Seoul Bike Sharing Demand Prediction Report

By: Yaqoob David

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.

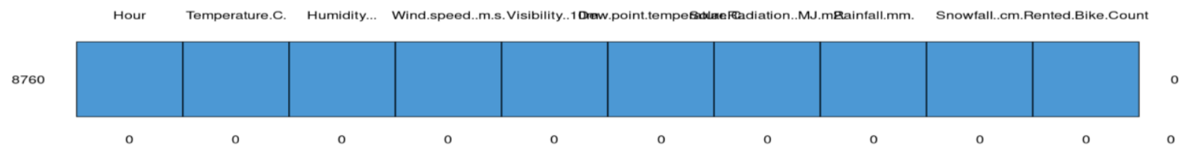
Dataset

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.

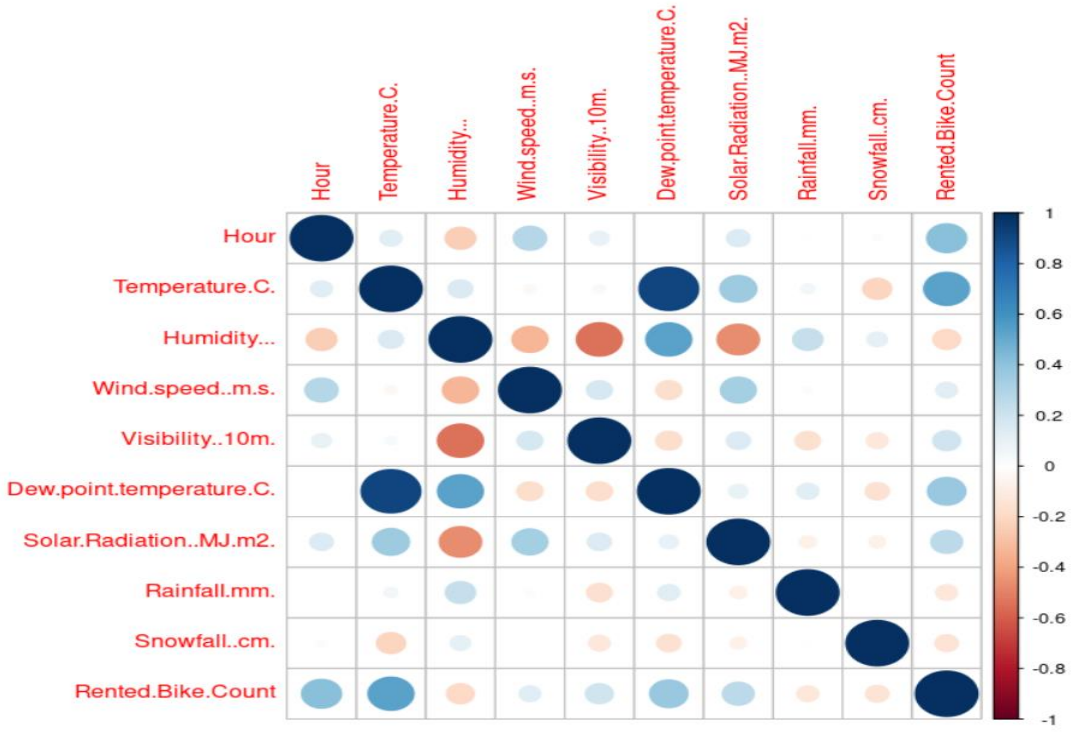
The observations were recorded over 365 days, from December 2017 to November 2018.

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

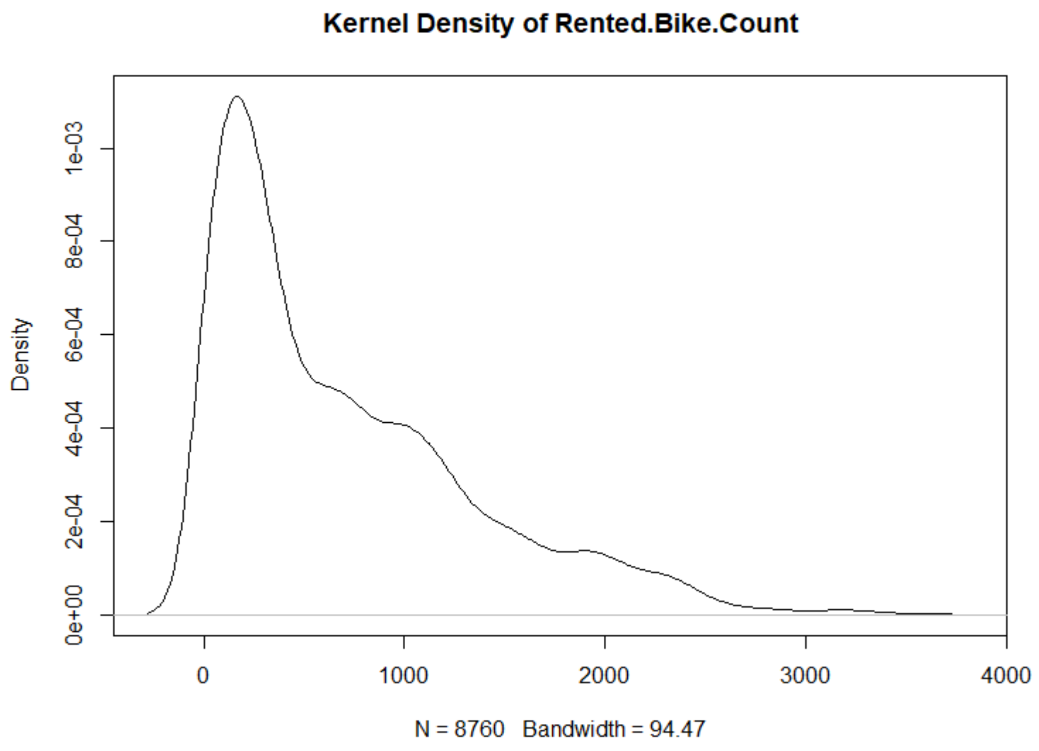
<u>Independent variables</u>

• Date: year-month-day

• Hour - Hour of the day

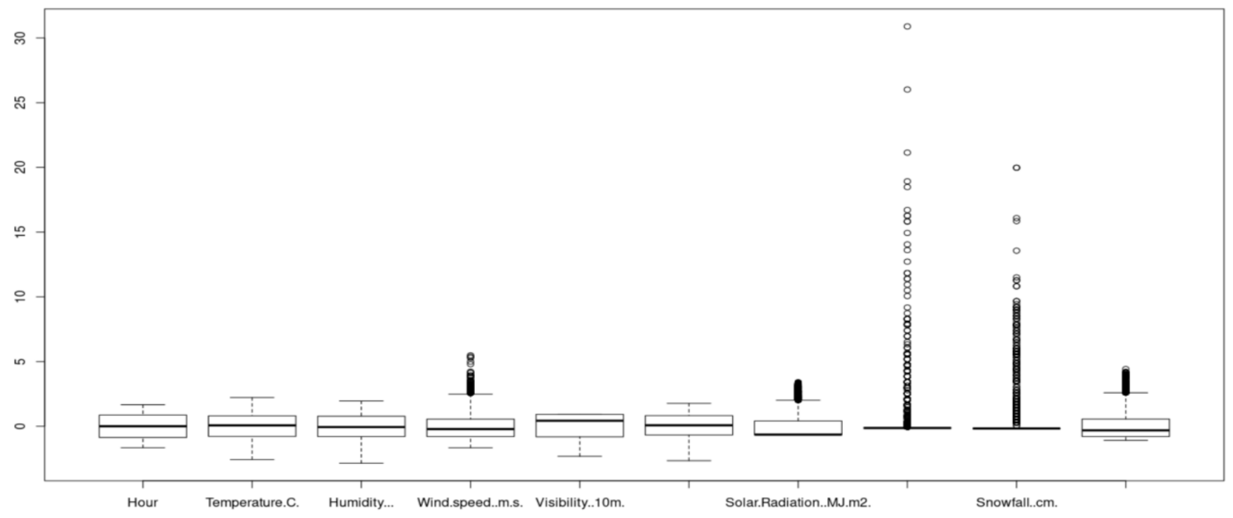
- Temperature-Temperature in Celsius
- Humidity - %
- Windspeed - m/s
- Visibility - 10m
- Dew point temperature - Celsius
- Solar radiation - MJ/m2
- Rainfall - mm
- Snowfall - cm
- Seasons - Winter, Spring, Summer, Autumn
- Holiday - Holiday/No holiday
- Functional Day - NoFunc(Non-Functional Hours), Fun(Functional hours)

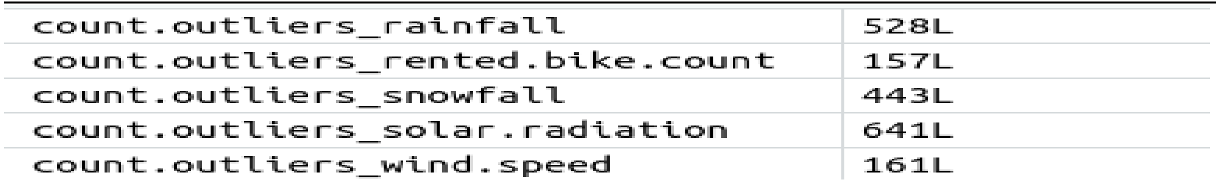
The density of the outcome variable



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
```

Models

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

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:

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

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

Mathematical Overview (Spline smoothing)

A major problem of using RSS to fit the data is overfitting. Spline smoothing can be used to eliminate too much wiggliness in the predictor function.

Another problem with some non-linear regression methods is the choice of the number of knot points. Spline smoothing also avoids this problem.

The penalized sum of squares that spline smoothing uses:

$$J(f,\lambda) = \sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int_{x_1}^{x_n} f''(x)^2 dx.$$

The integral of f″(x) in the above equation is a penalty term that penalizes wiggliness. The higher the wiggliness, the higher the value of the integral of f″(x). The aim is to minimize J. The objective is to minimize J. λ is a constant, and the user needs to choose the value. λ basically corresponds to the level of smoothing.

Mathematical Overview (Generalised additive Models):

A generalised additive model has the form:

$$Y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \ldots + f_p(x_{ip}) + \epsilon_i.$$

Where, each fk can be a different non-linear regression method.

The advantage of GAM is that we can add as many variables as we want in the model.

For simplicity, all fk's will be considered spline smoothing functions from here.

GAM uses backfitting algorithm to obtain all p estimating functions. The core idea of the algorithm is to keep away the effects of all spline smoothing functions other than fk while obtaining fk. For simplicity, the details of the algorithm are omitted in the report.

Model Evaluation (Random Forest)

Evaluation Metrics

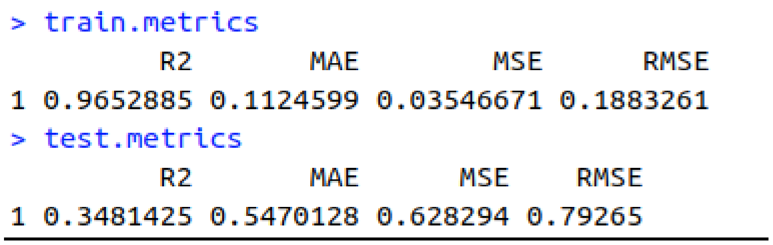
For the Evaluation of the model, we use MSE and RMSE.

The formula for RMSE is:
RMSE = √(1/n) * Σ(actual_value - predicted_value)^2

The formula for MSE is:
MSE = (1/n) * Σ(actual_value - predicted_value)^2

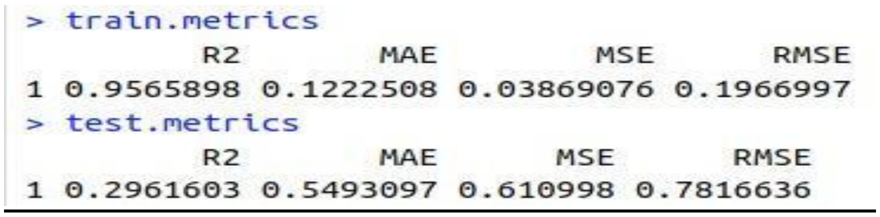
Performance Metrics of Baseline Model

After normalizing the dataset for the baseline model, I trained random forest along with the outliers. The below image illustrates the performance metrics. (code in appendix)


```
> train.metrics
         R2       MAE        MSE      RMSE
1 0.9652885 0.1124599 0.03546671 0.1883261
> test.metrics
         R2       MAE      MSE    RMSE
1 0.3481425 0.5470128 0.628294 0.79265
```

MSE and RMSE are sensitive to outliers, meaning that models that perform well on most of the data but perform very poorly on a small number of observations will have a high RMSE. Therefore, we removed the outliers described in the “Outliers” section. After removing the outliers, we again fitted the random forest model with a normalized dataset(code is in the appendix).

Performance metrics of Final Model


```
> train.metrics
         R2       MAE        MSE      RMSE
1 0.9565898 0.1222508 0.03869076 0.1966997
> test.metrics
         R2       MAE      MSE      RMSE
1 0.2961603 0.5493097 0.610998 0.7816636
```

The above metric clearly illustrates that after removing the outliers, the RMSE and MSE of the test dataset improved, and its better than the baseline.

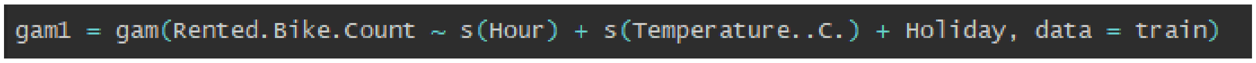
Fitting Process and RMSE (Spline smoothing and GAM)

Firstly, spline smoothing on hour was used to predict rented bike count. 80-20 train test split was done and cross validation was used for hyperparameter tuning. The degrees of freedom came out 24. The rmse on the test set came out 0.8370789.

Secondly, spline smoothing on temperature was used. 80-20 train test split was done and cross validation was used for hyperparameter tuning. The degrees of freedom came out 14.69512. The rmse on the test set came out 0.8255876.

Then, GAM was used to include multiple variables.

First GAM:


```
gam1 = gam(Rented.Bike.Count ~ s(Hour) + s(Temperature..C.) + Holiday, data = train)
```

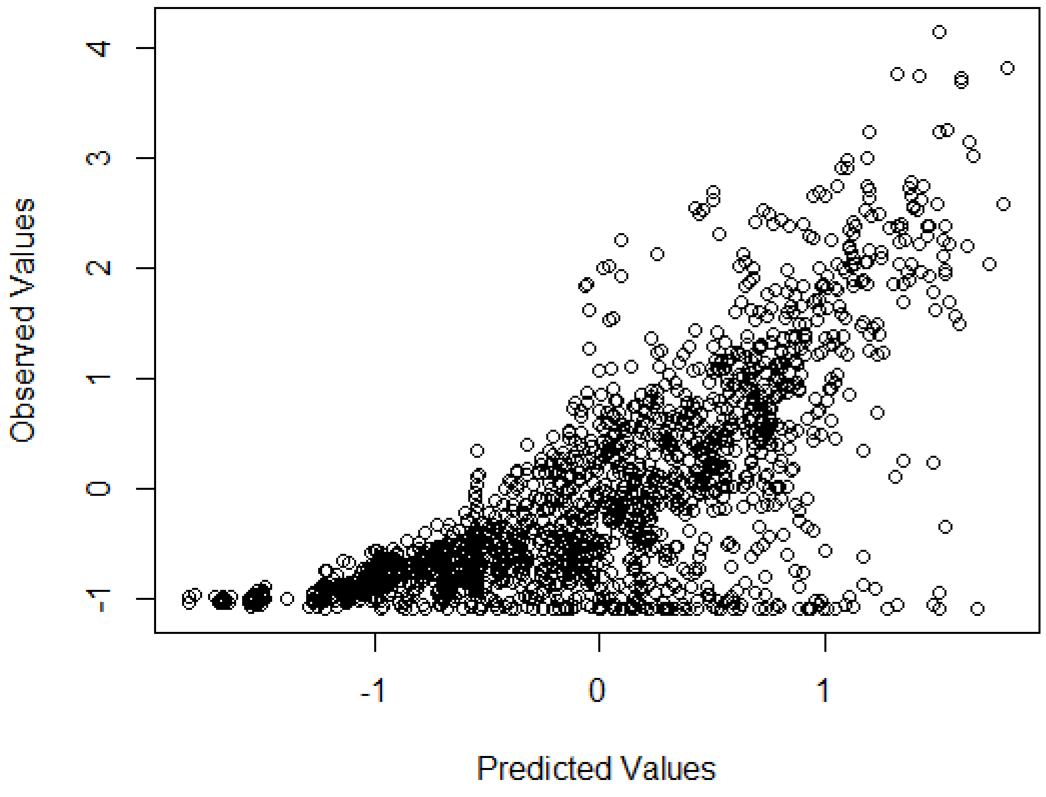
Spline smoothing was done on hour and temperature and the Holiday column was also added. 80-20 train test split was done. The rmse on the test set came out 0.6923511.

Second GAM:

```
gam2 = gam(Rented.Bike.Count ~ s(Hour) + s(Temperature..C.) + Holiday + Seasons, data = train)
```

Spline smoothing was done on hour and temperature and Holiday and Seasons was also added. 80-20 train test split was done. The rmse on the test set came out 0.6796763.

Below shows the observed vs predicted values of the final model:

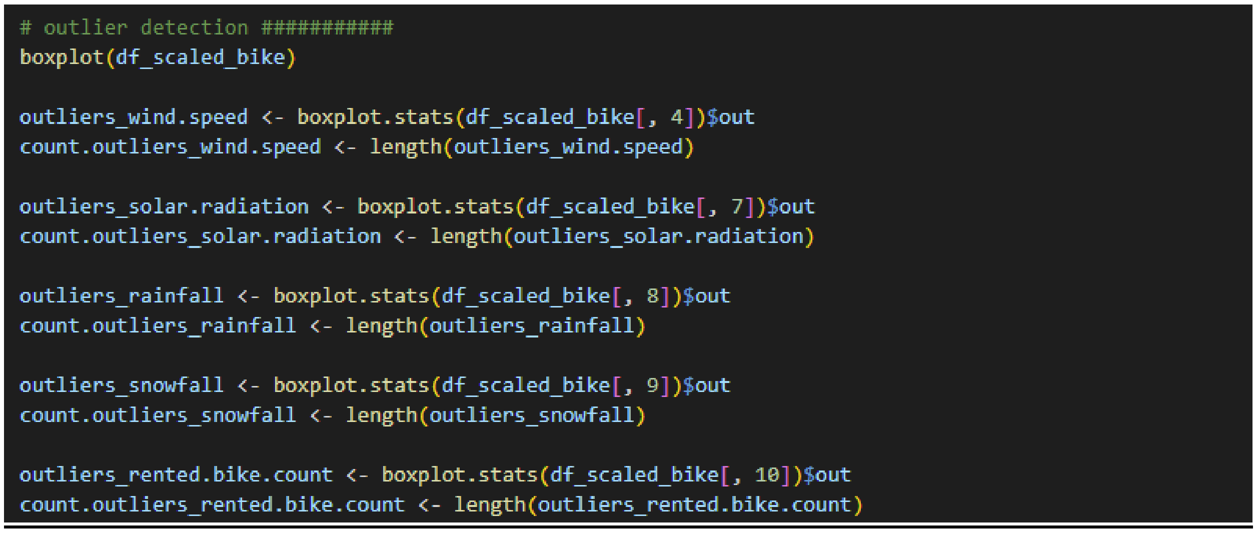


Comparison

We are using rmse for evaluate our models and the second GAM performed better than all other models with a rmse score of 0.6796763 on test set.

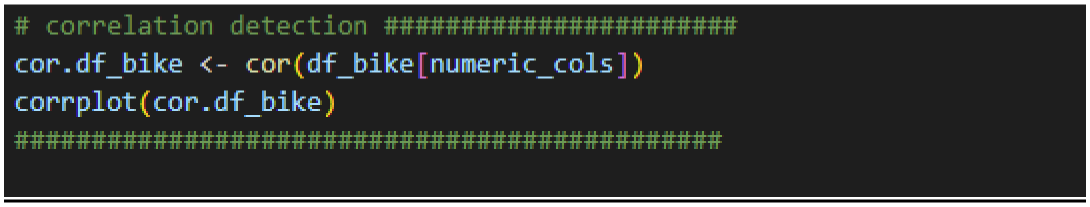
Appendix

Outlier



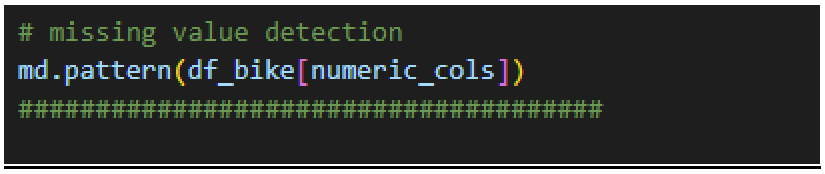
```r
# outlier detection ###########
boxplot(df_scaled_bike)

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)
```

Correlation matrix



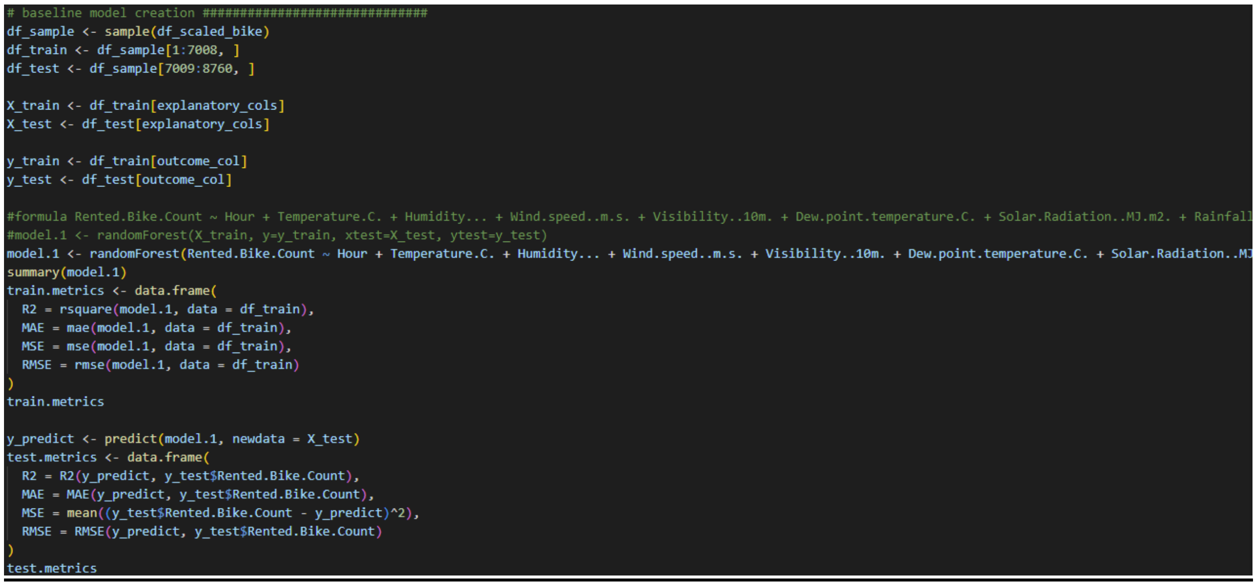
```r
# correlation detection #######################
cor.df_bike <- cor(df_bike[numeric_cols])
corrplot(cor.df_bike)
##############################################
```

Missing Values



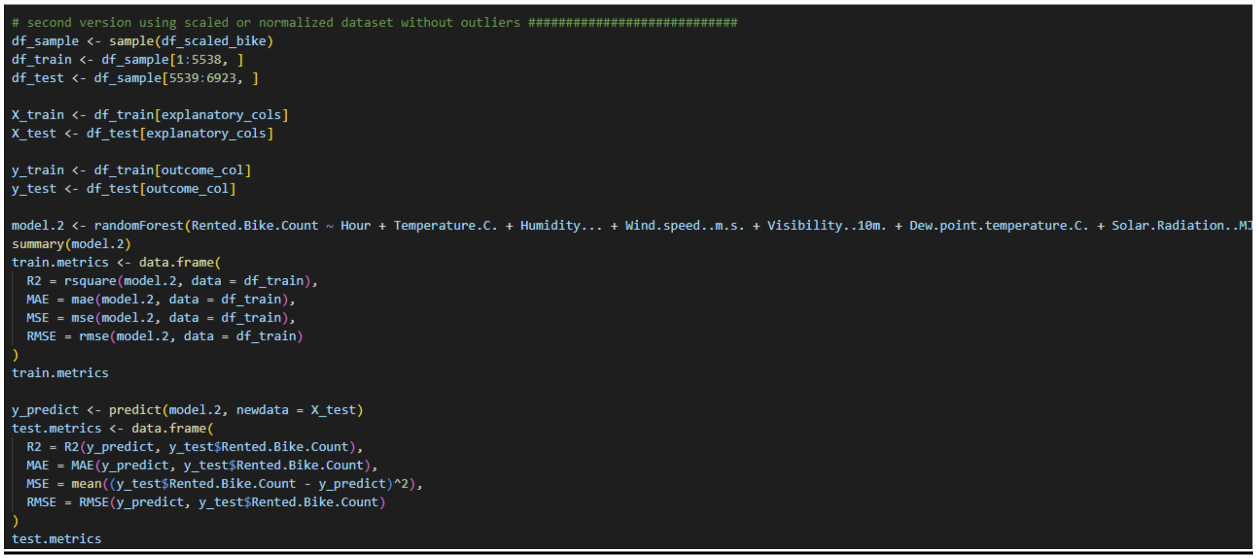
```r
# missing value detection
md.pattern(df_bike[numeric_cols])
########################################
```

Random forest Baseline with outliers



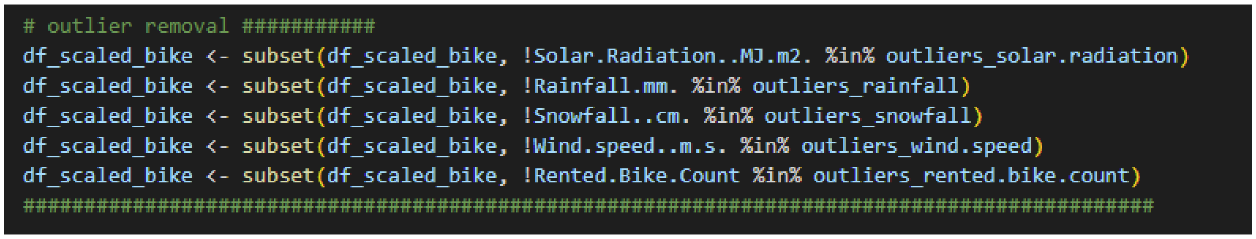
```
# baseline model creation ##############################
df_sample <- sample(df_scaled_bike)
df_train <- df_sample[1:7008, ]
df_test <- df_sample[7009:8760, ]

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
#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
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



```
# 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
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
```

Outlier Removal



```r
# outlier removal ###########
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)
##############################################################################################
```