Predicting Daily Bike Rental Count

R. B. Borate

15 April 2020

INDEX

CONTENTS **PAGE NO.**

CHAPTER 1

INTRODUCTION

1.1 Problem Statement:

The objective of this project is predication of daily bike rental count based on the environmental and seasonal settings. So we need to predict the count of bikes to be rented daily based on the parameters given in the data. We will use different Machine Learning models to fulfill our objective. We will use the past data given to create Machine Learning model for this and use it to predict the count of bikes to be rented in the future.

1.2 Data:

We will build the Regression model to predict the count of the bike to be rented daily. The past data given to us is shown below in some samples. It contains total 731 observations & 16 variables from which 15 are dependant and last one is our target variable.

Table.1. Given data sample with top 5 observations. (Columns 1- 8)

instant	dteday	season	yr	mnth	holiday	weekday	workingday
1	01/01/2011	1	0	1	0	6	0
2	02/01/2011	1	0	1	0	0	0
3	03/01/2011	1	0	1	0	1	1
4	04/01/2011	1	0	1	0	2	1
5	05/01/2011	1	0	1	0	3	1

Table.1. Given data sample with top 5 observations. (Columns 9- 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

From the data as we can see it contains combination of numeric and categorical Variables which are independent variables and “cnt” variable is actually our target variable.

Table 3. list of independent or predictor variables

Sr. No.	Variable Name
1	instant
2	dteday
3	season
4	yr
5	mnth
6	holiday
7	weekday
8	workingday
9	weathersit
10	temp
11	atemp
12	hum
13	windspeed
14	casual
15	registered

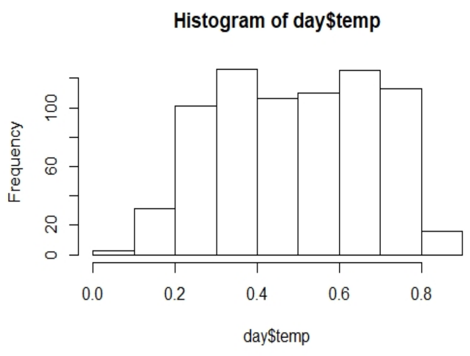
CHAPTER 2

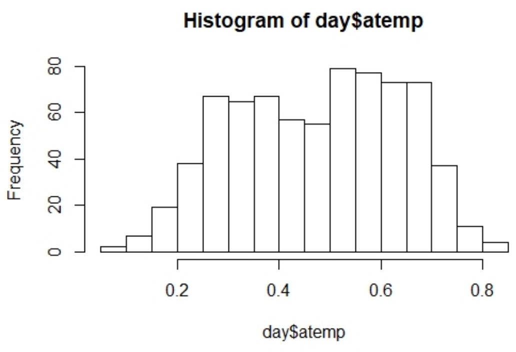
METHODOLOGY

2.1 Exploratory data analysis:

At first we have to set the working directory and load the libraries which are required to perform different operations. So we have uploaded the libraries and loaded the data. Now we will do the exploratory analysis of data that is nothing but visualizing the data and converting the data according to our requirement. It contains exploring the data, cleaning and visualizing with help of some plots and graphs. So after looking to the data we can easily see that it is a combination of categorical and numeric data. But in the data categories of the categorical variables are converted to some binary numbers (e.g. 1 , 2 etc) and because of that those variables data type is changed to ‘int’ so we have converted it to ‘factor’. Also ‘instant’ variable is just a index of data points from 1 to 731 so it does not provide any important information about the target variable so we have removed it from the data.

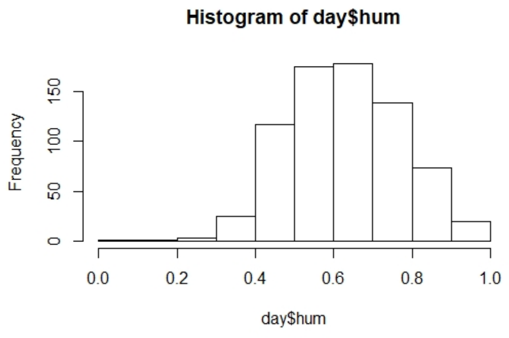
Now to visualize then data we can use different plots and graphs. We have used histogram, bar plot and scatter plots to visualize our data w.r.t to target variable. We have plotted the Histograms of all numeric variables to see the distribution of data in that particular variable. The plots are shown below,

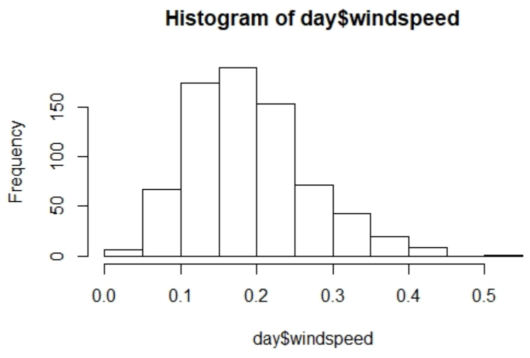




A

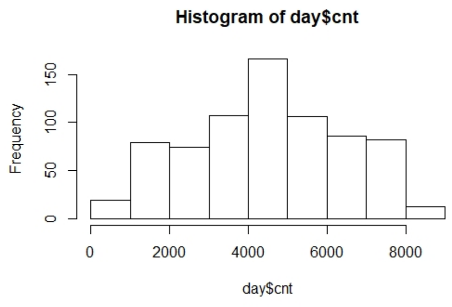
B





C

D

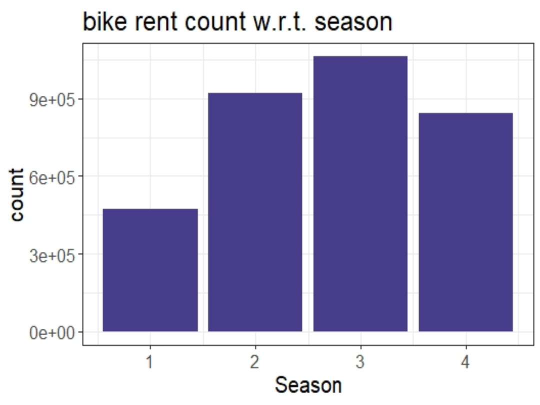


E

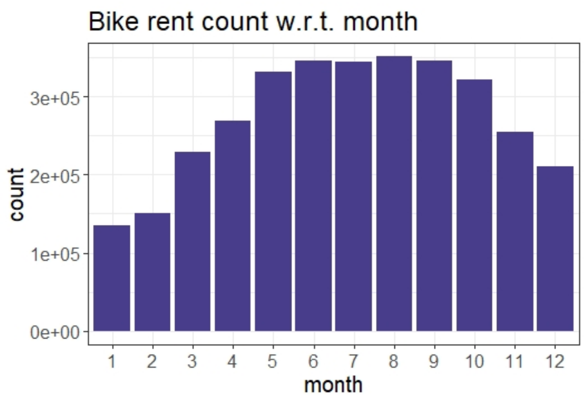
Fig. 2.1 Histograms of Numeric variables

As we can see the figures fig 2.1 A , B and E showing the nearly normal distribution but in fig 2.1 C and 2.1 D show some skewness in the data because the distribution is moved partially to the right and partially to the left in the 'hum' and 'windspeed' respectively. From this we can say that variables may contain outliers.

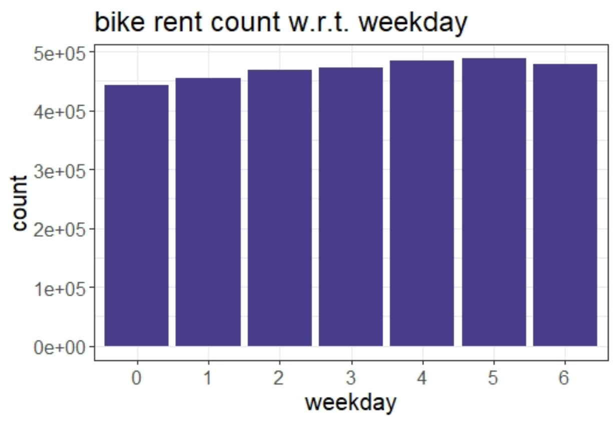
Bar graphs to see effect of categorical variables on target variables.



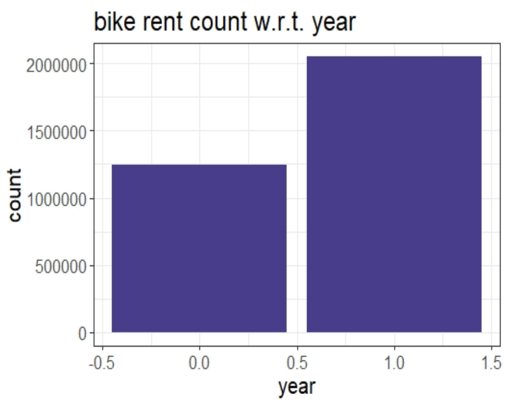
A



B



C

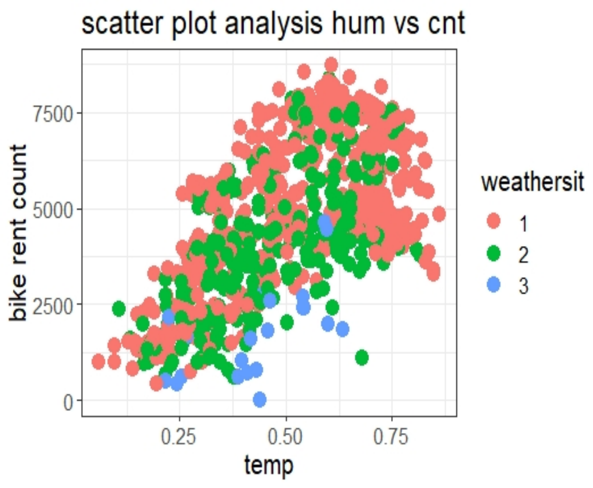


D

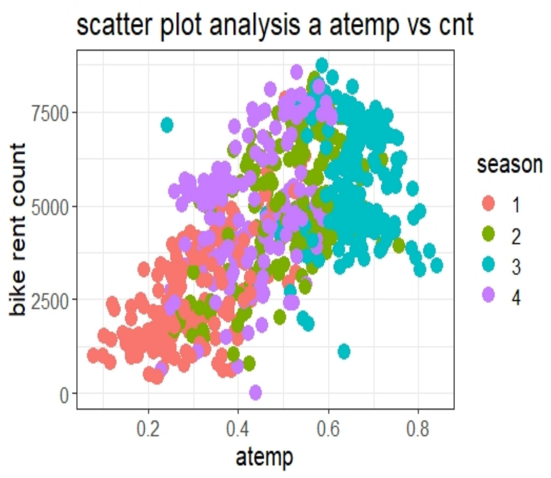
Fig. 2.2 Bar graphs of categorical variables vs Target variable

We can see in fig 2.2 A. that count of bike rented is very high in fall season as well as high in summer season. Also in fig 2.2 B. we can see that count is high in months of May, June, July, August and September. From fig 2.2 C. we can say that there is no much effect of weekday on count because it is high for all days. From fig 2.2 D. we can see count is high for year 2012.

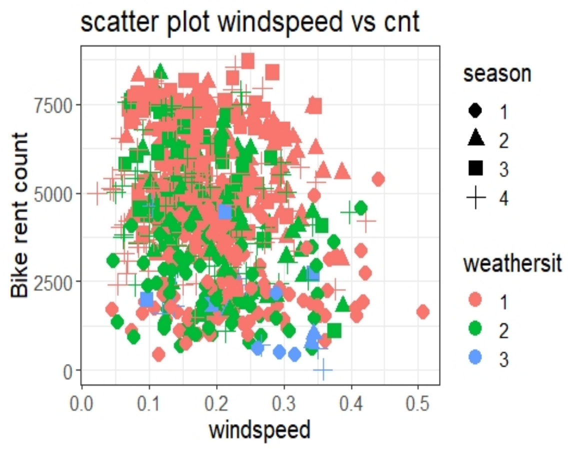
Now we will see some scatter plots of numeric variables w.r.t. our target variable and using another categorical variable in same plot.



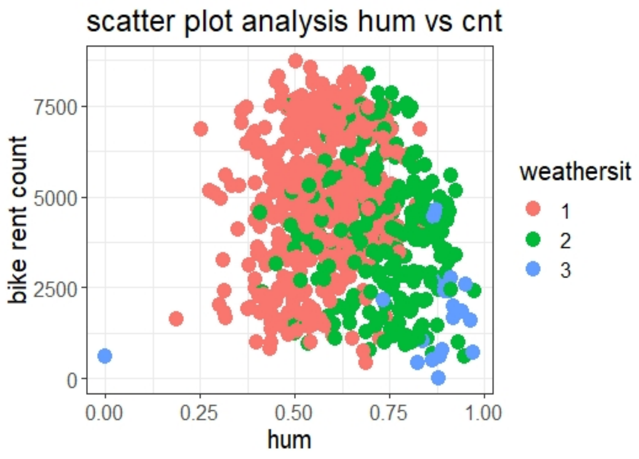
A



B



C



D

Fig 2.3 Scatter plots of some numeric variables vs cnt

From fig 2.3 A,B,C,D we can see the pattern data distribution for variable ‘temp’ and ‘atemp’ is linear as compared to variables ‘hum’ and ‘windspeed’ so ‘hum’ and ‘windspeed’ variables may contain the outliers. We have plotted all these plots w.r.t. some categorical variables that patterns also we can see in different plots. We can see that for the ‘weathersit’ category ‘1’ and ‘season’ variable category ‘3’ $ ‘4’ bike rent count is higher. Also count is low for ‘weathersit’ category ‘3’ and ‘season’ category ‘1’ as shown in fig.

2.2 Data Preprocessing:

2.2.1 Missing Value Analysis:

We need to do this analysis to check is there any missing values present in the data. We will do this on whole data and then find the variables who is having missing value percentage is less than 30 % if any variables is having more than 30% missing values then we will skip that variable directly. Now after getting the variables of missing values we will impute that values by using one of the method from *'Mean', 'Median' and 'KNN*, imputation method.

In our given data we have checked for missing values but our data is not having missing values so will proceed further to our next step.

2.2.2 Outlier Analysis:

This step is used for finding the skewed data entries present in the particular variables. We will find it and then replace that values with 'na' and then impute that 'na' values with one of the method from *'Mean', 'Median' and 'KNN, imputation method.* the outlier analysis can be done on numeric variables only. We will use very powerful tool to do outlier analysis that is Tukey's method. In this method the box plots are plotted for each variable. That may be by taking only values of particular variable or w.r.t. to target variable. Box plot contains lower fence and upper fence at the lower and top portion of box plot. Middle of plot is Median. The values crossing the lower or upper fence will be termed as outliers.

We have plotted the box plots for variables 'hum', 'windspeed', 'temp' and 'atemp' from numeric variables 'casual' and 'registered' variables are not taken for study. Because our target value is sum of casual+ registered variables so finding the outlier in that data is not significant. The box plots are shown below.

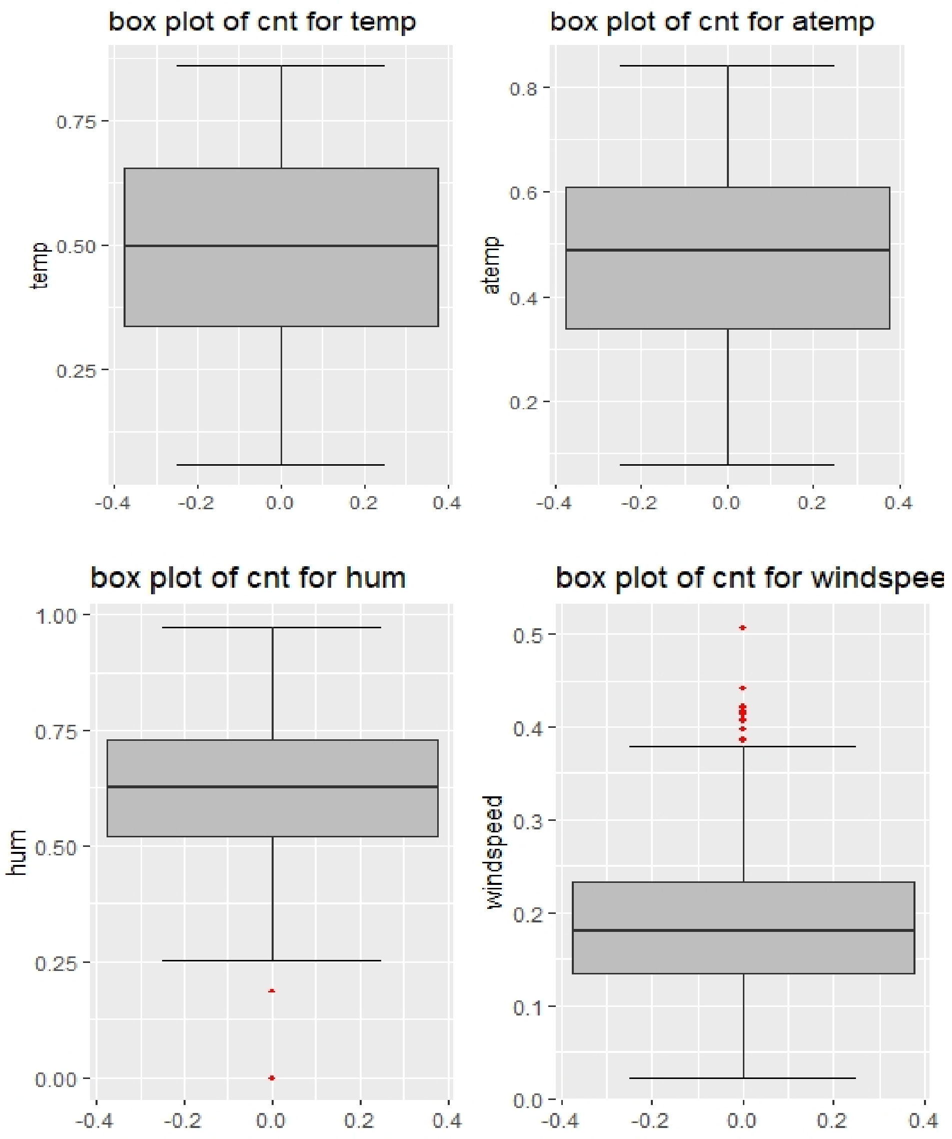


Fig 2.4 box plot for variables 'temp', 'atemp', 'hum' and 'windsped'

In fig 2.4 we can the box plot of all four variables but the box plot of 'hum' and 'windspeed' is showing some red dots below lower fence and above upper fence of 'hum' and 'windspeed' plot respectively. These dots are nothing but outliers. So we need to remove these outliers. I have imputed that outliers by first replacing them with 'na' and then imputed with the 'mean' method. After imputing that values, I again plotted the box plot and histogram of 'hum' and 'windspeed' variables which is shown below in fig 2.5. So we can clearly see form the figure that without outlier the data for both variables is shifted from the mean to towards left and right respectively. But after outlier analysis we can see the data is normally distributed.

Before Outlier Analysis After outlier analysis

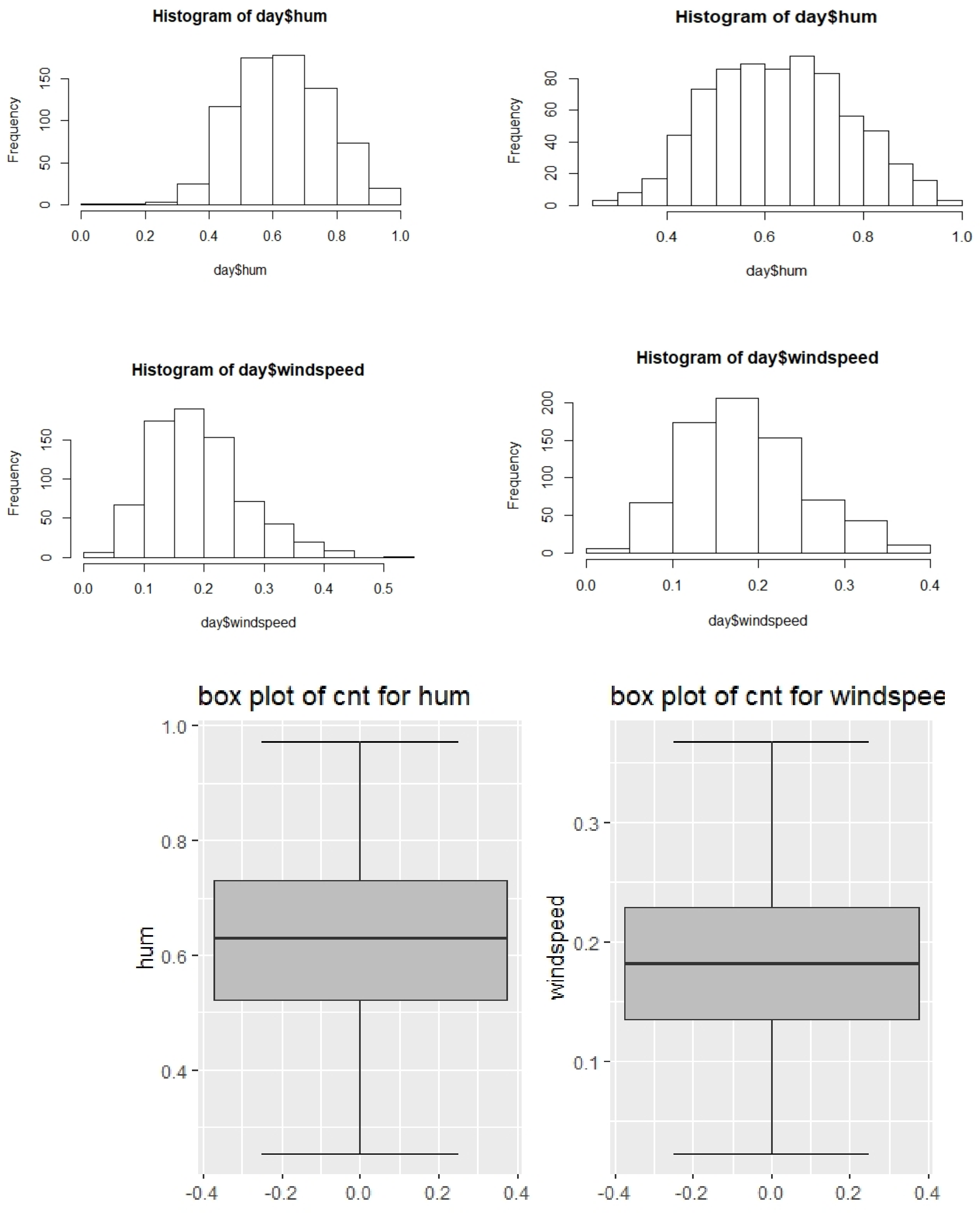
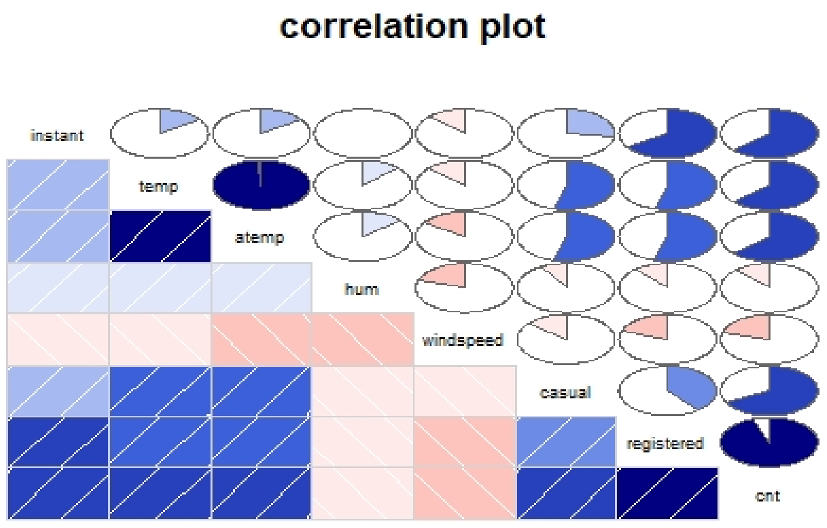


Fig 2.5 Effect of Outlier analysis on ‘hum’ and ‘windspeed’

2.2.3 Feature selection:

This is the most important step in data preprocessing. In this step will we check the dependencies between the variables. So for getting the good results there should be always very high dependency between the predictor and target variable and very low dependency between the one predictor variable to another predictor variable that is between the two independent variables. Because if two predictor variables is having high dependency it will just increase the data size and time to get the results. So there is no point in selecting both the variables for the further study so better is to drop one of the variable from them. So this will lead to good results and reduce the time required to calculate the result. This process is called as feature selection that means selecting only that variables which will be very strongly related to target variable and remove the predictor variables which are having high dependency between each other.

There are multiple tools and techniques present to do the feature selection. The main two are Wrapper and Filter methods. Filter method uses statistical method to check the dependency between the variables. For getting the correlation between the numeric variables (predictor and target both numeric) powerful tool used is correlation plot. One more tool used is multicollinearity analysis. To check the multicollinearity between the variables VIF is used which is Variance Inflation factor. Its value should be below 10 for lesser multicollinearity. From these two methods I have used correlation plot analysis to find the correlation of numeric predictor variables from our data to target variable. For the categorical variables we can either use Chi- square test or Analysis of variance test (ANOVA). So I have used ANOVA test to check dependency between our categorical predictor variable and numeric target variable.



,

Fig 2.6 Correlation plot for numeric variables

Fig 2.6 shows the correlation plot. It contains combination of pie chart and text panel below it. Both are explaining the same data we can refer any one of them to check the correlation. The dark blue colour indicates that there is very high positive correlation between two variables and dark red colour indicates that there is very high negative correlation between two variables. So by analyzing the plot we can see that 'temp' and 'atemp' are very highly and positively correlated to each other and also correlated to the 'casual' and 'registered' variables. Also we know that 'cnt'= 'casual' + 'registered'. And there is correlation between 'casual' and 'registered'. So we will remove 'atemp', 'casual' and 'registered' and also we can remove 'instant' and 'dteday' because these are just continuous index numbers so these are not containing any relevant information of target variable. After this we will keep the remaining numeric variables for further study.

Now we have some categorical variables in our data. So to check the dependency of these variables with target variable I have used ANOVA test because it is used when we have categorical Predictor variables and target variable is numeric. ANOVA calculates the mean of variables and compares it with another variable. So null hypothesis is means are equal so in our case it means both variables i.e. predictor and target variable are not dependant and alternate hypothesis as both variables are dependant. This can be tested by means of P-value which is the probability value. When the P<0.05 we reject the null hypothesis and vice versa.

I have performed ANOVA test on our data i.e. on predictor categorical variable w.r.t. to target variable. The results of ANOVA test are as follows. The Categorical variables we have are(*season, yr, mnth, holiday, weekday, workingday, weather sit*) and target variable(*cnt*)

```
[1] "season"
               Df    Sum Sq   Mean Sq F value Pr(>F)
factor_data[, i]   3 9.506e+08 316865289   128.8 <2e-16 ***
Residuals        727 1.789e+09   2460715
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "yr"
               Df    Sum Sq   Mean Sq F value Pr(>F)
factor_data[, i]   1 8.798e+08 879828893   344.9 <2e-16 ***
Residuals        729 1.860e+09   2551038
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "mnth"
               Df    Sum Sq  Mean Sq F value Pr(>F)
factor_data[, i]  11 1.070e+09 97290206    41.9 <2e-16 ***
Residuals        719 1.669e+09  2321757
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "holiday"
               Df    Sum Sq  Mean Sq F value Pr(>F)
factor_data[, i]   1 1.280e+07 12797494   3.421 0.0648 .
```

```
Residuals        729 2.727e+09  3740381
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "weekday"
               Df    Sum Sq Mean Sq F value Pr(>F)
factor_data[, i]   6 1.766e+07 2943170   0.783  0.583
Residuals        724 2.722e+09 3759498


[1] "workingday"
               Df    Sum Sq  Mean Sq F value Pr(>F)
factor_data[, i]   1 1.025e+07 10246038   2.737 0.0985 .
Residuals        729 2.729e+09  3743881
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "weathersit"
               Df    Sum Sq   Mean Sq F value Pr(>F)
factor_data[, i]   2 2.716e+08 135822286   40.07 <2e-16 ***
Residuals        728 2.468e+09   3389960
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

So from the above results we can see the P-value for variables holiday, weekday and workingday is less than 0.05. So for these variables we will accept null hypothesis i.e. these variables are not dependant on target variable so we can remove these variables from our study. P-values for other variable is less than 0.05 so we will take select variables for further study.

After performing the feature selection techniques we have selected the following variables for further study.

Variables selected for further study= (‘season’, ‘yr’, ‘mnth’, weathersit’, ‘temp’, ‘hum’, ‘windspeed’)

2.2.4 Feature Scaling:

Feature scaling is the method of scaling the data i.e. when we have the two numeric data variables and all values of one data are far away from the all values of other data then that data is not feasible for machine learning model. So there is a need to scale all the data variables in one common scale. For this we have two methods 1) Normalization 2) Standardization.

Normalization is used when the variable data in not normally distributed. It converts all the data in the range of “ 0 to 1”. After converting data to scale of 0 to 1 it is very easy to process that data in machine learning model and predict or classify our target variable more accurately.

Standardization is used when the data is normally distributed. It is also called as Z- score method this score ranges from “-1 to +1” so standardization converts the whole data in the range of “-1 to +1”, Z-score actually gives the information about how much that variable data is deviating from the mean of that same variable data. So positive Z-score means the value is above the mean and negative means value is below the mean.

Now our raw data is already normalized data. The numeric variables we have after feature selection are ‘temp’,’ hum’ and ‘windspeed’ are in the range of 0 to 1. So there is no need to go for feature scaling in our case. So we will proceed to further steps.

2.2.5 Sampling:

Sampling is nothing creating the subset of whole data with some conditions and methods. There are mainly three methods of sampling,

- **Simple Random Sampling:**

This is the most simple method of sampling. It can be used for both categorical and numeric variable. It is used mainly when the Target Variable is numeric. In this method of sampling the subset of data is created by use of rows. Just we need to pass the number of samples we want and randomly that number of samples are taken from original data.

- **Stratified Sampling:**

This method mostly used when the target variable is categorical. In this method we need to give the categorical variable as reference variable for creating samples. Strata’s are created which will take some given percentage of data form original data randomly. We need to give the numbers in term of percentage of data to be taken from each category of reference categorical variable

- **Systematic Sampling:**

This sampling method very complicated as compare simple random sampling method. In this method the samples are taken by the index numbers of data points in the original data. Index number is calculated by K value. K value is the ratio of total observations to the no of sample observations we require. So at start any observation from row is taken then the next observation is taken by adding the k value in the index of previous observation. So this is not the random sampling method.

For our study I have used Simple Random Sampling method because our target variable is numeric variable. By using this sampling method I have divided the data into train data and test data.

CHAPTER 3

MODEL BUILDING AND EVALUATION

3.1 Model Selection:

This is the actual step of creating the Machine Learning model. For building the model we want consider the type of target variable firstly.

The model may be of two types based on the target variable data type-

- **Classification model:**

Classification model means target variable is categorical and we build model for classifying the output in the categories of that variable.

- **Regression model:**

Regression model means target variable is numeric or continuous and we want to predict the continuous value of target variable for the sample input given in future cases by using the model

So as our target variable is numeric we will use regression models to predict the daily bike rental count. So I have build and tested accuracy of following three regression models

a) Decision tree model of regression
b) Linear regression model
c) Random Forest Regression model.

3.2 Decision Tree Regression Model:

Decision tree can be applied to both classification and regression. It creates the trees for the data. Which is having the parent node at top and each node under it denotes the class and leaf under it denotes the attributes. It is rule based method which creates some rules to predict the target variable. Each branch connects node with "and" operator & multiple branches are connected by "or" operator. For regression model it uses the central tendencies to derive the result of model.

The decision tree regression model can be build by using the statistical measures. I have first built the model on train data which we created by sampling by using the method of ANOVA and stored in fit object these are nothing but the rules of model which I have created. It contains different variables and pattern values at last stored below 'yval' column. So all this rules I have applied on test data to predict the target variable values. Now for model evaluation we will use some error metrics to study the performance of model. The rules from the model are as shown below,

```
split, n, deviance, yval
```

```
1) root 584 2379626000 4441.426
2) temp< 0.433333 240  602188700 2931.650
4) yr=0 144  158147900 2113.542
8) season=1,2 99   33150750 1624.273 *
9) season=4 45   49160230 3189.933
18) hum>=0.768333 7    9341423 1398.143 *
19) hum< 0.768333 38   13205340 3520.000 *
5) yr=1 96  203092400 4158.812
10) season=1 49   49585700 3144.082 *
11) season=2,4 47   50451180 5216.723 *
3) temp>=0.433333 344  848704200 5494.759
6) yr=0 165  117261400 4247.636
12) mnth=2,3,4,11,12 36   15474020 3394.111 *
13) mnth=5,6,7,8,9,10 129   68242240 4485.829
26) hum>=0.849375 12   14269030 3123.500 *
27) hum< 0.849375 117   29417670 4625.556 *
7) yr=1 179  238260500 6644.341
14) hum>=0.748542 37   64641700 5398.162 *
15) hum< 0.748542 142  101187400 6969.049
30) mnth=2,3,4,7,11,12 40   36274810 6193.200 *
31) mnth=5,6,8,9,10 102   31392630 7273.304 *
```

So I have applied the rules on test data and got some predicted values. The real and predicted values of 'cnt' variable are stored in datafarme and fig3.1 shows the plot of real and predicted values. We can clearly see in the graph that there is much variation in values of real and predicted.

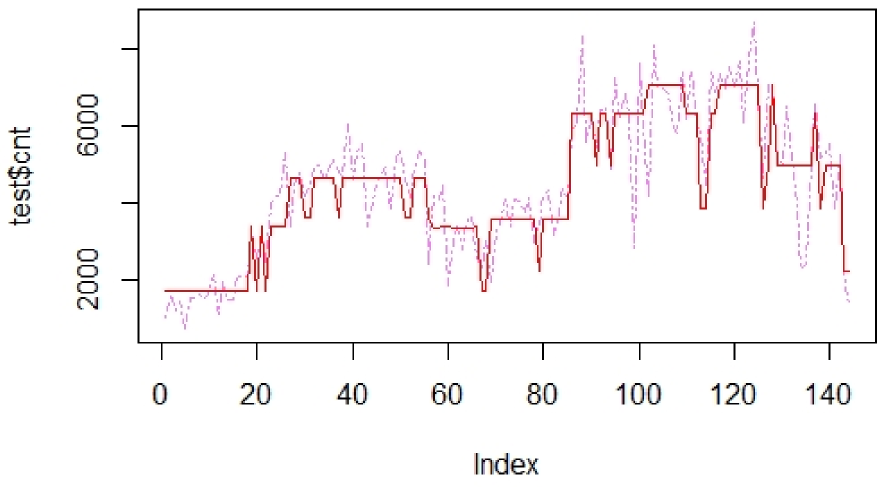


Fig 3.1 Graph or Real vs Predicted values of bike rental count.

(Real – Violet Colour, Predicted- Red colour)

The error metrics I have used to calculate the performance of the all model are,

1. **MAPE- Mean absolute Percentage Error.**

   This is the error calculated by following formula,

   MAPE= (y-yhat)/y))*100

   Where, y- real values, yhat- predicted values.

   The output of above error is in percentage. Which is the indication of how much difference is there between the real and predicted values.

   This should be lower for better performance.

2. **Accuracy:**

Accuracy is calculated by subtracting the MAPE value from 100. Which will give us the percentage of real and predicted values accurately predicted by our model.

3. **RMSE- Root Mean Square Error**

It is the standard deviation or prediction errors. It gives us the value that is measure of how far our data points from the regression line.

RMSE should be lower for better result.

4. **MAE- Mean Absolute Error**

I is a arithmetic average of absolute errors(prediction error)

MAE= $\frac{1}{n}\sum_{i=1}^{n} |y - x|$

Where, n= no of observations, y= predicted value, x= real value.

MAE should be lower for better result.

So after calculating the above 4 error metrics I have got the following values for the Decision tree model

❖ **PREDICTIVE PERFORMANCE OF MODEL USING ERROR METRICS:**

- MAPE= 18.59%
- Accuracy =81.41%
- RMSE=875.8407
- MAE=652.9456

The Decision tree model is having the accuracy of 81.41 which is quite good but we will still try to improve it so we will build linear regression model on the same data.

3.3 Linear Regression model:

This model is only applicable to target variable having numeric value. This is a one of the statistical model. This model calculates the coefficients of the independent variables to express the target variable. It means the coefficients or weights of independent variables will be applied on test data to predict target variable. Coefficients will express how target variable is behaving w.r.t. independent variable. i.e. when independent variable is increasing target variable is also increasing or decreasing or not and coefficients will be the amount by which it is deviating. This model will give the line equation of fit between the target and predictor variables.

Before building the model we must perform the multicollinearity analysis. It means to check whether the two predictor variables are correlated to each other or not like the same we checked in correlation analysis. Because it will affect the coefficients and also on the variance. This can be seen by the Variance Inflation factor. VIF compares the variance of one variable to variance of whole data and give some numeric value. When VIF<10 then there is no multicollinearity present in that variable.

So I have done that test on numeric variables and build the linear regression model on our same train data. Below is the summary of model.

Linear regression model-

```
Residuals:
    Min      1Q  Median      3Q     Max 
-3978.4  -344.3    76.3   434.5  3142.8 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1748.31     266.72   6.555 1.26e-10 ***
season2       891.72     215.75   4.133 4.12e-05 ***
season3      1018.01     256.72   3.965 8.27e-05 ***
season4      1657.92     209.39   7.918 1.29e-14 ***
yr1          2098.41      67.68  31.006  < 2e-16 ***
mnth2         156.94     158.40   0.991  0.32222    
mnth3         575.25     196.85   2.922  0.00361 ** 
mnth4         490.01     301.42   1.626  0.10458    
mnth5         827.57     322.07   2.570  0.01044 *  
mnth6         652.24     340.13   1.918  0.05567 .  
mnth7         -65.65     367.70  -0.179  0.85836    
mnth8         443.08     355.86   1.245  0.21361    
mnth9         812.31     306.61   2.649  0.00829 ** 
mnth10        453.13     281.50   1.610  0.10802    
mnth11        -71.79     265.01  -0.271  0.78656    
```

```
mnth12       -109.85     217.22  -0.506  0.61326
weathersit2  -488.12      93.21  -5.237 2.31e-07 ***
weathersit3 -1681.78     211.75  -7.942 1.08e-14 ***
temp        4160.98     453.85   9.168  < 2e-16 ***
hum        -1488.74     358.72  -4.150 3.84e-05 ***
windspeed   -2301.63     523.95  -4.393 1.34e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 776.8 on 563 degrees of freedom
Multiple R-squared:  0.8572,  Adjusted R-squared:  0.8522
F-statistic:   169 on 20 and 563 DF,  p-value: < 2.2e-16
```

As we can see the summary of model we have build. Residuals are nothing but the errors in our model minimum error is -3978.4 and maximum error is 3142.8. We have the estimates which are nothing but the coefficients for all predictor variables. When it is positive it will lead to increase the target variable and vice versa for e.g. temp value is +4160.98 here it means that when the temp will increase cnt will also increase by the 4160.98 times of temp variable. Then Std error also called as standard deviation error it measures the average amount that the coefficient of variable is deviating average of real value. Then the 't value' it gives the information about how many standard deviation of coefficients are away from zero. So that should be away from zero to have good correlation of target and predictor variable.

Now the P value it gives us the information about weather the predictor variable is significant with the target or not. The * indicates the significance 3 * means high significance. When P<0.05 that variable is most significant. Adjusted R-squared it provides the information about how much amount of our target variable variance is able to explain by predictor variables. It should be above 80%. For our model it is 85.22% which is good. F-statistic it means weather the predictor and target variable are dependent on each other or not. Its value should be more than 1. Our model is having 169 which is good. Again the P value for whole model is <0.05 it means our model is good for predicting the target variable.

I have stored the real and predicted values in data frame and fig 3.2 shows the graph of real vs predicted values of bike rental count by using Linear Regression model. We can see there is still variation in real vs predicted values is present. But which is slightly improved as compare with decision tree model.

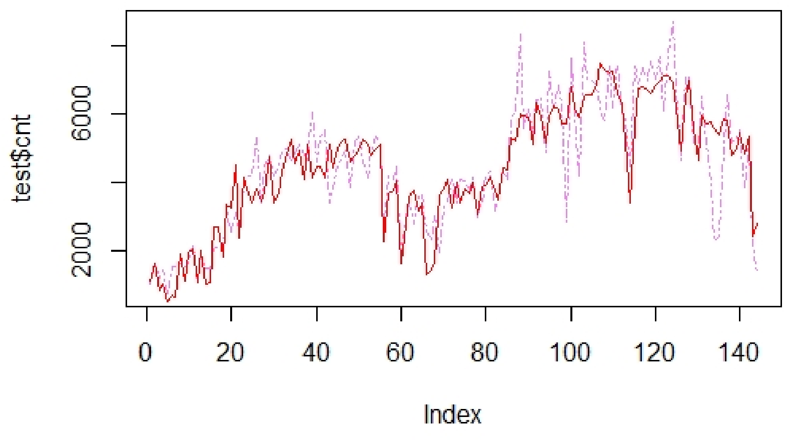


Fig 3.2 Graph or Real vs Predicted values of bike rental count.

(Real – Violet Colour, Predicted- Red colour)

❖ **PREDICTIVE PERFORMANCE OF MODEL USING ERROR METRICS**:

- MAPE(error rate)= 15.43%
- Accuracy = 84.57%
- adjusted R-squared= 85.22% which is greater than 80%
- RMSE=756.5901
- MAE= 569.2133

3.4 Random Forest Model:

This model is improved version of Decision tree model. It can be applied to both classification and regression method. It ensembles many decision trees to get the good result. It works on the principle of bagging and random selection of features. Bagging means error of one tree is feed to another tree to improve the accuracy and random feature selection means for every tree different features and data are used. It also provides the important variables from the predictor variables. It gives the output in terms of mean. The optimum number of trees should be in the range of 64-128. We will finalize the number of tree until there is no decrease in the error.

So I have built the Random Forest Regression model on our train data. Below are the details of model.

RF model:

```
Type of random forest: regression
Number of trees: 120
No. of variables tried at each split: 2
Mean of squared residuals: 356564.2
% Var explained: 91.25
```

In the above model we can see that I have used 120 tress to build the model. At each split 2 variables are taken and the variance explanation of target variables by predictor variables is 91.25 % which is very good. Next step is to convert the model to tree based format and extract the rules from the model as the model is providing some rules as output. The model is created 4438 rules. From that I have extracted top 2 rules as shown below,

```
[1] "X[,2] %in% c('0') & X[,3] %in% c('1','2','3','12') & X[,4] %in% c('2','3') & X[,6]<=0.712491 & X[,7]<=0.2349695 &
X[,7]<=0.0924354"
[2] "X[,2] %in% c('0') & X[,3] %in% c('1','2','3','12') & X[,4] %in% c('2','3') & X[,6]>0.712491 & X[,7]<=0.2349695 &
X[,7]<=0.0924354"
```

So rule 1 is when the 2nd variable in our train data is having value ’0’ and 3rd variables is having values '1','2','3','12' and 4th variable is having values ‘2’,’3’ and 6th variable is greater than 0.7124 and 7th variable is less than equal to 0.0234 and 0.0924. Then we will get highest output that is more bikes rented in our case. Then I have made these rules readable by giving the names to the variables instead of their index value. In the next step I have created the rule metrics in that we can see all that in one table form with the predicted output value in the last column.

Now next step is to apply the model on test data to predict the values of target variable in test data. So I have applied the model on our test data and stored the real predicted values in on data frame see below the 10 observations form that dataframe and fig 3.3 shows the graph of real vs predicted values of bike rental count by using Random Forest model.

Table 3.1 Real and Predicted values of 10 variables from test data.

Observation number	Real values	Predicted values
3	1349	1512.624
6	1606	1521.820
8	959	1466.579
10	1321	1493.130
12	1162	1480.760
15	1248	1615.132
16	1204	1566.913
17	1000	1526.417
19	1650	1521.773
20	1927	1606.902

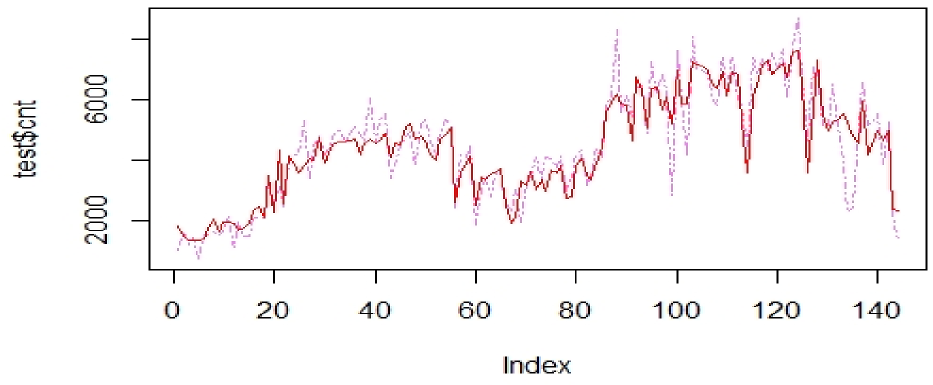


Fig 3.3 Fig 3.2 Graph or Real vs Predicted values of bike rental count.

(Real – Violet Colour, Predicted- Red colour)

❖ **PREDICTIVE PERFORMANCE OF MODEL USING ERROR METRICS:**

- MAPE(error rate)= 13.24%
- Accuracy = 86.76%
- RMSE=650.57
- MAE= 484.68

As we know that random forest model can give important variables in our data. So I have plotted the graph of important variables Vs percentage of MSE. Fig 3.4 shows the graph of that. From the graph we can say that ‘windspeed’, ‘mnth’, ‘hum’, weathersit’, ‘season’, ‘temp’ , ’yr’ variables are having low values of percentage error with ascending order. So these are the important variables form our data w.r.t. target variable.

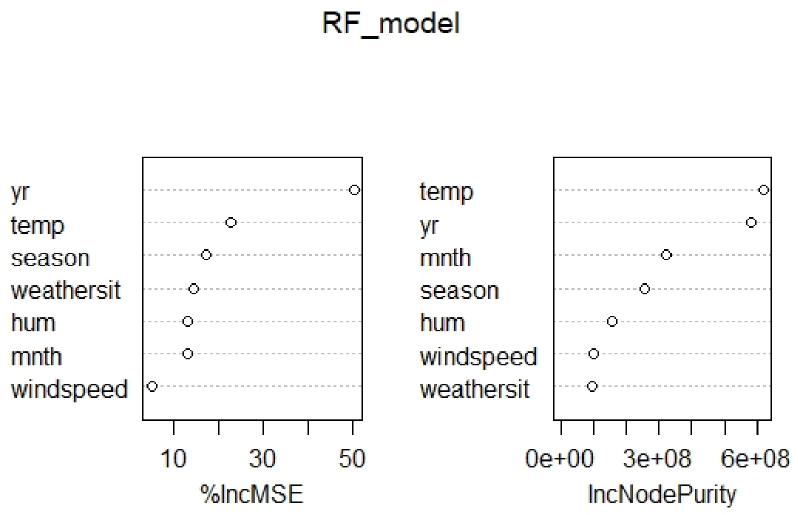


Fig 3.4 Important variables VS Percentage of MSE

The error we get in the model is based on the how many no of trees we are using. Generally when we have more no of trees we will get less error. Fig 3.5 shows the graph of no of trees used in RF model vs error. At last I have plotted scatter plot Fig 3.6 of real vs predicted values of daily bike rental count for test data. We can see there is linear relationship between both is present.

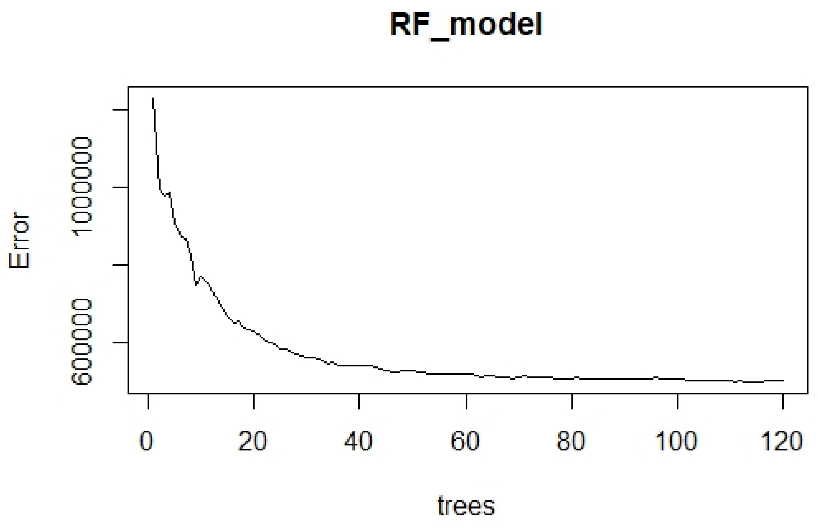


Fig 3.5 Graph of no of trees used in RF model vs error.

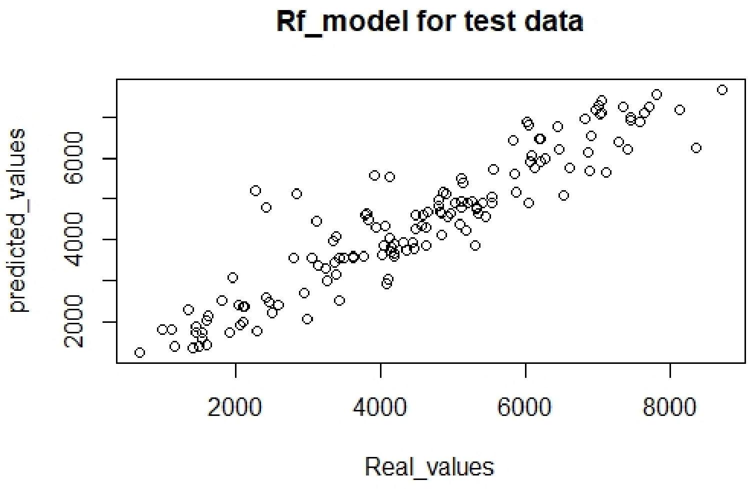


Fig 3.6 Scatter plot of Real vs Predicted values of Bike rental count of test data.

So we have built the three models for our target variable now next step is model evaluation and selection of final model to predict the daily bike rental count.

All the three models that we have studied until now are built on **"R"** software and in next step we will do evaluation and final selection of model by using error metrics that we have already calculated also in the '**Appendix A**' I have written the complete **'R'** code for this project

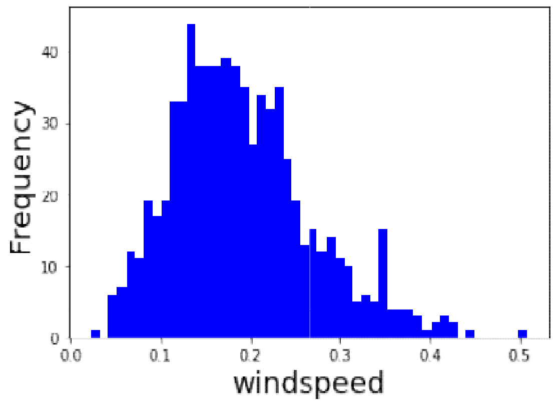
3.5 SAME PROJECT BY USING PYTHON:

I have also done the Python Programming of above steps that we have performed for building the Machine Learning Model for predicting the Daily Bike Rental Count. The procedure and steps are same for building the Machine Learning Model in **'Python'** that we done in **'R'** So in '**Python**' also I have build the same three models that we built above in **'R'**

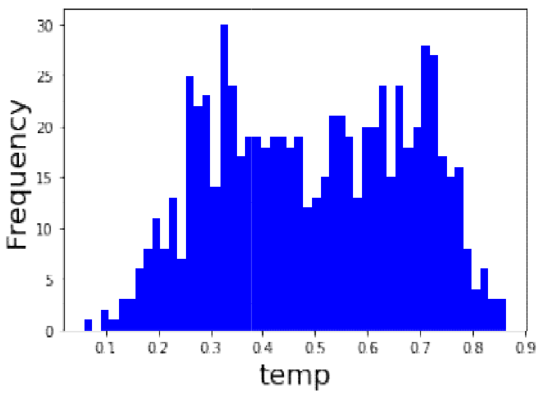
Now in next step that is model evaluation we will do the evaluation of all three models that was built in 'R' and 'Python'. The error metrics that I have calculated related to same models in Python are "MAPE", "MSE" "MAE" and "RMSE".

Visualizations made in python are shown in next chapter which are very similar to that we made in '**R**' also in **'Appendix B'** I have written complete '**Python**' code.

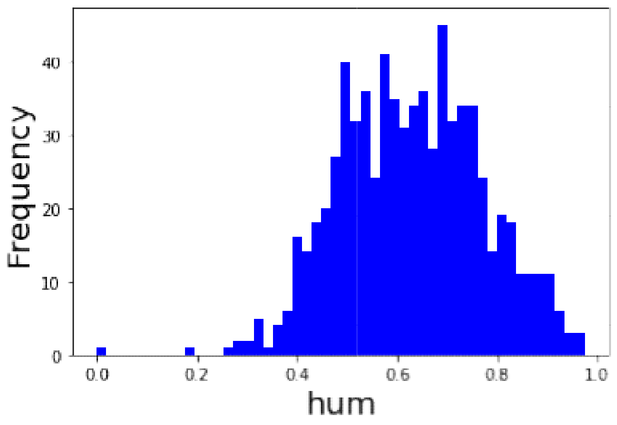
CHAPTER 4 -VISUALIZATIONS FROM PYTHON



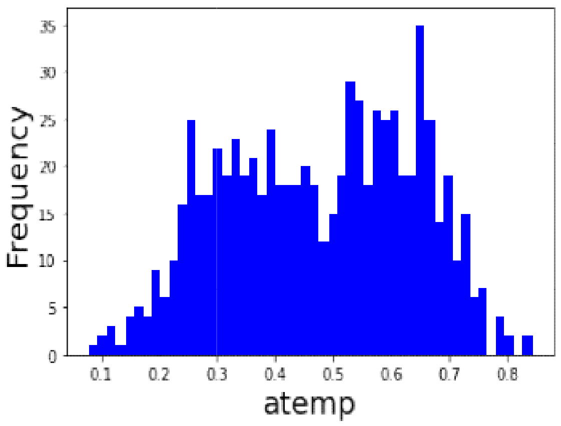
A: Histogram of 'windspeed'



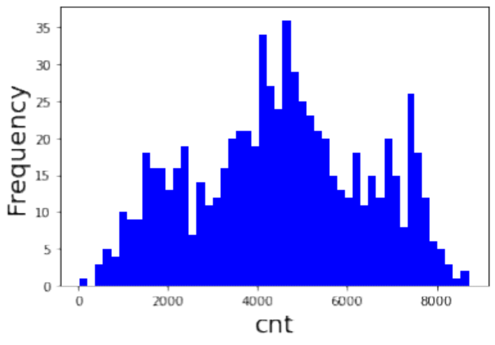
B: Histogram of 'temp'



C: Histogram of 'hum'

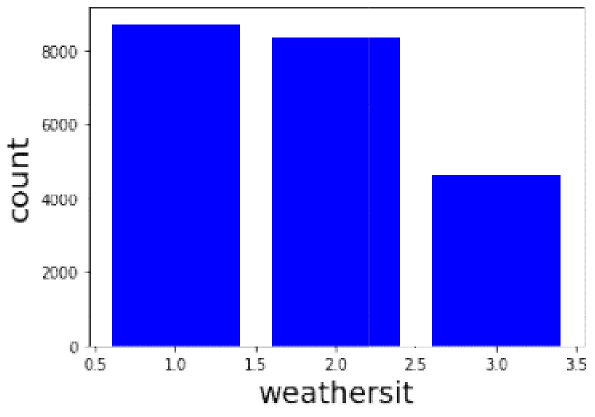


D: Histogram of 'atemp'

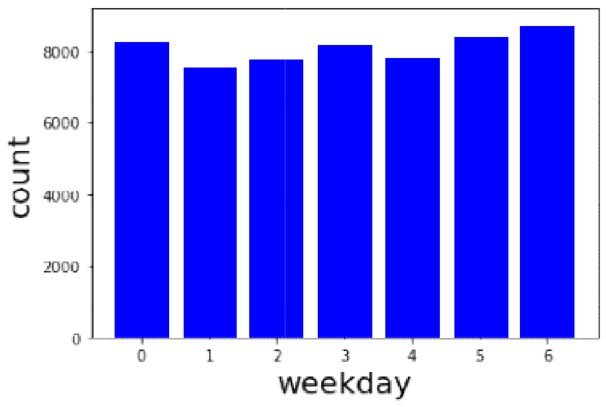


E: Histogram of 'cnt'

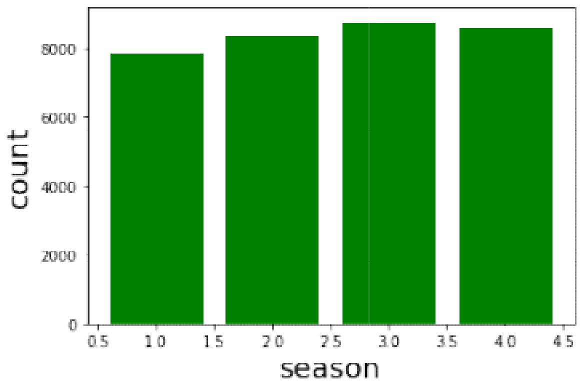
Fig 4.1 Histograms of all Numeric variables.



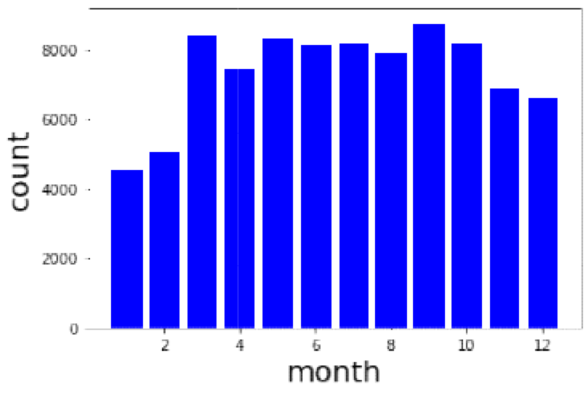
[A]



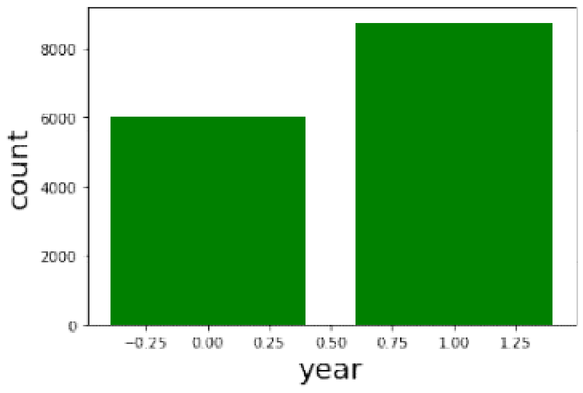
[B]



[C]



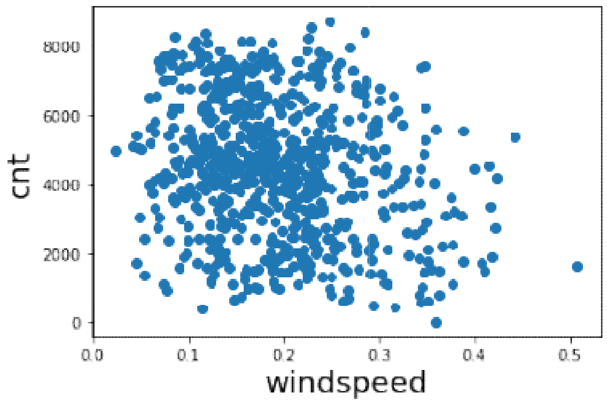
[D]



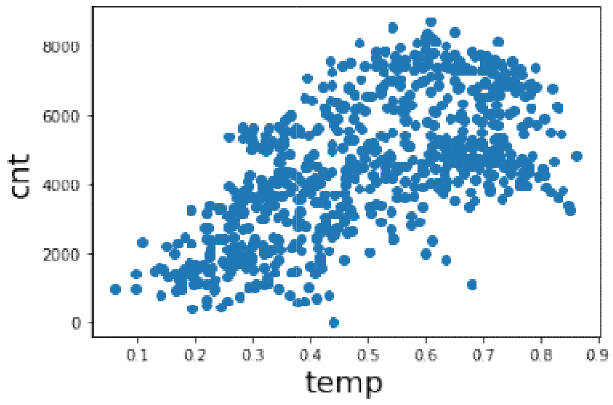
[E]

Fig 4.2 Bar Graphs of ‘cnt’ vs categorical variables

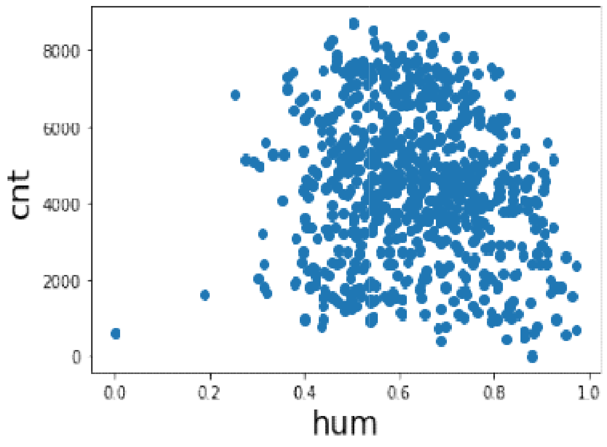
4.3 Scatter plots:



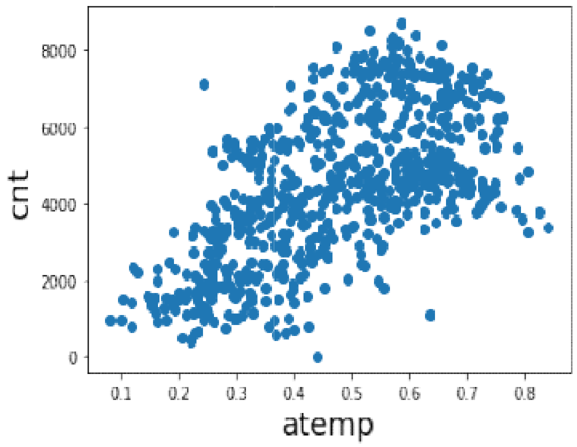
A: ‘cnt’ vs ‘windspeed’



B: ‘cnt’ vs ‘temp’



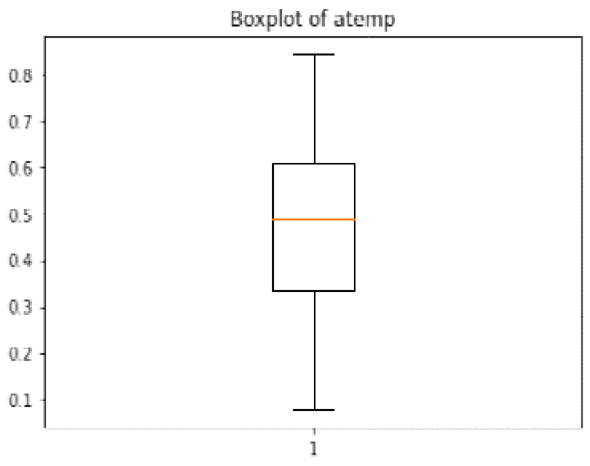
C: ‘cnt’ vs ‘hum’



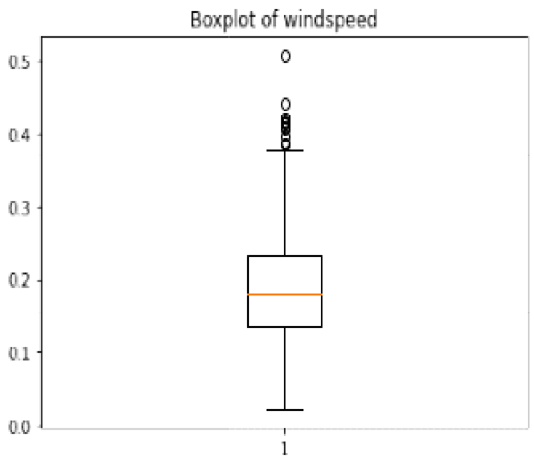
D: ‘cnt’ vs ‘atemp’

Fig 4.3 Scatter Plots of Categorical Variables vs Target Variable

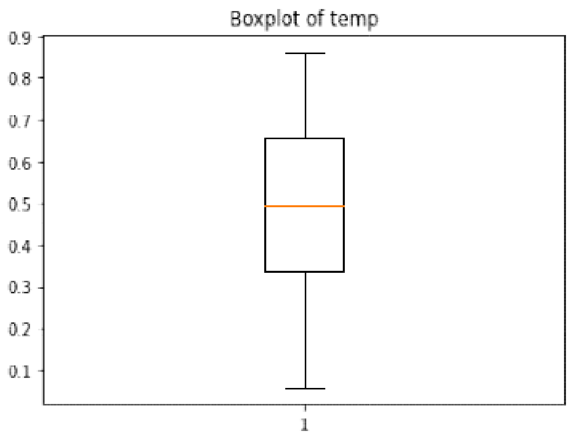
4.4 Box Plots:



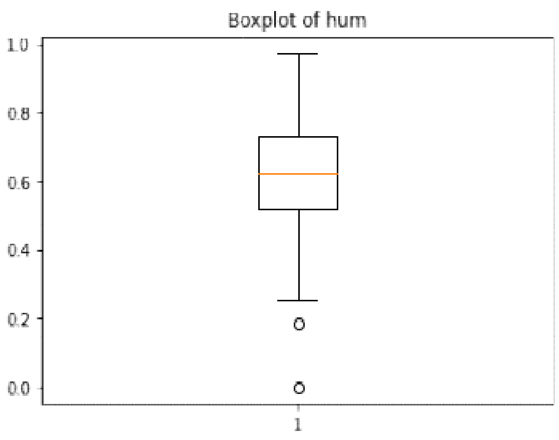
A: Boxplot of 'atemp'



B: Boxplot of 'windspeed'



C: Boxplot of 'temp'



D: Boxplot of 'hum'

Fig 4.4 Boxplots of Numeric Variables

As we can see in fig 4.4 B and D there are outliers present in variables 'windspeed' and 'hum'. We have got the outliers for these variables in our 'R' code also so same result we can get in Python.

4.5 Correlation plot: The plot between all numeric variables to check the dependency between them. So from the fig 4.5 below we can see that 'temp' and 'atemp' are positively correlated with each other.

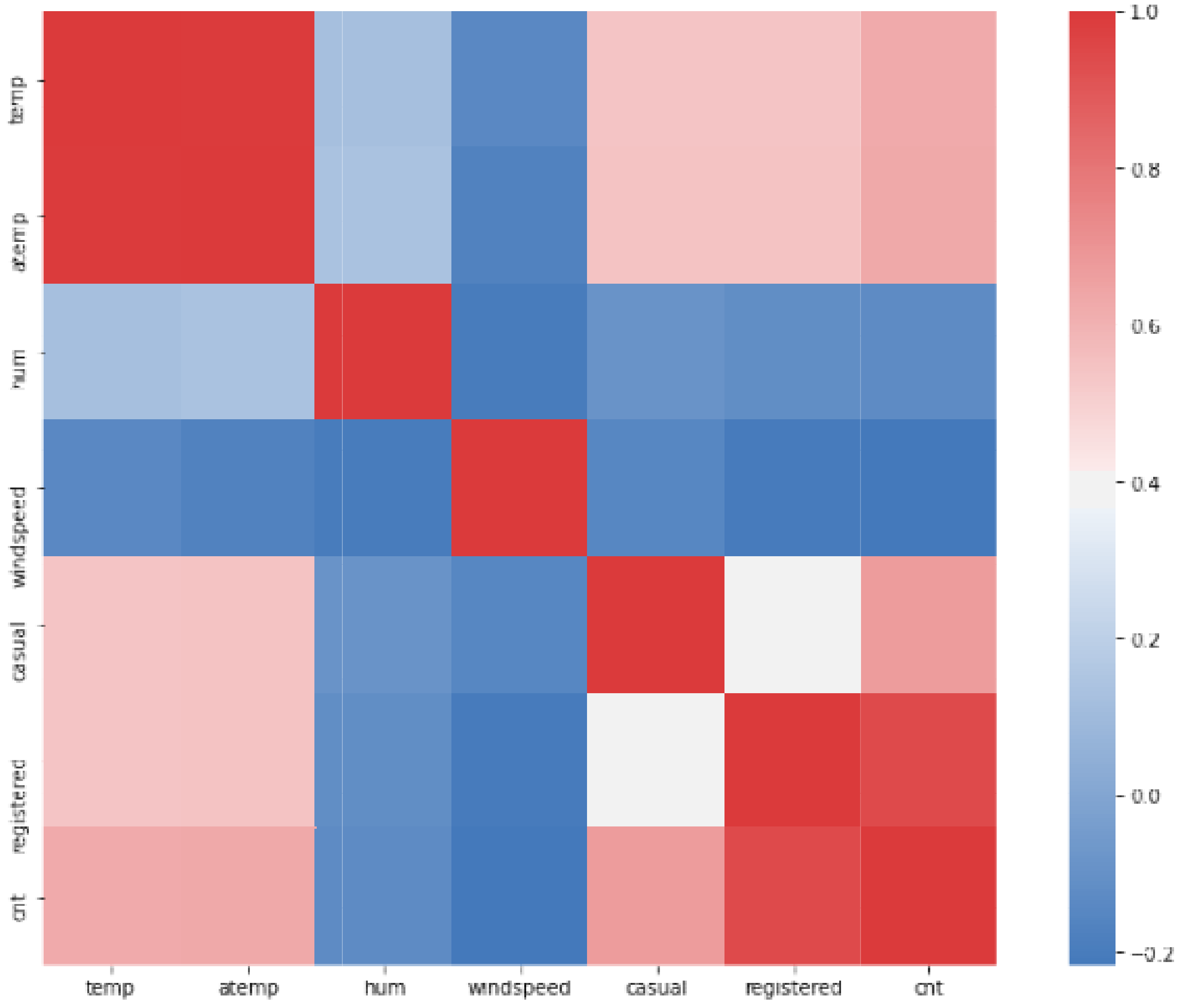
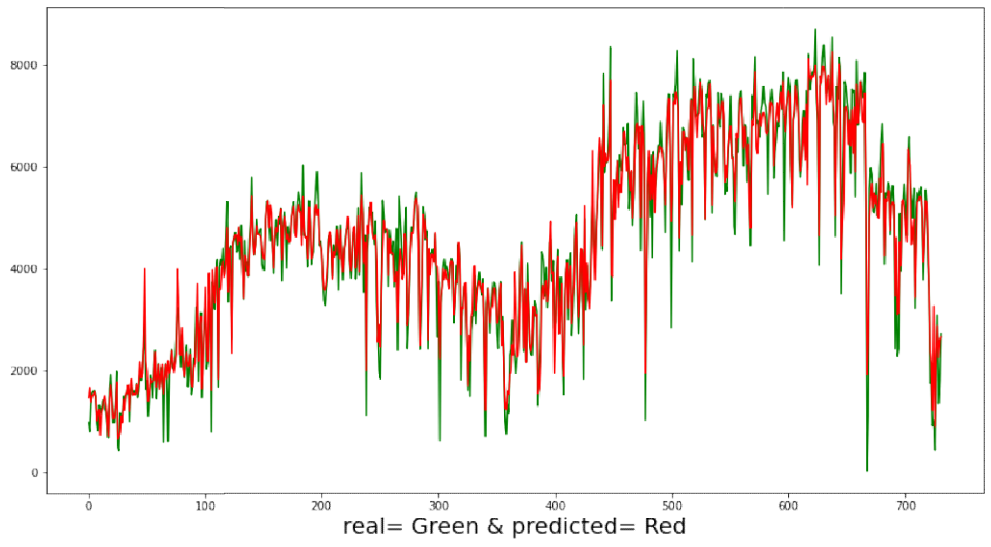
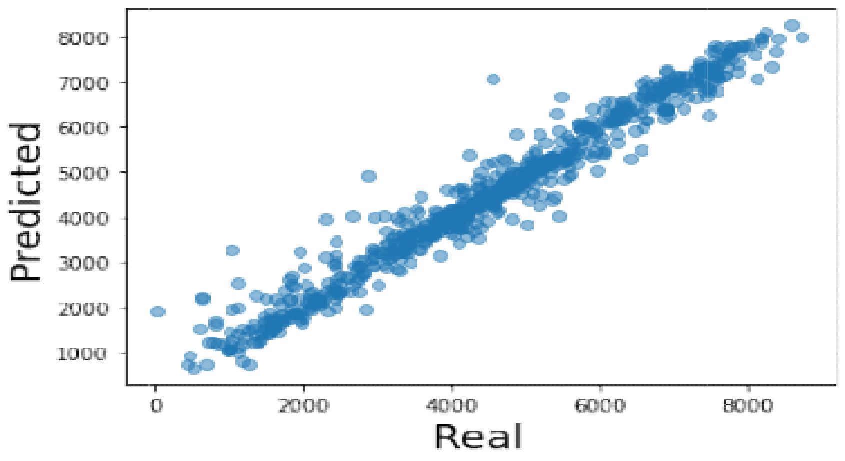


Fig 4.5 Correlation Plot between Numeric Variables

4.6 Results of Random Forest Selected Model: we have built 3 regression models on our data and from that we have selected Random Forest model due to its higher accuracy. So fig 4.6 A below showing the graph of real and predicted values we can see the significant result from it and fig 4.6 B showing the scatter plot of real vs predicted values.



A: Real vs Predicted values



B: Scatter plot of Real vs Predicted values

Fig 4.6 Results of Random Forest Selected model

CHAPTER 5

CONCLUSION

5.1Model Evaluation:

For the evaluation of models we have built I have selected three error metrics in ‘R’ those are,

- MAPE
- RMSE
- MAE

I have already calculated these three error metrics for all three models we have built. By using MAPE I have calculated Accuracy also for all three models. We know that the values of RMSE and MAE should be as low as possible to get the best results. Now we will compare these error metrics of all models and select the best model with higher accuracy. So all these error metrics are as shown in table below.

Table 5.1 Error metrics of all three models by ‘R’ software

Error Metrics	**Decision Tree model**	**Linear Regression Model**	**Random Forest Model**
MAPE (%)	18.59	15.43	13.24
RMSE	875.8407	756.59	650.57
MAE	652.9456	569.21	484.68
ACCURACY (%)	81.41	84.57	86.76

Now in python along with above three error metrics I have also calculated “Mean Square Error”, Table 4.2 below shows the error metrics or performance metrics of same Regression models that I have built in Python.

Table 5.2 Error metrics of all three models by 'Python' software

Error Metrics	**Decision Tree model**	**Linear Regression Model**	**Random Forest Model**
MAPE (%)	26.96	22.76	20.21
RMSE	2371.06	955.44	692.77
MAE	1958.58	709.16	535.24
MSE	5621993.20	912880.30	479937.73
ACCURACY (%)	73.04	77.24	79.79

5.2 Selecting the Final Model:

Model selection is done by comparing the error metrics of all three models. So as we can see in table 4.1 accuracy of all three models. Random Forest Model is having highest accuracy. Also RMSE and MSE values of Random Forest Model are low as compare to other two models.

The same result is seen in Python also accuracy of Random Forest model is high as compared with other two models. Also the values of all other error metrics are also less for Random Forest Model.

So we will finalize the RANDOM FOREST Model and select it for prediction of our target variable i.e. Daily Bike Rental Count. The accuracy of Random Forest Model that we got in "R" is higher than that we got in "Python". So it can be said that random Forest Model that we built in "R" Software is most suitable for our Project.

5.3 Predicting the output by Sample Input:

Now we will predict the value of test data by using our finalized Random Forest Model by giving the sample input of 2nd row and 87th row

Output1- Target variable i.e. Bike rental count for 2nd observation from test data.
Predicted value-1521.82
Real value – 1606

Output2- Target variable i.e. Bike rental count for 87th observation from test data.
Predicted value-4867.07
Real value - 5084

5.4 Saving the output with original data:

Now last step in our project is saving of Random Forest Model and its output as a csv file. For this I have created a new variable named as "predicted" in the original data set and saved all the predicted values of target variable by our model in that column. So the output files from our project are **"R_output.csv" , "day", "MAPE"** , **"RF_model"** here R_output.csv is data with predicted values column, "day" is the original data, "MAPE" is the mean absolute percentage error of Random Forest Model and "RF_model" is the Random Forest Model.

The output project in Python is stored as a csv file named as "**python_out_main.csv**" which contains the column "predicted" with predicted values of target variable and all other predictor variables.

APPENDIX A – COMPLETE R CODE FILE

```
rm(list=ls(all=T))
setwd("F:/data Scientist/project 1 bike renting/r code")
# load libraries
x= c("ggplot2","corrgram","DataCombine","scales","psych","gplots","Metrics"
,"inTrees","DMwR","car", "caret","rlang","usdm", "Information", "randomForest",
"unbalanced", "C50", "dummies", "e1071", "MASS", "ROSE", "rpart", "gbm")
install.packages(x)
lapply(x, require, character.only=TRUE)
rm(x)
day = read.csv("day.csv", header=T )
#load data
strings = c(" ", "", "NA")
day_copy=day
str(day)

#### EXPLORATORY DATA ANALYSIS

# convert variables to proper data type
day$season=as.factor(day$season)
day$yr=as.factor(day$yr)
day$mnth=as.factor(day$mnth)
day$holiday=as.factor(day$holiday)
day$weekday=as.factor(day$weekday)
day$workingday=as.factor(day$workingday)
day$weathersit=as.factor(day$weathersit)
day$dteday=as.Date(day$dteday)

###### Visualizations related to given data

# histograms of numerical variables
hist(day$temp)
hist(day$atemp)
hist(day$hum)
hist(day$windspeed)
hist(day$cnt)

# let us plot bar graph of each catagorical variable w.r.t target variable
# bike rent count w.r.t. month
ggplot(day, aes_string(x=day$mnth, y=day$cnt))+
  geom_bar(stat='identity', fill="DarkSlateBlue")+ theme_bw() +
  xlab("month") +ylab("count")+
  ggtitle("Bike rent count w.r.t. month")+
  theme(text=element_text(size=15 ))

#bike rent count w.r.t. weekday
ggplot(day, aes_string(x=day$weekday, y=day$cnt))+
  geom_bar(stat='identity', fill="DarkSlateBlue")+ theme_bw() +
  xlab("weekday") +ylab("count")+
  ggtitle("bike rent count w.r.t. weekday")+
  theme(text=element_text(size=15 ))

#bike rent count w.r.t. season
ggplot(day, aes_string(x=day$season, y=day$cnt))+
  geom_bar(stat='identity', fill="DarkSlateBlue")+ theme_bw() +
  xlab("Season") +ylab("count")+
  ggtitle("bike rent count w.r.t. season")+
  theme(text=element_text(size=15 ))

#bike rent count w.r.t. year
ggplot(day, aes_string(x=day$yr, y=day$cnt))+
  geom_bar(stat='identity', fill="DarkSlateBlue")+ theme_bw() +
  xlab("year") +ylab("count")+
  ggtitle("bike rent count w.r.t. year")+
  theme(text=element_text(size=15 ))

# scatter plot analysis

# scatter plot of hum vs cnt and weathersit
ggplot(day, aes_string(x=day$hum, y=day$cnt))+geom_point(aes_string(colour=
day$weathersit), size=4)+
  theme_bw()+ xlab("hum")+ylab("bike rent count")+ggtitle("scatter plot analysis
hum vs cnt")+ theme(text=element_text(size=15))+
  scale_color_discrete(name="weathersit")

# scatter plot windspeed vs cnt and weathersit,season
ggplot(day, aes_string(x=day$windspeed, y=day$cnt))+geom_point(aes_string(colour=
day$weathersit, shape=day$season), size=4)+
  theme_bw()+ xlab("windspeed")+ylab("Bike rent count")+ggtitle("scatter plot
windspeed vs cnt")+ theme(text=element_text(size=15))+
  scale_color_discrete(name="weathersit")+ scale_shape_discrete(name= "season")

# scatter plot of temp vs cnt and weathersit
ggplot(day, aes_string(x=day$temp, y=day$cnt))+geom_point(aes_string(colour=
day$weathersit), size=4)+
  theme_bw()+ xlab("temp")+ylab("bike rent count")+ggtitle("scatter plot analysis
temp vs cnt")+ theme(text=element_text(size=15))+
  scale_color_discrete(name="weathersit")

# scatter plot of atemp vs cnt and season
ggplot(day, aes_string(x=day$atemp, y=day$cnt))+geom_point(aes_string(colour=
day$season), size=4)+
  theme_bw()+ xlab("atemp")+ylab("bike rent count")+ggtitle("scatter plot analysis
a atemp vs cnt")+ theme(text=element_text(size=15))+
  scale_color_discrete(name="season")

###### DATA PREPROCESSING


#finding missing values in whole data
sum(is.na(day))

# No Missing values in data

####### outlier analysis

cnames_for_outlier= colnames(day[,c('temp', 'atemp', 'hum', 'windspeed')])


# find the outliers using box plot
for (i in 1: length(cnames_for_outlier)){
  assign(paste0("ab", i), ggplot(aes_string(y=(cnames_for_outlier[i])),
  data=subset(day))+stat_boxplot(geom = 'errorbar', width=0.5)+
    geom_boxplot(outlier.colour = "red",
  fill= "grey", outlier.shape = 18,
  outlier.size = 1, notch = FALSE)+
    theme(legend.position = "bottom")+
    labs(y=cnames_for_outlier[i])+ggtitle(paste("box plot of cnt for",
cnames_for_outlier[i])))
}


#plotting plots together
gridExtra::grid.arrange(ab1,ab2, ncol=2)
gridExtra::grid.arrange(ab3,ab4, ncol=2)



#replace outliers with na
for(i in cnames_for_outlier){
  val=day[,i][day[,i] %in% boxplot.stats(day[,i])$out]
  day[,i][day[,i] %in% val] = NA
}

sum(is.na(day))
#impute outliers
day$windspeed[is.na(day$windspeed)]= mean(day$windspeed,na.rm= T)
day$hum[is.na(day$hum)]= mean(day$hum, na.rm=T)

sum(is.na(day))

for (i in 1: length(cnames_for_outlier)){
  assign(paste0("cd", i), ggplot(aes_string(y=(cnames_for_outlier[i])),
                                  data=subset(day))+stat_boxplot(geom = 'errorbar',
width=0.5)+
           geom_boxplot(outlier.colour = "red",
                        fill= "grey", outlier.shape = 18,
                        outlier.size = 1, notch = FALSE)+
           theme(legend.position = "bottom")+
           labs(y=cnames_for_outlier[i])+ggtitle(paste("box plot of cnt for",
cnames_for_outlier[i])))
}

gridExtra::grid.arrange(cd1,cd2, ncol=2)
gridExtra::grid.arrange(cd3,cd4, ncol=2)

# boxplot after outlier analysis
hist(day$hum)
hist(day$windspeed)

#select numeric data for correlation plot
numeric_index=sapply(day[,1:16], is.numeric)
numeric_data= day[,numeric_index]

# Correlation plot to check correlation of independant numeric variables
corrgram(day[,numeric_index],order=F, upper.panel=panel.pie,
         text.panel=panel.txt, main="correlation plot")
```

```
# anova test to check dependancy of catagorical variables with target variable

factor_index=sapply(day, is.factor)
factor_data=day[,factor_index]

for(i in 1:7){
  print(names(factor_data)[i])
  cnt= day[,16]
  anova=aov(cnt~factor_data[,i], data=factor_data)
         print(summary(anova))
}

# Remove unwanted columns from data

day_deleted= subset(day, select=-c(instant, dteday, casual, registered, atemp,
holiday, weekday, workingday))
day_deleted

#clean the environment
library(DataCombine)
rmExcept("day","day_deleted" )

########  lets Build the Machine Learning models on the data
#data is having numeric variable as a target variable
# so we will build following Regression models
# Decision Tree, Linear Regression, Random Forest model and choose one with higher
accuracy


# divide data into train and test using Simple Random sampling method

train_index=sample(1:nrow(day_deleted), 0.8*nrow(day_deleted),replace=FALSE , prob
= NULL)
train=day_deleted[train_index,]
test=day_deleted[-train_index,]
######### DECISION TREE MODEL OF REGRESSION

#Decision tree for regression
fit=rpart(cnt~., data= train, method="anova")

# predict for new test data
predictions_dt= predict(fit, test[,-8])

# compare real and predicted values of target variable

comparision_dt=data.frame("Real"=test[,8], "Predicted"= predictions_dt)

# function to calculate MAPE
mape= function(y, yhat){
  mean(abs((y-yhat)/y))*100
}
mape(test[,8], predictions_dt)

# To find rmse and mae
rmse(test[,8],predictions_dt)
mae(test[,8],predictions_dt)
```

```
# compare real and predicted values of target variable

comparison_dt=data.frame("Real"=test[,8], "Predicted"= predictions_dt)

# plotting the graph for real and predicted values of target variable
plot(test$cnt, type="l", lty=4, col="violet")
lines(predictions_dt, col="red")

##Predictive performance of model using error metrics

# mape(error rate)= 18.59%
# Accuracy =81.41%
# rmse=875.8407
# mae=652.9456



###### LINEAR REGRESSION MODEL

#check multicolinearity
vif(day_deleted[,5:7])

# run regression model
lr_model=lm(cnt~., data=train)

#summary of regression model
summary(lr_model)

# prediction on test data
predictions_lr=predict(lr_model, test[,1:7])

# calculate mape,rmse,mae

mape= function(y, yhat){
  mean(abs((y-yhat)/y))*100
}
# calculate error metrics to evaluate the model

mape(test[,8],predictions_lr)
rmse(test[,8],predictions_lr)
mae(test[,8],predictions_lr)

# plotting the graph for real and predicted values of target variable

plot(test$cnt, type="l", lty=4, col="violet")
lines(predictions_lr, col="red")

#example of output with sample input of 2nd row

predict(lr_model, test[2,])

# compare real and predicted values of target variable

comparison_lr=data.frame("Real"=test[,8], "Predicted"= predictions_lr)


# #Predictive performance of model using error metrics
# mape(error rate)= 15.43%
# Accuracy of Linear regression model= 84.57%
# adjusted R-squared= 85.22% which is greater than 80%
# rmse=756.5901
# mae= 569.2133

##### RANDOM FOREST MODEL OF REGRESSION

# creating the model
RF_model= randomForest(cnt~., train, importance=TRUE, ntree= 120)

# convert rf object to trees
treeList= RF2List(RF_model)

# extract the rules from the model
rules=extractRules(treeList, train[,-8])
rules[1:2,]

# make rule readable

readablerules=presentRules(rules,colnames(train))

# get rule metrics

ruleMetric=getRuleMetric(rules, train[,-8], train$cnt)

# predict the test data using RF model
predictions_rf=predict(RF_model,test[,-8])

# Calculate error metrics to evaluate the performance of model

mape(test[,8],predictions_rf)
rmse(test[,8],predictions_rf)
mae(test[,8],predictions_rf)


# plotting the graph for real and predicted values of target variable

plot(test$cnt, type="l", lty=4, col="violet")
lines(predictions_rf, col="red")

# compare real and predicted values of target variable

comparison_rf=data.frame("Real"=test[,8], "Predicted"= predictions_rf)
#example of output with sample input of 2nd row

test[2,]
predict(RF_model, test[2,])

test[87,]
predict(RF_model, test[87,])

# #Predictive performance of model using error metrics

# mape(error rate)= 13.24%
# Accuracy of Random Forest Model= 86.76%
# rmse=650.57
# mae= 484.68

#################################################################################
##
# So after comparing the performances of all three models
# we can see that the Random Forest model is having high accuracy
# so we will select Random Forest Model to predict the bike rental count.

# selecting the same model as output model of this project
RF_model= randomForest(cnt~., train, importance=TRUE, ntree= 120)

# predict test data
predictions_rf=predict(RF_model, test[,-8])

# To plot the error w.r.t no of tress use for predicting output
plot(RF_model)

# show the important variables by using plot
varImpPlot(RF_model)

# plotting the graph for real and predicted values of target variable in test data

plot(test$cnt, predictions_rf, xlab= 'Real_values', ylab= 'predicted_values',
main='Rf_model for test data')

# calculate error metrics MAPE to see the result of model
mape(test$cnt,predictions_rf)

##### Now saving the output of RF model
# creating the new variable "predicted" in the given data
day_deleted$predicted= with(day, predict(RF_model, day[,-8]))
day$predicted= day_deleted$predicted
rmExcept(c("day", "RF_model", "mape" , "day_deleted"))
write.csv(day, 'R_output.csv', row.names=F)
write.csv(day[c("season","yr","mnth","weathersit","temp","hum","windspeed","cnt","
predicted")], 'R_output.csv',row.names=F)
```

APPENDIX B – COMPLETE PYTHON CODE FILE

```
# load all the libraries required for our project
import os
import pandas as pd
import numpy as np
import datetime
import scipy.stats as stats
from pandas import Timestamp
import random as rand
import matplotlib.pyplot as plt
import seaborn as sns
from random import randrange, uniform
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
#libraries for Random Forest Model
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
from sklearn import metrics
from sklearn.metrics import r2_score, mean_squared_error
# libraies of Decision tree model
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
# set working directory
os.chdir("F:/data Scientist/project 1 bike renting/python")
os.getcwd()
# load data
day= pd.read_csv("day.csv", sep=',')
day.shape
# delete the 'instant' variable.
day.head(5)
day=day.drop('instant', axis=1)
day.dtypes
#Exploratory Data Analysis
# converting variables to proper data types
day['season']= day['season'].astype(object)
day['yr']= day['yr'].astype(object)
day['mnth']= day['mnth'].astype(object)
day['holiday']= day['holiday'].astype(object)
day['weekday']= day['weekday'].astype(object)
day['workingday']= day['workingday'].astype(object)
day['weathersit']= day['weathersit'].astype(object)
#factor_var=day[['season', 'yr','mnth', 'holiday', 'weekday', 'workingday',
'weathersit']]

########    VISUALISATIONS for given data
# Histograms to see the distribution of data in numerical variables
# Histogram of "temp"
plt.hist(day['temp'], bins=50, color= 'b');
plt.xlabel('temp', fontsize=20)
plt.ylabel('Frequency', fontsize=20)
plt.show()

# Histogram of "atemp"
```

```
plt.hist(day['atemp'], bins=50, color= 'b');
plt.xlabel('atemp', fontsize=20)
plt.ylabel('Frequency', fontsize=20)
plt.show()
# Histogram of "hum"
plt.hist(day['hum'], bins=50, color= 'b');
plt.xlabel('hum', fontsize=20)
plt.ylabel('Frequency', fontsize=20)
plt.show()
# Histogram of "windspeed"
plt.hist(day['windspeed'], bins=50, color= 'b');
plt.xlabel('windspeed', fontsize=20)
plt.ylabel('Frequency', fontsize=20)
plt.show()
# Histogram of "Bike rental count"
plt.hist(day['cnt'], bins=50, color= 'b');
plt.xlabel('cnt', fontsize=20)
plt.ylabel('Frequency', fontsize=20)
plt.show()
# from figures it is clear that 'hum' & 'windspeed' data is skewed so it may
contain outliers.
# create bar plots for catagorical variables w.r.t.'cnt'
# plot of 'season' vs 'cnt'
plt.bar(day['season'], day['cnt'], color='g')
plt.xlabel('season', fontsize=20)
plt.ylabel('count', fontsize=20)
plt.show()
# plot of 'month' vs 'cnt'
plt.bar(day['mnth'], day['cnt'], color='b')
plt.xlabel('month', fontsize=20)
plt.ylabel('count', fontsize=20)
plt.show()
# plot of 'year' vs 'cnt'
plt.bar(day['yr'], day['cnt'], color='g')
plt.xlabel('year', fontsize=20)
plt.ylabel('count', fontsize=20)
plt.show()
# plot of 'weekday' vs 'cnt'
plt.bar(day['weekday'], day['cnt'], color='b')
plt.xlabel('weekday', fontsize=20)
plt.ylabel('count', fontsize=20)
plt.show()
# plot of 'weathersit' vs 'cnt'
plt.bar(day['weathersit'], day['cnt'], color='b')
plt.xlabel('weathersit', fontsize=20)
plt.ylabel('count', fontsize=20)
plt.show()
# from the bar plots it is clear that high count of rental bikes is seen at season
'3' category, and year category '2'
# scatter plots for numeric variables
# scatter plot of 'temp' vs 'cnt'
plt.scatter(day['temp'], day['cnt'], marker='o');
plt.xlabel('temp', fontsize=20)
plt.ylabel('cnt', fontsize=20)
plt.show()
# scatter plot of 'atemp' vs 'cnt'
plt.scatter(day['atemp'], day['cnt'], marker='o');
plt.xlabel('atemp', fontsize=20)
plt.ylabel('cnt', fontsize=20)
plt.show()
# scatter plot of 'hum' vs 'cnt'
plt.scatter(day['hum'], day['cnt'], marker='o');
plt.xlabel('hum', fontsize=20)
plt.ylabel('cnt', fontsize=20)
plt.show()
# scatter plot of 'windspeed' vs 'cnt'
plt.scatter(day['windspeed'], day['cnt'], marker='o');
plt.xlabel('windspeed', fontsize=20)
plt.ylabel('cnt', fontsize=20)
plt.show()
# scatter plots of 'hum' and 'windspeed' are not linear.

####### MISSING VALUE ANALYSIS
pd.DataFrame(day.isnull().sum())
#OUTLIER ANALYSIS USING BOXPLOT
# plotting of box plot for visualising outlier
#   store numeric variables considered for oulier analysis in cnames
cnames= ['temp', 'atemp', 'hum', 'windspeed']
#box plot for numreic varibales
for i in cnames:
    plt.boxplot(day[i])
    plt.title("Boxplot of " +i)
    plt.show()
#create function to calculate atrributes of box plot and replace the outlies with
'na'
def find_outlier(var):
    q75,q25=np.percentile(day[var], [75,25])
    iqr= q75-q25
    min=q25-(iqr*1.5)
    max=q75+(iqr*1.5)
    day.loc[day[var]<min,var]=np.nan
    day.loc[day[var]>max,var]=np.nan
# impute outilers of 'hum' varibale
find_outlier('hum')
# see the missing valueS These missing values ar enothing but the outliers that we
have replaced with .na
day['hum'].isnull().sum()
# impute na value with mean
day.fillna(day.mean(), inplace=True)
day['hum'].isnull().sum()
# impute outilers of 'windspeed' varibale
find_outlier('windspeed')
# see the missing valueS
day['windspeed'].isnull().sum()
# impute na value with mean
day.fillna(day.mean(), inplace=True)
day['windspeed'].isnull().sum()
day.isnull().sum()
# create copy of data without outliers
day_without_outliers=day
day_without_outliers.head(5)


#######FEATURE SELECTION
```

```
# Correlation analysis-  it is used to check the dependancies between the
variables.
# Correlation plot
cnames_numeric= ['temp', 'atemp', 'hum', 'windspeed', 'casual', 'registered',
'cnt']
day_corr= day.loc[:,cnames_numeric]
# setting the  height and width of plot
f, ax=plt.subplots(figsize=(13,9))
# correaltion matrix
cor_matrix=day_corr.corr()
# plotting correlation plot
sns.heatmap(cor_matrix,mask=np.zeros_like(cor_matrix, dtype=np.bool),
            cmap=sns.diverging_palette(250,12,as_cmap=True),square=True, ax=ax)
# form the correlation it is clear that 'temp' and 'atemp' are correlated so will
drop 'atemp'
#####ANOVA Test-  it is used to check the dependancies between categorical and
numeric variables.
# load statsmodels library to perfom ANOVA test
import statsmodels.api as sm
from statsmodels.formula.api import ols
factor_data=day[['season', 'mnth','yr','holiday', 'weekday', 'workingday',
'weathersit']]
#perform the anova test
for i in factor_data:
    print(i)
    factor_anova=ols('cnt~factor_data[i]', data=day).fit()
    aov_table= sm.stats.anova_lm(factor_anova, type=2)
    print(aov_table)
# if we see the output of anova test it is clear that 'holiday', 'weekday' and
'workingday' are having the P value<0.05
# so we can remove these variables. we will perfom chi square test on these
variables again
##### Chi square test-  it is done on cateorical variables onl to check the
dependancies between them.
# chi square test of independance
from scipy.stats import chi2_contingency
factor_data=day[['season', 'mnth','yr','holiday','weekday','workingday',
'weathersit']]
for i in factor_data:
    for j in factor_data:
        if(i!=j):
            chi2,p,dof, ex=chi2_contingency(pd.crosstab(day[i],day[j]))
            while(p<0.05):
                print(i)
                print(j)
                print(p)
                break
# from the above output of chi square test it shows the variables with p value
<0.05
# so common variables are 'holiday', 'weekday' and 'workingday' so we will skip
these variables.
day=day_without_outliers
day.head(5)
# removing the correlated varibales ( selecting only relevant variables w.r.t.
target variable)
day_deleted= day.drop(['atemp', 'casual', 'registered', 'holiday','weekday',
'workingday','dteday'],axis=1)
```

```
day_deleted.head(5)

################# MODEL BUILDING
# PREPARING THE MACHINE LEARNING MODELS FOR OUR DATA
# As our target variable is numeric so we will select regression models
# I have selected the following three models for this project
# I will check the accuracy of three models and select the best one
# 1)Decision tree
# 2)Linear regression
# 3)Random forest

# DECISION TREE REGRESSION MODEL
# this model creates the decision tree like flow chart and creates the rules to
predict the target variable.
# divide the data into train and test
train, test=train_test_split(day_deleted, test_size=0.2)
# create decision tree for regression
fit=DecisionTreeRegressor(max_depth= 2).fit(train.iloc[:,0:7], train.iloc[:,7])
# apply model on test data
predictions_dt=fit.predict(test.iloc[:,0:7])
# define the funtion to find MAPE for our model
import numpy as np
def MAPE(y_true, y_pred):
    mape= np.mean(np.abs((y_true-y_pred)/y_true))*100
    return mape
# MAPE- mean absolute percentage error- it provides the error between real and
predicted values
# so accuracy of model is 100- MAPE
# calculate MAPE
MAPE(test.iloc[:,7], predictions_dt)
 #calculate MAE MSE and RMSE
print('Mean Absolute Error:',metrics.mean_absolute_error(test['cnt'],
predictions_dt))
print('Mean Squared Error:', metrics.mean_squared_error(test['cnt'],
predictions_dt))
print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error(test['cnt'],
predictions_dt)))

#Predictive performance of model using error metrics
#MAPE:26.96%
#Accuracy: 73.04%
#MAE: 1958.58
#MSE:5621933.20
#RMSE:2371.06
# if we see the performace metrics of decision tree results are quite good but
needs improvements.
#create dataframe of real and predicted values
Result_dt=pd.DataFrame({'real':test.iloc[:,7],'predicted': predictions_dt})
Result_dt
# LINNEAR REGRESSION MODEL
# this model is only used for numeric target variable
# it creates the coefficients of predictor variable w.r.t. target variable.
# Training the model
train, test=train_test_split(day_deleted, test_size=0.2)
# build linear Regression model
lr_model=sm.OLS(train.iloc[:,7], train.iloc[:,0:7].astype(float)).fit()
lr_model.summary()
```

```
# if we see the liner regressionmodel summary above we can see that adj r squared
value is 96.5% whcih is good
# also F-statistic is far greater than 1 and P value<0.05
# model is good but still we need to improve the accuracy
test.shape
# do the predictions
predictions_lr=lr_model.predict(test.iloc[:,0:7])
MAPE(test.iloc[:,7], predictions_lr)
# calculate MAE MSE and RMSE
print('Mean Absolute Error:',metrics.mean_absolute_error(test['cnt'],
predictions_lr))
print('Mean Squared Error:', metrics.mean_squared_error(test['cnt'],
predictions_lr))
print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error(test['cnt'],
predictions_lr)))

#Predictive performance of model using error metrics
#MAPE:22.76%
#Accuracy: 77.24%
#MAE:709.16
#MSE:912880.30
#RMSE:955.44
# the accuracy of this model is good as compared to decision tree model.
#create dataframe of real and predicted values
Result_lr=pd.DataFrame({'real':test.iloc[:,7],'predicted': predictions_lr})
Result_lr

########### RANDOM FOREST MODEL
# this is the improved version of decision tree model it uses many trees in one
model to improve the accuracy.
# it feeds error of one tree to another tree to improve the accuracy.
# divide the data into train and test
x=day_deleted.values[:,0:7]
y=day_deleted.values[:,7]
x_train, x_test, y_train, y_test= train_test_split(x,y,
test_size=0.2,random_state=0)
# building a random forest model
rf_model= RandomForestRegressor(n_estimators=70, random_state=0)
rf_model.fit(x_train, y_train)
# applying the model to predict cnt on test data
predictions_rf= rf_model.predict(x_test)
predictions_rf
# calculate MAE MSE and RMSE for our model
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predictions_rf))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, predictions_rf))
print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error(y_test,
predictions_rf)))
def MAPE(y_true, y_pred):
    mape= np.mean(np.abs((y_true-y_pred)/y_true))*100
    return mape

MAPE(y_test, predictions_rf)
#Predictive performance of model using error metrics

#MAPE:20.21%
#Accuracy: 79.79%
#MAE: 535.24
```

```
#MSE: 479937.73
#RMSE:692.77

#########  chosing the model among 3 models we have used
#From the Error metrics of all three models we can say that Random Forest model is
having high accuracy
# Other error metrics like MAE MSE and RMASE values are also less than other 2
models
# so we will choose Random Forest as our output model of this project to predict
the bike rental count.
# selecting the RF model as output model of this project
rf_model_sel= rf_model
#Predicting the 'cnt' for whole data using RF model
predictions_rf_out= rf_model_sel.predict(x)
predictions_rf_out
# creating the dataframe of real and predicted values of cnt for whole data
Result__rf_out=pd.DataFrame({'real':y,'predicted': predictions_rf_out})
Result__rf_out
# Adding the predicted values in data
day_deleted.insert(8,"predicted", predictions_rf_out, True)
day_deleted.head(5)
day.shape
# line plot of real and predicted values of cnt
plt.figure(figsize=(15,8))
plt.plot(y,color= 'g')
plt.xlabel('real= Green & predicted= Red', fontsize= '20')
plt.title= ("whadfhkafafka")
plt.plot(predictions_rf_out, color='r')
plt.show()
#plt.xlabel('Graph of real and predicted values of cnt')
# the graph showing real and predicted values by RF model is looking pretty good
their is less variation in both the values.
# scatter plot or Real and Predicted values of Bike rent count.
plt.scatter(y,predictions_rf_out,alpha=0.5)
plt.xlabel('Real', fontsize=20)
plt.ylabel('Predicted', fontsize=20)
plt.show()
# scatter plot of real and predicted values is also showing linear relationship.
# so we can say that Random Forest model is best for our study.

# Adding the predicted values in  main data
day.insert(15,"predicted", predictions_rf_out, True)

# Now we will save the output of model as a 'python_output_main' in the data we
selected after preprocessing

day_deleted.to_csv("python_out_preprocessed.csv", index=False)

# Now we will save the output as 'python_output.csv' fpr the main data
day.to_csv('python_output_main.csv', index= True)
```