Predicting Rental Bikes

Mohit Kumar

July 2020

Contents

1.1 Problem Statement

The Bike Rental Data contains the daily count of rental bikes between the year 2011 and 2012 with corresponding weather and seasonal information. We would like to predict the daily count of rental count in order to automate the system.

1.2 Data

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 sample of the data set that we are using to predict the count:

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

instant	Dteday	Season	yr	mnth	holiday	weekday
1	1/1/2011	1	0	1	0	6
2	1/2/2011	1	0	1	0	0
3	1/3/2011	1	0	1	0	1
4	1/4/2011	1	0	1	0	2
5	1/5/2011	1	0	1	0	3

Table 1.2: Bike Rental Sample Data (Columns: 9-14)

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

Below are the variables we used to predict the count of bike rentals

Table 1.3: Bike Rental Predictors

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

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 distributions of the Numeric variables. Most analysis like regression, require the data to be normally distributed.*

2.1.1 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)

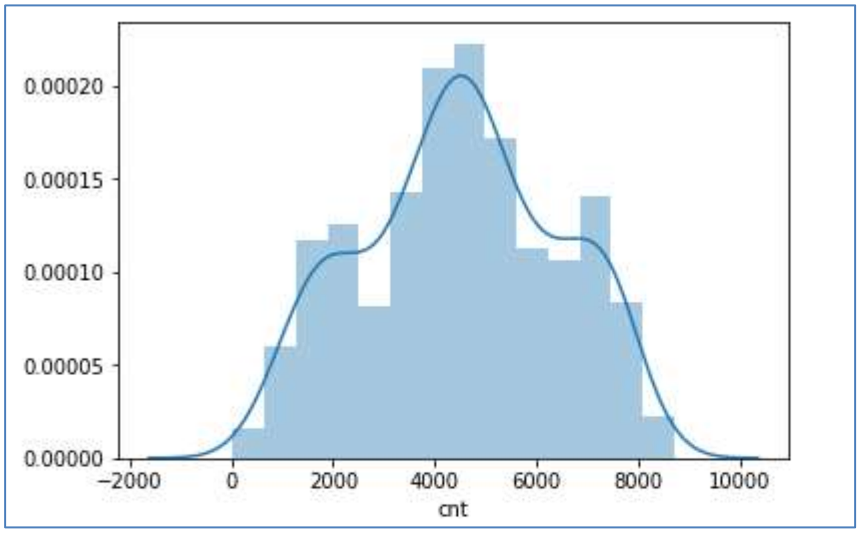
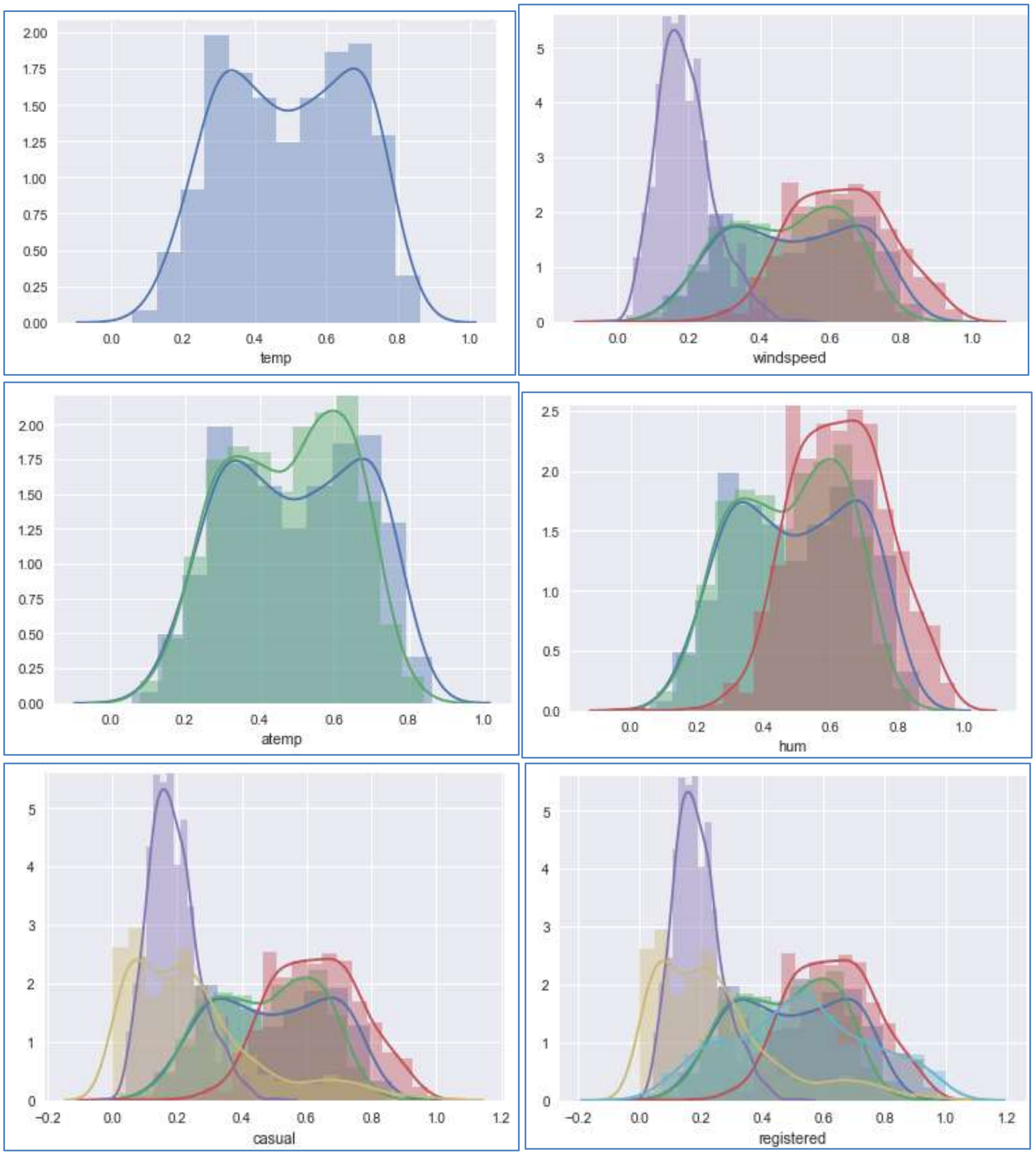


Figure 2.2 showing distribution of dependent variables (python code in Appendix B)



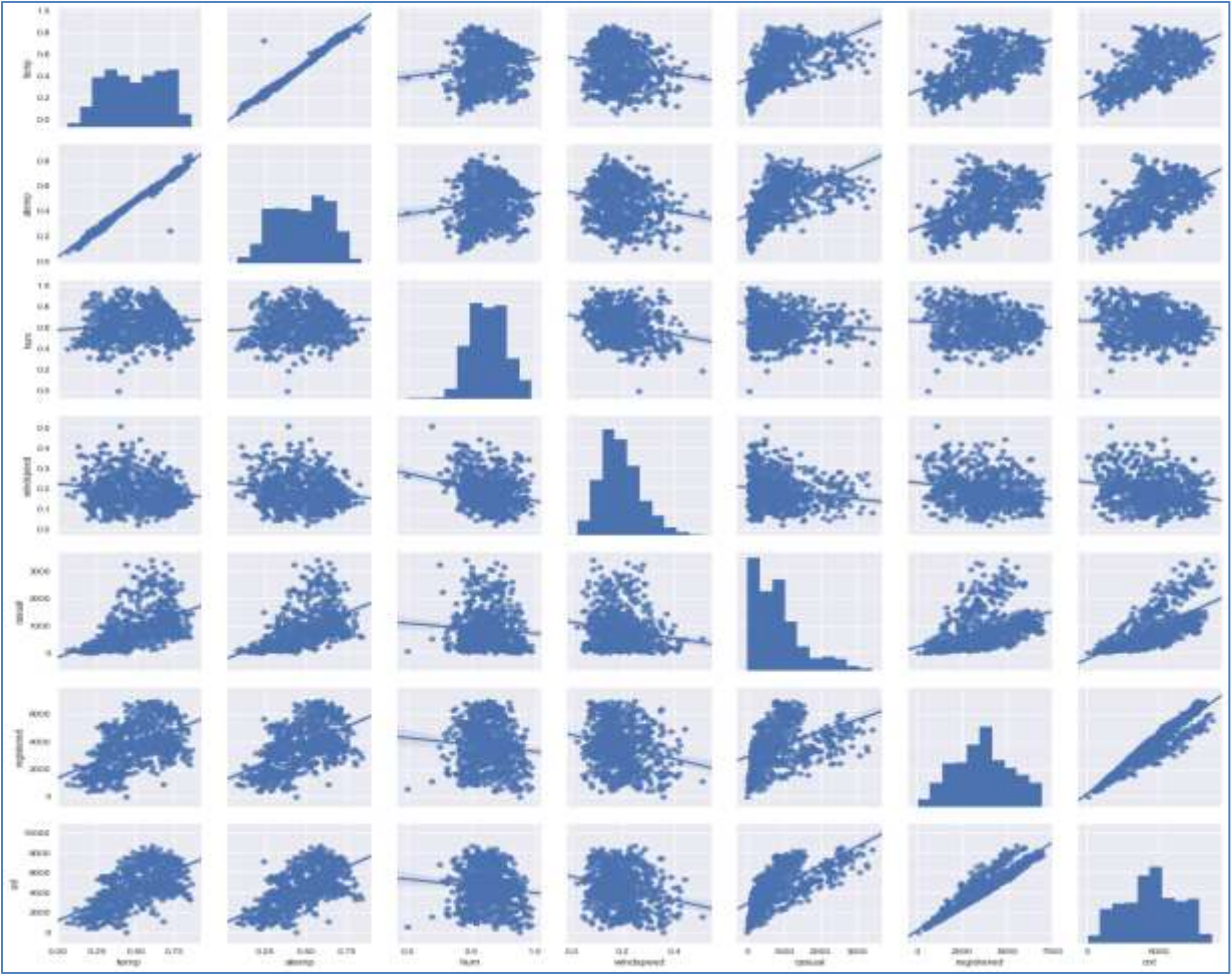
2.1.2 Bivariate Analysis

Ggpair function built upon ggplot2, GGally provides templates for combining plots into a matrix through the ggpairs function. Such a matrix of plots can be useful for quickly exploring the relationships between multiple columns of data in a data frame. The lower and upper arguments to the ggpairs function specifies the type of plot or data in each position of the lower or upper diagonal of the matrix, respectively. For continuous X and Y data, one can specify the smooth option to include a regression line.

Below figures shows relationship between independent variables and also with numeric target variable using ggpair

i. Below ggpair graph is showing clearly that relationship between independent variables 'temp' and 'atemp' are very strong.
ii. The relationship between 'hum' , 'windspeed' with target variable 'cnt' is less.

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



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.

Below table illustrate no missing value present in the data.

2.1 missing values

s.no	Variables	missing values
1	dteday	0
2	season	0
3	yr	0
4	mnth	0
5	holiday	0
6	weekday	0
7	workingday	0
8	weathersit	0
9	temp	0
10	atemp	0
11	hum	0
12	windspeed	0
13	casual	0
14	registered	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 methods to detect outliers is Boxplot

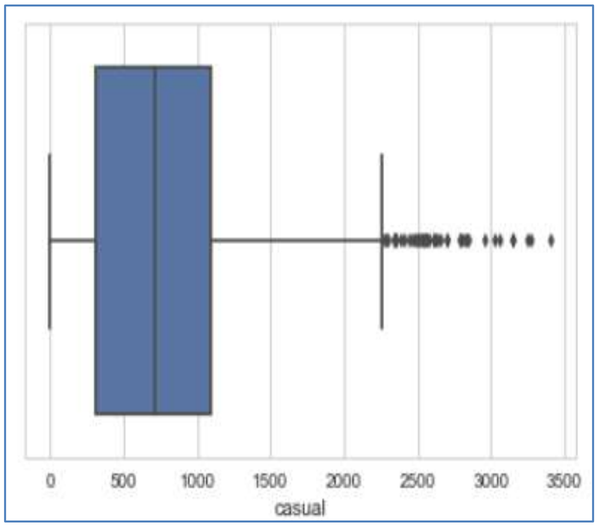
Fig 2.4 shows presence of Outliers in variable 'casual' and relationship between 'casual' and 'cnt' before removing Outliers

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’ Boxplot and relation between ‘cnt’ and’ casual’ (R code in Appendix B)



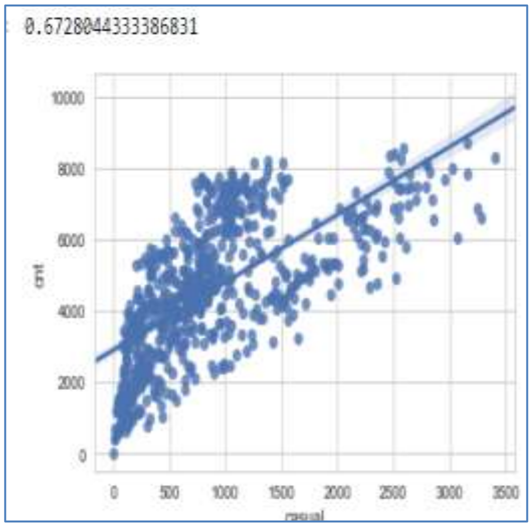
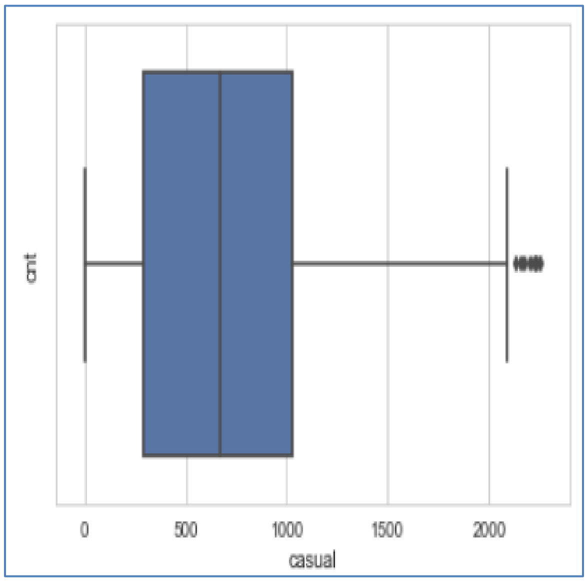
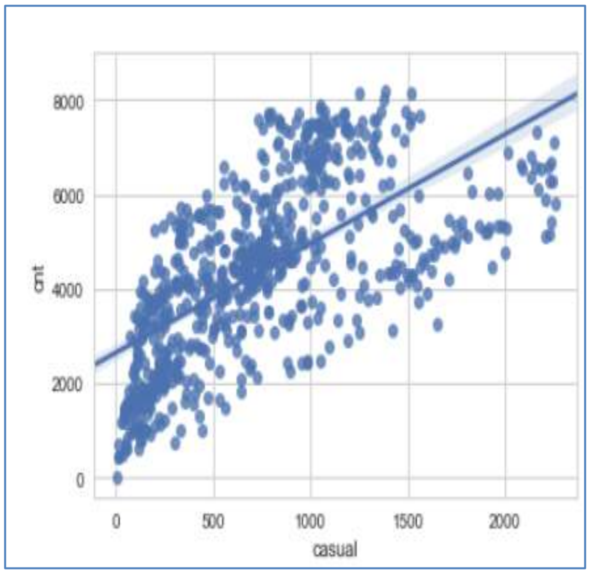


Figure 2.5 ‘casual’ Boxplot and relation between ‘casual’ and ‘cnt’ (R code in A





Since there is significant difference between Pearson coefficient correlation between before and after outlier detection for ‘casual’ and ‘cnt’ and losing nearly 40 observation so, we are not going to treat the outliers.

Correlation before outliers: 0.67 and after outlier treatment is 0.64

2.2.3 Features Selections

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.

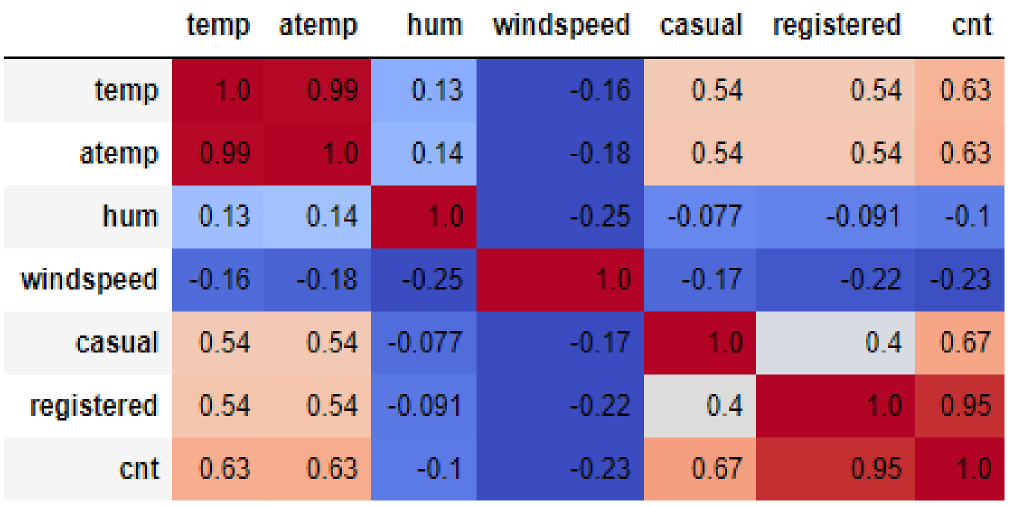
This becomes even more important when the number of features are very large. You need not use every feature at your disposal for creating an algorithm. You can assist your algorithm by feeding in only those features that are really important. I have myself witnessed feature subsets giving better results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

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 Correlogram plot.

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



	temp	atemp	hum	windspeed	casual	registered	cnt
temp	1.0	0.99	0.13	-0.16	0.54	0.54	0.63
atemp	0.99	1.0	0.14	-0.18	0.54	0.54	0.63
hum	0.13	0.14	1.0	-0.25	-0.077	-0.091	-0.1
windspeed	-0.16	-0.18	-0.25	1.0	-0.17	-0.22	-0.23
casual	0.54	0.54	-0.077	-0.17	1.0	0.4	0.67
registered	0.54	0.54	-0.091	-0.22	0.4	1.0	0.95
cnt	0.63	0.63	-0.1	-0.23	0.67	0.95	1.0

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

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

Chorogram: it helps us visualize the data in correlation matrices. correlograms are implemented through the **corrgram (x, order =, panel=, lower. panel=, upper. panel=, text. panel=, diag. panel=)**

2.4.1 Dimensionality Reduction for numeric variables

Above Fig 2.6 is showing

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

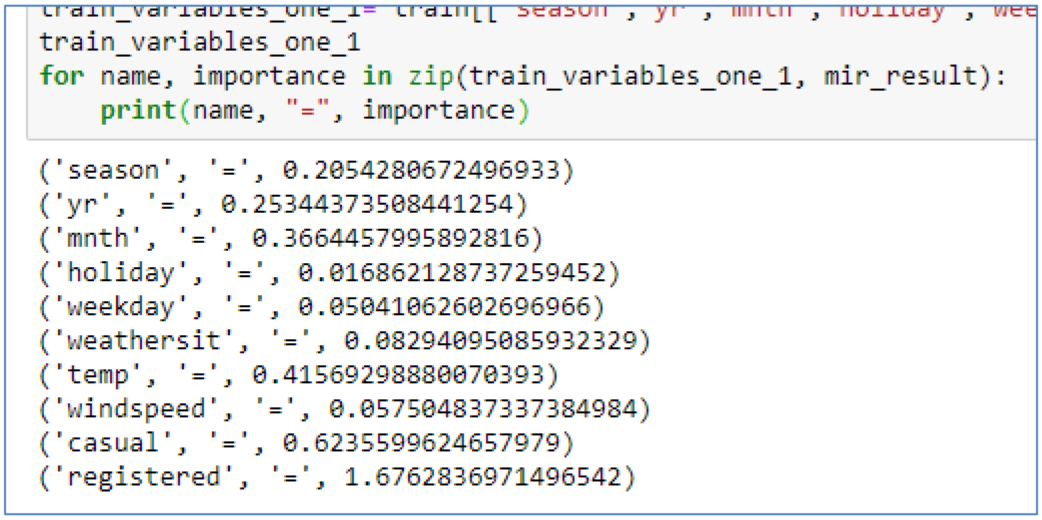
And it is also showing there is almost no relationship between independent variable ‘hum’ and dependent variable ‘cnt’. so, ‘hum’ is not so important to predict.

Subsetting two independent features ‘atemp’ and ‘hum’ from actual dataset.

2.4.2 Dimensional Reduction using Random Forest Variable Importance

There are several methods to check the relation between categorical variable , but here using Random Forest to get the importance of variables .

Figure 2.7 Variable Importance



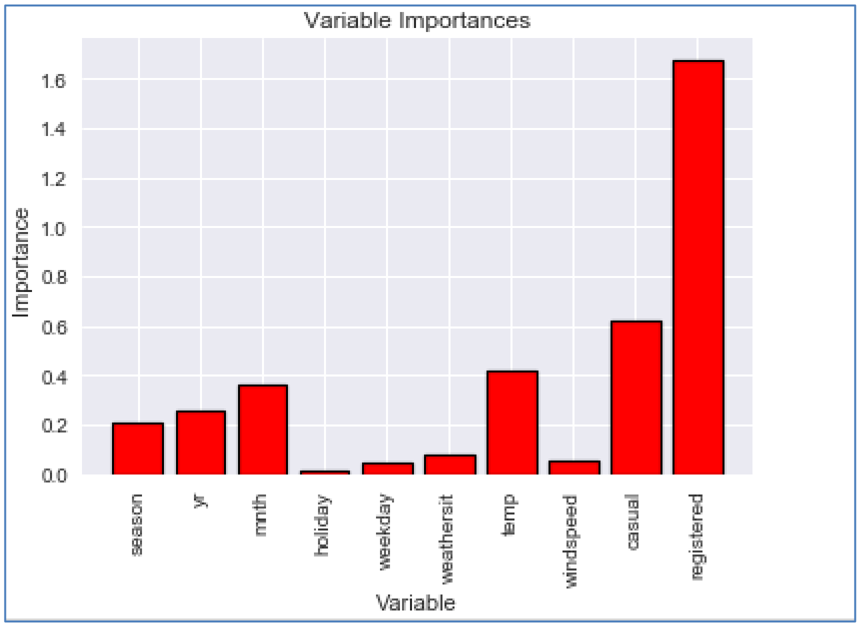
```
train_variables_one_1= train[['season','yr','mnth','holiday','wee
train_variables_one_1
for name, importance in zip(train_variables_one_1, mir_result):
    print(name, "=", importance)
```

```
('season', '=', 0.2054280672496933)
('yr', '=', 0.25344373508441254)
('mnth', '=', 0.3664457995892816)
('holiday', '=', 0.016862128737259452)
('weekday', '=', 0.05041062602696966)
('weathersit', '=', 0.08294095085932329)
('temp', '=', 0.41569298880070393)
('windspeed', '=', 0.057504837337384984)
('casual', '=', 0.6235599624657979)
('registered', '=', 1.6762836971496542)
```

The above figure shows that variabl’es ‘season’ ‘windspeed’ , ‘weekday ‘ ,‘weathersit’ and ‘holiday’ are less importance in predict the ‘cnt’ of Rental Bikes .

So these variable are removing while performaning Random Forest Model.

Figure 2.8 Variable Imporatnce Graph



2.2.4 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.

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

In rental dataset numeric variables like ‘temp’ , ‘atem’ ,’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
0.037852113	0.09384926
0.034624413	0.17455963
0.025234742	0.21628646
0.042840376	0.21628646
0.019366197	0.12575801

Chapter 3

Modelling

3.1 Model Selection

In out 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 two methods to evaluating performance of model

i. **MAPE** : (Mean Absolute Percent Error) measures the size of the error in percentage terms. It is calculated as the average of the unsigned percentage error.

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

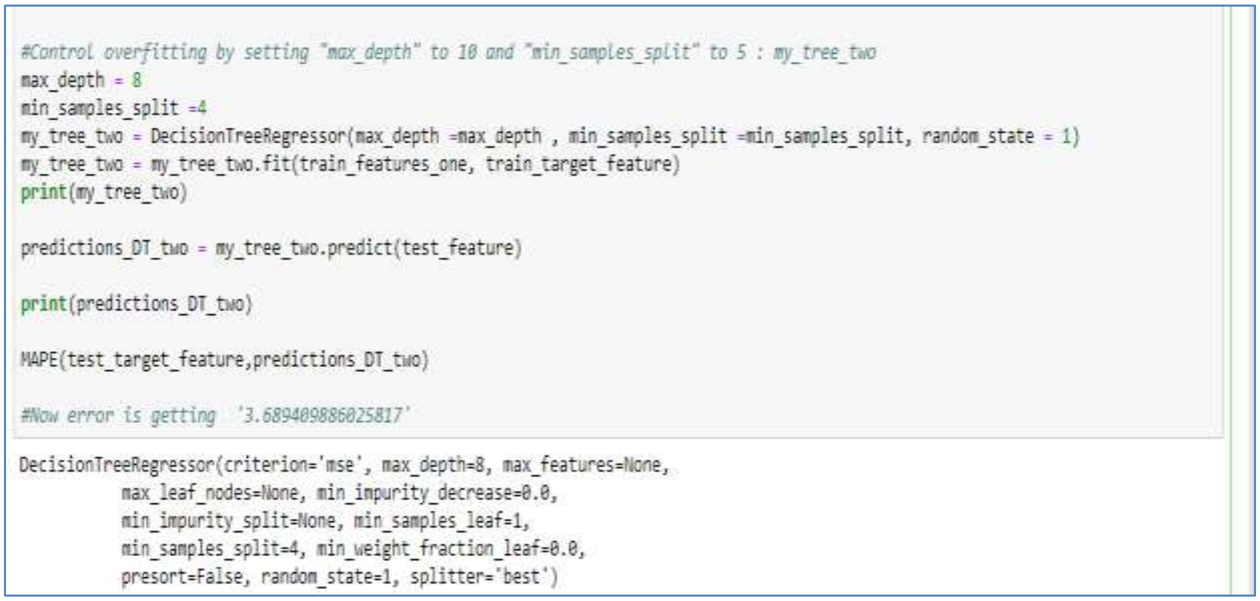
ii. **RMSE:** (Root Mean Square Error) is a frequently used measure of the difference between values predicted by a model and the values actually observed from the environment that is being modelled.

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

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.

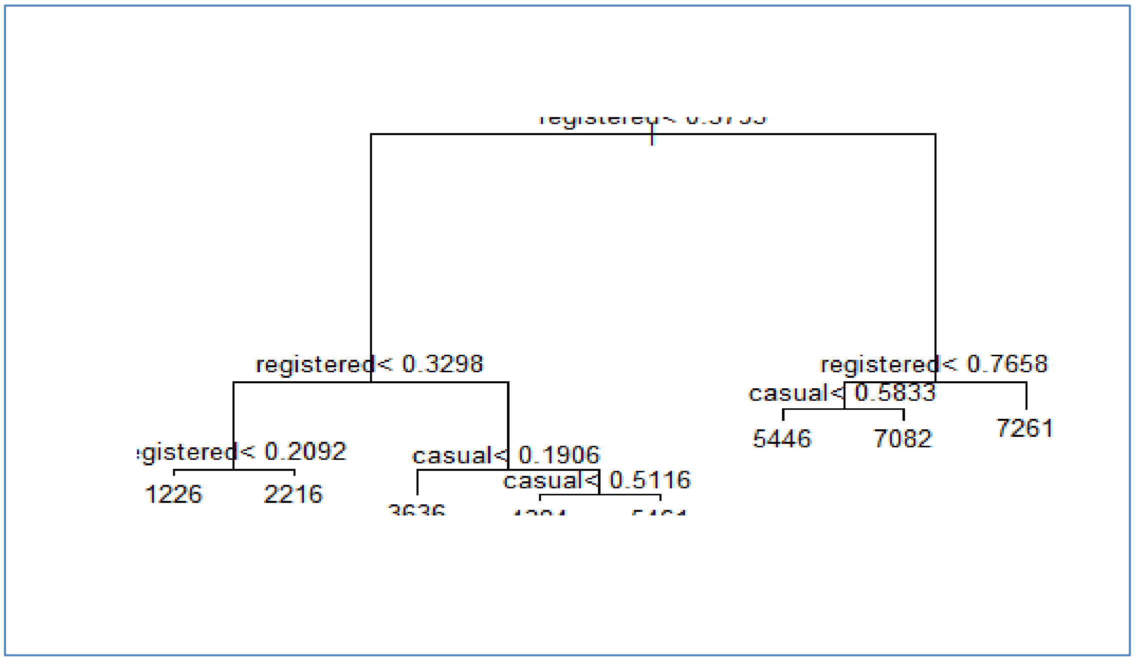
Figure 3.2.1 Decision Tree Algorithm



```
#Control overfitting by setting "max_depth" to 10 and "min_samples_split" to 5 : my_tree_two
max_depth = 8
min_samples_split =4
my_tree_two = DecisionTreeRegressor(max_depth =max_depth , min_samples_split =min_samples_split, random_state = 1)
my_tree_two = my_tree_two.fit(train_features_one, train_target_feature)
print(my_tree_two)

predictions_DT_two = my_tree_two.predict(test_feature)

print(predictions_DT_two)

MAPE(test_target_feature,predictions_DT_two)

#Now error is getting  '3.689409886025817'
```

```
DecisionTreeRegressor(criterion='mse', max_depth=8, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=4, min_weight_fraction_leaf=0.0,
           presort=False, random_state=1, splitter='best')
```

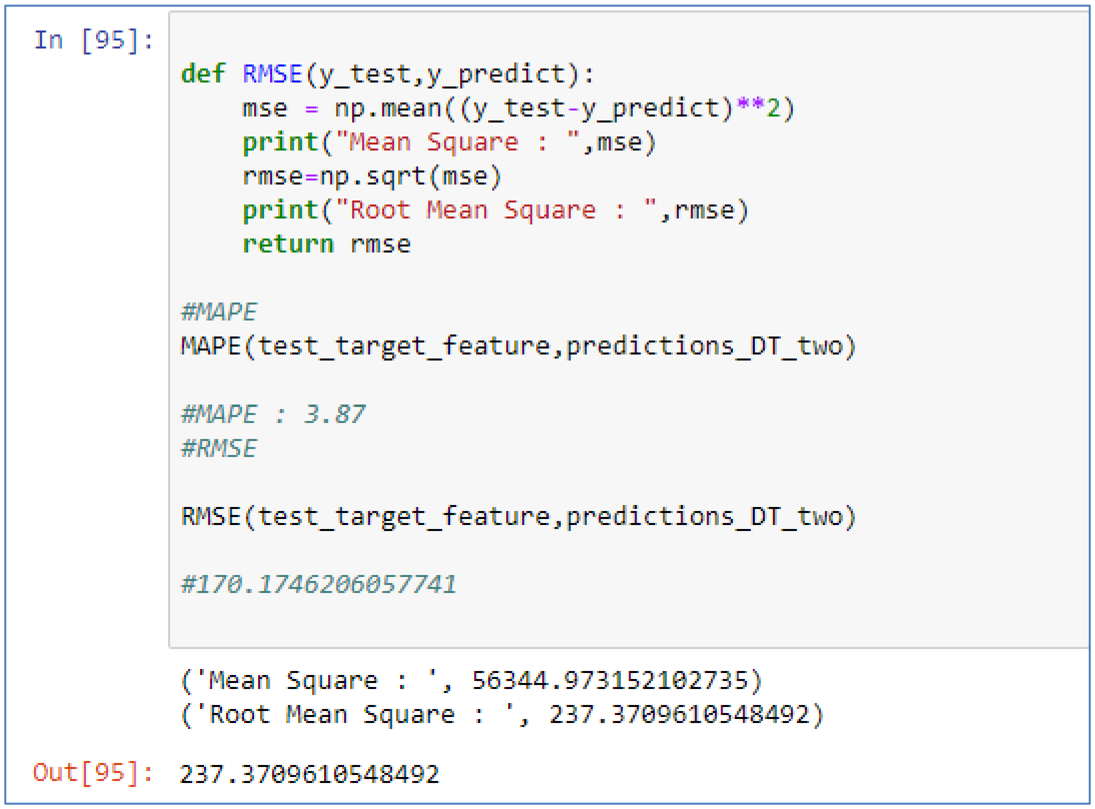
Figure 3.2.2 Graphical Representation of Decision tree



Look at the above figure 3.2 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 3.2.3 Evaluation of Decision Tree using MAPE and RMSE



```
In [95]:
def RMSE(y_test,y_predict):
    mse = np.mean((y_test-y_predict)**2)
    print("Mean Square : ",mse)
    rmse=np.sqrt(mse)
    print("Root Mean Square : ",rmse)
    return rmse

#MAPE
MAPE(test_target_feature,predictions_DT_two)

#MAPE : 3.87
#RMSE

RMSE(test_target_feature,predictions_DT_two)

#170.1746206057741

('Mean Square : ', 56344.973152102735)
('Root Mean Square : ', 237.3709610548492)

Out[95]: 237.3709610548492
```

In Figure 3.2.3 Model Accuracy is   1- 3.8 = 0.962   which is nearly 96.2% it is quite good but RMSE is 237 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.

3.3 Random Forest

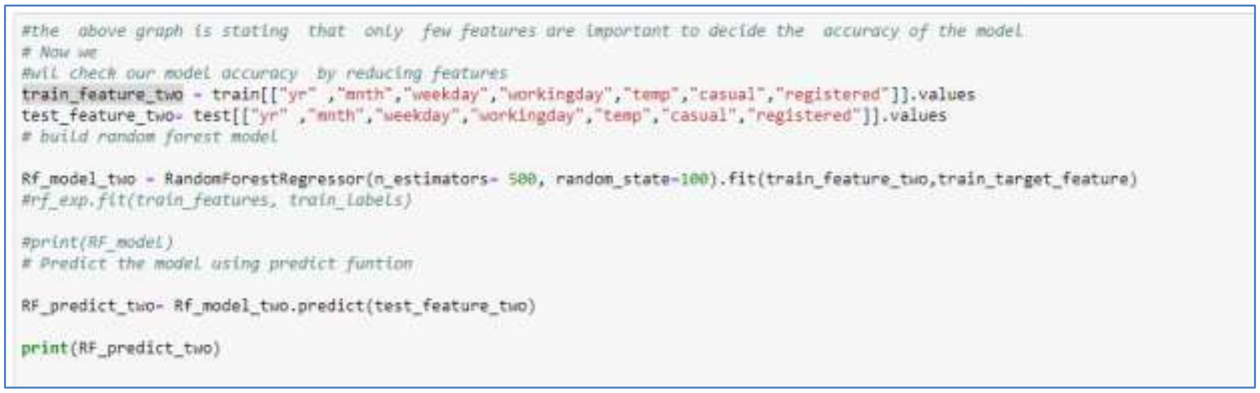
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

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 poor in order to improve the performance of the model developing model using Random Forest.

Figure 3.3.1 Random Forest Implementation



```
#the  above graph is stating  that  only  few features are important to decide the  accuracy of the model
# Now we
#wil check our model accuracy  by reducing features
train_feature_two = train[["yr" ,"mnth","weekday","workingday","temp","casual","registered"]].values
test_feature_two= test[["yr" ,"mnth","weekday","workingday","temp","casual","registered"]].values
# build random forest model

Rf_model_two = RandomForestRegressor(n_estimators= 500, random_state=100).fit(train_feature_two,train_target_feature)
#rf_exp.fit(train_features, train_labels)

#print(RF_model)
# Predict the model using predict funtion

RF_predict_two= Rf_model_two.predict(test_feature_two)

print(RF_predict_two)
```

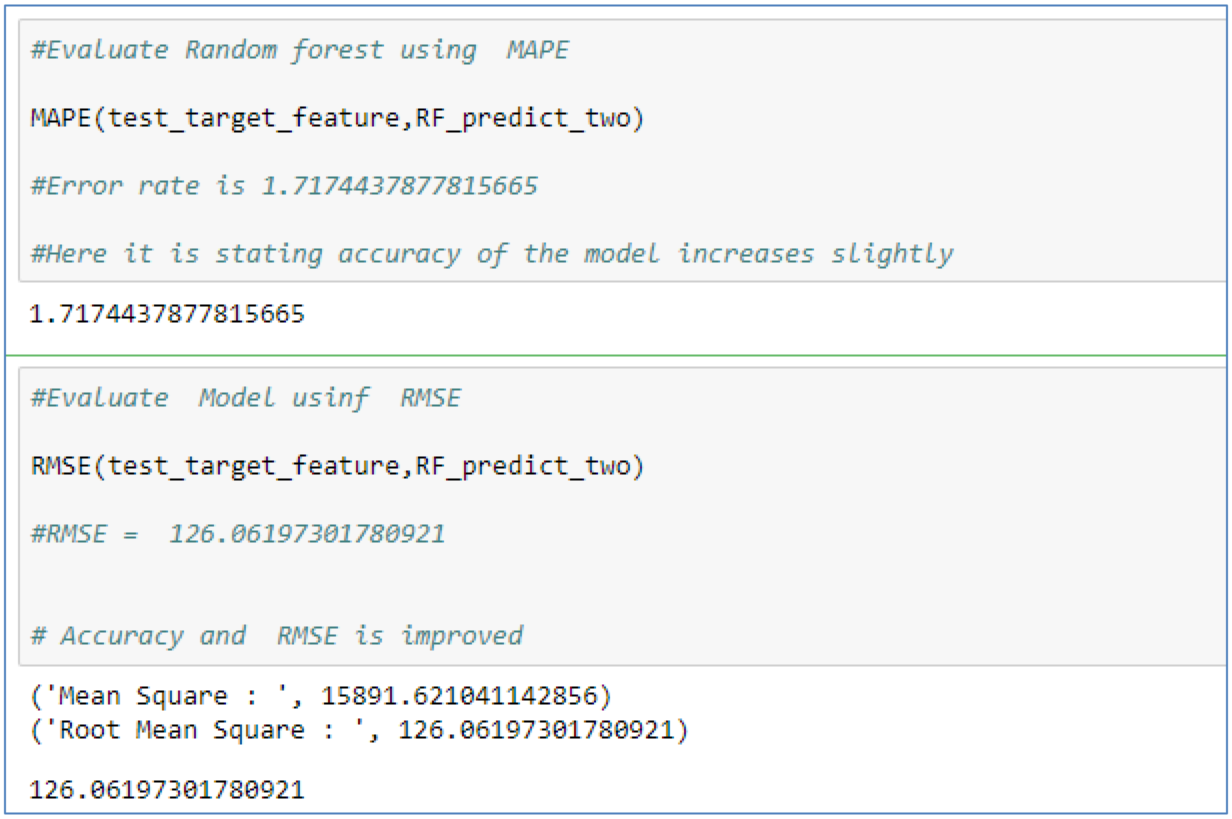
Mtry: Number of variables to split at each node i.e. 7.

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



```
#Evaluate Random forest using  MAPE

MAPE(test_target_feature,RF_predict_two)

#Error rate is 1.7174437877815665

#Here it is stating accuracy of the model increases slightly
```

```
1.7174437877815665
```

```
#Evaluate  Model usinf  RMSE

RMSE(test_target_feature,RF_predict_two)

#RMSE =  126.06197301780921


# Accuracy and  RMSE is improved
```

```
('Mean Square : ', 15891.621041142856)
('Root Mean Square : ', 126.06197301780921)

126.06197301780921
```

Fig 3.2.2 shows Random Forest model performs dramatically better than Decision tree on both training and test data and well also improve the Accuracy (MAPE = 1.71) and decrease the RMSE (126) of the model which is quite impressive.

Using Linear Regression, we will predict the 'cnt 'values and compare with Random Forest.

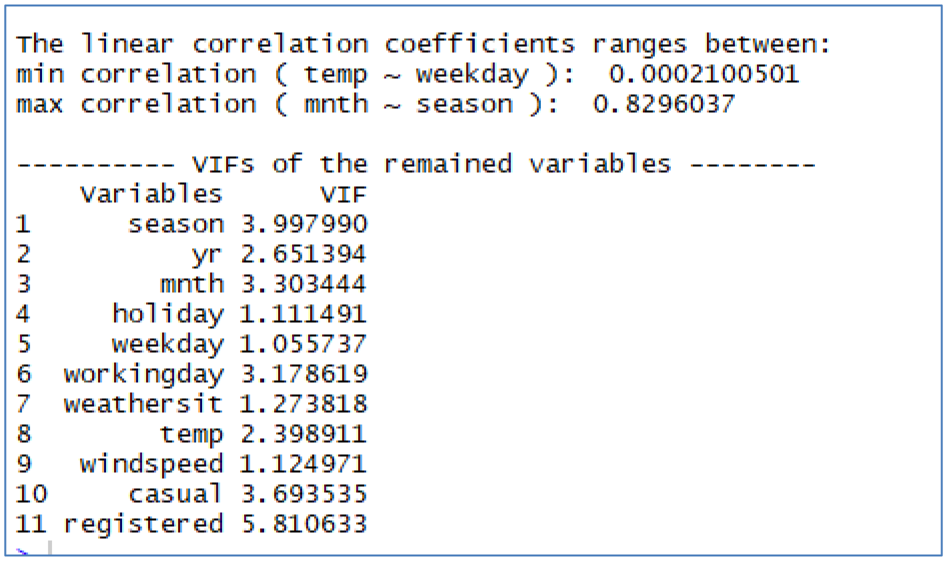
3.4 Linear Regression

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): It quantifies the multicollinearity between the independent variables.

As Linear regression will work well if multicollinearity between the Independent variables are less.

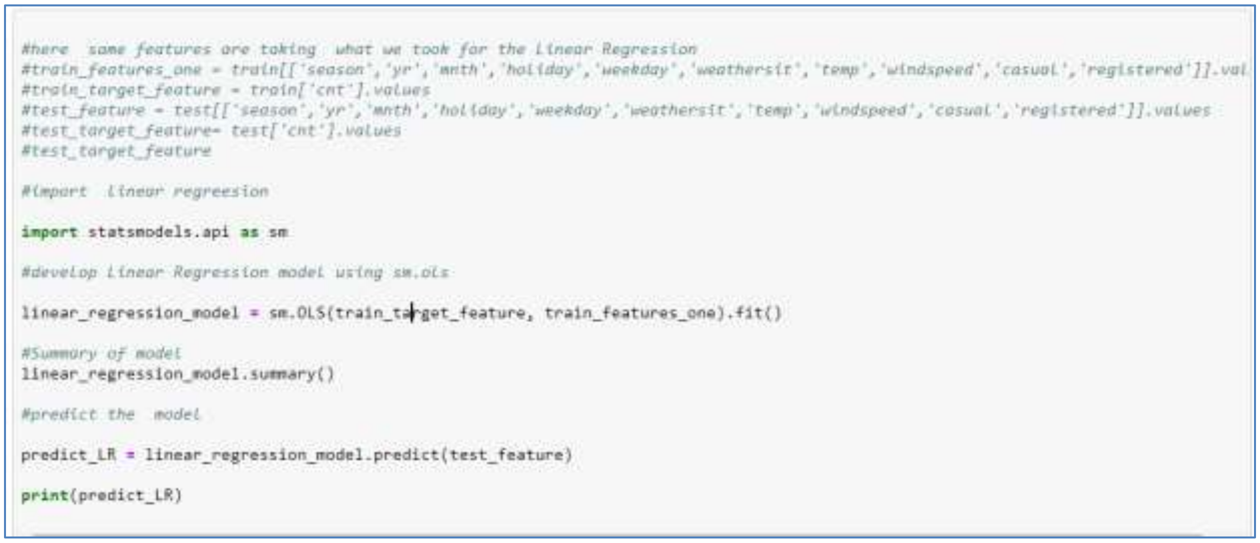
Figure 3.4.1 Multi collinearity between Independent variables



```
The linear correlation coefficients ranges between:
min correlation ( temp ~ weekday ):  0.0002100501
max correlation ( mnth ~ season ):  0.8296037

---------- VIFs of the remained variables --------
    Variables      VIF
1      season 3.997990
2          yr 2.651394
3        mnth 3.303444
4     holiday 1.111491
5     weekday 1.055737
6  workingday 3.178619
7  weathersit 1.273818
8        temp 2.398911
9   windspeed 1.124971
10     casual 3.693535
11 registered 5.810633
```

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



```
#here  same features are taking  what we took for the Linear Regression
#train_features_one = train[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].val
#train_target_feature = train['cnt'].values
#test_feature = test[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].values
#test_target_feature= test['cnt'].values
#test_target_feature

#import  linear regreesion

import statsmodels.api as sm

#develop linear Regression model using sm.ols

linear_regression_model = sm.OLS(train_target_feature, train_features_one).fit()

#Summary of model
linear_regression_model.summary()

#predict the  model

predict_LR = linear_regression_model.predict(test_feature)

print(predict_LR)
```

Dep. Variable:	y	**R-squared:**	1.000
Model:	OLS	Adj. R-squared:	1.000
Method:	Least Squares	F-statistic:	9.920e+07
Date:	Thu, 02 July 2020	Prob(F-statistic):	0.00
Time:	15:47:46	Log-Likelihood:	-1597.5
No. Observations:	584	AIC:	3215
Df Residuals:	574	BIC:	3259
Df Model:	10		
Covariance Type:	nonrobust		

Here:

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

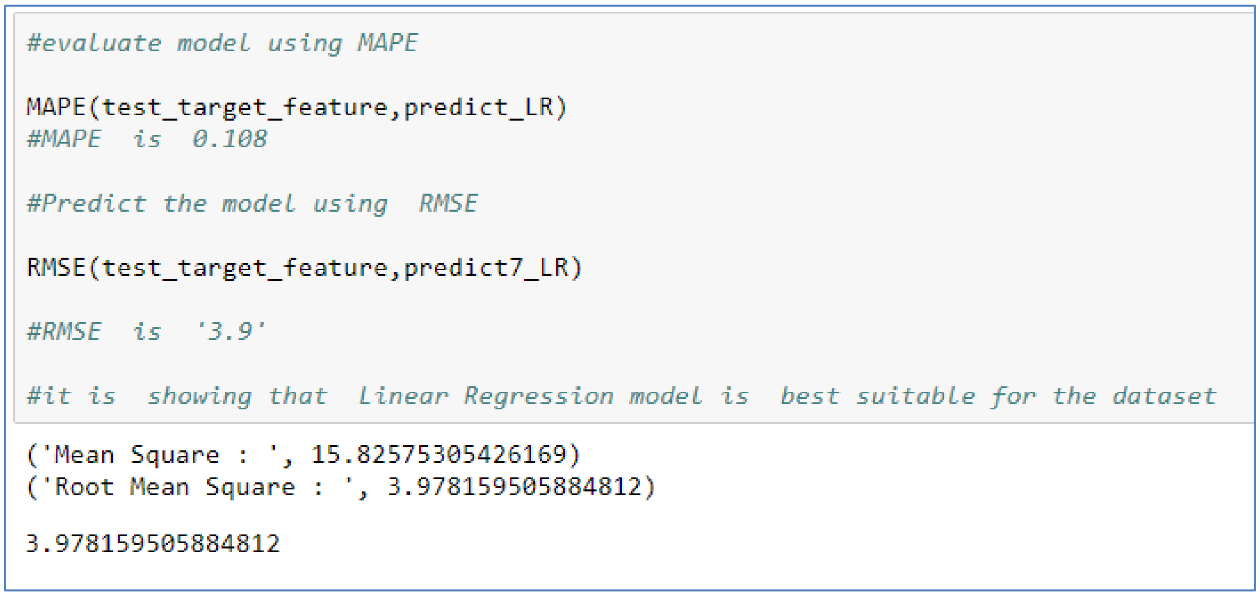
Multiple R-squared: 1, Adjusted R-squared: 1

Here residual Standard error is quite less so the distance between predicted values f(x[I,]) and actual 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.

3.4.2 Evaluation of Linear regression Model

Figure 3.4.3 Evaluation of Regression Model



```
#evaluate model using MAPE

MAPE(test_target_feature,predict_LR)
#MAPE  is  0.108

#Predict the model using  RMSE

RMSE(test_target_feature,predict7_LR)

#RMSE  is  '3.9'

#it is  showing that  Linear Regression model is  best suitable for the dataset
```

```
('Mean Square : ', 15.82575305426169)
('Root Mean Square : ', 3.978159505884812)

3.978159505884812
```

From above figure it is clearly showing that Model Accuracy is 99.9 % and RMSE is nearly equal to 3.9.

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 so conclusion is

Conclusion: - The bike rental data Linear Regression Model is best suited model to predict the count.

Appendix A

Figure 3.4 Relationship between Weekdays and cnt

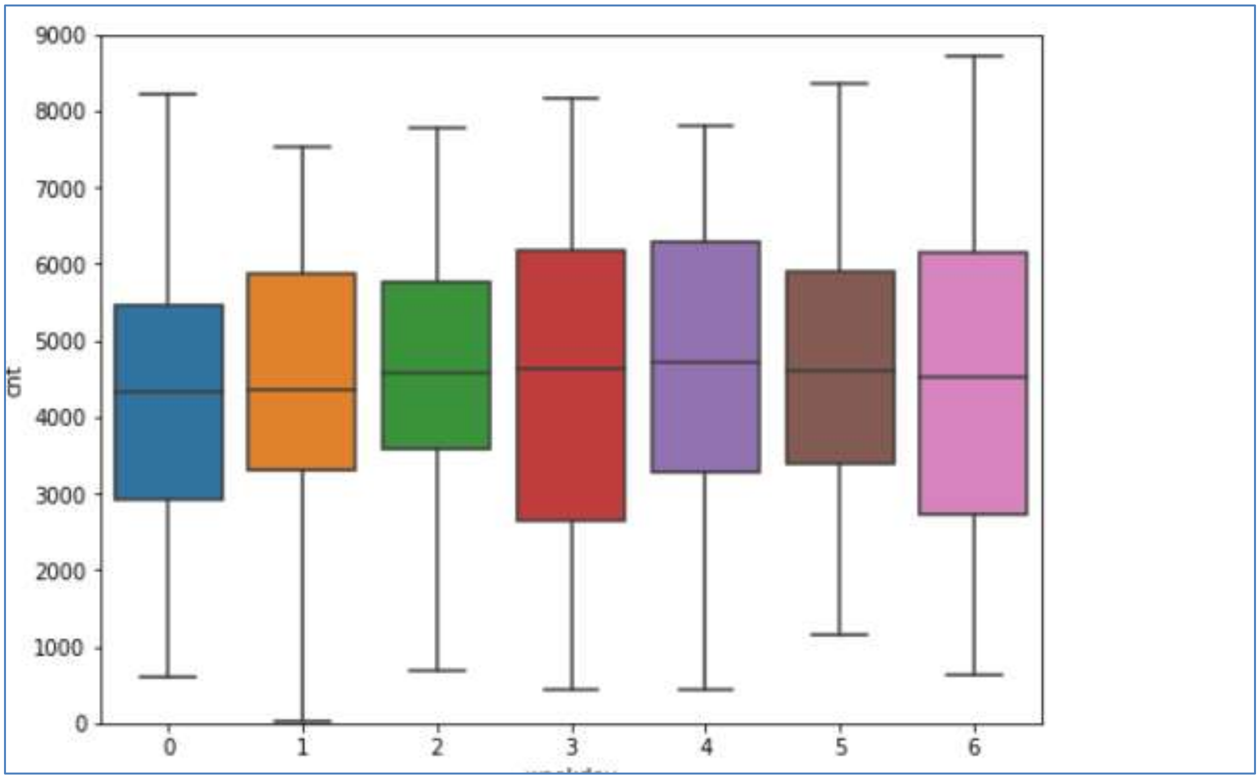
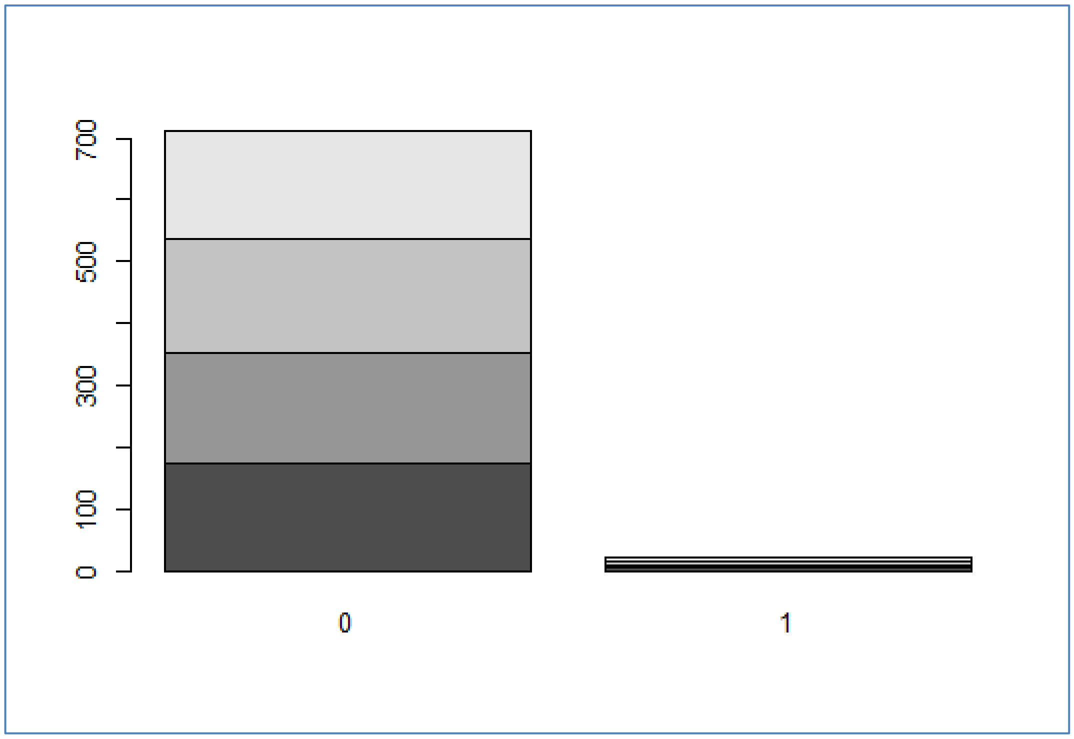


Figure 3.7 shows relationship between ‘mnth’ and ‘holiday’

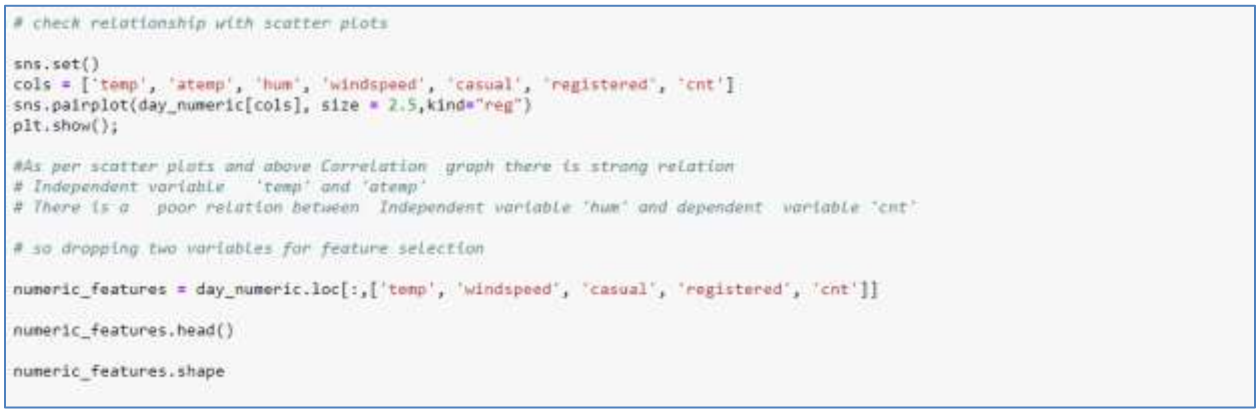


Appendix B - Python Code

Fig 2.1 and 2.2 Python Code

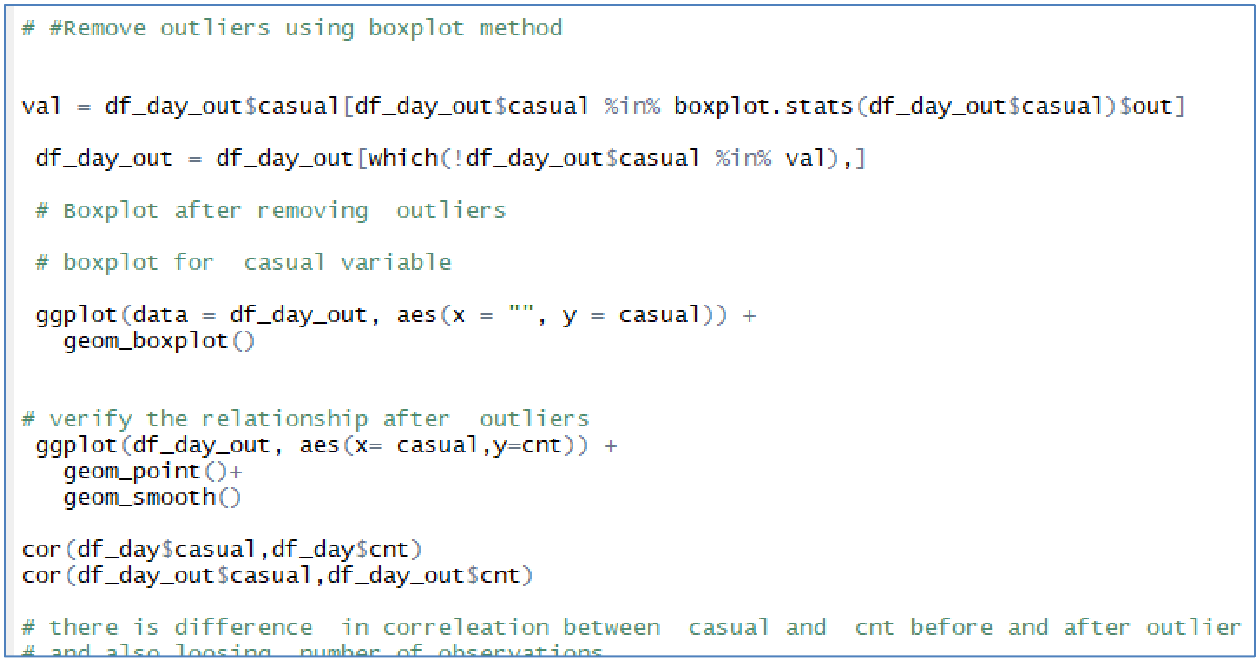
```
#Distribution  independent numeric variables
#Check whether  variable 'temp'is normal or not
sns.distplot(df_day['temp']);

#Check whether  variable 'atemp'is normal or not
sns.distplot(df_day['atemp']);

#Check whether  variable 'hum'is normal or not
sns.distplot(df_day['hum']);

#Check whether  variable 'windspeed'is normal or not
sns.distplot(df_day['windspeed']);


#Check whether  variable 'casual'is normal or not
sns.distplot(df_day['casual']);



#Check whether  variable 'registered'is normal or not
sns.distplot(df_day['registered']);


# it is clearly showing that chances of outliers present in  'casual' varible
```

Fig 2.3 Python Code



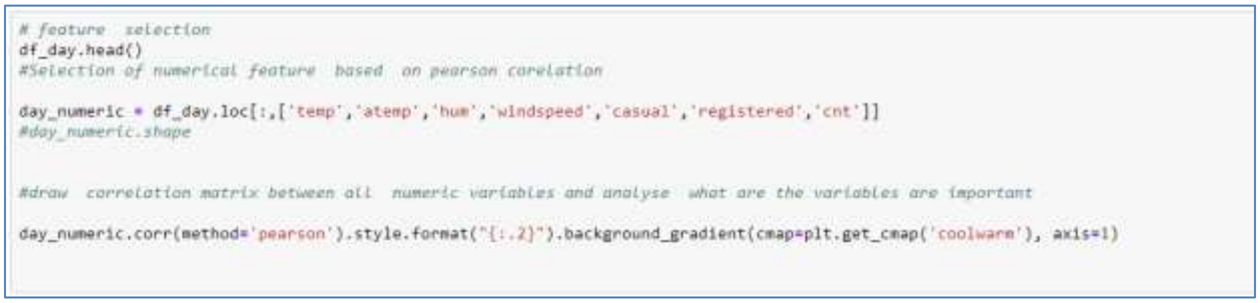
```
# check relationship with scatter plots

sns.set()
cols = ['temp', 'atemp', 'hum', 'windspeed', 'casual', 'registered', 'cnt']
sns.pairplot(day_numeric[cols], size = 2.5,kind="reg")
plt.show();

#As per scatter plots and above Correlation  graph there is strong relation
# Independent variable   'temp' and 'atemp'
# There is a   poor relation between  Independent variable 'hum' and dependent  variable 'cnt'

# so dropping two variables for feature selection

numeric_features = day_numeric.loc[:,['temp', 'windspeed', 'casual', 'registered', 'cnt']]

numeric_features.head()

numeric_features.shape
```

Fig 2.4 and 2.5 Python Code



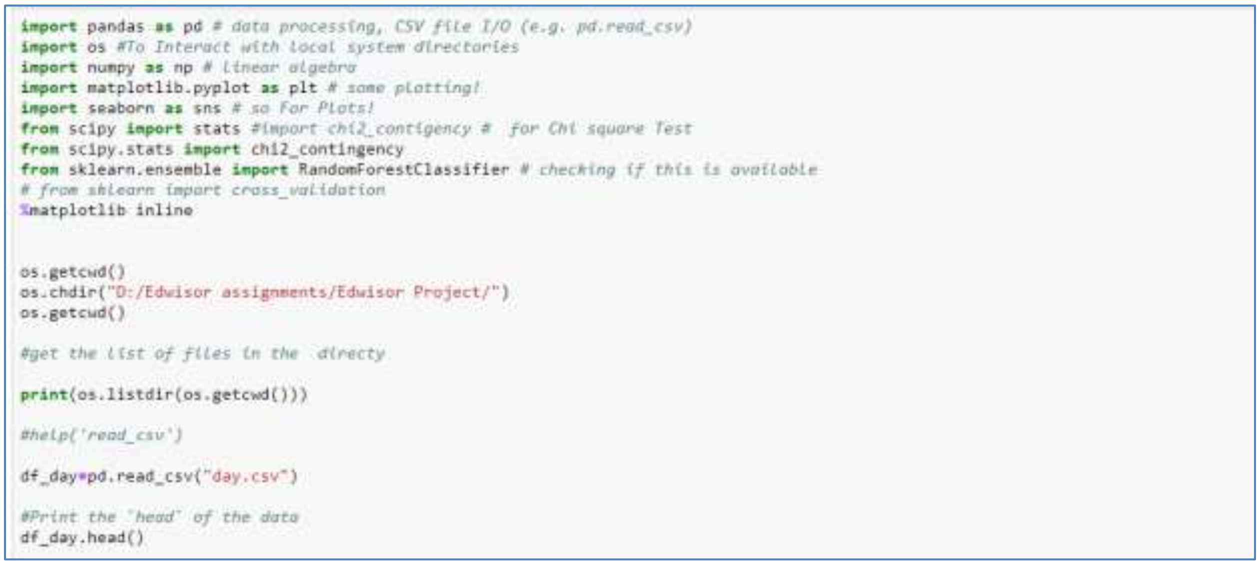
```
# #Remove outliers using boxplot method

val = df_day_out$casual[df_day_out$casual %in% boxplot.stats(df_day_out$casual)$out]

 df_day_out = df_day_out[which(!df_day_out$casual %in% val),]

 # Boxplot after removing  outliers

 # boxplot for  casual variable

 ggplot(data = df_day_out, aes(x = "", y = casual)) +
   geom_boxplot()


# verify the relationship after  outliers
 ggplot(df_day_out, aes(x= casual,y=cnt)) +
   geom_point()+
   geom_smooth()

cor(df_day$casual,df_day$cnt)
cor(df_day_out$casual,df_day_out$cnt)

# there is difference  in correleation between  casual and  cnt before and after outlier
# and also loosing  number of observations
```

Fig 2.6   Python Code

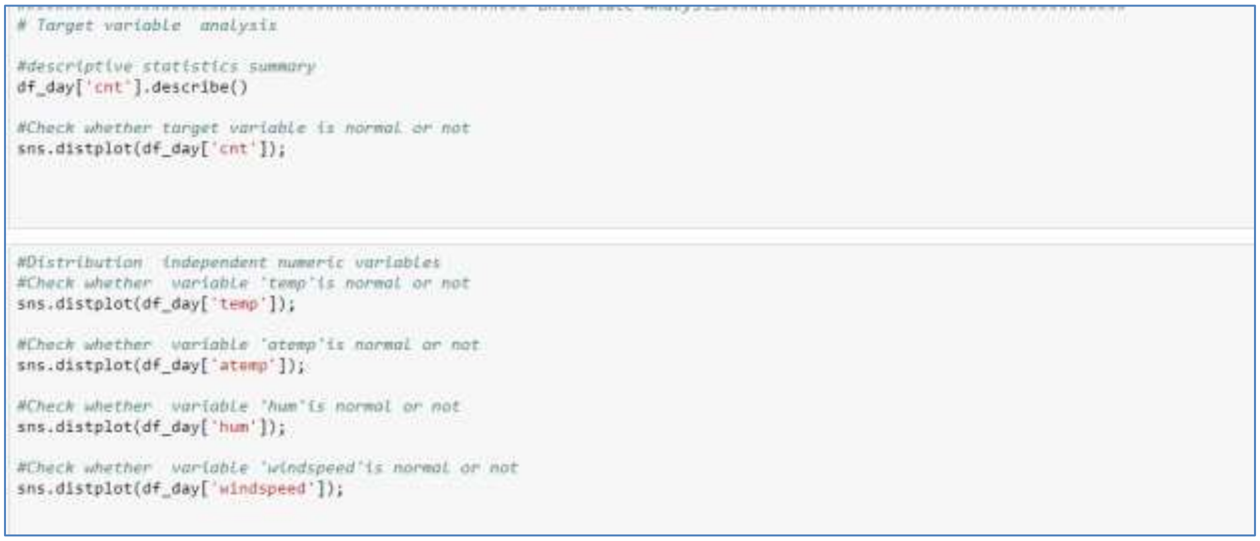


```
# feature  selection
df_day.head()
#Selection of numerical feature  based  on pearson corelation

day_numeric = df_day.loc[:,['temp','atemp','hum','windspeed','casual','registered','cnt']]
#day_numeric.shape


#draw  correlation matrix between all  numeric variables and analyse  what are the variables are important

day_numeric.corr(method='pearson').style.format("{:.2}").background_gradient(cmap=plt.get_cmap('coolwarm'), axis=1)
```

Complete Python File

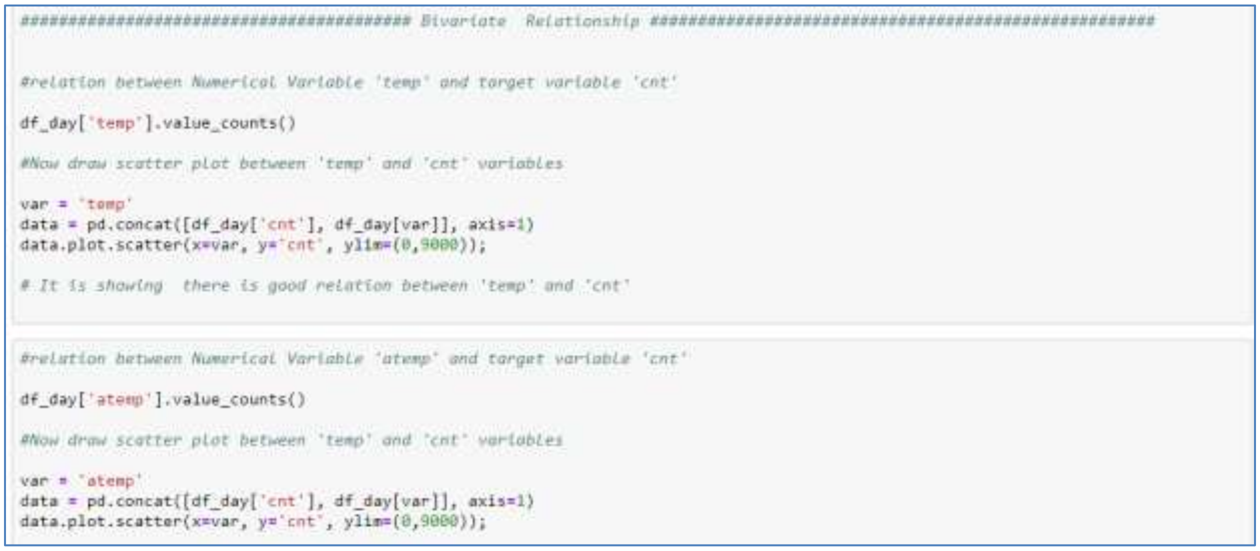


```
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os #To Interact with local system directories
import numpy as np # linear algebra
import matplotlib.pyplot as plt # some plotting!
import seaborn as sns # so For Plots!
from scipy import stats #import chi2_contigency #  for Chi square Test
from scipy.stats import chi2_contingency
from sklearn.ensemble import RandomForestClassifier # checking if this is available
# from sklearn import cross_validation
%matplotlib inline


os.getcwd()
os.chdir("D:/Edwisor assignments/Edwisor Project/")
os.getcwd()

#get the list of files in the  directy

print(os.listdir(os.getcwd()))

#help('read_csv')

df_day=pd.read_csv("day.csv")

#Print the 'head' of the data
df_day.head()
```



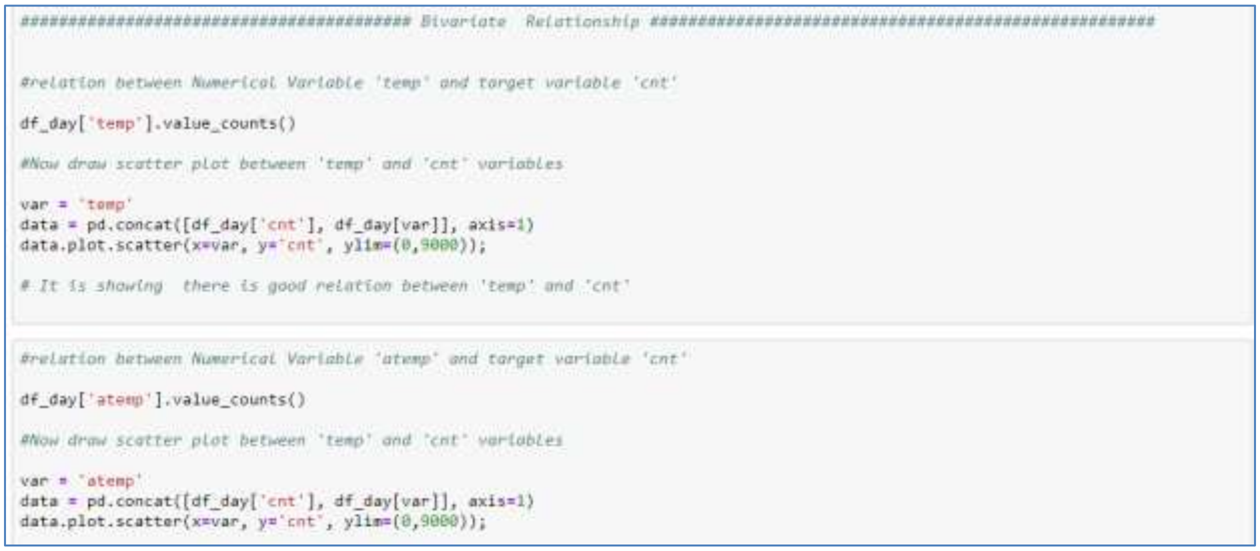
```
# Target variable  analysis

#descriptive statistics summary
df_day['cnt'].describe()

#Check whether target variable is normal or not
sns.distplot(df_day['cnt']);
```

```
#Distribution  independent numeric variables
#Check whether  variable 'temp'is normal or not
sns.distplot(df_day['temp']);

#Check whether  variable 'atemp'is normal or not
sns.distplot(df_day['atemp']);

#Check whether  variable 'hum'is normal or not
sns.distplot(df_day['hum']);

#Check whether  variable 'windspeed'is normal or not
sns.distplot(df_day['windspeed']);
```



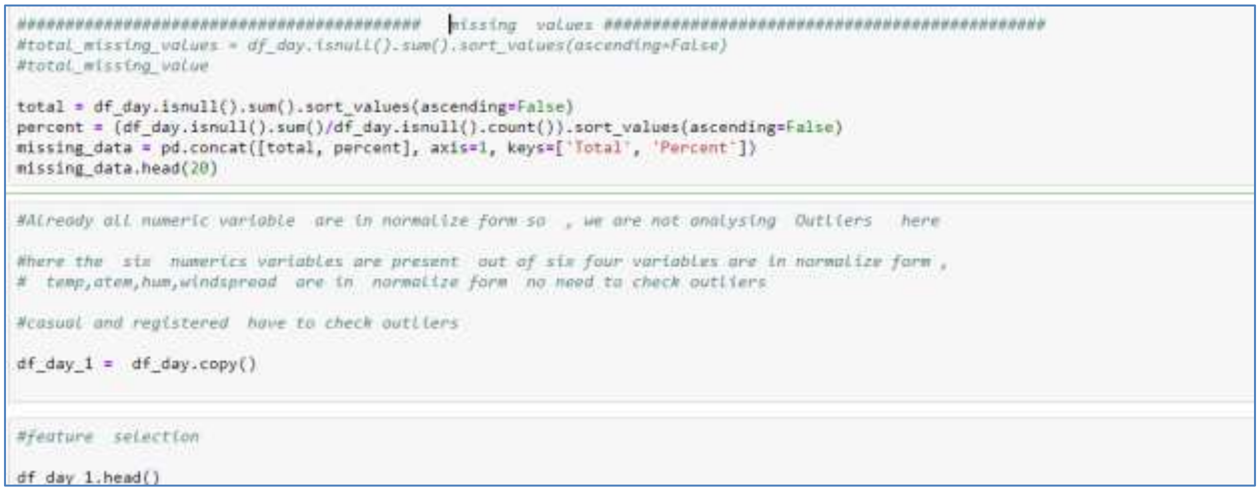
```
############################################ Bivariate  Relationship ######################################################


#relation between Numerical Variable 'temp' and target variable 'cnt'

df_day['temp'].value_counts()

#Now draw scatter plot between 'temp' and 'cnt' variables

var = 'temp'
data = pd.concat([df_day['cnt'], df_day[var]], axis=1)
data.plot.scatter(x=var, y='cnt', ylim=(0,9000));

# It is showing  there is good relation between 'temp' and 'cnt'
```

```
#relation between Numerical Variable 'atemp' and target variable 'cnt'

df_day['atemp'].value_counts()

#Now draw scatter plot between 'temp' and 'cnt' variables

var = 'atemp'
data = pd.concat([df_day['cnt'], df_day[var]], axis=1)
data.plot.scatter(x=var, y='cnt', ylim=(0,9000));
```



```
############################################ Bivariate  Relationship ######################################################


#relation between Numerical Variable 'temp' and target variable 'cnt'

df_day['temp'].value_counts()

#Now draw scatter plot between 'temp' and 'cnt' variables

var = 'temp'
data = pd.concat([df_day['cnt'], df_day[var]], axis=1)
data.plot.scatter(x=var, y='cnt', ylim=(0,9000));

# It is showing  there is good relation between 'temp' and 'cnt'
```

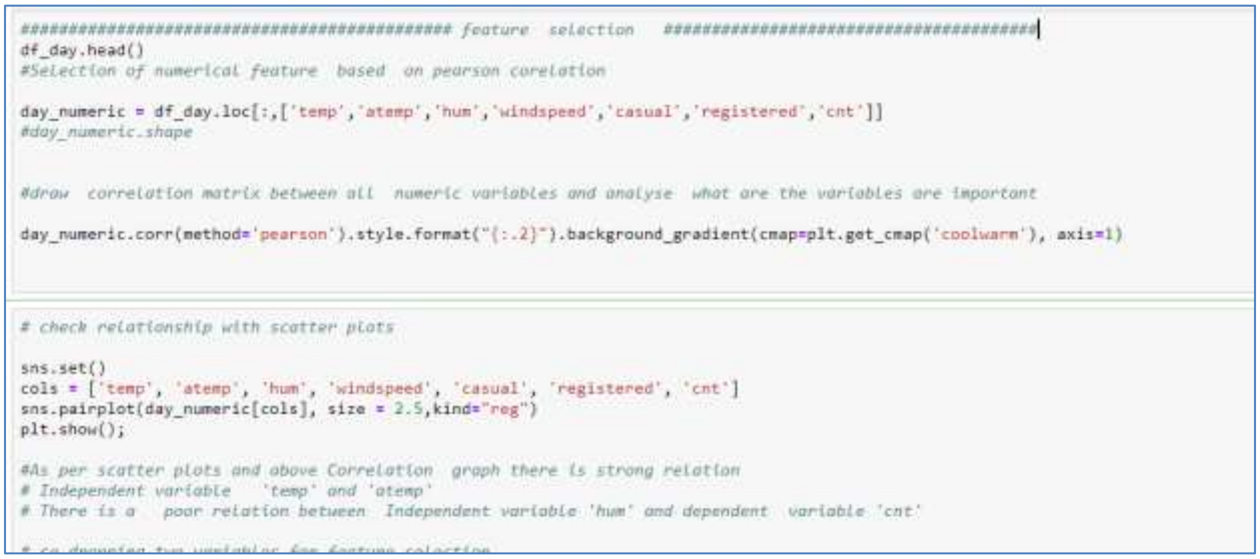
```
#relation between Numerical Variable 'atemp' and target variable 'cnt'

df_day['atemp'].value_counts()

#Now draw scatter plot between 'temp' and 'cnt' variables

var = 'atemp'
data = pd.concat([df_day['cnt'], df_day[var]], axis=1)
data.plot.scatter(x=var, y='cnt', ylim=(0,9000));
```



```
###########################################    missing  values ################################################
#total_missing_values = df_day.isnull().sum().sort_values(ascending=False)
#total_missing_value

total = df_day.isnull().sum().sort_values(ascending=False)
percent = (df_day.isnull().sum()/df_day.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

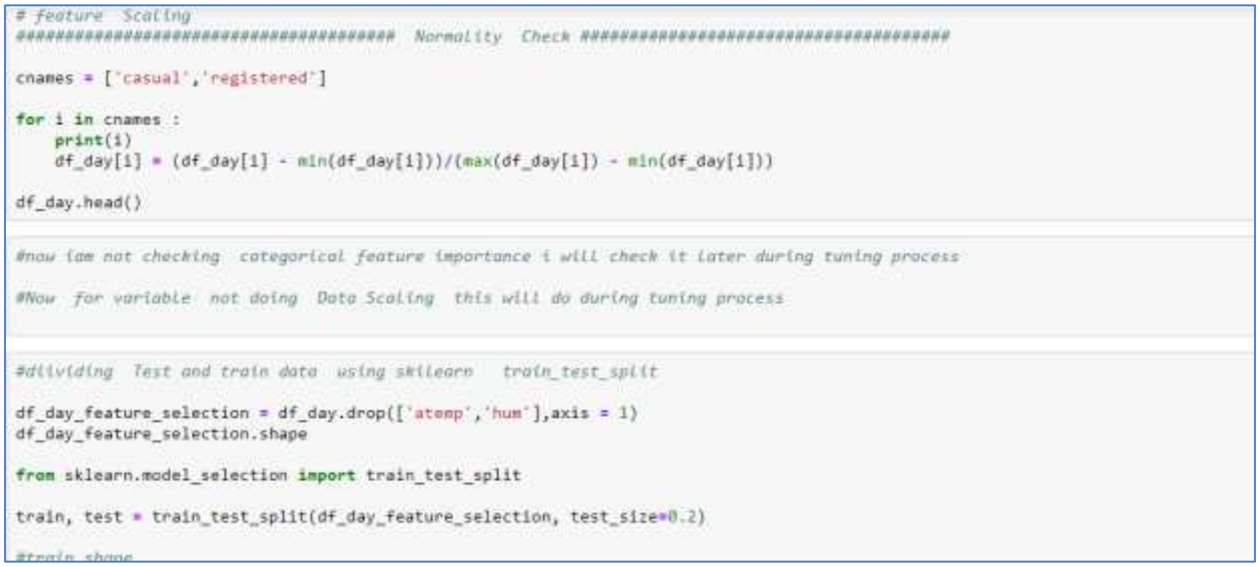
```
#Already all numeric variable  are in normalize form so  , we are not analysing  Outliers   here

#here the  six  numerics variables are present  out of six four variables are in normalize form ,
#  temp,atem,hum,windspread  are in  normalize form  no need to check outliers

#casual and registered  have to check outliers

df_day_1 =  df_day.copy()
```

```
#feature  selection

df_day_1.head()
```



```
################################################ feature  selection   ########################################
df_day.head()
#Selection of numerical feature  based  on pearson corelation

day_numeric = df_day.loc[:,['temp','atemp','hum','windspeed','casual','registered','cnt']]
#day_numeric.shape


#draw  correlation matrix between all  numeric variables and analyse  what are the variables are important

day_numeric.corr(method='pearson').style.format("{:.2}").background_gradient(cmap=plt.get_cmap('coolwarm'), axis=1)
```

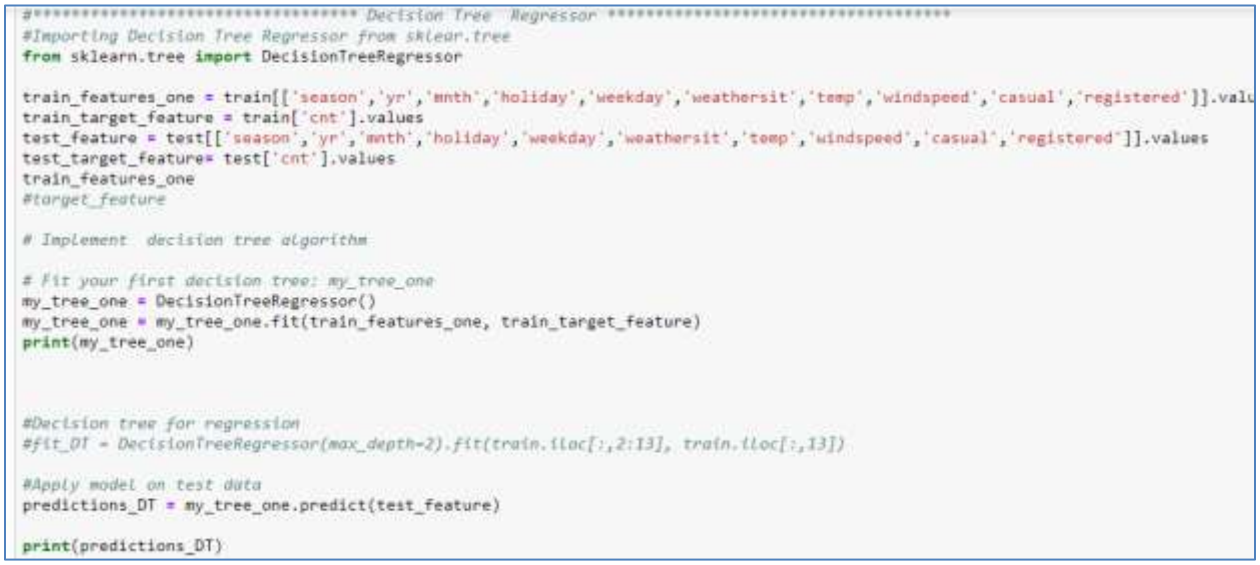
```
# check relationship with scatter plots

sns.set()
cols = ['temp', 'atemp', 'hum', 'windspeed', 'casual', 'registered', 'cnt']
sns.pairplot(day_numeric[cols], size = 2.5,kind="reg")
plt.show();

#As per scatter plots and above Correlation  graph there is strong relation
# Independent variable   'temp' and 'atemp'
# There is a   poor relation between  Independent variable 'hum' and dependent  variable 'cnt'

# so dropping two variables for feature selection
```



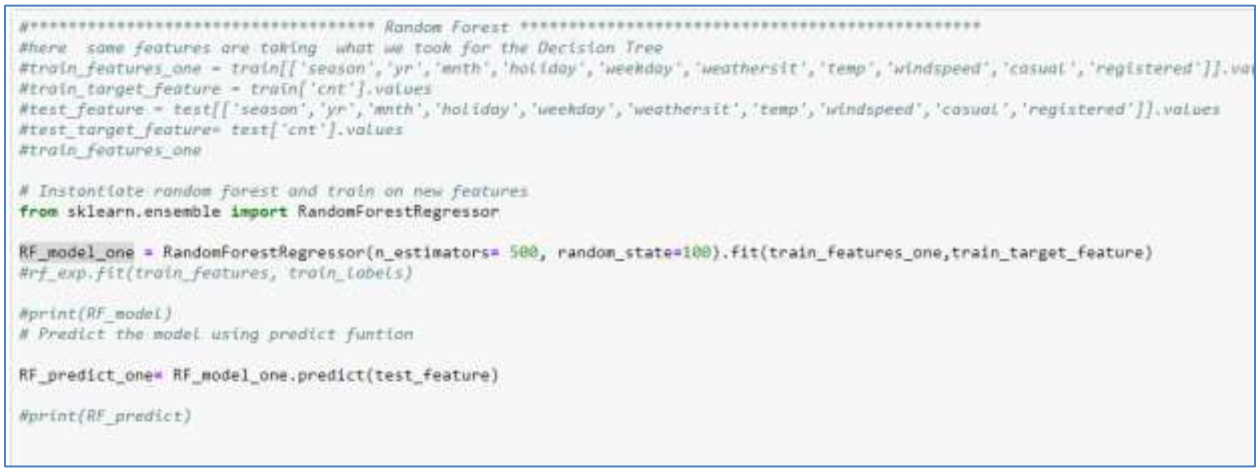
```
# feature  Scaling
#########################################  Normality  Check #########################################

cnames = ['casual','registered']

for i in cnames :
    print(i)
    df_day[i] = (df_day[i] - min(df_day[i]))/(max(df_day[i]) - min(df_day[i]))

df_day.head()
```

```
#now iam not checking  categorical feature importance i will check it later during tuning process

#Now  for variable  not doing  Data Scaling  this will do during tuning process
```

```
#dividing  Test and train data  using skilearn   train_test_split

df_day_feature_selection = df_day.drop(['atemp','hum'],axis = 1)
df_day_feature_selection.shape

from sklearn.model_selection import train_test_split

train, test = train_test_split(df_day_feature_selection, test_size=0.2)

#train_shape
```

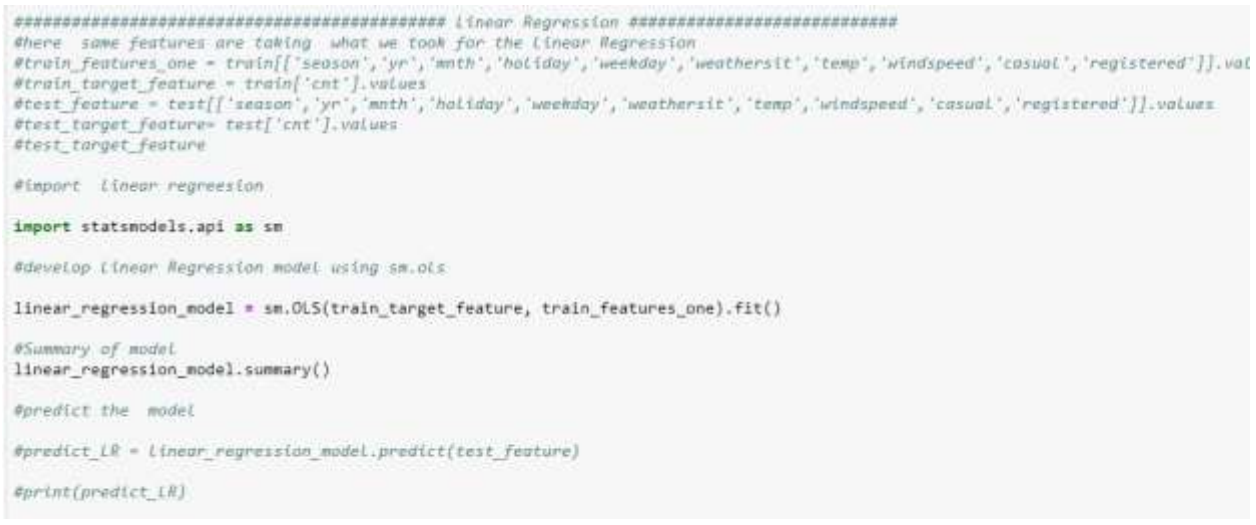


```python
#************************************* Decision Tree  Regressor **************************************
#Importing Decision Tree Regressor from sklear.tree
from sklearn.tree import DecisionTreeRegressor

train_features_one = train[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].valu
train_target_feature = train['cnt'].values
test_feature = test[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].values
test_target_feature= test['cnt'].values
train_features_one
#target_feature

# Implement  decision tree algorithm

# Fit your first decision tree: my_tree_one
my_tree_one = DecisionTreeRegressor()
my_tree_one = my_tree_one.fit(train_features_one, train_target_feature)
print(my_tree_one)



#Decision tree for regression
#fit_DT = DecisionTreeRegressor(max_depth=2).fit(train.iloc[:,2:13], train.iloc[:,13])

#Apply model on test data
predictions_DT = my_tree_one.predict(test_feature)

print(predictions_DT)
```

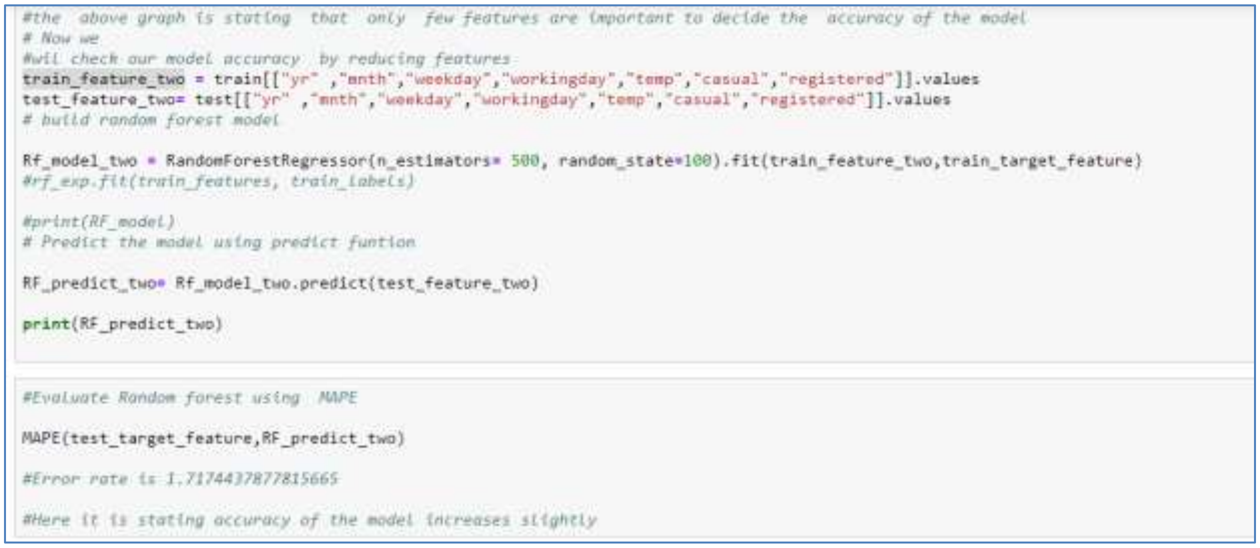


```python
#************************************** Random Forest ***************************************************
#here  same features are taking  what we took for the Decision Tree
#train_features_one = train[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].va
#train_target_feature = train['cnt'].values
#test_feature = test[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].values
#test_target_feature= test['cnt'].values
#train_features_one

# Instantiate random forest and train on new features
from sklearn.ensemble import RandomForestRegressor

RF_model_one = RandomForestRegressor(n_estimators= 500, random_state=100).fit(train_features_one,train_target_feature)
#rf_exp.fit(train_features, train_labels)

#print(RF_model)
# Predict the model using predict funtion

RF_predict_one= RF_model_one.predict(test_feature)

#print(RF_predict)
```

```python
#************************************** Random Forest ***************************************************
#here  same features are taking  what we took for the Decision Tree
#train_features_one = train[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].va
#train_target_feature = train['cnt'].values
#test_feature = test[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].values
#test_target_feature= test['cnt'].values
#train_features_one

# Instantiate random forest and train on new features
from sklearn.ensemble import RandomForestRegressor

RF_model_one = RandomForestRegressor(n_estimators= 500, random_state=100).fit(train_features_one,train_target_feature)
#rf_exp.fit(train_features, train_labels)

#print(RF_model)
# Predict the model using predict funtion

RF_predict_one= RF_model_one.predict(test_feature)

#print(RF_predict)
```

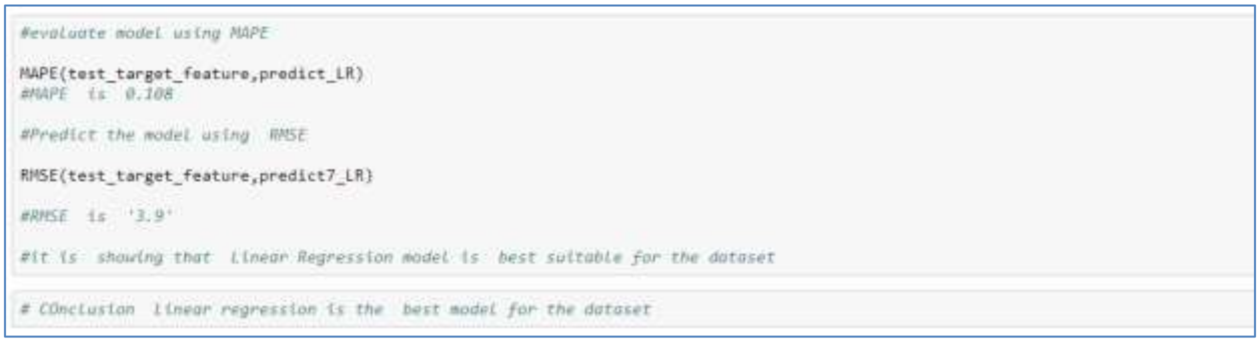


```
############################################# Linear Regression ############################
#here  same features are taking  what we took for the Linear Regression
#train_features_one = train[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].val
#train_target_feature = train['cnt'].values
#test_feature = test[['season','yr','mnth','holiday','weekday','weathersit','temp','windspeed','casual','registered']].values
#test_target_feature= test['cnt'].values
#test_target_feature

#import  Linear regreesion

import statsmodels.api as sm

#develop Linear Regression model using sm.ols

linear_regression_model = sm.OLS(train_target_feature, train_features_one).fit()

#Summary of model
linear_regression_model.summary()

#predict the  model

#predict_LR = linear_regression_model.predict(test_feature)

#print(predict_LR)
```



```
#the  above graph is stating  that  only  few features are important to decide the  accuracy of the model
# Now we
#wil check our model accuracy  by reducing features
train_feature_two = train[["yr" ,"mnth","weekday","workingday","temp","casual","registered"]].values
test_feature_two= test[["yr" ,"mnth","weekday","workingday","temp","casual","registered"]].values
# build random forest model

Rf_model_two = RandomForestRegressor(n_estimators= 500, random_state=100).fit(train_feature_two,train_target_feature)
#rf_exp.fit(train_features, train_labels)

#print(RF_model)
# Predict the model using predict funtion

RF_predict_two= Rf_model_two.predict(test_feature_two)

print(RF_predict_two)
```

```
#Evaluate Random forest using  MAPE

MAPE(test_target_feature,RF_predict_two)

#Error rate is 1.7174437877815665

#Here it is stating accuracy of the model increases slightly
```



```
#evaluate model using MAPE

MAPE(test_target_feature,predict_LR)
#MAPE  is  0.108

#Predict the model using  RMSE

RMSE(test_target_feature,predict7_LR)

#RMSE  is  '3.9'

#it is  showing that  Linear Regression model is  best suitable for the dataset
```

```
# COnclusion  linear regression is the  best model for the dataset
```