented is higher when the temp is higher

bike$datetime <- as.POSIXct(bike$datetime)

ggplot(bike,aes(datetime,count)) + geom_point(aes(color=temp),alpha=0.3) +
scale_color_continuous(low='green',high='red')
```



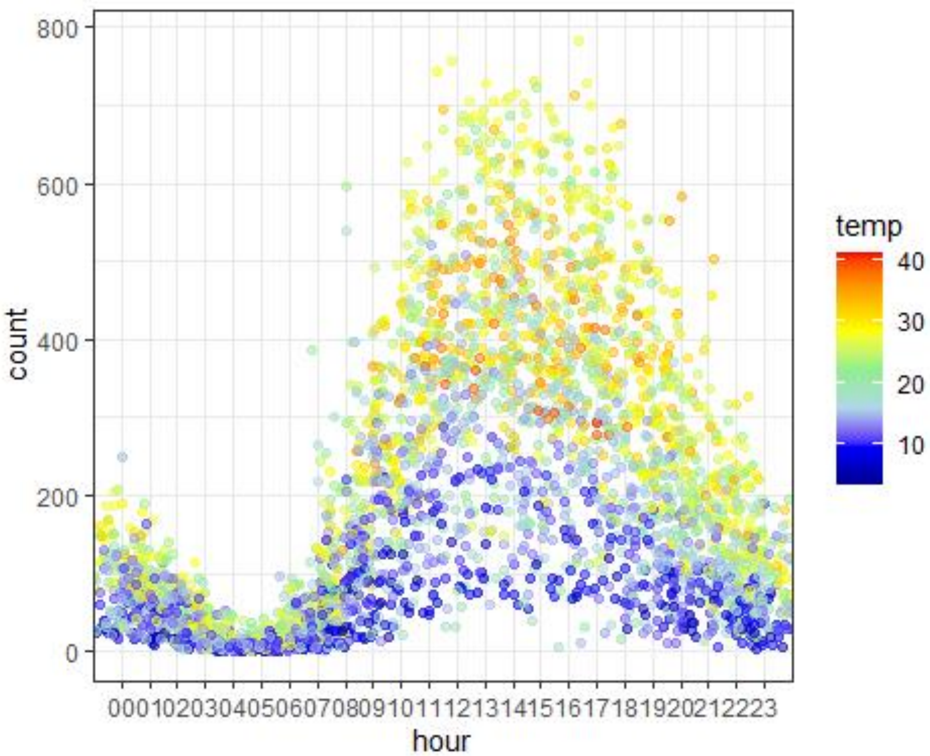

```r
# here we see that the bike rented is higher when its summer months

cor(bike[,c('temp','count')])

##            temp     count
## temp  1.0000000 0.3944536
## count 0.3944536 1.0000000

# from the above correlation function we see they are related by 0.4

pl <- ggplot(bike,aes(factor(season),count)) +
geom_boxplot(aes(color=factor(season)))

pl + theme_bw()
```




```
# we see from the above ggplot that counts of bike rented are higher when its
summer and fall
# making a new column for just hour to see which hour has the most rented
bike

bike$hour <- sapply(bike$datetime,function(x){format(x,'%H')})

pl <- ggplot(filter(bike,workingday==1),aes(hour,count)) +
geom_point(position=position_jitter(w=1, h=0),aes(color=temp),alpha=0.5)

pl <- pl + scale_color_gradientn(colours = c('dark blue','blue','light
blue','light green','yellow','orange','red'))
pl + theme_bw()
```




```
# from the ggplot we can deduce that rented bikes are higher during rush hour
in the weekdays

# Now we will see how is the rent of bike during the day in the weekends

pl <- ggplot(filter(bike,workingday==0),aes(hour,count)) +
geom_point(position=position_jitter(w=1, h=0),aes(color=temp),alpha=0.5)

pl <- pl + scale_color_gradientn(colours = c('dark blue','blue','light
blue','light green','yellow','orange','red'))

pl <- pl + theme_bw()

pl
```




```
# we see that during the weekend the rented bikes are more during afternoon
to sunset

# Now lets build the model as we have enough EDA

model <- lm(count ~ temp,bike)

summary(model)

##
## Call:
## lm(formula = count ~ temp, data = bike)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -293.32 -112.36  -33.36   78.98  741.44
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.0462     4.4394   1.362    0.173
## temp          9.1705     0.2048  44.783   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 166.5 on 10884 degrees of freedom
```

```
## Multiple R-squared:  0.1556, Adjusted R-squared:  0.1555
## F-statistic:  2006 on 1 and 10884 DF,  p-value: < 2.2e-16

# so we have our working model and if someone asks us how many bike will be
rented when the temp is 25
# we have two ways to answer that one is by using the residual data and the
other one is using predict function
# I like more the predict way so we will do that here
# know that our model is only made with two things in mind, count being
affected by temp so if someone asks how many bike will be
# rented during weekends, then our model cant answer that, just to remember
few things

temp.test <- data.frame(temp=c(25))

predict(model,temp.test)

##        1
## 235.3097

# we have our answer, if the temp is 25 then the bikes rented are 235

#### Now we will do something quite interesting and slick if you would like,
lets now see how it computes when we put it to test depending on all factors
#### not just the column temperature
#### lets see how well our model performs when all the factors are considered
into the equation.
#### EXCITED

#### now we will go a step further and made the model which can predict the
count based on all factors of the equation
#### checking which we can factor

str(bike)

## 'data.frame':    10886 obs. of  13 variables:
##  $ datetime  : POSIXct, format: "2011-01-01 00:00:00" "2011-01-01
01:00:00" ...
##  $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ workingday: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weather   : int  1 1 1 1 1 2 1 1 1 1 ...
##  $ temp      : num  9.84 9.02 9.02 9.84 9.84 ...
##  $ atemp     : num  14.4 13.6 13.6 14.4 14.4 ...
##  $ humidity  : int  81 80 80 75 75 75 80 86 75 76 ...
##  $ windspeed : num  0 0 0 0 0 ...
##  $ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
##  $ registered: int  13 32 27 10 1 1 0 2 7 6 ...
##  $ count     : int  16 40 32 13 1 1 2 3 8 14 ...
##  $ hour      : chr  "00" "01" "02" "03" ...
```

```r
#### factoring now

bike$season <- factor(bike$season)
bike$holiday <- factor(bike$holiday)
bike$workingday <- factor(bike$workingday)
bike$weather <- factor(bike$weather)

str(bike)

## 'data.frame':    10886 obs. of  13 variables:
##  $ datetime  : POSIXct, format: "2011-01-01 00:00:00" "2011-01-01
01:00:00" ...
##  $ season    : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
##  $ holiday   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ workingday: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ weather   : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 2 1 1 1 1 ...
##  $ temp      : num  9.84 9.02 9.02 9.84 9.84 ...
##  $ atemp     : num  14.4 13.6 13.6 14.4 14.4 ...
##  $ humidity  : int  81 80 80 75 75 75 80 86 75 76 ...
##  $ windspeed : num  0 0 0 0 0 ...
##  $ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
##  $ registered: int  13 32 27 10 1 1 0 2 7 6 ...
##  $ count     : int  16 40 32 13 1 1 2 3 8 14 ...
##  $ hour      : chr  "00" "01" "02" "03" ...

#### all set so now lets go for the modeling

sample <- sample.split(bike$count,SplitRatio = 0.7)

train <- subset(bike, sample == TRUE)
test <- subset(bike,sample == FALSE)

model <- lm(count ~ .,train)

summary(model)

##
## Call:
## lm(formula = count ~ ., data = train)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.802e-11 -1.940e-14  7.000e-16  2.150e-14  1.239e-11
##
## Coefficients:
##                Estimate Std. Error    t value Pr(>|t|)
## (Intercept) -1.052e-11  3.413e-13 -3.082e+01  < 2e-16 ***
## datetime     7.568e-21  2.582e-22  2.931e+01  < 2e-16 ***
## season2     -2.560e-13  1.385e-14 -1.849e+01  < 2e-16 ***
## season3     -1.767e-13  1.778e-14 -9.938e+00  < 2e-16 ***
```

```
## season4     -4.228e-14  1.248e-14 -3.389e+00 0.000704 ***
## holiday1    -7.796e-14  2.339e-14 -3.333e+00 0.000863 ***
## workingday1  2.098e-13  9.869e-15  2.126e+01  < 2e-16 ***
## weather2    -6.522e-14  9.162e-15 -7.118e+00 1.19e-12 ***
## weather3    -3.743e-13  1.569e-14 -2.385e+01  < 2e-16 ***
## weather4    -3.066e-13  3.243e-13 -9.450e-01 0.344496
## temp         2.231e-14  3.094e-15  7.211e+00 6.09e-13 ***
## atemp       -9.837e-15  2.712e-15 -3.627e+00 0.000289 ***
## humidity     8.560e-15  2.640e-16  3.243e+01  < 2e-16 ***
## windspeed    2.708e-15  5.024e-16  5.390e+00 7.26e-08 ***
## casual       1.000e+00  1.229e-16  8.134e+15  < 2e-16 ***
## registered   1.000e+00  4.721e-17  2.118e+16  < 2e-16 ***
## hour01      -1.098e-14  2.542e-14 -4.320e-01 0.665792
## hour02       1.591e-14  2.580e-14  6.170e-01 0.537519
## hour03       2.026e-14  2.587e-14  7.830e-01 0.433532
## hour04       3.624e-14  2.583e-14  1.403e+00 0.160630
## hour05       2.073e-14  2.524e-14  8.210e-01 0.411589
## hour06       1.564e-14  2.570e-14  6.090e-01 0.542823
## hour07       2.403e-14  2.689e-14  8.930e-01 0.371689
## hour08       4.514e-14  2.902e-14  1.556e+00 0.119860
## hour09       3.261e-14  2.642e-14  1.234e+00 0.217120
## hour10       2.953e-14  2.584e-14  1.143e+00 0.253070
## hour11       3.350e-14  2.624e-14  1.277e+00 0.201738
## hour12       2.598e-14  2.694e-14  9.640e-01 0.334880
## hour13       2.213e-14  2.699e-14  8.200e-01 0.412463
## hour14       3.433e-14  2.736e-14  1.255e+00 0.209591
## hour15       4.118e-14  2.677e-14  1.538e+00 0.124035
## hour16       3.787e-14  2.779e-14  1.363e+00 0.172993
## hour17       5.219e-14  2.987e-14  1.747e+00 0.080616 .
## hour18       8.852e-14  2.929e-14  3.022e+00 0.002518 **
## hour19       3.556e-14  2.720e-14  1.307e+00 0.191171
## hour20       3.244e-14  2.609e-14  1.243e+00 0.213804
## hour21       3.501e-14  2.586e-14  1.354e+00 0.175886
## hour22       2.989e-14  2.578e-14  1.160e+00 0.246267
## hour23       1.364e-14  2.568e-14  5.310e-01 0.595296
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.234e-13 on 7598 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 6.438e+31 on 38 and 7598 DF,  p-value: < 2.2e-16

#### prediction

predict.model <- predict(model,test[50:60,!names(test) %in% 'count'])

table(predict.model,test$count[50:60])

##
## predict.model       1 5 9 10 12 25 39 61 83 98
```

```
##   0.999999999999807 1 0 0  0  0  0  0  0  0  0
##   0.999999999999811 1 0 0  0  0  0  0  0  0  0
##   4.99999999999995  0 1 0  0  0  0  0  0  0  0
##   8.99999999999993  0 0 1  0  0  0  0  0  0  0
##   9.99999999999982  0 0 0  1  0  0  0  0  0  0
##   11.9999999999998  0 0 0  0  1  0  0  0  0  0
##   24.9999999999997  0 0 0  0  0  1  0  0  0  0
##   39                0 0 0  0  0  0  1  0  0  0
##   61                0 0 0  0  0  0  0  1  0  0
##   82.9999999999997  0 0 0  0  0  0  0  0  1  0
##   97.9999999999998  0 0 0  0  0  0  0  0  0  1
```

```
#### but to really check the accuracy we can't do on such a big continuous
data
#### because our model predicts the count, it will be a confusing to draw the
confusion matrix
#### so to solve that and to see the accuracy of our project we will do
something sneaky
#### we will select random row from the test and then compare to the
prediction




#### using just 1 row to really see, excited

single.row <- test[7,!names(test) %in% 'count']

View(test)
test[7,]

##                datetime season holiday workingday weather  temp atemp
humidity
## 20 2011-01-01 19:00:00      1       0          0       3 17.22 21.21
88
##    windspeed casual registered count hour
## 20   16.9979      6         31    37   19

test$count[7]

## [1] 37

#### the count on the test data shows 9 so lets see now

View(single.row)
single.row

##                datetime season holiday workingday weather  temp atemp
humidity
## 20 2011-01-01 19:00:00      1       0          0       3 17.22 21.21
88
```

```
##    windspeed casual registered hour
## 20   16.9979      6         31   19

predict.model.2 <- predict(model,single.row)
#### testing the prediction and model

print(predict.model.2)

## 20
## 37

table(predict.model.2,test$count[7])

##
## predict.model.2    37
##   36.9999999999999  1

#### seems pretty accurate

#### lets try one more time

single.row <- test[264,!names(test) %in% 'count']


predict.model.3 <- predict(model,single.row)

print(predict.model.3)

## 866
## 119

test$count[264]

## [1] 119

table(predict.model.3,test$count[264])

##
## predict.model.3 119
##             119   1

#### that sums it up, in statistics we never say our model is perfect or
solidly accurate but here i can say its a good model
```