code_4294489

Lucas Willocq

4/14/2022

Packages needed to work on the project.

```
library(gbm)
library(glmnet)
library(plyr)
library(lubridate)
library(leaps)
```

For starters, necessary to load the train file and examine it.

```
df <- read.csv(file = 'train.csv', header=T, stringsAsFactors = T)
attach(df)
summary(df)
```

```
##      Count              Date            Hour         Temperature
##  Min.   :   0.0   1/10/2018:  24   Min.   : 0.00   Min.   :-17.80
##  1st Qu.: 189.0   1/11/2018:  24   1st Qu.: 5.75   1st Qu.:  3.00
##  Median : 492.0   1/12/2017:  24   Median :11.50   Median : 13.60
##  Mean   : 702.9   1/3/2018 :  24   Mean   :11.50   Mean   : 12.59
##  3rd Qu.:1062.0   1/4/2018 :  24   3rd Qu.:17.25   3rd Qu.: 22.30
##  Max.   :3556.0   1/5/2018 :  24   Max.   :23.00   Max.   : 39.40
##                   (Other)  :6408
##     Humidity          Wind          Visibility          Dew
##  Min.   : 0.00   Min.   :0.000   Min.   :  27.0   Min.   :-30.600
##  1st Qu.:42.00   1st Qu.:0.900   1st Qu.: 970.8   1st Qu.: -5.425
##  Median :57.00   Median :1.550   Median :1708.0   Median :  5.200
##  Mean   :58.15   Mean   :1.764   Mean   :1448.8   Mean   :  3.799
##  3rd Qu.:74.00   3rd Qu.:2.400   3rd Qu.:2000.0   3rd Qu.: 14.500
##  Max.   :98.00   Max.   :7.400   Max.   :2000.0   Max.   : 26.800
##
##      Solar           Rainfall          Snowfall         Seasons
##  Min.   :0.0000   Min.   : 0.0000   Min.   :0.00000   Autumn:1704
##  1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.00000   Spring:1584
##  Median :0.0100   Median : 0.0000   Median :0.00000   Summer:1584
##  Mean   :0.5727   Mean   : 0.1572   Mean   :0.08365   Winter:1680
##  3rd Qu.:0.9400   3rd Qu.: 0.0000   3rd Qu.:0.00000
##  Max.   :3.5200   Max.   :35.0000   Max.   :8.80000
##
##        Holiday      Functioning        ID
##  Holiday   : 312   No : 247     Min.   :100187
##  No Holiday:6240   Yes:6305     1st Qu.:328166
```

```
##                                Median :547810
##                                Mean   :548331
##                                3rd Qu.:771858
##                                Max.   :999894
##
```

Get rid of unnecessary variables "Functioning" and "ID"

```
df <- subset(df, select = -Functioning)
df <- subset(df, select = -ID)
```

Next I converted all dates to numbers of days out of the year.

```
df$Date <- yday(as.Date(df$Date, format = "%d/%m/%Y"))
```

I then broke up the data into a test and training set.

```
line <- sample(1:nrow(df), nrow(df)*0.8)
df.train <- df[line, ]
df.test <- df[-line, ]
```

Just to get a better idea of what variables are important, I ran the full model with all these variables and then got the basic test MSE.

```
lm.full <- lm(Count ~., data=df.train)
summary(lm.full)
```

```
##
## Call:
## lm(formula = Count ~ ., data = df.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1426.03  -279.14   -43.87   233.46  2283.94
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        539.47759  127.54253   4.230 2.38e-05 ***
## Date                 0.14185    0.08293   1.711  0.08723 .
## Hour                27.59180    1.04785  26.332  < 2e-16 ***
## Temperature         23.68131    4.84594   4.887 1.06e-06 ***
## Humidity            -9.34172    1.35113  -6.914 5.28e-12 ***
## Wind                10.11376    6.90127   1.465  0.14285
## Visibility           0.01709    0.01418   1.205  0.22829
## Dew                  2.42415    5.04010   0.481  0.63056
## Solar              -78.97392   10.64875  -7.416 1.40e-13 ***
## Rainfall           -59.24392    5.63344 -10.516  < 2e-16 ***
## Snowfall            41.83427   14.41489   2.902  0.00372 **
## SeasonsSpring        0.83518   24.38883   0.034  0.97268
## SeasonsSummer      -16.01410   24.88016  -0.644  0.51983
## SeasonsWinter     -255.30382   29.46652  -8.664  < 2e-16 ***
## HolidayNo Holiday  145.88767   31.42221   4.643 3.52e-06 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 471.5 on 5226 degrees of freedom
## Multiple R-squared:  0.4916, Adjusted R-squared:  0.4903
## F-statistic:   361 on 14 and 5226 DF,  p-value: < 2.2e-16
```

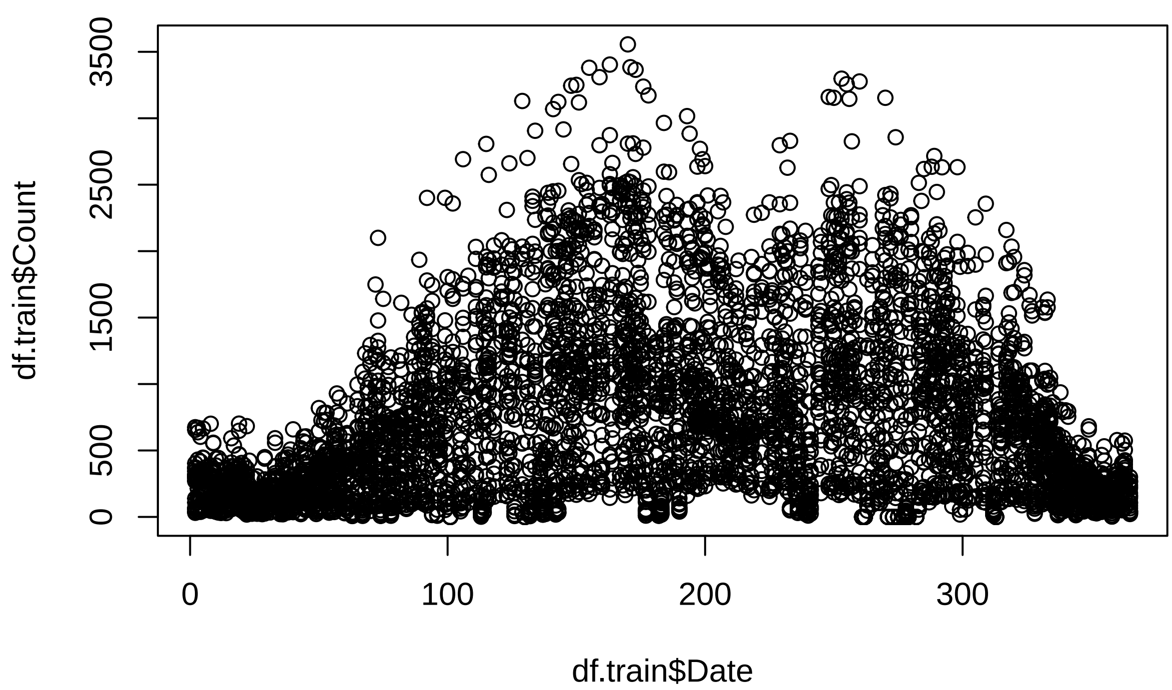
```r
predict.lm.full <- predict.lm(lm.full, data = df.test)
full.MSE <- mean((df.test$Count-predict.lm.full)^2)
```

```
## Warning in df.test$Count - predict.lm.full: longer object length is not a
## multiple of shorter object length
```

```r
full.MSE
```

```
## [1] 581508.5
```

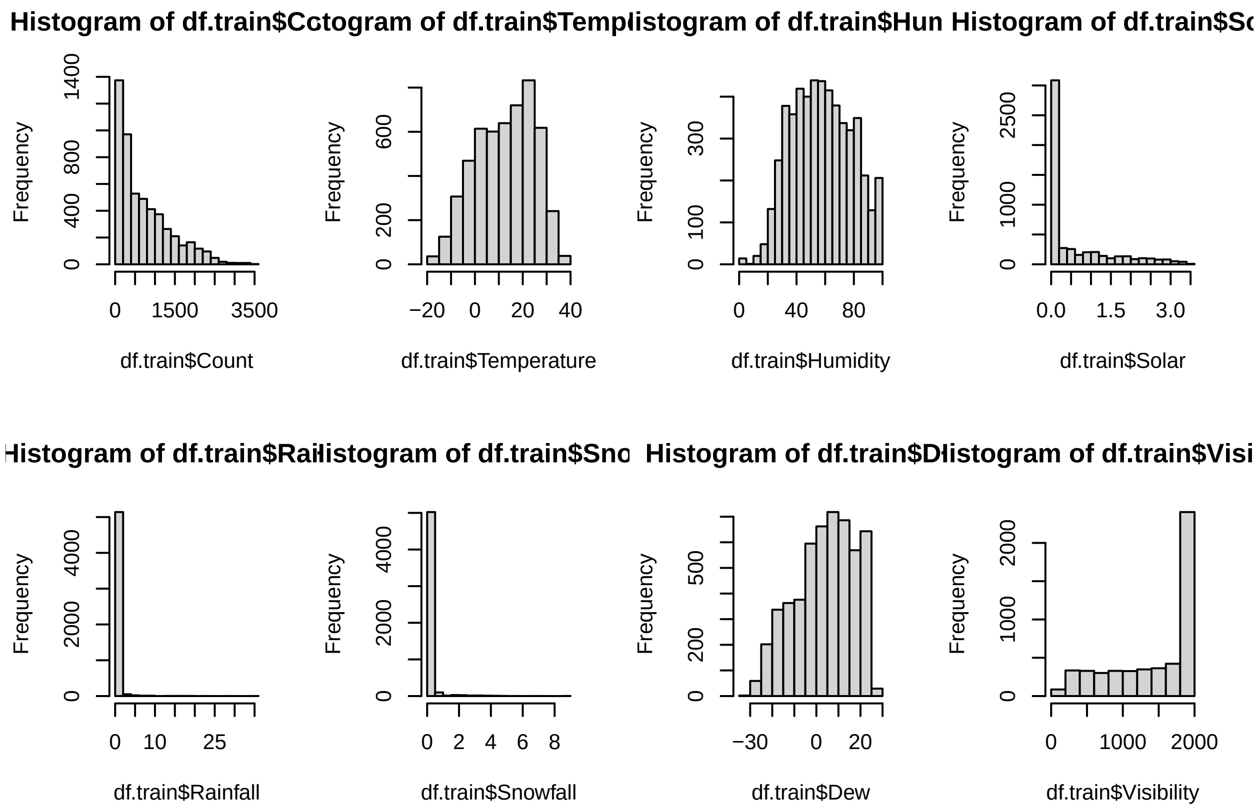
```r
plot(df.train$Date,df.train$Count)
```



From examining this distribution of bike rentals per day of the year, it follows a rather normal pattern. It is also apparent that the full linear model is not going to be a good fit, with a high test MSE. The majority of bike rentals happen in the middle of the year (warm months), with less on the ends when it is cold.

I examined some of the individual predictors to see if their frequencies lined up with Count, to compare if models are logical.

```r
par(mfrow = c(2,4))
hist(df.train$Count)
hist(df.train$Temperature)
hist(df.train$Humidity)
hist(df.train$Solar)
hist(df.train$Rainfall)
hist(df.train$Snowfall)
hist(df.train$Dew)
hist(df.train$Visibility)
```



I then decided to go through each individual predictor and examine which ones produced the lowest test MSE on their own.

```r
num_predictors <- ncol(df.train)-1

individual.MSEs <- rep()
names <- colnames(df.train[-1])

for(i in names){
  model <- lm(Count~df.train[[i]], data = df.train)
  predict.lm <- predict.lm(model, data = df.test)
  individual.MSEs[i]<- mean((df.test$Count-predict.lm)^2)
}
individual.MSEs
```

```
##         Date         Hour Temperature     Humidity         Wind   Visibility
```

```
##    376570.5    440941.5    493072.1    391540.6    375654.8    390181.1
##         Dew       Solar    Rainfall    Snowfall     Seasons     Holiday
##    427148.7    403817.5    377240.3    378915.4    457294.2    371122.7
```

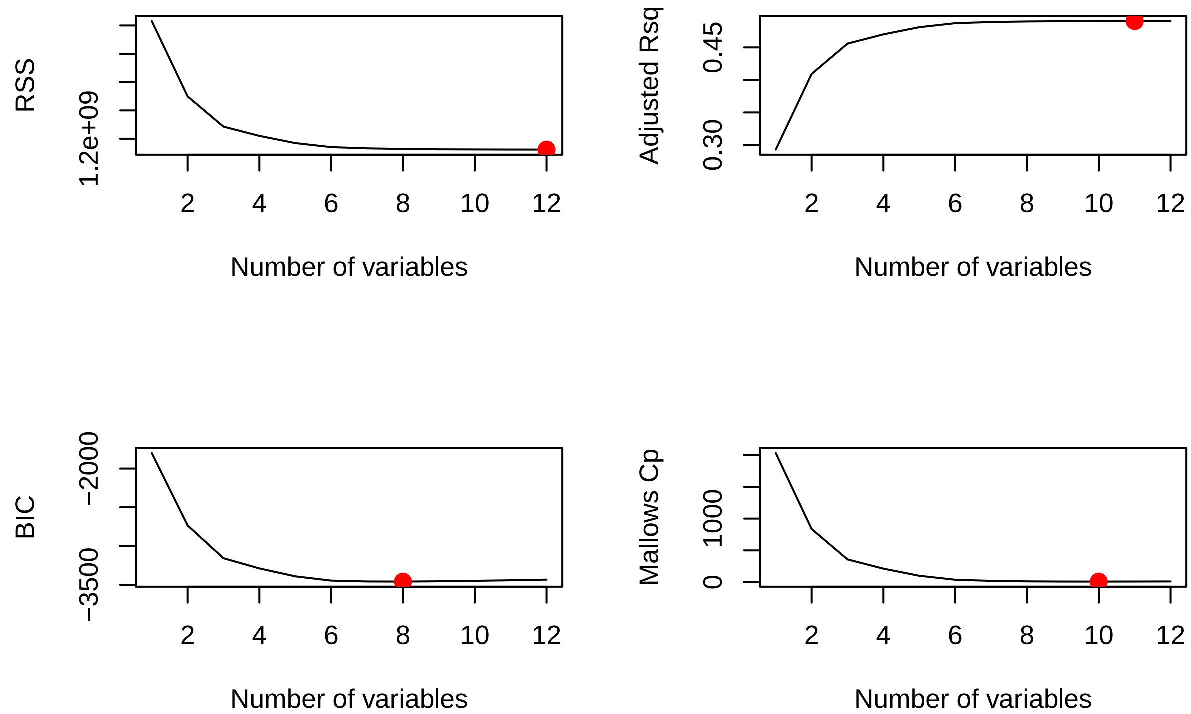
```
which.min(individual.MSEs) - 433744
```

```
## Holiday
## -433732
```

When testing individual predictors, it appears that the model with only Holiday produces the lowest MSE.

With these individual predictor models also not being a great fit, I decided to run some model selection criterias and examine graphs of the RSS, Adjusted R-squared, BIC and Cp levels for different model sizes.

```
bestsubset <- regsubsets(Count~., df.train, nvmax = 12)
bestsubset.summary <- summary(bestsubset)

par(mfrow = c(2,2))
plot(bestsubset.summary$rss, xlab = "Number of variables", ylab= "RSS", type = "l")
points(which.min(bestsubset.summary$rss), bestsubset.summary$rss[which.min(bestsubset.summary$rss)], co


plot(bestsubset.summary$adjr2, xlab = "Number of variables", ylab= "Adjusted Rsq", type = "l")
points(which.max(bestsubset.summary$adjr2), bestsubset.summary$adjr2[which.max(bestsubset.summary$adjr2

plot(bestsubset.summary$bic, xlab = "Number of variables", ylab= "BIC", type = "l")
points(which.min(bestsubset.summary$bic), bestsubset.summary$bic[which.min(bestsubset.summary$bic)], co

plot(bestsubset.summary$cp, xlab = "Number of variables", ylab= "Mallows Cp", type = "l")
points(which.min(bestsubset.summary$cp), bestsubset.summary$cp[which.min(bestsubset.summary$cp)], col =
```



As expected, the RSS is smallest with the most amount of predictors in the model. This will probably not be our ideal model however, as it will prove to be overly complicated. The BIC is lowest at 7 variables, while both the Cp and Adjusted Rsq are smallest at a 10 variable model. I then decided to look even further into these models.

```
coef(bestsubset, 7)
```

```
##       (Intercept)              Hour       Temperature          Humidity
##        574.460427         28.071620         25.276699         -8.889379
##             Solar          Rainfall     SeasonsWinter HolidayNo Holiday
##        -77.197146        -59.956166       -261.874771        137.171641
```

```
coef(bestsubset, 10)
```

```
##       (Intercept)              Date              Hour       Temperature
##       535.5266060         0.1548634        27.5615402        25.5519330
##          Humidity              Wind             Solar          Rainfall
##        -9.1192444        10.6621306       -82.3256181       -59.7914880
##          Snowfall     SeasonsWinter HolidayNo Holiday
##        40.3025374      -262.8161133       144.5934163
```

```
seven_var_fit <- lm(Count ~ Hour + Temperature + Humidity + Solar + Rainfall + Seasons + Holiday, data
seven_var.lm <- predict.lm(seven_var_fit, data = df.test)
seven_var.MSE<- mean((df.test$Count-seven_var.lm)^2)
seven_var.MSE
```

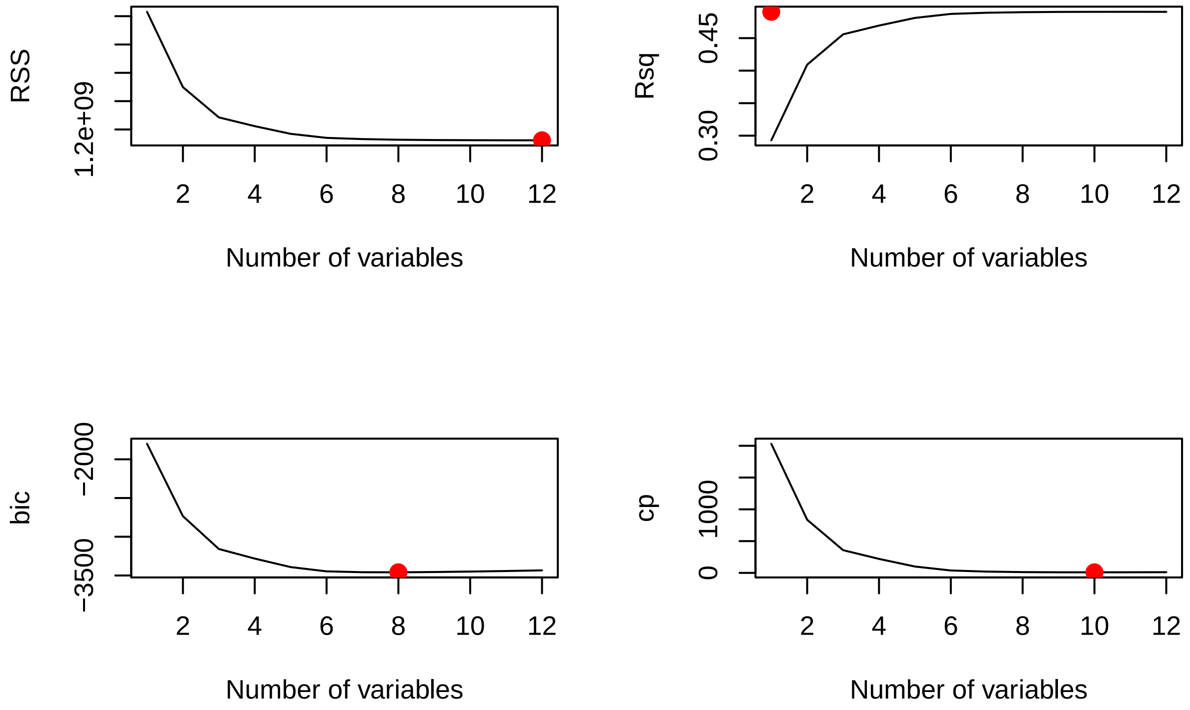
```
## [1] 581138
```

```
ten_var_fit <- lm(Count ~ Date+Hour+Temperature+Humidity+Wind+Solar+Rainfall+Snowfall+Seasons+Holiday, d
ten_var.lm <- predict.lm(ten_var_fit, data = df.test)
ten_var.MSE<- mean((df.test$Count-ten_var.lm)^2)
ten_var.MSE
```

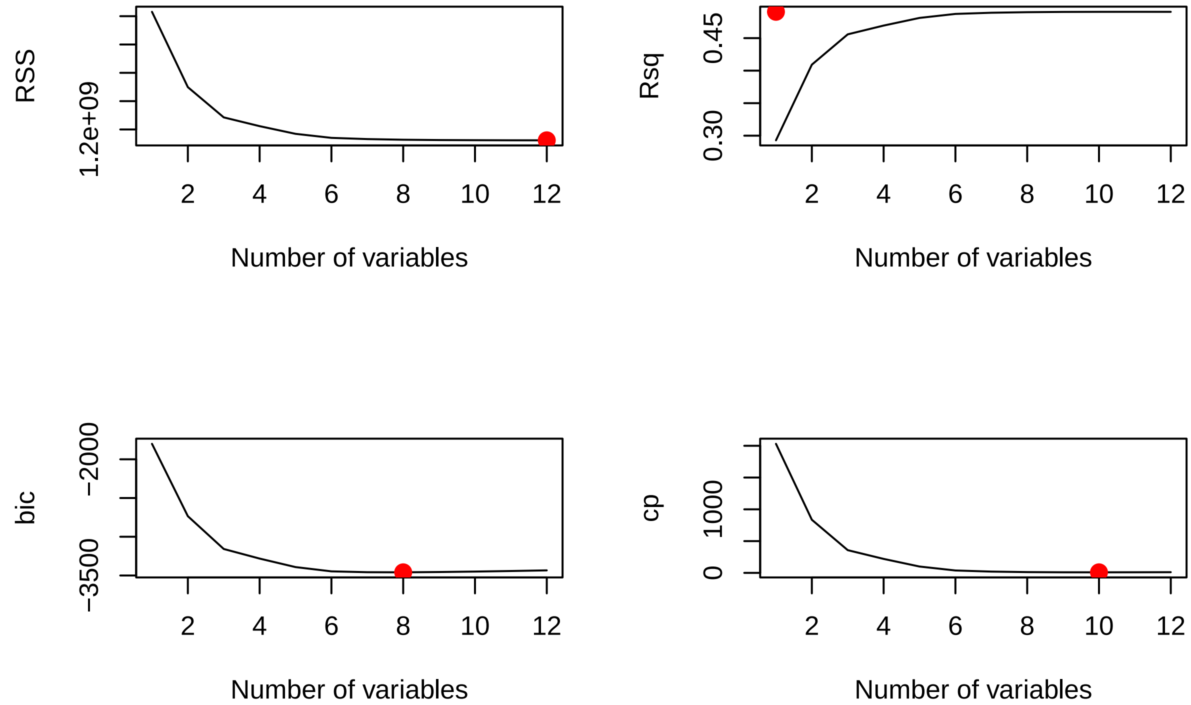
```
## [1] 581498.6
```

Surprisingly, these do not have very low MSEs, actually higher than previous models. I decided to then do a step-wise model selection to see if these results changed. I did a forward and backward selection and calculated the same stats from the different model sizes.

```
fit.forward <- regsubsets(Count ~., data = df.train, nvmax = 12, method = "forward")
par(mfrow = c(2,2))

forward.summary <- summary(fit.forward)

plot(forward.summary$rss, xlab = "Number of variables", ylab= "RSS", type = "l")
points(which.min(forward.summary$rss), forward.summary$rss[which.min(forward.summary$rss)], col = "red"


plot(forward.summary$adjr2, xlab = "Number of variables", ylab= "Rsq", type = "l")
points(which.min(forward.summary$adjr2), forward.summary$adjr2[which.max(forward.summary$adjr2)], col =

plot(forward.summary$bic, xlab = "Number of variables", ylab= "bic", type = "l")
points(which.min(forward.summary$bic), forward.summary$bic[which.min(forward.summary$bic)], col = "red"

plot(forward.summary$cp, xlab = "Number of variables", ylab= "cp", type = "l")
points(which.min(forward.summary$cp), forward.summary$cp[which.min(forward.summary$cp)], col = "red", c
```




```
fit.backward <- regsubsets(Count ~., data = df.train, nvmax = 12, method = "backward")

backward.summary <- summary(fit.backward)

plot(backward.summary$rss, xlab = "Number of variables", ylab= "RSS", type = "l")
points(which.min(backward.summary$rss), backward.summary$rss[which.min(backward.summary$rss)], col = "re


plot(backward.summary$adjr2, xlab = "Number of variables", ylab= "Rsq", type = "l")
points(which.min(backward.summary$adjr2), backward.summary$adjr2[which.max(backward.summary$adjr2)], col

plot(backward.summary$bic, xlab = "Number of variables", ylab= "bic", type = "l")
points(which.min(backward.summary$bic), backward.summary$bic[which.min(backward.summary$bic)], col = "re

plot(backward.summary$cp, xlab = "Number of variables", ylab= "cp", type = "l")
points(which.min(backward.summary$cp), backward.summary$cp[which.min(backward.summary$cp)], col = "red"
```




```
forward.summary
```

```
## Subset selection object
## Call: regsubsets.formula(Count ~ ., data = df.train, nvmax = 12, method = "forward")
## 14 Variables  (and intercept)
##                    Forced in Forced out
## Date                   FALSE      FALSE
## Hour                   FALSE      FALSE
## Temperature            FALSE      FALSE
## Humidity               FALSE      FALSE
## Wind                   FALSE      FALSE
## Visibility             FALSE      FALSE
## Dew                    FALSE      FALSE
## Solar                  FALSE      FALSE
## Rainfall               FALSE      FALSE
## Snowfall               FALSE      FALSE
## SeasonsSpring          FALSE      FALSE
## SeasonsSummer          FALSE      FALSE
## SeasonsWinter          FALSE      FALSE
## HolidayNo Holiday      FALSE      FALSE
## 1 subsets of each size up to 12
## Selection Algorithm: forward
##             Date Hour Temperature Humidity Wind Visibility Dew Solar Rainfall
## 1  ( 1 )  " "  " "  "*"         " "      " "  " "        " " " "   " "
## 2  ( 1 )  " "  "*"  "*"         " "      " "  " "        " " " "   " "
## 3  ( 1 )  " "  "*"  "*"         "*"      " "  " "        " " " "   " "
```

```
## 4  ( 1 )  " "  "*"  "*"         "*"      " "  " "        " " " "   " "
## 5  ( 1 )  " "  "*"  "*"         "*"      " "  " "        " " " "   "*"
## 6  ( 1 )  " "  "*"  "*"         "*"      " "  " "        " " "*"   "*"
## 7  ( 1 )  " "  "*"  "*"         "*"      " "  " "        " " "*"   "*"
## 8  ( 1 )  " "  "*"  "*"         "*"      " "  " "        " " "*"   "*"
## 9  ( 1 )  "*"  "*"  "*"         "*"      " "  " "        " " "*"   "*"
## 10  ( 1 ) "*"  "*"  "*"         "*"      "*"  " "        " " "*"   "*"
## 11  ( 1 ) "*"  "*"  "*"         "*"      "*"  "*"        " " "*"   "*"
## 12  ( 1 ) "*"  "*"  "*"         "*"      "*"  "*"        " " "*"   "*"
##           Snowfall SeasonsSpring SeasonsSummer SeasonsWinter HolidayNo Holiday
## 1  ( 1 )  " "      " "           " "           " "           " "
## 2  ( 1 )  " "      " "           " "           " "           " "
## 3  ( 1 )  " "      " "           " "           " "           " "
## 4  ( 1 )  " "      " "           " "           "*"           " "
## 5  ( 1 )  " "      " "           " "           "*"           " "
## 6  ( 1 )  " "      " "           " "           "*"           " "
## 7  ( 1 )  " "      " "           " "           "*"           "*"
## 8  ( 1 )  "*"      " "           " "           "*"           "*"
## 9  ( 1 )  "*"      " "           " "           "*"           "*"
## 10  ( 1 ) "*"      " "           " "           "*"           "*"
## 11  ( 1 ) "*"      " "           " "           "*"           "*"
## 12  ( 1 ) "*"      " "           "*"           "*"           "*"
```

```
backward.summary
```

```
## Subset selection object
## Call: regsubsets.formula(Count ~ ., data = df.train, nvmax = 12, method = "backward")
## 14 Variables  (and intercept)
##                    Forced in Forced out
## Date                   FALSE      FALSE
## Hour                   FALSE      FALSE
## Temperature            FALSE      FALSE
## Humidity               FALSE      FALSE
## Wind                   FALSE      FALSE
## Visibility             FALSE      FALSE
## Dew                    FALSE      FALSE
## Solar                  FALSE      FALSE
## Rainfall               FALSE      FALSE
## Snowfall               FALSE      FALSE
## SeasonsSpring          FALSE      FALSE
## SeasonsSummer          FALSE      FALSE
## SeasonsWinter          FALSE      FALSE
## HolidayNo Holiday      FALSE      FALSE
## 1 subsets of each size up to 12
## Selection Algorithm: backward
##           Date Hour Temperature Humidity Wind Visibility Dew Solar Rainfall
## 1  ( 1 )  " "  " "  "*"         " "      " "  " "        " " " "   " "
## 2  ( 1 )  " "  "*"  "*"         " "      " "  " "        " " " "   " "
## 3  ( 1 )  " "  "*"  "*"         "*"      " "  " "        " " " "   " "
## 4  ( 1 )  " "  "*"  "*"         "*"      " "  " "        " " " "   " "
## 5  ( 1 )  " "  "*"  "*"         "*"      " "  " "        " " " "   "*"
## 6  ( 1 )  " "  "*"  "*"         "*"      " "  " "        " " "*"   "*"
## 7  ( 1 )  " "  "*"  "*"         "*"      " "  " "        " " "*"   "*"
## 8  ( 1 )  " "  "*"  "*"         "*"      " "  " "        " " "*"   "*"
```

```
## 9  ( 1 )  "*"  "*"  "*"         "*"      " "  " "         " " "*"   "*"
## 10  ( 1 ) "*"  "*"  "*"         "*"      "*"  " "         " " "*"   "*"
## 11  ( 1 ) "*"  "*"  "*"         "*"      "*"  "*"         " " "*"   "*"
## 12  ( 1 ) "*"  "*"  "*"         "*"      "*"  "*"         " " "*"   "*"
##           Snowfall SeasonsSpring SeasonsSummer SeasonsWinter HolidayNo Holiday
## 1  ( 1 )  " "      " "           " "           " "           " "
## 2  ( 1 )  " "      " "           " "           " "           " "
## 3  ( 1 )  " "      " "           " "           " "           " "
## 4  ( 1 )  " "      " "           " "           "*"           " "
## 5  ( 1 )  " "      " "           " "           "*"           " "
## 6  ( 1 )  " "      " "           " "           "*"           " "
## 7  ( 1 )  " "      " "           " "           "*"           "*"
## 8  ( 1 )  "*"      " "           " "           "*"           "*"
## 9  ( 1 )  "*"      " "           " "           "*"           "*"
## 10  ( 1 ) "*"      " "           " "           "*"           "*"
## 11  ( 1 ) "*"      " "           " "           "*"           "*"
## 12  ( 1 ) "*"      " "           "*"           "*"           "*"
```

Backward and forward selection produced the same models as the Subset selection, so I decided to look at the models they all produced and examine their MSEs individually to see which model had the lowest test error.

```
Mses <- rep()


one_var_model <- lm(Count ~ Temperature, data = df.train)

one_var.lm <- predict.lm(one_var_model, data = df.test)
Mses[1]<- mean((df.test$Count-one_var.lm)^2)

two_var_model <- lm(Count ~ Temperature + Hour, data = df.train)
two_var.lm <- predict.lm(two_var_model, data = df.test)
Mses[2]<- mean((df.test$Count-two_var.lm)^2)

three_var_model <- lm(Count ~ Temperature + Hour + Humidity, data = df.train)

three_var.lm <- predict.lm(three_var_model, data = df.test)
Mses[3]<- mean((df.test$Count-three_var.lm)^2)

four_var_model <- lm(Count ~ Temperature + Hour + Humidity + Seasons, data = df.train)

four_var.lm <- predict.lm(four_var_model, data = df.test)
Mses[4]<- mean((df.test$Count-four_var.lm)^2)

five_var_model <- lm(Count ~ Temperature + Hour + Humidity + Seasons + Rainfall, data = df.train)

five_var.lm <- predict.lm(five_var_model, data = df.test)
Mses[5]<- mean((df.test$Count-five_var.lm)^2)

six_var_model <- lm(Count ~ Temperature + Hour + Humidity + Seasons + Rainfall + Solar, data = df.train

six_var.lm <- predict.lm(six_var_model, data = df.test)
Mses[6]<- mean((df.test$Count-six_var.lm)^2)
```

```r
seven_var_model <- lm(Count ~ Temperature + Hour + Humidity + Seasons + Rainfall + Solar + Holiday, data

seven_var.lm <- predict.lm(seven_var_model, data = df.test)
Mses[7]<- mean((df.test$Count-seven_var.lm)^2)

eight_var_model <- lm(Count ~ Temperature + Hour + Humidity + Seasons + Rainfall + Solar + Holiday + Sno

eight_var.lm <- predict.lm(eight_var_model, data = df.test)
Mses[8]<- mean((df.test$Count-eight_var.lm)^2)

nine_var_model <- lm(Count ~ Temperature + Hour + Humidity + Seasons + Rainfall + Solar + Holiday + Snow

nine_var.lm <- predict.lm(nine_var_model, data = df.test)
Mses[9]<- mean((df.test$Count-nine_var.lm)^2)

Mses
```

```
## [1] 493072.1 542880.3 566850.8 573086.9 578177.4 579867.2 581138.0 581499.7
## [9] 581395.3
```

```r
which.min(Mses)
```

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

The individual predictor (Temperature) produced the lowest test MSE, although it wasn't too far off from the rest of the models. It would probably make more sense to include more predictors. I decided to go a new route after these investigations and look into lasso and ridge regression on the entire data set.

```r
set.seed(1)

train.matrix <- model.matrix(Count~., data = df.train)
test.matrix <- model.matrix(Count~., data = df.test)
grid <- 10^seq(10,-2,length = 100)

lasso<-glmnet(train.matrix,df.train$Count,alpha=1,lambda=grid)
cv.lasso<-cv.glmnet(train.matrix,df.train$Count,alpha=1,lambda=grid)
bestlam.lasso <- cv.lasso$lambda.min
pred.lasso<-predict(lasso,s=bestlam.lasso,newx = test.matrix)
error_lasso <- mean((df.test$Count-pred.lasso)^2)
error_lasso
```

```
## [1] 202269.6
```

```r
lasso.coef <- predict(lasso, type = "coefficients", s= bestlam.lasso)
lasso.coef
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)        488.73236535
## (Intercept)          .
## Date                 0.13473232
```

```
## Hour                 27.63202705
## Temperature          25.33878136
## Humidity             -8.56745429
## Wind                  8.31375532
## Visibility            0.01782537
## Dew                   .
## Solar               -74.29373752
## Rainfall            -58.78156114
## Snowfall             36.84374427
## SeasonsSpring         .
## SeasonsSummer        -6.19735289
## SeasonsWinter      -257.39825915
## HolidayNo Holiday   138.59014668
```

```
ridge<-glmnet(train.matrix,df.train$Count,alpha=0,lambda=grid)
cv.ridge<-cv.glmnet(train.matrix,df.train$Count,alpha=0,lambda=grid)
bestlam.ridge <- cv.ridge$lambda.min
pred.ridge<-predict(ridge,s=bestlam.ridge,newx = test.matrix)
error_ridge <- mean((df.test$Count-pred.ridge)^2)
error_ridge
```

```
## [1] 202520.6
```

```
ridge.coef <- predict(ridge, type = "coefficients", s= bestlam.ridge)
ridge.coef
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                              s1
## (Intercept)        604.55844301
## (Intercept)           .
## Date                  0.15238033
## Hour                 27.63509392
## Temperature          19.58184091
## Humidity            -10.02529264
## Wind                 10.15067578
## Visibility            0.02036222
## Dew                   6.13187085
## Solar               -72.37822130
## Rainfall            -58.65333890
## Snowfall             40.18986009
## SeasonsSpring         3.51779866
## SeasonsSummer        -9.60282362
## SeasonsWinter      -257.54906593
## HolidayNo Holiday   145.50826853
```

The ridge and lasso methods produced significantly less test MSE than previous methods, with regression having slightly less. Lasso managed to shrink every variable by a ton, while ridge kept it more interpretable. While these models were probably overly complicated, I decided to try out the 7 variable model that the model selection had delegated to me with ridge regression.

```
train.matrix <- model.matrix(Count ~ Hour + Temperature + Humidity + Solar + Rainfall + Seasons + Holida
test.matrix <- model.matrix(Count ~ Hour + Temperature + Humidity + Solar + Rainfall + Seasons + Holiday
```

```r
grid <- 10^seq(10,-2,length = 100)

ridge<-glmnet(train.matrix,df.train$Count,alpha=0,lambda=grid)
cv.ridge<-cv.glmnet(train.matrix,df.train$Count,alpha=0,lambda=grid)
bestlam.ridge <- cv.ridge$lambda.min
pred.ridge<-predict(ridge,s=bestlam.ridge,newx = test.matrix)
error_ridge <- mean((df.test$Count-pred.ridge)^2)
error_ridge
```

```
## [1] 202287.6
```

```r
ridge.coef <- predict(ridge, type = "coefficients", s= bestlam.ridge)
ridge.coef
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                             s1
## (Intercept)         580.439436
## (Intercept)           .
## Hour                 28.041871
## Temperature          25.463008
## Humidity             -8.785116
## Solar               -74.849829
## Rainfall            -59.830602
## SeasonsSpring       -31.457814
## SeasonsSummer       -27.109005
## SeasonsWinter      -274.461249
## HolidayNo Holiday   139.182394
```

This model seems to be the best fit, as well as the easiest to understand with it's coefficients and predictors.

I went ahead and matched it up to the final datafile, and made sure to change some of the negative values to 0, and any decimals points rounded.

```r
final.df <- read.csv(file = 'test.csv', header=T, stringsAsFactors = T)
final.df$Date <- yday(as.Date(final.df$Date, format = "%d/%m/%Y"))
final.df['Count']=rep(0)
test.matrix <- model.matrix(Count ~ Hour + Temperature + Humidity + Solar + Rainfall + Seasons + Holiday

pred.ridge<-predict(ridge,s=bestlam.ridge,newx = test.matrix)
final.df$Count=pred.ridge
final.df <- subset(final.df, select = c(ID,Count))
final.df['Student Id'] = 4294489

n <- nrow(final.df)
for(i in 1:n){
  if(final.df$Count[i]<0){
    final.df$Count[i]=0}
}
final.df <- round(final.df,digits = 0)

write.csv(final.df,"testing_prediction_4294489.csv")
```