# Bike Sharing Demand

*Thilaksha Silva*

## Contents

## Introduction

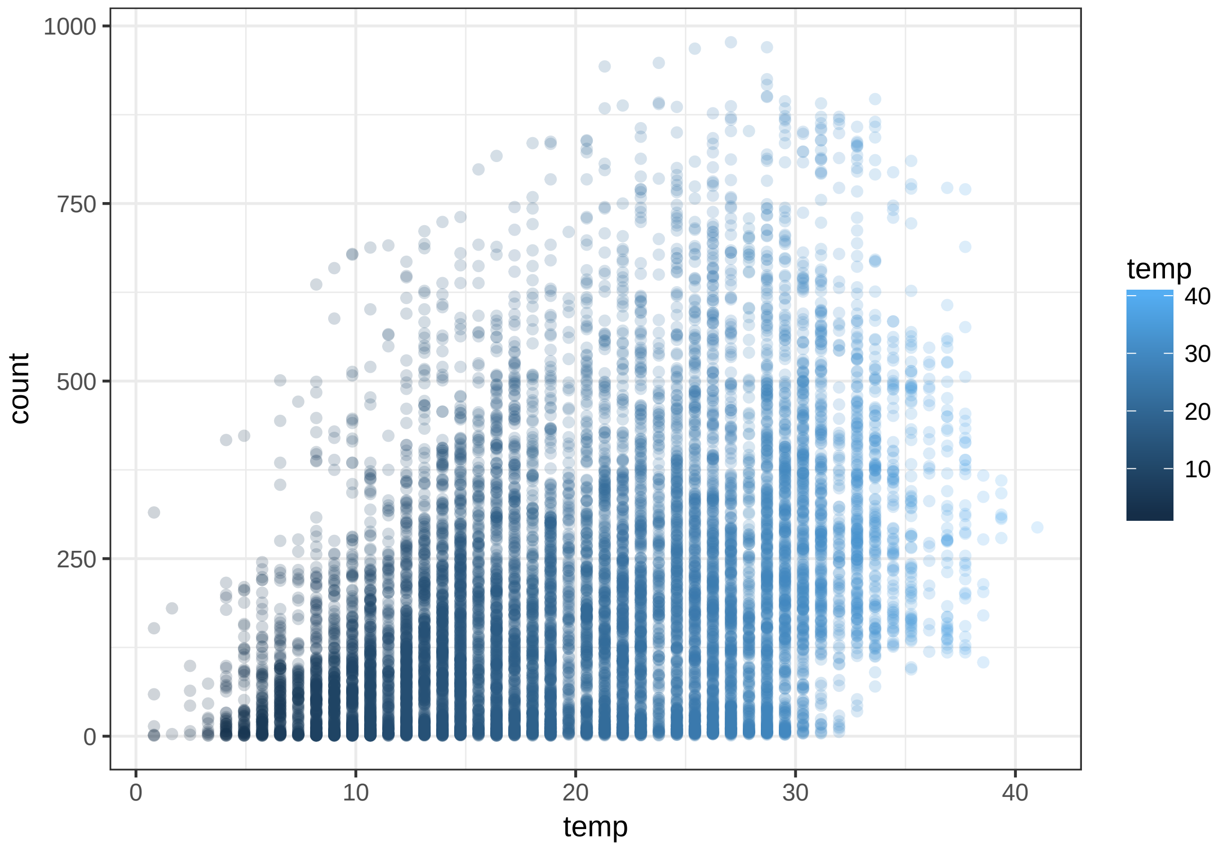
In this Kaggle challenge I employ three machine learning techniques to forecast the bike rental demand. I will guide you through my journey of predicting bike demand.

```r
# Load all the packages required for the analysis
library(ggplot2) # Visualisation
library(dplyr) # Data Wrangling
library(e1071) # Prediction: SVR
library(randomForest) # Prediction: Random Forest
```

```r
train = read.csv('train.csv')
test = read.csv('test.csv')
```

## Exploratory Data Analysis

```r
ggplot(train,aes(temp,count)) +
  geom_point(aes(color=temp),alpha=0.2) + theme_bw()
```

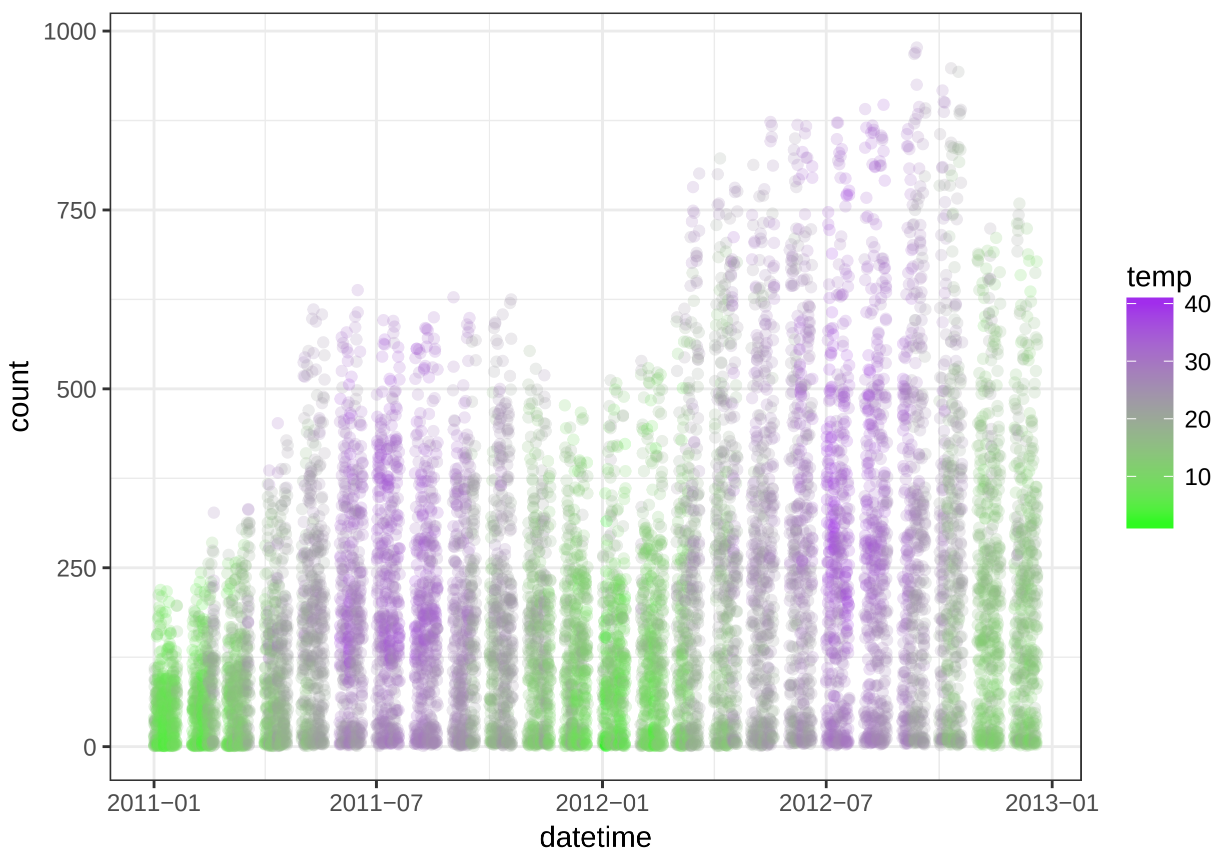


This is a scatterplot of temperature versus count with a color gradient based on temperature. The plot depicts the bike rental count increases as the temperature increases.

Let's plot a scatterplot of datetime versus count with a color gradient based on temperature. However, we need to first convert the datetime into POSIXct format.

```r
train$datetime = as.POSIXct(train$datetime, format="%Y-%m-%d %H:%M:%S")

ggplot(train,aes(datetime,count)) +
  geom_point(aes(color=temp),alpha=0.2) +
  scale_color_gradient(high='purple',low='green') +
  theme_bw()
```

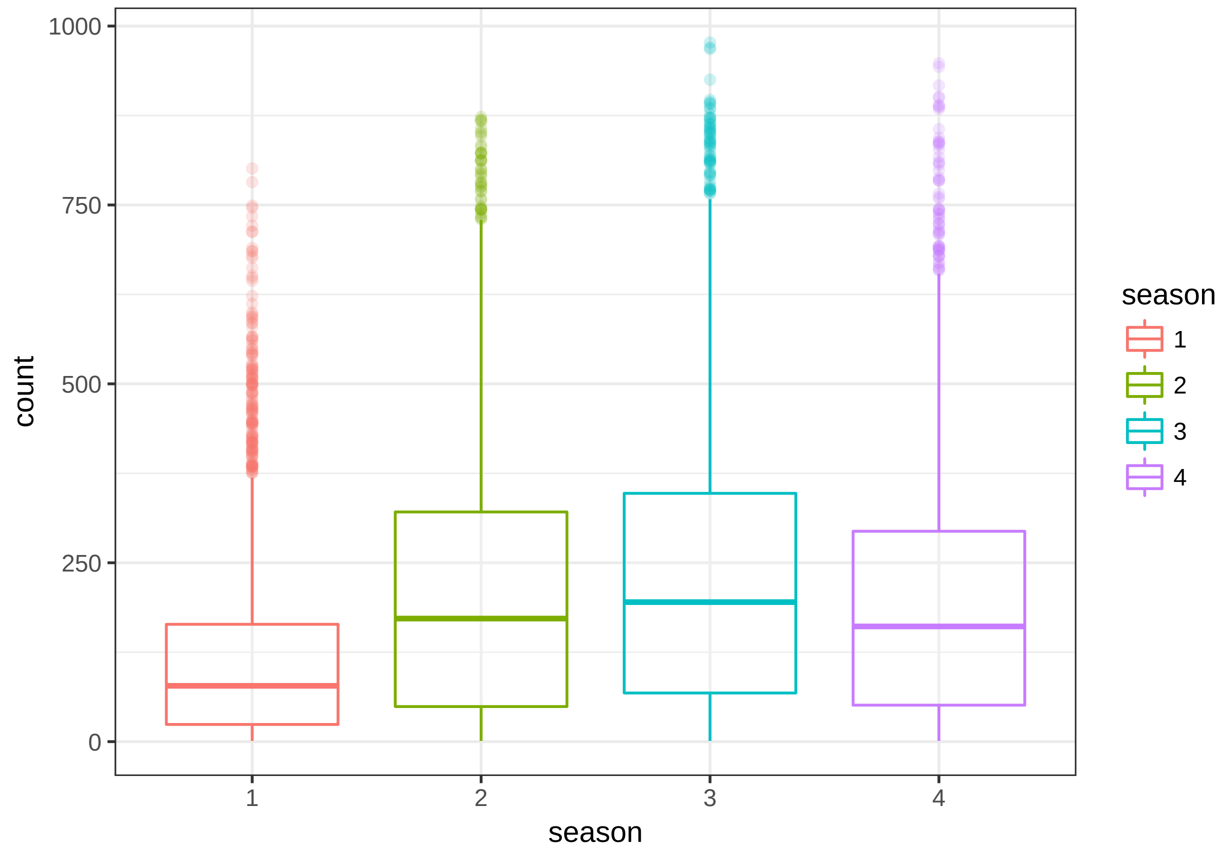


It can be clearly seen the winter and summer seasonalities of the data. As seen on the previous scatterplot, this scatterplot shows the rental counts are increasing in general.

Let's explore the season data using a box plot.

```
train$season = factor(train$season)

ggplot(train,aes(season,count)) +
  geom_boxplot(aes(color=season),alpha=0.2) +
  theme_bw()
```



Surprisingly, there are more rentals in winter (season 4) than in spring (season 1).

## Feature Engineering

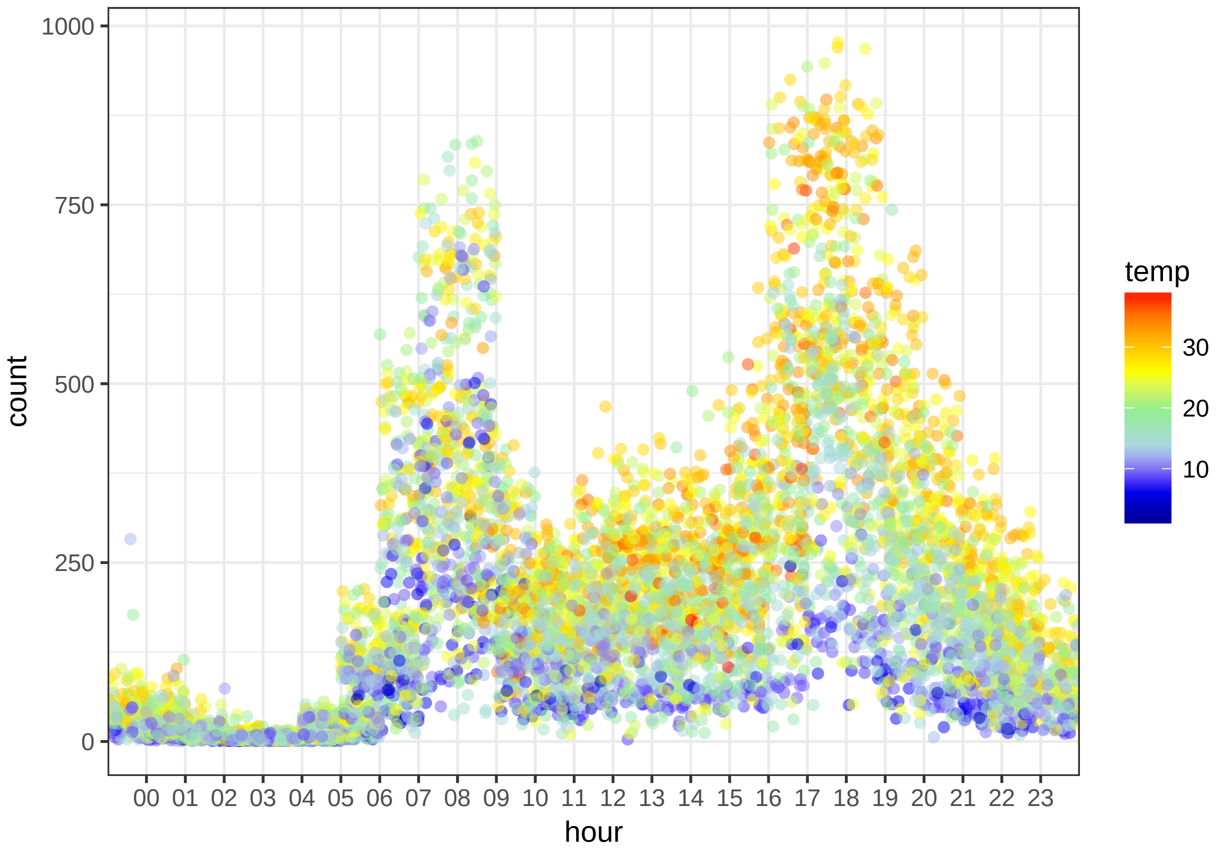
I engineer a new feature from the datetime column. Let's create an **hour** column that takes the hour from the datetime column.

```
train$hour = sapply(train$datetime, function(x) format(x,"%H"))
head(train)
```

```
##              datetime season holiday workingday weather temp  atemp
## 1 2011-01-01 00:00:00      1       0          0       1 9.84 14.395
## 2 2011-01-01 01:00:00      1       0          0       1 9.02 13.635
## 3 2011-01-01 02:00:00      1       0          0       1 9.02 13.635
## 4 2011-01-01 03:00:00      1       0          0       1 9.84 14.395
## 5 2011-01-01 04:00:00      1       0          0       1 9.84 14.395
## 6 2011-01-01 05:00:00      1       0          0       2 9.84 12.880
##   humidity windspeed casual registered count hour
## 1       81    0.0000      3         13    16   00
## 2       80    0.0000      8         32    40   01
## 3       80    0.0000      5         27    32   02
## 4       75    0.0000      3         10    13   03
## 5       75    0.0000      0          1     1   04
## 6       75    6.0032      0          1     1   05
```

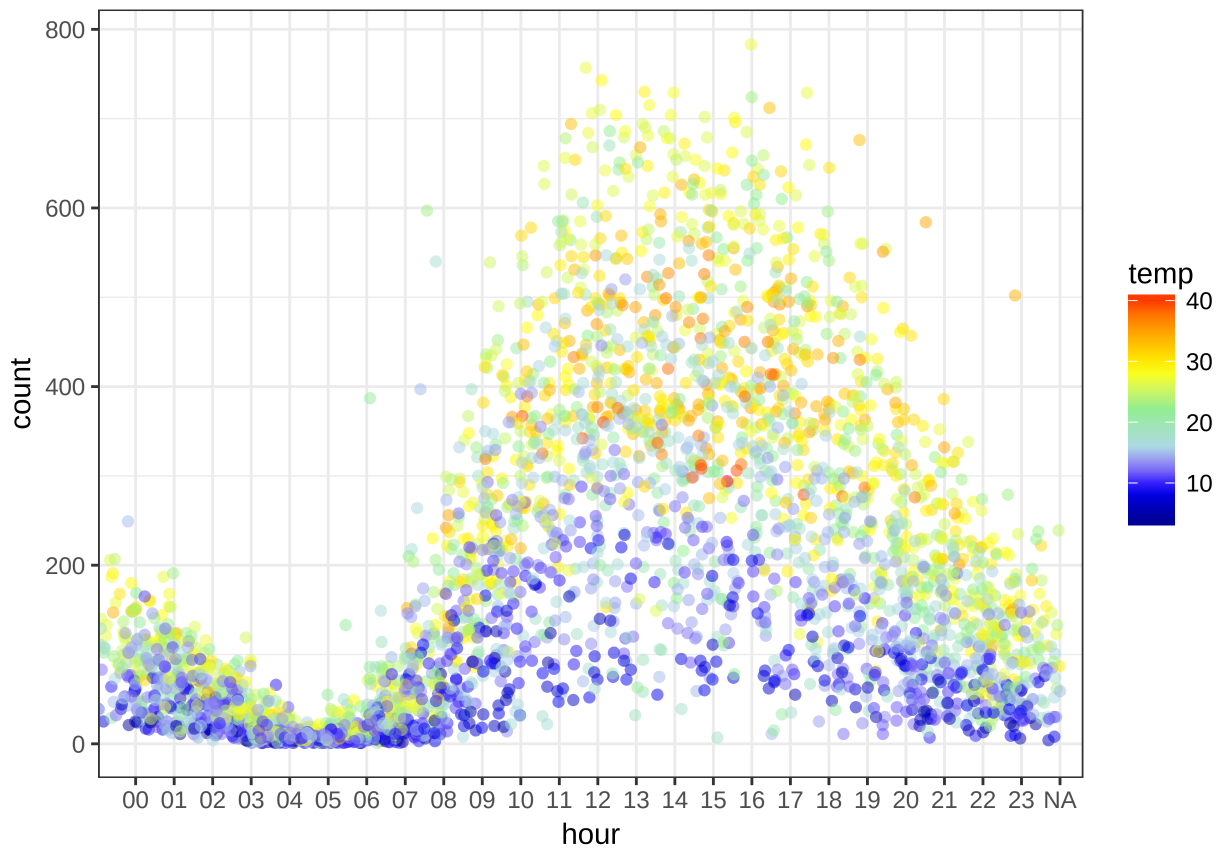
Now I create a scatterplot of hour versus count with color scale based on temperature. First create the plot for the working days.

```
ggplot(filter(train,workingday==1),aes(hour,count)) +
  geom_point(aes(color=temp),alpha=0.5,position=position_jitter(w=1, h=0)) + #position_jitter adds rand
  scale_color_gradientn(colors=c('dark blue','blue','light blue','light green','yellow','orange','red'))
  theme_bw()
```



Now create the same plot for the non working days.

```
ggplot(filter(train,workingday==0),aes(hour,count)) +
  geom_point(aes(color=temp),alpha=0.5,position=position_jitter(w=1, h=0)) +
  scale_color_gradientn(colors=c('dark blue','blue','light blue','light green','yellow','orange','red'))
  theme_bw()
```



The plots illustrate the working days have peak bike activity during the morning (around 8am) and right after work (around 6pm), with some lunchtime activity. Whereas the non-working days have a steady rise and fall for the afternoon.

## Prediction

Change the hour column to a column of numeric values.

```
train$hour = sapply(train$hour, as.numeric)
```

Let's check whether missing values exist.

```
# Checking missing data
sapply(train, function(x) sum(is.na(x)))
```

```
##   datetime     season    holiday workingday    weather       temp 
##          2          0          0          0          0          0 
##      atemp   humidity  windspeed     casual registered      count 
##          0          0          0          0          0          0 
##       hour 
##          2
```

```
# Extract the rows with missing values
filter(train, is.na(hour)==TRUE|hour=='')
```

```
##   datetime season holiday workingday weather  temp atemp humidity
## 1     <NA>      4       0          0       3 13.94 15.15       87
## 2     <NA>      4       0           0       1 17.22 21.21       62
##   windspeed casual registered count hour
## 1   19.9995      3         27    30   NA
## 2    8.9981     10         64    74   NA
```

```
# Remove missing data rows
train = filter(train, is.na(hour)!=TRUE)
```

```
cor(train$temp,train$atemp)
```

```
## [1] 0.984949
```

The features temp and atemp are strongly correlated. If both features are included in the model, this will cause the issue of **Multicollinearity** (a given feature in the model can be approximated by a linear combination of the other features in the model).

- Hence I include only one temperature feature into the model.
- The features **casual** and **registered** are omitted because that is what we are going to predict.
- The feature **datetime** is omitted since the new feature **hour** is included in the model and which is more meaningful in predicting the bike rental demand.

Let's continue on building a model to forecast bike rental demand.

## Multiple Linear Regression

```
# Fitting Multiple Linear Regression to the Training set
train_subset = select(train, season, holiday, workingday, weather, temp, humidity, windspeed, count, hou
regressor = lm(formula = count ~ . , data = train_subset)

# Choosing the best model by AIC in a Stepwise Algorithm
# The step() function iteratively removes insignificant features from the model.
regressor = step(regressor)
```

```
## Start:  AIC=108617.4
## count ~ season + holiday + workingday + weather + temp + humidity +
##     windspeed + hour
##
##              Df Sum of Sq       RSS    AIC
## - workingday  1      7005 234415199 108616
## - windspeed   1      7497 234415691 108616
## - holiday     1     20510 234428704 108616
## <none>                    234408194 108617
## - weather     1     92060 234500254 108620
## - season      3   8953132 243361326 109019
## - humidity    1  11305547 245713741 109128
## - temp        1  19062534 253470728 109466
## - hour        1  24603679 259011873 109702
##
## Step:  AIC=108615.7
## count ~ season + holiday + weather + temp + humidity + windspeed +
##     hour
##
##             Df Sum of Sq       RSS    AIC
## - windspeed  1      7393 234422591 108614
## - holiday    1     15948 234431147 108614
## <none>                   234415199 108616
## - weather    1     94157 234509356 108618
## - season     3   8947933 243363132 109017
## - humidity   1  11299475 245714674 109126
## - temp       1  19069798 253484997 109465
## - hour       1  24620409 259035607 109701
##
## Step:  AIC=108614.1
## count ~ season + holiday + weather + temp + humidity + hour
##
##            Df Sum of Sq       RSS    AIC
## - holiday   1     15643 234438234 108613
## <none>                  234422591 108614
## - weather   1     88607 234511198 108616
## - season    3   8940999 243363590 109015
## - humidity  1  12585707 247008298 109181
## - temp      1  19081089 253503680 109464
## - hour      1  24728159 259150751 109704
##
## Step:  AIC=108612.8
## count ~ season + weather + temp + humidity + hour
##
##            Df Sum of Sq       RSS    AIC
## <none>                  234438234 108613
## - weather   1     88199 234526433 108615
## - season    3   8934766 243373000 109014
## - humidity  1  12584122 247022356 109180
## - temp      1  19088993 253527227 109463
## - hour      1  24727490 259165725 109702
```

```r
# Predicting the Test set results
test$datetime = as.POSIXct(test$datetime, format="%Y-%m-%d %H:%M:%S")
test$hour = sapply(test$datetime, function(x) format(x,"%H"))
test$hour = sapply(test$hour, as.numeric)
test$season = factor(test$season)
y_pred = predict(regressor, test)

# Save the results
results <- data.frame(datetime = test$datetime, count = y_pred)

# Write the results to a csv file
write.csv(results, file = 'BikeSharingDemand_MLR.csv', row.names = FALSE, quote=FALSE)
```

## Support Vector Regression

```r
# Fitting SVR to the dataset
regressor = svm(formula = count ~ .,
                data = train_subset,
                type = 'eps-regression',
                kernel = 'radial')

# Predicting a new result
```

```r
y_pred = predict(regressor, test)

# Save the results
results <- data.frame(datetime = test$datetime, count = y_pred)

# Write the results to a csv file
write.csv(results, file = 'BikeSharingDemand_SVR.csv', row.names = FALSE, quote=FALSE)
```

### Random Forest Regression

```r
# Fitting Random Forest Regression to the dataset
set.seed(743)
regressor = randomForest(x = train_subset[,-which(names(train_subset)=="count")],
                         y = train_subset$count)

# Predicting a new result with Random Forest Regression
y_pred = predict(regressor, test)

# Save the results
results <- data.frame(datetime = test$datetime, count = y_pred)

# Write the results to a csv file
write.csv(results, file = 'BikeSharingDemand_RandomForest.csv', row.names = FALSE, quote=FALSE)
```

## Conclusion

Predictions from Multiple Linear Regression and Support Vector Regression contain negative values. Random Forest Regression prediction results were successfully submitted.

I am eager to learn more and develop skills on machine learning. Any feedback is very welcome!