PREDICTING BIKE RENTAL COUNT

MUKHESH NARRA

Contents

1. INTRODUCTION

1.1 Problem statement

A bike rental is a bicycle business that rents bikes for short periods of time. Most rentals are provided by bike shops as a sideline to their main businesses of sales and service, but some shops specialize in rentals. Bike rental shops rent by the day or week as well as by the hour, and these provide an excellent opportunity for people who don't have access to a vehicle, typically travelers and particularly tourists. Specialized bike rental shops thus typically operate at beaches, parks, or other locations that tourists frequent. In this case, the fees are set to encourage renting the bikes for a few hours at a time, rarely more than a day. The objective of this Case is to predict the bike rental count based on the environmental and seasonal settings, So that required bikes would be arranged and managed by the shops according to environmental and seasonal conditions.

1.2 Data

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

Table for first 8 columns:

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

Table for next 8 columns:

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	0.200000	0.212122	0.590435	0.160296	108	1454	1562
1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

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

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

dteday: Date

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

yr: Year (0: 2011, 1:2012)

mnth: Month (1 to 12)

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

weekday: Day of the week

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

weathersit: (extracted fromFreemeteo)

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

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

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

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

casual: count of casual users

registered: count of registered users

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

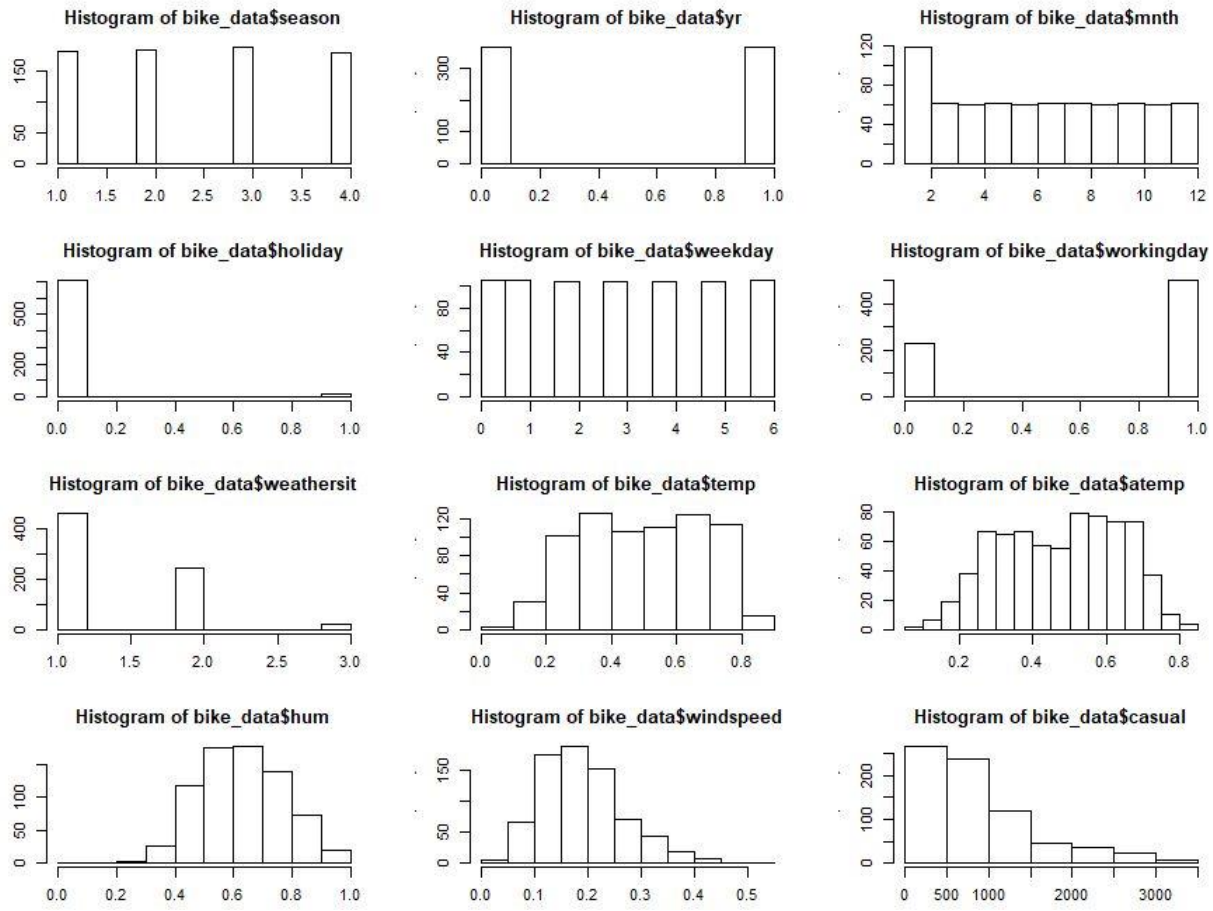
2. Methodology

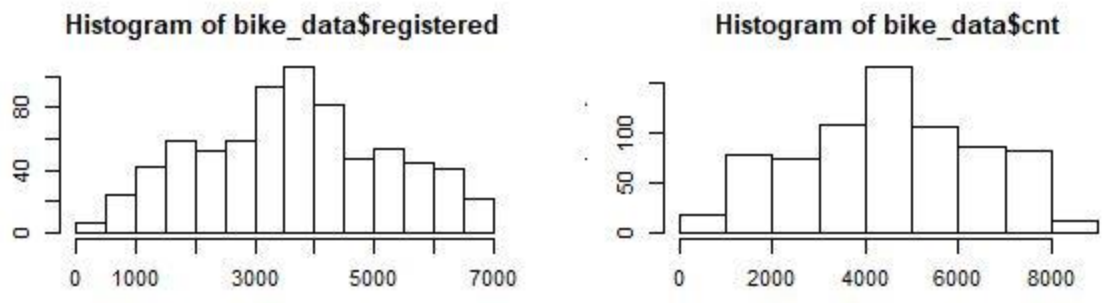
2.1 Pre Processing

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

2.1.1 Exploratory Data Analysis

Lets see the variable data distribution and plotting is seen below:





Important points to be noted from plot above:

1. Where here there are three dependent variables like casual, registered, cnt variable. Where sum of casual and registered will give cnt variable (Casual+registered=cnt).
2. As we can see casual variable is skewed where we can do transformation like square root, reciprocal, log which can make skewed data to normal distribution.
3. Where variables like season, yr, mnth, weekday, workingday, holiday, weathersit are to converted numerical into categorical.
4. Where dropping the instant variable where it is same as index which does'nt carry any information.

2.1.2 Missing Value Analysis:

Missing value analysis is done to check is there any missing value present in given dataset. Missing values can be easily treated using various methods like mean, median method, knn imputation method to impute missing value.

	Missing value count
dteday	0
season	0
yr	0
mnth	0
holiday	0
weekday	0
workingday	0
weathersit	0
temp	0
atemp	0
hum	0
windspeed	0
casual	0
registered	0
cnt	0

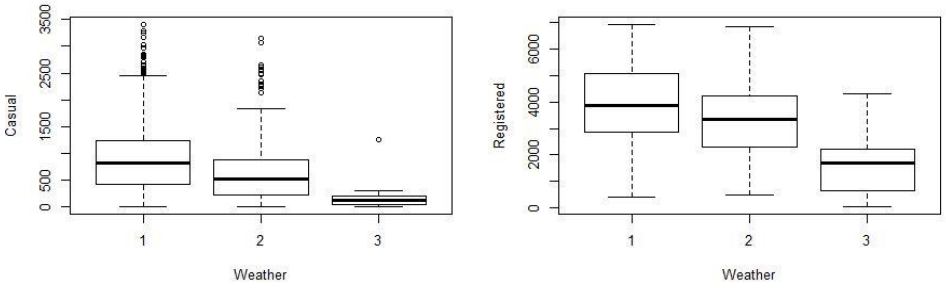
Where we can't see any missing values in above table.

2.1.3 Intuition about data:

- **Rain:** The demand of bikes will be lower on a rainy day as compared to a sunny day. Similarly, higher humidity will cause to lower the demand and vice versa.
- **Daily Trend:** Registered user will be demand more bike on weekdays compared to weekends or holiday.
- **Temperature:** In India, temperature has negative correlation with bike demand.(we can see in feature selection)

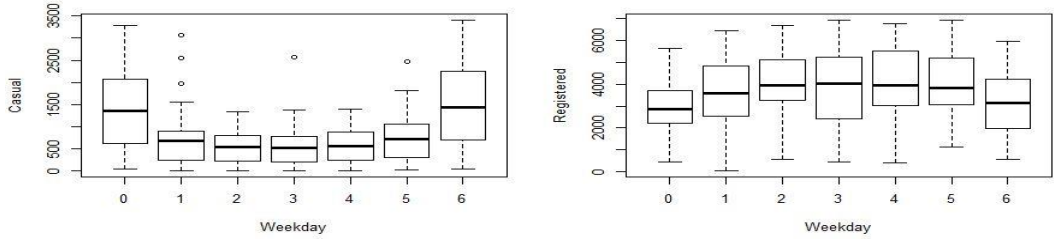
To prove above points let us plot the graphs:

Rain: We don't have the 'rain' variable with us but have 'weathersit' which is sufficient to test our intuition. As per variable description, weathersit 3 represents light rain and weathersit 4 represents heavy rain. Take a look at the plot:



Which says our intuition about data is correct where we see that no one is opting for bike when the weather is heavy rain.

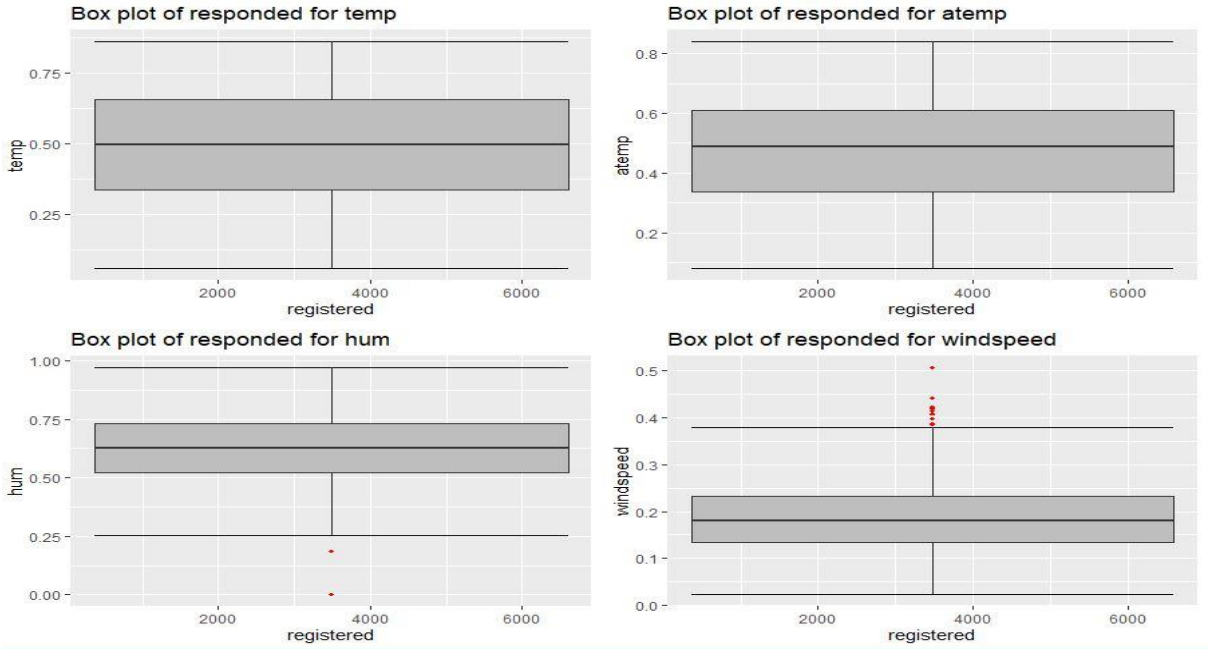
Daily Trend: Let us plot Shows casual and registered demand over days.

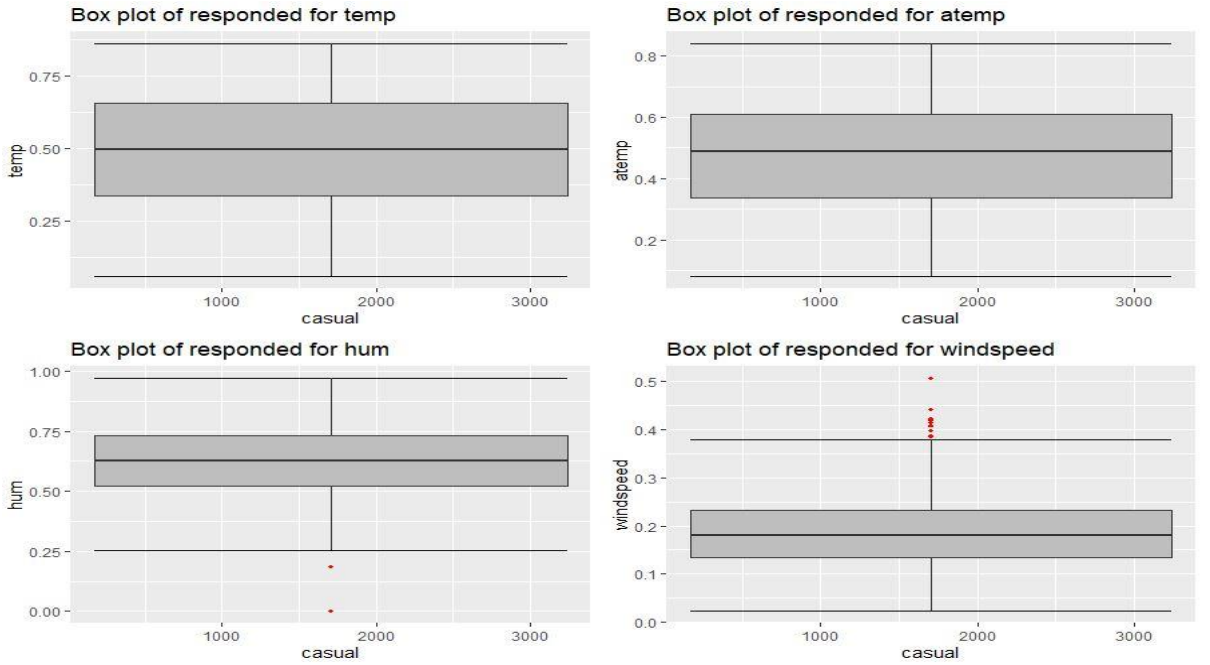


Intuition we can see that casual users are opting for bike rental in weekends and registered users opting for bike rental in weekdays.

2.1.4 Outer Analysis

Outlier analysis is done to handle all inconsistent observations present in given dataset. As outlier analysis can only be done on continuous variable. Figures are visualization of numeric variable present in our dataset to detect outliers using boxplot. Outliers will be detected with red points.





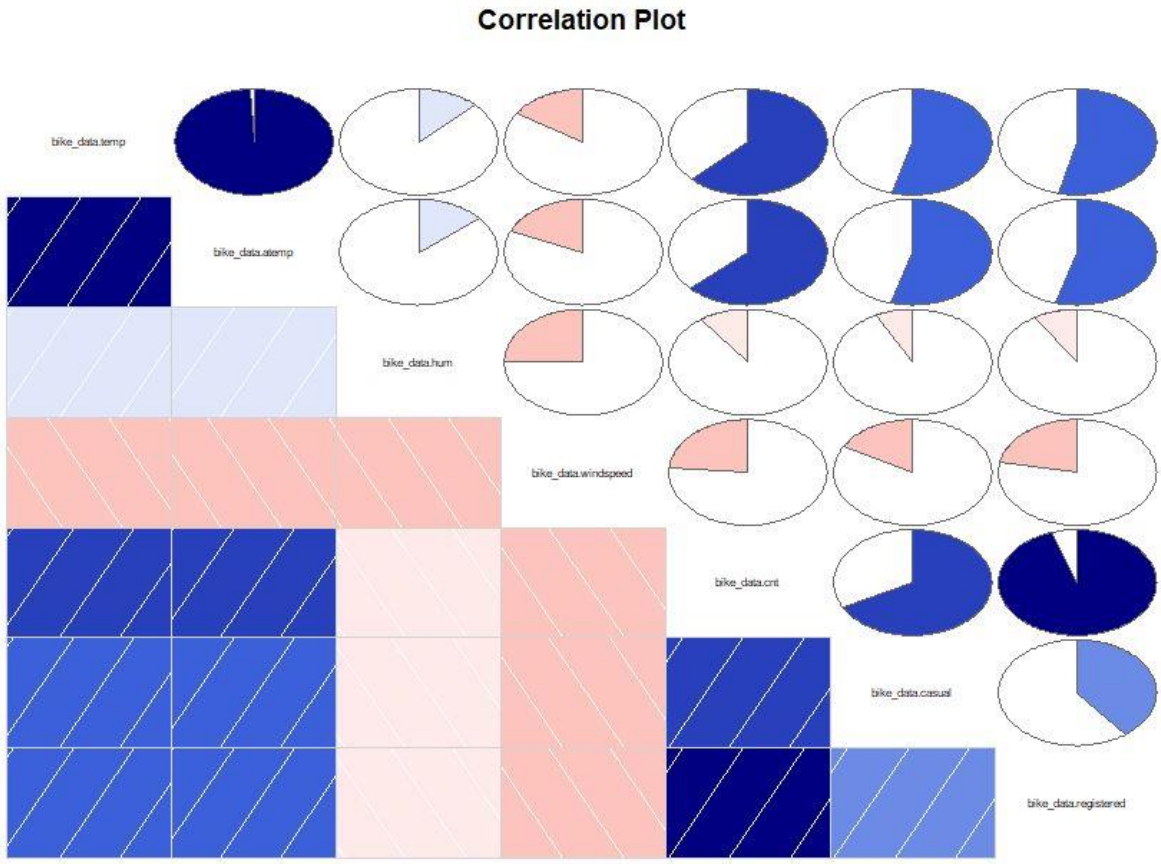
According to above visualizations there is no outlier found in temp and atemp variable but there are few outliers found in windspeed and hum variable. As windspeed variable defines the windspeed on a particular day and hum defines the humidity of that day so we can neglect these outliers because both these variable define environmental condition. Due to drastic change in weather like storm, heavy rain condition.

2.1.5 Feature Engineering

2.1.5.1 Feature Selection

Feature selection analysis is done to Select subsets of relevant features (variables, predictors) to be in model construction. As our target variable is continuous so we can only go for correlation check. As chi-square test is only for categorical variable.

Let us plot a correlation plot :



Where we can see that temp and atemp variable high correlation apart from casual and registered and count variables. So in temp and atemp variables we need to select one variable.

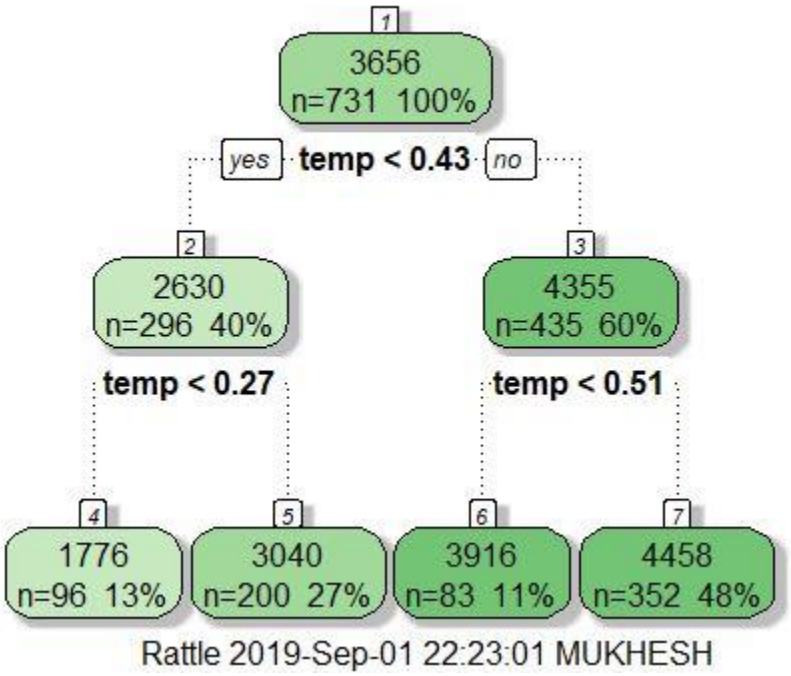
One point to be noted that we are predicting three dependent variables casual, registered, cnt. So we develop the model for casual and registered separately. And then sum the predicted value to get the cnt variable.

2.1.6 Feature Creation

In addition to existing independent variables, we will create new variables to improve the prediction power of model. Where there is increase in accuracy of model after a training without additional features.

Based on Intuition we have created bins for temperature for both registered and casuals users. Variables created are (temp_reg and temp_casual) by using decision Tree algorithm for splitting.

```
regr_DT_tree=rpart(registered~temp,data=bike_data)
fancyRpartPlot(regr_DT_tree)
```

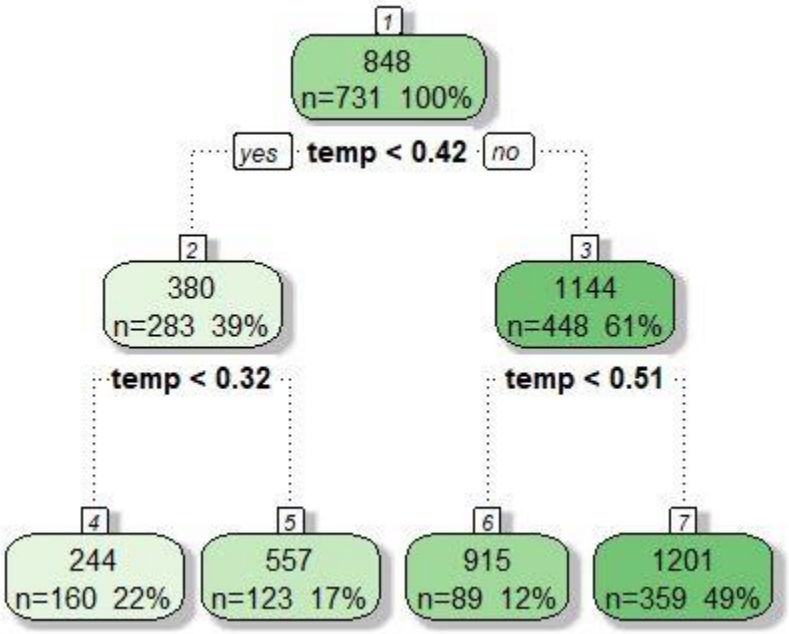


By analyzing above plot we can split the temp variable based on registered variable

```
bike_data$temp_regr[bike_data$temp<0.27]=1
bike_data$temp_regr[bike_data$temp>=0.51]=2
bike_data$temp_regr[bike_data$temp>=0.27 & bike_data$temp<0.43]=3
bike_data$temp_regr[bike_data$temp>=0.43 & bike_data$temp<0.51]=4
bike_data$temp_regr=as.factor(bike_data$temp_regr)
```

As same process we same thing for creating temp_casual

```
casual_DT_tree=rpart(casual~temp,data=bike_data)
fancyRpartPlot(casual_DT_tree)
bike_data$temp_casual[bike_data$temp<0.32]=1
bike_data$temp_casual[bike_data$temp>=0.51]=2
bike_data$temp_casual[bike_data$temp>=0.32 & bike_data$temp<0.42]=3
bike_data$temp_casual[bike_data$temp>=0.42 & bike_data$temp<0.51]=4
bike_data$temp_casual=as.factor(bike_data$temp_casual)
```



2.1.6 Feature Scaling

Feature scaling includes two functions normalization and standardization. It is done reduce unwanted variation either within or between variables and to bring all of the variables into proportion with one another. In given dataset all numeric values are already present in normalized form.

3. Modeling

3.1 Data Splitting

For Every model building we need train and test data so we will sample the data into train and test data in 80:20 ratio that means 80% of data is used as train data is used for training the model and 20% of data is used as test data to test the model for measuring the model accuracy.

```
#sampling where we spliting the data into train and test split in 80:20 ratio
train_index=sample(1:nrow(bike_data),0.8*nrow(bike_data))
train_data=bike_data[train_index,]
test_data=bike_data[-train_index,]
```

3.2 Model Selection

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

1. Decision Tree

2. Random Forest

3. Linear regression

Where we have three dependent variables casual, registered and cnt such that model is used to predict the two dependent variables (casual and registered) and sum of two predicted variables gives third dependent variables. And also, casual and registered variables are skewed so we apply the log transformation to casual and registered variables.

3.2.1 Decision Tree Algorithm

Here we are using the Decision Tree Algorithm for predicting the two variables casual and registered. Then we apply model to test data to predict the casual and registered variables and sum of predictions of casual and registered variable will give total count of bike rental.

Creating Model for train data as we spitted the data as we seen in data splitting section.
In R:

```
---------------------------------------#Decision Tree-----------------------------------------
#model to predict casual variable
casual_DT=rpart(log(casual)~dteday+season+yr+mnth+holiday+weekday+workingday+temp_casual+weathersit+atemp+hum+windspeed,data=train_data,method='anova')
summary(casual_DT)
predictions_casual_DT=predict(casual_DT,test_data[,-c(12,13,14,15)])
predictions_casual_DT=exp(predictions_casual_DT)

#model to predict registered variable
registerd_DT=rpart(log(registered)~dteday+season+yr+mnth+holiday+temp_regr+weekday+workingday+weathersit+atemp+hum+windspeed,data=train_data,method='anova')
summary(registerd_DT)
predictions_registered_DT=predict(registerd_DT,test_data[,-c(12,13,14,16)])
predictions_registered_DT=exp(predictions_registered_DT)

#sum of predictions of casual variable  and registered will be give predictions of total count
predictions_DT=predictions_casual_DT+predictions_registered_DT
```

And examining the summary of created model

```
rpart(formula = log(registered) ~ dteday + season + yr + mnth +
    holiday + temp_regr + weekday + workingday + weathersit +
    atemp + hum + windspeed, data = train_data, method = "anova")
  n= 584

          CP nsplit rel error    xerror      xstd
1  0.26412275      0 1.0000000 1.0016934 0.1438642
2  0.12080295      1 0.7358772 0.8248205 0.1473414
3  0.08371332      2 0.6150743 0.6553109 0.1589659
4  0.07168769      3 0.5313610 0.6047079 0.1588575
5  0.05274853      4 0.4596733 0.5739875 0.1593174
6  0.03611989      5 0.4069248 0.5967260 0.1600876
7  0.01870422      6 0.3708049 0.5789305 0.1604093
8  0.01712001      7 0.3521007 0.5620762 0.1649591
9  0.01565666      8 0.3349806 0.5482245 0.1649671
10 0.01454934     10 0.3036673 0.5438220 0.1649388
11 0.01047309     11 0.2891180 0.5384682 0.1652126
12 0.01000000     12 0.2786449 0.5373229 0.1649899

Variable importance
     atemp       mnth  temp_regr     season         yr     dteday  windspeed        hum    weekday workingday
        21         17         16         14         12          7          4          4          2          2
```

```
rpart(formula = log(casual) ~ dteday + season + yr + mnth + holiday +
    weekday + workingday + temp_casual + weathersit + atemp +
    hum + windspeed, data = train_data, method = "anova")
  n= 584

          CP nsplit rel error    xerror       xstd
1  0.40180031      0 1.0000000 1.0028134 0.07908949
2  0.10272449      1 0.5981997 0.6775073 0.07357489
3  0.09767670      2 0.4954752 0.5606403 0.06587557
4  0.04723731      3 0.3977985 0.4503476 0.06291656
5  0.04674776      4 0.3505612 0.4128612 0.06106889
6  0.02249965      5 0.3038134 0.3899402 0.06062380
7  0.01926251      6 0.2813138 0.3854310 0.06143185
8  0.01380050      7 0.2620513 0.4033824 0.06201258
9  0.01226676      8 0.2482508 0.3826973 0.04955097
10 0.01000000      9 0.2359840 0.3825658 0.04950857

Variable importance
      atemp temp_casual        mnth      season     weekday  workingday  weathersit      dteday
         26          21          18          11           8           7           2           2
        hum   windspeed          yr
          2           1           1
```

As we can see the both predicted values are stored in predicted_DT variable.

3.2.2 Random Forest Algorithm

Here we are using the Random Forest Algorithm for predicting the two variables casual and registered. Then we apply model to test data to predict the casual and registered variables and sum of predictions of casual and registered variable will give total count of bike rental.

Creating Model for train data as we spitted the data as we seen in data splitting section.
In R:

```
------------------------------#RandomForrest#-------------------------------------------------
#model to predict casual variable
casual_RF=randomForest(log(casual)~dteday+season+yr+mnth+holiday+temp_casual+weekday+workingday+weathersit+atemp+hum+windspeed,data=train_data)
predictions_casual_RF=predict(casual_RF,test_data[,-c(12,13,14,15)])
predictions_casual_RF=exp(predictions_casual_RF)

#model to predict registered variable
registerd_RF=randomForest(log(registered)~dteday+season+yr+mnth+holiday+temp_regr+weekday+workingday+weathersit+atemp+hum+windspeed,data=train_data)
predictions_registerd_RF=predict(registerd_RF,test_data[,-c(12,13,14,16)])
predictions_registerd_RF=exp(predictions_registerd_RF)

#sum of predictions of casual variable  and registered will be give predictions of total count
predictions_RF=predictions_casual_RF+predictions_registerd_RF

#we can visualize the rules of random forrest for casual  variable
treelist_casual=RF2List(casual_RF)
Rules_casual=extractRules(treelist_casual,train_data[,-c(12,13,14,15)])
readableRules_casual=presentRules(Rules_casual,colnames(train_data))

#we can visualize the rules of random forrest for casual  variable
treelist_registered=RF2List(registered_RF)
Rules_registered=extractRules(treelist_registered,train_data[,-c(12,13,14,16)])
readableRules_registered=presentRules(Rules_registered,colnames(train_data))
```

Where we stored the rules of both casual and registered variables into readableRules_casual, readableRules_registered variables.

3.2.2 Linear Regression

Here we are using the Random Forrest Algorithm for predicting the two variables casual and registered. Then we apply model to test data to predict the casual and registered variables and sum of predictions of casual and registered variable will give total count of bike rental.

Where for linear regression category variable should be converted into numerical for weights calculation for that we need create to dummy variables for more than two category variable.

Note:

In R programming library is smart enough to create detect the categories and create dummy variables. But in python programming before passing the data into model we need to create the dummy variables by get_dummies function in pandas library.

```
#where for categorical variable dummy variables are created
#where creating dummy variables for categorical variables to be used in linea
#taking category variable in variable
catnames=['dteday','season','yr','mnth','holiday','weekday','workingday','wea

cnames.remove('temp')
cnames=['casual','registered','cnt']+cnames
bike_linear_regression=bike_data[cnames]
#creating dummy variables
for i in catnames:
    temp=pd.get_dummies(bike_data[i],prefix=i)
    bike_linear_regression=bike_linear_regression.join(temp)

#sampling the data into train and test data
```

Creating Model for train data as we spitted the data as we seen in data splitting section.
In R:

```
------------------------------#linear regression-----------------------------------------------------
#model to predict casual variable
#where for categorical variable dummy variables are created
casual_lg=lm(log(casual)~dteday+season+yr+mnth+holiday+weekday+workingday+weathersit+temp_casual+atemp+hum+windspeed,data=train_data)
predictions_casual_lg=predict(casual_lg,test_data[,-c(12,13,14,15)])
predictions_casual_lg=exp(predictions_casual_lg)
summary(casual_lg)

#model to predict registered variable
registerd_lg=lm(log(registered)~dteday+season+yr+mnth+holiday+weekday+temp_regr+workingday+weathersit+atemp+hum+windspeed,data=train_data)
predictions_registerd_lg=predict(registerd_lg,test_data[,-c(12,13,14,16)])
predictions_registerd_lg=exp(predictions_registerd_lg)
summary(registered_lg)
predictions_lg=predictions_casual_lg+predictions_registerd_lg
```

And examining the summary of created model for casual

```
Call:
lm(formula = log(casual) ~ dteday + season + yr + mnth + holiday +
    weekday + workingday + weathersit + temp_casual + atemp +
    hum + windspeed, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-3.6081 -0.1807  0.0318  0.2022  1.2696

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.752416   0.192403  29.898  < 2e-16 ***
dteday02     0.001228   0.132457   0.009 0.992609
dteday03     0.109554   0.136456   0.803 0.422425
dteday04     0.069109   0.133352   0.518 0.604505
dteday05     0.066210   0.131120   0.505 0.613800
dteday06     0.125688   0.132121   0.951 0.341885
dteday07     0.027591   0.131200   0.210 0.833520
dteday08    -0.077418   0.130890  -0.591 0.554457
dteday09    -0.014471   0.134347  -0.108 0.914266
dteday10    -0.066870   0.131423  -0.509 0.611099
dteday11     0.107612   0.137064   0.785 0.432741
dteday12    -0.132406   0.136914  -0.967 0.333951
dteday13    -0.039603   0.144096  -0.275 0.783551
dteday14     0.036751   0.133947   0.274 0.783907
dteday15     0.007951   0.132164   0.060 0.952051
dteday16    -0.032684   0.133330  -0.245 0.806448
dteday17     0.140668   0.156776   0.897 0.369997
dteday18    -0.075030   0.137437  -0.546 0.585351
dteday19     0.105197   0.129933   0.810 0.418524
dteday20     0.064550   0.131376   0.491 0.623393
dteday21     0.101497   0.136761   0.742 0.458332
dteday22    -0.101180   0.145029  -0.698 0.485704
dteday23     0.004342   0.132853   0.033 0.973940
dteday24    -0.037574   0.134380  -0.280 0.779886
dteday25     0.141681   0.136591   1.037 0.300092
dteday26     0.041997   0.137706   0.305 0.760504
dteday27    -0.126807   0.134752  -0.941 0.347118
dteday28    -0.008614   0.134801  -0.064 0.949076
dteday29    -0.378477   0.133252  -2.840 0.004683 **
dteday30    -0.114824   0.138865  -0.827 0.408685
dteday31     0.178932   0.150938   1.185 0.236370
season2      0.398356   0.107140   3.718 0.000222 ***
season3      0.367756   0.131252   2.802 0.005268 **
season4      0.263146   0.111500   2.360 0.018638 *
yr1          0.372651   0.035272  10.565  < 2e-16 ***
mnth2        0.284113   0.086877   3.270 0.001145 **
mnth3        0.807138   0.100720   8.014 7.30e-15 ***
mnth4        0.699589   0.148384   4.715 3.11e-06 ***
mnth5        0.606391   0.161821   3.747 0.000199 ***
mnth6        0.416665   0.168171   2.478 0.013540 *
mnth7        0.358452   0.187773   1.909 0.056814 .
mnth8        0.465417   0.181361   2.566 0.010558 *
mnth9        0.564493   0.165063   3.420 0.000675 ***
mnth10       0.645714   0.151475   4.263 2.39e-05 ***
```

```
mnth11          0.675389    0.143553    4.705 3.26e-06 ***
mnth12          0.461111    0.111481    4.136 4.11e-05 ***
holiday1        0.710469    0.114780    6.190 1.22e-09 ***
weekday1       -0.845574    0.065246  -12.960   < 2e-16 ***
weekday2       -0.905230    0.063698  -14.211   < 2e-16 ***
weekday3       -0.925455    0.064388  -14.373   < 2e-16 ***
weekday4       -0.898691    0.064992  -13.828   < 2e-16 ***
weekday5       -0.639653    0.064038   -9.989   < 2e-16 ***
weekday6        0.098582    0.064226    1.535 0.125404
workingday1           NA          NA       NA       NA
weathersit2    -0.178895    0.048541   -3.685 0.000252 ***
weathersit3    -1.343998    0.120175  -11.184   < 2e-16 ***
temp_casual2    0.864790    0.146513    5.902 6.44e-09 ***
temp_casual3    0.344280    0.078157    4.405 1.28e-05 ***
temp_casual4    0.626414    0.109916    5.699 2.02e-08 ***
atemp           1.311646    0.410549    3.195 0.001483 **
hum            -0.953113    0.177817   -5.360 1.25e-07 ***
windspeed      -1.475822    0.253960   -5.811 1.08e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4143 on 523 degrees of freedom
Multiple R-squared:  0.8619,    Adjusted R-squared:  0.8461
F-statistic: 54.42 on 60 and 523 DF,  p-value: < 2.2e-16
```

Where F Statistic gives us a power to judge whether that relationship is statistically significant. Here it is p-value is less than 0.05 we can eliminate null hypothesis which means that adding the temp_casual variable show model fit well.

And examining the summary of created model for registered.

```
Call:
lm(formula = log(registered) ~ dteday + season + yr + mnth +
    holiday + weekday + temp_regr + workingday + weathersit +
    atemp + hum + windspeed, data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-3.9468 -0.1049   0.0134   0.1343   0.9434

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.240844   0.132010  54.851   < 2e-16 ***
dteday02     0.006159   0.093157   0.066 0.947314
dteday03     0.070118   0.096146   0.729 0.466152
dteday04     0.110085   0.094235   1.168 0.243259
dteday05     0.101327   0.092638   1.094 0.274551
dteday06     0.119168   0.093067   1.280 0.200952
dteday07     0.041593   0.092426   0.450 0.652887
dteday08     0.027151   0.092387   0.294 0.768965
dteday09     0.104517   0.094708   1.104 0.270287
dteday10     0.061464   0.092546   0.664 0.506892
dteday11     0.143387   0.096605   1.484 0.138342
dteday12     0.042214   0.096464   0.438 0.661847
dteday13     0.036773   0.101559   0.362 0.717439
dteday14     0.054826   0.094485   0.580 0.561988
dteday15     0.094912   0.092992   1.021 0.307894
dteday16     0.101295   0.094060   1.077 0.282013
dteday17     0.139862   0.110003   1.271 0.204137
dteday18     0.001446   0.096939   0.015 0.988105
dteday19     0.064024   0.091579   0.699 0.484792
dteday20     0.090040   0.092535   0.973 0.330987
dteday21     0.136368   0.096159   1.418 0.156741
dteday22     0.084120   0.102055   0.824 0.410168
dteday23    -0.060866   0.093540  -0.651 0.515531
dteday24    -0.079392   0.094697  -0.838 0.402202
dteday25    -0.087061   0.096302  -0.904 0.366390
dteday26    -0.007281   0.097146  -0.075 0.940287
dteday27    -0.092921   0.094973  -0.978 0.328330
dteday28     0.056779   0.095028   0.598 0.550431
dteday29    -0.278500   0.093991  -2.963 0.003185 **
dteday30    -0.005509   0.097835  -0.056 0.955116
dteday31     0.219550   0.106581   2.060 0.039900 *
season2      0.271030   0.075377   3.596 0.000354 ***
season3      0.457146   0.092184   4.959 9.59e-07 ***
season4      0.701080   0.078788   8.898   < 2e-16 ***
yr1          0.484378   0.024897  19.455   < 2e-16 ***
mnth2        0.129742   0.062159   2.087 0.037350 *
mnth3        0.067686   0.071805   0.943 0.346306
mnth4       -0.006529   0.105096  -0.062 0.950484
mnth5        0.005606   0.114500   0.049 0.960971
mnth6       -0.037305   0.118623  -0.314 0.753283
mnth7       -0.236996   0.132222  -1.792 0.073645 .
mnth8       -0.193379   0.128004  -1.511 0.131462
mnth9       -0.117802   0.116732  -1.009 0.313362
mnth10      -0.369656   0.107020  -3.454 0.000597 ***
```

```
mnth11       -0.160148   0.101634  -1.576 0.115693
mnth12       -0.136207   0.079639  -1.710 0.087800 .
holiday1     -0.357891   0.080773  -4.431 1.14e-05 ***
weekday1      0.231037   0.046082   5.014 7.33e-07 ***
weekday2      0.305507   0.044821   6.816 2.58e-11 ***
weekday3      0.338521   0.045325   7.469 3.42e-13 ***
weekday4      0.347337   0.045876   7.571 1.68e-13 ***
weekday5      0.298605   0.045114   6.619 8.98e-11 ***
weekday6      0.077479   0.045166   1.715 0.086862 .
temp_regr2    0.561045   0.111296   5.041 6.39e-07 ***
temp_regr3    0.207318   0.058705   3.532 0.000450 ***
temp_regr4    0.407461   0.089514   4.552 6.62e-06 ***
workingday1         NA         NA      NA       NA
weathersit2  -0.082242   0.034255  -2.401 0.016703 *
weathersit3  -0.988971   0.085058 -11.627  < 2e-16 ***
atemp         0.283311   0.280331   1.011 0.312660
hum          -0.386349   0.125195  -3.086 0.002136 **
windspeed    -0.752634   0.179378  -4.196 3.19e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2919 on 523 degrees of freedom
Multiple R-squared:  0.7736,    Adjusted R-squared:  0.7477
F-statistic: 29.79 on 60 and 523 DF,  p-value: < 2.2e-16
```

Where F Statistic gives us a power to judge whether that relationship is statistically significant.Here it is p-value is less than 0.05 which means that we can eliminate null hypothesis which means adding the temp_regr variable show model fit well.

3.3 Model Evaluation

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

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

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

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

3.3.1 Mean Absolute Percentage Error (MAPE)

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

MAPE function in R:

```
#function for evaluation
mape=function(y,y_pred)
{
  mean(abs(y-y_pred)/y)*100
}
```

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

Model error percentages in R:

```
> mape(test_data[,14],predictions_RF)#-
[1] 17.80648
> mape(test_data[,14],predictions_lg)#-
[1] 17.86032
> mape(test_data[,14],predictions_DT)#-
[1] 26.09761
```

Model error percentages in python:

```
In [27]: print(mape(test_data.iloc[:,13],predictions_RF))
13.39479198313368

In [28]: print(mape(test_data.iloc[:,13],predictions_lg))
17.293919437985387

In [29]: print(mape(test_data.iloc[:,13],predictions_DT))
18.288430772238005
```

Where
predictions_DT are predicted values from decision tree model,
predictions_RF are predicted values from random forest model,
predictions_lg are predicted values from linear regression model

3.4 Model Selection

We can see that from both R and Python Random forest model performs best out of decision tree and linear regression. So random forest model is selected with 82.2% accuracy in R and with 86.7% accuracy in python.

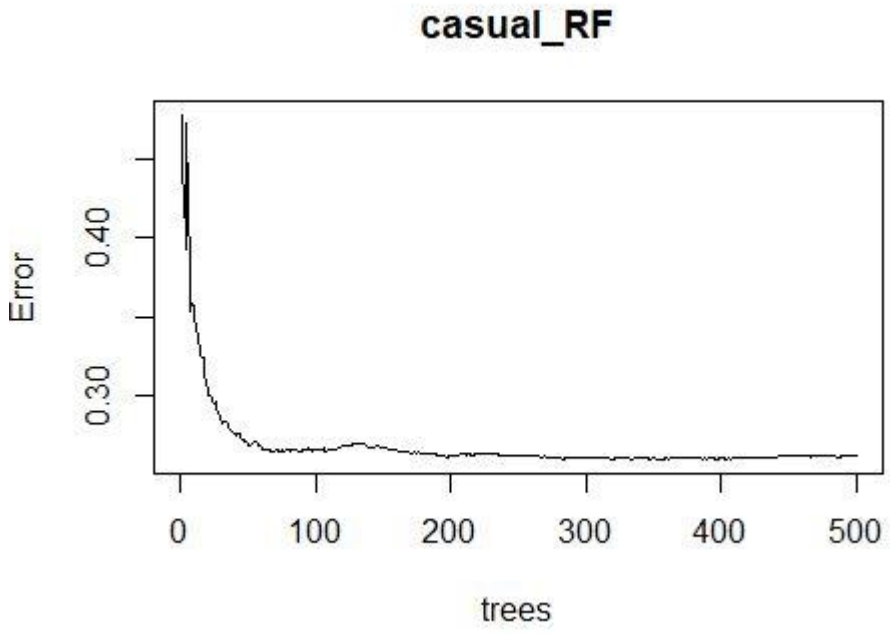
3.5 Hyper-parameter Tuning

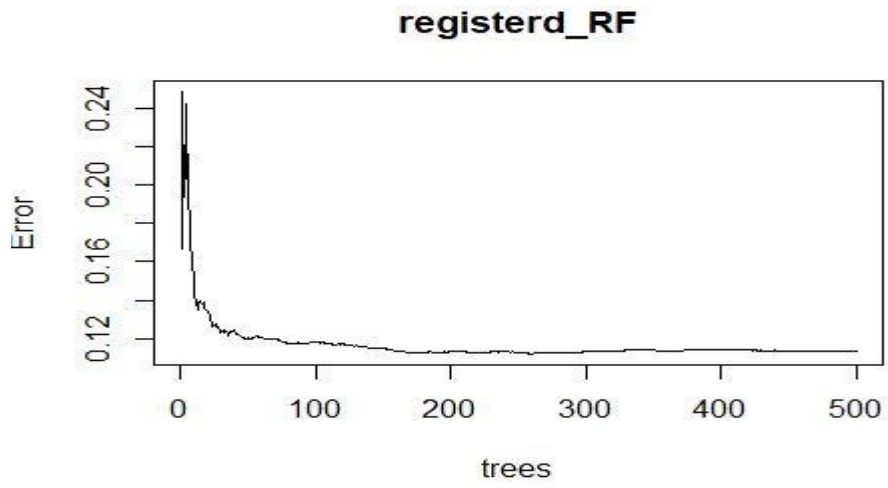
Where we have selected the random forest model but we can tune the random forest parameters for increasing the accuracy of the model.

3.5.1 Hyperparameter tuning in R:

Where basically two parameters are significant
1.ntree (number of trees to construct for rules making)
2.mtry(no. of variables for random split)

Ntree:
For ntree we can get optimal value by plotting





While analyzing the plots we can see that optimal value is 200

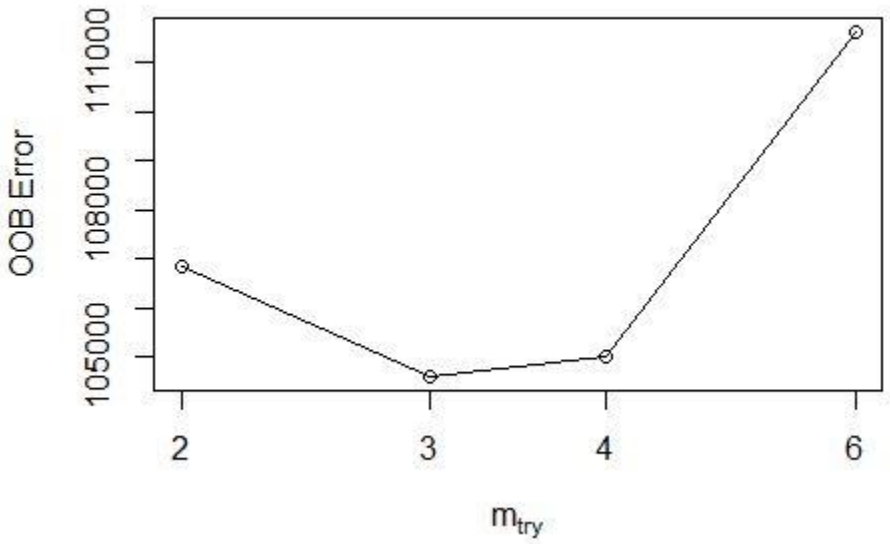
Mtry:

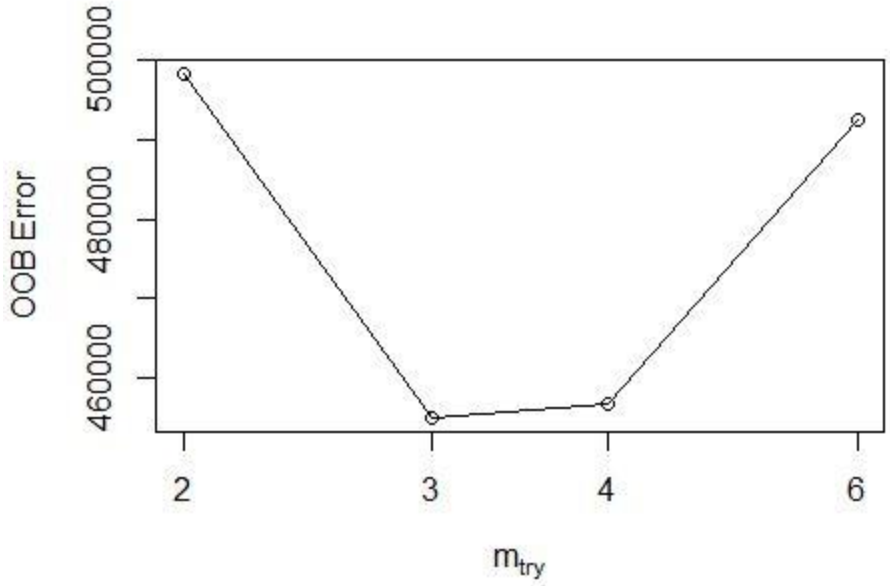
For mtry we can get optimal value by tuneRF library

```
x_casual<-train_data[,-c(12,13,14,16)]
y_casual<-train_data[,12]
bmtry<-tuneRF(x_casual,y_casual,ntreeTry = 200,stepFactor = 1.5, improve = 1e-5)
bmtry
#as we for mtry variables for random split is 3 where oob error is low

x_regr<-train_data[,-c(12,13,14,15)]
y_regr<-train_data[,13]
bmtry<-tuneRF(x_regr,y_regr,ntreeTry = 500,stepFactor = 1.5, improve = 1e-5)
bmtry
#as we for mtry variables for random split is 3 where oob error is low
```

As we can get optimal value by analyzing the plots





By analyzing the plots we can optimal value is 3.

Again we are training the model and measuring the accuracy

```
#train again and check if accuarcy is increase or not

#model to predict casual variable
casual_RF=randomForest(log(casual)~dteday+season+yr+mnth+holiday+temp_casual+weekday+workingday+weathersit+atemp+hum+windspeed,data=train_data,mtry=3,ntree=200)
predictions_casual_RF=predict(casual_RF,test_data[,-c(12,13,14,15)])
predictions_casual_RF=exp(predictions_casual_RF)

#model to predict registered variable
registerd_RF=randomForest(log(registered)~dteday+season+yr+mnth+holiday+temp_regr+weekday+workingday+weathersit+atemp+hum+windspeed,data=train_data,mtry=3,ntree=200)
predictions_registerd_RF=predict(registerd_RF,test_data[,-c(12,13,14,16)])
predictions_registerd_RF=exp(predictions_registerd_RF)

#sum of predictions of casual variable  and registered will be give predictions of total count
predictions_RF=predictions_casual_RF+predictions_registerd_RF

mape(test_data[,14],predictions_RF)
```

```
> mape(test_data[,14],predictions_RF)#
[1] 17.71964
```

while comparing with previous training we get the accuracy of 82.39% where comparing between previous one gives minute change.

So, we are finalizing this model.

Extracted predicted value of random forest model are saved with .csv file format.

3.5.1 Hyperparameter tuning in Python:

In python, we will use GridSearchCv for hyperparameter tuning where we are using two parameters max_features and n_estimators to tune.

Let see best parameters of model for predicting the casual variable

```
#-------------Hyperparameter Tuning-----------------
#creating variable for options for hyperparameter tuning
parameters={'max_features':[1,2,3,4,5,6,7,8,9,10],'n_estimators':[50,100,150,200,250,300,350]}
#creating object for getting the best parameters
grid=GridSearchCV(RFT(),param_grid=parameters,cv=10)
#training for getting best parameters for casual variable
grid.fit(train_data.iloc[:,[0,1,2,3,4,5,6,7,8,9,10,15]],train_data.iloc[:,11])

print(grid.best_score_)
#printing the best parameters
print(grid.best_params_)

#creating object for getting the best parameters
```

As we got below values for best parameters

```
In [46]: print(grid.best_score_)
0.8241661363138095

In [47]: print(grid.best_params_)
{'max_features': 6, 'n_estimators': 250}
```

Let see best parameters of model for predicting the casual variable

```
#creating object for getting the best parameters
grid=GridSearchCV(RFT(),param_grid=parameters,cv=10)
#training for getting best parameters for registered variable
grid.fit(train_data.iloc[:,[0,1,2,3,4,5,6,7,8,9,10,14]],train_data.iloc[:,12])

print(grid.best_score_)
#printing the best parameters
print(grid.best_params_)
```

As we got below values for best parameters

```
In [51]: print(grid.best_score_)
0.7873007085335708

In [52]: print(grid.best_params_)
{'max_features': 4, 'n_estimators': 100}
```

So, finally we got best parameters for two models then let us test the two model and get accuracy for comparing with previous model.

```
#train again and check if accuarcy is increase or not
#model to predict casual variable
casual_RF=RFT(max_features=6, n_estimators= 250).fit(train_data.iloc[:,[0,1,2,3,4,5,6,7,8,9,10,15]],train_data.iloc[:,11])
predictions_casual_RF=casual_RF.predict(test_data.iloc[:,[0,1,2,3,4,5,6,7,8,9,10,15]])
predictions_casual_RF=np.exp(predictions_casual_RF)

#model to predict registered variable
registerd_RF=RFT(max_features=4, n_estimators= 100).fit(train_data.iloc[:,[0,1,2,3,4,5,6,7,8,9,10,14]],train_data.iloc[:,12])
predictions_registerd_RF=registerd_RF.predict(test_data.iloc[:,[0,1,2,3,4,5,6,7,8,9,10,14]])
predictions_registerd_RF=np.exp(predictions_registerd_RF)

#sum of predictions of casual variable  and registered will be give predictions of total count
predictions_RF=predictions_casual_RF+predictions_registerd_RF

#printing the mape value
print(mape(test_data.iloc[:,13],predictions_RF))#-->12.51
#where we can see there is increase in accuracy
```

Lets see the results of MAPE to compare with previous one

```
In [54]: print(mape(test_data.iloc[:,13],predictions_RF))
12.517089122704139
```

while comparing with previous training we get the accuracy of 87.59% where comparing between previous one got good increase in accuracy.

So, we are finalizing this model.

Extracted predicted value of random forest model are saved with .csv file format.

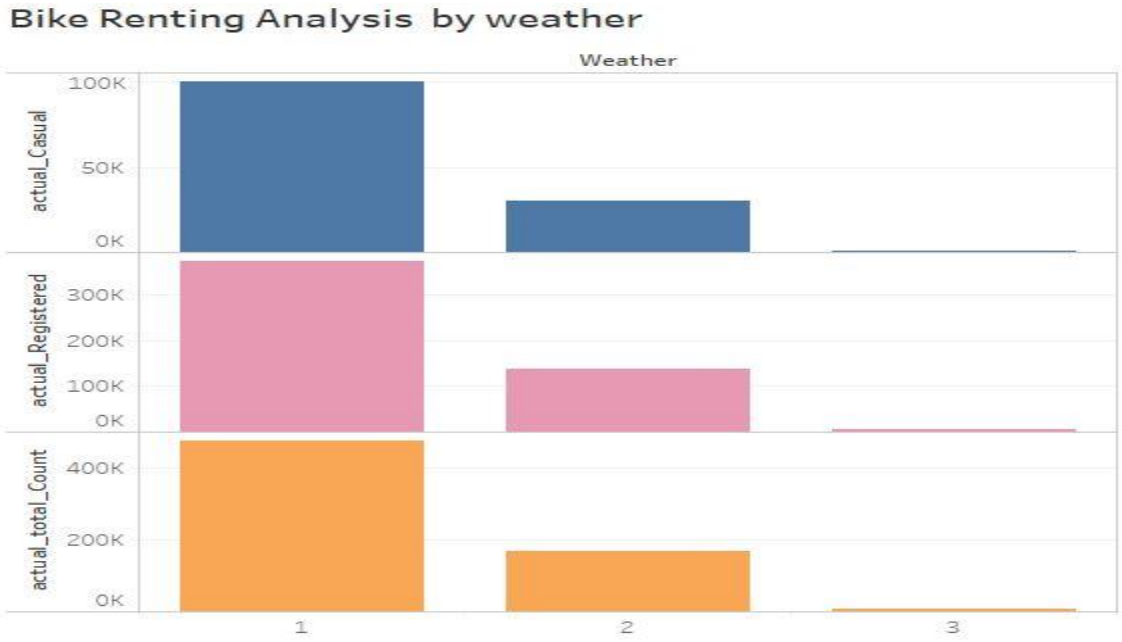
4. Visualizations

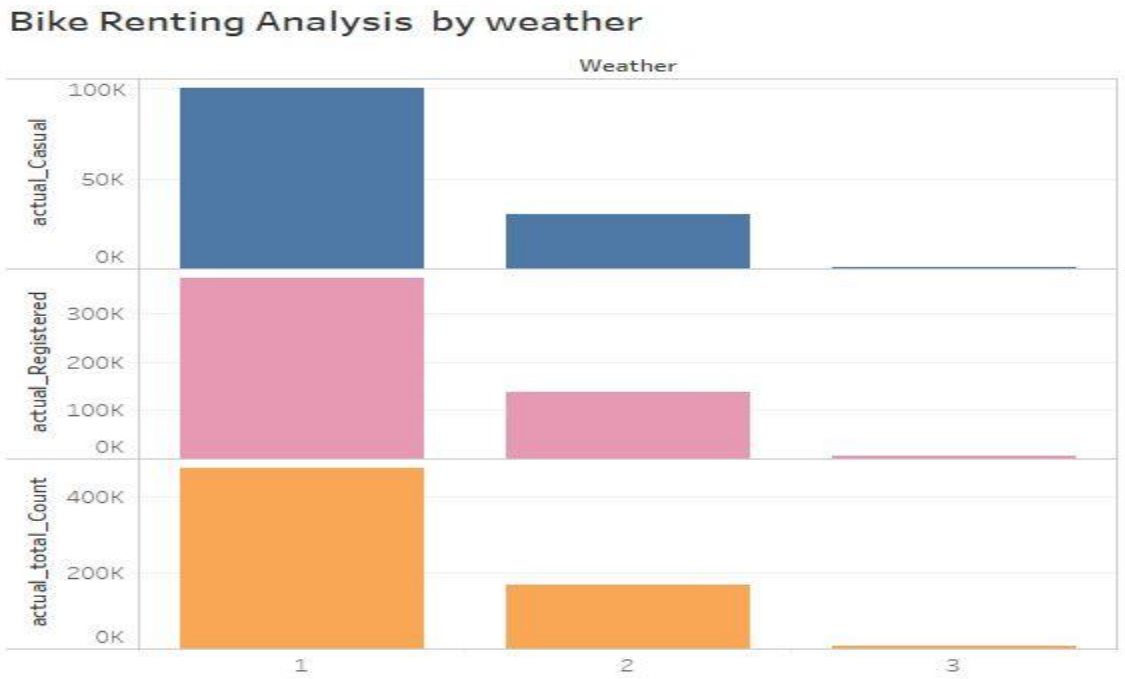
Where Visualization is done by the tableau where while importing data from excel all variables are treated as numeric in tableau so converted them into categorical variables.

4.1 Visualization on users by seasonal condition

season: Season (1:springer, 2:summer, 3:fall, 4:winter)
Above two bar graph represents the comparison of predicted count value and actual count value for casual and registered users also total users based on seasonal condition.

4.2 Visualization on users by weather conditions





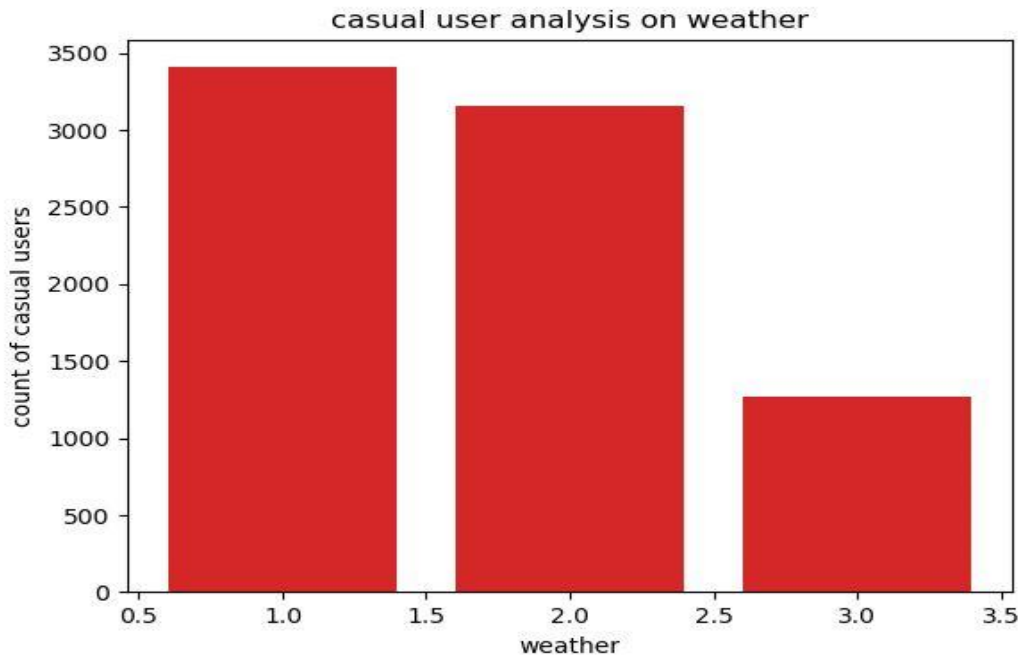
1: Clear, Few clouds, Partly cloudy, Partly cloudy 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

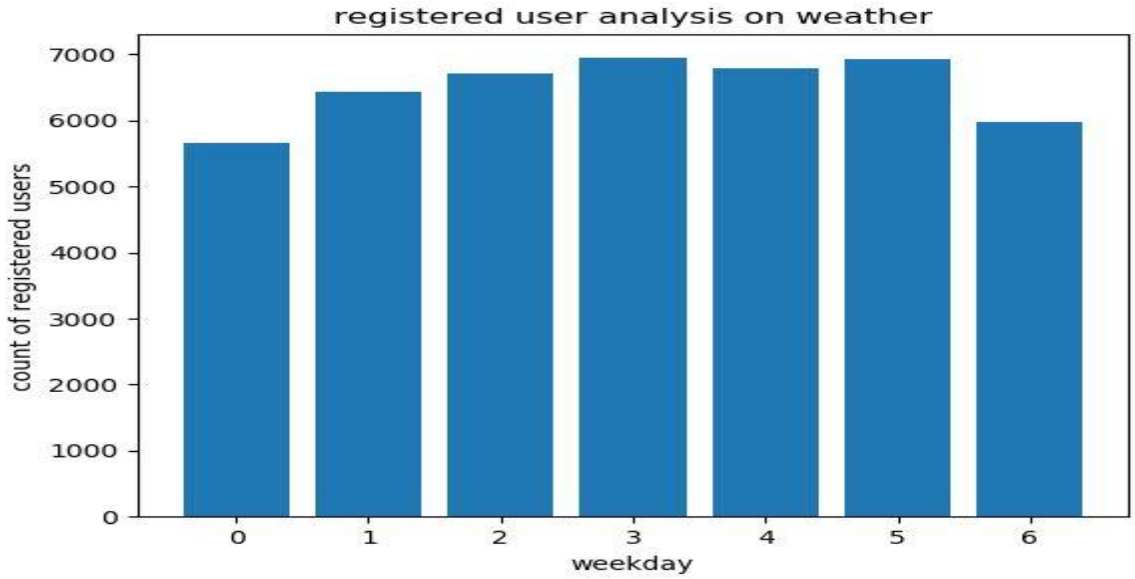
Above two bar graph represents the comparison of predicted count value and actual count value for casual and registered users also total users based on seasonal condition.

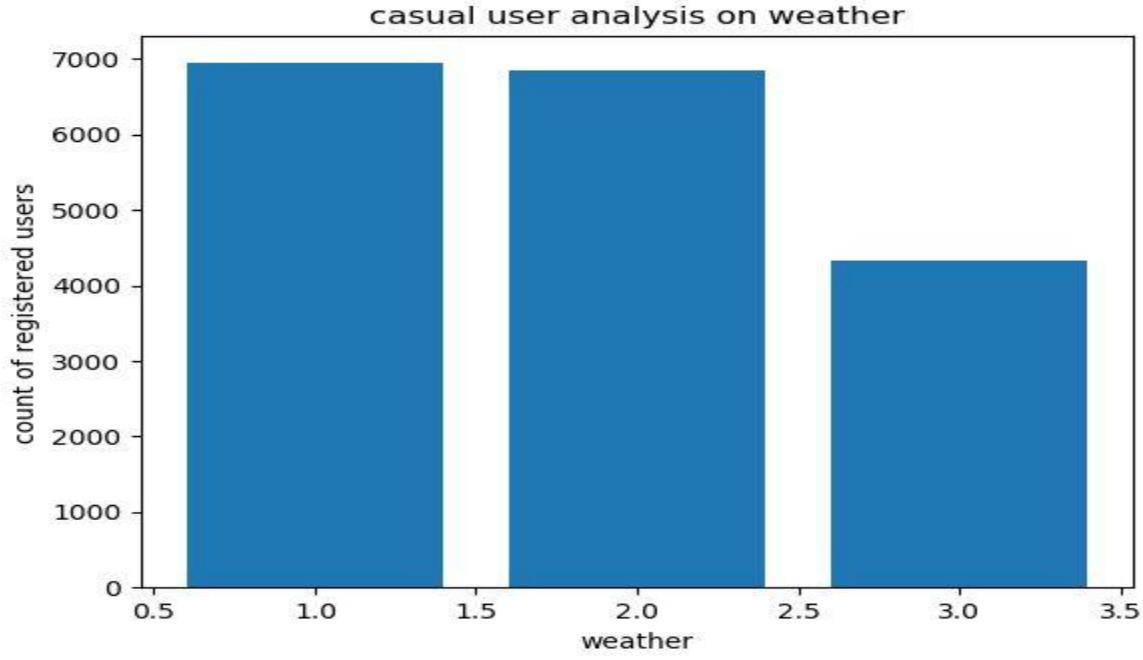
According to Seasonal and Weather Condition based on test data created in data splitting section,

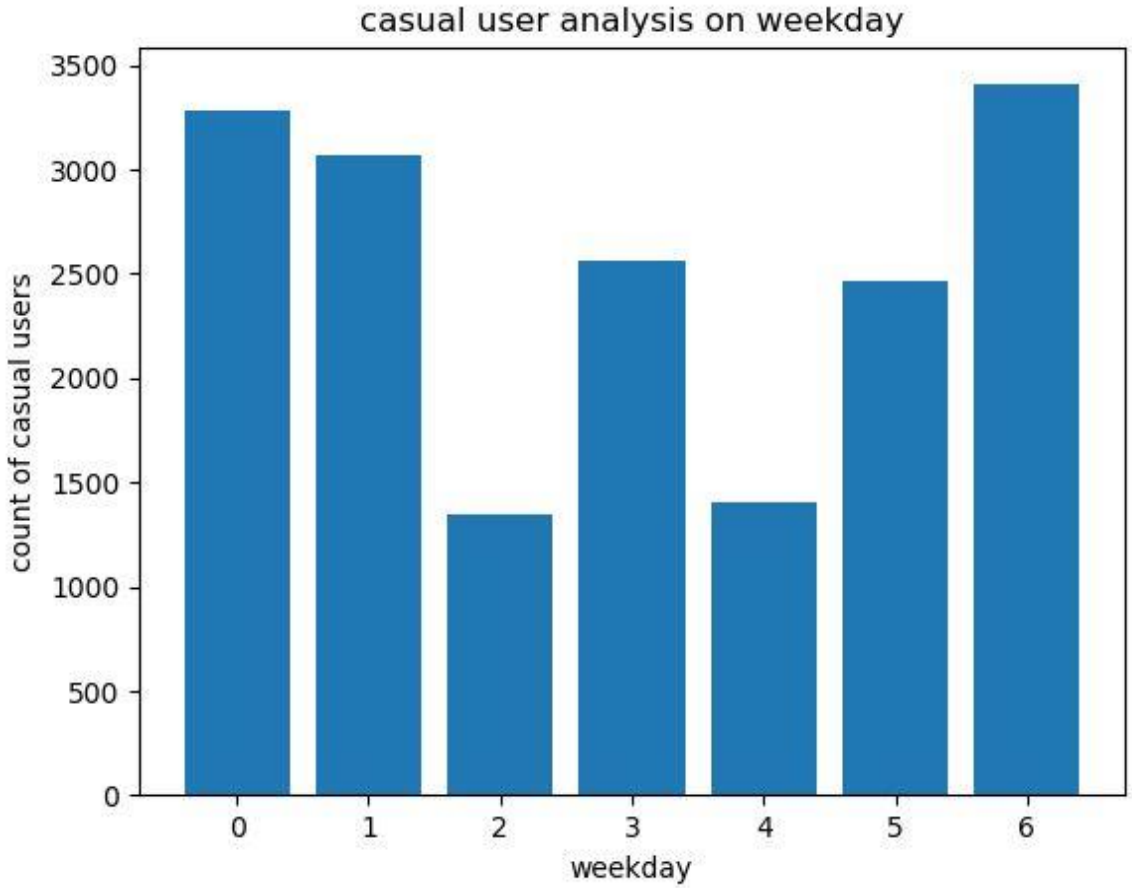
1. we can clearly notice that summer season and where weather conditions are clear, few or partly cloudy on these conditions bike rent count of casual users is quite high than any other condition.
2. we can clearly notice that winter season and where weather conditions are clear, few or partly cloudy on these conditions bike rent count of registered users is quite high than any other condition.
3. we can clearly notice that winter season and where weather conditions are clear, few or partly cloudy on these conditions bike rent count of total users is quite high than any other condition.

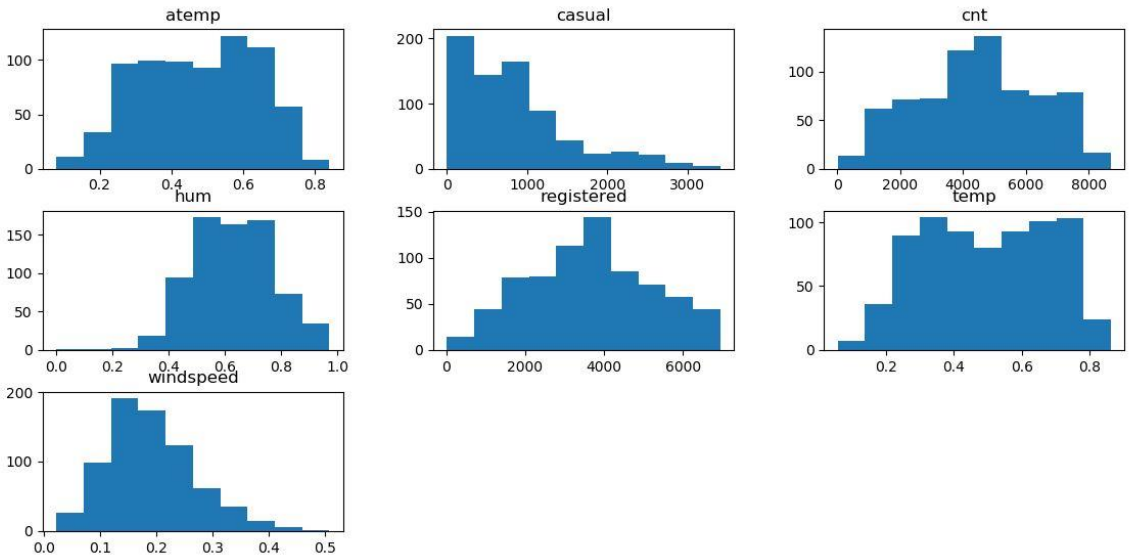
Appendix


casual user analysis on weather
count of casual users
3500
3000
2500
2000
1500
1000
500
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
weather


registered user analysis on weather
count of registered users
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
6
weekday


casual user analysis on weather
count of registered users
7000
6000
5000
4000
3000
2000
1000
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
weather


casual user analysis on weekday
count of casual users
3500
3000
2500
2000
1500
1000
500
0
0
1
2
3
4
5
6
weekday


atemp
casual
cnt
hum
registered
temp
windspeed