Seoul Bike Sharing Demand Prediction Report

Table of Contents

Problem Statement

The Idea of a bike-sharing system is almost 50 years old. First presented in Ernest Callenbach's utopian novel Ecotopia(1975) illustrates the idea of a society that survives without using any fossil fuel. Currently, almost every developed urban city has a very advanced bike-sharing system. It not only has a very positive impact on the financial and transport system of the cities, but it also has a tremendously significant impact on the environment. As always, benefits come with significant challenges. It is always a challenge to have enough bikes that can facilitate all the citizens and have fewer waiting times regardless of the weather or traffic conditions. Therefore, it is essential to predict the bike count each hour for a stable supply of rented bikes.

Data Description

The dataset contains weather information (Temperature, Humidity, Windspeed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), the number of bikes rented per hour, and date information.

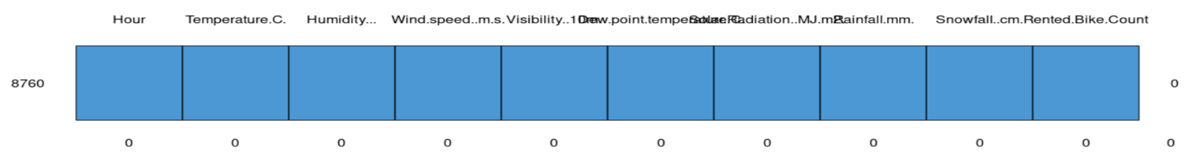
Dataset

For this project, Seoul Bike Share program data is used. This dataset contains the total count of rented bikes at each hour, the date of observation, and meteorological information (Humidity, Snowfall, Rainfall, Temperature Season, and so on) for that hour.

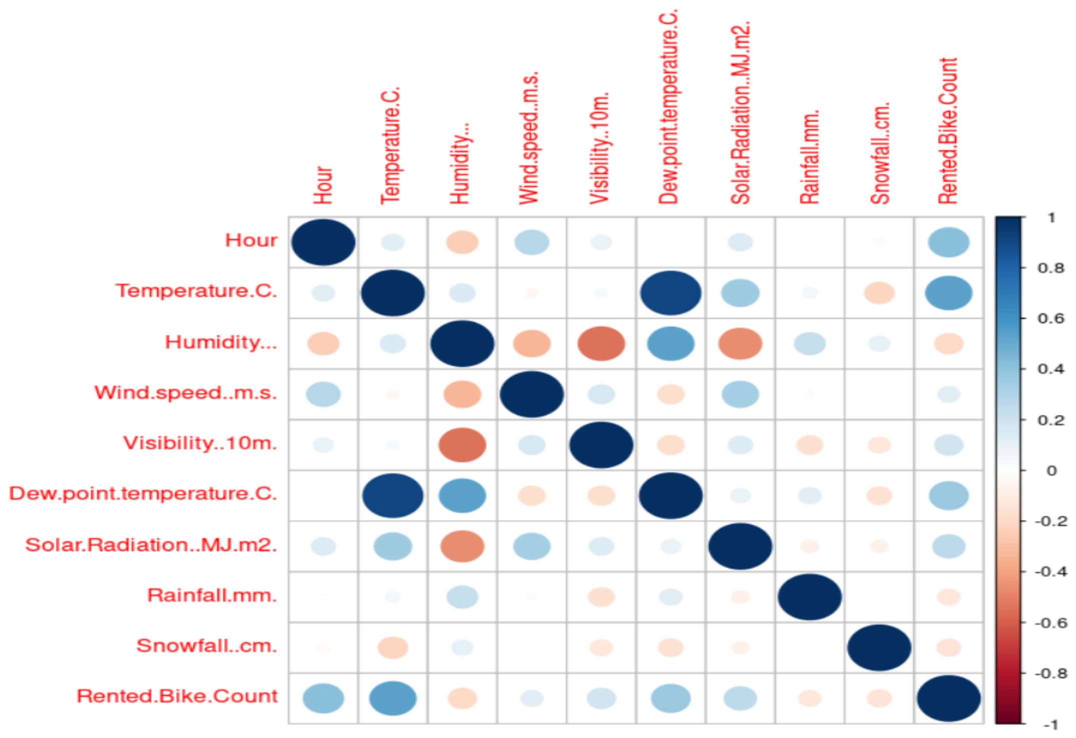
The observations were recorded over 365 days, from December 2017 to November 2018.

Data Information

• Dataset contains 8760 rows and 14 columns.
• Dataset contains 0 Duplicate rows.
• This Dataset contains four categorical and ten numerical variables.
• It contains 0 missing values.



Correlation Matrix



According to the correlation matrix, the “Rented Bike count” has a higher number of dependencies which means this variable is dependent on other variables in the dataset. So, we decide to use this variable as a predictor.

<u>Dependent variable</u>

• Rented Bike count - Count of bikes rented at each hour

<u>Independent variables</u>

• Date: year-month-day

• Hour - Hour of the day

• Temperature-Temperature in Celsius

• Humidity - %

• Windspeed - m/s

• Visibility - 10m

• Dew point temperature - Celsius

• Solar radiation - MJ/m2

• Rainfall - mm

• Snowfall - cm

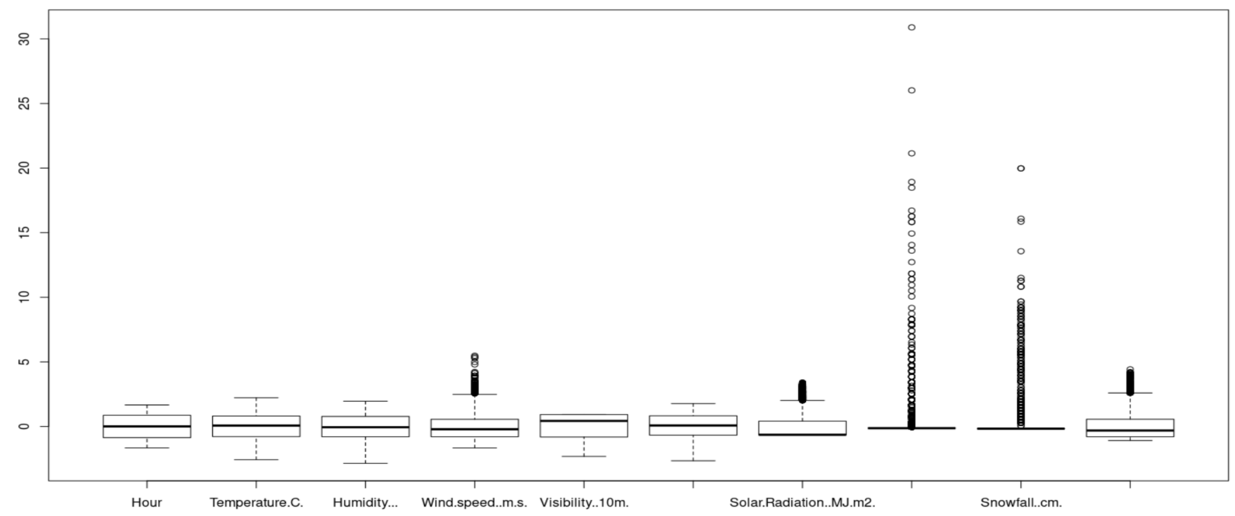
• Seasons - Winter, Spring, Summer, Autumn

• Holiday - Holiday/No holiday

• Functional Day - NoFunc(Non-Functional Hours), Fun(Functional hours)

Outliers

The box plot in the image below illustrates our dataset's outliers.



count.outliers_rainfall	528L
count.outliers_rented.bike.count	157L
count.outliers_snowfall	443L
count.outliers_solar.radiation	641L
count.outliers_wind.speed	161L

Model Creation:

Random Forest

For the prediction, we have used the Random Forest algorithm.
Random forest is a supervised machine learning model used for regression and classification problems, Random Forest has numerous tree models combined, and more trees mean a more robust model.

Only one tree is based on training data in a single decision tree model. The tree measures all the training parameters, but accuracy gets very low if new data has come.

This is called overfitting, and it is due to our model attracting noise along with patterns. So the accuracy of training data is high, but in the testing phase, it’s less. These models have low bias and high variance. This problem can be solved using the random forest because it works on different data trees in the dataset.

Algorithm

1. For b = 1 to B:

   (a) Draw a bootstrap sample Z∗ of size N from the training data.
   (b) Grow the random-forest tree Tb to the bootstrapped data by recursively repeating the following steps for each terminal node of the tree until the minimum node size min is reached.
      (i) Select m variables at random from the p variables.
      (ii) Pick the best variable/split point among the m
      (iii) Split the node into two daughter nodes.

2. Output the ensemble of trees {Tb}B 1.

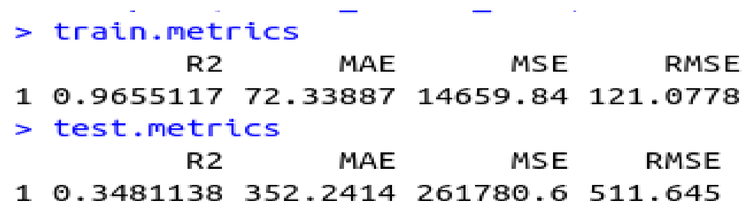
To predict a new point x:

$$\textit{Regression:}\;\; \hat{f}^{B}_{\mathrm{rf}}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x).$$

In the “Performance Metrics ..” sections, we showed 4 performance metrics which are R^2, MAE, MSE and RMSE. Among these 4 metrics, we specifically used R^2 (Goodness of Fit) and RMSE for model training dataset and RMSE for test dataset.

Performance Metrics of Baseline Model

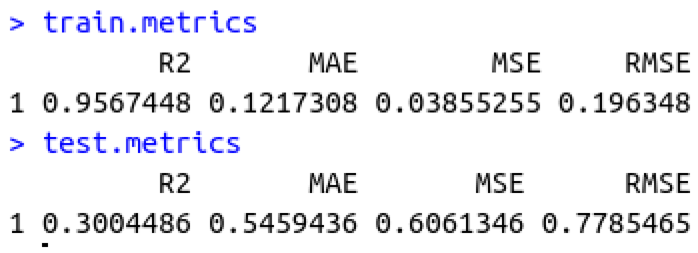
For the baseline model, I have trained random forest along with the outliers. The below image illustrates the performance metrics. (code in appendix)



```
> train.metrics
         R2      MAE      MSE     RMSE
1 0.9655117 72.33887 14659.84 121.0778
> test.metrics
         R2      MAE      MSE    RMSE
1 0.3481138 352.2414 261780.6 511.645
```

Then, we removed the outliers which are described in the “Outliers” section. After removing the outliers, we again fitted the random forest model with a normalized dataset. The performance metric is given in the next section.

Performance metrics of Final Model

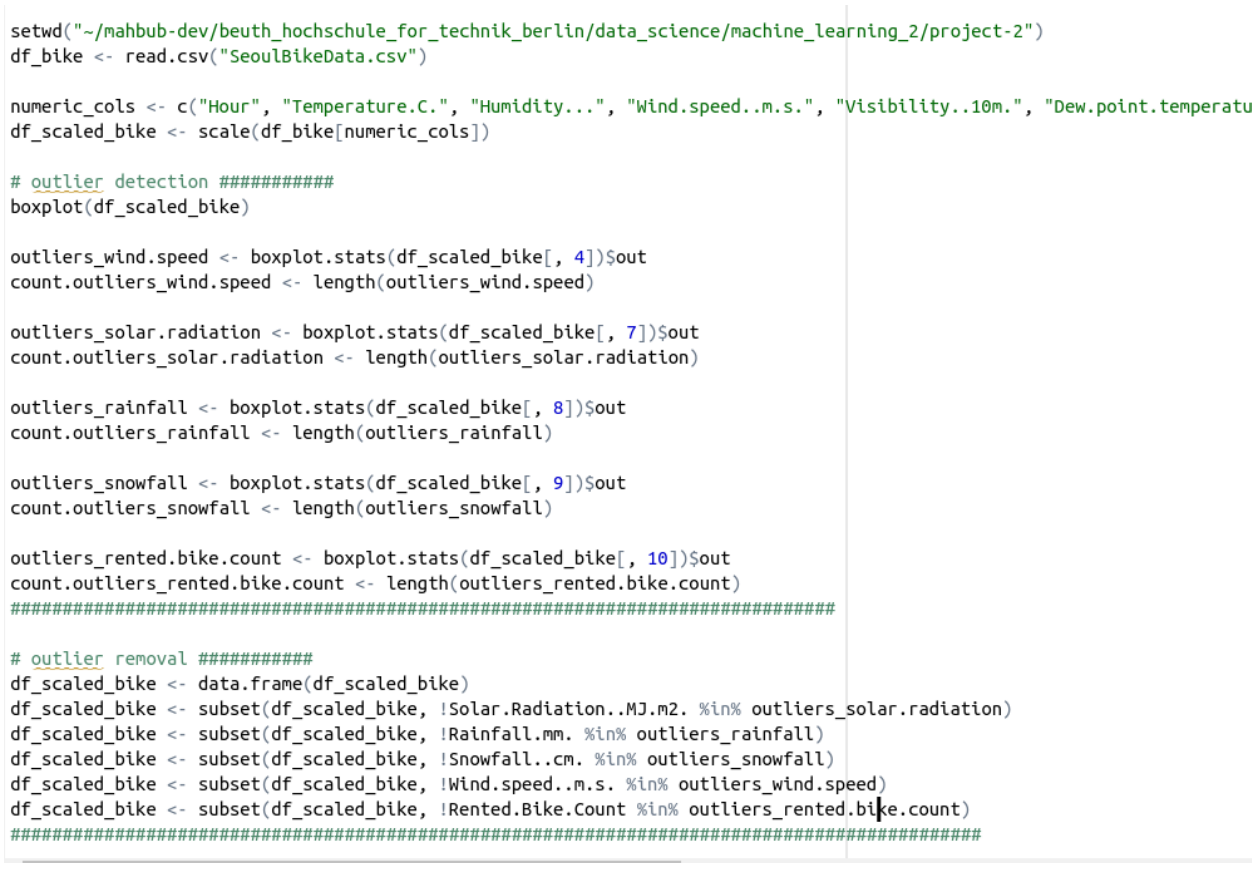


```
> train.metrics
         R2       MAE        MSE     RMSE
1 0.9567448 0.1217308 0.03855255 0.196348
> test.metrics
         R2       MAE       MSE      RMSE
1 0.3004486 0.5459436 0.6061346 0.7785465
```

According to the above performance metric, the RMSE of the test dataset is better than baseline’s test dataset’s RMSE.

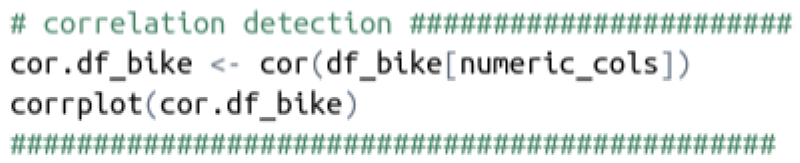
Appendix

Outlier



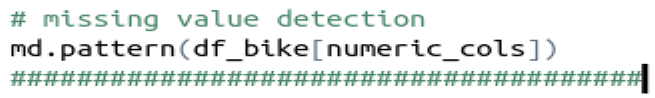
```
setwd("~/mahbub-dev/beuth_hochschule_for_technik_berlin/data_science/machine_learning_2/project-2")
df_bike <- read.csv("SeoulBikeData.csv")

numeric_cols <- c("Hour", "Temperature.C.", "Humidity...", "Wind.speed..m.s.", "Visibility..10m.", "Dew.point.temperatu
df_scaled_bike <- scale(df_bike[numeric_cols])

# outlier detection ###########
boxplot(df_scaled_bike)

outliers_wind.speed <- boxplot.stats(df_scaled_bike[, 4])$out
count.outliers_wind.speed <- length(outliers_wind.speed)

outliers_solar.radiation <- boxplot.stats(df_scaled_bike[, 7])$out
count.outliers_solar.radiation <- length(outliers_solar.radiation)

outliers_rainfall <- boxplot.stats(df_scaled_bike[, 8])$out
count.outliers_rainfall <- length(outliers_rainfall)

outliers_snowfall <- boxplot.stats(df_scaled_bike[, 9])$out
count.outliers_snowfall <- length(outliers_snowfall)

outliers_rented.bike.count <- boxplot.stats(df_scaled_bike[, 10])$out
count.outliers_rented.bike.count <- length(outliers_rented.bike.count)
###############################################################################

# outlier removal ###########
df_scaled_bike <- data.frame(df_scaled_bike)
df_scaled_bike <- subset(df_scaled_bike, !Solar.Radiation..MJ.m2. %in% outliers_solar.radiation)
df_scaled_bike <- subset(df_scaled_bike, !Rainfall.mm. %in% outliers_rainfall)
df_scaled_bike <- subset(df_scaled_bike, !Snowfall..cm. %in% outliers_snowfall)
df_scaled_bike <- subset(df_scaled_bike, !Wind.speed..m.s. %in% outliers_wind.speed)
df_scaled_bike <- subset(df_scaled_bike, !Rented.Bike.Count %in% outliers_rented.bike.count)
#############################################################################################
```

Correlation matrix



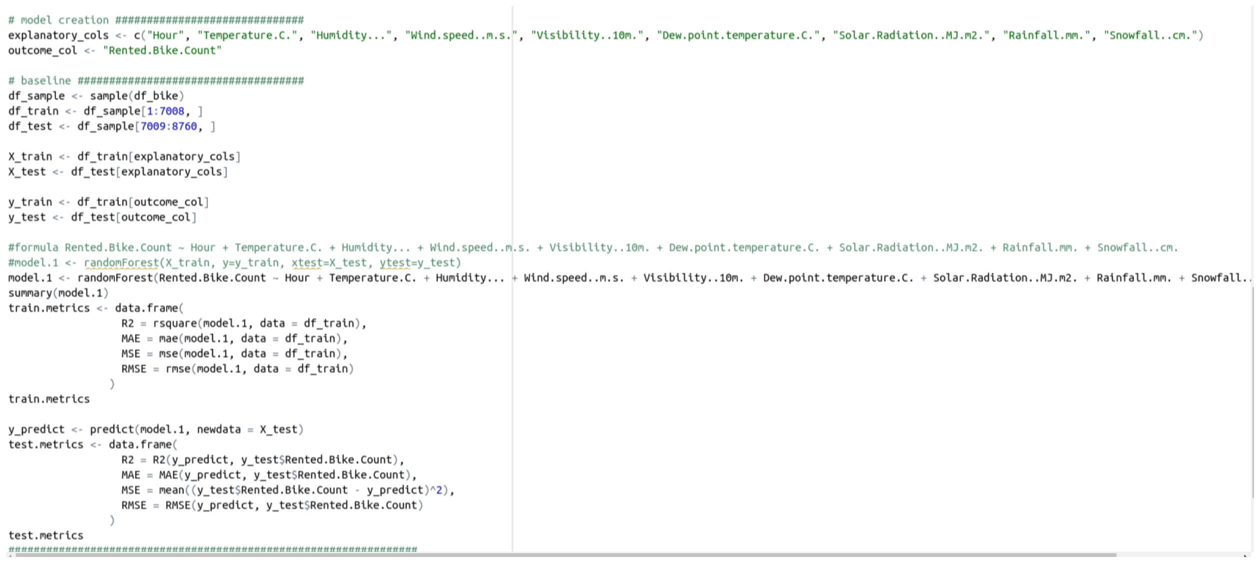
```
# correlation detection ########################
cor.df_bike <- cor(df_bike[numeric_cols])
corrplot(cor.df_bike)
###############################################
```

Missing Values



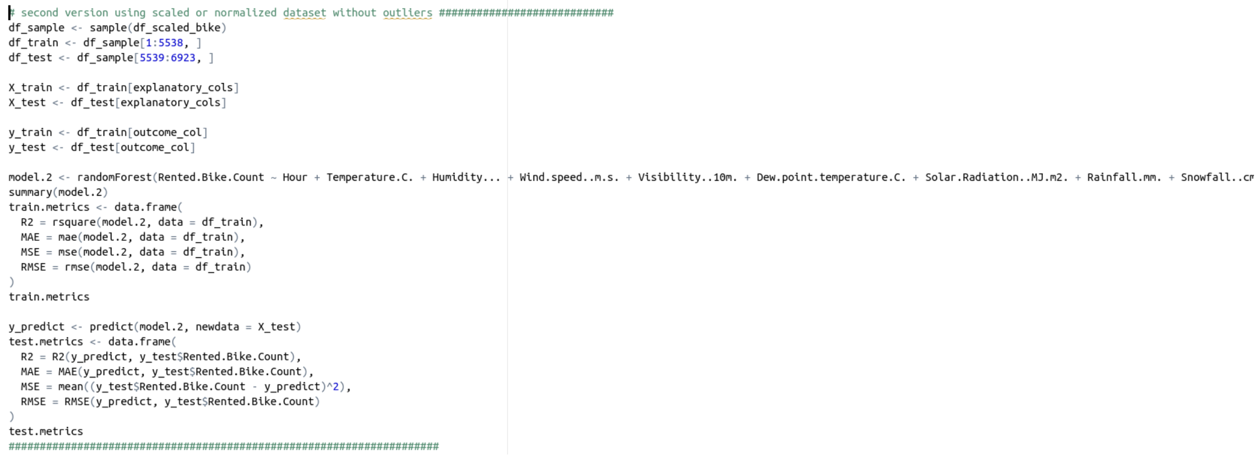
```
# missing value detection
md.pattern(df_bike[numeric_cols])
#######################################
```

Random forest Baseline with outliers



```r
# model creation ##############################
explanatory_cols <- c("Hour", "Temperature.C.", "Humidity...", "Wind.speed..m.s.", "Visibility..10m.", "Dew.point.temperature.C.", "Solar.Radiation..MJ.m2.", "Rainfall.mm.", "Snowfall..cm.")
outcome_col <- "Rented.Bike.Count"

# baseline ####################################
df_sample <- sample(df_bike)
df_train <- df_sample[1:7008, ]
df_test <- df_sample[7009:8760, ]

X_train <- df_train[explanatory_cols]
X_test <- df_test[explanatory_cols]

y_train <- df_train[outcome_col]
y_test <- df_test[outcome_col]

#formula Rented.Bike.Count ~ Hour + Temperature.C. + Humidity... + Wind.speed..m.s. + Visibility..10m. + Dew.point.temperature.C. + Solar.Radiation..MJ.m2. + Rainfall.mm. + Snowfall..cm.
#model.1 <- randomForest(X_train, y=y_train, xtest=X_test, ytest=y_test)
model.1 <- randomForest(Rented.Bike.Count ~ Hour + Temperature.C. + Humidity... + Wind.speed..m.s. + Visibility..10m. + Dew.point.temperature.C. + Solar.Radiation..MJ.m2. + Rainfall.mm. + Snowfall..
summary(model.1)
train.metrics <- data.frame(
                  R2 = rsquare(model.1, data = df_train),
                  MAE = mae(model.1, data = df_train),
                  MSE = mse(model.1, data = df_train),
                  RMSE = rmse(model.1, data = df_train)
                )
train.metrics

y_predict <- predict(model.1, newdata = X_test)
test.metrics <- data.frame(
                  R2 = R2(y_predict, y_test$Rented.Bike.Count),
                  MAE = MAE(y_predict, y_test$Rented.Bike.Count),
                  MSE = mean((y_test$Rented.Bike.Count - y_predict)^2),
                  RMSE = RMSE(y_predict, y_test$Rented.Bike.Count)
                )
test.metrics
#################################################################
```

Random Forest Final Model



```r
# second version using scaled or normalized dataset without outliers ###########################
df_sample <- sample(df_scaled_bike)
df_train <- df_sample[1:5538, ]
df_test <- df_sample[5539:6923, ]

X_train <- df_train[explanatory_cols]
X_test <- df_test[explanatory_cols]

y_train <- df_train[outcome_col]
y_test <- df_test[outcome_col]

model.2 <- randomForest(Rented.Bike.Count ~ Hour + Temperature.C. + Humidity... + Wind.speed..m.s. + Visibility..10m. + Dew.point.temperature.C. + Solar.Radiation..MJ.m2. + Rainfall.mm. + Snowfall..cm
summary(model.2)
train.metrics <- data.frame(
  R2 = rsquare(model.2, data = df_train),
  MAE = mae(model.2, data = df_train),
  MSE = mse(model.2, data = df_train),
  RMSE = rmse(model.2, data = df_train)
)
train.metrics

y_predict <- predict(model.2, newdata = X_test)
test.metrics <- data.frame(
  R2 = R2(y_predict, y_test$Rented.Bike.Count),
  MAE = MAE(y_predict, y_test$Rented.Bike.Count),
  MSE = mean((y_test$Rented.Bike.Count - y_predict)^2),
  RMSE = RMSE(y_predict, y_test$Rented.Bike.Count)
)
test.metrics
####################################################################
```