Debarun Banerjee

Bike Rental Prediction – Project 3

Date- 06/05/2019

Contents

[**Chapter 1: Introduction** 4](#_Toc7891034)

[**1.1 Problem Statement** 4](#_Toc7891035)

[**1.2 Dataset observations:** 4](#_Toc7891036)

[**1.3 Data** 4](#_Toc7891037)

[**1.4 Unique Values** 5](#_Toc7891038)

[**Chapter 2: Methodology** 5](#_Toc7891039)

[**2.1 Pre – Processing** 5](#_Toc7891040)

[**2.1.1 Transforming Datatypes** 5](#_Toc7891041)

[**2.2 Missing Value Analysis** 6](#_Toc7891042)

[**2.3 Outlier Analysis** 7](#_Toc7891043)

[**2.4 Exploratory Data Analysis:** 8](#_Toc7891044)

[**2.4.1 Distribution of Categorical Variable** 8](#_Toc7891045)

[**2.4.1.1 Factor Plot:** 8](#_Toc7891046)

[**2.4.2 Distribution of Continuous Variables** 10](#_Toc7891047)

[**2.4.2.1 Histogram Plots:** 10](#_Toc7891048)

[**2.5 Variable Reduction/Feature Selection Techniques** 11](#_Toc7891049)

[**2.5.1 Chi-Square test for Categorical variables** 11](#_Toc7891050)

[**2.5.2 VIF to check multicollinearity** 11](#_Toc7891051)

[**2.5.3 After reduction checking the correlation** 12](#_Toc7891052)

[**2.6 Normalization of Continuous Variables:** 12](#_Toc7891053)

[**2.7 Scaling of Categorical Variables:** 13](#_Toc7891054)

[**2.8 Creating Dummy Variables for Categorical columns:** 13](#_Toc7891055)

[**3. Model Building** 13](#_Toc7891056)

[**3.1 Splitting data into train and test** 13](#_Toc7891057)

[**3.2 Decision tree** 14](#_Toc7891058)

[**3.2.1 Tuning parameter of Decision Tree (max\_depth)** 14](#_Toc7891059)

[**3.2.2 Tuning parameter of Decision Tree (min\_sample\_split)** 15](#_Toc7891060)

[**3.2.3 Decision Tree with optimum parameters** 15](#_Toc7891061)

[**3.3 Random Forest Regression** 15](#_Toc7891062)

[**3.3.1 Randomized CV search technique to get best parameters of Random Forest** 16](#_Toc7891063)

[**3.3.2 Tuning Random Forest with Best Parameters** 16](#_Toc7891064)

[**3.4 Linear Regression** 17](#_Toc7891065)

[**3.5 Ridge Regression:** 17](#_Toc7891066)

[**3.6 LASSO Regression** 18](#_Toc7891067)

[**3.7 KNN Algorithm for Regression** 18](#_Toc7891068)

[**3.7.1 Parameter tuning for KNN** 19](#_Toc7891069)

[**3.7.2 Building KNN with best K value** 19](#_Toc7891070)

[**3.8 XGBoost** 19](#_Toc7891071)

[**3.9 ARIMA** 20](#_Toc7891072)

[**3.10 Model Selection and output tables** 20](#_Toc7891073)

[**3.10.1 RMSE** 20](#_Toc7891074)

[**3.10.2 R-Squared value** 20](#_Toc7891075)

[**3.10.3 MSE** 21](#_Toc7891076)

[**3.10.4 MAPE** 21](#_Toc7891077)

[**3.11 Key Points:** 21](#_Toc7891078)

[**3.11.1 K Fold Cross Validation** 21](#_Toc7891079)

[**3.11.2 Exception Handling in python** 21](#_Toc7891080)

[**3.12 Conclusion** 21](#_Toc7891081)

[**R- Code:** 21](#_Toc7891082)

[**Additional Plots:** 21](#_Toc7891083)

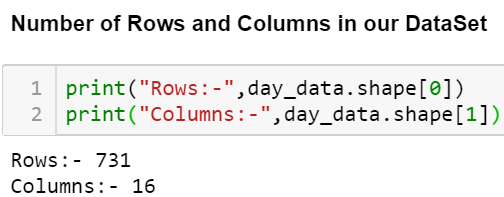
# **Chapter 1: Introduction**

## **1.1 Problem Statement**

The aim of this project is to predict the count of bike rentals based on the seasonal and environmental settings. By predicting the count, it would be possible to help accommodate in managing the number of bikes required daily basis and being prepared for high demand of bikes during peak periods.

## **1.2 Dataset observations:**

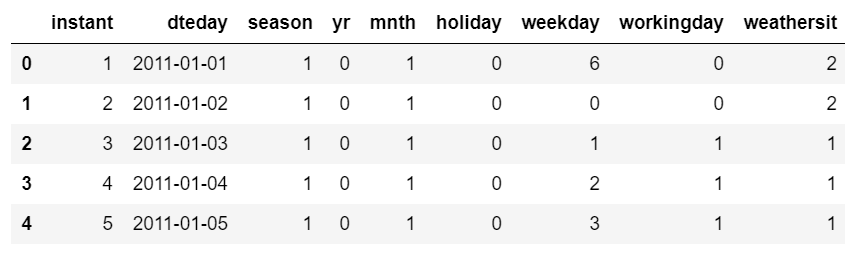
Before going to predict any model, it is very important to get the overview of the data that how many observations we have along with number of variables.



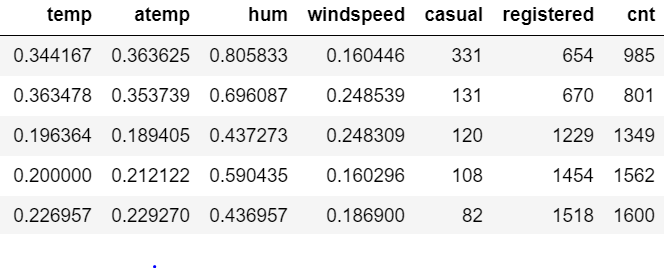
## **1.3 Data**

The goal is to build regression models which will predict the number of bikes used based on the environmental and season behaviour. Given below is a sample of the data set that we are using to predict the number of bikes:

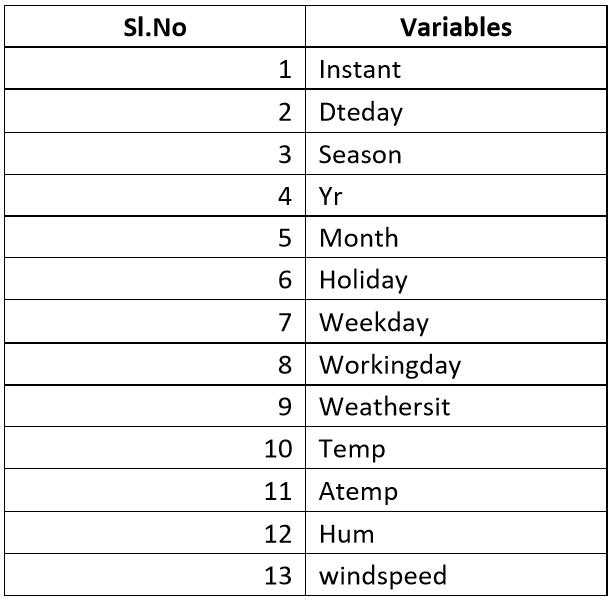
**Table 1: Col (1-9)**



**Table 2: col (10-16)**

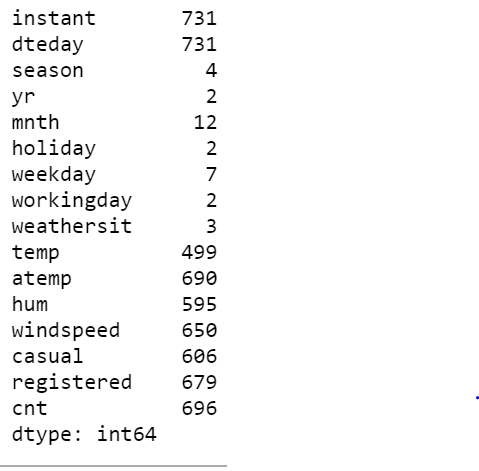


As we can see, in our dataset we have 16 columns in total. Out of that we can see our target variable “cnt” is the summation of other two continuous variables “registered” and “casual”. Hence, we need to predict the total count using other 13 predictor variables.



## **1.4 Unique Values**

We need to check the number of unique values of each column which will give us the idea of either a column is continuous or categorical in nature. Below is the number of unique values in each columns of Dataset.



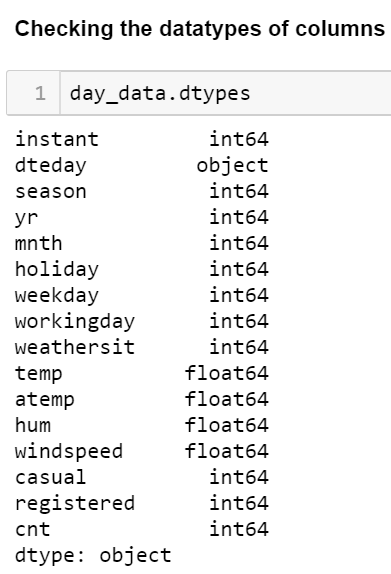
# **Chapter 2: Methodology**

## **2.1 Pre – Processing**

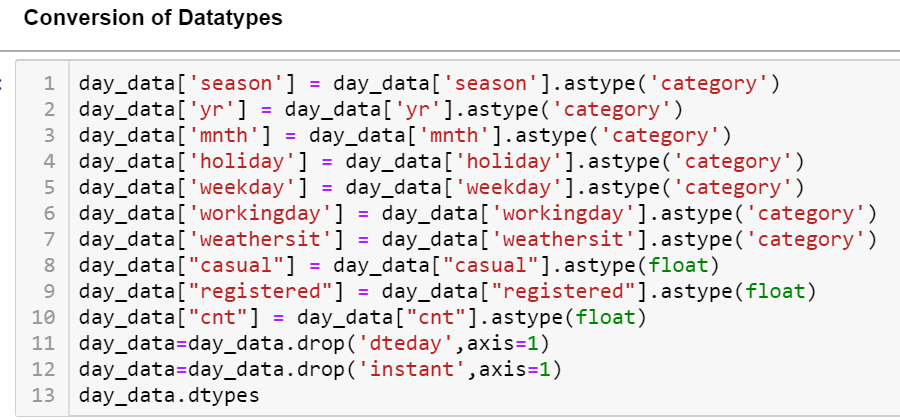
A predictive model requires that we look at the data before we start to create a model. However, in data mining, looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is known as Exploratory Data Analysis. In this project we look at the distribution of categorical variables and continuous variables. We also look at the missing values in the data and the outliers present in the data.

## **2.1.1 Transforming Datatypes**

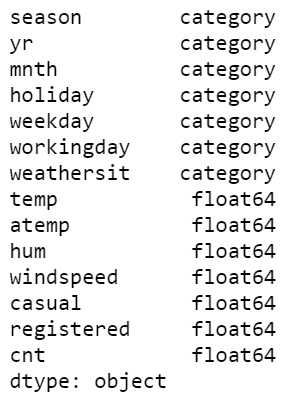
Before entering any pre-processing steps, we need to check the datatypes of the dataset.



So, we have converted our datatypes as required and eliminated “instant” as this is an ID column of observations which generally contribute nothing to predict trends and “dtedate” column as the same information is carried by other variables as well.

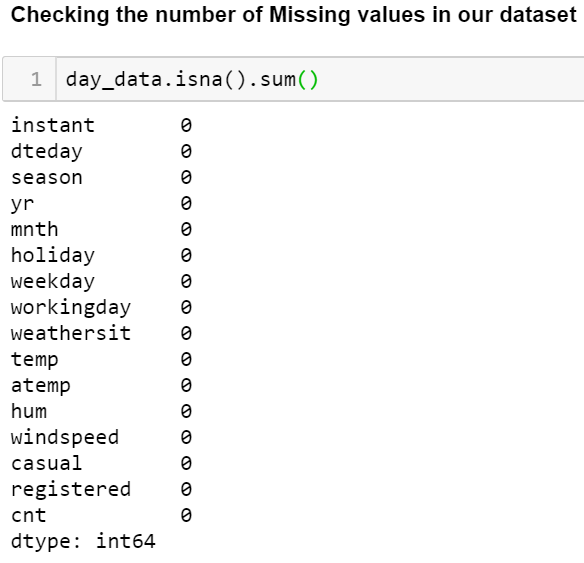


Output is as given:



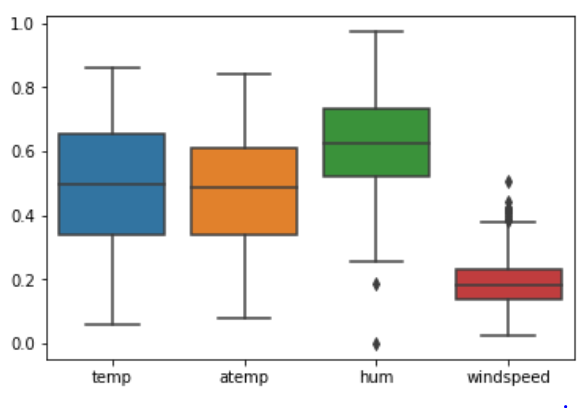
## **2.2 Missing Value Analysis**

We have checked that there are no missing values present in our dataset.

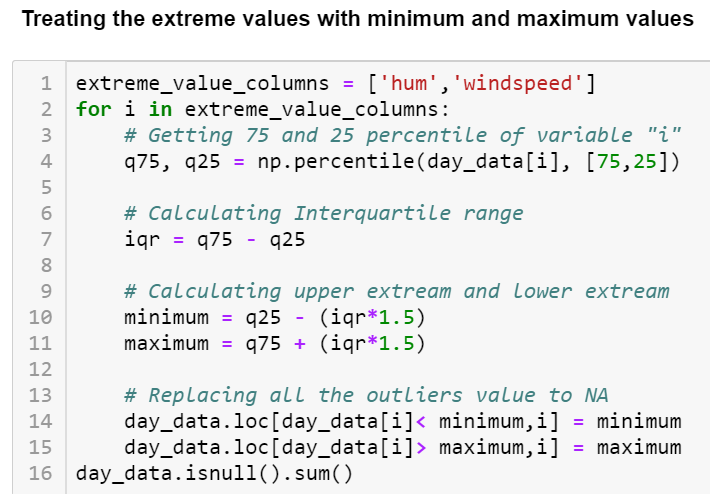


## **2.3 Outlier Analysis**

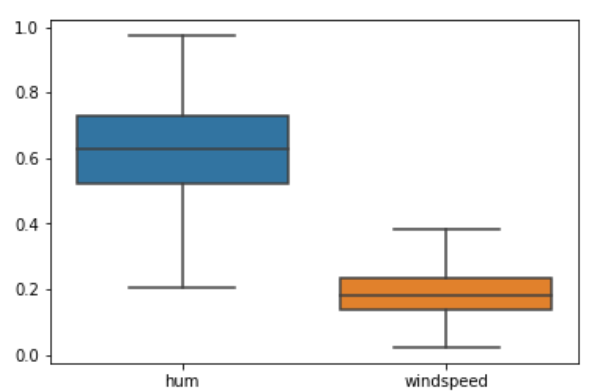
It can be observed from the distribution of variables that almost none of the variables are normally distributed. The skew in these distributions can be explained by the presence of outliers and extreme values in the data. One of the steps in pre-processing involves the detection and removal of such outliers. In this project, we use boxplot to visualize and remove outliers. Any value lying outside of the lower and upper whisker of the boxplot are outliers.



We can see that outliers have been detected in two columns “hum”, “windspeed”. So, we need to remove those outliers and put the desired values. In the below code, we have imputed the outliers in Maximum and Minimum values.



We have calculated the 75th quartile and 25th quartile. The difference between these two indicates the Interquartile Range (IQR). 1.5 times of IQR if we subtract from minimum, we get the lower whisker and the same addition with maximum gives us Upper Whisker. Any value lies outside these upper and lower whiskers are detected as outliers. We have imputed the outliers above upper whisker with maximum value and outliers below the lower whisker with minimum value. After the imputation, we have checked in the boxplot again and found no outliers this time which indicates successful imputation of the outliers.



## **2.4 Exploratory Data Analysis:**

Exploratory data analysis is an approach to analysing data sets to summarize their main characteristics using visual methods. As a part of Exploratory Data Analysis, we have created separated plots for continuous and categorical variables.

## **2.4.1 Distribution of Categorical Variable**

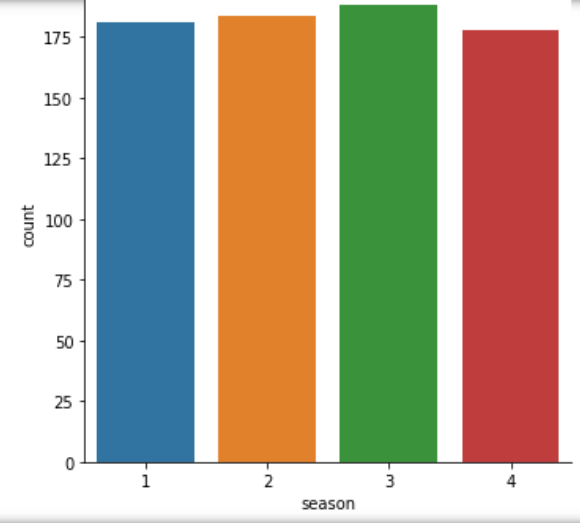
In order to analyse the distribution between categorical variable, we have created Factor plot.

### **2.4.1.1 Factor Plot:**

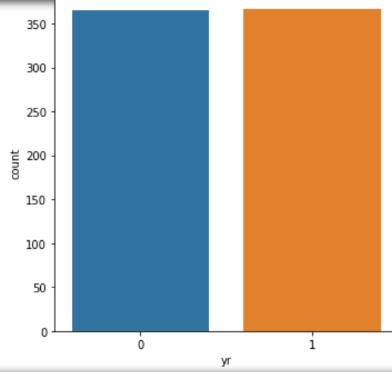
A factor plot is simply the same plot generated for different response and factor variables and arranged on a single page. This plot is very useful for univariate and bivariate analysis.

Given below are the plots we created for categorical variables.

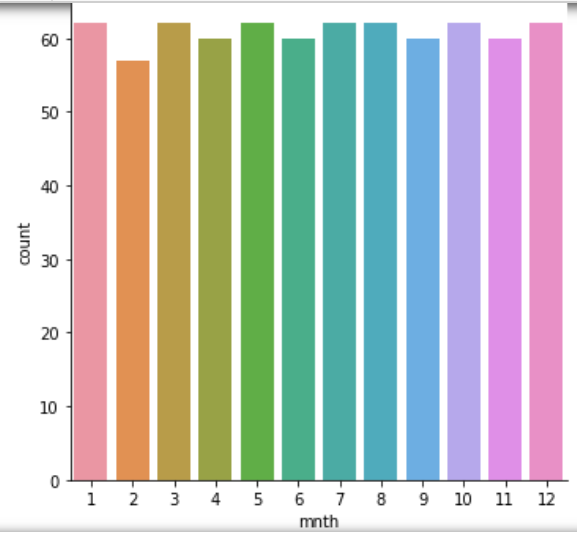
**Season**: We can see from the below plot that the count of bike hiring is high in the fall and low comparative in winter.



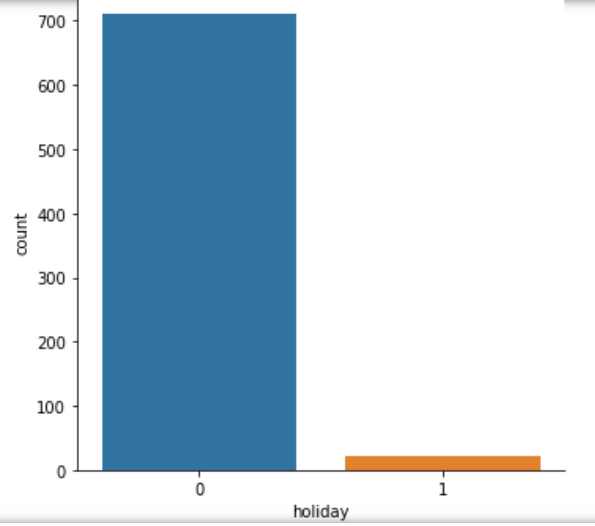
**Year**: Almost in both year the count is not varying that much.



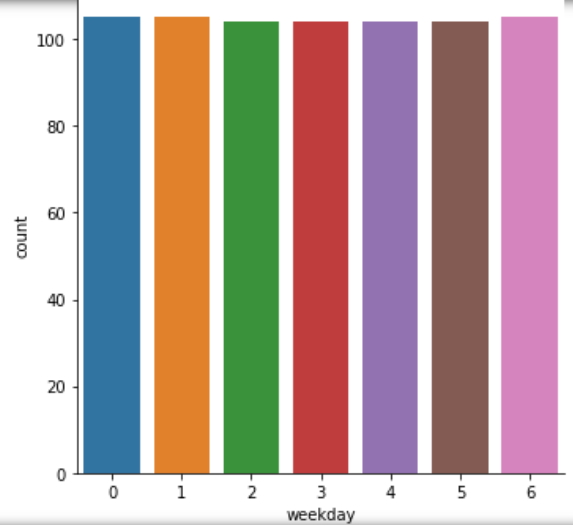
**Month**: We can see from the below plot that Feb, April, June, Sep, Nov are having less count than other months.



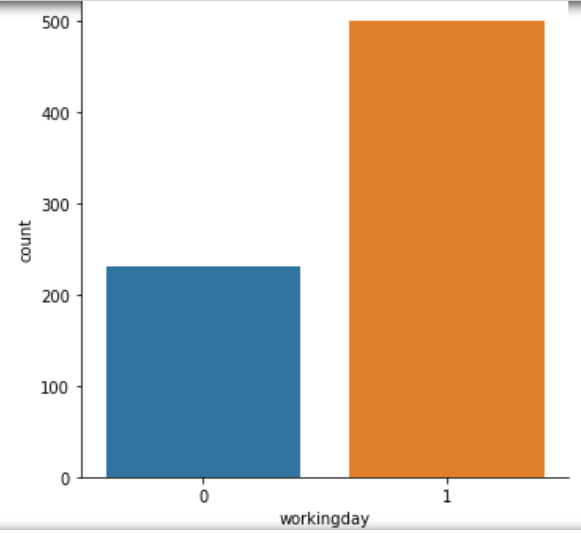
Holiday: In comparison to Holiday, we can see the count is too high if it is not a Holiday.



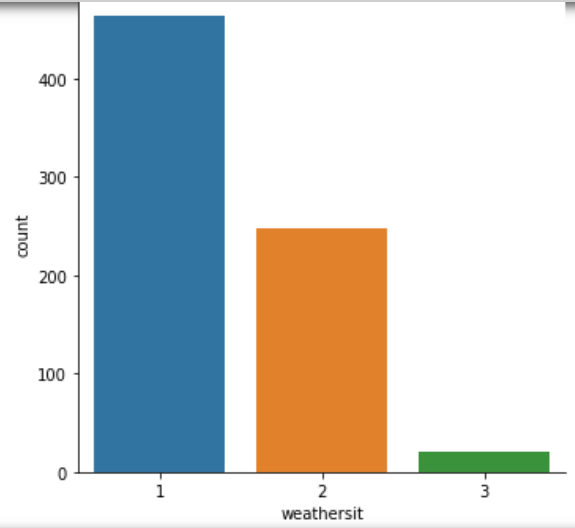
**Weekday**: We can see from the below plot that there is not that much variation depending upon the weekday.



**Working Day**: We have come to know that the count is comparatively too high if it a working day rather than a non-working day.



**Weathersit**: The count for “Clear, few clouds, Partly cloudy, Partly cloudy” is most, then the count is less for “Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds”.

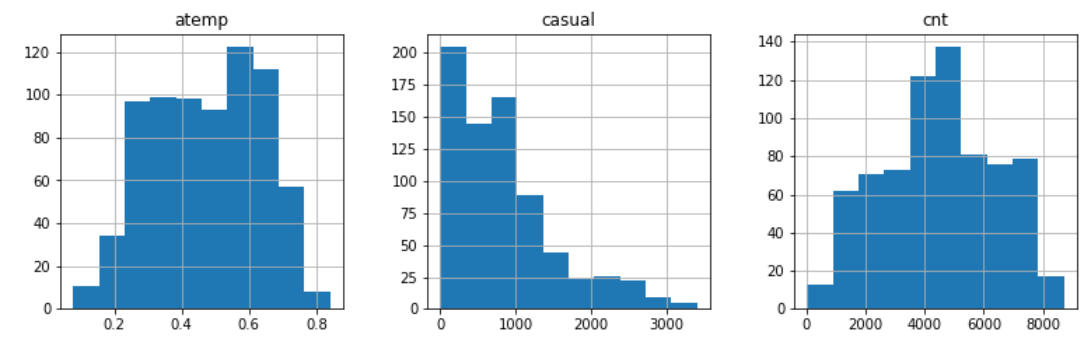


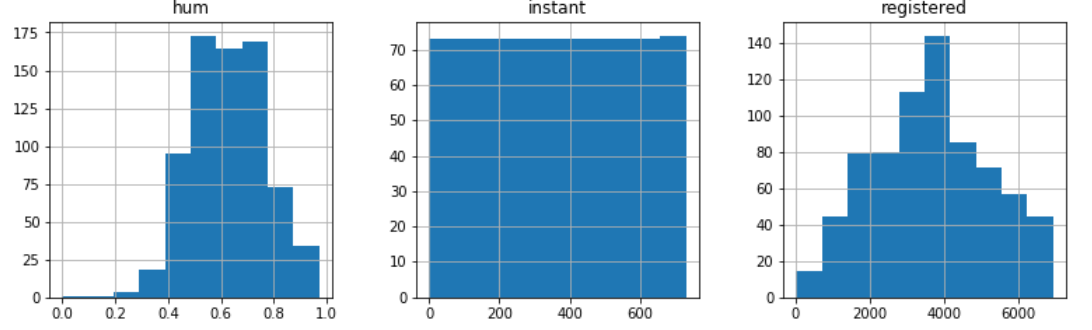
## **2.4.2 Distribution of Continuous Variables**

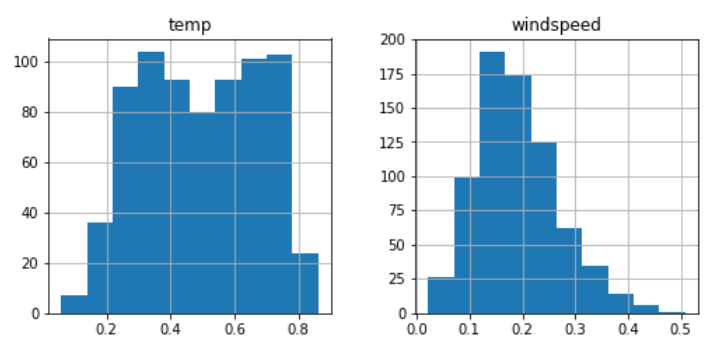
To see the distribution of continuous variables, we have used Histogram.

### **2.4.2.1 Histogram Plots:**

A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable.







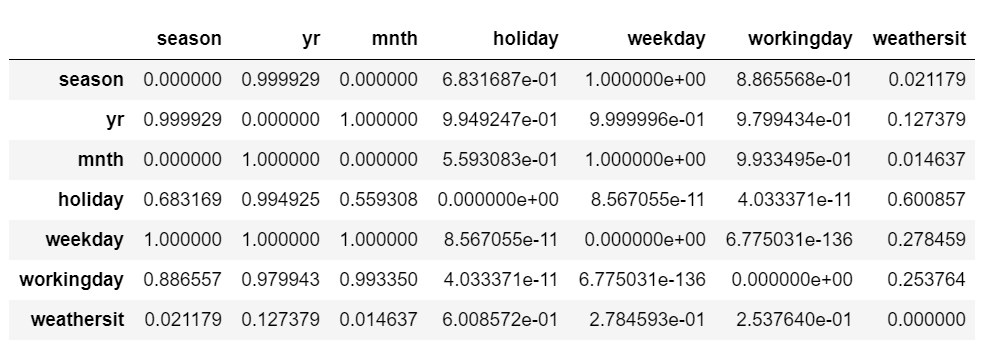
From the above plots we can see that none of our continuous variables are properly normally distributed. Hence, we need to do Normalization.

## **2.5 Variable Reduction/Feature Selection Techniques**

Before going to create any model, we need to check the variable importance so that we can reduce number of variables while making the model and choose only those variables which are having high importance.

### **2.5.1 Chi-Square test for Categorical variables**

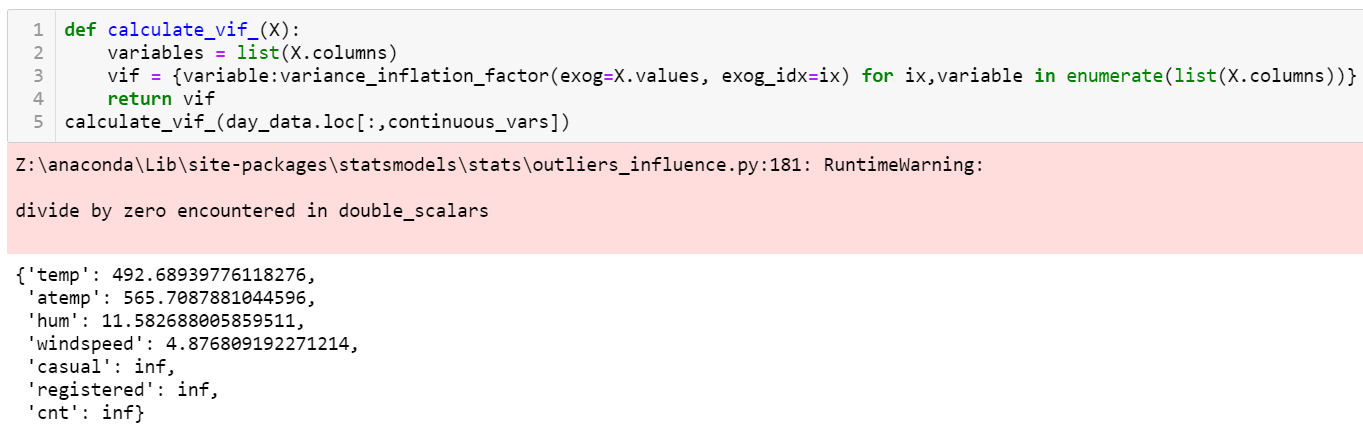
We often use the Chi Square test to view the dependencies between categorical variables before fitting the model to test and predict data. Pearson Chi square gives us the value of Chi square between each variable and we can eliminate the variables which are having less chi square value. Given below is the table of chi square values from all our categorical variables in dataset.



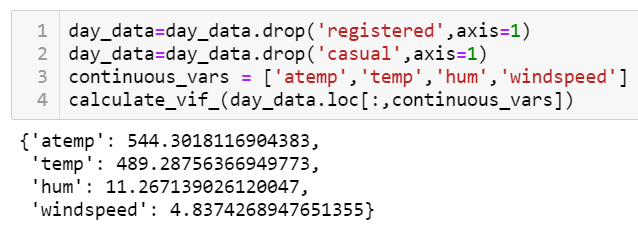
### **2.5.2 VIF to check multicollinearity**

The Variance Inflation Factor (VIF) is a measure of co-linearity among predictor variables. We generally use VIF as it can detect multicollinearity and we can drop the variables with high VIF value.Multicollinearity is a phenomenon in which two or more predictor variables (Independent variables) in a regression model are highly correlated, meaning that one can be linearly predicted from the others with a substantial degree of accuracy. VIF tells us by checking the tolerance of each predicters if any collinearity exists in our dataset. Then we can remove that variable. Given below are the result that shows first, there is a collinearity detected in our dataset and then we removed the higher VIF variables and checked if VIF is coming consistent throughout each predictor or not.

* From our first iteration, we can see that we have two variables “Registered” and “Casual” which is showing VIF value as infinity. Hence, we need to remove those two variables. We can easily make the conclusion that our target variable “cnt” is nothing but the summation of these two variables. Hence, we can remove those two variables.

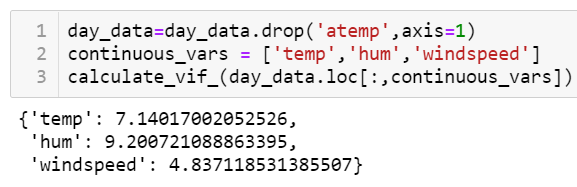


* The result from second iteration is showing below.



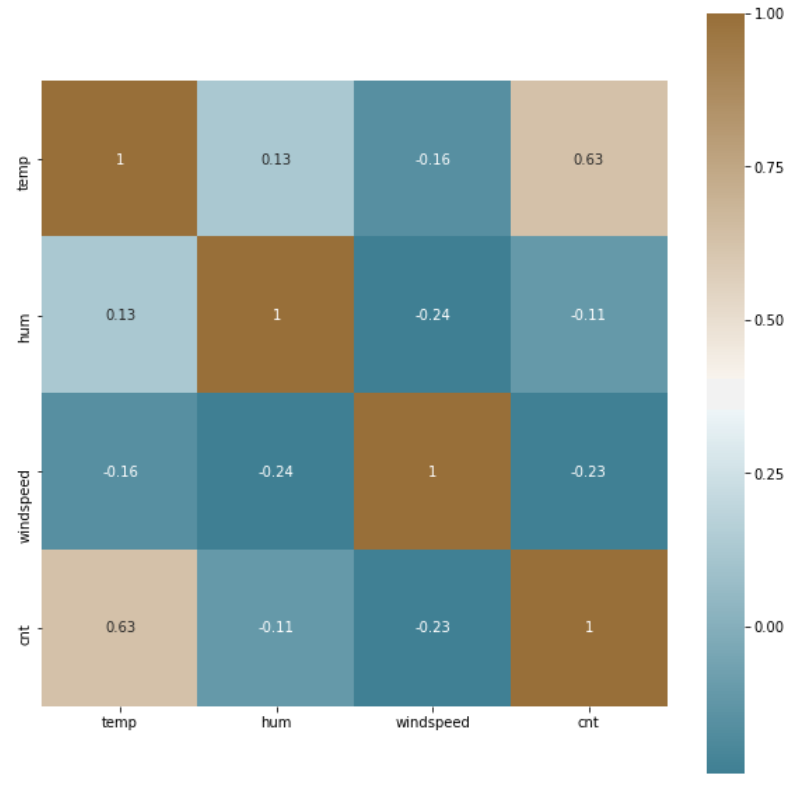
Still we can see high VIF value for “atemp” column which defines the multicollinearity exist in our dataset and hence, we need to remove the variable with high VIF value.

* In our third iteration, we can see we have received the VIF values within industry standard which is less than 10. Hence, we can conclude that we have finally able to independent from the problem of linearity and our model can predict well in case of any new variable arrive further. Given below is the result from third iteration.



### **2.5.3 After reduction checking the correlation**

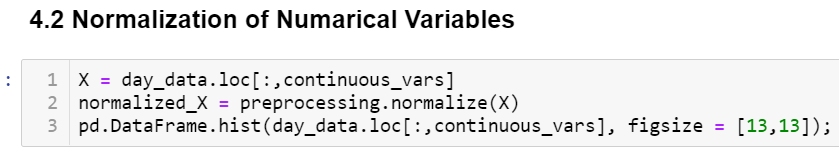
The Correlation coefficient measures the linear relationship between two datasets. Pearson’s Correlation requires that each dataset to be normally distributed. Like other correlation coefficient it also has range between +1 to -1. 0 implies there is no correlation, 1 implies there is a linear relationship where y will also increase with increment of the value of x. -1 denotes there is a linear relationship but in opposite direction.



After removal of lower importance variables or the variables which carries same information with other variables, we can see in the above correlation plot that we don’t have any variable with much correlation value.

## **2.6 Normalization of Continuous Variables:**

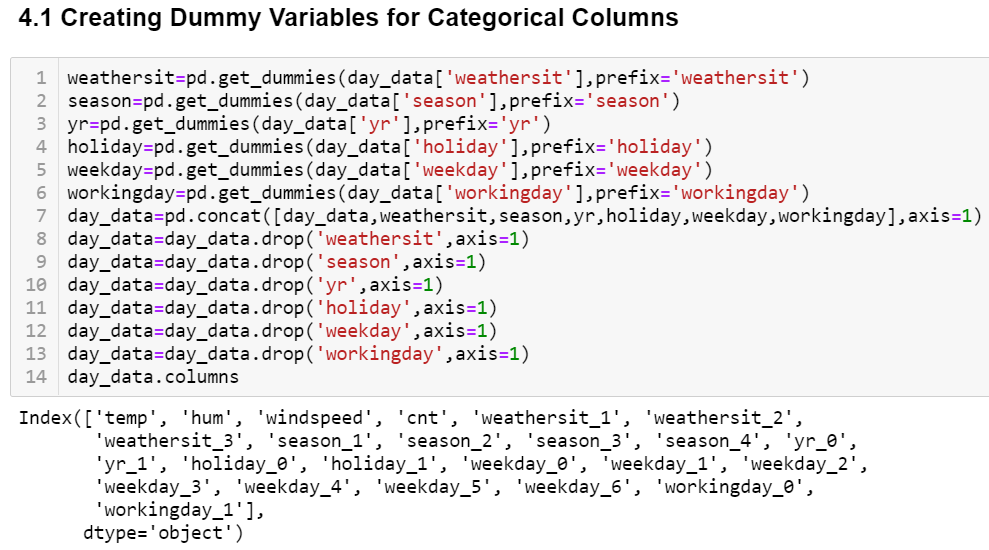
Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. Given below is the code we used to normalize the numerical variables to get a better matrix for model.



We have used Processing package to use the function normalize. Hereby passing the numerical column into the function, we received the normalize form. Hence, we have checked by plotting the histogram again and found it is showing much better format than previous.

## **2.7 Creating Dummy Variables for Categorical columns:**

For each factor variables, we have created dummy variables columns as many factors as the column have and dropped the original variables.

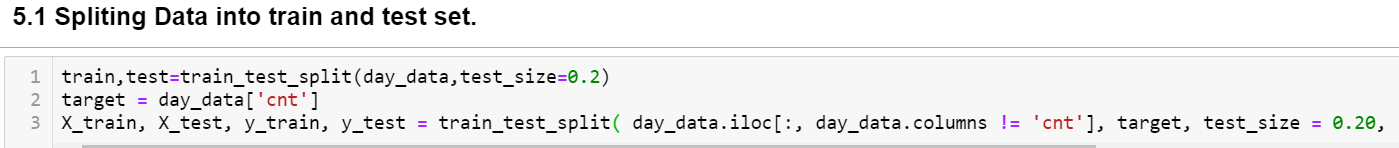


# **3. Model Building**

After cleaning our data with several pre-processing technique, now we will create few regression models to predict our target variable. As our target variable “Count” is a continuous variable so we will build Decision Tree, Random Forest and Linear Regression, KNN Algorithm, RandomizedCV, Ridge Regression, LASSO Regression, XGBoost. There are few regression metrics named Root Mean Square Error (RMSE), Co efficient of Determination R^2, MAPE (Mean Absolute Percentage Error), MAE (Mean Absolute Error) we have checked to select the final model.

## **3.1 Splitting data into train and test**

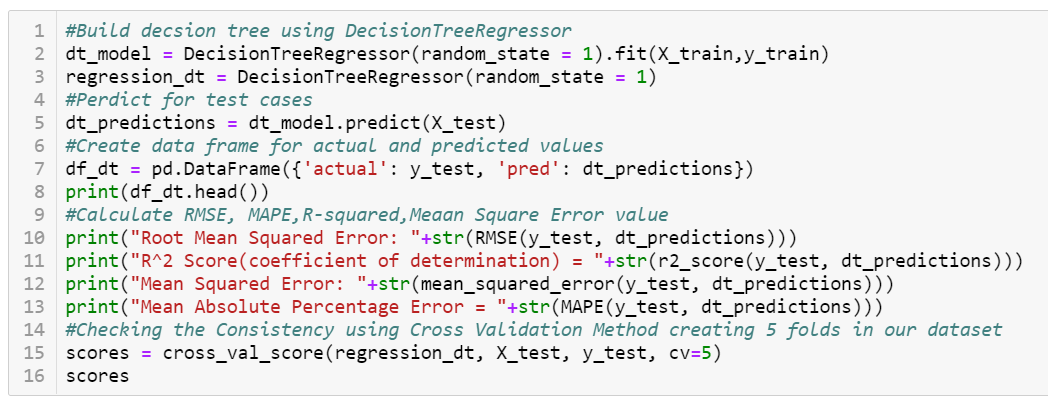
We have used the function train\_test\_split from sklearn.model\_selection package which Split data frame into random train and test subsets.



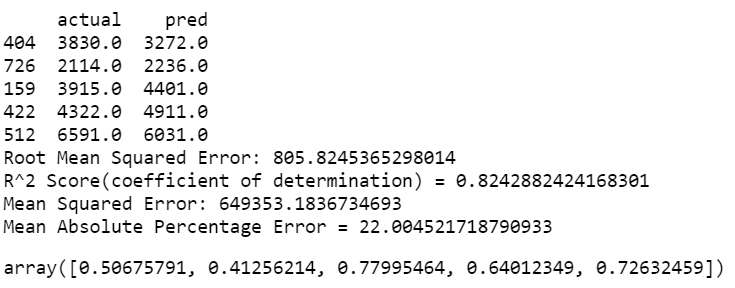
* In this function we have passed the dataframe without target variable as X, target variable as Y.
* We have defined the “test\_size” as 0.20 which denotes to split the data 20 percent into test and rest for train a model.
* Random\_state works to generate random seeds. If it is false then internally it use the np.random numpy random number generator.
* We have enabled the suffle parameter which basically shuffles the dataset before splitting.

## **3.2 Decision tree**

Decision tree is a supervised machine learning algorithm which is generally used for both regression and classification problems. Here, each node represents an attribute and each branch denote a rule or decision and each leaf is a outcome which is in our case target variable.  
Basically we are going to create a decision tree model which will predict the count variable of our test data.



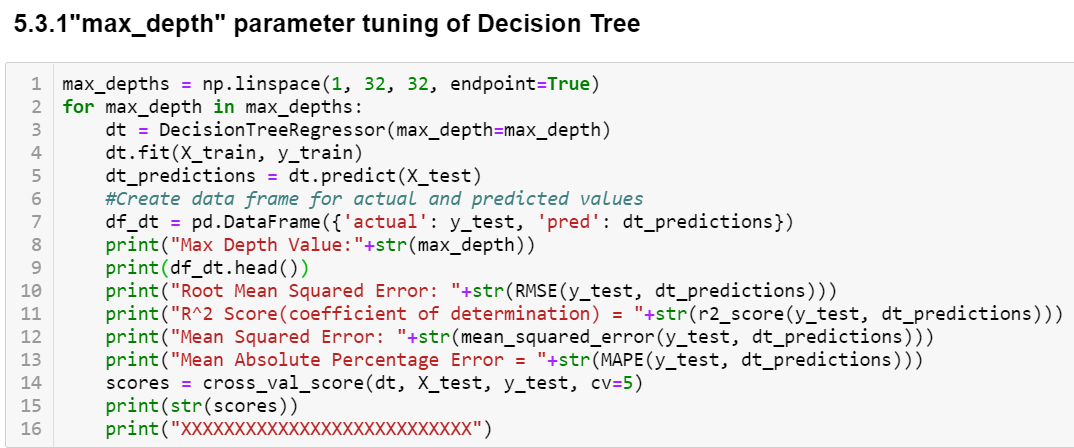
In the above snippet we have used the DecisiontreeRegressor function to fit the train data and then predict the test data outcomes using the decision tree model.



From the below outcome we can see the values of Decision Tree for different matrices. As we observed that our Cross-Validation Score ranged between 0.41 to 0.77. So, there is a part of dataset where there is a tendency of biasness or Overfitting of data. In order to reduce the biasness and to get a consistent score for cross validation, we need to tune the parameters of our decision tree.

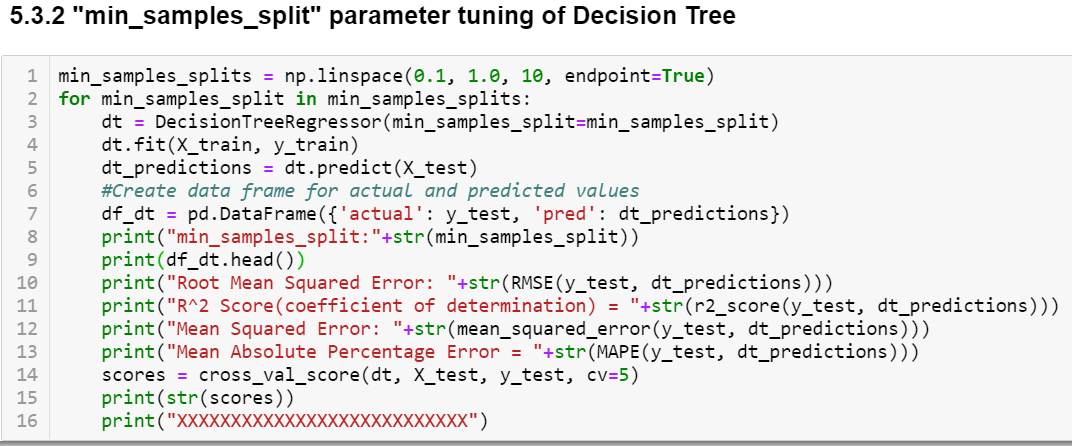
### **3.2.1 Tuning parameter of Decision Tree (max\_depth)**

This parameter defines the maximum depth of a decision tree. We have checked the outcomes for values of this parameter up to 32 and choose the best value in terms of comparing the regression error matrices.



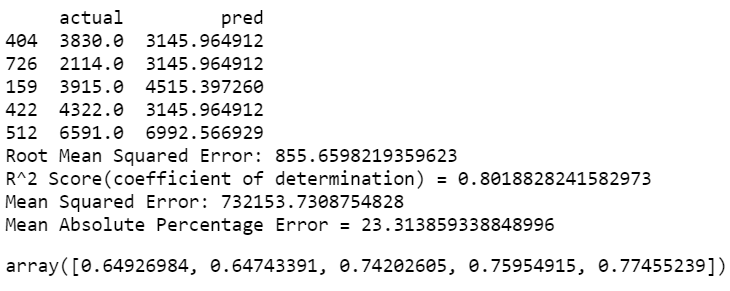
### **3.2.2 Tuning parameter of Decision Tree (min\_sample\_split)**

This parameter defines the minimum sample split allows to make rule-based decision. We have checked the values within range of 0.1 to 1.0 and chose the best value which gives lower error.



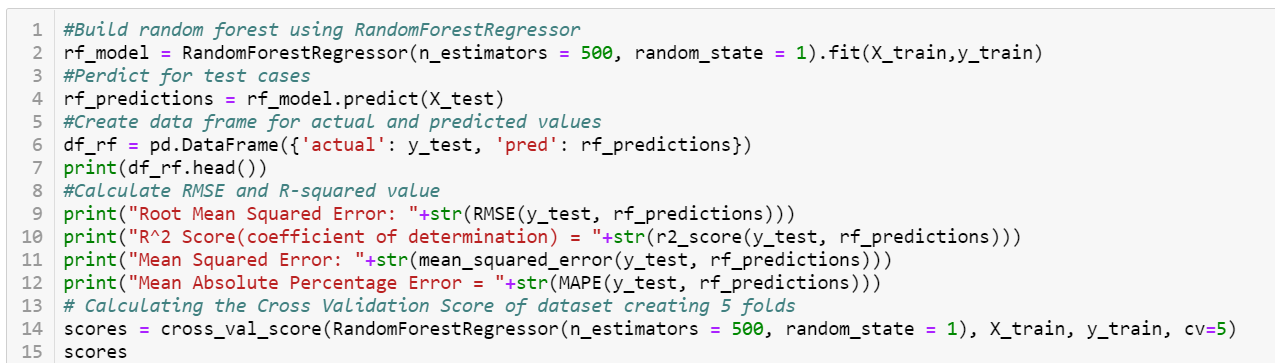
### **3.2.3 Decision Tree with optimum parameters**

After tuning few parameters, we build the final decision tree with optimum parameters and observed significant improvement in our result. We are getting a more consistent Cross validation score in each five validation sample sets.

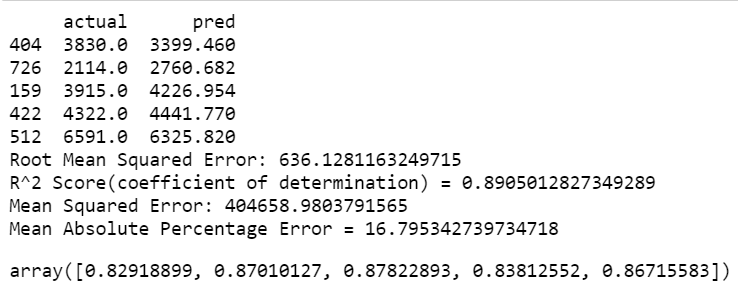


## **3.3 Random Forest Regression**

Random forest is also a supervised machine learning model which is a collection of Decision trees and merge them together to get a more accurate and stable prediction. The method of combining trees is known as an ensemble method. Ensemble is nothing but a combination of weak learners (individual trees) to produce a strong learner.



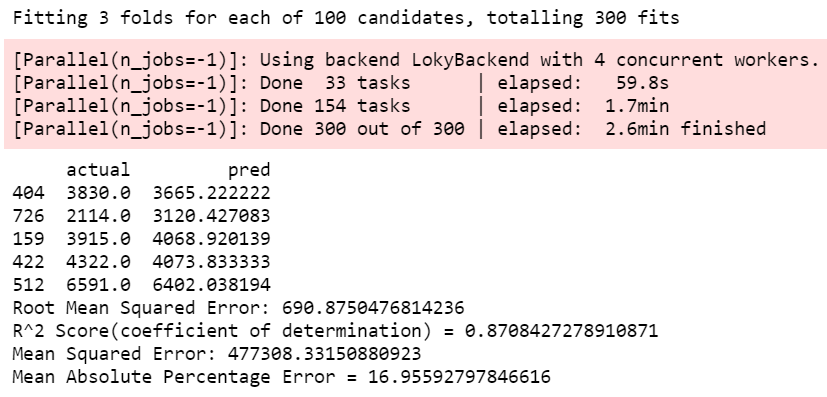
In the above snippet we have used RandomForestRegressor function from Scikit Learn package. We have fit our train data to build the model and then fit the test data to predict the target variable i.e. the count of bike.



Above screenshot is the result we get from the Random Forest and checked the consistency of Cross Validation score on 5 validation datasets. Then we have tuned the values to get the best result and to minimize the errors.

### **3.3.1 Randomized CV search technique to get best parameters of Random Forest**

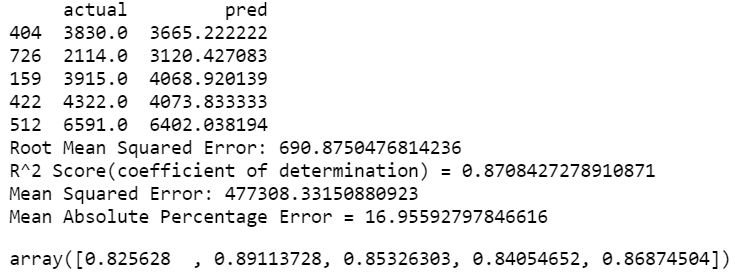
To use RandomizedSearchCV, we first need to create a parameter grid to sample during fitting. On each iteration, the algorithm will choose a difference combination of the features. The benefit of a random search is that we are not trying every combination but selecting at random to sample a wide range of values. Given below is the result of our Randomized Search CV. From this model we will further iterate the Random Forest with best parameter suggested by this grid.



### **3.3.2 Tuning Random Forest with Best Parameters**

We got the best parameter for creating our optimum Random Forest from the Grid Search and now we will create the Random Forest which will provide us the less error comparative to the last time.

Given below is the result of our Random Forest with tuned parameter.

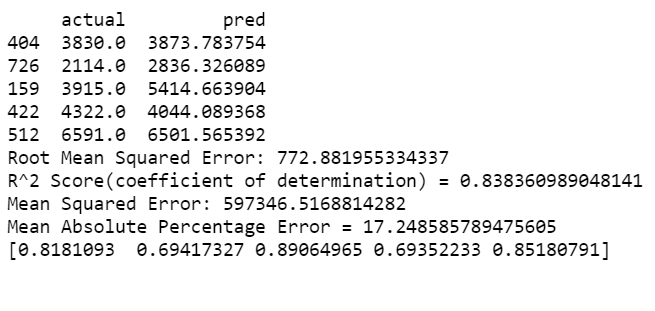


We can see now our model is giving much consistent result in all validation sets.

## **3.4 Linear Regression**

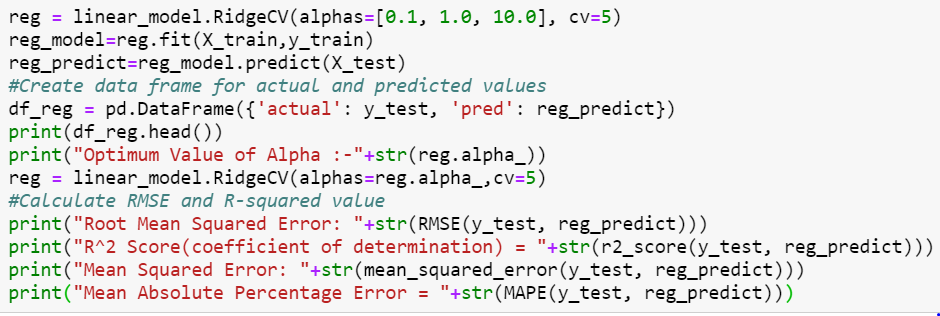
Linear Regression is a model to find a relationship between one or more feature (independent variables) and a continuous target variable (dependent variable). When there is only feature it is called Uni-variate Linear Regression and if there are multiple features, it is called Multiple Linear Regression. Here The line for which the error between the predicted values and the observed values is minimum is called the best fit line or the regression line. These errors are also called as residuals.

Given below is the result of Linear regression for our dataset.

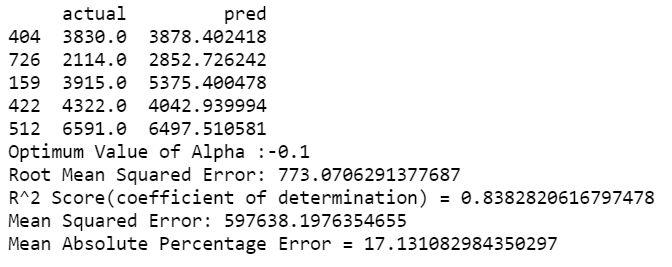


## **3.5 Ridge Regression:**

It is a regulation technique which use L2 regulation on top of Linear Regression and we can see that there is a slight improvement in our model because the value of the R-Square has been increased. Note that value of alpha, which is hyperparameter of Ridge, which are not automatically learned by the model instead setting manually.

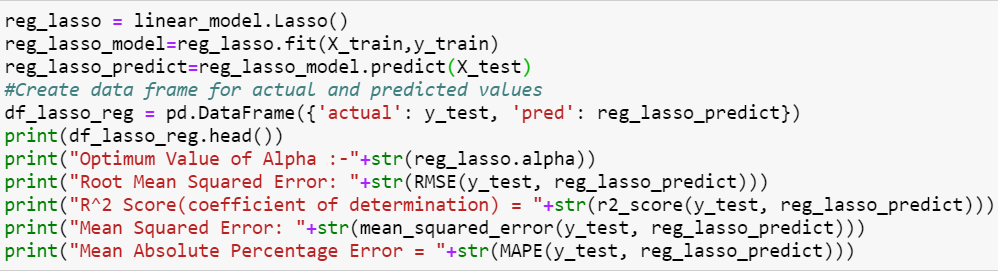


we increase the value of alpha, the magnitude of the coefficients decreases, where the values reach to zero but not absolute zero. We can see the result which defines much better than Linear Regression as it handles the multicollinearity by reducing the coefficient shrinkage.

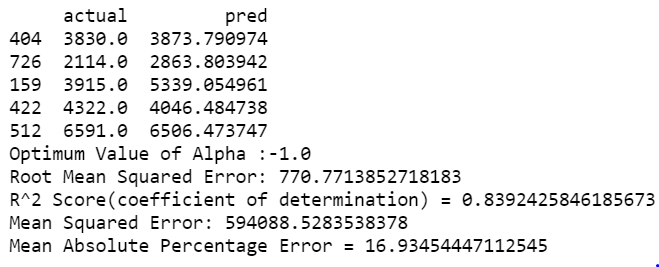


## **3.6 LASSO Regression**

LASSO (Least Absolute Shrinkage Selector Operator), is similar to ridge. It is a L1 regulation technique. In our case, we can see LASSO is giving much better result than Linear and Ridge.

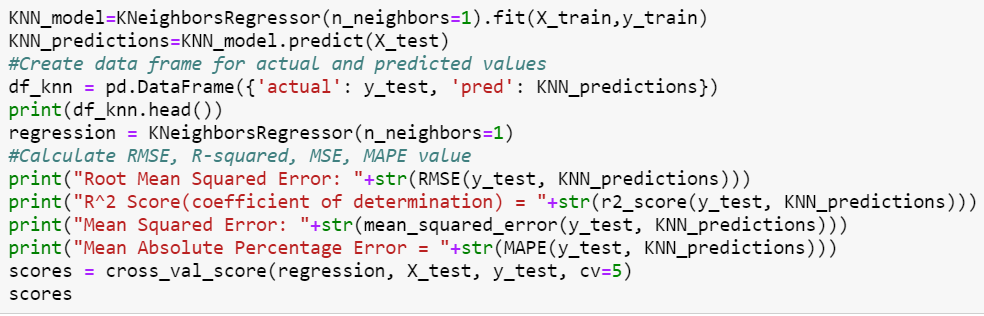


In Ridge, coefficients were approaching towards zero, but in case of lasso, even at smaller alpha’s, our coefficients are reducing to absolute zeroes. Therefore, lasso selects the only some feature while reduces the coefficients of others to zero. This property is known as feature selection and which is absent in case of ridge.

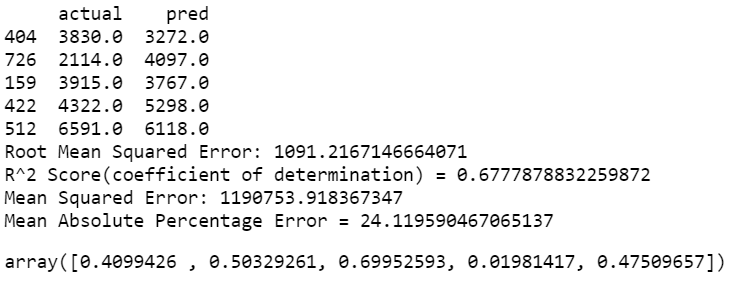


## **3.7 KNN Algorithm for Regression**

KNN can be used for both classification and regression problems. The algorithm uses ‘**feature similarity**’ to predict values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set.

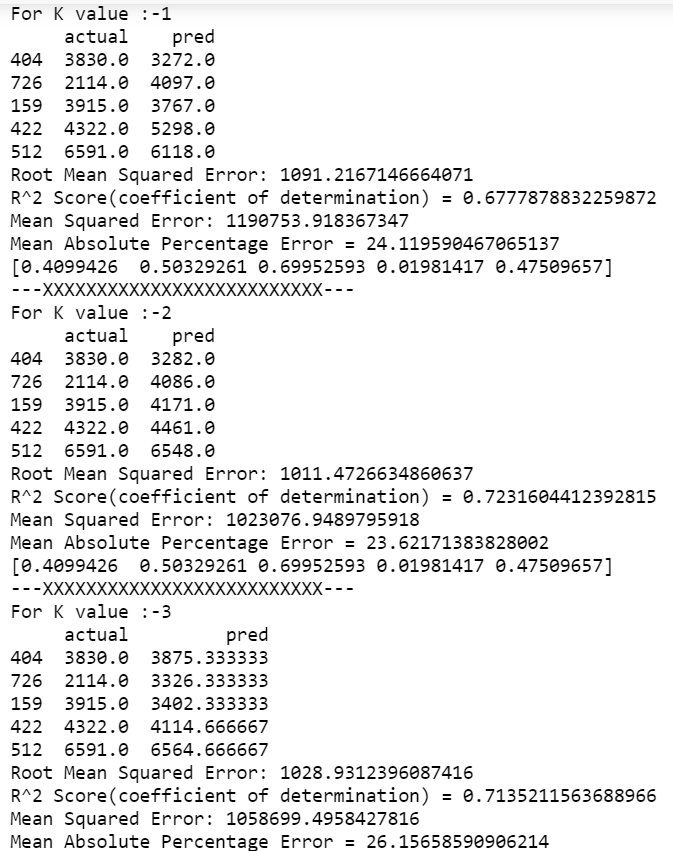


We have checked the score for all validation sets and found it is not consistent. Hence, there is a chance of Biasness in our dataset. We need to tune the parameters for KNN to get consistent values. Given below is the result of KNN.



### **3.7.1 Parameter tuning for KNN**

We have tuned the number of neighbours in KNN regression within range of 1 to 5 and choose the value which gives less error rate and choose that value only to make the final KNN model. The optimum n\_neighbours value is 2.



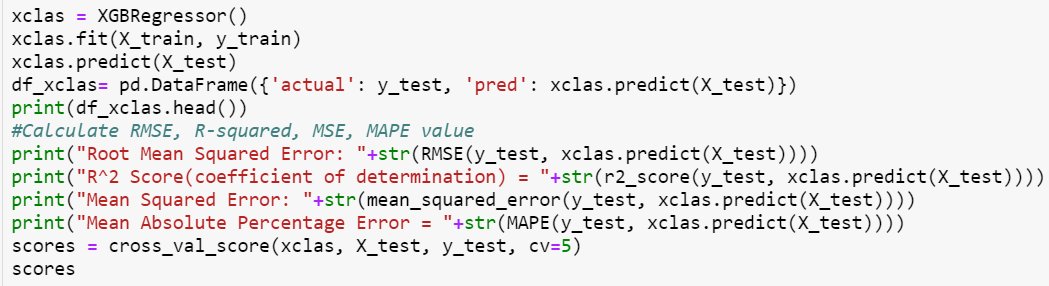
### **3.7.2 Building KNN with best K value**

After tuning the parameter, we have built the model with optimum result and achieve the consistency in result. Given below is the final model’s outcome.

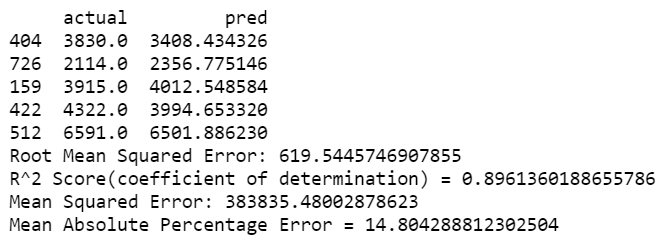


## **3.8 XGBoost**

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. XGBoost provides a wrapper class to allow models to be treated like classifiers or regressors in the scikit-learn framework. This means we can use the full scikit-learn library with XGBoost models. The XGBoost model for regression is called **XGBRegressor**. We can create and fit it to our training dataset. Models are fit using the scikit-learn API and the **model.fit()** function.



We have received much better result than other algorithms and it is consistent through out all validation sets. Given below is the result of XGBoost.

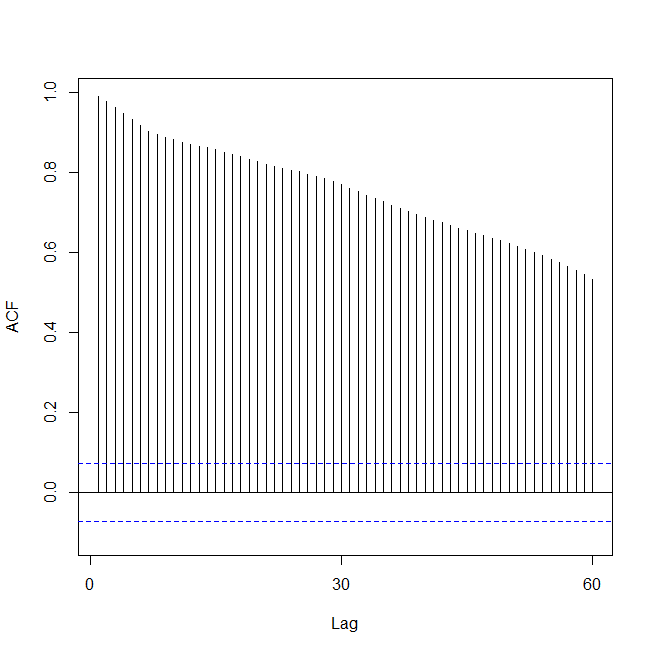




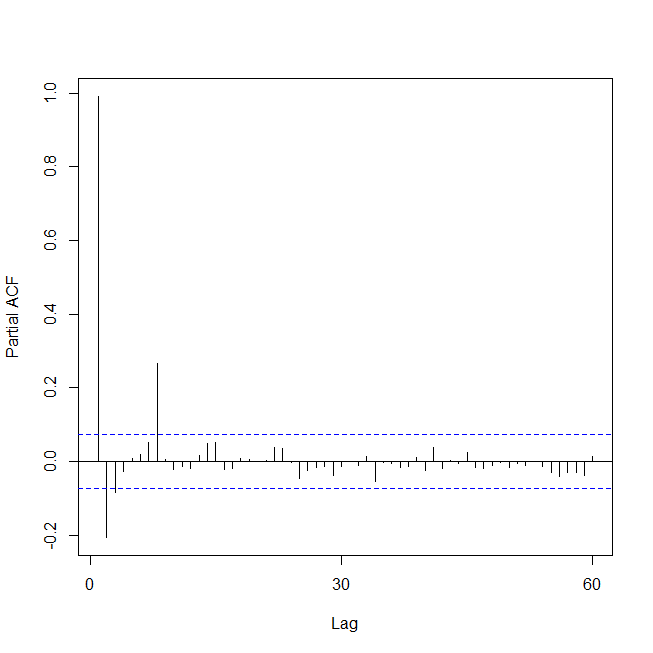
## **3.9 ARIMA**

ARIMA (autoregressive integrated moving average) is a commonly used technique utilized to fit time series data and forecasting. It is a generalized version of ARMA (autoregressive moving average) process. Three numbers p, d and q specify ARIMA model and the ARIMA model is said to be of order (p, d, q). Here p, d and q are the orders of AR part, Difference and the MA part respectively.AR is called “Auto Regression” and MA is called “Moving Average”. AR and MA- both are different techniques to fit stationary time series data. ARMA (and ARIMA) is a combination of these two methods for better fit of the model.

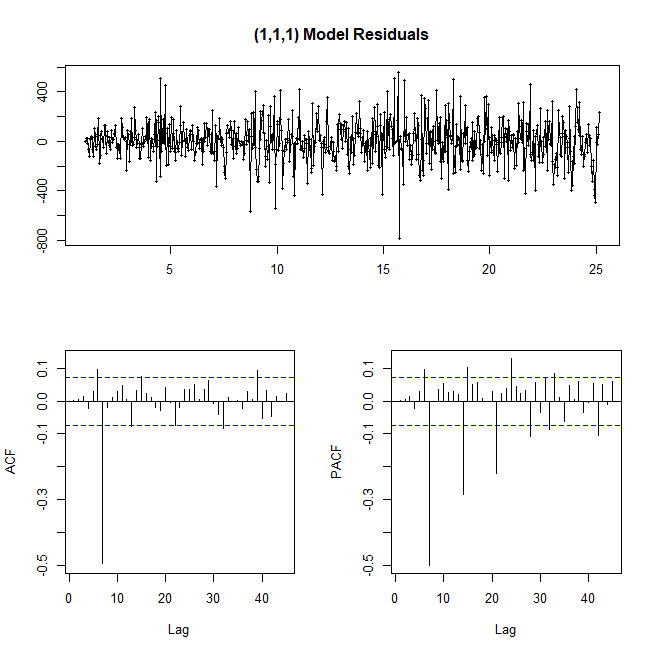
**Given below is the plot for ACF MA Component:**



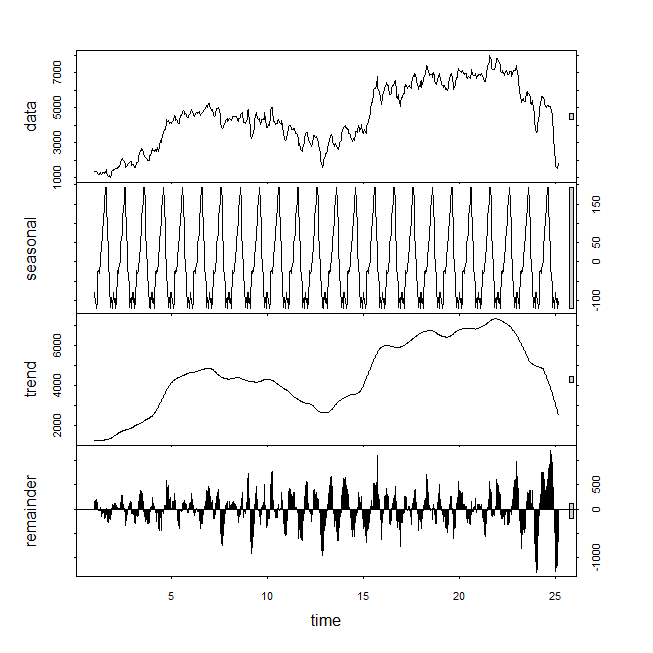
**PACF MA Component:**



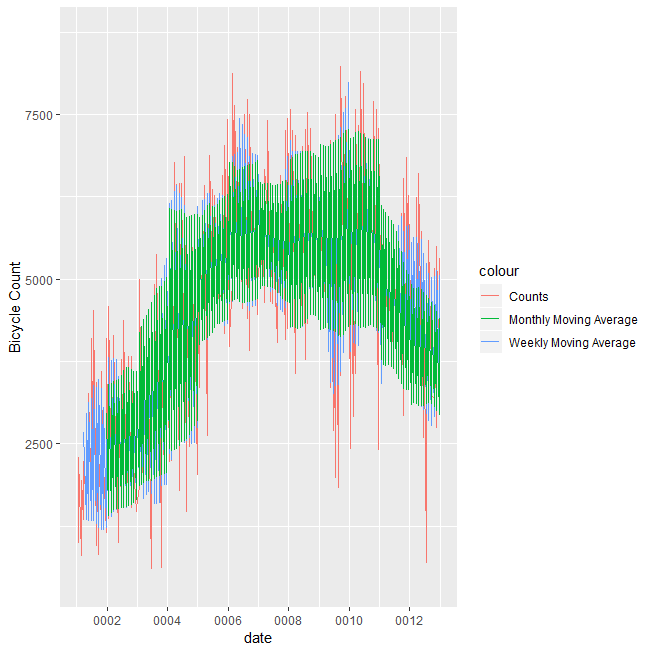
**ARIMA Residuals:**



**Time Series Decomposition:**



**Prediction Moving Average:**

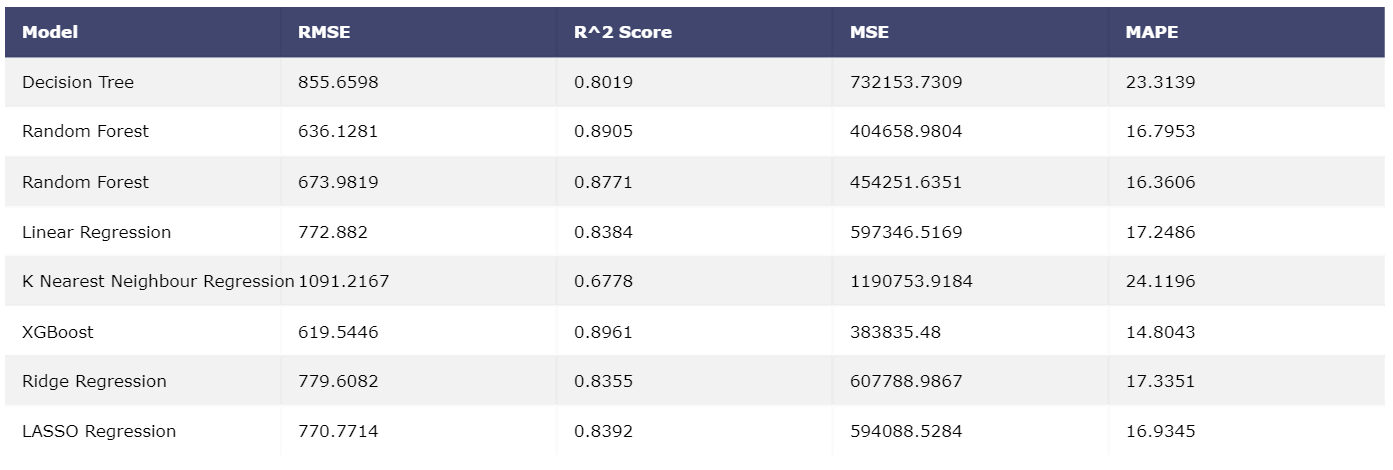


**Time Series Plot:**



## **3.10 Model Selection and output tables**

We have calculated several Regression matrices and plotted the output of every model in a table.



### **3.10.1 RMSE**

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are.

### **3.10.2 R-Squared value**

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

R-squared is always between 0 and 100%:

* 0% indicates that the model explains none of the variability of the response data around its mean.
* 100% indicates that the model explains all the variability of the response data around its mean.

The higher the R-squared, the better the model fits our data. We use R square in case of precise precision.

### **3.10.3 MSE**

The mean squared error or mean squared deviation of an estimator measures the average of the squares of the errors—that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss.

### **3.10.4 MAPE**

The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics, for example in trend estimation, also used as a loss function for regression problems in machine learning. Lower the value of MAPE indicates better accuracy to the model.

## **3.11 Key Points:**

### **3.11.1 K Fold Cross Validation**

Cross-validation is a statistical method used to estimate the skill of machine learning models. That k-fold cross validation is a procedure used to estimate the skill of the model on new data.

It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

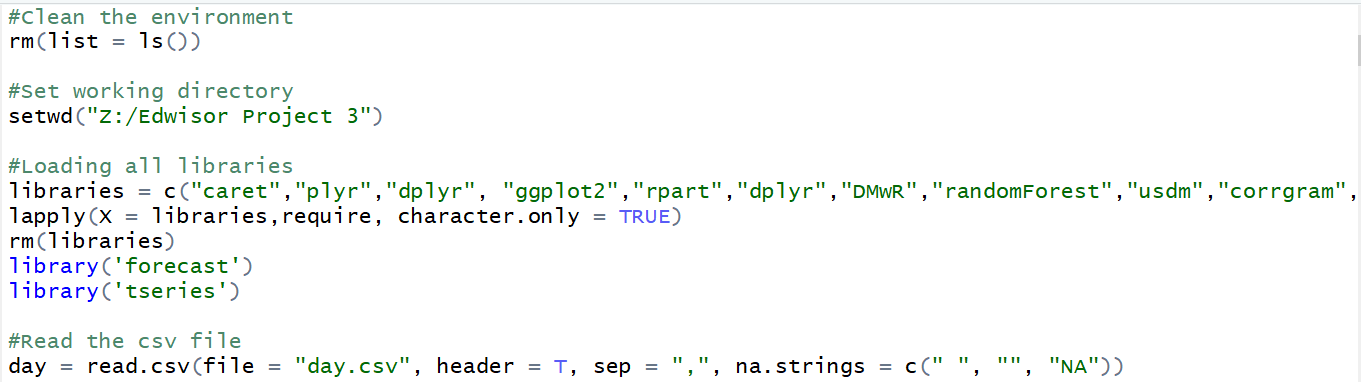
### **3.11.2 Exception Handling in python**

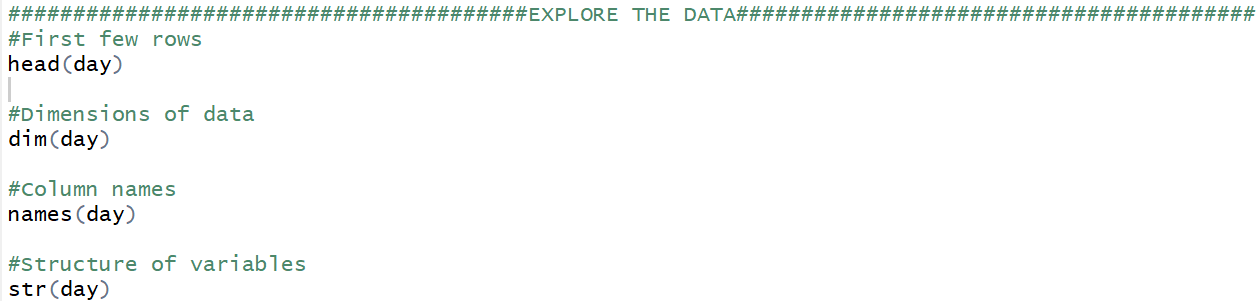
In Python, exceptions can be handled using a try statement. A critical operation which can raise exception is placed inside the try clause and the code that handles exception is written in except clause. It is up to us, what operations we perform once we have caught the exception.

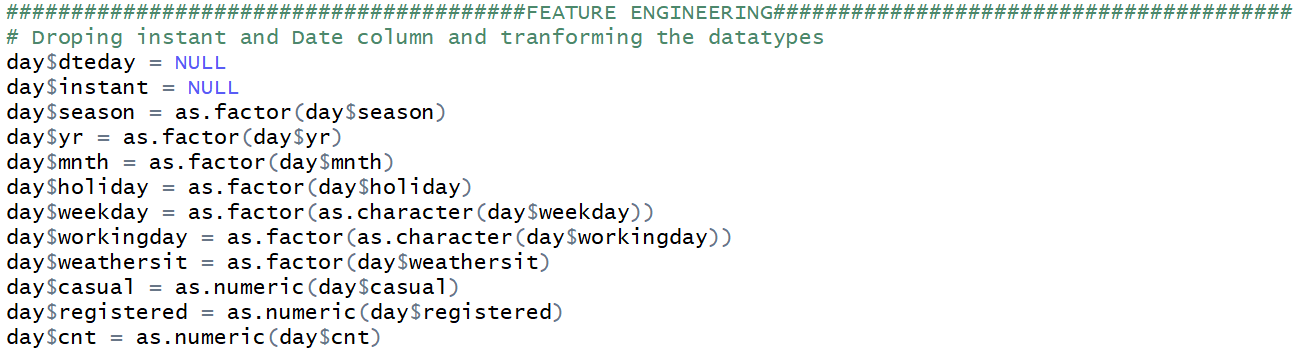
## **3.12 Conclusion**

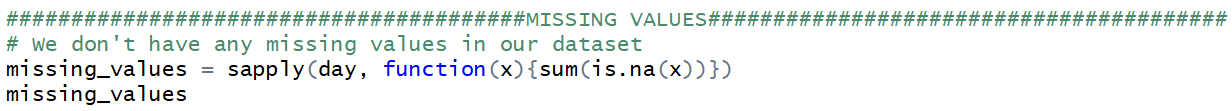
As a conclusion, after observing each matric we can go for XGBoost which is giving better result than other prediction models.

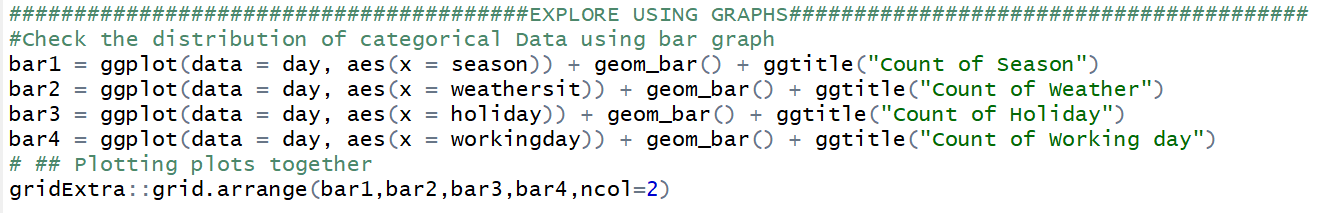
# **R- Code:**

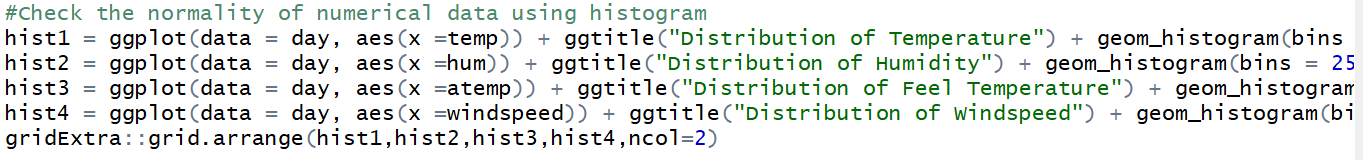


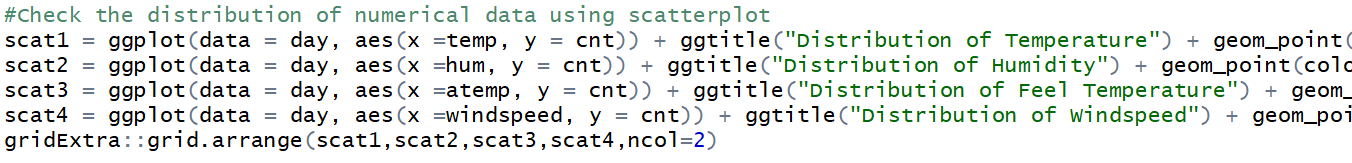


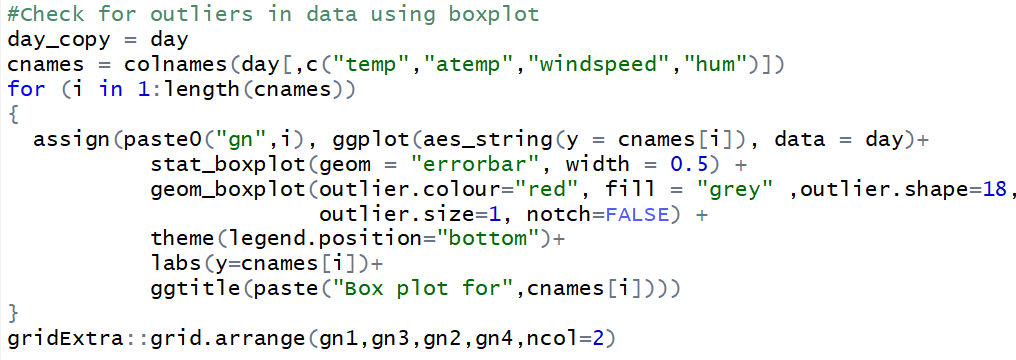


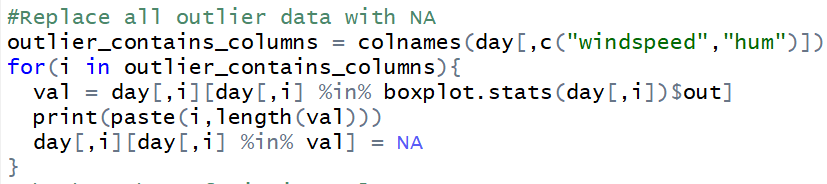


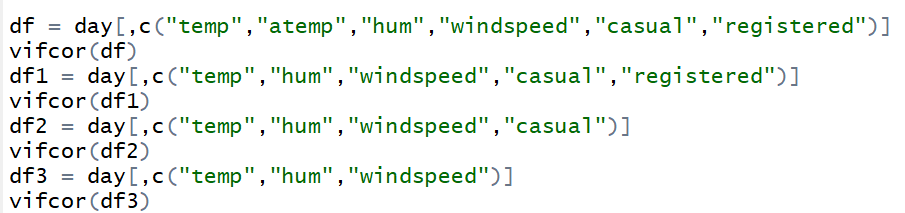


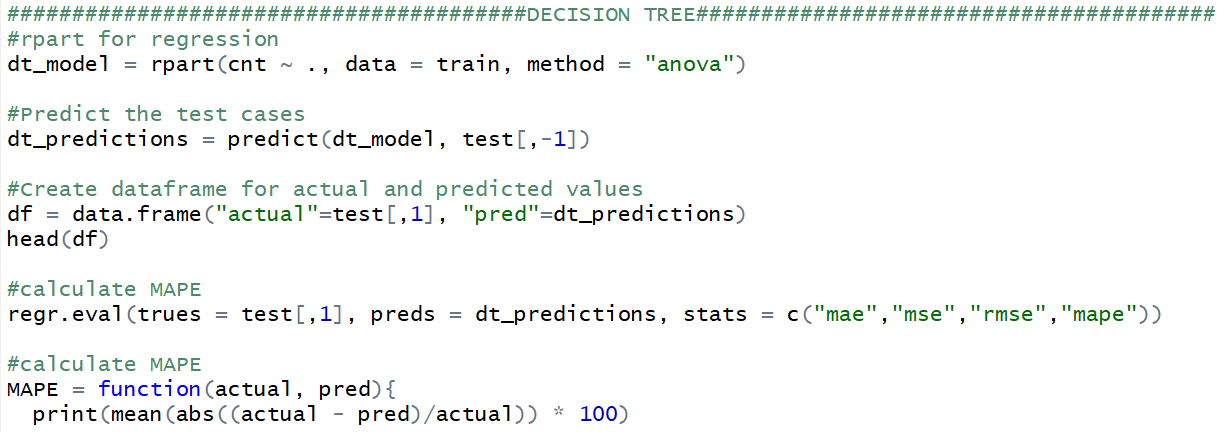


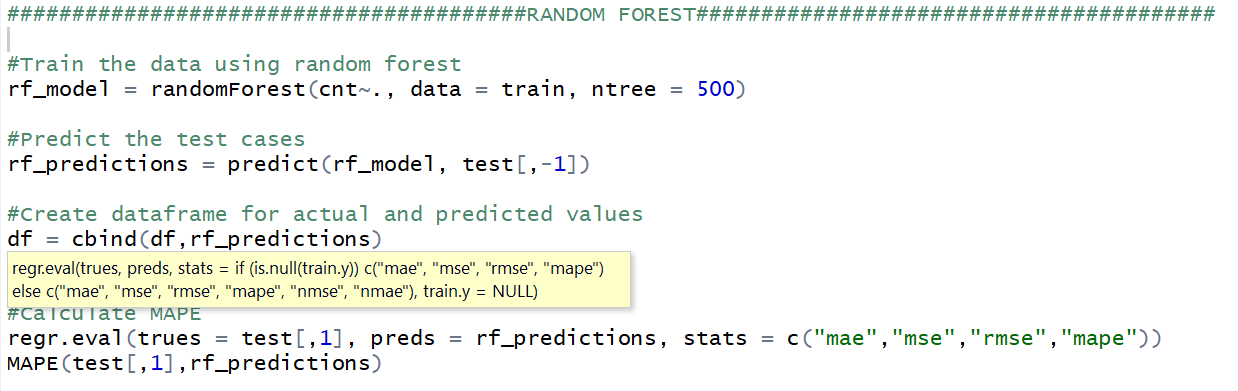


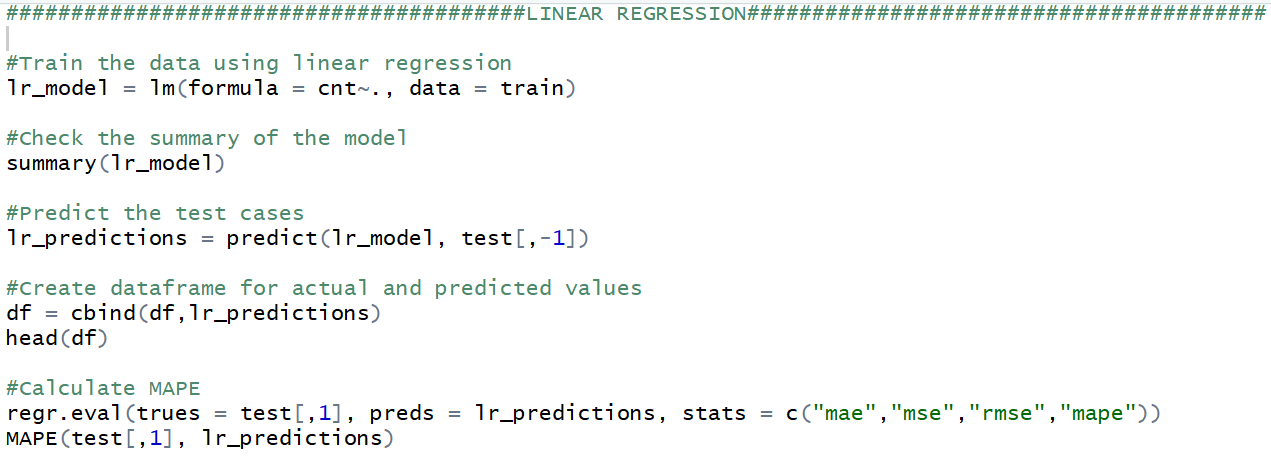


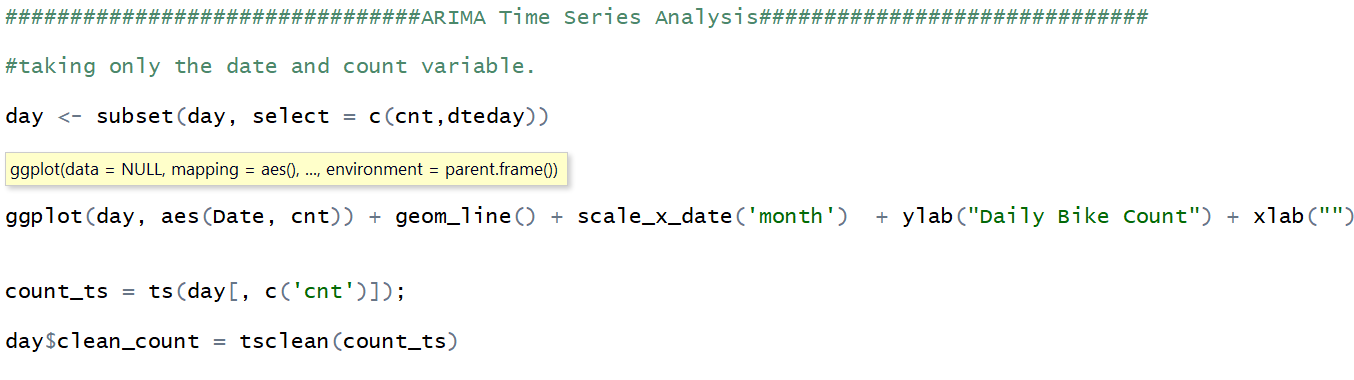


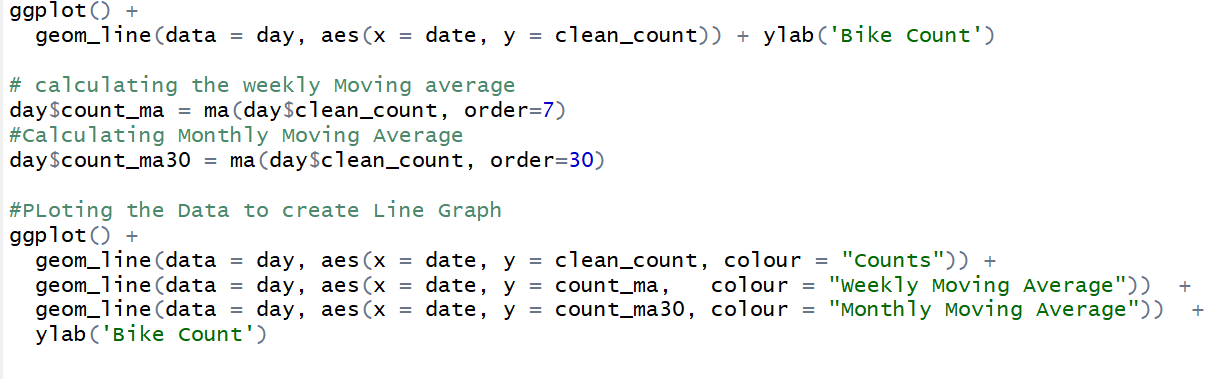


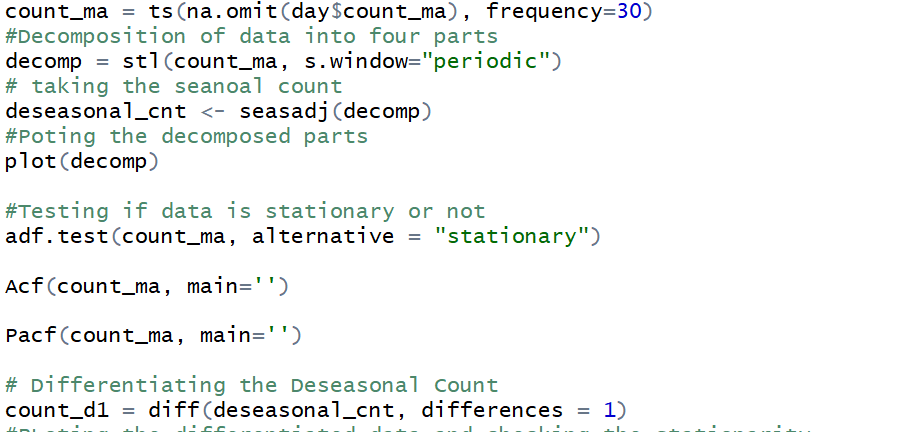


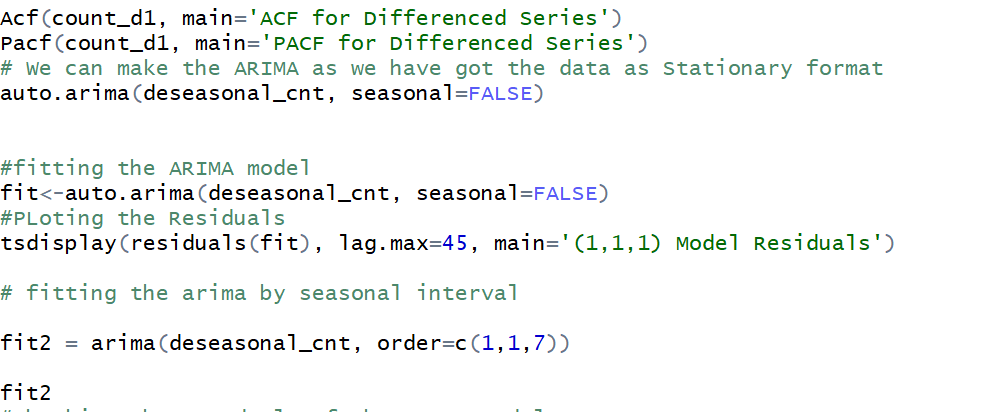


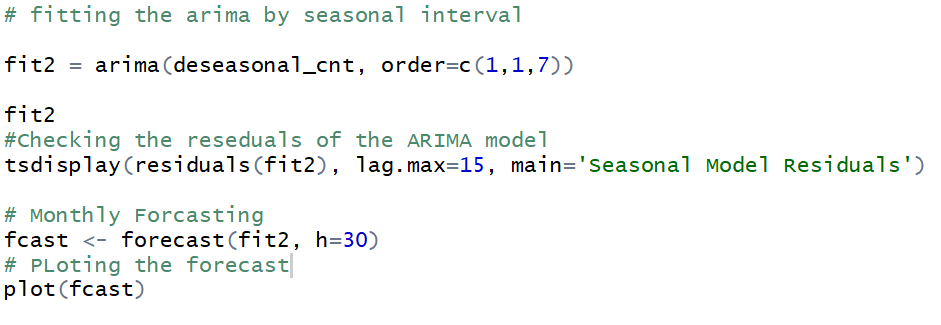












# **Additional Plots:**

