Bike Renting

Abhijeet Nag
05/8/2019

Contents

1. INTRODUCTION

1.1 Problem statement

Bike sharing systems are new generation of traditional bike rentals where whole process from membership, rental and return back has become automatic. Through these systems, user is able to easily rent a bike from a particular position and return back at another position. Currently, there are about over 500 bike-sharing programs around the world which is composed of over 500 thousands bicycles. Today, there exists great interest in these systems due to their important role in traffic, environmental and health issues.

Bike-sharing rental process is highly correlated to the environmental and seasonal settings. For instance, weather conditions, precipitation, day of week, season etc. can affect the rental behaviors. The task given in this assignment is predication of bike rental count daily based on the environmental and seasonal settings

1.2 Data

Our task is to build regression models which will predict the count of bike rented depending on various environmental and seasonal conditions Given below is a sample of the data set that we are using to predict the count of bike rents:

Table 1.1: Sample Data (Columns: 1-8)

instant	dteday	season	yr	mnth	holiday	weekday	workingday
1	1/1/2011	1	0	1	0	6	0
2	1/2/2011	1	0	1	0	0	0
3	1/3/2011	1	0	1	0	1	1
4	1/4/2011	1	0	1	0	2	1
5	1/5/2011	1	0	1	0	3	1
6	1/6/2011	1	0	1	0	4	1

Table 1.2: Sample Data (Columns: 7-16)

weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	0.363478	0.353739	0.696087	0.248539	131	670	801
1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
1	0.2	0.212122	0.590435	0.160296	108	1454	1562
1	0.226957	0.22927	0.436957	0.1869	82	1518	1600
1	0.204348	0.233209	0.518261	0.0895652	88	1518	1606

Variables present in given dataset are instant, dteday, season, yr, mnth, holiday, weekday, workingday, weathersit, temp, atemp, hum, windspeed, casual, registered, cnt

The details of variable present in the dataset are as follows - instant:
Record index

dteday: Date

season: Season (1:springer, 2:summer, 3:fall, 4:winter) yr: Year

(0: 2011, 1:2012)

mnth: Month (1 to 12)

hr: Hour (0 to 23)

holiday: weather day is holiday or not (extracted fromHoliday Schedule) weekday:

Day of the week

workingday: If day is neither weekend nor holiday is 1, otherwise is 0. weathersit:

(extracted fromFreemeteo)

1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

temp: Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min),
t_min=-8, t_max=+39 (only in hourly scale)
atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_maxt_min),
t_min=-16, t_max=+50 (only in hourly scale)

hum: Normalized humidity. The values are divided to 100 (max)

windspeed: Normalized wind speed. The values are divided to 67 (max) casual:

count of casual users

registered: count of registered users

cnt: count of total rental bikes including both casual and registered

2. Methodology

2.1 Pre Processing

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis.

2.1.1 Exploratory Data Analysis

In exploring the data we have

- While doing EDA converted season, mnth, workingday, weathersit into categorical variables
- **Feature Engineering** :Changed dteday variables's date value to day of date and converted to categorical variable having 31 levels as a month has 31 days.
- Deleted instant variable due to no contribution to the model as it is nothing but an index.
- Visualized the effect of season and weather on bike rental count daily.

2.1.2 Missing Value Analysis

Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model because we have not analysed the behavior and relationship with other variables correctly. It can lead to wrong prediction or classification.

Treating Missing Values:-
First we will check how much percentage of the data has missing value, if it is greater than 30% then we will discard the data set. On the other hand if it is not then either we can delete those observation or can impute it. Deleting observation may lead to loss of important information so it is better to impute it. This are the following ways to impute a missing value:-

- Deletion
- Mean/Mode/Median Imputation
- KNN Imputation

In R function(x){sum(is.na(x))} is the function used to check the sum of missing values.
In python bike_train.isnull().sum() is used to detect any missing values.

In our data there are no missing values in any of the predictor variable.

dteday	0
season	0
yr	0
mnth	0
holiday	0
weekday	0
workingday	0
weathersit	0
temp	0
hum	0
windspeed	0
cnt	0

There is no missing value found in given dataset.

2.1.3 Outlier Analysis

Outlier is a commonly used terminology by analysts and data scientists as it needs close attention else it can result in wildly wrong estimations. Simply speaking, Outlier is an observation that appears far away and diverges from an overall pattern in a sample.
Outlier can be of two types: Univariate and Multivariate. Above, we have discussed the example of univariate outlier. These outliers can be found when we look at distribution of a single variable. Multi-variate outliers are outliers in an n-dimensional space. In order to find them, you have to look at distributions in multi-dimensions.

Figure 2.1 and 2.2 are visualization of predictor variable temp, atemp, hum, windspeed against target variable cnt which is a boxplot. Let us take a look

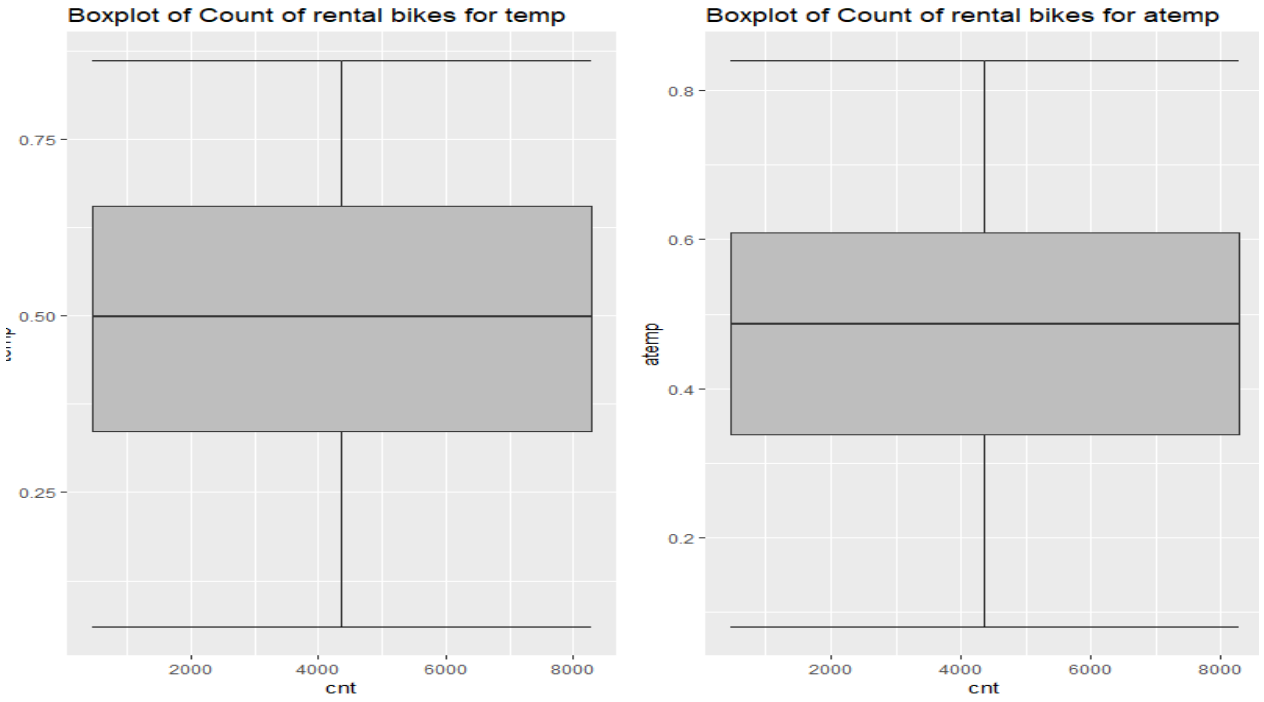


Figure 2.1 Boxplot graph of temp and atemp variables

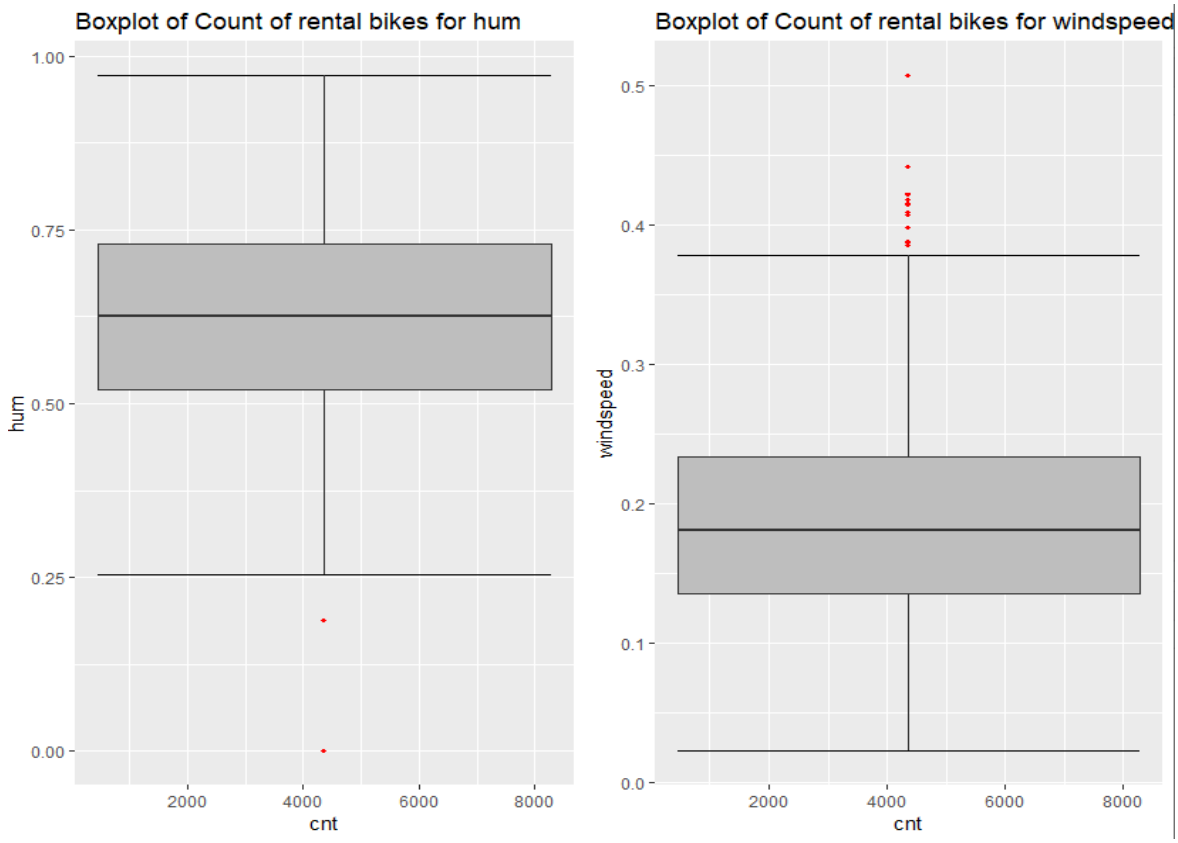


Figure 2.2 Boxplot graph of hum and windspeed variables

According to above visualizations there is no outlier found in temp and atemp variable but there are few outliers found in windspeed and hum variable.
For treating the missing value first created NA wherever outlier is present then treating it like a missing value. Imputed all the missing value using KNN.

2.1.4 Feature Selection

In Feature selection we plot the correlational analysis of the predictor variable and target variable too. We will look for high correlation between predictor and target variable and low correlation among the predictor variable. If there is high correlation between two predictor variable then we will delete one of them. This is called **Dimension Reduction**.
As our target variable is continuous so we can only go for correlation check. As chi-square test is only for categorical variable.

Figure 2.4 show a correlation plot for all numeric variable present in dataset

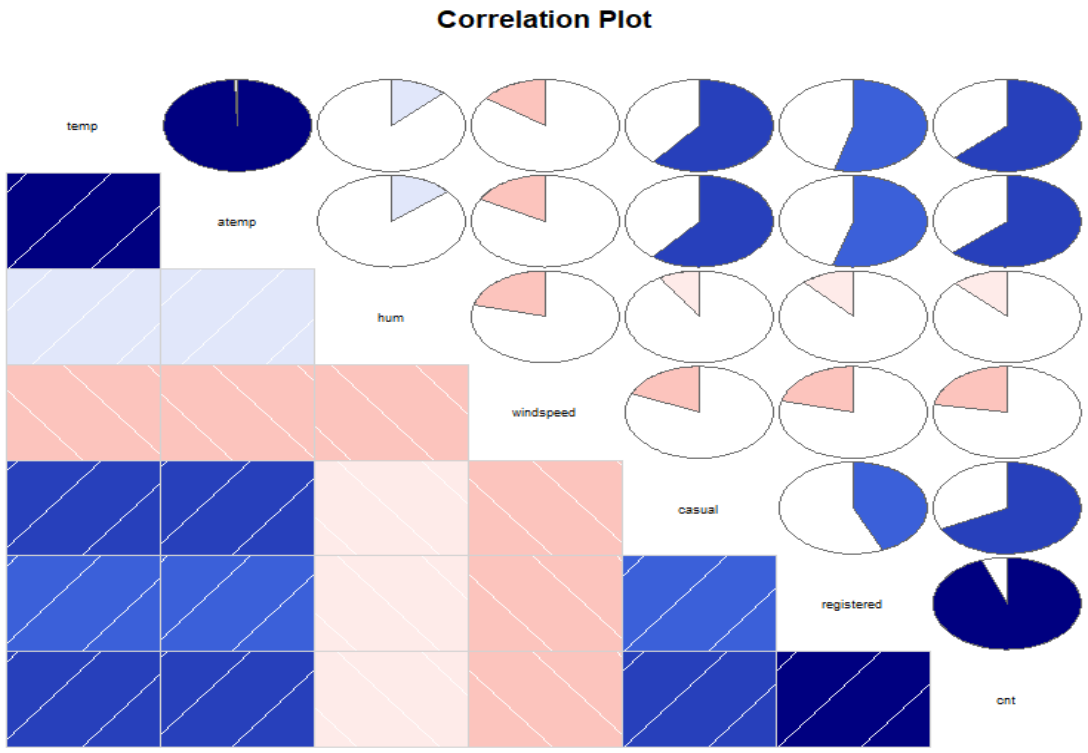


Figure 2.4 correlation plot

In above visualization we can see that only 2 variables are highly correlated with each other. Dark blue color represent highly correlated and light color represent very less correlated so we have a choice to remove either temp or atemp because these variables contains nearly equal information.
So I have removed atemp variable from dataset.

2.1.5 Feature Scaling

Data scaling is very important step in data preprocessing when we are dealing with variables which have different scales. For example let us consider a data set which has ‘n’ no. of different independent variable out of which two of them are Age and Income. Range of Age is 20 to 75 and range of Income is 20,000 to 5,75,000. Here as we can clearly see the two variables are of different scale which will cause anomaly in the final model as Income will completely dominate the model result. So to bring all the independent variables in same range we use Data Scaling/Feature Scaling. There are two methods to do data scaling:-
a) Normalization
b) Standardization/Z-score

Both the method are used for variables which are continuous in nature. But based upon the distribution of data we will select the method. If the data is normally distributed or uniformly distributed then we will go for Standardization. If the data is negatively skewed/left skewed or positively skewed/right skewed then we will go for Normalization. Here negatively skewed indicates that the mean of the data is less than the median and positively skewed means mean of the data is larger

2.2 Modeling

2.2.1 Model Selection

In this case we have to predict the count of bike renting according to environmental and seasonal condition. So the target variable here is a continuous variable. For Continuous we can use various Regression models. Model having less error rate and more accuracy will be our final model.
Models built are

1. c50 (Decision tree for regression target variable)
2. Random Forest (with 200 trees)
3. Linear regression

2.2.2 C50

This model is also known a Decision tree for regression target variable.
For this model we have divided the dataset into train and test part using random sampling. Where train contains 80% data of data set and test contains 20% data and contains 14 variable where 14th variable is the target variable.

Creating Model In
R :-

```
######################## Decision TREE ######################

regressor_DT= rpart(formula = cnt ~., data = training_set, method = "anova")
#Predicting the test set result
y_pred_DT= predict(regressor_DT, test_set[, -14])
```

In python:-

```
###################### Decision Tree ######################
from sklearn.tree import DecisionTreeRegressor
regressor_DT = DecisionTreeRegressor(max_depth=10).fit(train.iloc[:,0:13], train.iloc[:,13])
predictions_DT = regressor_DT.predict(test.iloc[:,0:13])
```

2.2.3Random Forest

In Random forest we have divided the dataset into train and test part using random sampling. For this model we have divided the dataset into train and test part using random sampling Where train contains 80% data of data set and test contains 20% data and contains 14 variable where 14th variable is the target variable.

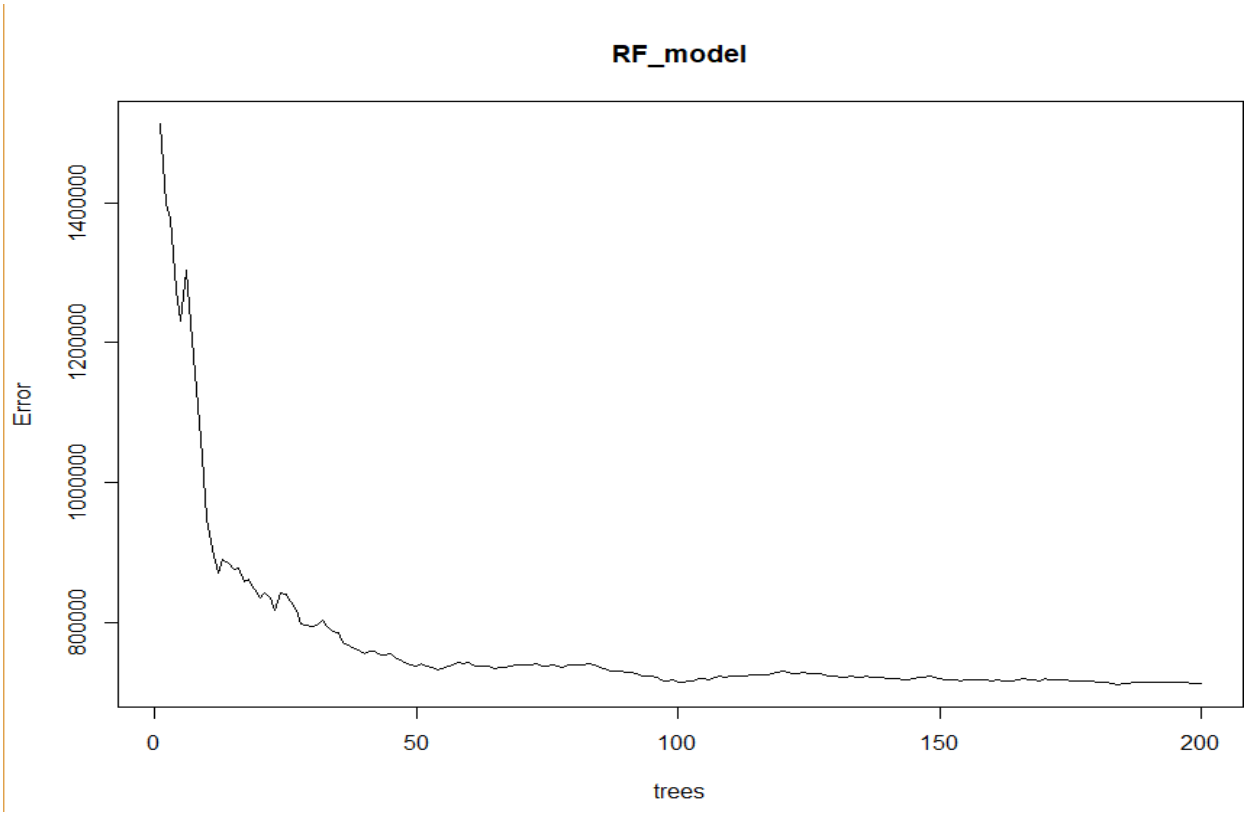


Figure2.2.3

Above Figure2.2.3 represents the curve of error rate as the number of trees increases. After 200 trees the error rate reaches to be constant.

In this model we are using 200 trees to predict the target variable.

Creating Model

In Python

```
###################### Random Forest ########################
from sklearn.ensemble import RandomForestRegressor
regressor_RF = RandomForestRegressor(n_estimators = 200).fit(train.iloc[:,0:13], train.iloc[:,13])
RF_Predictions = regressor_RF.predict(test.iloc[:,0:13])
```

In R

```
regressor_RF= randomForest(cnt ~., training_set, importance= TRUE, ntree= 200)
#Predicting the test set result
y_pred_RF= predict(regressor_RF, test_set[,-14])
```

2.2.3 Linear Regression

It is one of the most widely known modeling technique. Linear regression is usually among the first few topics which people pick while learning predictive modeling. In this technique, the dependent variable is continuous, independent variable(s) can be continuous or discrete, and nature of regression line is linear.

Further the data is again divided into train and test with 80 % train data and 20 % test data using random sampling.

Creating Model

In R

```
################### Multiple Linear Regression ##############################
regressor= lm(formula = cnt ~., data = training_set)
# Adjusted R- square = 0.9842, MAPE= 4.5
#Predicting the test set results
y_pred= predict(regressor,test_set[, -14])
```

In Python

```
################ Linear Regression ######################
from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(train.iloc[:, 0:13], train.iloc[:, 13])
predictions_LR= regressor.predict(test.iloc[:, 0:13])
```

**

Model summary

```
Call:
lm(formula = cnt ~ ., data = training_set)

Residuals:
    Min      1Q  Median      3Q     Max
-369.32 -119.41  -33.37   55.34 1494.57

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   26.55537  106.11395   0.250 0.802490
dteday02       6.18919   75.64438   0.082 0.934821
dteday03      12.69558   79.60191   0.159 0.873346
dteday04      91.86916   76.65647   1.198 0.231283
dteday05     -17.28706   78.36092  -0.221 0.825483
dteday06      73.83854   82.24873   0.898 0.369733
dteday07      45.55614   77.46206   0.588 0.556712
dteday08     -74.70047   77.28562  -0.967 0.334215
dteday09     105.84334   78.05504   1.356 0.175680
dteday10     -83.75032   82.75686  -1.012 0.312003
dteday11     -14.84889   78.69716  -0.189 0.850414
dteday12     -38.45150   77.77562  -0.494 0.621238
dteday13     -48.22397   80.22873  -0.601 0.548046
dteday14      10.64545   81.95987   0.130 0.896706
dteday15      57.99551   76.75746   0.756 0.450248
dteday16      57.64757   80.55453   0.716 0.474536
dteday17     -44.86315   79.85002  -0.562 0.574463
dteday18     -15.01064   81.82215  -0.183 0.854512
dteday19     -29.99449   80.20586  -0.374 0.708579
dteday20     112.23572   78.21892   1.435 0.151914
dteday21      -6.47972   80.23307  -0.081 0.935663
dteday22      50.93426   83.49705   0.610 0.542118
dteday23     115.38789   82.99372   1.390 0.165022
dteday24     -77.75219   85.01583  -0.915 0.360843
dteday25     -45.43095   76.00203  -0.598 0.550258
dteday26     -19.15049   76.98785  -0.249 0.803654
dteday27     149.59593   79.82402   1.874 0.061477 .
dteday28     -20.71262   83.48579  -0.248 0.804156
dteday29      45.19735   85.64216   0.528 0.597899
dteday30     -48.09563   78.72890  -0.611 0.541529
dteday31      51.44158   88.78179   0.579 0.562558
season2      123.63724   65.35365   1.892 0.059066 .
season3      -28.40498   76.19899  -0.373 0.709468
season4       23.43467   68.15262   0.344 0.731093
yr1           44.92511   36.32438   1.237 0.216724
mnth2        -32.69746   51.72031  -0.632 0.527533
mnth3        -27.49758   59.57637  -0.462 0.644594
mnth4       -116.67121   90.77090  -1.285 0.199242
mnth5        -86.97828   97.77140  -0.890 0.374084
mnth6       -140.30424  101.24256  -1.386 0.166391
mnth7        -64.75782  112.73363  -0.574 0.565922
mnth8       -106.19240  109.49220  -0.970 0.332562
mnth9         17.47105   95.22761   0.183 0.854503
mnth10       -28.65135   85.65483  -0.334 0.738138
mnth11       -79.42726   81.77033  -0.971 0.331824
mnth12       -58.24251   63.03252  -0.924 0.355908
holiday1     266.91103   68.99510   3.869 0.000123 ***
weekday1    -171.90753   51.98426  -3.307 0.001008 **
weekday2    -170.22444   54.23728  -3.139 0.001794 **
```

```
weekday3    -166.38779   55.34727  -3.006 0.002772 **
weekday4    -194.65508   54.07132  -3.600 0.000348 ***
weekday5    -169.13925   50.53122  -3.347 0.000875 ***
weekday6     113.33892   38.04699   2.979 0.003027 **
workingday1         NA         NA      NA       NA
weathersit2    3.06401   29.94243   0.102 0.918534
weathersit3   60.18913   80.00131   0.752 0.452177
temp         298.41922  162.11904   1.841 0.066225 .
hum         -192.29736  115.03759  -1.672 0.095199 .
windspeed    104.98412  168.94004   0.621 0.534588
casual         0.96803    0.04180  23.158  < 2e-16 ***
registered     1.02859    0.01801  57.097  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 242.3 on 524 degrees of freedom
Multiple R-squared:  0.9859,   Adjusted R-squared:  0.9843
F-statistic: 621.3 on 59 and 524 DF,  p-value: < 2.2e-16
```

3. Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case of Bike Renting, the latter two, Interpretability and Computation Efficiency, do not hold much significance. Therefore we will use Predictive performance as the criteria to compare and evaluate models.

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

3.1.1 Mean Absolute Percentage Error (MAPE)

MAPE is one of the error measures used to calculate the predictive performance of the model. We will apply this measure to our models that we have generated in the previous sections

```
# MAPE Function
def MAPE(y, yhat):
    mape = np.mean(np.abs((y - yhat) / y))*100
    print("MAPE:", mape)
```

In above function y is the actual value and yhat is the predicted value. It will provide the error percentage of model.

3.1.2 Root Mean Squared Error(RMSE)

RMSE is the most popular evaluation metric used in regression problems. It follows an assumption that error are unbiased and follow a normal distribution.

```python
#RMSE Function
def RMSE(y_test,yhat):
    mse = np.mean((y_test-yhat)**2)
    print("Mean Square : ",mse)
    rmse=np.sqrt(mse)
    print("Root Mean Square : ",rmse)
```

MAPE & RMSE value in Python are as follow

```python
#Mape for Linear Regression
MAPE(test.iloc[:,13],predictions_LR)
RMSE(test.iloc[:,13],predictions_LR)
```

```
MAPE: 1.8615869314321472
Mean Square :  18503.736740826887
Root Mean Square :  136.02844092625222
```

```python
#Mape for Decision Tree
MAPE(test.iloc[:,13],predictions_DT)
RMSE(test.iloc[:,13],predictions_DT)
```

```
MAPE: 4.0471362073593795
Mean Square :  63131.00691280461
Root Mean Square :  251.2588444469261
```

```python
#Mape for Random Forest
MAPE(test.iloc[:,13],RF_Predictions)
RMSE(test.iloc[:,13],RF_Predictions)
```

```
MAPE: 2.1659028272006298
Mean Square :  20716.912411413417
Root Mean Square :  143.933708391792
```

```python
## Extracting Predicted values from the Multiple Linear Regression Model
result=pd.DataFrame(test.iloc[:,0:13])
result['predicted_cnt'] = (predictions_LR)

#result.to_csv("Bike_Count python.csv",index=False)
```

MAPE & RMSE values is R are as follow

```
> #alternate method
> regr.eval(test_set[, 14], y_pred, stats = c('mae', 'rmse', 'mape', 'mse'))
         mae         rmse         mape          mse
1.771691e+02 3.005583e+02 4.847615e-02 9.033531e+04
> regr.eval(test_set[, 14], y_pred_DT, stats = c('mae', 'rmse', 'mape', 'mse'))
         mae         rmse         mape          mse
4.330851e+02 5.653848e+02 1.158677e-01 3.196600e+05
> regr.eval(test_set[, 14], y_pred_RF, stats = c('mae', 'rmse', 'mape', 'mse'))
         mae         rmse         mape          mse
2.926175e+02 4.177748e+02 1.003798e-01 1.745358e+05
>
```

3.2 Model Selection

We can see that in both R and Python Linear Regression Model fits the best out of Decision Tree and Random Forest. MAPE Value is the lowest for Linear Regression. So to improve the model and enhance its performance so that it can perform efficiently on new test set, implemented **K Fold Cross Validation**.

In R:-

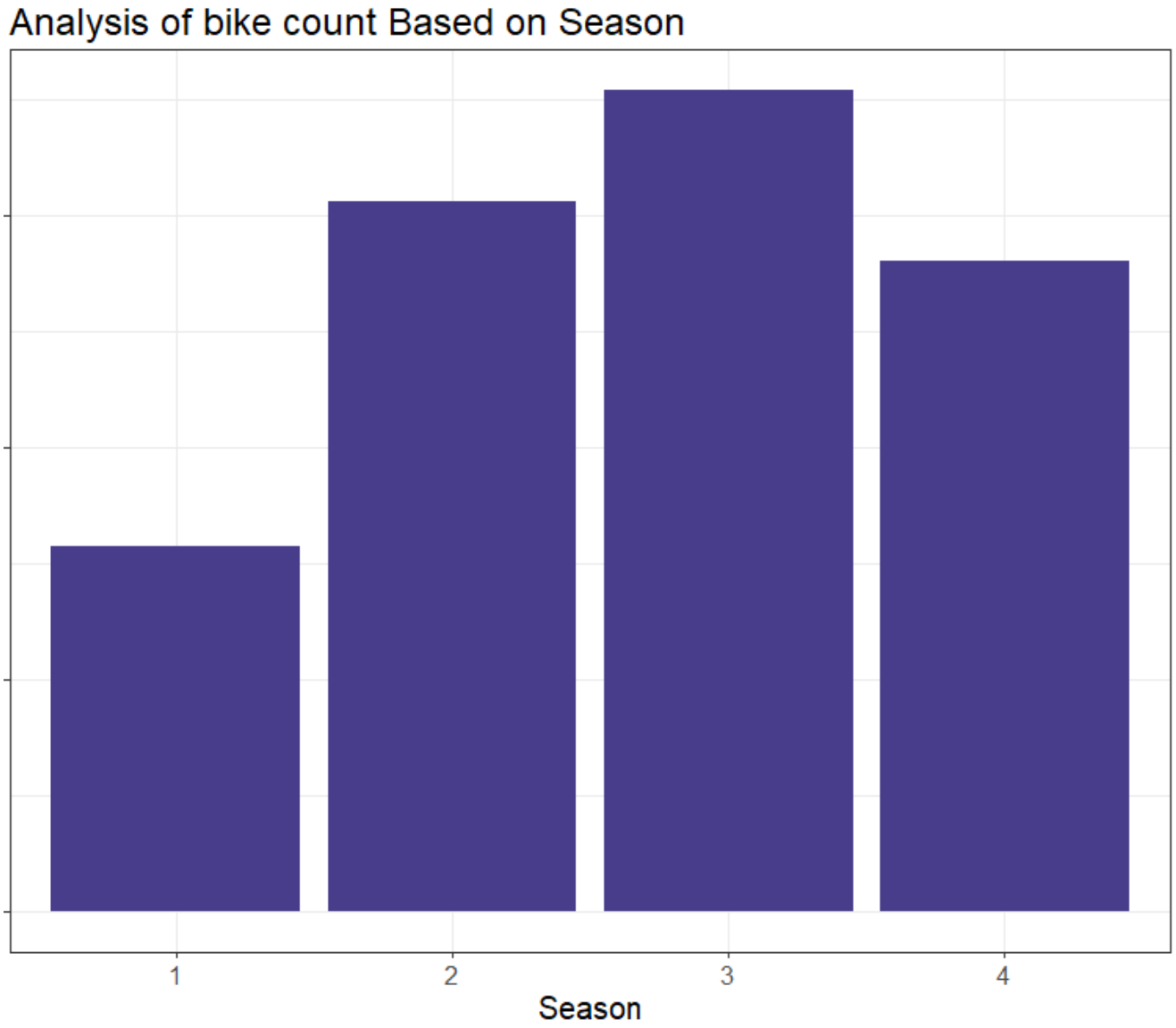
```
# Applying K-fold Cross Validation to deal with variance problem while testing the model with
# another test set
library(caret)
folds= createFolds(training_set$cnt, k= 10)
cv= lapply(folds, function(x) {
  training_fold= training_set[-x, ]
  test_fold= training_set[x, ]
  regressor= lm(formula = cnt ~., data = training_fold)
  y_pred= predict(regressor,test_fold[, -14])
  mape= function(y, yhat){
    mean(abs((y-yhat)/y))*100
  }
  error= mape(test_fold[,14], y_pred)
  return(error)
})
MAPE_LR=mean(as.numeric(cv))
```

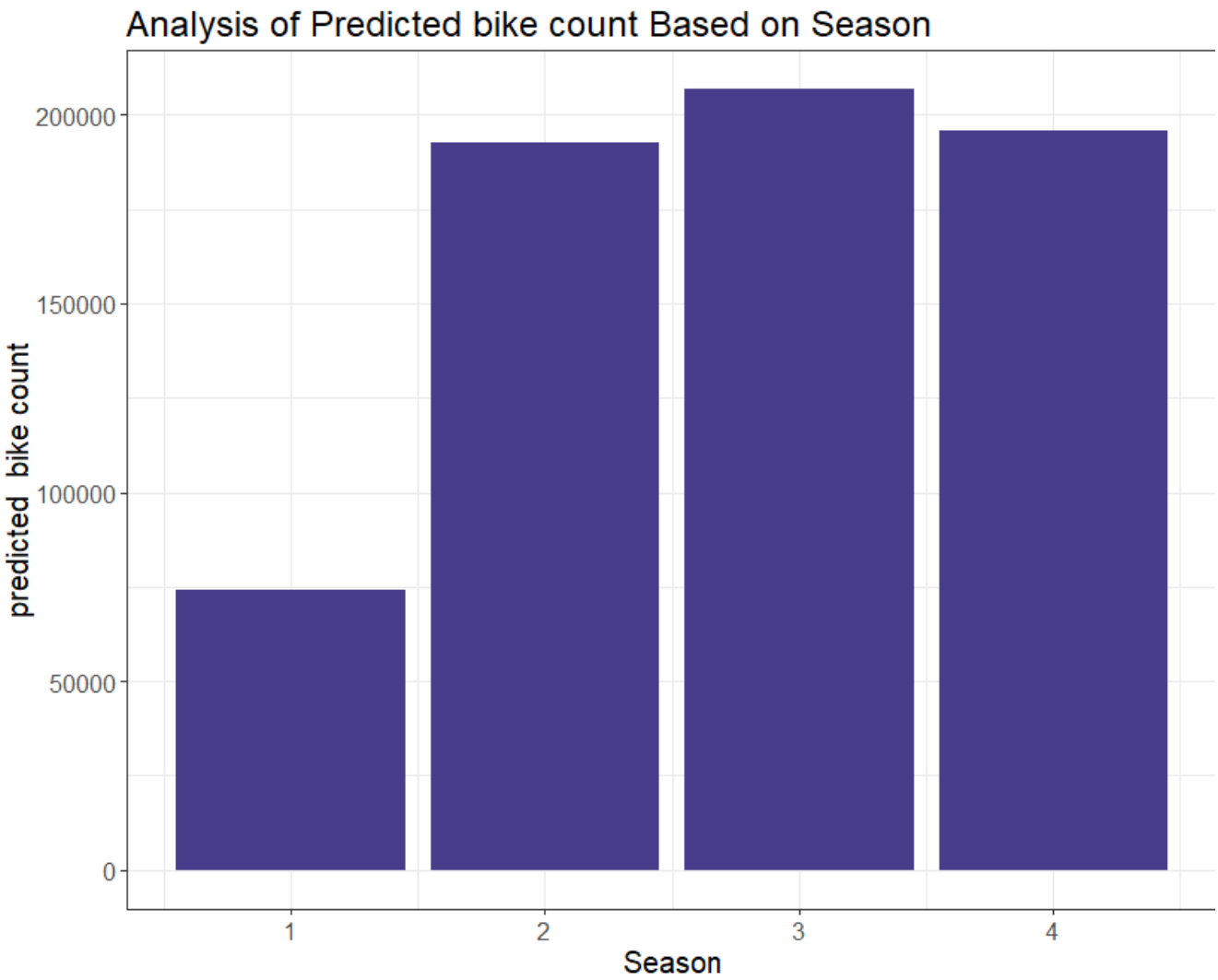
In Python:-

```
# After Calculating MAPE & RMSE for all the models we can see that Linear model fits the data best. So applying K Fold Cross Val
# to check the performance of the model on different test set.
#Applying K Fold Cross Validation
from sklearn.model_selection import cross_val_score
accuracies= cross_val_score(estimator= regressor, X= train.iloc[:,0:13], y= train.iloc[:,13], cv= 10)
accuracies.mean()
```

4. Visualizations

4.1 Visualization on result stored on seasonal settings

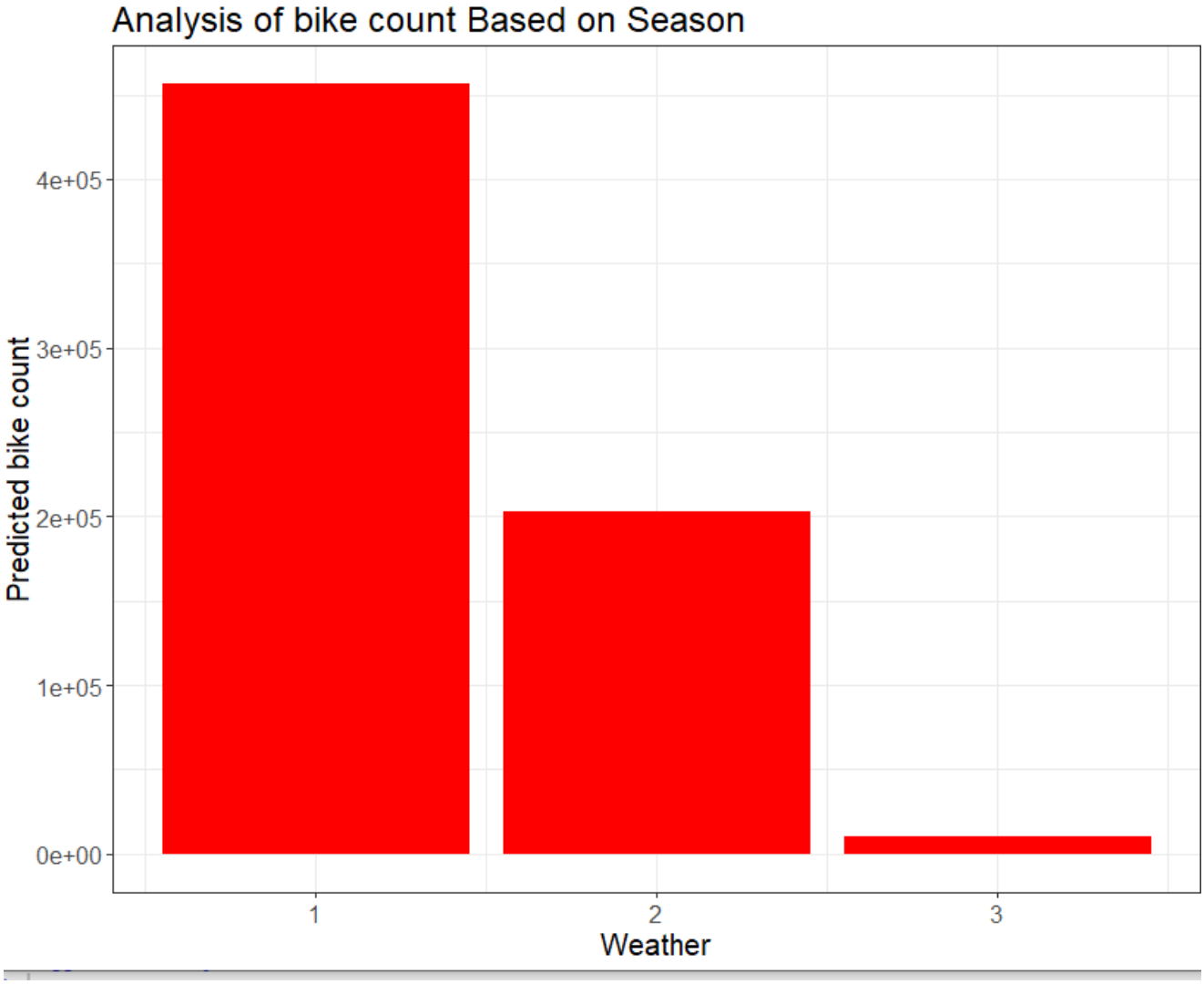


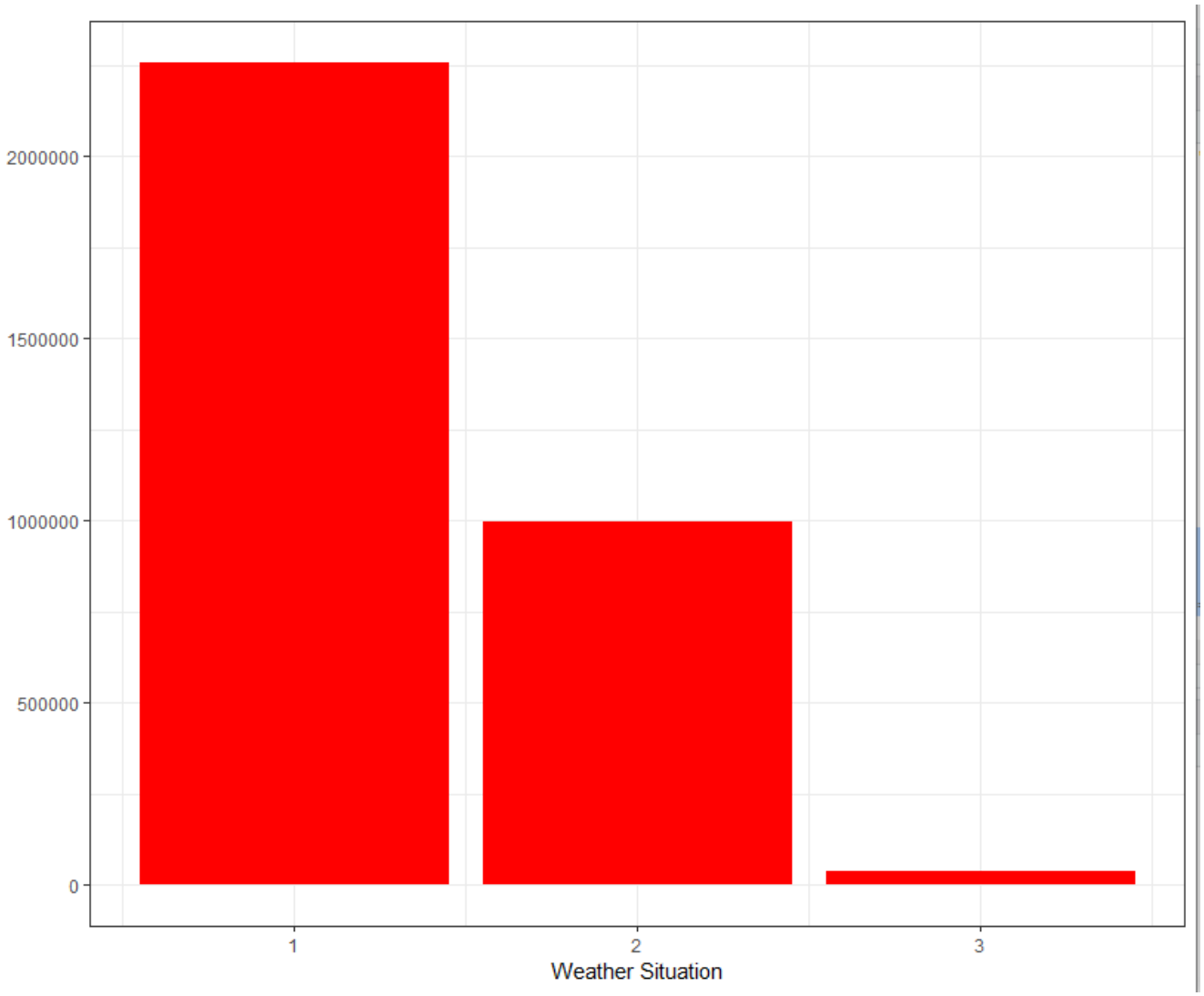


season: Season (1:springer, 2:summer, 3:fall, 4:winter)

Above two bar graph represents the comparison of predicted count value and actual count value based on seasonal condition.

4.2 Visualization on result stored on weather conditions





1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

Above bar graph shows predicted count and actual count based on weather conditions

According to Seasonal and weather condition bar graph we can clearly notice that fall season that is autumn and where weather conditions are clear, few or partly cloudy on these conditions bike rent count is quite high than any other condition.