

Seoul Bike Count Regression Project

Kasvina Tirumal

2024-08-30

Goal: Predict hourly bike rental count in Seoul using environmental information

Loading and Renaming Columns Firstly, we started with loading in a dataset that contains the count of public bicycles rented per hour in the Seoul Bike Sharing System, with corresponding weather data and holiday information. This dataset was obtained from the UC Irvine Machine Learning Repository, which can be found in the following link: <https://archive.ics.uci.edu/dataset/560/seoul+bike+sharing+demand>.

```
bike_df <- read.csv("seoulbikedata.csv", fileEncoding = "latin1")
dim(bike_df)

## [1] 8760   14

head(bike_df)

##           Date Rented.Bike.Count Hour Temperature..C. Humidity...
## 1 01/12/2017            254     0        -5.2      37
## 2 01/12/2017            204     1        -5.5      38
## 3 01/12/2017            173     2        -6.0      39
## 4 01/12/2017            107     3        -6.2      40
## 5 01/12/2017             78     4        -6.0      36
## 6 01/12/2017            100     5        -6.4      37
##   Wind.speed..m.s. Visibility..10m. Dew.point.temperature..C.
## 1            2.2          2000        -17.6
## 2            0.8          2000        -17.6
## 3            1.0          2000        -17.7
## 4            0.9          2000        -17.6
## 5            2.3          2000        -18.6
## 6            1.5          2000        -18.7
##   Solar.Radiation..MJ.m2. Rainfall.mm. Snowfall..cm. Seasons Holiday
## 1                  0          0          0 Winter No Holiday
## 2                  0          0          0 Winter No Holiday
## 3                  0          0          0 Winter No Holiday
## 4                  0          0          0 Winter No Holiday
## 5                  0          0          0 Winter No Holiday
## 6                  0          0          0 Winter No Holiday
##   Functioning.Day
## 1            Yes
## 2            Yes
## 3            Yes
## 4            Yes
## 5            Yes
## 6            Yes
```

Upon loading the dataset, we found that this dataset has 8760 observations and 14 columns, with one of the columns being `Rentad.Bike.Count` which is our *response variable* that we ultimately want to predict within this project using the information provided by the other features.

As seen below, few columns were then renamed to improve readability and make analysis easier later on.

```
colnames(bike_df)[colnames(bike_df) == 'Temperature..C.'] <- 'Temperature'
colnames(bike_df)[colnames(bike_df) == 'Humidity...'] <- 'Humidity'
colnames(bike_df)[colnames(bike_df) == 'Wind.speed..m.s.'] <- 'Wind.Speed'
colnames(bike_df)[colnames(bike_df) == 'Visibility..10m.'] <- 'Visibility'
colnames(bike_df)[colnames(bike_df) == 'Dew.point.temperature..C.'] <- 'Dew.Point.Temp'
colnames(bike_df)[colnames(bike_df) == 'Solar.Radiation..MJ.m2.'] <- 'Solar.Radiation'
colnames(bike_df)[colnames(bike_df) == 'Rainfall.mm.'] <- 'Rainfall'
colnames(bike_df)[colnames(bike_df) == 'Snowfall..cm.'] <- 'Snowfall'
head(bike_df)

##           Date Rented.Bike.Count Hour Temperature Humidity Wind.Speed Visibility
## 1 01/12/2017              254     0      -5.2       37      2.2      2000
## 2 01/12/2017              204     1      -5.5       38      0.8      2000
## 3 01/12/2017              173     2      -6.0       39      1.0      2000
## 4 01/12/2017              107     3      -6.2       40      0.9      2000
## 5 01/12/2017               78     4      -6.0       36      2.3      2000
## 6 01/12/2017              100     5      -6.4       37      1.5      2000
##   Dew.Point.Temp Solar.Radiation Rainfall Snowfall Seasons Holiday
## 1      -17.6            0         0        0  Winter No Holiday
## 2      -17.6            0         0        0  Winter No Holiday
## 3      -17.7            0         0        0  Winter No Holiday
## 4      -17.6            0         0        0  Winter No Holiday
## 5      -18.6            0         0        0  Winter No Holiday
## 6      -18.7            0         0        0  Winter No Holiday
##   Functioning.Day
## 1          Yes
## 2          Yes
## 3          Yes
## 4          Yes
## 5          Yes
## 6          Yes
```

Data Cleaning First and foremost, we want to check whether our dataset has any missing values and deal with them accordingly by either deleting them or if the columnns is significant, use tools such as K Nearest Neighbours to estimate their values.

```
sum(is.na(bike_df))
```

```
## [1] 0
```

As seen above, we found that our dataset is clean and has no missing values. Now, we can move on to inspecting the features within the dataset.

```
head(bike_df)
```

```

##           Date Rented.Bike.Count Hour Temperature Humidity Wind.Speed Visibility
## 1 01/12/2017                  254     0      -5.2       37      2.2      2000
## 2 01/12/2017                  204     1      -5.5       38      0.8      2000
## 3 01/12/2017                  173     2      -6.0       39      1.0      2000
## 4 01/12/2017                  107     3      -6.2       40      0.9      2000
## 5 01/12/2017                   78     4      -6.0       36      2.3      2000
## 6 01/12/2017                  100     5      -6.4       37      1.5      2000
##   Dew.Point.Temp Solar.Radiation Rainfall Snowfall Seasons    Holiday
## 1      -17.6                 0        0       0 Winter No Holiday
## 2      -17.6                 0        0       0 Winter No Holiday
## 3      -17.7                 0        0       0 Winter No Holiday
## 4      -17.6                 0        0       0 Winter No Holiday
## 5      -18.6                 0        0       0 Winter No Holiday
## 6      -18.7                 0        0       0 Winter No Holiday
##   Functioning.Day
## 1          Yes
## 2          Yes
## 3          Yes
## 4          Yes
## 5          Yes
## 6          Yes

```

By viewing our dataset, we see that we have a column `Functioning.Day` which indicates whether bikes were available for rent during that hour or not. Since our goal is to predict the count of rented bikes, we only need the data of operational days since the count is simply 0 on non-operational days. Therefore, we should remove observations made during non-functioning days. After removing said observations, we then drop the `Functioning.Day` column since we have no more use of it anymore.

```

bike_df2 <- bike_df %>%
  filter(Functioning.Day == "Yes")

bike_df3 <- bike_df2 %>%
  select(-Functioning.Day)

```

We then looked at a summary of each feature within the dataset to further understand the features we are dealing with and determine the next step forward.

```
summary(bike_df3)
```

```

##           Date      Rented.Bike.Count      Hour      Temperature
##  Length:8465      Min. : 2.0      Min. : 0.00      Min. :-17.80
##  Class :character 1st Qu.:214.0     1st Qu.: 6.00     1st Qu.: 3.00
##  Mode  :character Median :542.0      Median :12.00      Median :13.50
##                               Mean  :729.2      Mean  :11.51      Mean  :12.77
##                               3rd Qu.:1084.0     3rd Qu.:18.00     3rd Qu.:22.70
##                               Max. :3556.0      Max. :23.00      Max. :39.40
##   Humidity      Wind.Speed      Visibility      Dew.Point.Temp
##  Min.   : 0.00      Min.   :0.000      Min.   : 27      Min.   :-30.600
##  1st Qu.:42.00     1st Qu.:0.900     1st Qu.: 935     1st Qu.: -5.100
##  Median :57.00     Median :1.500     Median :1690     Median :  4.700
##  Mean   :58.15     Mean   :1.726     Mean   :1434     Mean   :  3.945
##  3rd Qu.:74.00     3rd Qu.:2.300     3rd Qu.:2000     3rd Qu.: 15.200
##  Max.   :98.00     Max.   :7.400     Max.   :2000     Max.   : 27.200

```

```

## Solar.Radiation      Rainfall          Snowfall          Seasons
## Min.   :0.0000    Min.   : 0.0000    Min.   :0.00000  Length:8465
## 1st Qu.:0.0000    1st Qu.: 0.0000    1st Qu.:0.00000  Class  :character
## Median :0.0100    Median : 0.0000    Median :0.00000  Mode   :character
## Mean   :0.5679    Mean   : 0.1491    Mean   :0.07769
## 3rd Qu.:0.9300    3rd Qu.: 0.0000    3rd Qu.:0.00000
## Max.   :3.5200    Max.   :35.0000   Max.   :8.80000
## 
## Holiday
## Length:8465
## Class :character
## Mode  :character
##
##
```

We also check whether the numeric columns contain only numeric values to ensure that we are able to perform numerical operations on these during analysis later.

```

indices <- c(2, 4:9)
whether_numeric <- sapply(bike_df3[, indices], is.numeric)
print(whether_numeric)
```

```

## Rented.Bike.Count      Temperature        Humidity        Wind.Speed
##           TRUE            TRUE            TRUE            TRUE
## Visibility     Dew.Point.Temp  Solar.Radiation
##           TRUE            TRUE            TRUE
```

As seen above, the aforementioned columns contain purely numeric values. Note that I excluded the Hour column from the above check since it will be treated as a factor when building the prediction model.

Now that the data is clean, we move on to some feature engineering.

Feature Engineering

Firstly, we note that the Rainfall and Snowfall columns record the actual heights of rain and snow respectively. However, since these measurements are not available for future predictions, we instead rely on available forecasts that indicate whether rain or snow is expected. Therefore, we transformed the Rainfall and Snowfall columns into binary indicators: 1 representing the occurrence of rain or snow during that hour, while 0 indicates that there was no rain or snow respectively. Using binary encoding in this manner helps to ensure best possible model performance since it is more compatible with algorithms and prevents ordinal assumptions.

```

bike_df3$Rained <- ifelse(bike_df3$Rainfall > 0, 1, 0)
bike_df3$Snowed <- ifelse(bike_df3$Snowfall > 0, 1, 0)
bike_df4 <- bike_df3[ , !(names(bike_df3) %in% c("Rainfall", "Snowfall"))]
```

Next, instead of having the dates of each observation, we simply want to know whether that day is a weekend or not. This is because we would expect bike rental patterns to differ between weekdays and weekends. For example, they may be more used for commuting to and from work on the weekdays and more recreationally during the weekends. To achieve this, I will create an `Is.Weekend` column.

```

bike_df4$Date <- as.Date(bike_df4$Date, format = "%d/%m/%Y")
bike_df4$Is.WEEKEND <- as.integer(weekdays(bike_df4$Date) %in% c("Saturday", "Sunday"))
bike_df4 <- bike_df4 %>%
  select(-Date)

```

We now have an `Is.WEEKEND` column that has the value 1 if that observation was recorded on a weekend and 0 otherwise. Notice that we also removed the `Date` column since we do not need it anymore.

Next, based on literature within the field of meteorology, we identified three key indicators of human comfort based on environmental factors:

1. Heat Index: known as the apparent temperature, and is what the temperature feels like to the human body when relative humidity is combined with the air temperature. The formula to calculate the heat index is as follows: $HI = 0.5*T + 61.0 + [(T - 68.0) * 1.2] + (RH * 0.094)$, where HI is the heat index, T represents temperature in farenheit and RH the relative humidity.
2. Wind Chill: describes what the air temperature feels like to the human skin due to the combination of cold temperatures and winds blowing (wind speed) on exposed skin. The formula to calculate wind chill is as follows: $WC = 35.74 + 0.6215T^*35.75(V^{0.16}) + 0.4275T(V^{0.16})$, where WC is wind chill, T represents temperature in Farenheit whereas V represents wind speed in miles per hour.
3. Humidex: describes how hot the weather feels to the average person, by combining the effect of heat and humidity. The humidex formula is as follows: $H = T_{air} + 0.5555\{6.11 \times \exp[5417.7530(\frac{1}{273.15} - \frac{1}{273.15+T_{dew}})] - 10\}$ where H represent the humidex, T_{air} the temperature in celcius, and T_{dew} the dew point temperature in celcius.

Therefore, we will proceed by calculating the above 3 indexes using the given columns of temperature, humidity, wind speed and dew point temperature and adding them to the dataset as new variables: `HeatIdx`, `WindChill` and `HumiDex`

```

bike_df4$HeatIdx <- 0.5 * ((bike_df4$Temperature * (9/5) + 32) + 61 + (((bike_df4$Temperature * (9/5) +
  bike_df4$WindChill <- 35.74 + 0.6215*(bike_df4$Temperature * (9/5) + 32) - 35.75*(bike_df4$Wind.Speed*2
  bike_df4$HumiDex <- bike_df4$Temperature + 0.5555*(6.11 * exp(5417.7530*(1/273.15 - 1/(273.15 + bike_df4$Humidity
summary(bike_df4)

```

```

##   Rented.Bike.Count      Hour      Temperature      Humidity
##   Min.    : 2.0      Min.    : 0.00      Min.    :-17.80      Min.    : 0.00
##   1st Qu.: 214.0     1st Qu.: 6.00      1st Qu.: 3.00      1st Qu.:42.00
##   Median  : 542.0     Median :12.00      Median : 13.50      Median :57.00
##   Mean    : 729.2     Mean   :11.51      Mean   : 12.77      Mean   :58.15
##   3rd Qu.:1084.0     3rd Qu.:18.00      3rd Qu.: 22.70      3rd Qu.:74.00
##   Max.    :3556.0     Max.    :23.00      Max.    : 39.40      Max.    :98.00
##   Wind.Speed        Visibility      Dew.Point.Temp      Solar.Radiation
##   Min.    :0.0000     Min.    : 27      Min.    :-30.600      Min.    :0.00000
##   1st Qu.: 0.900     1st Qu.: 935     1st Qu.: -5.100      1st Qu.:0.00000
##   Median  : 1.500     Median :1690      Median :  4.700      Median :0.0100
##   Mean    : 1.726     Mean   :1434      Mean   :  3.945      Mean   :0.5679
##   3rd Qu.: 2.300     3rd Qu.:2000     3rd Qu.: 15.200      3rd Qu.:0.9300
##   Max.    : 7.400     Max.    :2000      Max.    : 27.200      Max.    :3.5200
##   Seasons          Holiday          Rained          Snowed
##   Length:8465       Length:8465       Min.    :0.000000      Min.    :0.000000
##   Class  :character  Class  :character  1st Qu.:0.000000      1st Qu.:0.000000
##   Mode   :character  Mode   :character  Median :0.000000      Median :0.000000

```

```

##                                     Mean      : 0.06096   Mean      : 0.05233
##                                     3rd Qu.: 0.00000   3rd Qu.: 0.00000
##                                     Max.     : 1.00000   Max.     : 1.00000
## Is.Weekend      HeatIdx      WindChill      HumiDex
## Min.    : 0.0000  Min.    : -8.464  Min.    : -14.00  Min.    : -23.00589
## 1st Qu.: 0.0000  1st Qu.: 33.885  1st Qu.: 35.68  1st Qu.: 0.06872
## Median   : 0.0000  Median   : 54.252  Median   : 56.93  Median   : 12.87849
## Mean     : 0.2884  Mean     : 52.920  Mean     : 55.22  Mean     : 13.47299
## 3rd Qu.: 1.0000  3rd Qu.: 72.925  3rd Qu.: 75.55  3rd Qu.: 27.11391
## Max.     : 1.0000  Max.     :104.745  Max.     :110.57  Max.     : 49.61308

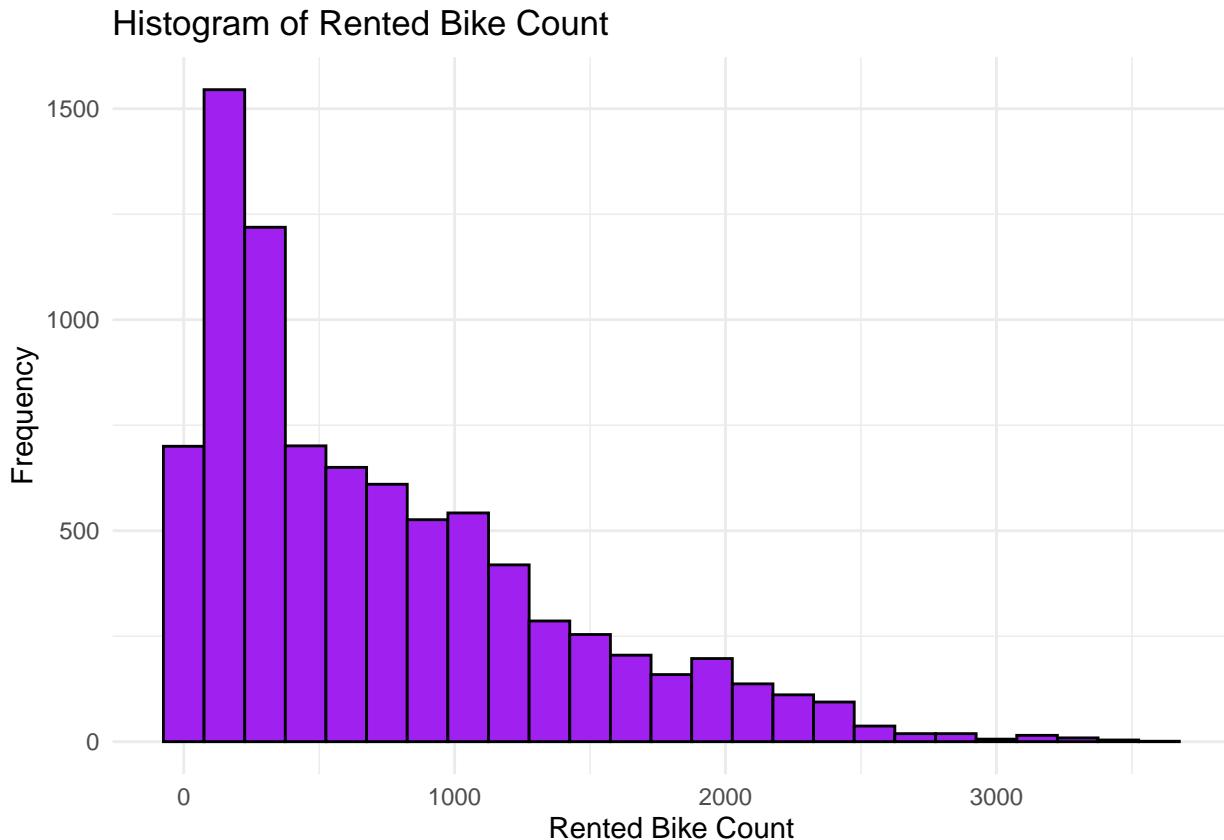
```

Outlier Detection and Removal Now that we have a set of informative features, we proceed by dealing with outliers and oddities within our dataset. Firstly, we check the distribution of our response variable, `Rented.Bike.Count`.

```

ggplot(bike_df4, aes(x = Rented.Bike.Count)) +
  geom_histogram(binwidth = 150, fill = "purple", color = "black") +
  labs(title = "Histogram of Rented Bike Count",
       x = "Rented Bike Count",
       y = "Frequency") +
  theme_minimal()

```

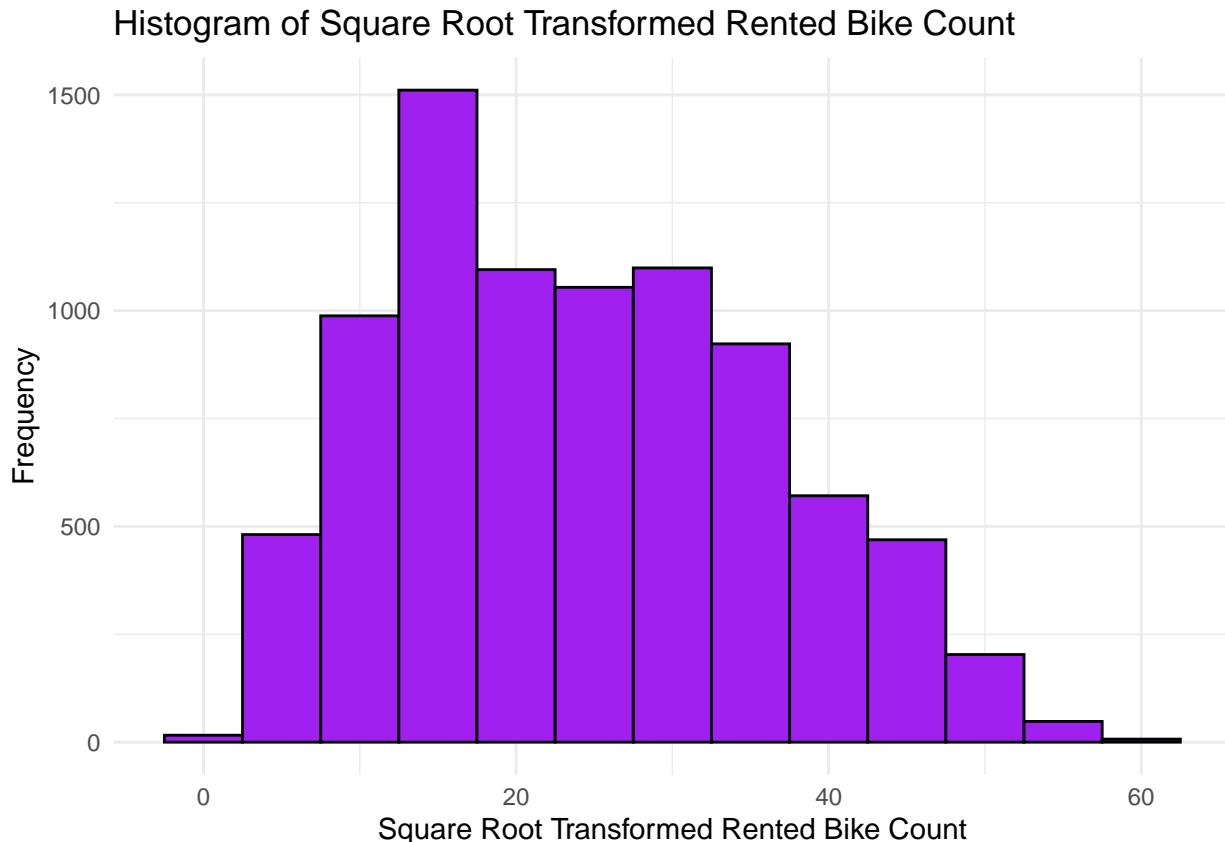


From the histogram above, we notice that the distribution of our response variable is right-skewed which is not ideal for machine learning algorithms which work better with normally distributed responses. Therefore, we attempt to improve the distribution by taking the square root of the whole column.

```

bike_df4$SqrRBC <- sqrt(bike_df4$Rented.Bike.Count)
ggplot(bike_df4, aes(x = SqrRBC)) +
  geom_histogram(binwidth = 5, fill = "purple", color = "black") +
  labs(title = "Histogram of Square Root Transformed Rented Bike Count",
       x = "Square Root Transformed Rented Bike Count",
       y = "Frequency") +
  theme_minimal()

```



Evidently, the distribution of SqrRBC looks a lot closer to a normal distribution compared to the distribution of Rented.Bike.Count, albeit not perfect. Since we are satisfied with this distribution, we can remove the variable Rented.Bike.Count from the dataset.

```

bike_df4 <- bike_df4 %>%
  select(-Rented.Bike.Count)

```

Next, we want to examine the mean and standard deviation of the square root transformed rental bike count to identify and remove outliers. For a comprehensive approach, we will calculate means and standard deviations separately for each season and excluded observations that fall outside of three standard deviations of their respective seasonal mean.

```

mean_rented_by_season <- bike_df4 %>%
  group_by(Seasons) %>%
  summarize(mean = mean(SqrRBC), sd = sd(SqrRBC))
mean_rented_by_season

```

```
## # A tibble: 4 x 3
```

```

##   Seasons  mean     sd
##   <chr>    <dbl> <dbl>
## 1 Autumn    28.4 10.8
## 2 Spring    24.6 11.8
## 3 Summer    30.1 11.3
## 4 Winter    14.2  5.00

bike_df5 <- bike_df4 %>%
  left_join(mean_rented_by_season, by = "Seasons")

bike_df6 <- bike_df5 %>%
  filter(
    SqrRBC >= (mean - 3 * sd) &
    SqrRBC <= (mean + 3 * sd)
  )
dim(bike_df6)

## [1] 8460   18

```

Consequently, we removed 5 outliers from the dataset.

Categorical Variables Next, we examine the categorical variables within the dataset:

```



```

We see that each season has an approximately similar number of observations, whereas there is significantly more observations on days which are not holidays compared to those with holiday. This imbalance is a natural outcome where there is significantly more non-holidays than holidays in a year. I will proceed by performing one-hot encoding on both these columns and the columns rained and snowed.

```

dummy_var <- dummyVars(~ Seasons + Holiday, data = bike_df6)
encoded_df <- predict(dummy_var, newdata = bike_df6)
encoded_df2 <- as.data.frame(encoded_df)
bike_df7 <- cbind(bike_df6[, !(names(bike_df6) %in% c("Seasons", "Holiday"))], encoded_df2)
head(bike_df7)

```

```

##   Hour Temperature Humidity Wind.Speed Visibility Dew.Point.Temp
## 1    0      -5.2       37      2.2      2000      -17.6
## 2    1      -5.5       38      0.8      2000      -17.6
## 3    2      -6.0       39      1.0      2000      -17.7

```

```

## 4     3      -6.2      40      0.9      2000      -17.6
## 5     4      -6.0      36      2.3      2000      -18.6
## 6     5      -6.4      37      1.5      2000      -18.7
##   Solar.Radiation Rained Snowed Is.Weekend HeatIdx WindChill    HumiDex
## 1             0       0       0           0 16.343 16.16761 -9.889089
## 2             0       0       0           0 15.796 20.60613 -10.189089
## 3             0       0       0           0 14.853 18.55981 -10.696245
## 4             0       0       0           0 14.504 18.67138 -10.889089
## 5             0       0       0           0 14.712 14.23246 -10.758285
## 6             0       0       0           0 13.967 15.70413 -11.164921
##   SqrRBC SeasonsAutumn SeasonsSpring SeasonsSummer SeasonsWinter
## 1 15.937377          0           0           0           1
## 2 14.282857          0           0           0           1
## 3 13.152946          0           0           0           1
## 4 10.344080          0           0           0           1
## 5 8.831761           0           0           0           1
## 6 10.000000          0           0           0           1
##   HolidayHoliday HolidayNo Holiday
## 1             0           1
## 2             0           1
## 3             0           1
## 4             0           1
## 5             0           1
## 6             0           1

bike_df7$Hour <- as.factor(bike_df7$Hour)
bike_df8 <- bike_df7[, !(names(bike_df7) %in% c("mean", "sd"))]
head(bike_df8)

##   Hour Temperature Humidity Wind.Speed Visibility Dew.Point.Temp
## 1   0      -5.2      37      2.2      2000      -17.6
## 2   1      -5.5      38      0.8      2000      -17.6
## 3   2      -6.0      39      1.0      2000      -17.7
## 4   3      -6.2      40      0.9      2000      -17.6
## 5   4      -6.0      36      2.3      2000      -18.6
## 6   5      -6.4      37      1.5      2000      -18.7
##   Solar.Radiation Rained Snowed Is.Weekend HeatIdx WindChill    HumiDex
## 1             0       0       0           0 16.343 16.16761 -9.889089
## 2             0       0       0           0 15.796 20.60613 -10.189089
## 3             0       0       0           0 14.853 18.55981 -10.696245
## 4             0       0       0           0 14.504 18.67138 -10.889089
## 5             0       0       0           0 14.712 14.23246 -10.758285
## 6             0       0       0           0 13.967 15.70413 -11.164921
##   SqrRBC SeasonsAutumn SeasonsSpring SeasonsSummer SeasonsWinter
## 1 15.937377          0           0           0           1
## 2 14.282857          0           0           0           1
## 3 13.152946          0           0           0           1
## 4 10.344080          0           0           0           1
## 5 8.831761           0           0           0           1
## 6 10.000000          0           0           0           1
##   HolidayHoliday HolidayNo Holiday
## 1             0           1
## 2             0           1
## 3             0           1

```

```

## 4          0          1
## 5          0          1
## 6          0          1

```

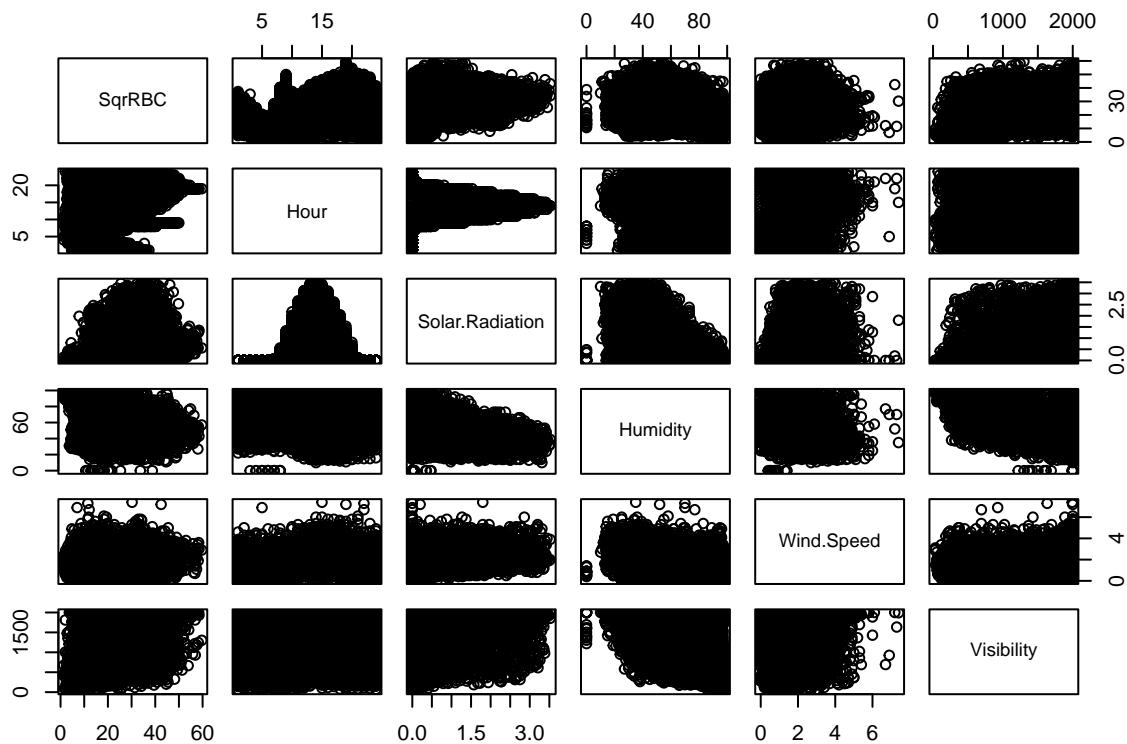
Numerical Variables Next, we then examine our numerical variables and visualize each of them against SqRBC:

```

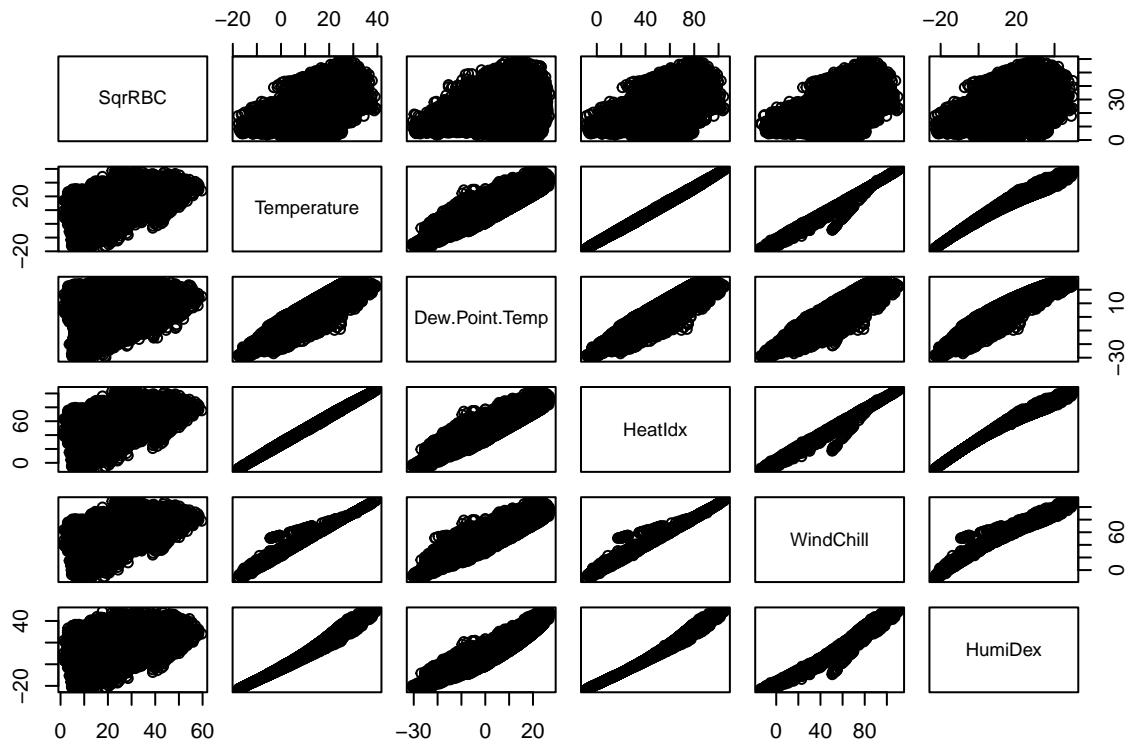
visual_df <- bike_df8 %>%
  select(SqrRBC, Hour, Solar.Radiation, Humidity, Wind.Speed, Visibility)
visual_df2 <- bike_df8 %>%
  select(SqrRBC, Temperature, Dew.Point.Temp, HeatIdx, WindChill, HumiDex)

pairs(visual_df)

```



```
pairs(visual_df2)
```

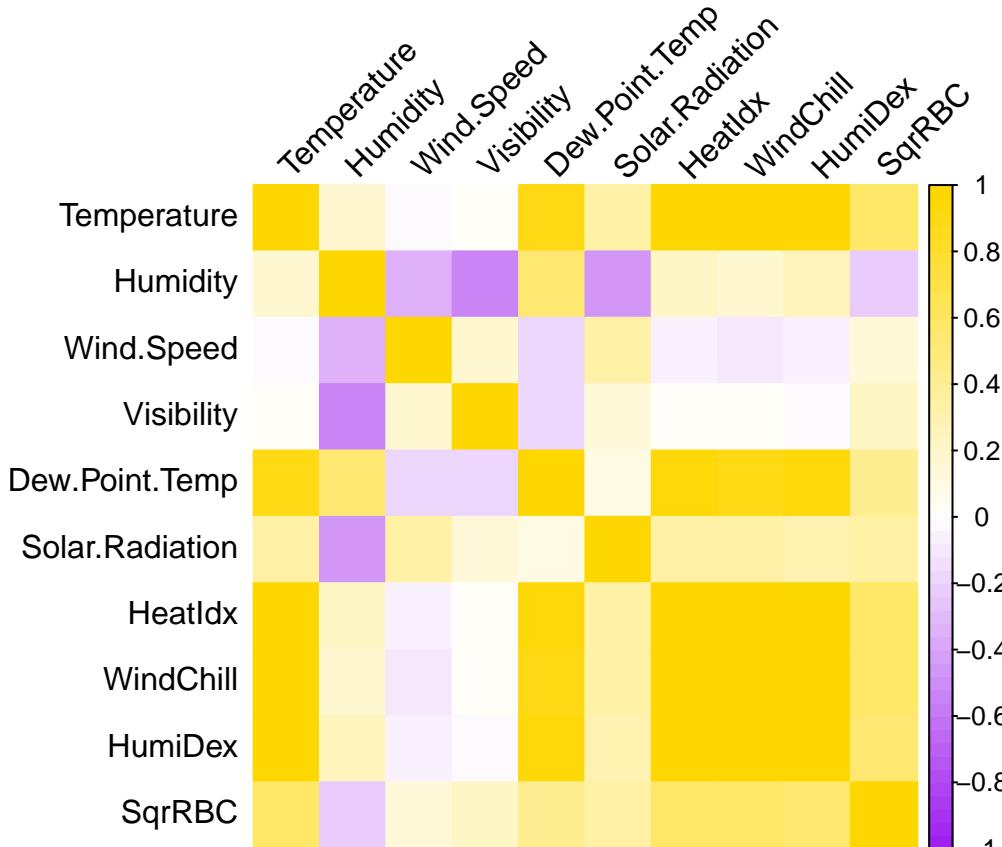


From the

plots above, we see that our square root transformed rented bike count has a small peak at 8-9am and is the highest at 6-7pm which coincides with times the people go to and leave work respectively. Next, Temperature, Dew.Point.Temperature, HeatIndex, WindChill and HumiDex all seem to have some positive correlation with SqrRBC.

Final Variable Selection However, the aforementioned five features excluding SqrRBC also seem to have a positive correlation between each other. Therefore, we need to perform correlation analysis as we want to avoid including highly correlated features within the dataset as this may skew our model.

```
cor_matrix <- cor(bike_df8[, c(2:7,11:14)])
corrplot(cor_matrix, method = "color", type = "full", col = colorRampPalette(c("purple", "white", "gold")))
```



From the heatmap, it can evidently be seen that the features `Temperature`, `Dew.Point.Temperature`, `HeatIndex`, `WindChill` and `HumiDex` are highly correlated with each other. This is to be expected since the latter 3 features were all computed using `Temperature`, with one of them also computed using `Dew.Point.Temperature`. Thus, those two variables should be removed. Although `HeatIndex`, `WindChill` and `HumiDex` are still highly correlated, they may improve model performance, which is why we did not remove these predictors. Instead, we will also remove `Wind.Speed` and `Humidity` since those features were also used to calculate `WindChill` and `HeatIdx` respectively.

Next, to remove dependency within the dataset, we also want to remove `SeasonsWinter`, `HolidayNo`, `Holiday`, `RainedYes` and `SnowedYes` since one one-hot encoded column for each variable must be removed.

From the heatmap, it is evident that the features `Temperature`, `Dew.Point.Temperature`, `HeatIndex`, `WindChill` and `HumiDex` are highly correlated with each other. This is to be expected since `HeatIndex`, `WindChill` and `HumiDex` are all computed using `Temperature`. In addition, `HumiDex` is also computed using `Dew.Point.Temperature`. Given this high correlation, we will remove `Temperature` and `Dew.Point.Temperature` from the dataset to reduce multicollinearity. Although `HeatIndex`, `WindChill` and `HumiDex` remain highly correlated, we will retain these features since they may still contribute valuable information.

Next, we will also remove `Wind.Speed` and `Humidity` since `WindChill` and `HeatIndex` are dependant on them respectively.

Finally, we will remove one column from each one-hot encoded categorical variable: `SeasonsWinter` and `HolidayNo` `Holiday` to avoid multicollinearity.

```

bike_df9 <- bike_df8[, !(names(bike_df8) %in% c("Temperature", "Dew.Point.Temp", "Wind.Speed", "Humidity"))]
head(bike_df9)

##   Hour Visibility Solar.Radiation Rained Snowed Is.Weekend HeatIdx WindChill
## 1     0        2000          0       0       0         0  16.343  16.16761
## 2     1        2000          0       0       0         0  15.796  20.60613
## 3     2        2000          0       0       0         0  14.853  18.55981
## 4     3        2000          0       0       0         0  14.504  18.67138
## 5     4        2000          0       0       0         0  14.712  14.23246
## 6     5        2000          0       0       0         0  13.967  15.70413
##   HumiDex    SqrRBC SeasonsAutumn SeasonsSpring SeasonsSummer HolidayHoliday
## 1 -9.889089 15.937377          0          0          0         0
## 2 -10.189089 14.282857          0          0          0         0
## 3 -10.696245 13.152946          0          0          0         0
## 4 -10.889089 10.344080          0          0          0         0
## 5 -10.758285  8.831761          0          0          0         0
## 6 -11.164921 10.000000          0          0          0         0

```

Now, we are ready to train a model using the above dataset.

Model Building In order to train a machine learning model, we first need a train set and a test set. In this project, we are using a 80-20 split where 80% of the observations from our dataset will be randomly sampled as our train set, whereas the remaining 20% will serve as the set.

```

set.seed(435)
train_size <- floor(0.8 * nrow(bike_df9))
train <- sample(1:nrow(bike_df9), train_size)
test <- (-train)
train_data <- bike_df9[train, ]
test_data <- bike_df9[test, ]

```

Multiple Linear Regression We chose to predict our response variable, `SqrRBC` using multiple linear regression (MLR) since it is one of the most simple and easily interpretable models which is invaluable when presenting findings to clients. To build the model, we simply used the `lm` function and all the features within the train set since we prepped the dataset to only include features that are independant and we believe will be useful in predicting hourly count of rented bikes.

```

lm1 <- lm(SqrRBC ~ ., data=train_data)
summary(lm1)

```

```

##
## Call:
## lm(formula = SqrRBC ~ ., data = train_data)
##
## Residuals:
##      Min      1Q      Median      3Q      Max 
## -29.0687 -3.4202   0.1553   3.5856  24.5825 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -1.318e+01  1.140e+00 -11.570 < 2e-16 ***
## Hour1       -2.328e+00  4.846e-01  -4.804 1.59e-06 ***

```

```

## Hour2      -5.258e+00  4.887e-01 -10.758 < 2e-16 ***
## Hour3      -7.448e+00  4.921e-01 -15.134 < 2e-16 ***
## Hour4      -1.057e+01  4.916e-01 -21.498 < 2e-16 ***
## Hour5      -1.043e+01  4.879e-01 -21.379 < 2e-16 ***
## Hour6      -4.686e+00  4.922e-01 -9.520 < 2e-16 ***
## Hour7      1.300e+00  4.846e-01  2.683 0.007322 **
## Hour8      7.453e+00  4.960e-01  15.025 < 2e-16 ***
## Hour9      1.414e+00  5.022e-01  2.815 0.004891 **
## Hour10     -2.538e+00  5.238e-01 -4.845 1.29e-06 ***
## Hour11     -2.411e+00  5.440e-01 -4.432 9.49e-06 ***
## Hour12     -1.290e+00  5.581e-01 -2.311 0.020852 *
## Hour13     -1.552e+00  5.642e-01 -2.752 0.005947 **
## Hour14     -1.237e+00  5.546e-01 -2.231 0.025710 *
## Hour15     -8.138e-02  5.396e-01 -0.151 0.880132
## Hour16     1.586e+00  5.227e-01  3.035 0.002416 **
## Hour17     5.623e+00  4.978e-01 11.294 < 2e-16 ***
## Hour18     1.169e+01  4.974e-01 23.503 < 2e-16 ***
## Hour19     8.139e+00  4.925e-01 16.526 < 2e-16 ***
## Hour20     6.628e+00  4.916e-01 13.484 < 2e-16 ***
## Hour21     7.202e+00  4.906e-01 14.680 < 2e-16 ***
## Hour22     5.539e+00  4.872e-01 11.368 < 2e-16 ***
## Hour23     1.879e+00  4.876e-01  3.854 0.000117 ***
## Visibility   1.397e-03  1.256e-04 11.127 < 2e-16 ***
## Solar.Radiation 9.797e-01  1.640e-01  5.974 2.43e-09 ***
## Rained      -1.375e+01  3.174e-01 -43.329 < 2e-16 ***
## Snowed       2.880e-01  3.405e-01  0.846 0.397656
## Is.Weekend   -1.997e+00  1.547e-01 -12.911 < 2e-16 ***
## HeatIdx      7.802e-01  4.811e-02 16.218 < 2e-16 ***
## WindChill    7.981e-02  2.703e-02  2.953 0.003157 **
## HumiDex      -9.919e-01  4.893e-02 -20.273 < 2e-16 ***
## SeasonsAutumn 5.984e+00  3.043e-01 19.661 < 2e-16 ***
## SeasonsSpring 3.384e+00  2.955e-01 11.454 < 2e-16 ***
## SeasonsSummer 5.623e+00  4.302e-01 13.070 < 2e-16 ***
## HolidayHoliday -2.761e+00  3.224e-01 -8.566 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.708 on 6732 degrees of freedom
## Multiple R-squared:  0.769, Adjusted R-squared:  0.7678
## F-statistic: 640.1 on 35 and 6732 DF, p-value: < 2.2e-16

```

The summary of our MLR model boasts a multiple R-squared of 0.769, indicating that 76.9% of the variability in SqrRBC is explained by the model, which suggests a strong fit. Additionally, most predictors are statistically significant in predicting SqrRBC, with p-values less than 0.05. However, `Snowed` is an exception, with a p-value of 0.397656. This implies that the presence of snow might not significantly influence bike rental decisions of consumer. Therefore, to determine whether we should remove `Snowed` from the model, we will perform an ANOVA test as follows:

```

lm2 <- lm(SqrRBC ~ Hour + Visibility + Solar.Radiation + Rained + Is.Weekend + HeatIdx + WindChill + HumiDex + SeasonsAutumn + SeasonsSpring + SeasonsSummer + HolidayHoliday)

summary(lm2)

##

```

```

## Call:
## lm(formula = SqrRBC ~ Hour + Visibility + Solar.Radiation + Rained +
##      Is.Weekend + HeatIdx + WindChill + HumiDex + SeasonsAutumn +
##      SeasonsSpring + SeasonsSummer + HolidayHoliday, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -29.1013  -3.4264   0.1501   3.5971  24.5679 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -1.303e+01  1.125e+00 -11.583 < 2e-16 ***
## Hour1        -2.328e+00  4.846e-01  -4.804 1.59e-06 ***
## Hour2        -5.258e+00  4.887e-01  -10.759 < 2e-16 ***
## Hour3        -7.447e+00  4.921e-01  -15.132 < 2e-16 ***
## Hour4        -1.057e+01  4.915e-01  -21.498 < 2e-16 ***
## Hour5        -1.043e+01  4.879e-01  -21.386 < 2e-16 ***
## Hour6        -4.690e+00  4.922e-01  -9.529 < 2e-16 ***
## Hour7         1.300e+00  4.846e-01   2.683 0.007314 **  
## Hour8         7.450e+00  4.960e-01   15.021 < 2e-16 ***
## Hour9         1.406e+00  5.021e-01   2.801 0.005115 **  
## Hour10        -2.545e+00  5.237e-01  -4.860 1.20e-06 ***
## Hour11        -2.416e+00  5.440e-01  -4.442 9.06e-06 *** 
## Hour12        -1.296e+00  5.581e-01  -2.322 0.020283 *   
## Hour13        -1.559e+00  5.641e-01  -2.764 0.005722 **  
## Hour14        -1.246e+00  5.545e-01  -2.246 0.024710 *  
## Hour15        -8.576e-02  5.396e-01  -0.159 0.873729  
## Hour16         1.581e+00  5.227e-01   3.024 0.002501 **  
## Hour17         5.623e+00  4.978e-01  11.296 < 2e-16 ***
## Hour18         1.169e+01  4.974e-01  23.511 < 2e-16 ***
## Hour19         8.144e+00  4.924e-01  16.538 < 2e-16 ***
## Hour20         6.632e+00  4.915e-01  13.492 < 2e-16 ***
## Hour21         7.200e+00  4.906e-01  14.676 < 2e-16 ***
## Hour22         5.536e+00  4.872e-01  11.364 < 2e-16 ***
## Hour23         1.878e+00  4.876e-01   3.852 0.000118 *** 
## Visibility     1.386e-03  1.249e-04  11.100 < 2e-16 *** 
## Solar.Radiation 9.881e-01  1.637e-01   6.037 1.65e-09 *** 
## Rained        -1.375e+01  3.174e-01  -43.337 < 2e-16 *** 
## Is.Weekend    -2.001e+00  1.546e-01  -12.945 < 2e-16 *** 
## HeatIdx        7.785e-01  4.806e-02  16.197 < 2e-16 *** 
## WindChill     7.871e-02  2.699e-02   2.916 0.003558 **  
## HumiDex        -9.889e-01  4.880e-02  -20.265 < 2e-16 *** 
## SeasonsAutumn  5.972e+00  3.040e-01   19.644 < 2e-16 *** 
## SeasonsSpring  3.359e+00  2.939e-01   11.428 < 2e-16 *** 
## SeasonsSummer  5.614e+00  4.301e-01   13.053 < 2e-16 *** 
## HolidayHoliday -2.765e+00  3.223e-01  -8.577 < 2e-16 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.708 on 6733 degrees of freedom
## Multiple R-squared:  0.7689, Adjusted R-squared:  0.7678 
## F-statistic:  659 on 34 and 6733 DF,  p-value: < 2.2e-16

```

```

anova(lm1, lm2)

## Analysis of Variance Table
##
## Model 1: SqrRBC ~ Hour + Visibility + Solar.Radiation + Rained + Snowed +
##           Is.Weekend + HeatIdx + WindChill + HumiDex + SeasonsAutumn +
##           SeasonsSpring + SeasonsSummer + HolidayHoliday
## Model 2: SqrRBC ~ Hour + Visibility + Solar.Radiation + Rained + Is.Weekend +
##           HeatIdx + WindChill + HumiDex + SeasonsAutumn + SeasonsSpring +
##           SeasonsSummer + HolidayHoliday
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
## 1     6732 219323
## 2     6733 219346 -1    -23.31 0.7155 0.3977

```

From the anova test, the p-value of 0.3977 indicates that the difference in the fit of the two models is not statistically significant. This means that the predictor `Snowed` does not provide meaningful predictive power to the model, and can be safely removed from the model without compromising the model's performance. This is further supported by identical adjusted R-squared values for both the first and second model, indicating that the model performs just as well without `Snowed`. Now that we have obtained our model, let us evaluate its performance by calculating its mean squared error (MSE). Note that I use MSE here instead of root mean squared error since it will have the same units as `Rented.Bike.Count` which is what we are ultimately predicting, which facilitates interpretation.

```

predictions <- predict(lm2, newdata = test_data)
mse <- mean((test_data$SqrRBC - predictions)^2)

```

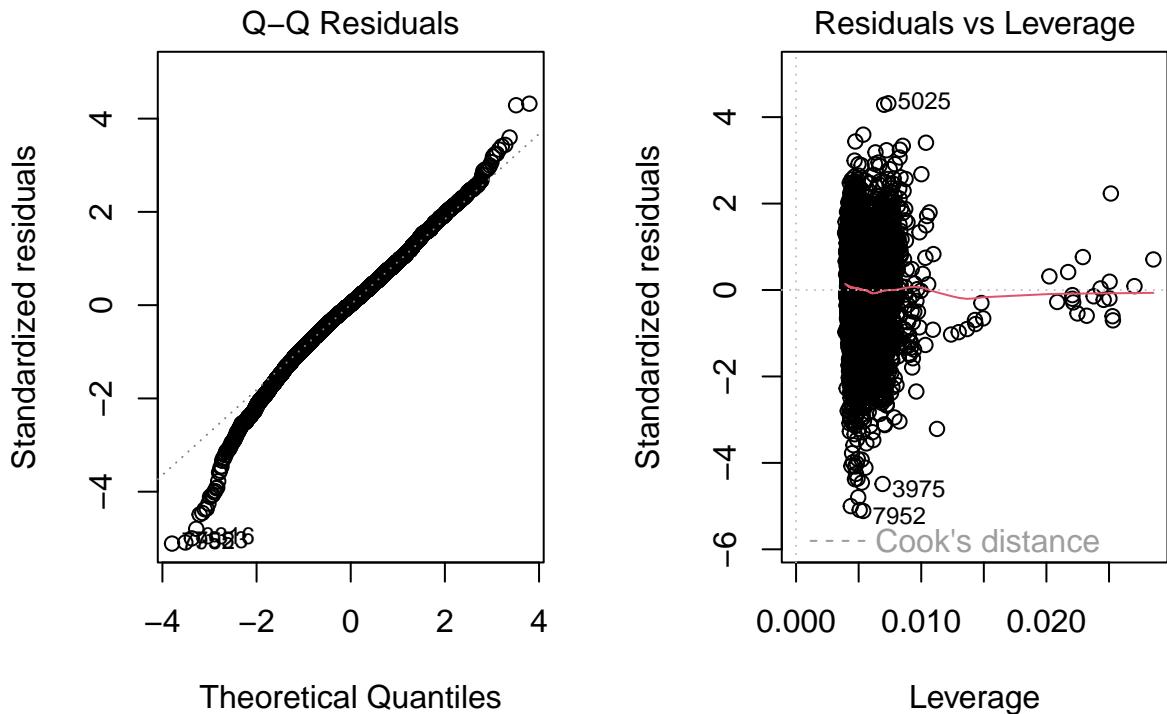
We found that our model has an MSE of 32.3268, which seems reasonable given that rented bike count in our dataset ranges from 2 to 3,556, with a median of 542. However, to further improve our model, we will examine its residual plots to check for behaviour that could indicate areas where the model may not be performing well.

```

# Generate the plots for the linear model
par(mfrow = c(1, 2))

plot(lm2, which = c(2,5))

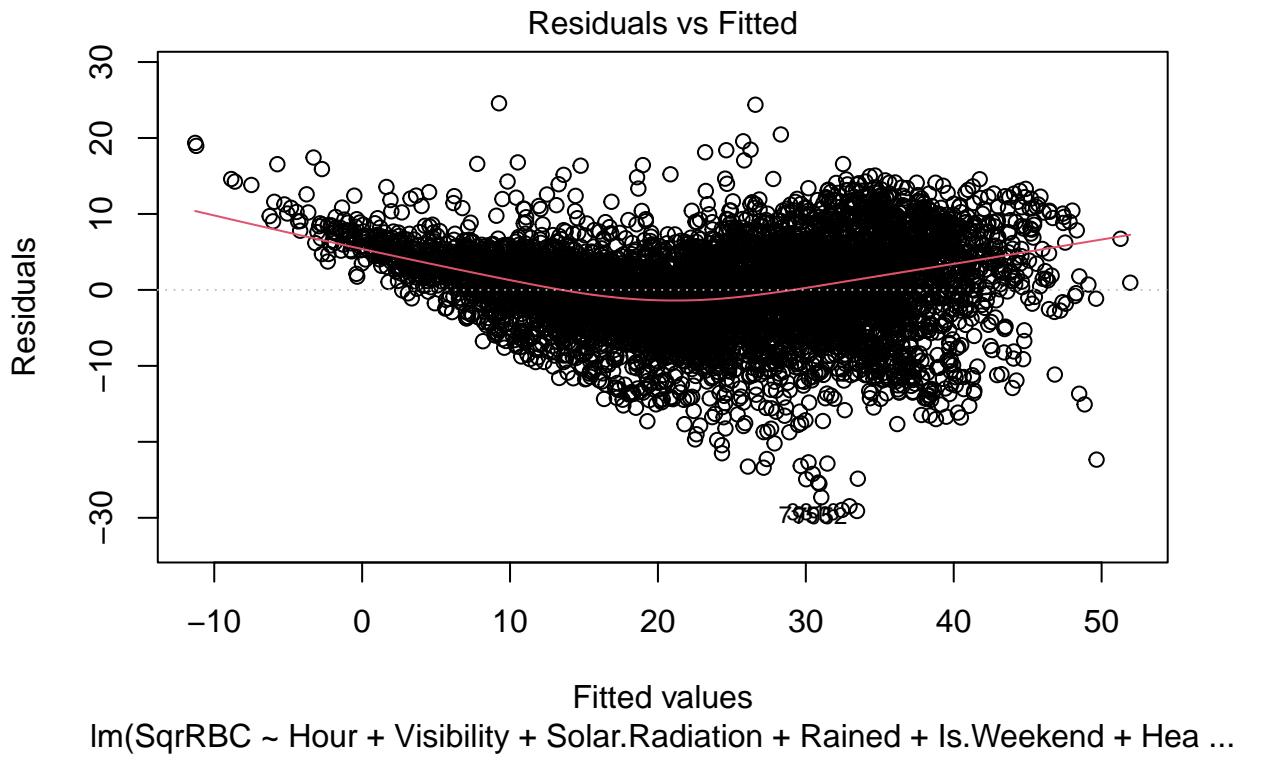
```



```
par(mfrow = c(1,1))
```

Firstly, most of the residuals generally fall on the Q-Q line, indicating that the residuals are generally normally distributed. However, although there is only a slight deviation on the right tail, there is a noticeable deviation on the left tail, which suggests the presence of extreme negative residuals which may be adversely impacting the performance of the model. Similarly, the residuals vs. leverage plot shows no high-leverage points, but it does reveal several outliers with extreme negative residuals. These observations highlight the presence of outliers that could be influencing the model. To further investigate these outliers, we will examine the residuals plot to gain additional insights.

```
plot(lm2, which = 1)
```



$\text{lm}(\text{SqrRBC} \sim \text{Hour} + \text{Visibility} + \text{Solar.Radiation} + \text{Rained} + \text{Is.Weekend} + \text{Hea} \dots)$

In the residuals vs fitted plot, we immediately notice a distinct quadratic shape, along with many outliers with heavily negative residuals. This quadratic pattern suggests that there is a relationship between the predictors is not being fully captured by the linear model. To address this, we decided to add natural splines to the linear model to help to capture non-linear relationships and improve model performance.

Referring back to the pair plots of each of the numerical predictors against SqrRBC, we notice that Solar.Radiation and Visibility exhibit non-linear relationships with SqrRBC. Therefore, we will incorporate splines for these predictors. We will also add splines on HeatIdx, WindChill and HumiDex because, while they show some linear relationship with SqrRBC, it is not particularly strong.

```
spline_model <- lm(SqrRBC ~ Hour + ns(Visibility, df=2) + ns(Solar.Radiation, df = 5) + ns(HeatIdx, df=7) + ns(WindChill, df = 6) + ns(HumiDex, df = 7) + SeasonsAutumn + SeasonsSpring + SeasonsSummer + HolidayHoliday + Rained, data = train_data)
#summary(spline_model)
```

Note that the degrees of freedom for each predictor was chosen through trial and error to maximize the value of the Adjusted R-Squared.

```
summary(spline_model)
```

```
##
## Call:
## lm(formula = SqrRBC ~ Hour + ns(Visibility, df = 2) + ns(Solar.Radiation,
##       df = 5) + ns(HeatIdx, df = 7) + ns(WindChill, df = 6) + ns(HumiDex,
##       df = 7) + SeasonsAutumn + SeasonsSpring + SeasonsSummer +
##       HolidayHoliday + Rained, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -35.552  -10.000   -0.000  10.000  35.552 
```

```

## -31.718 -2.770 0.218 3.060 20.076
##
## Coefficients:
## Estimate Std. Error t value Pr(>|t|)
## (Intercept) 9.6020 0.9124 10.524 < 2e-16 ***
## Hour1 -2.3981 0.4345 -5.519 3.54e-08 ***
## Hour2 -5.3200 0.4383 -12.137 < 2e-16 ***
## Hour3 -7.5637 0.4416 -17.129 < 2e-16 ***
## Hour4 -10.4904 0.4415 -23.761 < 2e-16 ***
## Hour5 -10.3272 0.4386 -23.547 < 2e-16 ***
## Hour6 -5.3339 0.4486 -11.891 < 2e-16 ***
## Hour7 -1.0609 0.4644 -2.284 0.022385 *
## Hour8 2.5854 0.5184 4.987 6.27e-07 ***
## Hour9 -5.2195 0.5690 -9.173 < 2e-16 ***
## Hour10 -9.3773 0.5943 -15.779 < 2e-16 ***
## Hour11 -8.8796 0.6074 -14.619 < 2e-16 ***
## Hour12 -7.3582 0.6167 -11.932 < 2e-16 ***
## Hour13 -7.3147 0.6192 -11.813 < 2e-16 ***
## Hour14 -7.0958 0.6113 -11.609 < 2e-16 ***
## Hour15 -6.0160 0.5993 -10.038 < 2e-16 ***
## Hour16 -4.5610 0.5862 -7.781 8.29e-15 ***
## Hour17 -0.2768 0.5596 -0.495 0.620829
## Hour18 6.9606 0.5174 13.452 < 2e-16 ***
## Hour19 5.2858 0.4717 11.206 < 2e-16 ***
## Hour20 5.6011 0.4500 12.447 < 2e-16 ***
## Hour21 6.9676 0.4403 15.824 < 2e-16 ***
## Hour22 5.4424 0.4369 12.456 < 2e-16 ***
## Hour23 1.7394 0.4372 3.979 7.00e-05 ***
## ns(Visibility, df = 2)1 2.3352 0.6640 3.517 0.000440 ***
## ns(Visibility, df = 2)2 -0.4117 0.1985 -2.074 0.038103 *
## ns(Solar.Radiation, df = 5)1 3.8813 0.4414 8.792 < 2e-16 ***
## ns(Solar.Radiation, df = 5)2 8.6634 0.4996 17.341 < 2e-16 ***
## ns(Solar.Radiation, df = 5)3 6.9769 0.6284 11.102 < 2e-16 ***
## ns(Solar.Radiation, df = 5)4 9.9940 0.8124 12.302 < 2e-16 ***
## ns(Solar.Radiation, df = 5)5 4.5304 0.7711 5.875 4.42e-09 ***
## ns(HeatIdx, df = 7)1 74.1549 35.5714 2.085 0.037136 *
## ns(HeatIdx, df = 7)2 101.1374 34.9880 2.891 0.003857 **
## ns(HeatIdx, df = 7)3 122.3391 35.3271 3.463 0.000537 ***
## ns(HeatIdx, df = 7)4 140.5975 35.3880 3.973 7.17e-05 ***
## ns(HeatIdx, df = 7)5 148.8599 24.4101 6.098 1.13e-09 ***
## ns(HeatIdx, df = 7)6 267.0809 64.8379 4.119 3.85e-05 ***
## ns(HeatIdx, df = 7)7 209.8050 27.8843 7.524 6.00e-14 ***
## ns(WindChill, df = 6)1 13.3751 2.5321 5.282 1.32e-07 ***
## ns(WindChill, df = 6)2 16.1764 2.7451 5.893 3.98e-09 ***
## ns(WindChill, df = 6)3 12.3438 2.9924 4.125 3.75e-05 ***
## ns(WindChill, df = 6)4 10.4672 4.5222 2.315 0.020665 *
## ns(WindChill, df = 6)5 -28.2256 11.5692 -2.440 0.014725 *
## ns(WindChill, df = 6)6 -73.4801 17.4373 -4.214 2.54e-05 ***
## ns(HumiDex, df = 7)1 -81.4263 34.7810 -2.341 0.019255 *
## ns(HumiDex, df = 7)2 -107.6122 34.4551 -3.123 0.001796 **
## ns(HumiDex, df = 7)3 -125.5769 34.6492 -3.624 0.000292 ***
## ns(HumiDex, df = 7)4 -137.0446 34.6733 -3.952 7.81e-05 ***
## ns(HumiDex, df = 7)5 -138.8575 23.7615 -5.844 5.34e-09 ***
## ns(HumiDex, df = 7)6 -225.0712 62.9611 -3.575 0.000353 ***

```

```

## ns(HumiDex, df = 7)7      -131.6092   19.1044  -6.889 6.13e-12 ***
## SeasonsAutumn             6.5914     0.3236   20.372 < 2e-16 ***
## SeasonsSpring              3.3294     0.3129   10.639 < 2e-16 ***
## SeasonsSummer              5.5328     0.4139   13.367 < 2e-16 ***
## HolidayHoliday             -2.1273    0.2949   -7.215 6.00e-13 ***
## Rained                      -12.4175   0.3004   -41.340 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.115 on 6712 degrees of freedom
## Multiple R-squared:  0.815, Adjusted R-squared:  0.8135
## F-statistic: 537.6 on 55 and 6712 DF, p-value: < 2.2e-16

```

From the summary of our natural spline model, we can observe that with the addition of natural splines, our multiple R-squared value increased from 0.769 to 0.815, which is a 5.98% increase.

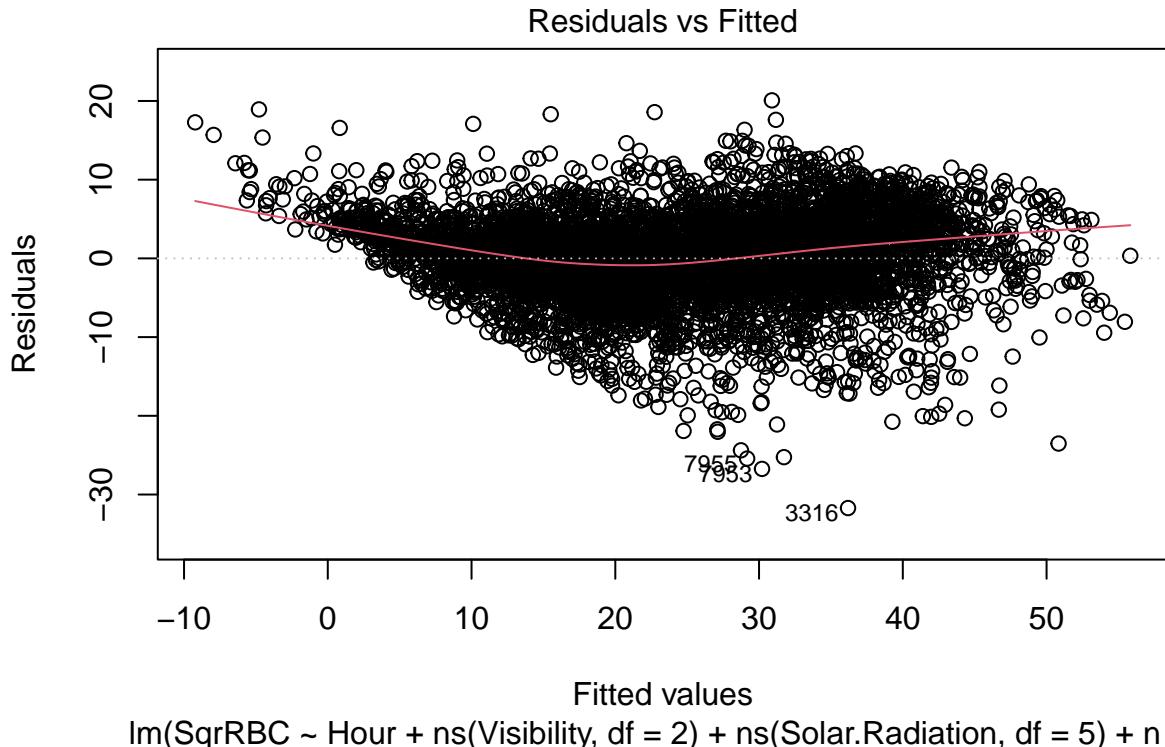
```

predictions <- predict(spline_model, newdata = test_data)
mse2 <- mean((test_data$SqrRBC - predictions)^2)

```

We also achieved a lower MSE of `round(mse2,4)`, which is a decrease of -7.5221 from our previous MSE. Despite that, the residuals vs fitted plot of the spline model shown below indicates that there are still some relationships between predictors that remain uncaptured by the natural spline model.

```
plot(spline_model, which=1)
```



Given these residual patterns, we decided that it is more appropriate to train a more flexible model that is capable of capturing these non-linear relationships more reliably. This calls for Random Forests since they are highly capable in capturing non-linear relationships and interactions between predictors without the need for explicit modeling.

Hence, we will proceed with a Random Forest model, which should help address the limitations we have observed with the spline model, and lead to improvements in predictive performance.

Random Forests First of all, though we will be using the same data points for both the train and test set, we will be re-introducing some of the predictors that were previously removed due to dependencies. This is because since random forests are ensemble methods that builds multiple decision trees and aggregates their predictions, they are robust to multidisciplinary and are able to perform well with dependent variables which allows them to learn more patterns. Therefore, the predictors used for the random forest model will be as follows: Hour, Temperature, Humidity, Wind.Speed, Visibility, Dew.Point.Temp, Solar.Radiation, Rained, Snowed, Is.Weekend, HeatIdx, WindChill, HumiDex, SeasonsAutumn, SeasonsSpring, SeasonsSummer and HolidayHoliday.

```
set.seed(435)

bike_df10 <- bike_df8[, c(-18,-20)]

train_data2 <- bike_df10[train, ]
test_data2 <- bike_df10[test, ]

rf_model <- randomForest(SqrRBC ~ .,
                           data = train_data2,
                           ntree = 501)
```

Now that our model is trained, we will test its predictive power by calculating its MSE.

```
predictions <- predict(rf_model, newdata = test_data2)

actuals <- test_data2$SqrRBC
mse3 <- mean((predictions - actuals)^2)
print(mse3)

## [1] 11.91972
```

As seen above, the MSE of the random forest model is 11.9197 which is lower than the MSE of the spline regression model which was 24.8046 which is a decrease of -51.95 %.

However, it is important to recognize that our Random Forest model currently includes 17 predictors, which may be too many to ensure ease of use for our clients. To streamline the model while maintaining its predictive power, we aim to remove some predictors. To do so, we will evaluate the importance of each predictor in the model as follows:

```
importance(rf_model)

##           IncNodePurity
## Hour        254285.0142
## Temperature 123042.9559
## Humidity    95142.5309
## Wind.Speed   16551.9637
## Visibility   23201.5001
## Dew.Point.Temp 34720.5095
## Solar.Radiation 41251.1313
```

```

## Rained          60740.0882
## Snowed          862.2766
## Is.WEEKEND     12199.9062
## HeatIdx         90603.0163
## WindChill       98875.0787
## HumiDex         55873.1058
## SeasonsAutumn   20471.3502
## SeasonsSpring    4205.1826
## SeasonsSummer    3176.3765
## HolidayHoliday   2784.4792

```

From the table above, we can surmise that the predictors `Snowed` and `HolidayHoliday` have low importance in the decision-making process of the Random Forest model. While `SeasonsSpring` and `SeasonsSummer` also show lower importance, we decided to keep them because `SeasonsAutumn` has a high importance score, and all three only require a single input from the client (type of season). Therefore, we will proceed by removing `Snowed` and `HolidayHoliday` from our dataset and retrain the Random Forest model.

```

set.seed(435)

bike_df11 <- bike_df10[, c(-9,-18)]

train_data2 <- bike_df11[train, ]
test_data2 <- bike_df11[test, ]

rf_model2 <- randomForest(SqrRBC ~ .,
                           data = train_data2,
                           ntree = 501)

predictions <- predict(rf_model2, newdata = test_data2)
actuals <- test_data2$SqrRBC

mse4 <- mean((predictions - actuals)^2)
print(mse4)

## [1] 12.10591

```

After dropping the two predictors, we notice a slight increase in the MSE which increased from 11.9197 to 12.1059. Despite this minor loss in predictive accuracy, the trade-off is beneficial as it simplifies the model, and enhances its usability and interpretability for clients.

Next, we can further improve our Random Forest model by tuning the `mtry` value instead of using the default value by using `tuneRF` as shown below:

```

set.seed(435)

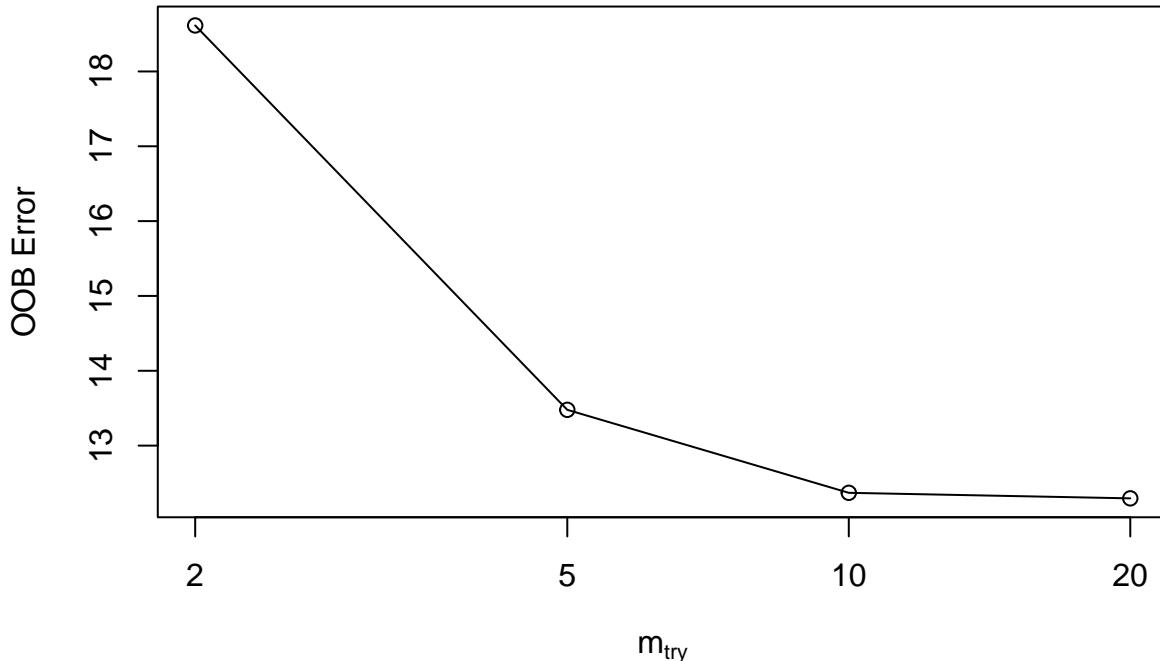
tuneMt <- tuneRF(train_data2[,-which(names(train_data2) == "SqrRBC")],
                  train_data2$SqrRBC,
                  stepFactor = 0.5,
                  plot = TRUE,
                  ntreeTry = 501,
                  trace = TRUE,
                  improve = 0.05)

```

```

## mtry = 5  OOB error = 13.47796
## Searching left ...
## mtry = 10      OOB error = 12.36911
## 0.08227102 0.05
## mtry = 20      OOB error = 12.29514
## 0.005980782 0.05
## Searching right ...
## mtry = 2      OOB error = 18.6146
## -0.5049259 0.05

```



```
opt_mtry <- tuneMt[which.min(tuneMt[, "OOBError"]), "mtry"]
```

As seen from the graph above, the optimal mtry value obtained by using tuneRF is 20 as it results in the lowest Out of Bag (OOB) error. Next, we will retrain our Random Forest Model using this optimal mtry value and calculate the MSE of the new model.

```

set.seed(435)
rf_model3 <- randomForest(SqrRBC ~ .,
                            data = train_data2,
                            ntree = 501,
                            mtry = opt_mtry)

```

```

predictions <- predict(rf_model3, newdata = test_data2)
actuals <- test_data2$SqrRBC

mse5 <- mean((predictions - actuals)^2)
print(mse5)

```

```
## [1] 11.26956
```

We observe that our MSE has decreased again, from 12.1059 to 11.2696, which is a decrease of 0.8364, showing that using the optimal mtry value did indeed improve the predictive power of the random forest model.

Thus, we have found that between linear regression, spline regression and random forests, the random forests model is the most accurate as it has the lowest MSE of 11.2696 compared to 32.3268 of linear regression and 24.8046 of spline regression. Finally, let us perform k-fold cross validation to ensure that our random forest model is robust.

```
set.seed(435)

k <- 10

# Create folds for cross-validation
folds <- createFolds(train_data2$SqrRBC, k = k, list = TRUE)

# Initialize vector to store cross-validation errors
cv_errors <- numeric(k)

# Cross-validation loop
for (i in seq_along(folds)) {
  train_fold <- train_data2[-folds[[i]], ]
  test_fold <- train_data2[folds[[i]], ]

  rfm <- randomForest(SqrRBC ~ ., data = train_fold, ntree = 501, mtry = opt_mtry)

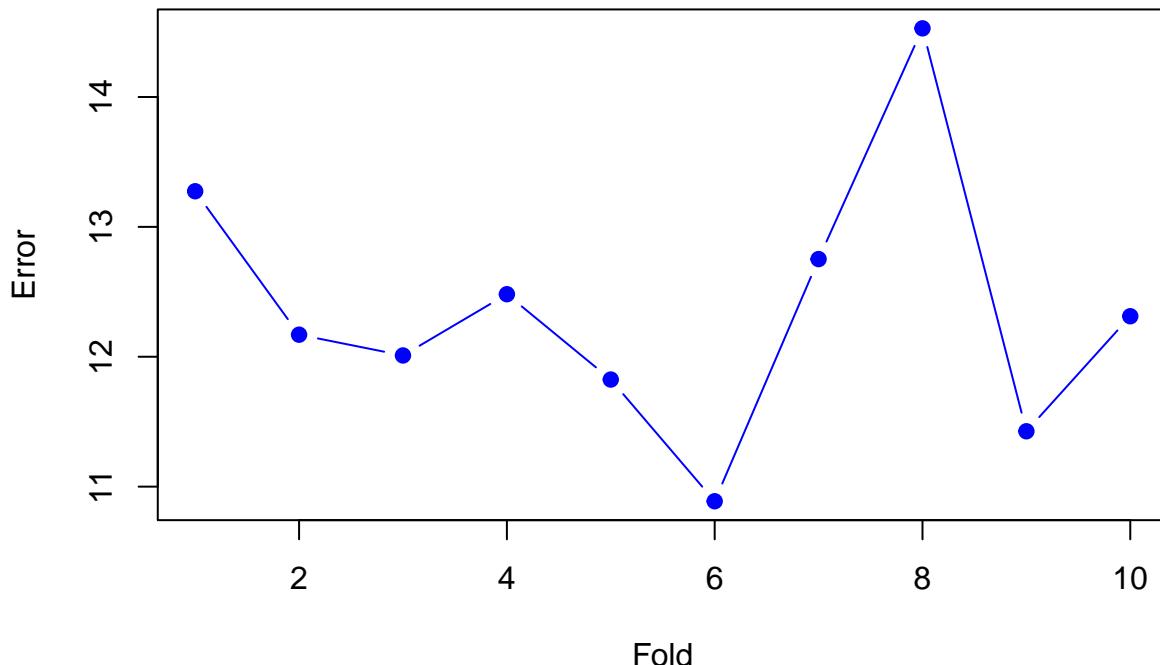
  # Predict on the validation set
  predictions <- predict(rfm, newdata = test_fold)

  # Compute mean squared error for the given fold
  cv_errors[i] <- mean((predictions - test_fold$SqrRBC)^2)
}

# Compute average cross-validation error
mean_cv_error <- mean(cv_errors)

# Plot the cross-validation errors
plot(cv_errors, type = "b", pch = 19, col = "blue",
     xlab = "Fold", ylab = "Error", main = "10-fold Cross-Validation Mean Squared Error")
```

10-fold Cross-Validation Mean Squared Error



Since the mean of the MSE is 12.37 which is similar to the prior attained mean squared error of `round(mse5, 2)`, we can conclude that our random forest model is robust.

With this, we have successfully trained a random foerest model to predict hourly bike rental count in Seoul using environmental information. The predictors used in the model are as follows: Hour, Temperature, Humidity, Wind Speed, Visibility, Dew Point Temperature, Solar Radiation, Heat Index, Wind Chill, Humidex, Seasons and Rained.

Lastly, I will create a new point and use our random forest model to predict the rented bike count during the given hour of that data point.

```
predict_rbc <- function(hr, temp, hum, ws, visibility, dpt, sr, rain, weektype, season) {

  # Create a new data point for prediction
  data_point <- data.frame(
    Hour = hr,
    Temperature = temp,
    Humidity = as.integer(hum),
    Wind.Speed = ws,
    Visibility = as.integer(visibility),
    Dew.Point.Temp = dpt,
    Solar.Radiation = sr,
    Rained = rain,
    Is.Weekend = weektype,
    HeatIdx = 0.5 * ((temp * (9/5) + 32) + 61 + (((temp * (9/5) + 32)-68)*1.2) + (hum*0.094)),
    WindChill = 35.74 + 0.6215*(temp * (9/5) + 32) - 35.75*(ws*2.23694)^(0.16) + 0.4275*(temp * (9/5) +
    HumiDex = temp + 0.5555*(6.11 * exp(5417.7530*(1/273.15 - 1/(273.15 + dpt)))-10),
    SeasonsAutumn = as.integer(season == "Autumn"),
    SeasonsSpring = as.integer(season == "Spring"),
```

```
    SeasonsSummer = as.integer(season == "Summer")
  )

data_point$Hour <- factor(12, levels = levels(train_data2$Hour))

prediction <- (predict(rf_model3, newdata = data_point))^2

return(paste(round(prediction), "bikes"))
}

predict_rbc(17, 30.1, 41, 5, 2000, 12.8, 1.4, 0, 0, "Autumn")

## [1] "1007 bikes"
```