Predicting Rental Bikes

Monesh soni

12 April 2020

Contents

1.1 Problem Statement

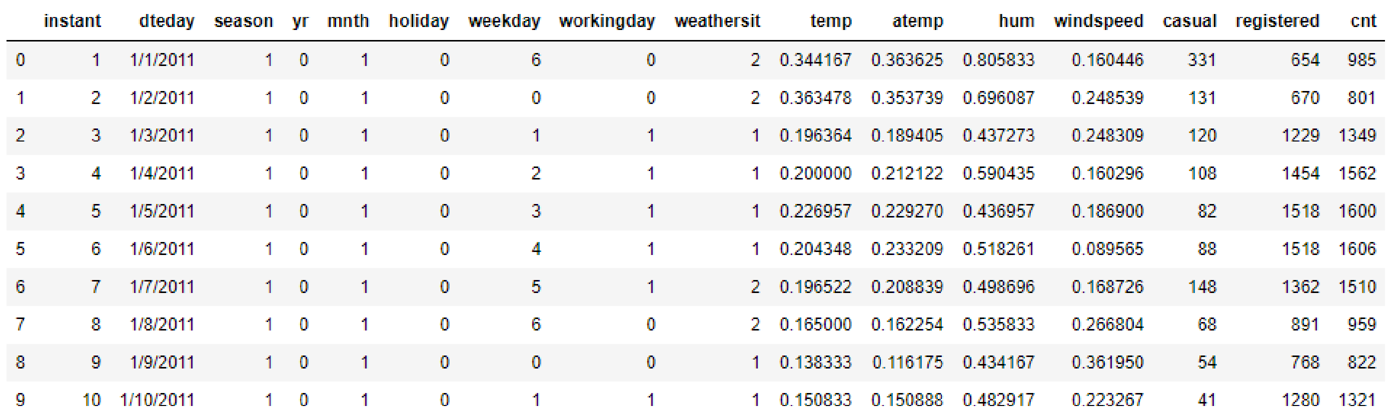
Predicting Bike rental count on daily based on the environmental and seasonal settings.

The aim of our project is predicting bike rental based on available data weather based or daily registered and casual customer based on thus data we predict Bike rental count on daily based .

1.2 Data

Our task is to build model which count of rental bike based on weather and season daily based.
Our task is to build Regression model which will give the daily count of rental bikes based on weather and season Given below is a 10 observation sample of the data set that we are using to predict the count:

Table 1.1: Bike Rental Sample Data



	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	1/1/2011	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	1/2/2011	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	1/3/2011	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	1/4/2011	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	5	1/5/2011	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600
5	6	1/6/2011	1	0	1	0	4	1	1	0.204348	0.233209	0.518261	0.089565	88	1518	1606
6	7	1/7/2011	1	0	1	0	5	1	2	0.196522	0.208839	0.498696	0.168726	148	1362	1510
7	8	1/8/2011	1	0	1	0	6	0	2	0.165000	0.162254	0.535833	0.266804	68	891	959
8	9	1/9/2011	1	0	1	0	0	0	1	0.138333	0.116175	0.434167	0.361950	54	768	822
9	10	1/10/2011	1	0	1	0	1	1	1	0.150833	0.150888	0.482917	0.223267	41	1280	1321

As you can see in the table below we have the following 14 variables, using which we have to correctly predict daily based bike rental :

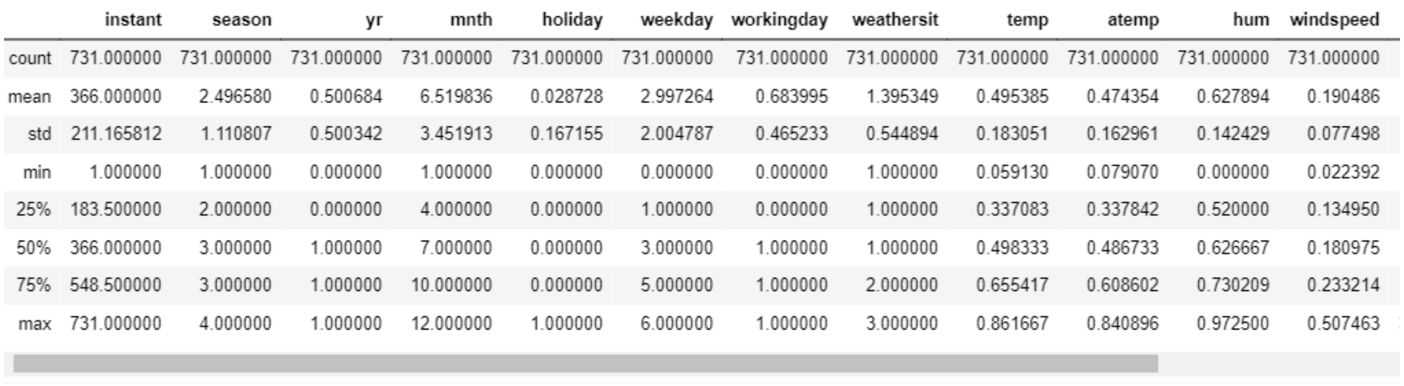
Table 1.2: Bike Rental Predictors variable

s.no	Variables
1	Dteday
2	Season
3	Yr
4	Mnth
5	Holiday
6	Weekday
7	Workingday
8	Weathersit
9	Temp
10	Atemp
11	Hum
12	Windspeed
13	Casual
14	Registered

Describe of dataset :

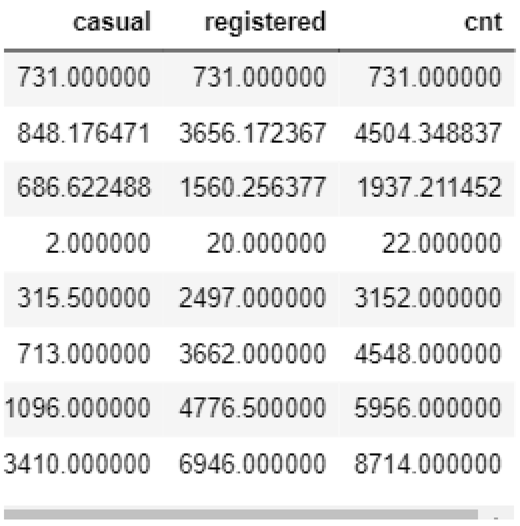
Describe returns the count, mean, standard deviation, minimum and maximum values and the quantiles of the Bike rental dataset.

Bike rental sample data Table 1.3 column(1-12)



	instant	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000
mean	366.000000	2.496580	0.500684	6.519836	0.028728	2.997264	0.683995	1.395349	0.495385	0.474354	0.627894	0.190486
std	211.165812	1.110807	0.500342	3.451913	0.167155	2.004787	0.465233	0.544894	0.183051	0.162961	0.142429	0.077498
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.059130	0.079070	0.000000	0.022392
25%	183.500000	2.000000	0.000000	4.000000	0.000000	1.000000	0.000000	1.000000	0.337083	0.337842	0.520000	0.134950
50%	366.000000	3.000000	1.000000	7.000000	0.000000	3.000000	1.000000	1.000000	0.498333	0.486733	0.626667	0.180975
75%	548.500000	3.000000	1.000000	10.000000	0.000000	5.000000	1.000000	2.000000	0.655417	0.608602	0.730209	0.233214
max	731.000000	4.000000	1.000000	12.000000	1.000000	6.000000	1.000000	3.000000	0.861667	0.840896	0.972500	0.507463

Bike rental sample data Table 1.4 column(12-15)



casual	registered	cnt
731.000000	731.000000	731.000000
848.176471	3656.172367	4504.348837
686.622488	1560.256377	1937.211452
2.000000	20.000000	22.000000
315.500000	2497.000000	3152.000000
713.000000	3662.000000	4548.000000
1096.000000	4776.500000	5956.000000
3410.000000	6946.000000	8714.000000

Chapter 2
Methodology

2.1 Pre Processing

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis. To start this process we will first try and look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable.

2.1.1 Univariate Analysis

Univariate analysis is perhaps the simplest form of statistical analysis.The key fact is that only one variable is involved in univariate analysis.
In Figure 2.1 and 2.2 we have plotted the probability density functions numeric variables present in the data including target variable cnt.

i. Target variable cnt is normally distributed
ii. Independent variables like 'temp','atemp', and 'regestered' data is distributed normally.
iii. Independent variable 'casual' data is slightly skewed to the right so, there is chances of getting outliers.
iv. Other Independent variable 'hum' data is slightly skewed to the left , here data is already in normalize form so outliers are discarded.

Figure 2.1 Distribution of target variable (CNT) (python code in Appendix B)

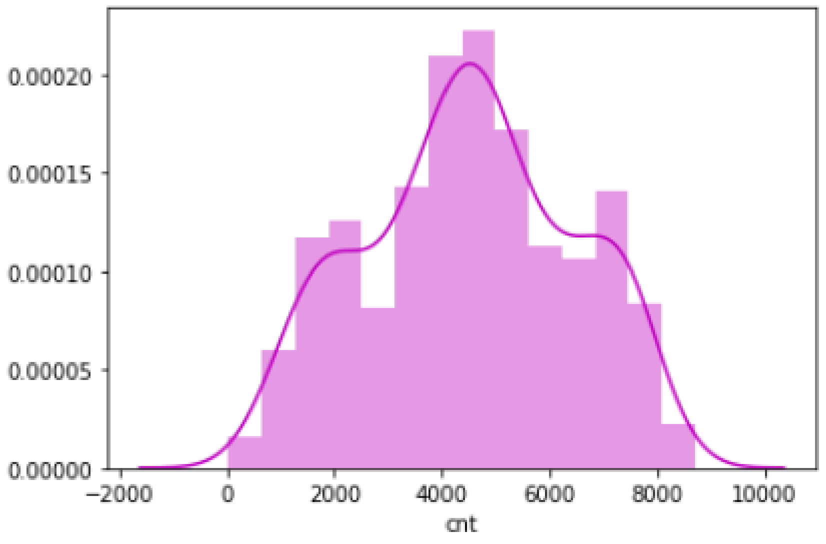
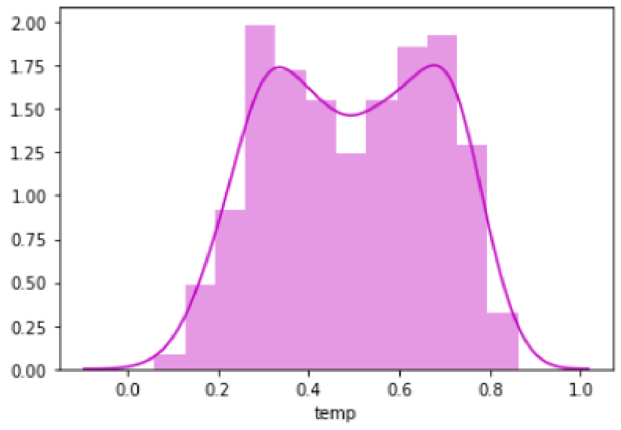
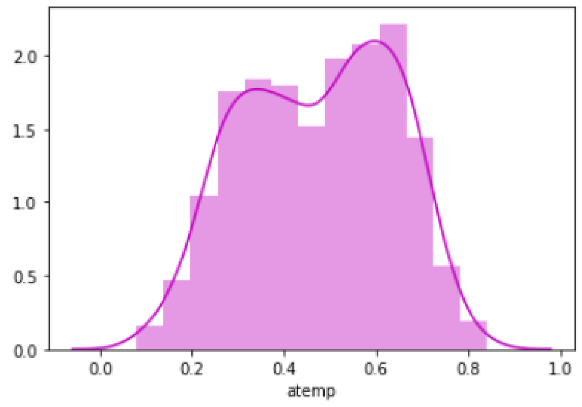
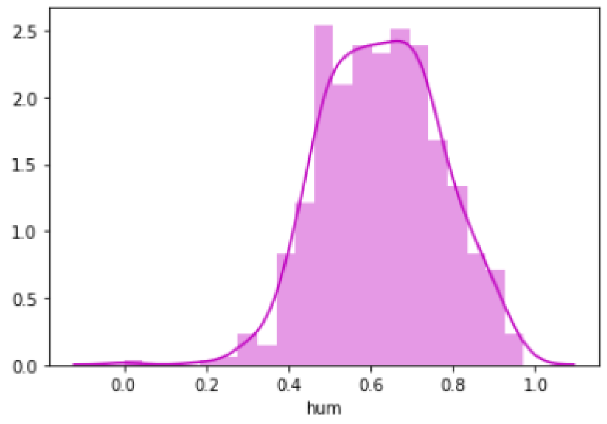
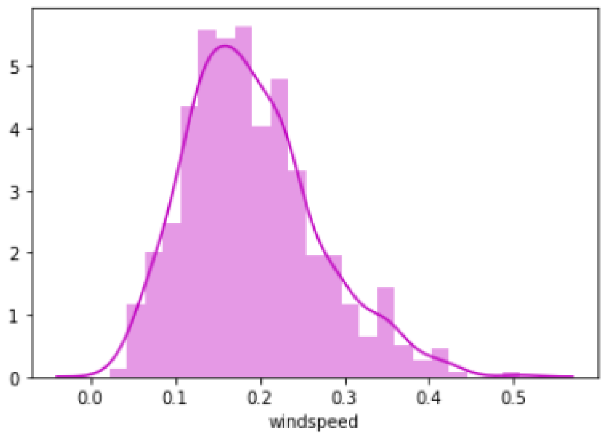


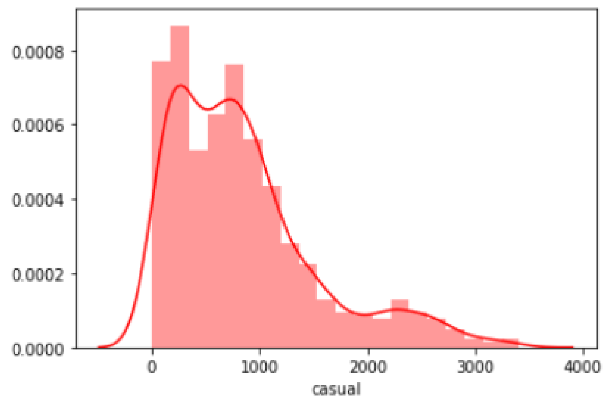
Figure 2.2 showing distribution of dependent variables (python code in Appendix B)

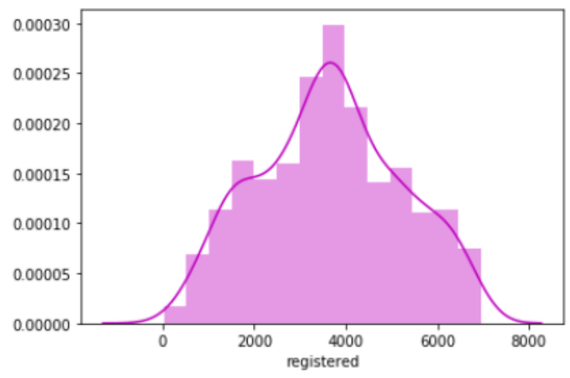

2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
temp


2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
atemp


2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
hum


5
4
3
2
1
0
0.0
0.1
0.2
0.3
0.4
0.5
windspeed


0.0008
0.0006
0.0004
0.0002
0.0000
0
1000
2000
3000
4000
casual


0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0
2000
4000
6000
8000
registered

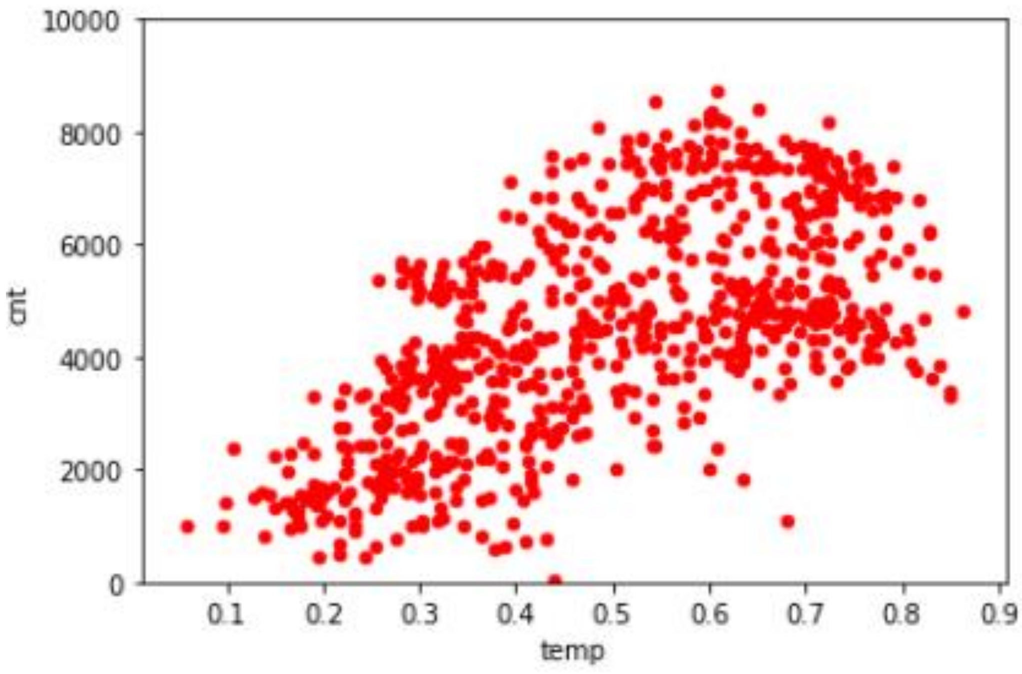
2.1.2 Bivariate Analysis

Bivariate analysis is one of the simplest forms of quantitative analysis.Bivariate analysis involve two variable. Bivariate analysis helpful in testing simple hypotheses of association.

Let's we check now numeric variable temp or target variable cnt Bivariate relationshipBelow ggpair graph is showing clearly that relationship between independent variables ‘temp’ and ‘atemp’ are very strong.

1.The relationship between ‘hum’ , ‘windspeed’ with target variable ‘cnt’

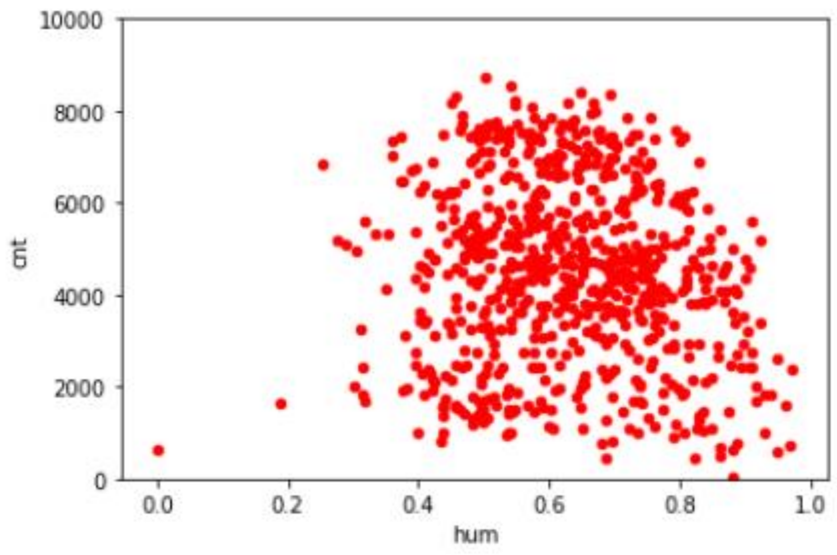
Figure 2.3 relationship between numeric variables (python code in Appendix B)



Scatter plot is showing bivariate relationship between temp or cnt variable is high positive relationship

2.Relation between Numerical Variable 'hum' and target variable 'cnt'

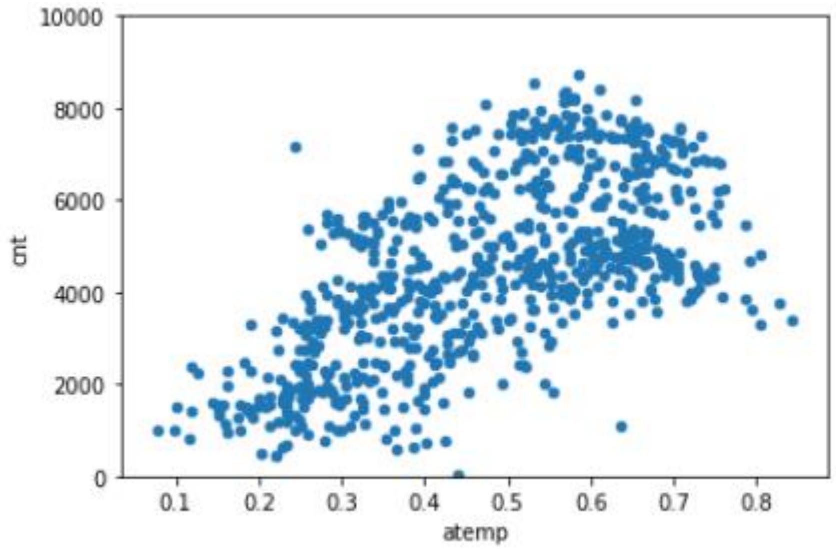
Figure 2.4 relationship between numeric variables (python code in Appendix B)



scatter plot is showing average relationship between hum variable or target variable cnt

3. Relation between Numerical Variable 'atemp' and target variable 'cnt'

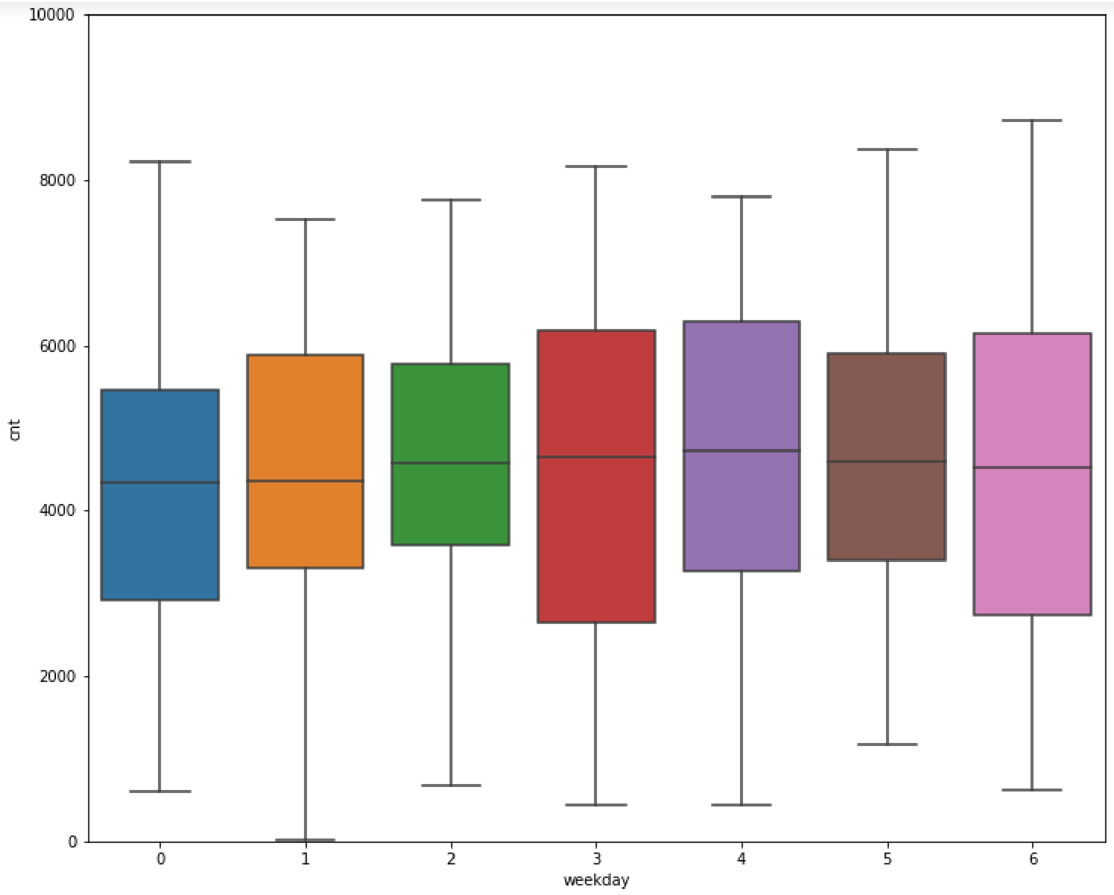
Figure 2.5 relationship between numeric variables (python code in Appendix B)



Scatter plot showing highly positive relationship between “atemp” variable or target variable “cnt”

4. Bivariate relationship weekday is catogaries data or target variable cnt box plot to show distributions with respect to categories. box plot 'Weekdays' with 'cnt'

Figure 2.6 relationship between categories variables (python code in Appendix B)

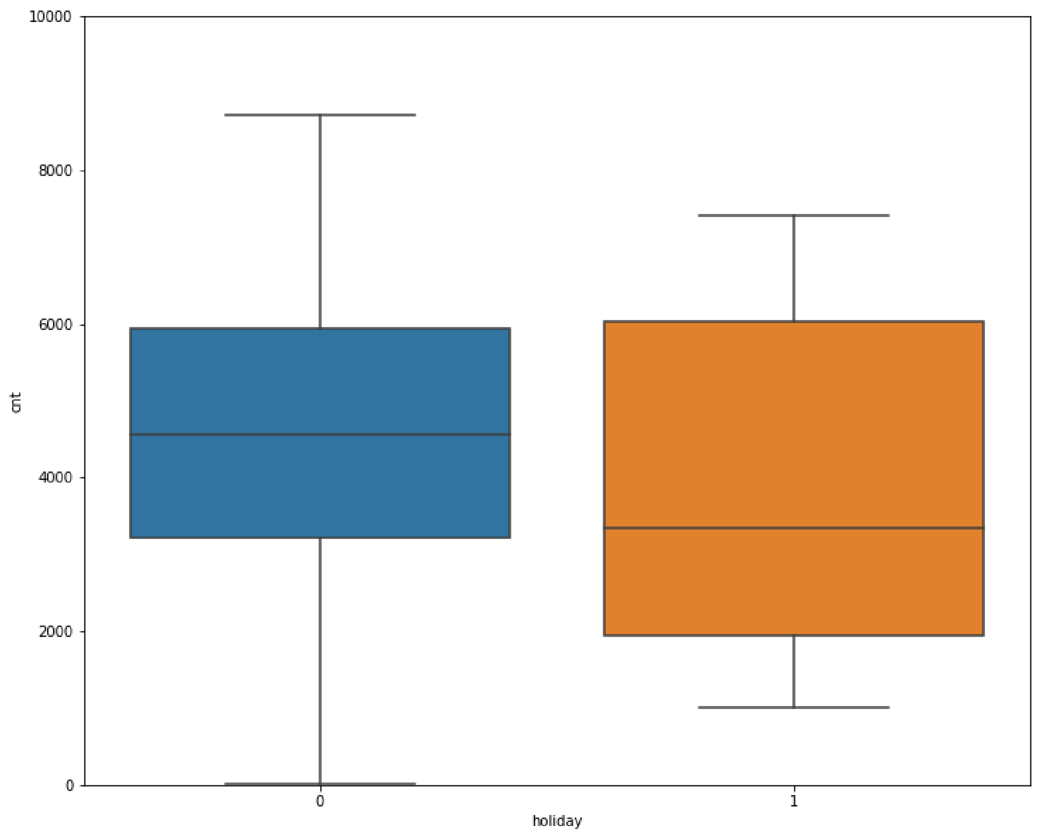


Boxplot shows most median value between 4000-5000

5.Bivariate relationship : holiday is catogaries data or target variable cnt box plot to show distributions with respect to categories

box plot 'holiday' with 'cnt'

Figure 2.7 relationship between categories variables (python code in Appendix B)



2.2 Pre-processing technique

It is a data mining technique that transforms raw data into an understandable format. Raw data(real world data) is always incomplete and that data cannot be sent through a model. That would cause certain errors. That is why we need to preprocess data before sending through a model.

2.2.1 Missing Value Analysis

Missing values in data is a common phenomenon in real world problems. Knowing how to handle missing values effectively is a required step to reduce bias and to produce powerful models.

missing values, occur when no data value is stored for the variable in an observation. Missing data are a common occurrenceand can have a significant effect on the conclusions that can be drawn from the data.

Below table illustrate no missing value present in the Bike rental dataset.

missing values Table

	Total	Percent
cnt	0	0.0
registered	0	0.0
casual	0	0.0
windspeed	0	0.0
hum	0	0.0
atemp	0	0.0
temp	0	0.0
weathersit	0	0.0
workingday	0	0.0
weekday	0	0.0
holiday	0	0.0
mnth	0	0.0
yr	0	0.0
season	0	0.0
dteday	0	0.0
instant	0	0.0

2.2.2 Outlier Analysis

The Other steps of Preprocessing Technique is Outliers analysis , an outlier is an observation point that is distant from other observations. Outliers in data can distort predictions and affect the accuracy, if you don't detect and handle them appropriately especially in regression models..

As we are observed in fig 2.2 the data is skewed so, there is chance of outlier in independent variable 'casual' , one of the best method to detect outliers is Boxplot

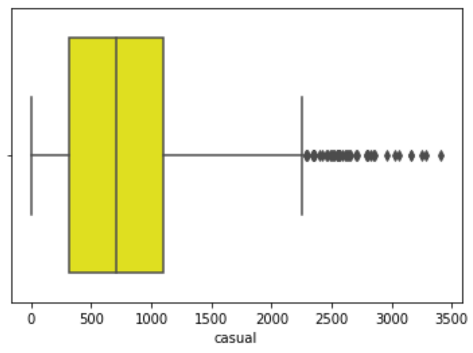
Fig 2.4 shows presence of Outliers in variable 'casual' and relationship between 'casual' and 'cnt' before removing Outliers

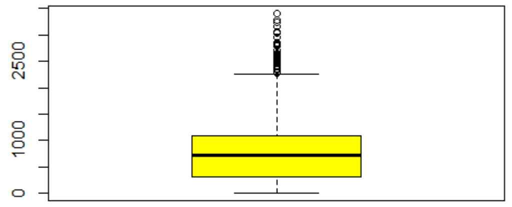
Fig 2.5 shows boxplot of 'casual' after removing outliers and relationship between 'casual' and 'cnt' after removing outliers

---

Boxplot :- boxplot is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles

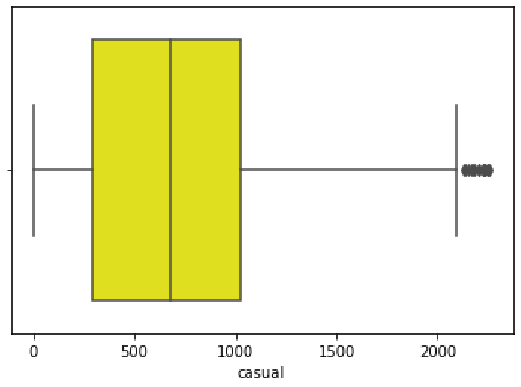
Figure 2.4 'casual' Baxoplot and relation between 'cnt' and' casual' (R or python code in Appendix)

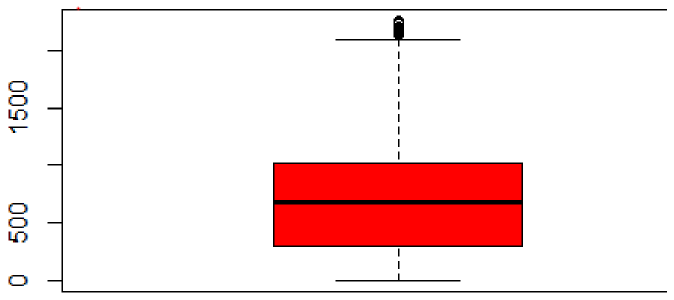




Below Boxplot in after removing outliers in casual variable

Figure 2.5 'casual' Boxplot and relation between 'casual' and 'cnt' (R or python code in Appendix)





Since there is significant difference between Pearson coefficient correlation between before and after outlier detection for 'casual' and 'cnt' and losing nearly 40 observation

2.2.3 Features Selections

Feature selection or variable subset selection, is the process of selecting a subset of relevant.

Machine learning works on a simple rule – if you put garbage in, you will only get garbage to come out. By garbage here, I mean noise in data.

This becomes even more important when the number of features are very large. You need not use every feature at your disposal for creating an algorithm. You can assist your algorithm by feeding in only those features that are really important. I have myself witnessed feature subsets giving better results than complete set of feature for the same algorithm or – "Sometimes, less is better!".

Why should we require feature selection

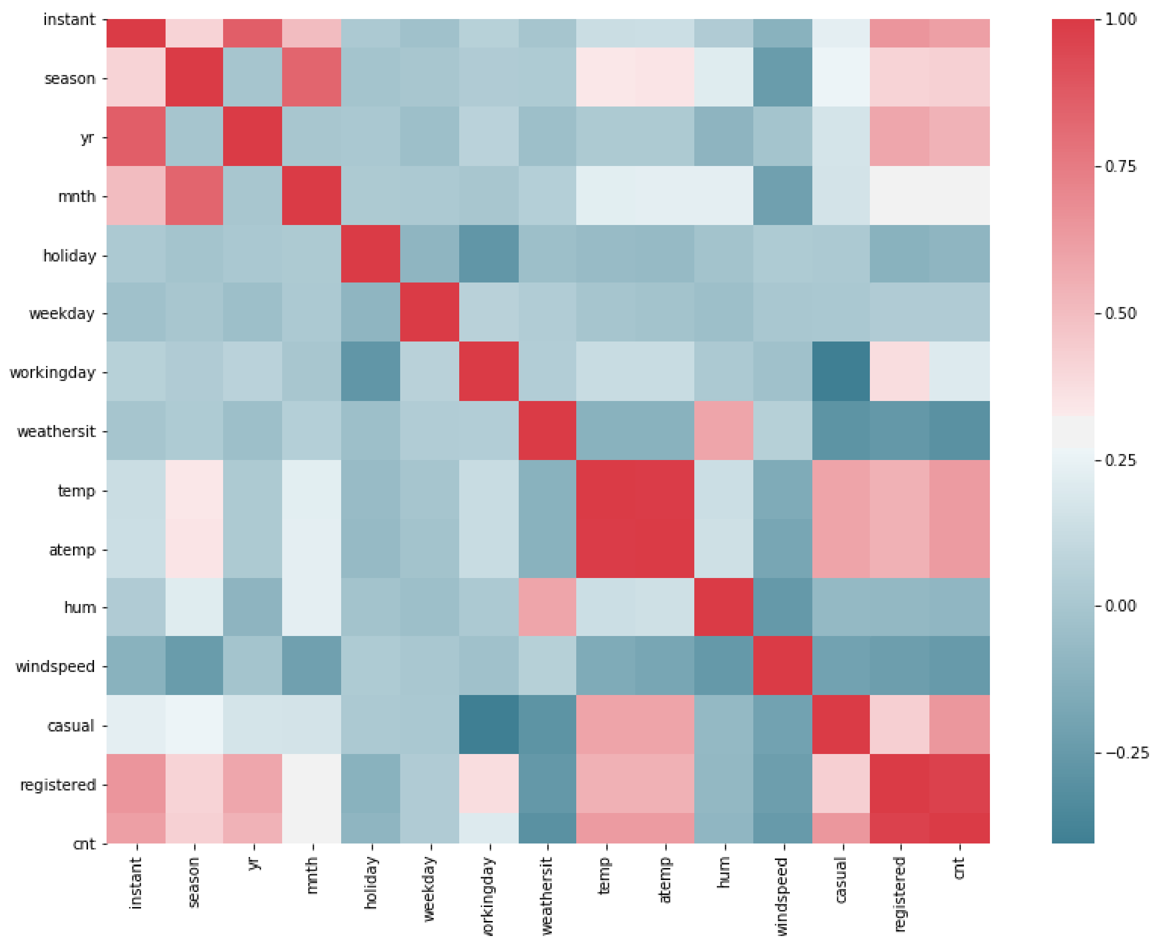
- simplification of models to make them easier to interpret by researchers/users
- shorter training times.
- to avoid the curse of dimensionality,
- enhanced generalization by reducing overfitting (formally, reduction of variance)

We should consider the selection of feature for model based on below criteria

i. The relationship between two independent variable should be less and
ii. The relationship between Independent and Target variables should be high.

Below fig 2.6 illustrates that relationship between all numeric variables using heatmap plot .

Figure 2.6 correlation plot of variables (python code in Appendix B)



Color dark red indicates there is strong positive relationship and if darkness is decreasing indicates relation between variables are decreasing.

Color dark blue indicates there is strong negative relationship and if darkness is decreasing indicates relationship between variables are decreasing.

Above correlation plot is showing “temp” or “atemp” variable is highly positive correlated the are red color represented in above plot or “hum” variable highly negative relationship between target variable “cnt”

2.2.4 Dimensionality Reduction for variables

Above Fig 2.6 is showing

there is strong positive relationship between independent variables ‘temp’ and ‘atemp’   so considering any one feature enough to predict the better Bike rental count .

And it is also showing there is almost no relationship between independent variable ‘hum’ and dependent variable ‘cnt’. so, ‘hum’ is not so important to predict Bike rental count .

So we drop thus two independent variable “atemp” or “hum” variable in our dataset using of drop() function .

2.2.5 Features Scaling

The word “normalization” is used informally in statistics, and so the term normalized data can have multiple meanings. In most cases, when you normalize data you eliminate the units of measurement for data, enabling you to more easily compare data from different places. Some of the more common ways to normalize data include:

Transforming data using a z-score or t-score. This is usually called standardization. In the vast majority of cases, if a statistics textbook is talking about normalizing data, then this is the definition of “normalization” they are probably using.

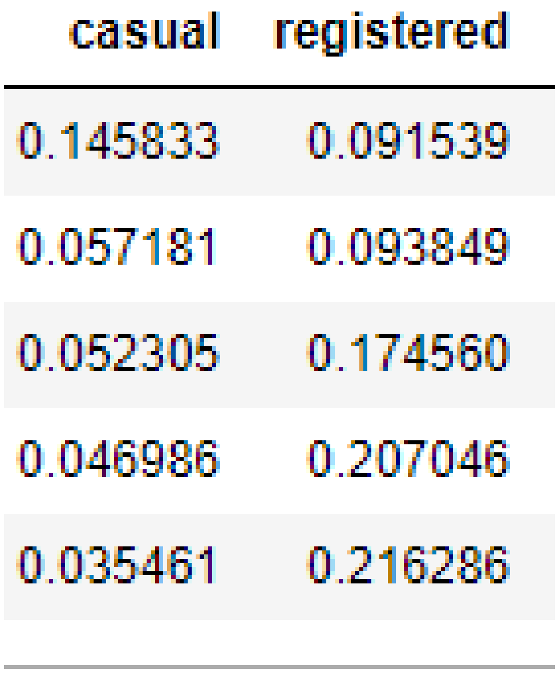
Rescaling data to have values between 0 and 1. This is usually called feature scaling. One possible formula to achieve this is.

$$\boxed{x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}}$$

In rental dataset numeric variables like ‘temp’ , ‘atemp’ ,’hum’ and ‘ windspeed’ are in normalization form so , we have to Normalize two variables ‘casual’ and ‘registered’

After normalize ‘casual’ and ‘registered’ variables look like in table below where all values between 0 and 1

Table Normalization of ‘casual’ and ‘registered



casual	registered
0.145833	0.091539
0.057181	0.093849
0.052305	0.174560
0.046986	0.207046
0.035461	0.216286

Chapter 3

Modelling

3.1 Model Selection

In earlier stage of analysis we have come to understand that few variables like 'temp' ,'casual, 'registered ' are going to play key role in model development for model development dependent variable may fall under below categories

i. Nominal
ii. Ordinal
iii. Interval
iv. Ratio

In our case dependent variable is interval so, the predictive analysis that we can perform is **Regression** Analysis

We will start our model building from Decision Tree .

3.1.1 Evaluating Regression Model

The main concept of looking at what is called **residuals** or difference between our predictions f(x[I,]) and actual outcomes y[i].

We are using three methods to evaluating performance of model

i. **MAPE** : The mean absolute percentage error, also known as mean absolute percentage deviation, 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. (Mean Absolute Percent Error) measures the size of the error in percentage terms. It is calculated as the average of the unsigned percentage error.

$$\left(\frac{1}{n}\sum\frac{|Actual - Forecast|}{|Actual|}\right)*100$$

ii. **MSE: mean squared error** (**MSE**) or **mean squared deviation** (**MSD**) of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. MSE is a risk function, corresponding to the expected value of the squared error loss. The fact that MSE is almost always strictly positive (and not zero) is be because of randomness or because the estimator does not account for information that could produce a more accurate estimate.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

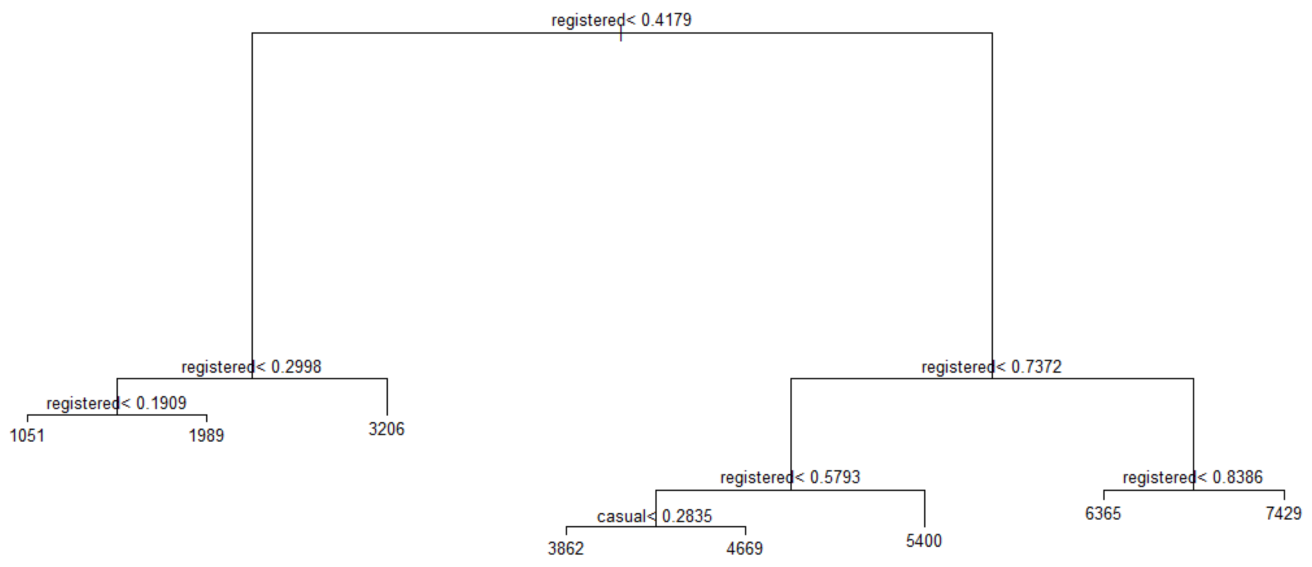
iii. **RMSE :** (Root Mean Square Error) is a frequently used measure of the difference between values predicted by a model and the values actually observed from the environment that is being modelled. The **root-mean-square deviation** (**RMSD**) or **root-mean-square error** (**RMSE**) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed. The RMSD represents the square root of the second sample moment of the differences between predicted values and observed values or the quadratic mean of these differences.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_{obs,i} - X_{model,i})^2}{n}}$$

3.2 Decision Tree

A tree has many analogies in real life, and turns out that it has influenced a wide area of **machine learning**, covering both **classification and regression**. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions.

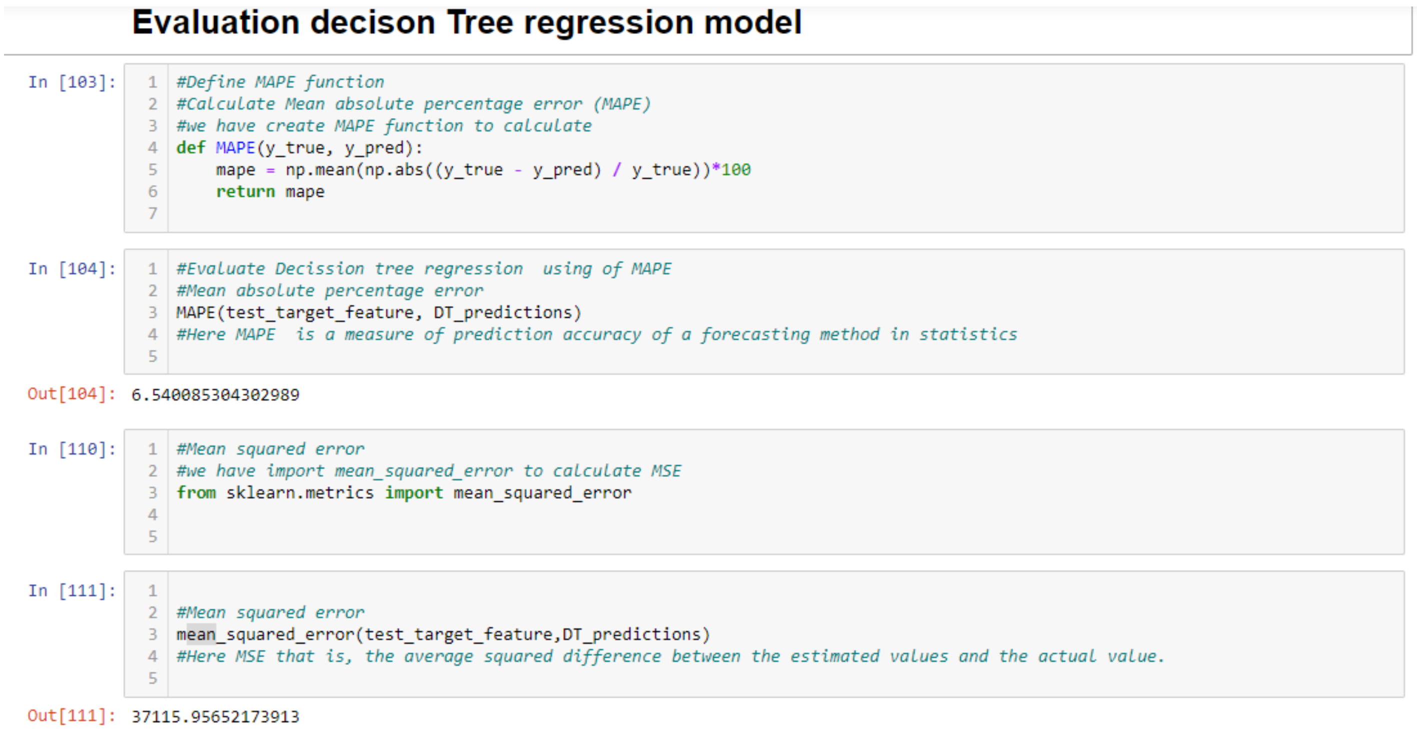
Figure 3.2.1 Decision Tree Algorithm



Look at the above figure 3.2.1 here decision tree is using only two predictors variables to predict the model , which is not very impressive here the model is overfitted and biased towards only two predictors i.e 'casual' and 'registered' .

3.2.1 Evaluation of Decision Tree Model

Figure 3.2.2 Evaluation of Decision Tree using MAPE and RMSE



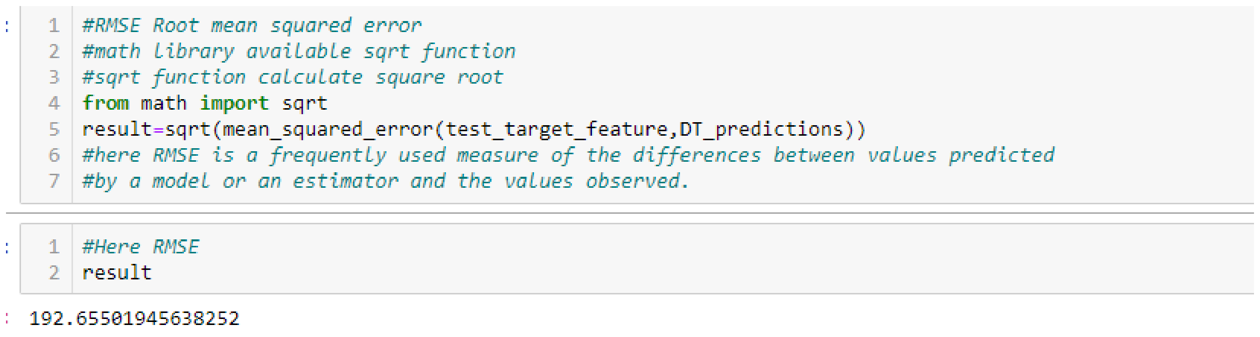
Evaluation decison Tree regression model

```
In [103]:
#Define MAPE function
#Calculate Mean absolute percentage error (MAPE)
#we have create MAPE function to calculate
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape

```

```
In [104]:
#Evaluate Decission tree regression  using of MAPE
#Mean absolute percentage error
MAPE(test_target_feature, DT_predictions)
#Here MAPE  is a measure of prediction accuracy of a forecasting method in statistics

Out[104]: 6.540085304302989
```

```
In [110]:
#Mean squared error
#we have import mean_squared_error to calculate MSE
from sklearn.metrics import mean_squared_error

```

```
In [111]:

#Mean squared error
mean_squared_error(test_target_feature,DT_predictions)
#Here MSE that is, the average squared difference between the estimated values and the actual value.

Out[111]: 37115.95652173913
```



```
:
#RMSE Root mean squared error
#math library available sqrt function
#sqrt function calculate square root
from math import sqrt
result=sqrt(mean_squared_error(test_target_feature,DT_predictions))
#here RMSE is a frequently used measure of the differences between values predicted
#by a model or an estimator and the values observed.
```

```
:
#Here RMSE
result

: 192.65501945638252
```

In Figure 3.2.3 Model Accuracy is   1- 6.54 = 0.934   which is nearly 93.4% it is quite good but RMSE is 192.65 which is very high so it's clearly stating that our Decision Tree Model is Overfitted and it working well for training data but won't predict good for new set of data. To overcome this overfit we have to tune the model using Random Forest Algorithm.

3.3 Random Forest

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

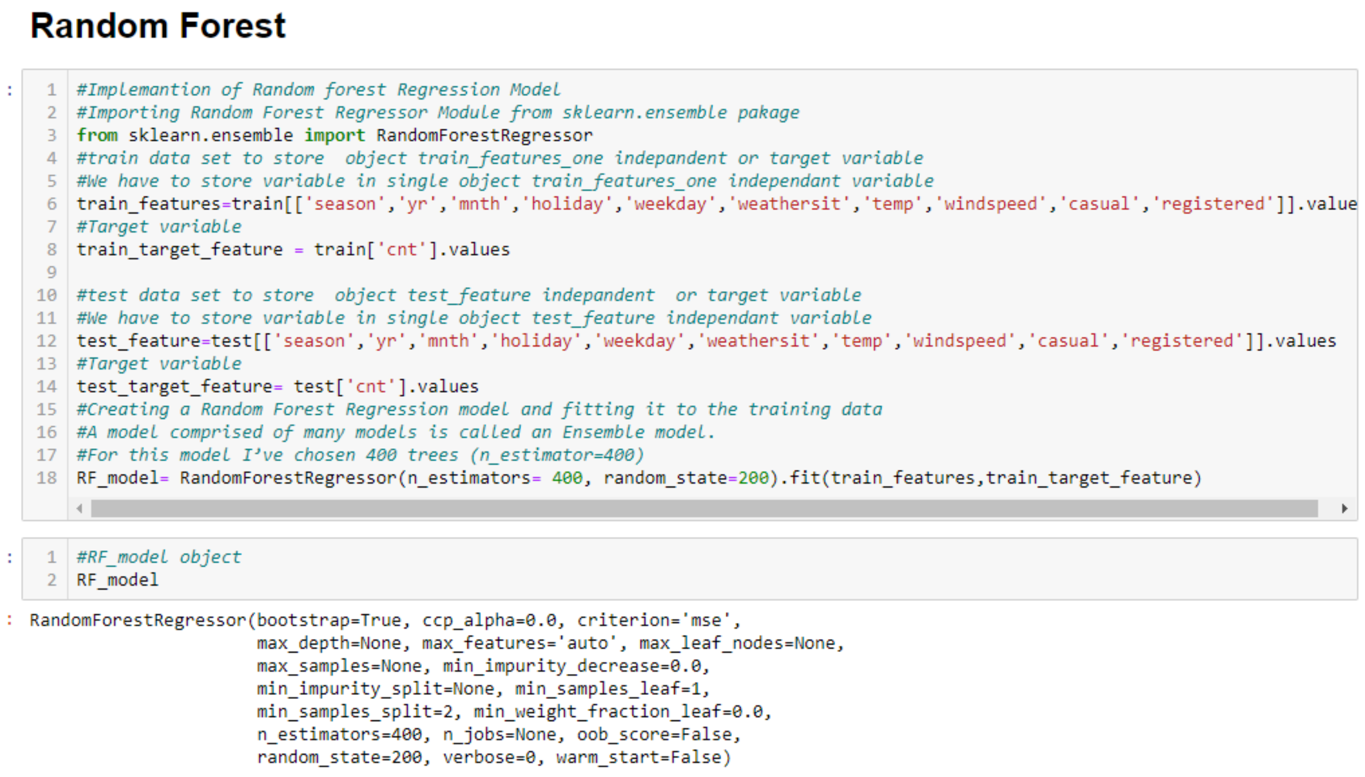
Put simply: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Random forest functions in below way

i. Draws a bootstrap sample from training data.
ii. For each sample grow a decision tree and at each node of the tree
    a. Randomly draws a subset of mtry variable and p total of features that are available
    b. Picks the best variable and best split from the subset of mtry variable
    c. Continues until the tree is fully grown.

As we saw in section 3.2 Decision tree is overfitting and its accuracy MAPE and RMSE is also require to improve the performance of the regression model develop model using Random Forest Regressor we try to improve the model.

Figure 3.3.1 Random Forest Implementation

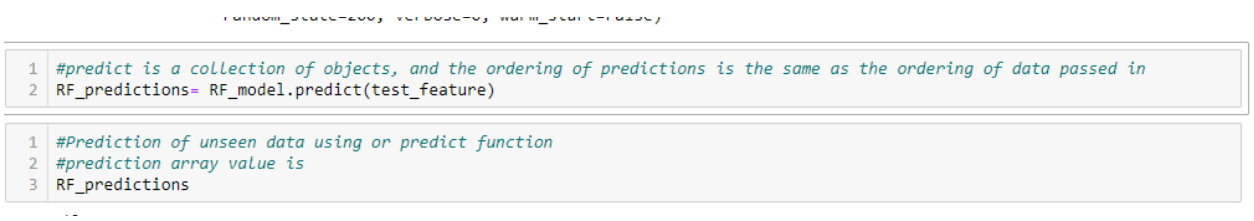


Random Forest

```
#Implemantion of Random forest Regression Model
#Importing Random Forest Regressor Module from sklearn.ensemble pakage
from sklearn.ensemble import RandomForestRegressor
#train data set to store  object train_features_one independant or target variable
#We have to store variable in single object train_features_one independant variable
train_features=train[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].value
#Target variable
train_target_feature = train['cnt'].values

#test data set to store  object test_feature independant  or target variable
#We have to store variable in single object test_feature independant variable
test_feature=test[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].values
#Target variable
test_target_feature= test['cnt'].values
#Creating a Random Forest Regression model and fitting it to the training data
#A model comprised of many models is called an Ensemble model.
#For this model I’ve chosen 400 trees (n_estimator=400)
RF_model= RandomForestRegressor(n_estimators= 400, random_state=200).fit(train_features,train_target_feature)
```

```
#RF_model object
RF_model
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=400, n_jobs=None, oob_score=False,
                      random_state=200, verbose=0, warm_start=False)
```



```
random_state=200, verbose=0, warm_start=False)
```

```
#predict is a collection of objects, and the ordering of predictions is the same as the ordering of data passed in
RF_predictions= RF_model.predict(test_feature)
```

```
#Prediction of unseen data using or predict function
#prediction array value is
RF_predictions
```

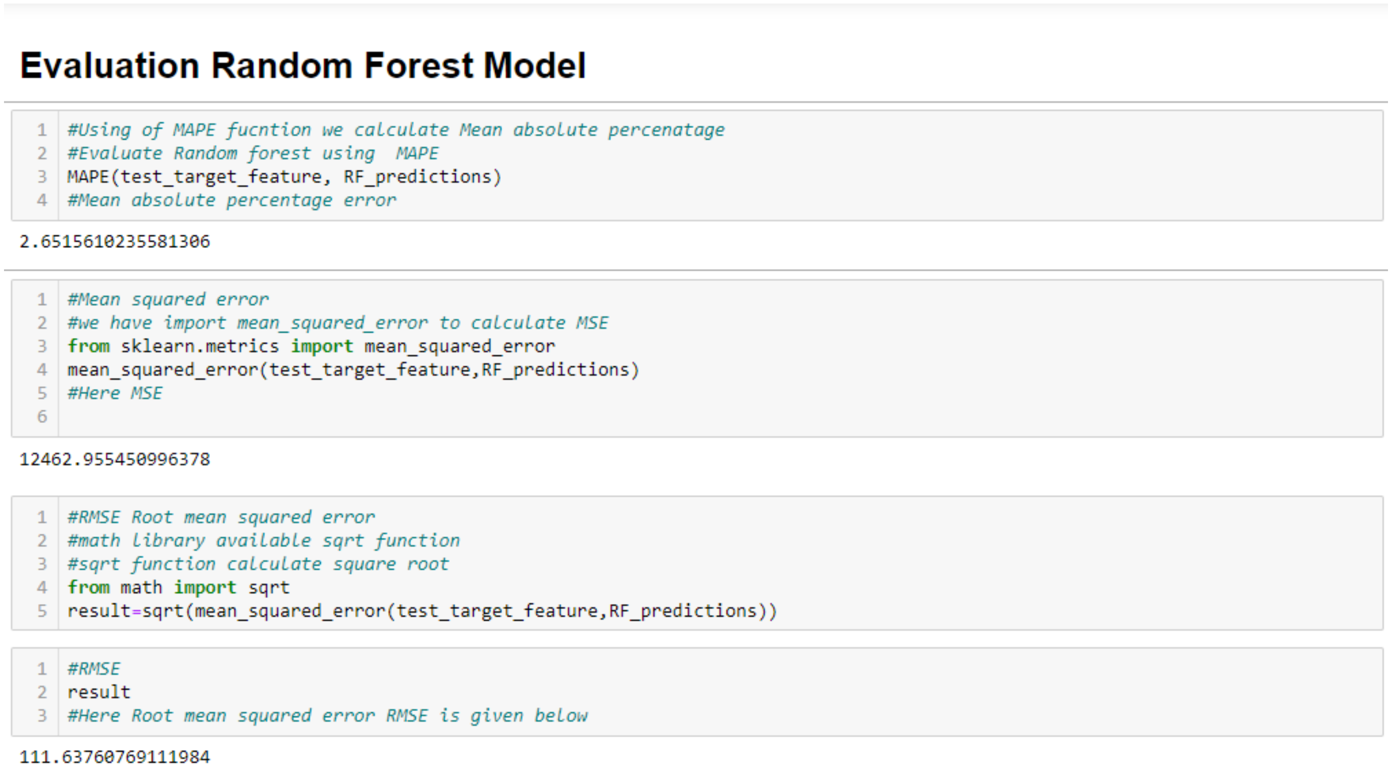
Mtry : Number of variables to split at each node i.e. 7.

Node size : size of each node is 10

Our Random Forest model is looking quite good where it utilized maximum variables to predict the count values

3.3.1 Evaluation of Random Forest

Figure 3.2.2 Random Forest Evaluation



Evaluation Random Forest Model

```
#Using of MAPE fucntion we calculate Mean absolute percenatage
#Evaluate Random forest using  MAPE
MAPE(test_target_feature, RF_predictions)
#Mean absolute percentage error
```

```
2.6515610235581306
```

```
#Mean squared error
#we have import mean_squared_error to calculate MSE
from sklearn.metrics import mean_squared_error
mean_squared_error(test_target_feature,RF_predictions)
#Here MSE

```

```
12462.955450996378
```

```
#RMSE Root mean squared error
#math library available sqrt function
#sqrt function calculate square root
from math import sqrt
result=sqrt(mean_squared_error(test_target_feature,RF_predictions))
```

```
#RMSE
result
#Here Root mean squared error RMSE is given below
```

```
111.63760769111984
```

Fig 3.2.2 shows Random Forest model performs better than Decision tree on both training and test data and well also improve

(MAPE = 2.61) Accuracy =100-2.61: so model Accuracy =97.39

and RMSE decrease the RMSE=(111) of the model which is quite impressive compression with Decision Tree model.

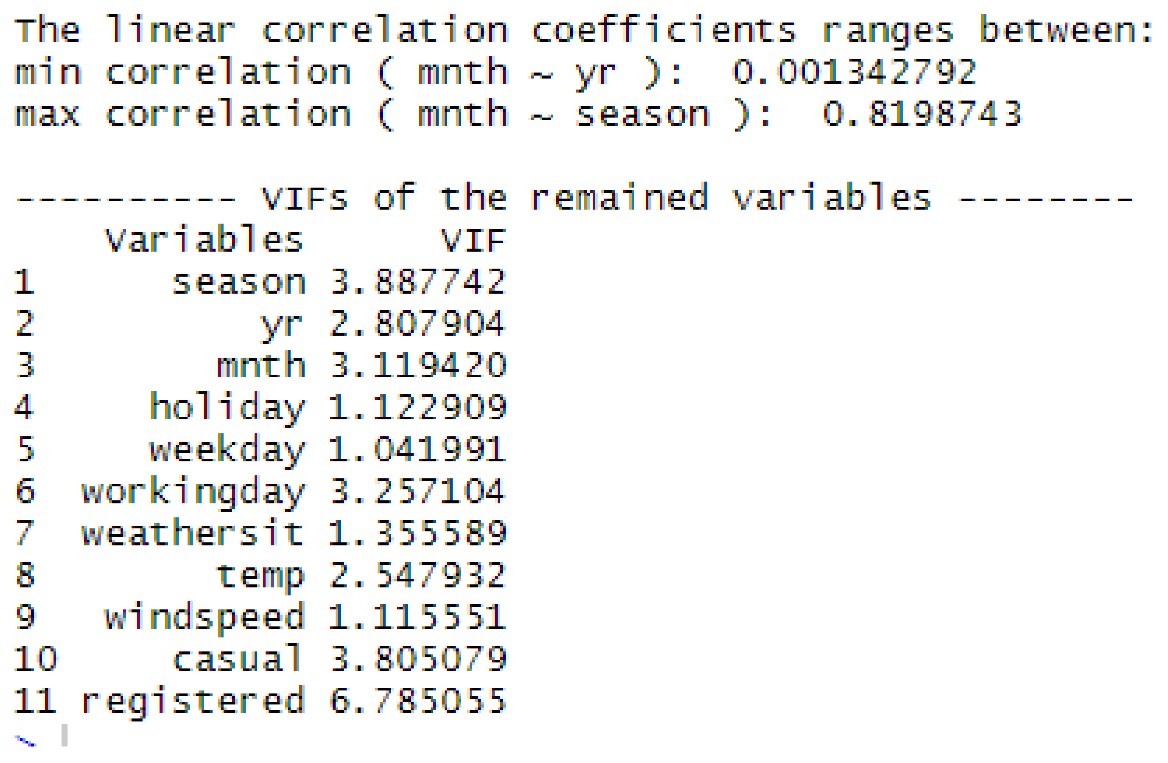
Using Linear Regression we will predict the Target variable 'cnt ' values and compare with Decision Tree model or Random Forest model.

3.4 Linear Regression

Multiple linear regression is the most common form of linear regression analysis. As a predictive analysis, the multiple linear regression is used to explain the relationship between one continuous dependent variable and two or more independent variables. The independent variables can be continuous or categorical.

VIF (Variance Inflation factor) : the variance inflation factor is the quotient of the variance in a model with multiple terms by the variance of a model with one term alone. It quantifies the severity of multicollinearity in an ordinary least squares regression analysis. As Linear regression will work well if multicollinearity between the Independent variables are less.

Figure 3.4.1 Multi collinearity between Independent variables



```
The linear correlation coefficients ranges between:
min correlation ( mnth ~ yr ):  0.001342792
max correlation ( mnth ~ season ):  0.8198743

---------- VIFs of the remained variables --------
    Variables      VIF
1      season 3.887742
2          yr 2.807904
3        mnth 3.119420
4     holiday 1.122909
5     weekday 1.041991
6  workingday 3.257104
7  weathersit 1.355589
8        temp 2.547932
9   windspeed 1.115551
10     casual 3.805079
11 registered 6.785055
```

In the above figure it is showing there is strong correlation between two independent variable 'mnth' and 'season' so , it is enough to consider any one variable.

Figure 3.4.2 Multiple Linear Regression Model

Linear Regression Model

```python
#We take here feature or data set
#train data set to store  object train_features_one independant or target variable
#We have to store variable in single object train_features_one independant variable
train_features = train[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].val
#Target variable
train_target_feature = train['cnt'].values

#test data set to store  object test_feature independent  or target variable
#We have to store variable in single object test_feature independant variable
test_feature = test[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].values
#Target variable
test_target_feature= test['cnt'].values
```

```python
#statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models,
#as well as for conducting statistical tests, and statistical data exploration
import statsmodels.api as sm
#implement of Linear regression model
#OLS stands for Ordinary Least Squares
LR_model= sm.OLS(train_target_feature, train_features).fit()

```

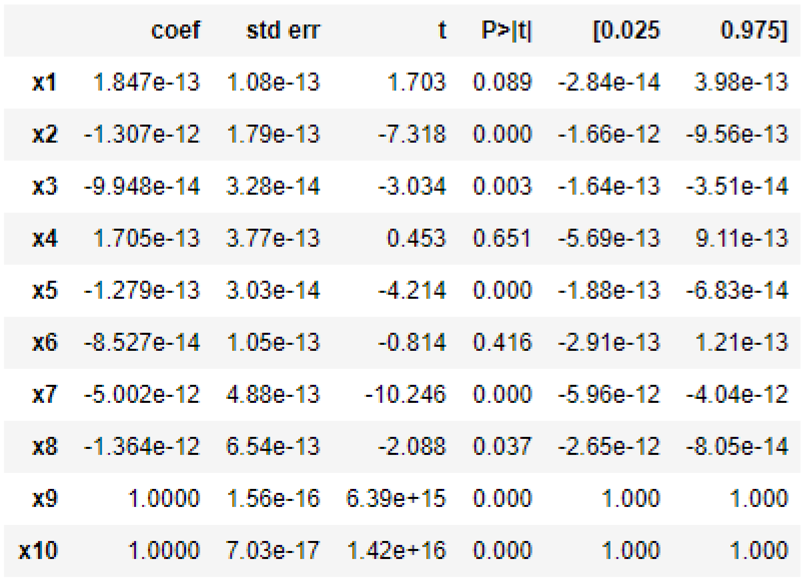
```python
#summary of Linear Regression  model
#LR_model.summary()
#Using of summary function we get summary of Linear Regression model
LR_model
```

```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x2b6f6db8048>
```

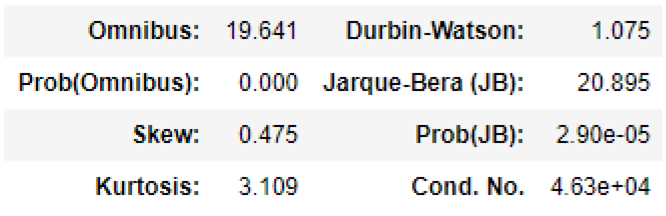
```python
#predicting Linear regression model using predict function
LR_predictions = LR_model.predict(test_feature)
```

OLS Regression Results

Dep. Variable:	y	**R-squared (uncentered):**	1.000
Model:	OLS	**Adj. R-squared (uncentered):**	1.000
Method:	Least Squares	**F-statistic:**	6.604e+32
Date:	Sat, 11 Apr 2020	**Prob (F-statistic):**	0.00
Time:	22:04:34	**Log-Likelihood:**	14223.
No. Observations:	549	**AIC:**	-2.843e+04
Df Residuals:	539	**BIC:**	-2.838e+04
Df Model:	10		
Covariance Type:	nonrobust		



	coef	std err	t	P>|t|	[0.025	0.975]
x1	1.847e-13	1.08e-13	1.703	0.089	-2.84e-14	3.98e-13
x2	-1.307e-12	1.79e-13	-7.318	0.000	-1.66e-12	-9.56e-13
x3	-9.948e-14	3.28e-14	-3.034	0.003	-1.64e-13	-3.51e-14
x4	1.705e-13	3.77e-13	0.453	0.651	-5.69e-13	9.11e-13
x5	-1.279e-13	3.03e-14	-4.214	0.000	-1.88e-13	-6.83e-14
x6	-8.527e-14	1.05e-13	-0.814	0.416	-2.91e-13	1.21e-13
x7	-5.002e-12	4.88e-13	-10.246	0.000	-5.96e-12	-4.04e-12
x8	-1.364e-12	6.54e-13	-2.088	0.037	-2.65e-12	-8.05e-14
x9	1.0000	1.56e-16	6.39e+15	0.000	1.000	1.000
x10	1.0000	7.03e-17	1.42e+16	0.000	1.000	1.000



Omnibus:	19.641	**Durbin-Watson:**	1.075
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	20.895
Skew:	0.475	**Prob(JB):**	2.90e-05
Kurtosis:	3.109	**Cond. No.**	4.63e+04

Here :

Residual standard error: 3.231e-12 on 576 degrees of freedom

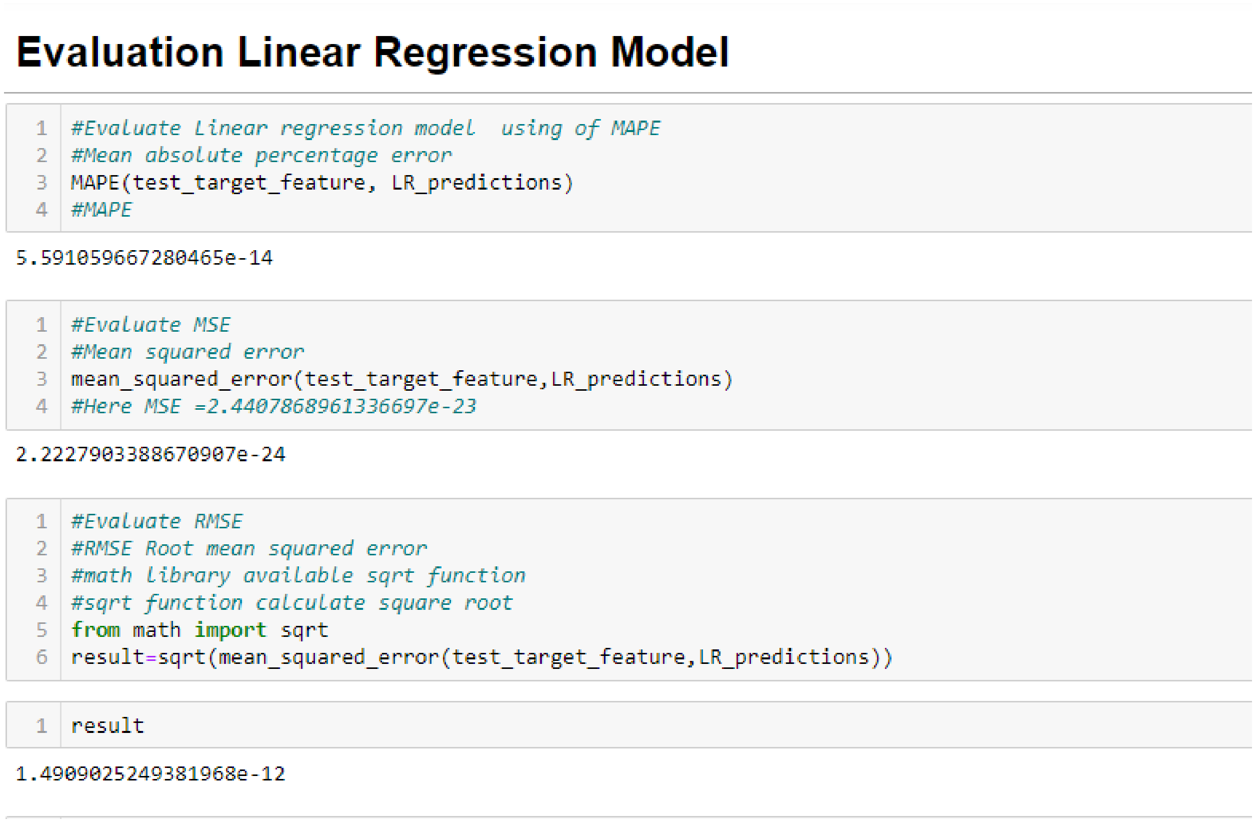
Multiple R-squared: 1, Adjusted R-squared: 1

Here residual Standard error is quite less so the distance between predicted values f(x[I,]) and actaual values f(x) are very less so this model is predicted almost accurate values.

And Multiple R-Square value is 1 so, we can explain about 100 % of the data using our multiple linear regression model. This is very impressive in Linear Regression Model.

3.4.1 Evaluation of Linear regression Model

Figure 3.4.2 Evaluation of Regression Model



Evaluation Linear Regression Model

```
#Evaluate Linear regression model  using of MAPE
#Mean absolute percentage error
MAPE(test_target_feature, LR_predictions)
#MAPE
```

```
5.591059667280465e-14
```

```
#Evaluate MSE
#Mean squared error
mean_squared_error(test_target_feature,LR_predictions)
#Here MSE =2.4407868961336697e-23
```

```
2.2227903388670907e-24
```

```
#Evaluate RMSE
#RMSE Root mean squared error
#math library available sqrt function
#sqrt function calculate square root
from math import sqrt
result=sqrt(mean_squared_error(test_target_feature,LR_predictions))
```

```
result
```

```
1.4909025249381968e-12
```

From above figure it is clearly showing that Model Accuracy is 99.9 % and RMSE is nearly equal to 1.49.

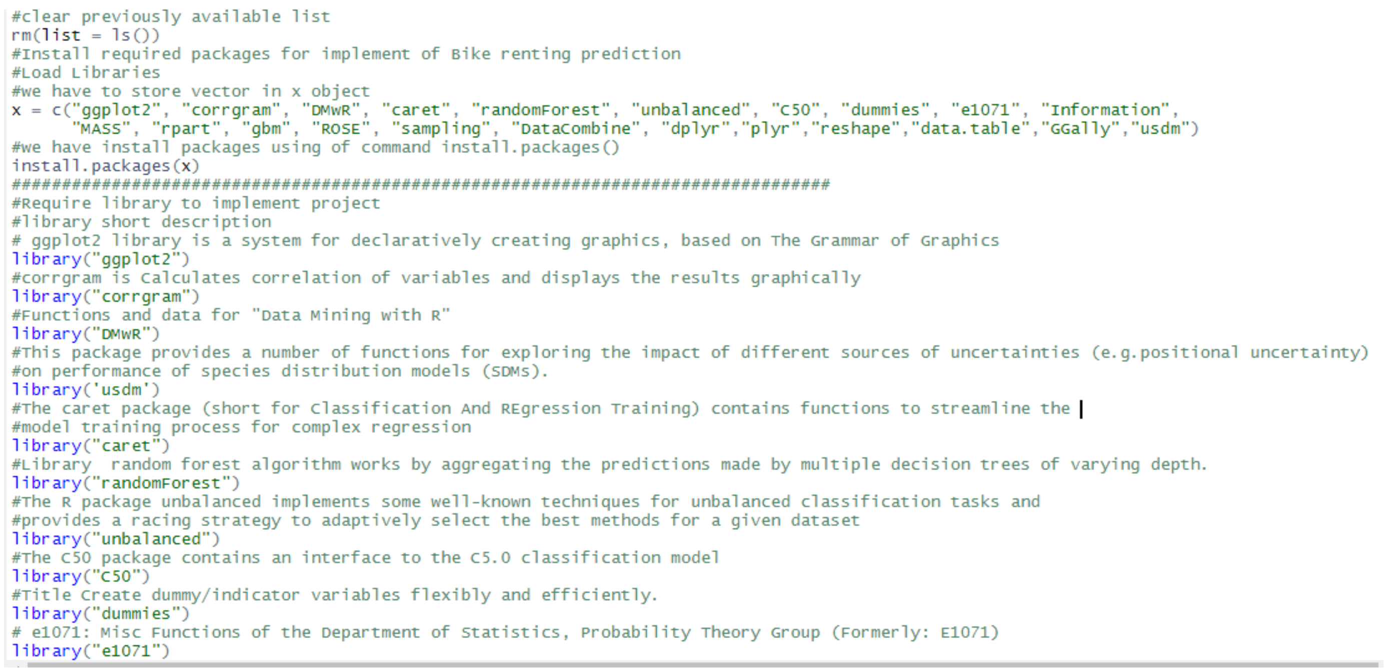
Model Selection

As we predicted counts for Bike Rental using three Models Decision Tree, Random Forest and Linear Regression as MAPE is high and RMSE is less for the Linear regression Model

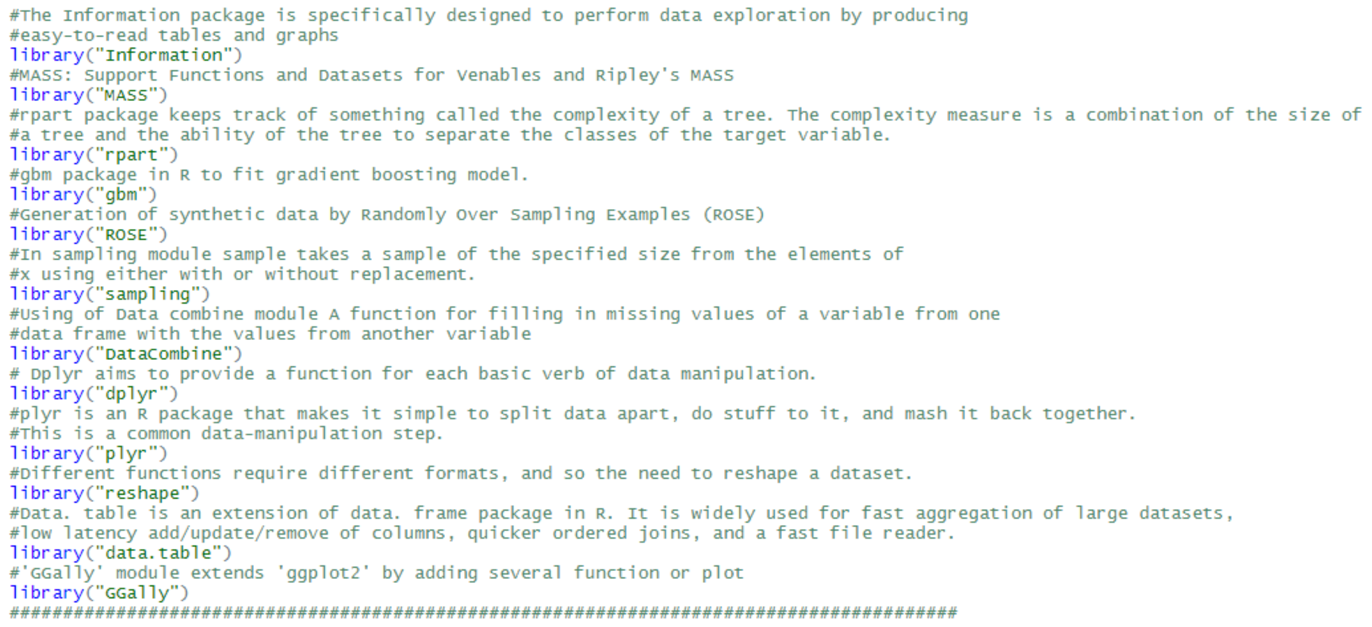
Conclusion: - For the Bike Rental Data Linear Regression Model is best model to predict the count rental Bike daily based on environmental or seasonal .

Appendix A -R Code

R code :



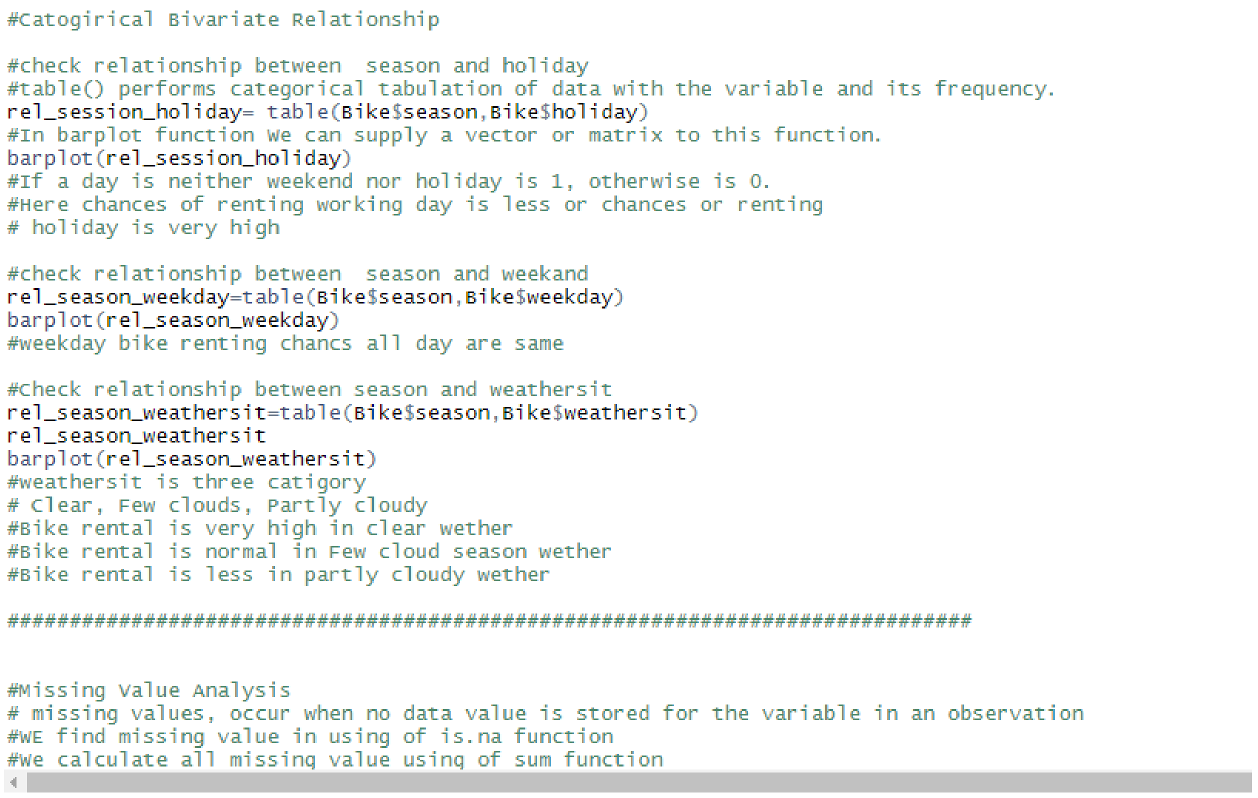
```r
#clear previously available list
rm(list = ls())
#Install required packages for implement of Bike renting prediction
#Load Libraries
#we have to store vector in x object
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071", "Information",
      "MASS", "rpart", "gbm", "ROSE", "sampling", "DataCombine", "dplyr","plyr","reshape","data.table","GGally","usdm")
#we have install packages using of command install.packages()
install.packages(x)
##################################################################################
#Require library to implement project
#library short description
# ggplot2 library is a system for declaratively creating graphics, based on The Grammar of Graphics
library("ggplot2")
#corrgram is Calculates correlation of variables and displays the results graphically
library("corrgram")
#Functions and data for "Data Mining with R"
library("DMwR")
#This package provides a number of functions for exploring the impact of different sources of uncertainties (e.g.positional uncertainty)
#on performance of species distribution models (SDMs).
library('usdm')
#The caret package (short for Classification And REgression Training) contains functions to streamline the
#model training process for complex regression
library("caret")
#Library  random forest algorithm works by aggregating the predictions made by multiple decision trees of varying depth.
library("randomForest")
#The R package unbalanced implements some well-known techniques for unbalanced classification tasks and
#provides a racing strategy to adaptively select the best methods for a given dataset
library("unbalanced")
#The C50 package contains an interface to the C5.0 classification model
library("C50")
#Title Create dummy/indicator variables flexibly and efficiently.
library("dummies")
# e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071)
library("e1071")
```



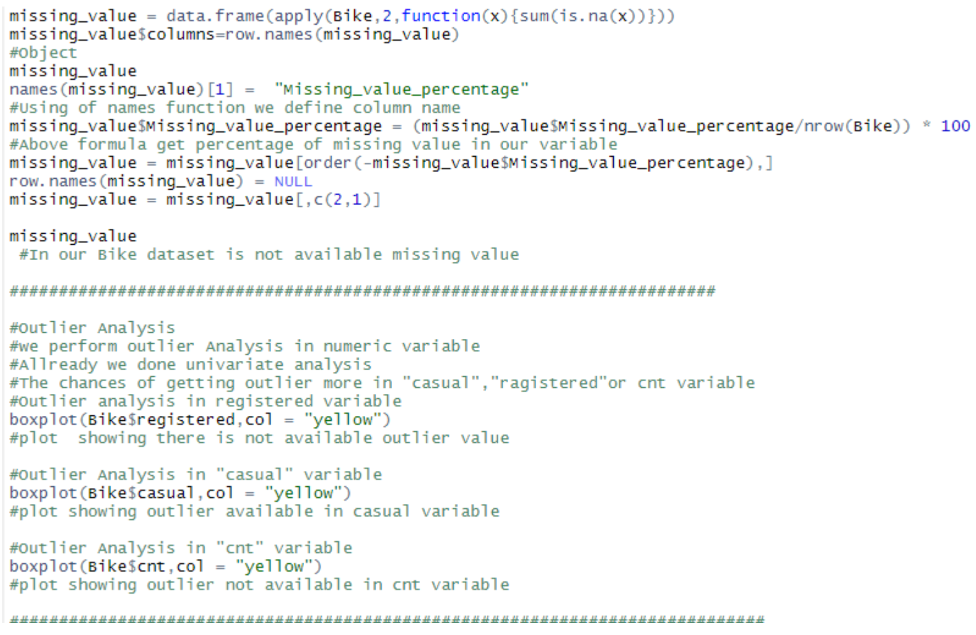
```r
#The Information package is specifically designed to perform data exploration by producing
#easy-to-read tables and graphs
library("Information")
#MASS: Support Functions and Datasets for Venables and Ripley's MASS
library("MASS")
#rpart package keeps track of something called the complexity of a tree. The complexity measure is a combination of the size of
#a tree and the ability of the tree to separate the classes of the target variable.
library("rpart")
#gbm package in R to fit gradient boosting model.
library("gbm")
#Generation of synthetic data by Randomly Over Sampling Examples (ROSE)
library("ROSE")
#In sampling module sample takes a sample of the specified size from the elements of
#x using either with or without replacement.
library("sampling")
#Using of Data combine module A function for filling in missing values of a variable from one
#data frame with the values from another variable
library("DataCombine")
# Dplyr aims to provide a function for each basic verb of data manipulation.
library("dplyr")
#plyr is an R package that makes it simple to split data apart, do stuff to it, and mash it back together.
#This is a common data-manipulation step.
library("plyr")
#Different functions require different formats, and so the need to reshape a dataset.
library("reshape")
#Data. table is an extension of data. frame package in R. It is widely used for fast aggregation of large datasets,
#low latency add/update/remove of columns, quicker ordered joins, and a fast file reader.
library("data.table")
#'GGally' module extends 'ggplot2' by adding several function or plot
library("GGally")
###########################################################################################
```

```
#get Working directory
#using of get function we check current working diroctory
getwd()
#Set working directory
#Using of set function we set our working diroctory
setwd("C:/Users/ASUS/Desktop")

#Load Day dataset
Bike=read.csv("Day.csv",header =TRUE)
#Using of head function we get  five observation in our  Bike dataset
head(Bike)
#In Bike dataset available 16 variable
#In this variable 15 are independant variable or cnt is target variable
#Using of tail fucntion we get last five observation in our Bike dataset
tail(Bike)
#Measures of centrality are mean, median. Measures of spread are
#standard deviation, quantiles, min and max, range, interquartile range.
#Using of summary function we get  description of Bike object
summary(Bike)
#str() It provides great information about the structure of some object.
#str() function is provide great information of Object
str(Bike)
##################################################################################
#Univariate Analysis#

#Univariate analysis is the simplest form of analyzing data. “Uni” means “one”,
#so in other words your data has only one variable. It doesn't deal with causes
#or relationships (unlike regression) and it's major purpose is to describe;
#it takes data, summarizes that data and finds patterns in the data

#glimpse function is helpful for a first inspection of the data set at hand.
glimpse(Bike)




#Univariate analysis in Numeric variable

#Analyze the distribution of  target variable "cnt"
hist(Bike$cnt,right = FALSE,col = "blue")
#Above histogram graphical cnt variable show normal distribution

#Analyze the distribution of independant variable "casual"
hist(Bike$casual,right = F,col = "blue")
#Aboove histogram graph showing casual variable is not normaly distributed and
#chances of outlier is high in this variable

#Analyze the distribution of independant variable "temp"
hist(Bike$temp,right = F,col = "blue")
#Aboove histogram graph showing temp variable is normaly distributed

#Analyze the distribution of independant variable "registered"
hist(Bike$registered,right = F,col = "blue")
#Aboove histogram graph showing ragistered variable is normaly distributed

#Analyze the distribution of independant variable "hum"
hist(Bike$hum,right = F,col = "blue")
#Aboove histogram graph showing hum variable is normaly distributed

###################################################################################
#Categorical variable Univariaite analysis

# Visualize categorical Variable "mnth" with target variable "cnt"
#Change x varable factor catogroy using of as.factor function
#using of fill function we fill color in our plot
ggplot(Bike, aes(x=as.factor(mnth), y=cnt),fill="blue") +
  stat_summary(fun="mean", geom="bar",color="red")
#Using of aes() aesthatic function we fill color visualize plot just like (boxplot,histogram)
#These visual caracteristics are known as aesthetics (or aes) and include:
#color and fill. points shape.
```

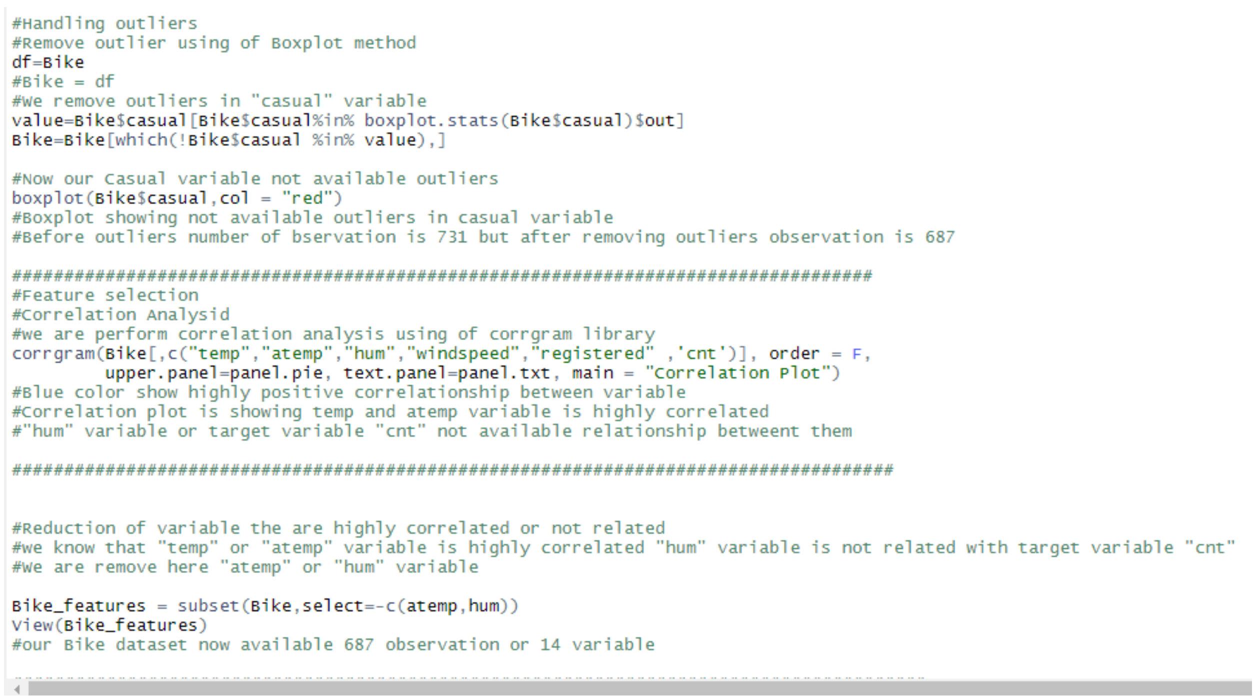
```r
ggplot(Bike)+
  geom_histogram(aes(x=cnt,y=..density..),
                 fill= "red")+
  geom_density(aes(x=cnt,y=..density..))

# Visualize categorical Variable 'holiday'

ggplot(Bike) +
  geom_bar(aes(x=holiday),fill="red")
#Number of count is very high in holiday time renting bike
# it is showing that  the mostly cycle rentals are happening  on holidays

# Visualize categorical Variable 'weekday'

ggplot(Bike) +
  geom_bar(aes(x=weekday),fill="red")
#Above plot is showing number of counts is same in all weekdays

# Visualize categorical Variable 'weathersit'
ggplot(Bike) +
  geom_bar(aes(x=weathersit),fill="blue")
#Number of count is very high in whether is "clear"
#Number of count is average if whether "Few clouds"
#Number of count is less if whwther is  "Partly cloudy, Partly cloudy"

##################################################################################
#Bivariate Analysis

#Bivariate relationship in numeric variable
#Firt we check Bivariate relationship "temp" and "hum" variable
#Using of geom_pointThe point geom is used to create scatterplots.
#geom_smooth understands the following aesthetics (required aesthetics are in bold):
#x.v.alpha.colour.fill.linetvpe.size.weight

ggplot(Bike, aes(x= temp,y=hum)) +
  geom_point()+
  geom_smooth()
#It is showing  graph  humadity increasing rapidly till 0.65 after after the
#have decressing gredualy


#we check Bivariate relationship "temp" and "atemp" variable
ggplot(Bike, aes(x= temp,y=atemp)) +
  geom_point()+
  geom_smooth()
#It is showing graph "temp" or "atemp" varaible is strongly positive bivariate relationship
#temp or atemp variable is highly positive correlated

#we check Bivariate relationship "temp" and "windspeed" variable
ggplot(Bike, aes(x= temp,y=windspeed)) +
  geom_point()+
  geom_smooth()
#It is showing graph "temp" or "windspeed" varaible is less negative correlation
#tempraure is increases thus time windspeed is decrease

############################################################################

#Checking relationship between all numeric variable
#We check  thus numeric relationship using of pair plot
ggpairs(Bike[,c('atemp','temp','hum','windspeed','cnt')])
#Above plot is showing positive correlation atemp or target variable cnt
#plot is showing positive correlation temp or target variable cnt
#plot is showing negative correlation hum or target variable cnt
#plot is showing positive correlation windspeed or target variable cnt

###############################################################################
```



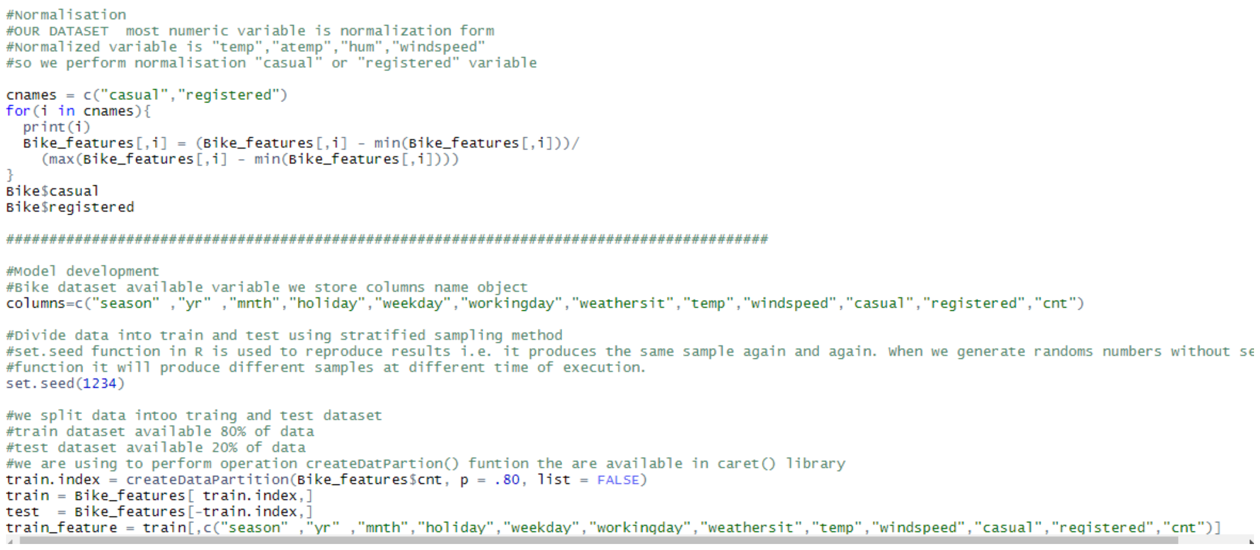
```
#Catogirical Bivariate Relationship

#check relationship between  season and holiday
#table() performs categorical tabulation of data with the variable and its frequency.
rel_session_holiday= table(Bike$season,Bike$holiday)
#In barplot function we can supply a vector or matrix to this function.
barplot(rel_session_holiday)
#If a day is neither weekend nor holiday is 1, otherwise is 0.
#Here chances of renting working day is less or chances or renting
# holiday is very high

#check relationship between  season and weekand
rel_season_weekday=table(Bike$season,Bike$weekday)
barplot(rel_season_weekday)
#weekday bike renting chancs all day are same

#Check relationship between season and weathersit
rel_season_weathersit=table(Bike$season,Bike$weathersit)
rel_season_weathersit
barplot(rel_season_weathersit)
#weathersit is three catigory
# Clear, Few clouds, Partly cloudy
#Bike rental is very high in clear wether
#Bike rental is normal in Few cloud season wether
#Bike rental is less in partly cloudy wether

##############################################################################


#Missing Value Analysis
# missing values, occur when no data value is stored for the variable in an observation
#WE find missing value in using of is.na function
#We calculate all missing value using of sum function
```



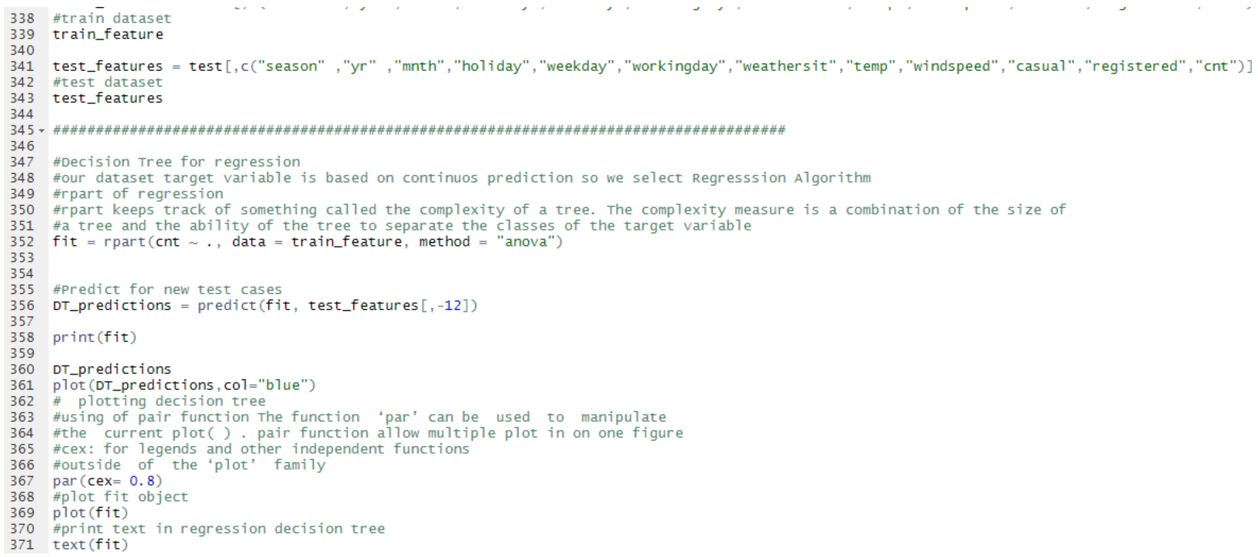
```
missing_value = data.frame(apply(Bike,2,function(x){sum(is.na(x))}))
missing_value$columns=row.names(missing_value)
#Object
missing_value
names(missing_value)[1] =  "Missing_value_percentage"
#Using of names function we define column name
missing_value$Missing_value_percentage = (missing_value$Missing_value_percentage/nrow(Bike)) * 100
#Above formula get percentage of missing value in our variable
missing_value = missing_value[order(-missing_value$Missing_value_percentage),]
row.names(missing_value) = NULL
missing_value = missing_value[,c(2,1)]

missing_value
 #In our Bike dataset is not available missing value

########################################################################

#Outlier Analysis
#we perform outlier Analysis in numeric variable
#Allready we done univariate analysis
#The chances of getting outlier more in "casual","ragistered"or cnt variable
#Outlier analysis in registered variable
boxplot(Bike$registered,col = "yellow")
#plot  showing there is not available outlier value

#Outlier Analysis in "casual" variable
boxplot(Bike$casual,col = "yellow")
#plot showing outlier available in casual variable

#Outlier Analysis in "cnt" variable
boxplot(Bike$cnt,col = "yellow")
#plot showing outlier not available in cnt variable

############################################################################
```



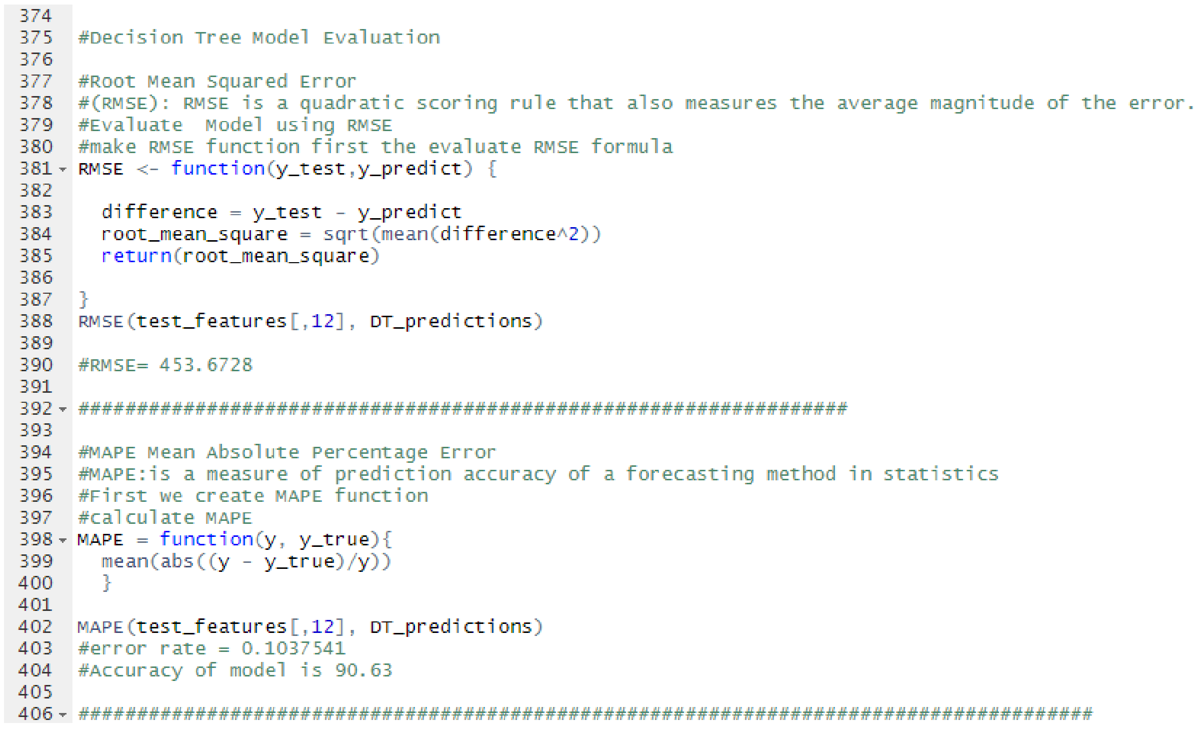
```
#Handling outliers
#Remove outlier using of Boxplot method
df=Bike
#Bike = df
#We remove outliers in "casual" variable
value=Bike$casual[Bike$casual%in% boxplot.stats(Bike$casual)$out]
Bike=Bike[which(!Bike$casual %in% value),]

#Now our Casual variable not available outliers
boxplot(Bike$casual,col = "red")
#Boxplot showing not available outliers in casual variable
#Before outliers number of bservation is 731 but after removing outliers observation is 687

###################################################################################
#Feature selection
#Correlation Analysid
#we are perform correlation analysis using of corrgram library
corrgram(Bike[,c("temp","atemp","hum","windspeed","registered" ,'cnt')], order = F,
         upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
#Blue color show highly positive correlationship between variable
#Correlation plot is showing temp and atemp variable is highly correlated
#"hum" variable or target variable "cnt" not available relationship betweent them

#####################################################################################


#Reduction of variable the are highly correlated or not related
#we know that "temp" or "atemp" variable is highly correlated "hum" variable is not related with target variable "cnt"
#We are remove here "atemp" or "hum" variable

Bike_features = subset(Bike,select=-c(atemp,hum))
View(Bike_features)
#our Bike dataset now available 687 observation or 14 variable
```



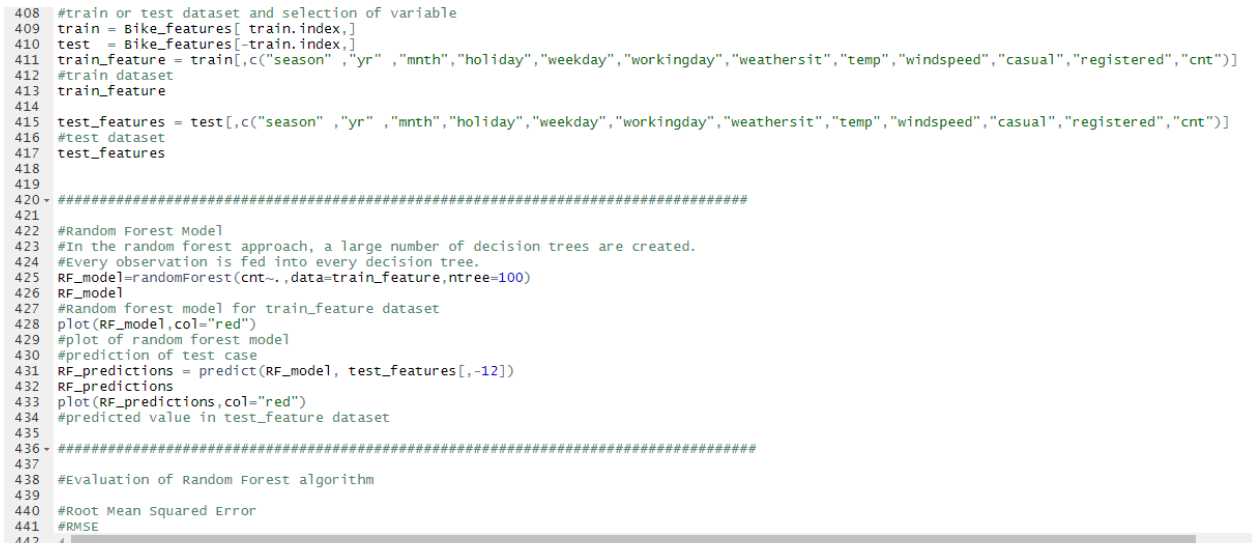
```
#Normalisation
#OUR DATASET  most numeric variable is normalization form
#Normalized variable is "temp","atemp","hum","windspeed"
#so we perform normalisation "casual" or "registered" variable

cnames = c("casual","registered")
for(i in cnames){
  print(i)
  Bike_features[,i] = (Bike_features[,i] - min(Bike_features[,i]))/
    (max(Bike_features[,i] - min(Bike_features[,i])))
}
Bike$casual
Bike$registered

#########################################################################################

#Model development
#Bike dataset available variable we store columns name object
columns=c("season" ,"yr" ,"mnth","holiday","weekday","workingday","weathersit","temp","windspeed","casual","registered","cnt")

#Divide data into train and test using stratified sampling method
#set.seed function in R is used to reproduce results i.e. it produces the same sample again and again. When we generate randoms numbers without se
#function it will produce different samples at different time of execution.
set.seed(1234)

#we split data intoo traing and test dataset
#train dataset available 80% of data
#test dataset available 20% of data
#we are using to perform operation createDatPartion() funtion the are available in caret() library
train.index = createDataPartition(Bike_features$cnt, p = .80, list = FALSE)
train = Bike_features[ train.index,]
test  = Bike_features[-train.index,]
train_feature = train[,c("season" ,"yr" ,"mnth","holiday","weekday","workingday","weathersit","temp","windspeed","casual","registered","cnt")]
```



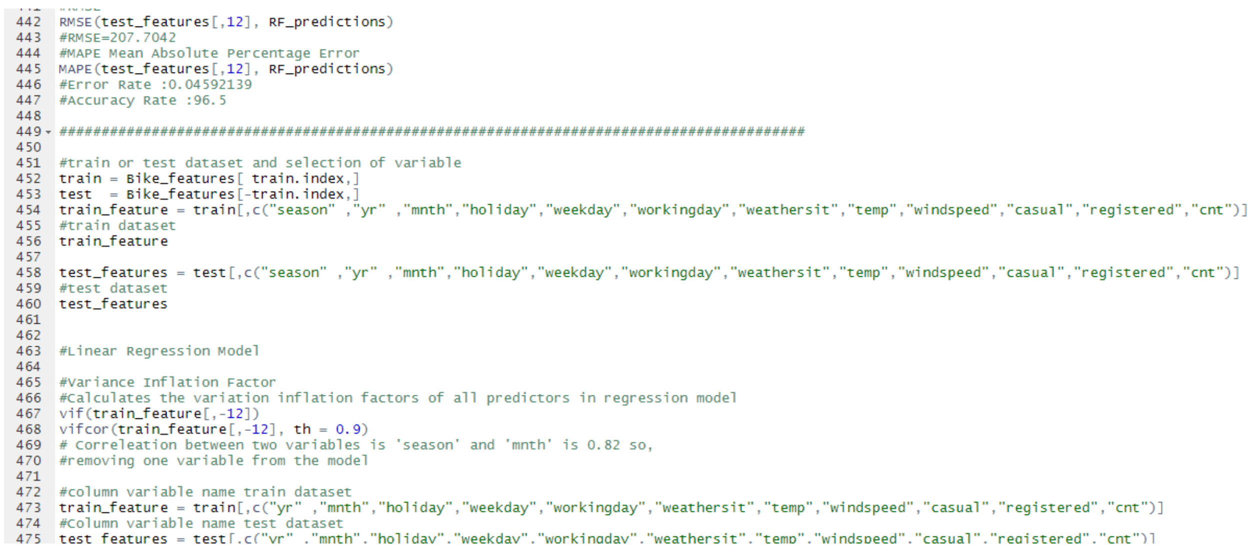
```r
#train dataset
train_feature

test_features = test[,c("season" ,"yr" ,"mnth","holiday","weekday","workingday","weathersit","temp","windspeed","casual","registered","cnt")]
#test dataset
test_features

##################################################################################

#Decision Tree for regression
#our dataset target variable is based on continuos prediction so we select Regresssion Algorithm
#rpart of regression
#rpart keeps track of something called the complexity of a tree. The complexity measure is a combination of the size of
#a tree and the ability of the tree to separate the classes of the target variable
fit = rpart(cnt ~ ., data = train_feature, method = "anova")


#Predict for new test cases
DT_predictions = predict(fit, test_features[,-12])

print(fit)

DT_predictions
plot(DT_predictions,col="blue")
#  plotting decision tree
#using of pair function The function  'par' can be  used  to  manipulate
#the  current plot( ) . pair function allow multiple plot in on one figure
#cex: for legends and other independent functions
#outside  of  the 'plot'  family
par(cex= 0.8)
#plot fit object
plot(fit)
#print text in regression decision tree
text(fit)
```



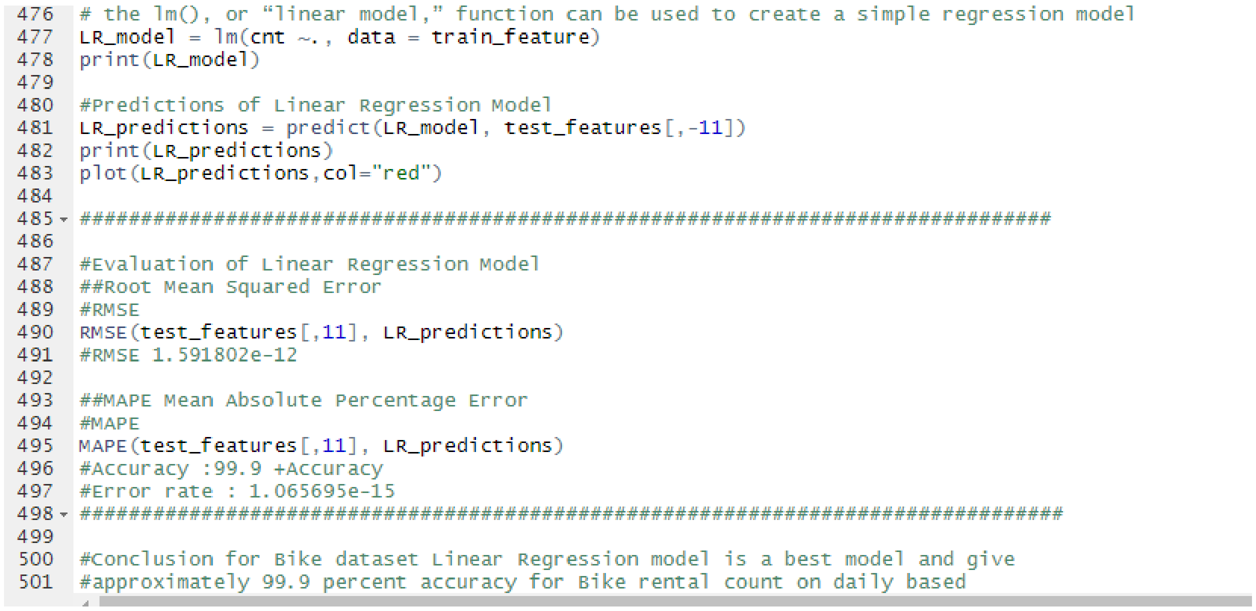
```r

#Decision Tree Model Evaluation

#Root Mean Squared Error
#(RMSE): RMSE is a quadratic scoring rule that also measures the average magnitude of the error.
#Evaluate  Model using RMSE
#make RMSE function first the evaluate RMSE formula
RMSE <- function(y_test,y_predict) {

  difference = y_test - y_predict
  root_mean_square = sqrt(mean(difference^2))
  return(root_mean_square)

}
RMSE(test_features[,12], DT_predictions)

#RMSE= 453.6728

#####################################################################

#MAPE Mean Absolute Percentage Error
#MAPE:is a measure of prediction accuracy of a forecasting method in statistics
#First we create MAPE function
#calculate MAPE
MAPE = function(y, y_true){
  mean(abs((y - y_true)/y))
  }

MAPE(test_features[,12], DT_predictions)
#error rate = 0.1037541
#Accuracy of model is 90.63

###########################################################################################
```



```r
#train or test dataset and selection of variable
train = Bike_features[ train.index,]
test  = Bike_features[-train.index,]
train_feature = train[,c("season" ,"yr" ,"mnth","holiday","weekday","workingday","weathersit","temp","windspeed","casual","registered","cnt")]
#train dataset
train_feature

test_features = test[,c("season" ,"yr" ,"mnth","holiday","weekday","workingday","weathersit","temp","windspeed","casual","registered","cnt")]
#test dataset
test_features


######################################################################################

#Random Forest Model
#In the random forest approach, a large number of decision trees are created.
#Every observation is fed into every decision tree.
RF_model=randomForest(cnt~.,data=train_feature,ntree=100)
RF_model
#Random forest model for train_feature dataset
plot(RF_model,col="red")
#plot of random forest model
#prediction of test case
RF_predictions = predict(RF_model, test_features[,-12])
RF_predictions
plot(RF_predictions,col="red")
#predicted value in test_feature dataset

######################################################################################

#Evaluation of Random Forest algorithm

#Root Mean Squared Error
#RMSE
```



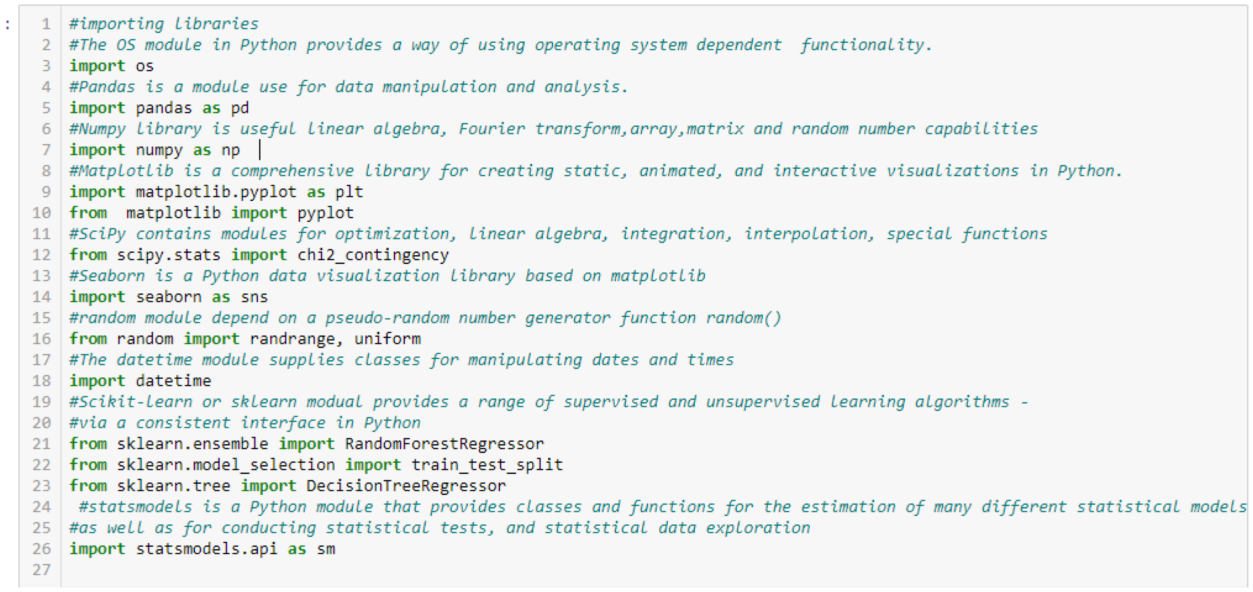
```r
RMSE(test_features[,12], RF_predictions)
#RMSE=207.7042
#MAPE Mean Absolute Percentage Error
MAPE(test_features[,12], RF_predictions)
#Error Rate :0.04592139
#Accuracy Rate :96.5

######################################################################################

#train or test dataset and selection of variable
train = Bike_features[ train.index,]
test  = Bike_features[-train.index,]
train_feature = train[,c("season" ,"yr" ,"mnth","holiday","weekday","workingday","weathersit","temp","windspeed","casual","registered","cnt")]
#train dataset
train_feature

test_features = test[,c("season" ,"yr" ,"mnth","holiday","weekday","workingday","weathersit","temp","windspeed","casual","registered","cnt")]
#test dataset
test_features


#Linear Regression Model

#Variance Inflation Factor
#Calculates the variation inflation factors of all predictors in regression model
vif(train_feature[,-12])
vifcor(train_feature[,-12], th = 0.9)
# Correleation between two variables is 'season' and 'mnth' is 0.82 so,
#removing one variable from the model

#column variable name train dataset
train_feature = train[,c("yr" ,"mnth","holiday","weekday","workingday","weathersit","temp","windspeed","casual","registered","cnt")]
#Column variable name test dataset
test_features = test[,c("yr" ,"mnth","holiday","weekday","workingday","weathersit","temp","windspeed","casual","registered","cnt")]
```



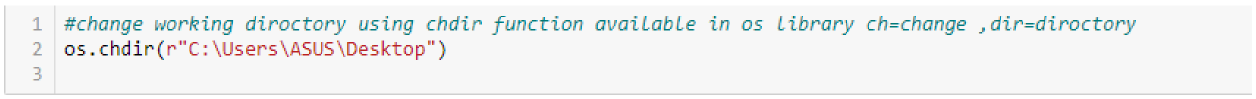
```r
# the lm(), or “linear model,” function can be used to create a simple regression model
LR_model = lm(cnt ~., data = train_feature)
print(LR_model)

#Predictions of Linear Regression Model
LR_predictions = predict(LR_model, test_features[,-11])
print(LR_predictions)
plot(LR_predictions,col="red")

##############################################################################

#Evaluation of Linear Regression Model
##Root Mean Squared Error
#RMSE
RMSE(test_features[,11], LR_predictions)
#RMSE 1.591802e-12

##MAPE Mean Absolute Percentage Error
#MAPE
MAPE(test_features[,11], LR_predictions)
#Accuracy :99.9 +Accuracy
#Error rate : 1.065695e-15
###############################################################################

#Conclusion for Bike dataset Linear Regression model is a best model and give
#approximately 99.9 percent accuracy for Bike rental count on daily based
```

Appendix B -Python Code

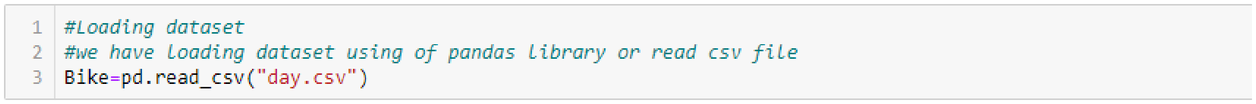
Python code:



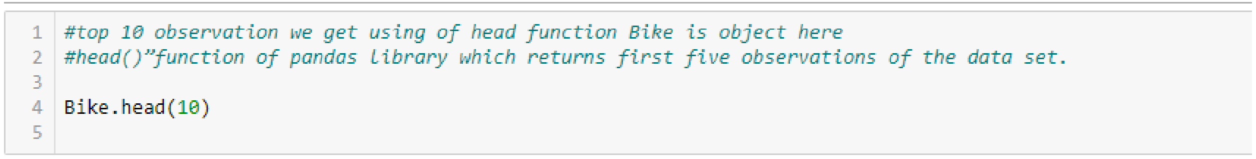
```python
#importing libraries
#The OS module in Python provides a way of using operating system dependent  functionality.
import os
#Pandas is a module use for data manipulation and analysis.
import pandas as pd
#Numpy library is useful linear algebra, Fourier transform,array,matrix and random number capabilities
import numpy as np
#Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
import matplotlib.pyplot as plt
from  matplotlib import pyplot
#SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions
from scipy.stats import chi2_contingency
#Seaborn is a Python data visualization library based on matplotlib
import seaborn as sns
#random module depend on a pseudo-random number generator function random()
from random import randrange, uniform
#The datetime module supplies classes for manipulating dates and times
import datetime
#Scikit-learn or sklearn modual provides a range of supervised and unsupervised learning algorithms -
#via a consistent interface in Python
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
 #statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models
#as well as for conducting statistical tests, and statistical data exploration
import statsmodels.api as sm

```



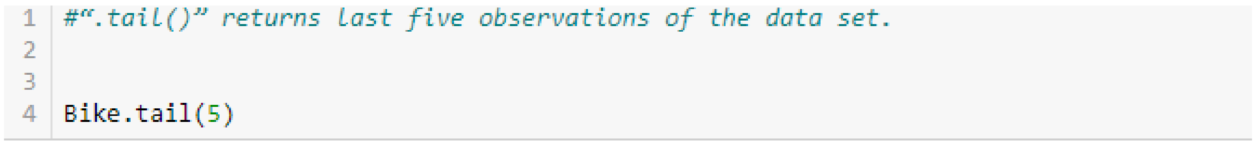
```python
#change working dirctory using chdir function available in os library ch=change ,dir=dirctory
os.chdir(r"C:\Users\ASUS\Desktop")

```



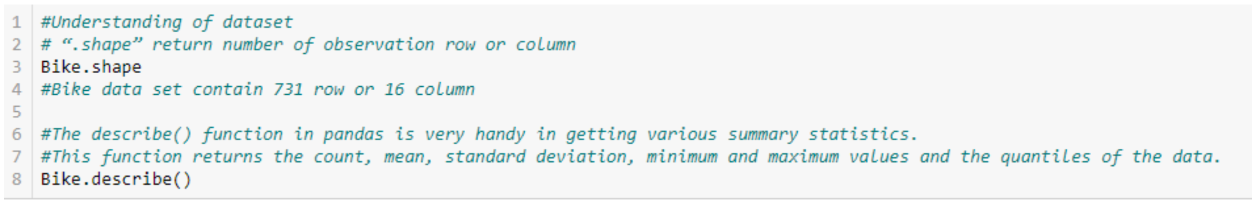
```python
#Loading dataset
#we have loading dataset using of pandas library or read csv file
Bike=pd.read_csv("day.csv")
```



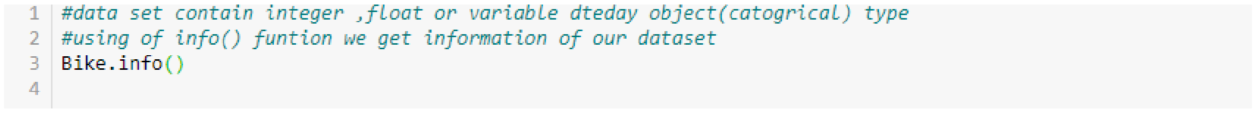
```python
#top 10 observation we get using of head function Bike is object here
#head()"function of pandas library which returns first five observations of the data set.

Bike.head(10)

```



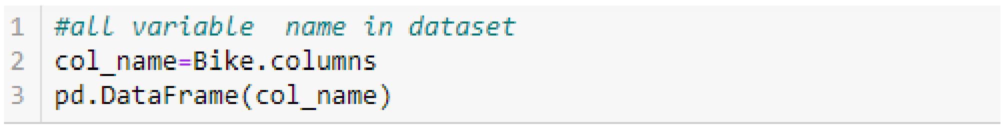
```python
#".tail()" returns last five observations of the data set.


Bike.tail(5)
```



```python
#Understanding of dataset
# ".shape" return number of observation row or column
Bike.shape
#Bike data set contain 731 row or 16 column

#The describe() function in pandas is very handy in getting various summary statistics.
#This function returns the count, mean, standard deviation, minimum and maximum values and the quantiles of the data.
Bike.describe()
```



```python
#data set contain integer ,float or variable dteday object(catogrical) type
#using of info() funtion we get information of our dataset
Bike.info()

```



```python
#all variable  name in dataset
col_name=Bike.columns
pd.DataFrame(col_name)
```

Univariate Analysis

```
]:
#Univariate analysis is perhaps the simplest form of statistical analysis.
#The key fact is that only one variable is involved in univariate analysis.
```

```
]:

#Target variable  analysis
#descriptive statistics summary
#describe() function in pandas is very handy in getting various summary statistics
#cnt variable is our target variable
Bike['cnt'].describe()

```

```
#Check whether target variable is normal or not
#most value cnt variable is 0.00005 to 0.00015
#Flexibly plot a univariate distribution of observations.
sns.distplot(Bike['cnt'],color="m")
#Below plot is showing cnt variable is normaly distributed
```

```
#check atemp variable normal or not their available outlier value or not or range in numeric value using of seborn library
#distplot function plot to atemp  variable
sns.distplot(Bike['atemp'],color="m")


```

```
#check windspeed variable normal or not their available outlier value or not or range in numeric value using of seborn libra
#distplot function plot to windspeed  variable

sns.distplot(Bike['windspeed'],color="m")
```

```
#skewnwss is Assymetry distributation
#Here skewness  value is -0.047353 their is negative skewness
#Here Kurtosis value is -0.811922 if value is less than 0 than their is playtocartic
#Here kurtosis is very less so target variable is playticartic distribitation
print("Skewness: %f" % Bike['cnt'].skew())
print("Kurtosis: %f" % Bike['cnt'].kurt())
```

Bivariate Relationship

```
#Bivariate analysis is one of the simplest forms of quantitative analysis.
#Bivariate analysis involve two variable.
# Bivariate analysis  helpful in testing simple hypotheses of association.
#Let's we check now numeric variable temp or target variable cnt Bivariate relationship
#Using of value_counts() function returns object containing counts of unique values.
Bike['temp'].value_counts()
#Now we drow scatter plot in betweent temp or cnt variable
variable = 'temp'
#using of concati function in pandas library
#Concatenate pandas objects along a particular axis with optional set logic along the other axes
Bi_ralationship= pd.concat([Bike['cnt'], Bike[variable]], axis=1)
#Using of matplotlib library scatter(x,y) creates a scatter plot with circles at the locations specified by
#the vectors x and y
Bi_ralationship.plot.scatter(x=variable, y='cnt', ylim=(0,10000),color="red")
#scatter plot is showing positive relationship between temp or cnt variable


```

```python
#relation between Numerical Variable 'hum' and target variable 'cnt'
Bike['hum'].value_counts()
#Now we draw scatter plot between 'hum' and 'cnt' variables
variable = 'hum'
#Concatenate pandas objects along a particular axis with optional set logic along the other axes
Bi_ralationship= pd.concat([Bike['cnt'], Bike[variable]], axis=1)
#Using of matplotlib library scatter(x,y) creates a scatter plot with circles at the locations specified by
#the vectors x and y
Bi_ralationship.plot.scatter(x=variable, y='cnt', ylim=(0,10000),color="red")
#scatter plot is showing average relationship between hum variable or target variable cnt


```

```python
#relation between Numerical Variable 'windspeed' and target variable 'cnt'
Bike['windspeed'].value_counts()
#Now we draw scatter plot between 'atemp' and 'cnt' variables
variable = 'windspeed'
#Concatenate pandas objects along a particular axis with optional set logic along the other axes
Bi_ralationship= pd.concat([Bike['cnt'], Bike[variable]], axis=1)
#Using of matplotlib library scatter(x,y) creates a scatter plot with circles at the locations specified by the vectors x an
Bi_ralationship.plot.scatter(x=variable, y='cnt', ylim=(0,10000))
#scatter plot is showing negative relationship between windspeed and target variable cnt
```

```python
#Bivariate relationship  weekday is catogaries  data or target variable cnt
#box plot to show distributions with respect to categories
#box plot 'Weekdays' with 'cnt'
variable_weekdays = 'weekday'
#Concatenate pandas objects along a particular axis with optional set logic along the other axes
data = pd.concat([Bike['cnt'], Bike[variable_weekdays]], axis=1)
#subplot(): The function returns a figure object and a tuple containing axes objects equal to nrows*ncols
f, ax = plt.subplots(figsize=(12, 10))
#seaborn. boxplot. Draw a box plot to show distributions with respect to categories
fig = sns.boxplot(x=variable_weekdays, y="cnt", data=data)
fig.axis(ymin=0, ymax=10000)
#Boxplot shows most median value between 4000-5000

```

```python
#Bivariate relationship  holiday is catogaries  data or target variable cnt
#box plot to show distributions with respect to categories
#box plot 'holiday' with 'CNT'
variable_holiday = 'holiday'
#Concatenate pandas objects along a particular axis with optional set logic along the other axes
data = pd.concat([Bike['cnt'], Bike[variable_holiday]], axis=1)
#subplot(): The function returns a figure object and a tuple containing axes objects equal to nrows*ncols
f, ax = plt.subplots(figsize=(12, 10))
#seaborn. boxplot. Draw a box plot to show distributions with respect to categories
fig = sns.boxplot(x=variable_holiday, y="cnt", data=data)
fig.axis(ymin=0, ymax=10000)
# If day is neither weekend nor holiday is 1
#otherwise 0
#below Boxplot is showing that median  high on holidays when compare to weekdays
```

Missing value Analysis

```python
#missing values, occur when no data value is stored for the variable in an observation. Missing data are a common occurrence
#and can have a significant effect on the conclusions that can be drawn from the data.
#The isnull() function is used to detect missing values for an array-like object
#The sort() method sorts the elements of a given list in a specific order - Ascending or Descending.
#here we short descending order
total = Bike.isnull().sum().sort_values(ascending=False)
percent = (Bike.isnull().sum()/Bike.isnull().count()).sort_values(ascending=False)
#Total,percent column name
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(100)
#In Bike dataset is not available missing value
```

Outlier Analysis

```
# An outlier is an element of a data set that distinctly stands out from the rest of the data.
#In other words, outliers are those data points that lie outside
#below plot is show variable is available outlier
sns.boxplot(Bike['casual'],color="yellow")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2b6f5730e88>
```

```
#in registered numeric variable is not available outlier value
sns.boxplot(Bike['registered'],color="yellow")
```

```
cnames=['casual']
```

```

cnames
```

```
['casual']
```

```
#Detact and delete outlier using of for loop and Iqr
for i in cnames:
    #print variable name
    print(i)
    # percentile() function used to compute the nth precentile of the given data (array elements)

    q75,q25=np.percentile(Bike.loc[:,i],[75,25])
    #IQR can be used to identify outliers in a data set.
    # Formula of IQR=Q3-Q1
    iqr=q75-q25
    min=q25-(iqr*1.5)
    max=q75+(iqr*1.5)
    #print minimum value in variable
    print(min)
    #print maximum value in variable
    print(max)
    #using of drop function we have drope outlier value
    Bike=Bike.drop(Bike[Bike.loc[:,i]<min].index)
    Bike=Bike.drop(Bike[Bike.loc[:,i]>max].index)
    #Below show minimum value in casual variable is -855.25
    #Below show maximum value in casual variable is 2266.75
```

```
#After remove outlier in casual variable
#plot show their is not available now outliers
sns.boxplot(Bike['casual'],color="yellow")
```

Feature selection

```
#Feature selection or variable subset selection,is the process of selecting a subset of relevant
#features for use in model construction.
#correlation plot
Bike_corr=Bike
```

```
Bike_corr.shape#total observation is 687 and 16 column
```

```
(687, 16)
```

```
#set the height and width of the plot
#subplots is function set height and width
#figsize height =15 or width =10
f,ax=plt.subplots(figsize=(15,10))
#Generate correlation matrix using of corr funtion
corr=Bike_corr.corr()
#plot using seaborn library
#heatmap is a function available in seaborn library help to plot correlation visualization
#diverging_palette function to set color of correlation plot
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, ax=ax)
```

```
#correlation range is -1 to 1
#-1 means highly negitive correlated data plot show blue color
#1 meand highly positive correlated data plot show red color
#our dataset "atemp" or "temp" independant variable is highly positive correlated variable
#Independent variable "hum" or target variable "cnt" is no relationship
#so we drop two variable atemp or hum variable in our Bike dataset
#droping highly correlated variable
Bike_day = Bike.drop(['atemp','hum'], axis=1)


```

```
#Twp correlated varaible is drop now 14 column is available in Bike dataset
Bike_day.shape
```

```
(687, 14)
```

```
#Top observation after drop atemp or hum variable
Bike_day.head()
```

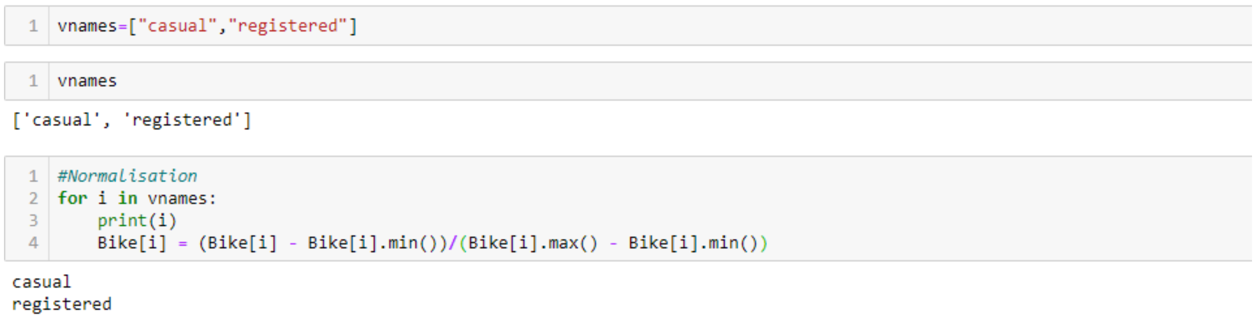
Feature Scaling

```
#Normalization
#Normalization is used to scale the data of an attribute so that it falls in a smaller range, such as -1.0 to 1.0 or 0.0 to
#In our dataset most numeric variable is normalize form so can't require to check outlier analysis thus variable
#Normaliz varaible name is temp,atemp,hum ,windspeed thus are
Bike_df =  Bike.copy()
```

```
#allready in our dataset four variable is normal form "temp","atemp","hum","windspeed"
#two variable casual or registered are not normalized form so we convert their value in normal form
Bike.shape
```

```
(687, 16)
```

```
#Normality check in casual variable
plt.hist(Bike["casual"],bins="auto")
```



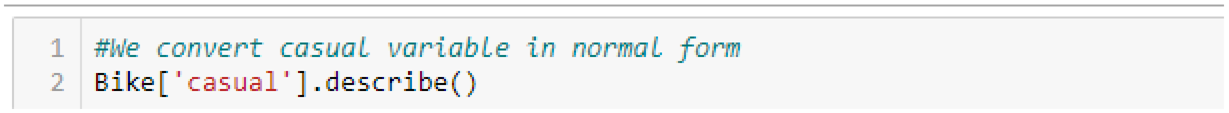
```python
vnames=["casual","registered"]
```

```python
vnames
```

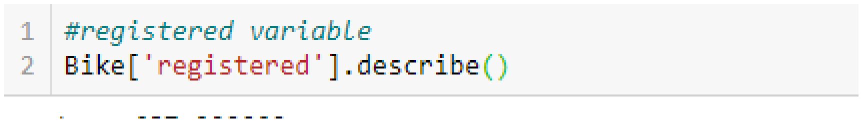
```
['casual', 'registered']
```

```python
#Normalisation
for i in vnames:
    print(i)
    Bike[i] = (Bike[i] - Bike[i].min())/(Bike[i].max() - Bike[i].min())
```

```
casual
registered
```

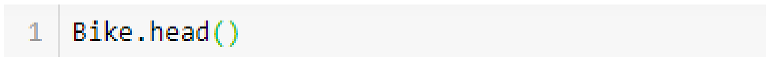


```python
#We convert casual variable in normal form
Bike['casual'].describe()
```

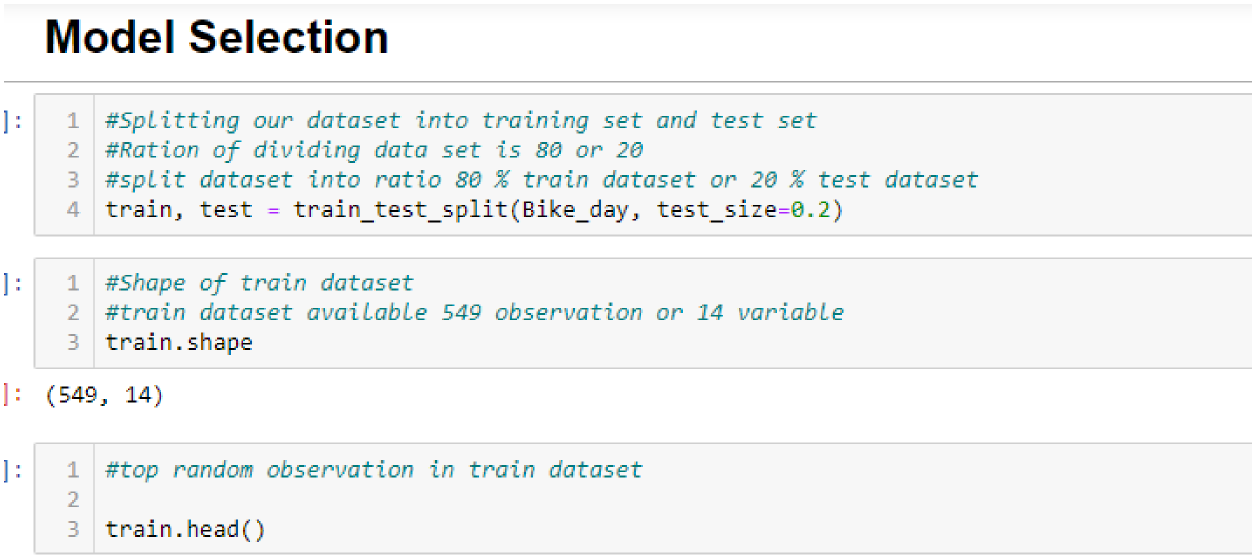


```python
#registered variable
Bike['registered'].describe()
```

```
.     --- ------
```



```python
Bike.head()
```



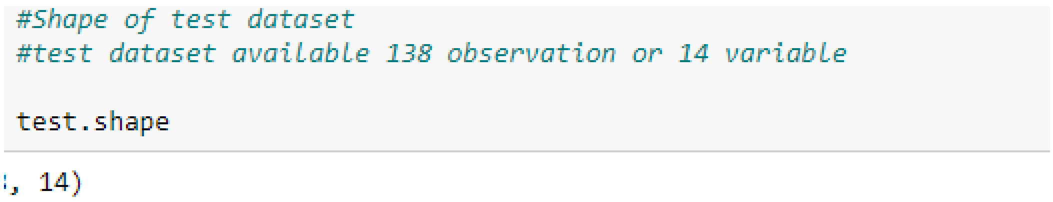
Model Selection

```python
#Splitting our dataset into training set and test set
#Ration of dividing data set is 80 or 20
#split dataset into ratio 80 % train dataset or 20 % test dataset
train, test = train_test_split(Bike_day, test_size=0.2)
```

```python
#Shape of train dataset
#train dataset available 549 observation or 14 variable
train.shape
```

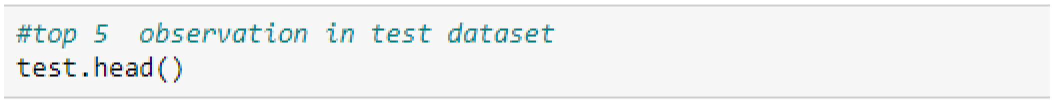
```
(549, 14)
```

```python
#top random observation in train dataset

train.head()
```



```python
#Shape of test dataset
#test dataset available 138 observation or 14 variable

test.shape
```

```
, 14)
```



```python
#top 5  observation in test dataset
test.head()
```

Decision Tree Regressor Model

```python
#Importing Decison Tree Regressor Module from sklearn
from sklearn.tree import DecisionTreeRegressor
#We take here feature or data set
#train data set to store  object train_features_one independant or target variable
#We have to store variable in single object train_features_one independant variable
train_features = train[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].val
#Target variable
train_target_feature = train['cnt'].values

#test data set to store  object test_feature independent  or target variable
#We have to store variable in single object test_feature independant variable
test_feature = test[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].values
#Target variable
test_target_feature= test['cnt'].values
```

```python
#Decsion Tree for regression
DT_fit = DecisionTreeRegressor()
#fit function Build a decision tree from the training set (X, y).
DT_fit = DT_fit.fit(train_features, train_target_feature)
```

```python

DT_fit
```

```python
#Apply model on test data
#predict function Predict class or regression target for X.
#predict is a function the predict unseen array in data set
DT_predictions = DT_fit.predict(test_feature)
```

```python
#This is prediction value
DT_predictions
```

Evaluation decison Tree Regressor model

```python
#Define MAPE function
#Calculate Mean absolute percentage error (MAPE)
#we have create MAPE function to calculate
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape

```

```python
#Evaluate Decission tree regressor  using of MAPE
#Mean absolute percentage error
MAPE(test_target_feature, DT_predictions)
#Here MAPE  is a measure of prediction accuracy of a forecasting method in statistics

```

```
17.15871570190131
```

```python
#Mean squared error
#we have import mean_squared_error to calculate MSE
from sklearn.metrics import mean_squared_error


```

```python

#Mean squared error
mean_squared_error(test_target_feature,DT_predictions)
#Here MSE that is, the average squared difference between the estimated values and the actual value.

```

```python
#RMSE Root mean squared error
#math library available sqrt function
#sqrt function calculate square root
from math import sqrt
result=sqrt(mean_squared_error(test_target_feature,DT_predictions))
#here RMSE is a frequently used measure of the differences between values predicted
#by a model or an estimator and the values observed.
```

```python
#Here RMSE
result
```

Random Forest

```python
#Implemantion of Random forest Regressor Model
#Importing Random Forest Regressor Module from sklearn.ensemble pakage
from sklearn.ensemble import RandomForestRegressor
#train data set to store  object train_features_one independant or target variable
#We have to store variable in single object train_features_one independant variable
train_features=train[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].value
#Target variable
train_target_feature = train['cnt'].values

#test data set to store  object test_feature independant  or target variable
#We have to store variable in single object test_feature independant variable
test_feature=test[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].values
#Target variable
test_target_feature= test['cnt'].values
#Creating a Random Forest Regressor model and fitting it to the training data
#A model comprised of many models is called an Ensemble model.
#For this model I've chosen 400 trees (n_estimator=400)
RF_model= RandomForestRegressor(n_estimators= 400, random_state=200).fit(train_features,train_target_feature)
```

```python
#RF_model object
RF_model
```

```python
#Prediction of unseen data using or predict function
#prediction array value is
RF_predictions
```

Evaluation Random Forest Model

```python
#Using of MAPE fucntion we calculate Mean absolute percenatage
#Evaluate Random forest using  MAPE
MAPE(test_target_feature, RF_predictions)
#Mean absolute percentage error
```

```
231.64270531827168
```

```python
#Mean squared error
#we have import mean_squared_error to calculate MSE
from sklearn.metrics import mean_squared_error
mean_squared_error(test_target_feature,RF_predictions)
#Here MSE

```

```
8611360.08128433
```

```python
#RMSE Root mean squared error
#math library available sqrt function
#sqrt function calculate square root
from math import sqrt
result=sqrt(mean_squared_error(test_target_feature,RF_predictions))
```

```python
#RMSE
result
#Here Root mean squared error RMSE is given below
```

Linear Regression Model

```python
#We take here feature or data set
#train data set to store  object train_features_one independant or target variable
#We have to store variable in single object train_features_one independant variable
train_features = train[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].val
#Target variable
train_target_feature = train['cnt'].values

#test data set to store  object test_feature independent  or target variable
#We have to store variable in single object test_feature independant variable
test_feature = test[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].values
#Target variable
test_target_feature= test['cnt'].values
```

```python
#statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models,
#as well as for conducting statistical tests, and statistical data exploration
import statsmodels.api as sm
#implement of Linear regression model
#OLS stands for Ordinary Least Squares
LR_model= sm.OLS(train_target_feature, train_features).fit()

```

```python
#summary of Linear Regression  model
LR_model.summary()
#Using of summary function we get summary of Linear Regression model
#LR_model
```

```python
#predicting Linear regression model using predict function
LR_predictions = LR_model.predict(test_feature)
```

```python
LR_predictions
```

Evaluation Linear Regression Model

```
]:  #Evaluate Linear regression model  using of MAPE
    #Mean absolute percentage error
    MAPE(test_target_feature, LR_predictions)
    #MAPE
```

```
]:  1.4168440321532624e-13
```

```
]:  #Evaluate MSE
    #Mean squared error
    mean_squared_error(test_target_feature,LR_predictions)
    #Here MSE =2.4407868961336697e-23
```

```
]:  9.18985272183296e-25
```

```
]:  #Evaluate RMSE
    #RMSE Root mean squared error
    #math library available sqrt function
    #sqrt function calculate square root
    from math import sqrt
    result=sqrt(mean_squared_error(test_target_feature,LR_predictions))
```

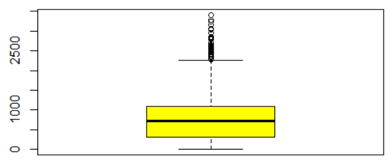
```
]:  result
```

```
]:  9.586371952846895e-13
```

```
]:  """Conclusion for Bike dataset Linear Regression model is a best model and give
       approximately 99.9 percent accuracy for Bike rental count on daily based """
```

Appendix C-Extra Figure

Figure show With outliers in casual variable :



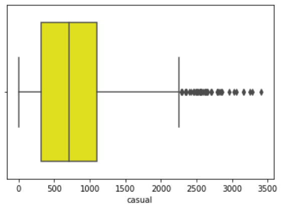
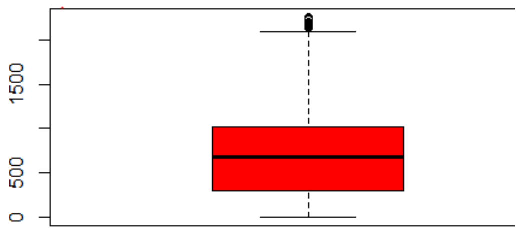


Figure show Without outliers in casual variable:



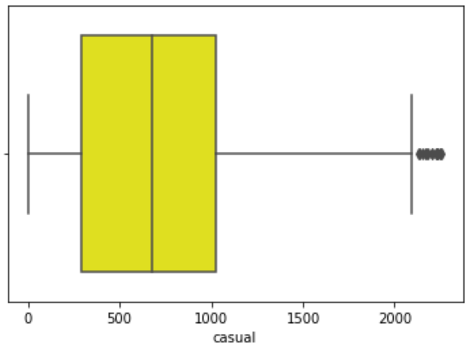


Figure show Relationship between Weekdays and cnt target variable:

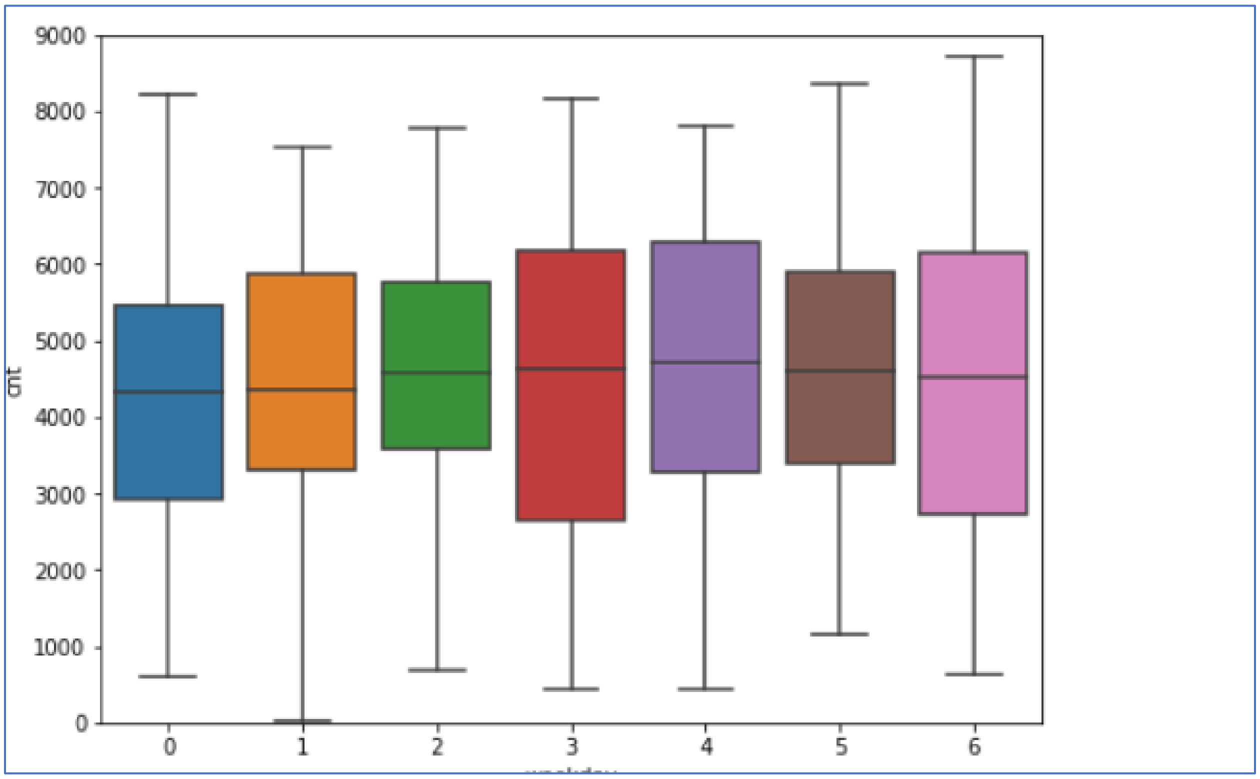


Figure show Numeric relationship pairs plot:

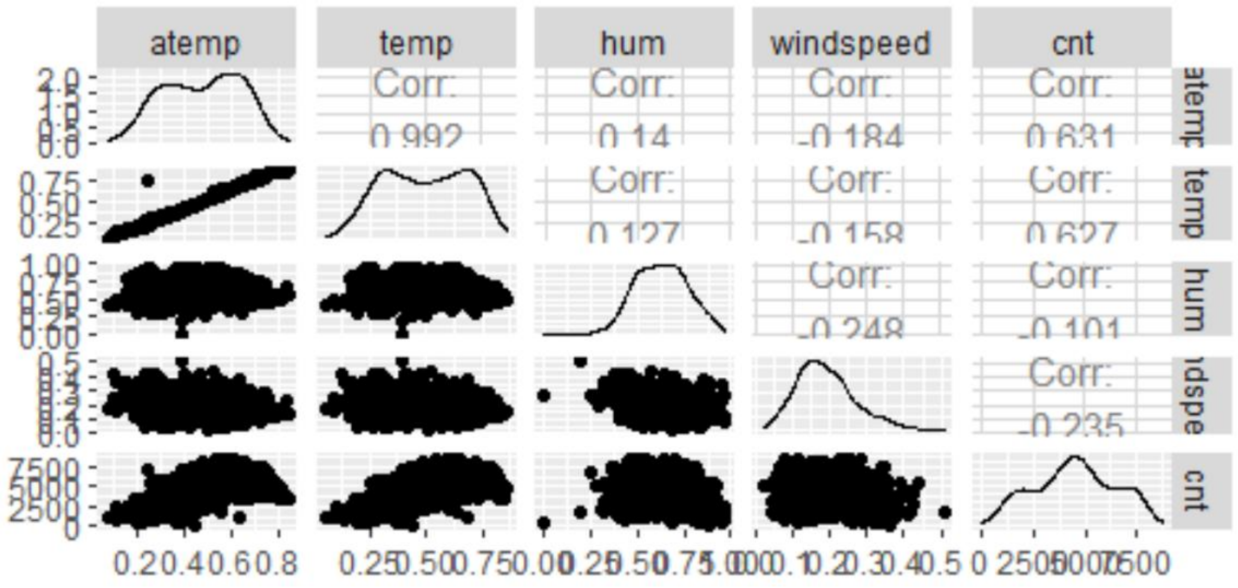


Figure show Correlation plot:

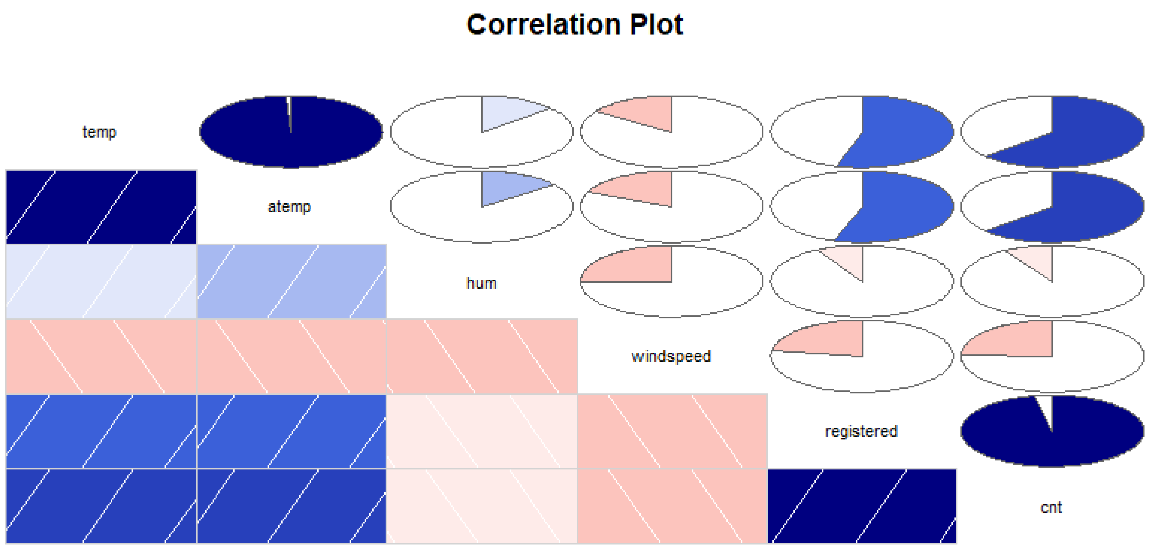


Figure shows Decision Tree predictors index:

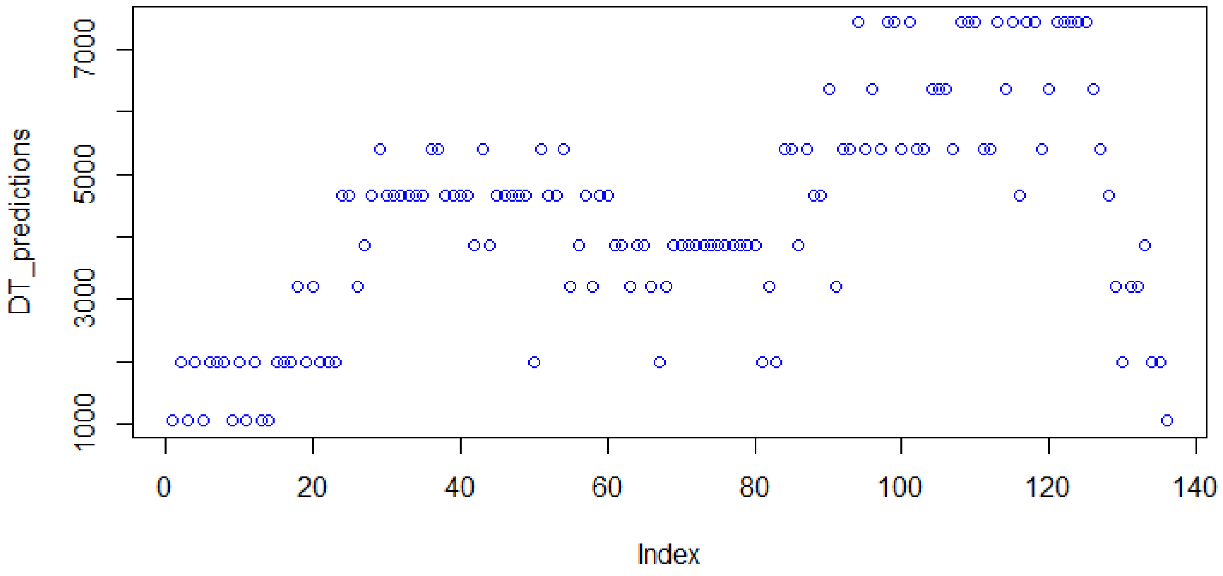


Figure shows Random Forest error plot:

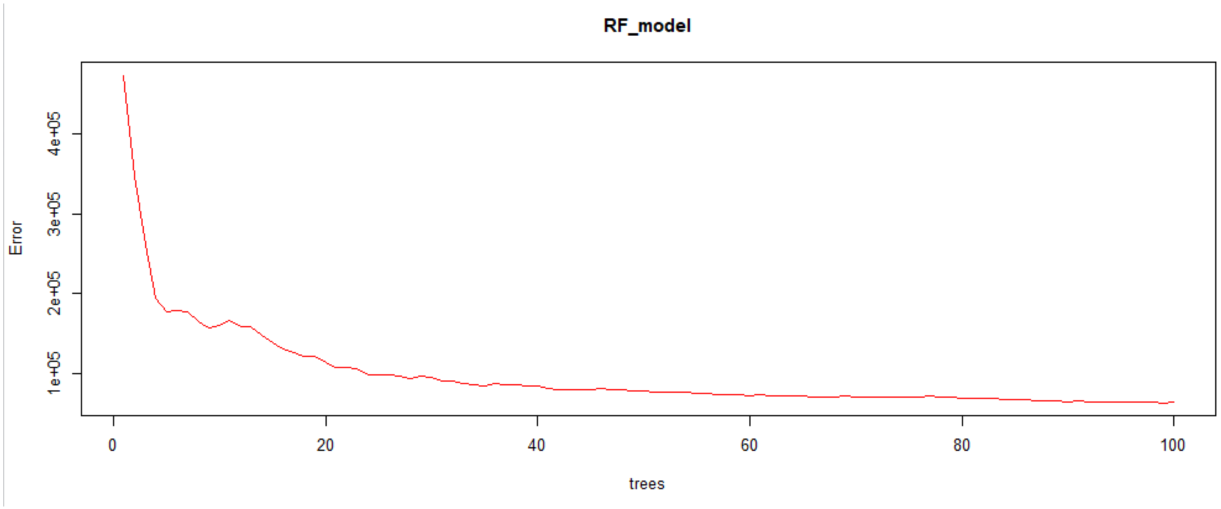


Figure shows Linear Regression model prediction plot:

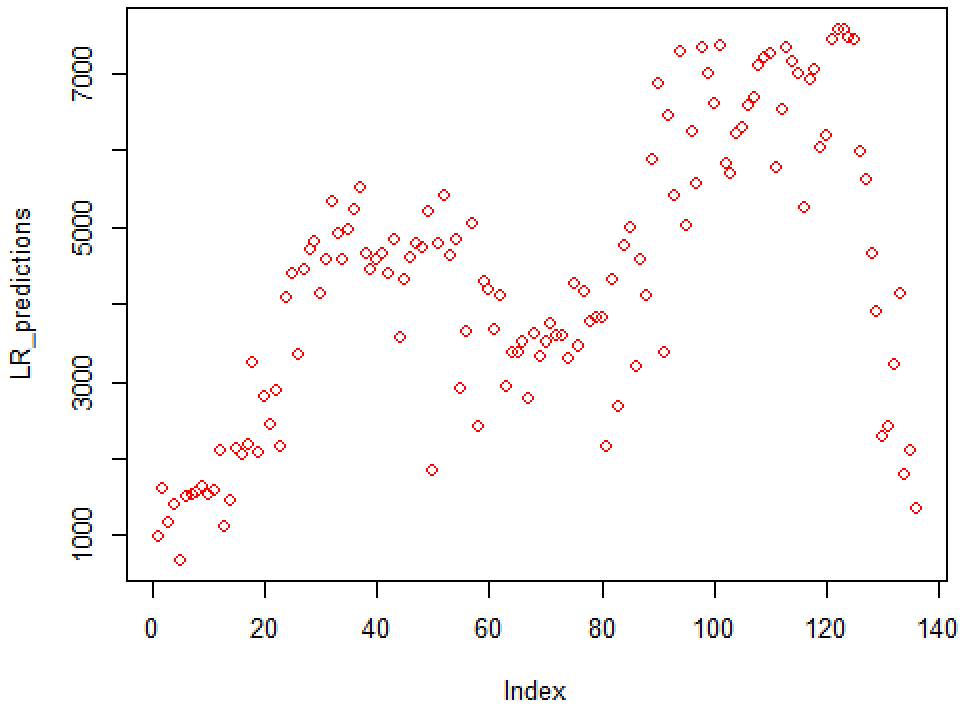


Figure shows Relationship between Holiday and Weathersit:

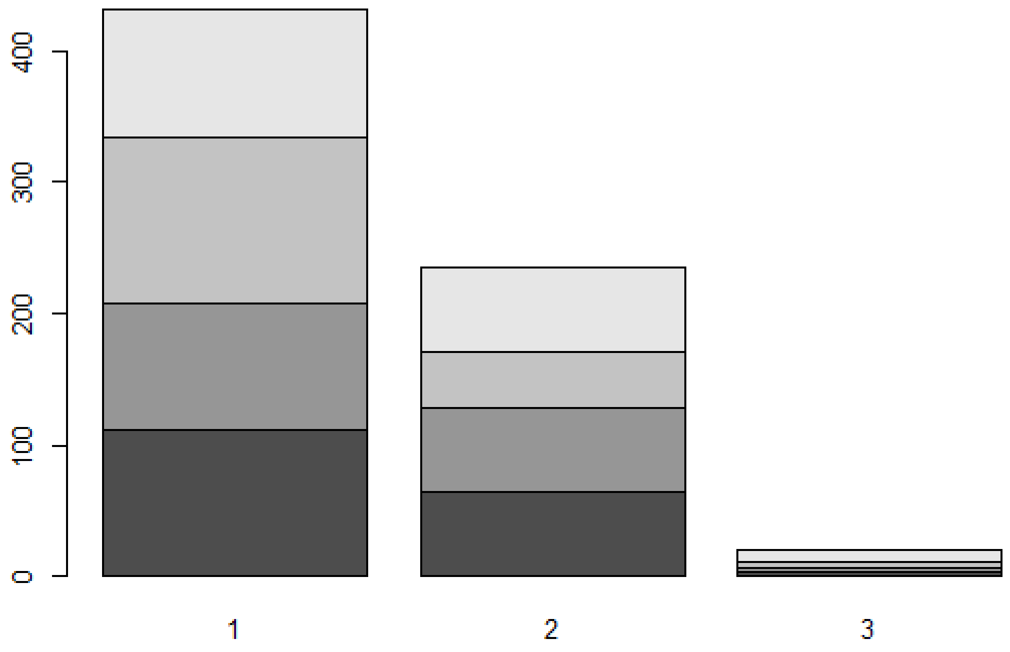
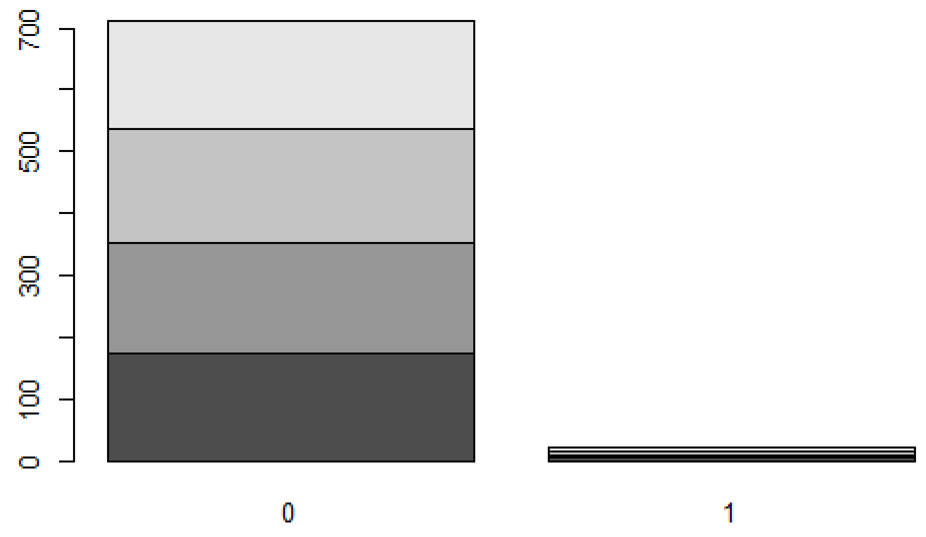


Figure shows relationship between ‘mnth’ and ‘holiday’


0
100
300
500
700
0
1