Bike Renting

Diksha P. Jagre

Contents

Chapter 1

Introduction

1.1 Problem Statement

The Objective of this Case is to Predication of bike rental on daily based on the environmental and seasonal settings.

1.2 Data

The dataset will be given:

1) day.csv

The details of data attributes 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)
hr	Hour (0 to 23)
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

There are total 16 variables in given dataset.

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. But Before exploring data, you should spend some time thinking about the business problem, gaining the domain knowledge and may be gaining first-hand experience of the problem.

The dataset shows hourly rental data for two years (2011 and 2012).The training data set is for the first 19 days of each month. The test dataset is from 20th day to month's end. We are required to predict the total count of bikes rented during each hour covered by the test set.

In training data set, they have separately given bike demand by registered, casual users and sum of both is given as count. Training data set has 12 variables and Test has 9 (excluding registered, casual and count)

2.1.1 Outlier Analysis

Outlier analysis is one of the technique to understand, clean and prepare data for building a predictive model. Here we used multivariate analysis for testing the dataset.

Till now, we have got a fair understanding of the data set. Now, let's test the hypothesis which we had generated earlier. Here I have added some additional hypothesis from the dataset. Let's test them one by one:

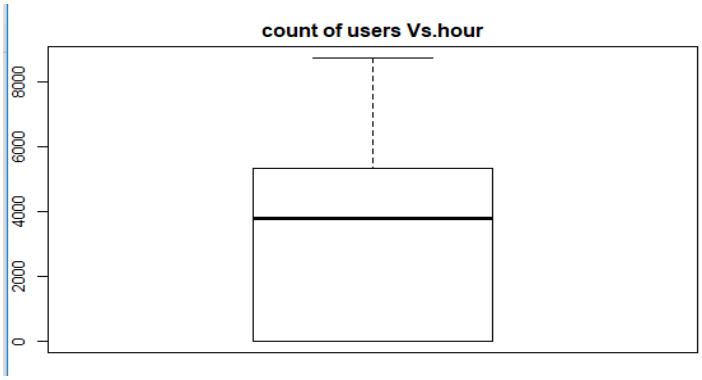
- **Hourly trend**: We don't have the variable 'hour' with us right now. But we can extract it using the datetime (dteday) column.

```
data$hour=substr(data$dteday,12,13)
data$hour=as.factor(data$hour)
```

Let's plot the hourly trend of count over hours and check if our hypothesis is correct or not. We will separate train and test data set from combined one.

```
train=data[as.integer(substr(data$dteday,9,10))<20,]
test=data[as.integer(substr(data$dteday,9,10))>19,]
```

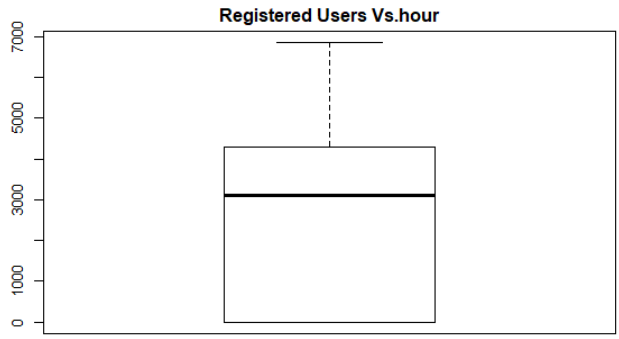
```
boxplot(train$cnt ~ train$hour,xlab="hour",ylab="count of users",main="count of users Vs.hour")
```

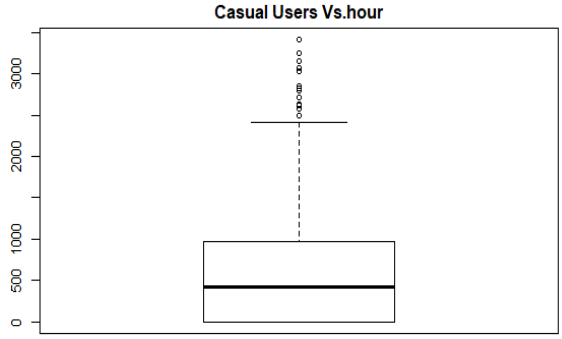


Above, you can see the trend of bike demand over hours.

Here I have analyzed the distribution of total bike demand. Let's look at the distribution of registered and casual users separately.

```
#Distribution Of Registered and Casual Users Seperately
boxplot(train$registered ~ train$hour,xlab="hour",ylab="Registered users",main="Registered Users Vs.hour")
boxplot(train$casual ~ train$hour,xlab="hour",ylab="Casual users",main="Casual Users Vs.hour")
```



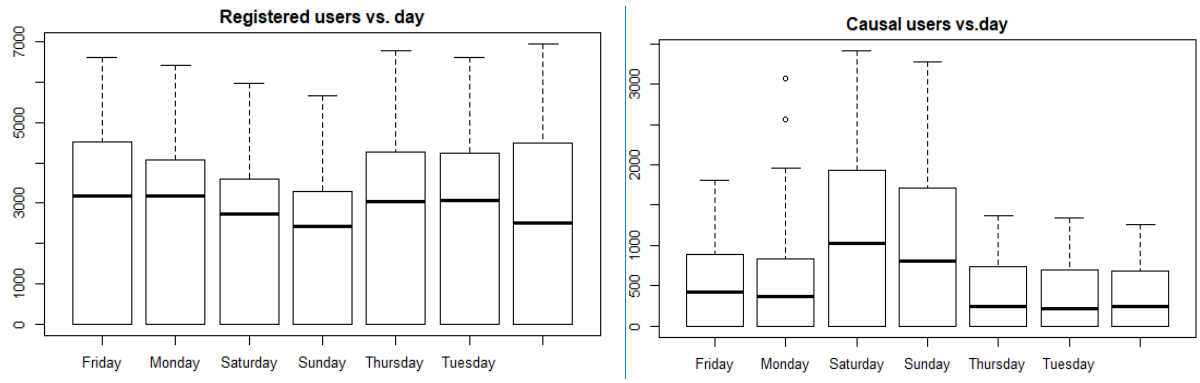


Above you can see that registered users have similar trend as count. Whereas, casual users have different trend. Thus, we can say that 'hour' is significant variable and our hypothesis is 'true'.

- **Daily Trend:** Like Hour, we will generate a variable for day from datetime variable and after that we'll plot it.

```
date=substr(data$dteday,1,10)
days<-weekdays(as.Date(date))
data$day=days
```

Plot shows registered and casual users' demand over days.

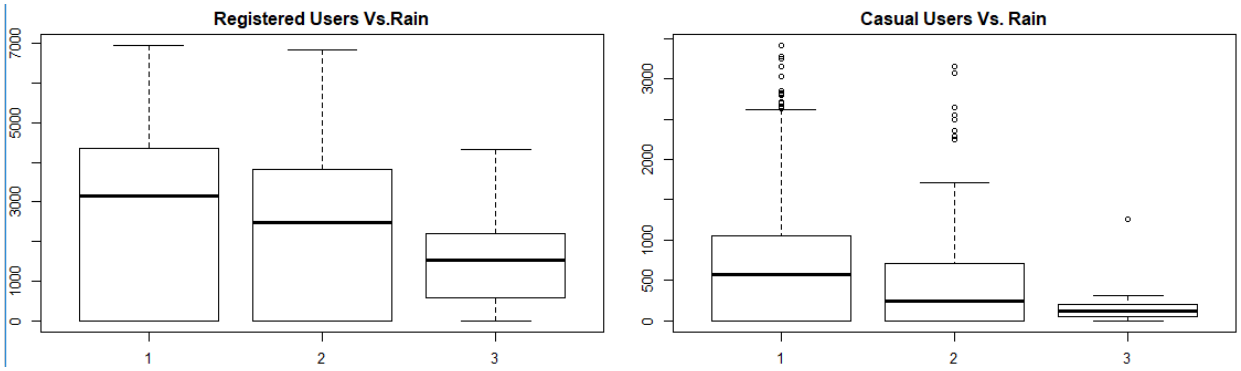


While looking at the plot, I can say that the demand of causal users increases over weekend.

- **Rain:** We don't have the 'rain' variable with us but have 'weathersit' which is sufficient to test our hypothesis. As per variable description, weather 3 represents light rain and weather 4 represents heavy rain

```
data$rain=substr(data$weathersit,1,4)
data$rain=as.factor(data$rain)

train=data[as.integer(substr(data$weathersit,1,4))<20,]
test=data[as.integer(substr(data$weathersit,1,4))>19,]

boxplot(data$registered ~ data$rain,xlab="rain",ylab="registered users",main="Registered Users Vs.Rain")
boxplot(data$casual ~ data$rain,xlab="rain",ylab="casual users",main="Casual Users Vs. Rain")
```

Take a look at the plot:



It is clearly satisfying our hypothesis.

- **Temperature, Windspeed and Humidity:** These are continuous variables so we can look at the correlation factor to validate hypothesis.

```
sub=data.frame(train$registered,train$casual,train$cnt,train$temp,train$hum,train$atemp,train$windspeed)
cor(sub)
```

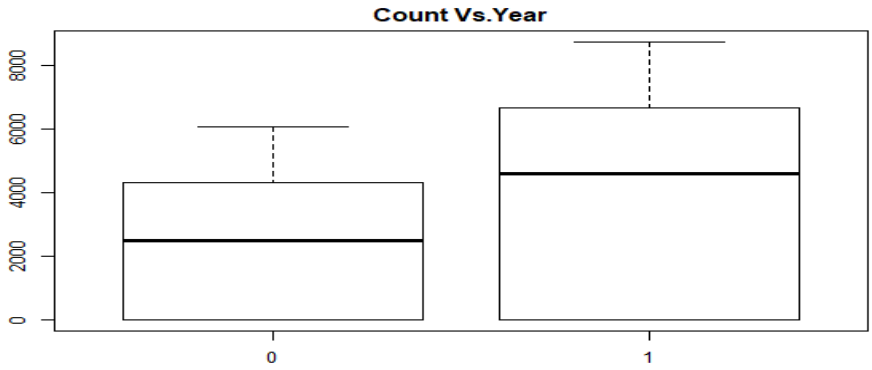
	train.registered	train.casual	train.cnt	train.temp	train.hum	train.atemp	train.windspeed
train.registered	1.000000000	0.63443625	0.97758592	0.2866871	-0.003863205	0.2867688	-0.11682693
train.casual	0.634436252	1.00000000	0.78295571	0.3721885	-0.063902104	0.3714565	-0.09972262
train.cnt	0.977585921	0.78295571	1.00000000	0.3320947	-0.020514191	0.3319611	-0.12118201
train.temp	0.286687137	0.37218847	0.33209474	1.0000000	0.126962939	0.9917016	-0.15794412
train.hum	-0.003863205	-0.06390210	-0.02051419	0.1269629	1.000000000	0.1399881	-0.24848910
train.atemp	0.286768782	0.37145646	0.33196107	0.9917016	0.139988060	1.0000000	-0.18364297
train.windspeed	-0.116826932	-0.09972262	-0.12118201	-0.1579441	-0.248489099	-0.1836430	1.00000000

Here are a few inferences you can draw by looking at the above histograms:

- Variable temp is positively correlated with dependent variables (casual is more compare to registered)
- Variable atemp is highly correlated with temp.
- Windspeed has lower correlation as compared to temp and humidity

- **Time:** Let's extract year of each observation from the datetime column and see the trend of bike demand over year.

```
boxplot(data$cnt ~ data$yr,xlab="year",ylab="count",main="Count Vs.Year")
```



Here 0 represent 2011 and 1 represent 2012.You can see that 2012 has higher bike demand as compared to 2011.

- **Pollution & Traffic:** We don't have the variable related with these metrics in our data set so we cannot test this hypothesis.

2.1.2 Feature Selection

In addition to existing independent variables, we will create new variables to improve the prediction power of model. Initially, you must have noticed that we generated new variables like hour & day.

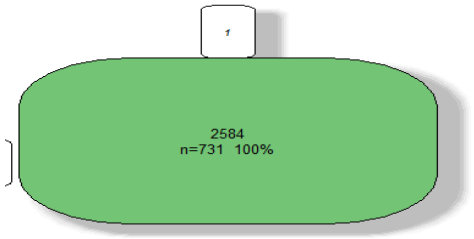
Here we will create more variables, let's look at the some of these:

A) **Hour Bins:** Initially, we have broadly categorized the hour into three categories. Let's create bins for the hour variable separately for casual and registered users. Here we will use decision tree to find the accurate bins.

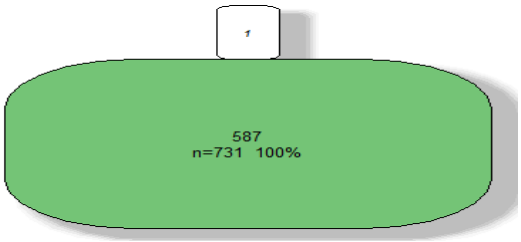
```
train$hour = as.integer(train$hour)          #Convert hour to integer
test$hour = as.integer(test$hour)            #Modifying in both train and test data set
```

We use the library rpart for decision tree algorithm.

```
library(rpart)
library(rattle)                #These libraries will be used to get a good visual plot for the decision model
library(rpart.plot)
library(RColorBrewer)

hour_reg=rpart(registered~hour,data=train)
summary(hour_reg)
fancyRpartPlot(hour_reg)

hour_cas=rpart(casual~hour,data=train)
summary(hour_cas)
fancyRpartPlot(hour_cas)
```




```
> summary(hour_reg)
Call:
rpart(formula = registered ~ hour, data = train)
  n= 731

    CP nsplit rel error xerror xstd
1 0.01      0         1      0    0

Node number 1: 731 observations
  mean=2584.101, MSE=4586268
```

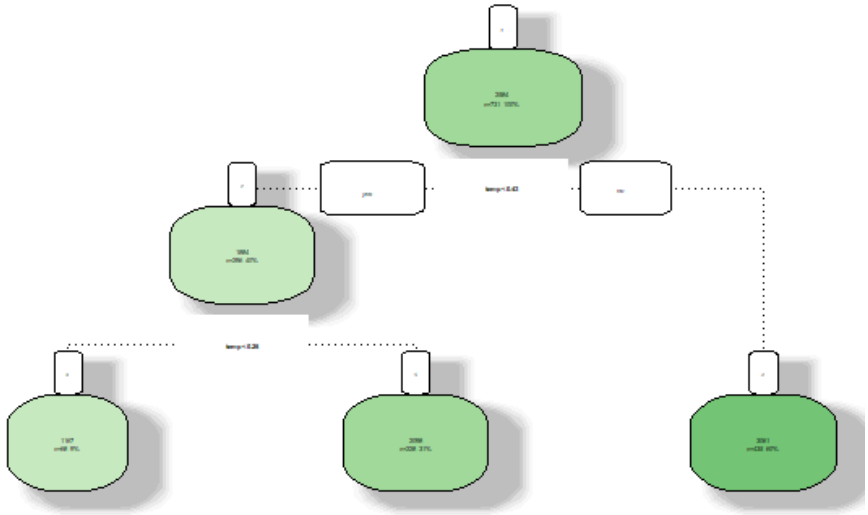



```
> summary(hour_cas)
Call:
rpart(formula = casual ~ hour, data = train)
  n= 731

    CP nsplit rel error xerror xstd
1 0.01      0         1      0    0

Node number 1: 731 observations
  mean=586.7948, MSE=461459.1
```

B) **Temp Bins:** Using similar methods, we have created bins for temperature for both registered and casuals users. Variables created are (temp_reg and temp_cas).

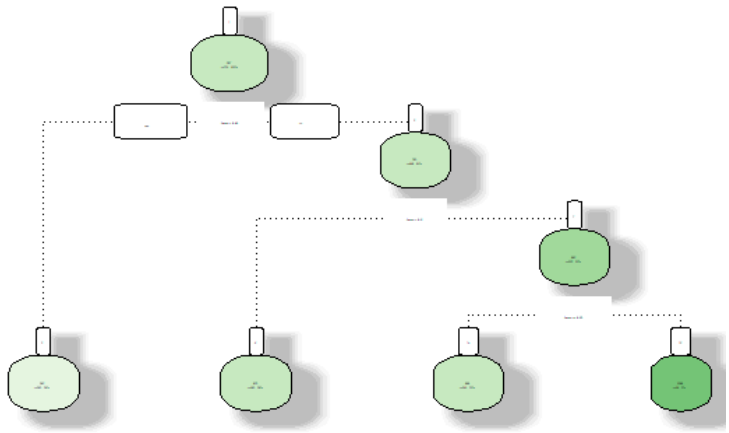
```
temp_reg=rpart(registered~temp,data=train)
summary(temp_reg)
fancyRpartPlot(temp_reg)

temp_cas=rpart(casual~temp,data=train)
summary(temp_cas)
fancyRpartPlot(temp_cas)
```

Temp_Reg :



```
> summary(temp_reg)
Call:
rpart(formula = registered ~ temp, data = train)
  n= 731

          CP nsplit rel error    xerror       xstd
1 0.07272883      0 1.0000000 1.0028436 0.03226063
2 0.01352644      1 0.9272712 0.9330093 0.03222866
3 0.01000000      2 0.9137447 0.9249804 0.03204435

Variable importance
temp
 100

Node number 1: 731 observations,    complexity param=0.07272883
  mean=2584.101, MSE=4586268
  left son=2 (296 obs) right son=3 (435 obs)
  Primary splits:
      temp < 0.432373 to the left,  improve=0.07272883, (0 missing)

Node number 2: 296 observations,    complexity param=0.01352644
  mean=1883.966, MSE=2738206
  left son=4 (68 obs) right son=5 (228 obs)
  Primary splits:
      temp < 0.255    to the left,  improve=0.05595031, (0 missing)

Node number 3: 435 observations
  mean=3060.515, MSE=5283276

Node number 4: 68 observations
  mean=1167.25, MSE=843382.3

Node number 5: 228 observations
  mean=2097.724, MSE=3104433
```

Temp_cas:

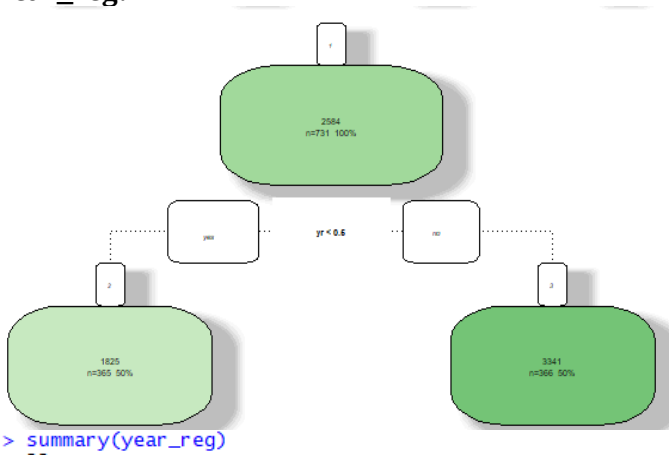


```
> summary(temp_cas)
Call:
rpart(formula = casual ~ temp, data = train)
  n= 731

         CP nsplit rel error    xerror       xstd
1 0.14174709      0 1.0000000 1.0032693 0.07374400
2 0.01922027      1 0.8582529 0.8873767 0.06471807
3 0.01104391      2 0.8390326 0.9108731 0.06603437
4 0.01000000      3 0.8279887 0.9325730 0.06830915

Variable importance
temp
 100

Node number 1: 731 observations,    complexity param=0.1417471
  mean=586.7948, MSE=461459.1
  left son=2 (283 obs) right son=3 (448 obs)
  Primary splits:
      temp < 0.41875   to the left,  improve=0.1417471, (0 missing)

Node number 2: 283 observations
  mean=265.0071, MSE=122611.6

Node number 3: 448 observations,    complexity param=0.01922027
  mean=790.067, MSE=568777.8
  left son=6 (193 obs) right son=7 (255 obs)
  Primary splits:
      temp < 0.5995835 to the left,  improve=0.02544424, (0 missing)

Node number 6: 193 observations
  mean=651.7876, MSE=482672.6


Node number 7: 255 observations,    complexity param=0.01104391
  mean=894.7255, MSE=608522.2
  left son=14 (241 obs) right son=15 (14 obs)
  Primary splits:
      temp < 0.6108335 to the right, improve=0.02400806, (0 missing)

Node number 14: 241 observations
  mean=865.5934, MSE=543514

Node number 15: 14 observations
  mean=1396.214, MSE=1461491
```

C) **Year Bins:** We had a hypothesis that bike demand will increase over time and we have proved it also.

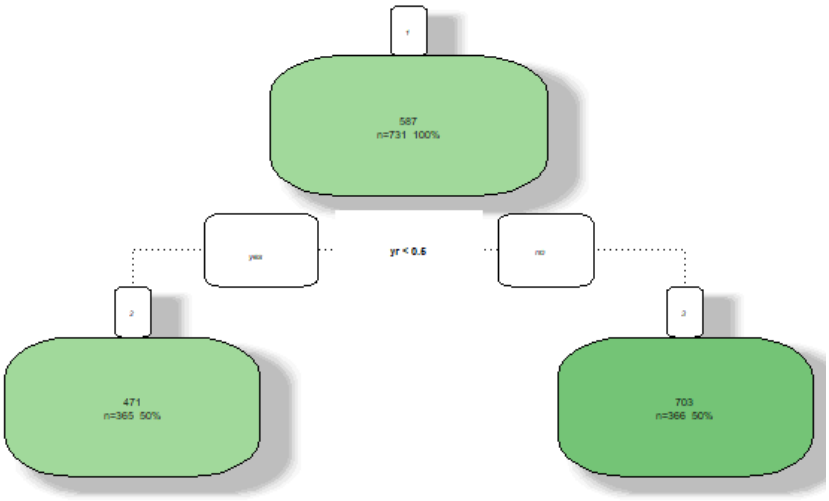
```
#We have created bins for year for both registered and casual users

year_reg=rpart(registered~yr,data = train)
summary(year_reg)
fancyRpartPlot(year_reg)

year_cas=rpart(casual~yr,data = train)
summary(year_cas)
fancyRpartPlot(year_cas)
```

Year_reg:




```
> summary(year_reg)
Call:
rpart(formula = registered ~ yr, data = train)
  n= 731

         CP nsplit rel error    xerror       xstd
1 0.1252215      0 1.0000000 1.0030085 0.03229134
2 0.0100000      1 0.8747785 0.8794609 0.03020053

Variable importance
 yr
100

Node number 1: 731 observations,    complexity param=0.1252215
  mean=2584.101, MSE=4586268
  left son=2 (365 obs) right son=3 (366 obs)
  Primary splits:
      yr < 0.5 to the left,  improve=0.1252215, (0 missing)

Node number 2: 365 observations
  mean=1825.238, MSE=2348571

Node number 3: 366 observations
  mean=3340.891, MSE=5670821
```

Year_cas:

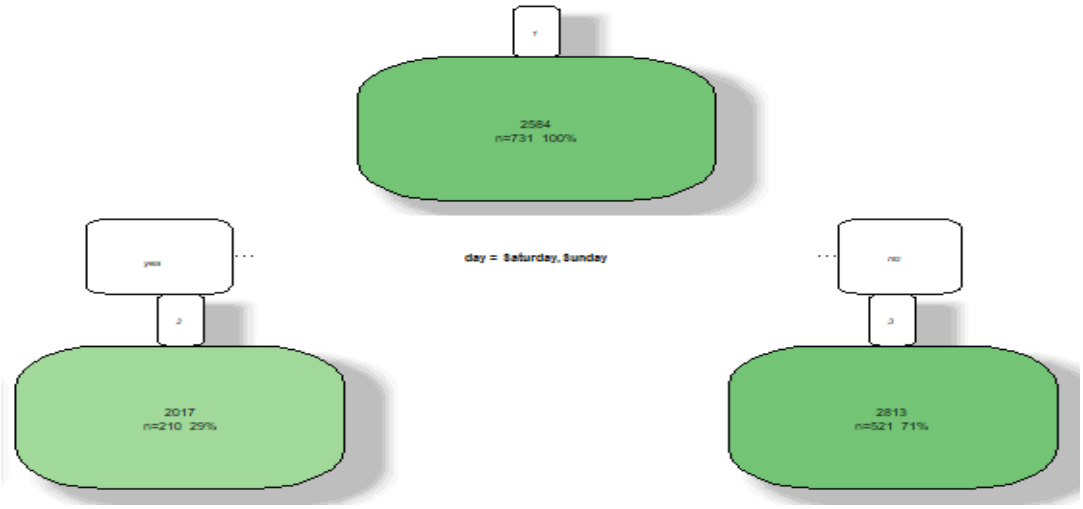



```
> summary(year_cas)
Call:
rpart(formula = casual ~ yr, data = train)
  n= 731

         CP nsplit rel error    xerror       xstd
1 0.02909663      0 1.0000000 1.0046979 0.07377145
2 0.01000000      1 0.9709034 0.9764941 0.07002081

Variable importance
 yr
100

Node number 1: 731 observations,    complexity param=0.02909663
  mean=586.7948, MSE=461459.1
  left son=2 (365 obs) right son=3 (366 obs)
  Primary splits:
      yr < 0.5 to the left,  improve=0.02909663, (0 missing)

Node number 2: 365 observations
  mean=470.7616, MSE=331805.3

Node number 3: 366 observations
  mean=702.5109, MSE=563941.6
```

D) **Day Type:** Created a variable having categories like “weekday”, “weekend” and “holiday”.

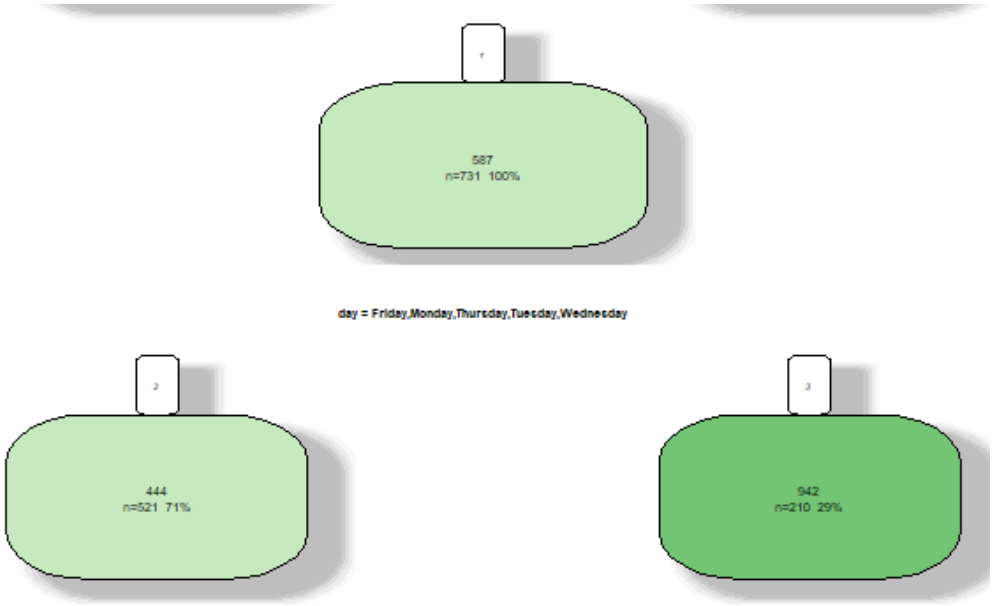
```
#We have created bins for day for both registered and casual users
day_reg=rpart(registered~day,data = train)
summary(day_reg)
fancyRpartPlot(day_reg)

day_cas=rpart(casual~day,data = train)
summary(day_cas)
fancyRpartPlot(day_cas)
```

Day_Reg:




```
> summary(day_reg)
Call:
rpart(formula = registered ~ day, data = train)
  n= 731

          CP nsplit rel error    xerror       xstd
1 0.02826606      0 1.0000000 1.0020013 0.03217838
2 0.01000000      1 0.9717339 0.9831353 0.03160182

Variable importance
day
100

Node number 1: 731 observations,    complexity param=0.02826606
  mean=2584.101, MSE=4586268
  left son=2 (210 obs) right son=3 (521 obs)
  Primary splits:
      day splits as  RRLLRRR, improve=0.02826606, (0 missing)

Node number 2: 210 observations
  mean=2016.986, MSE=2954071

Node number 3: 521 observations
  mean=2812.689, MSE=5062271
```

Day_cas:




```
> summary(day_cas)
Call:
rpart(formula = casual ~ day, data = train)
  n= 731

         CP nsplit rel error    xerror       xstd
1 0.1103542      0 1.0000000 1.0028475 0.07375327
2 0.0100000      1 0.8896458 0.8972791 0.05693164

Variable importance
day
100

Node number 1: 731 observations,    complexity param=0.1103542
  mean=586.7948, MSE=461459.1
  left son=2 (521 obs) right son=3 (210 obs)
  Primary splits:
      day splits as  LLRRLLL, improve=0.1103542, (0 missing)

Node number 2: 521 observations
  mean=443.5259, MSE=213524

Node number 3: 210 observations
  mean=942.2381, MSE=899310.4
```

2.2 Modeling

2.2.1 Model Selection

In our early stage of analysis during feature selection, we have come to understand that using decision tree the test result will be different. Therefore, we can neither combine the data sets nor use a single model for predicting variables. Hence, we need to analyzed the data sets separately and generate separate models for data sets.

The dependent variable can fall in either of the four categories:

1. Nominal
2. Ordinal
3. Interval
4. Ratio

If the dependent variable, in our case Registered, Casual and Count, is Nominal the only predictive analysis that we can perform is Classification, and if the dependent variable is Interval or Ratio the normal method is to do a Regression analysis, or classification after binning. But the dependent variable we are dealing with is Interval or Ratio, for which classification and regression can be done, because the Registered, Casual and Count variable has intervals.

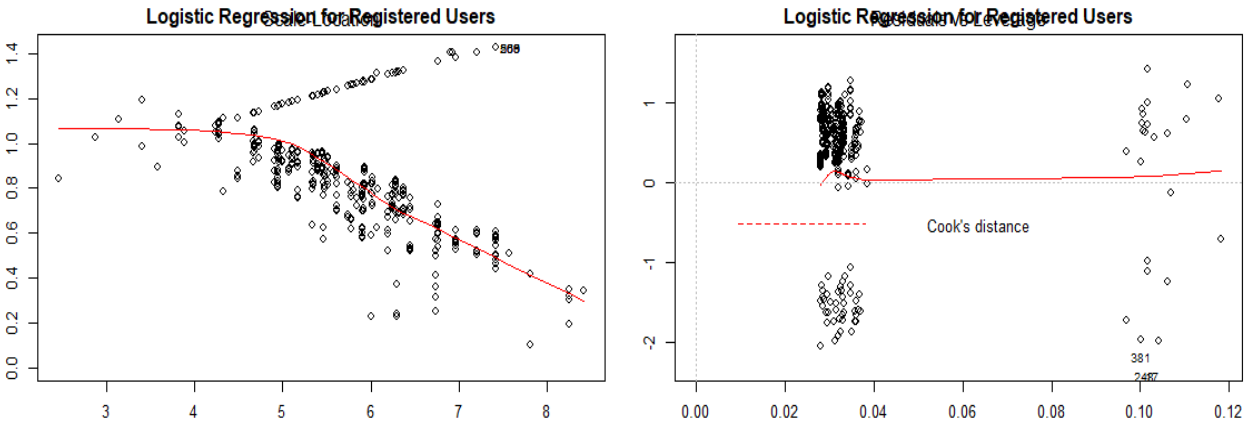
You always start your model building from the simplest to more complex. Therefore, we use Logistic Regression

2.2.2 Logistic Regression

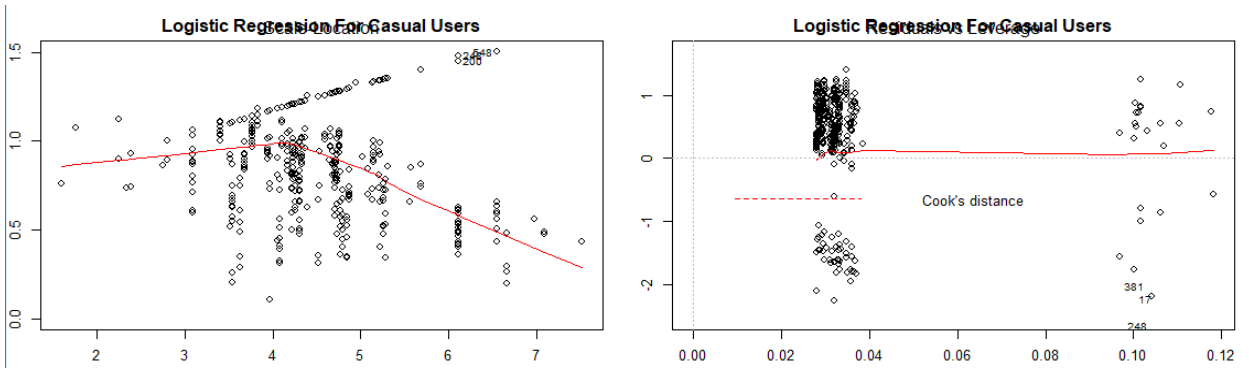
Logistic Regression is statistical model which input can be continuous or categorical. The input variables is in factor with 2 or more levels. That means you have only categorical variables to perform logistic Regression.

Here Registered and Casual users are predicted using Logistic Regression.

```
#Perform Logistic Regression for Registered Users
logit_model_1 <-glm(logreg ~ season+mnth+holiday+workingday+weathersit, data=train)
pred_model_1 <-predict(logit_model_1,test)
test$logreg=pred_model_1
print(logit_model_1)
plot(logit_model_1,main="Logistic Regression for Registered Users")
```




```
#Perform Logistic Regression For Casual Users
logit_model_2 <-glm(logcas ~season+mnth+holiday+workingday+weathersit,data=train)
pred_model_2 <-predict(logit_model_2,test)
test$logcas=pred_model_2
print(logit_model_2)
plot(logit_model_2,main="Logistic Regression For Casual Users")
```



2.2.3 Random Forest

Random forest is a tree-based algorithm which involves building several trees (decision trees), then combining their output to improve generalization ability of the model. The method of combining trees is known as an ensemble method. Ensembling is nothing but a combination of weak learners (individual trees) to produce a strong learner.

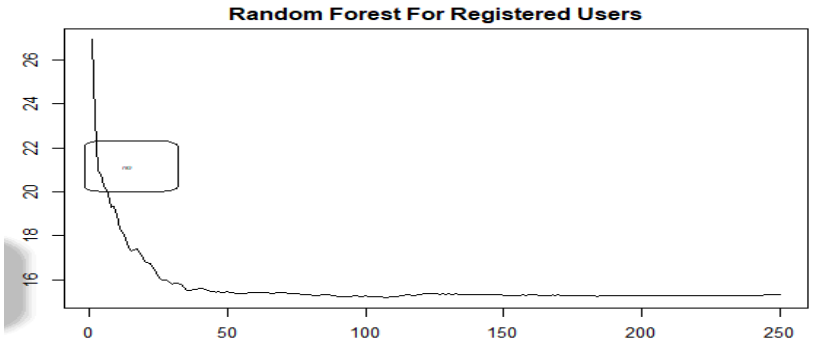
Random Forest can be used to solve regression and classification problems. In regression problems, the dependent variable is continuous. In classification problems, the dependent variable is categorical.

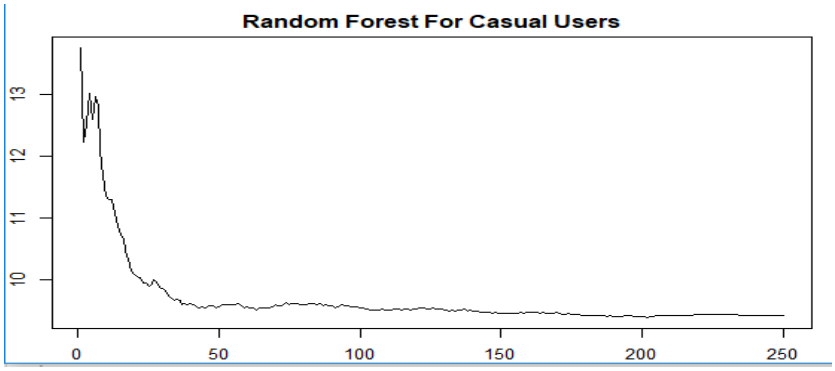
So, In our given dataset the dependent (target) variable is continuous so we solve this problem using regression.

```r
#Before executing random forest,execute following steps
#Convert discrete variables into factor(weathersit,season,hour,holiday,workingday,mnth)

train$hour = as.factor(train$hour)
test$hour = as.factor(test$hour)

train$weathersit = as.factor(train$weathersit)
test$weathersit  = as.factor(test$weathersit)

train$season = as.factor(train$season)
test$season = as.factor(test$season)

train$holiday = as.factor(train$holiday)
test$holiday = as.factor(test$holiday)

train$workingday = as.factor(train$workingday)
test$workingday = as.factor(test$workingday)

train$mnth = as.factor(train$mnth)
test$mnth = as.factor(test$mnth)
```

```r
#log transformation for some skewed variables, which can be seen from their distribution.
train$reg1=train$registered+1
train$cas1=train$casual+1
train$logcas=log(train$cas1)
train$logreg=log(train$reg1)
test$logreg=0
test$logcas=0
```

```r
#predicting the log of registered users.
library(randomForest)
set.seed(415)
fit1 <- randomForest(logreg ~ hour +workingday+holiday+hum+atemp+windspeed+season+weathersit+yr, data=train,importa
pred1=predict(fit1,test)
test$logreg=pred1
print(fit1)
plot(fit1,main="Random Forest For Registered Users")
```





Chapter 3

Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, 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. ROC Curve
2. Concordance & Discordance
3. ks-plot
4. ks-stat
5. Confusion Matrix

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

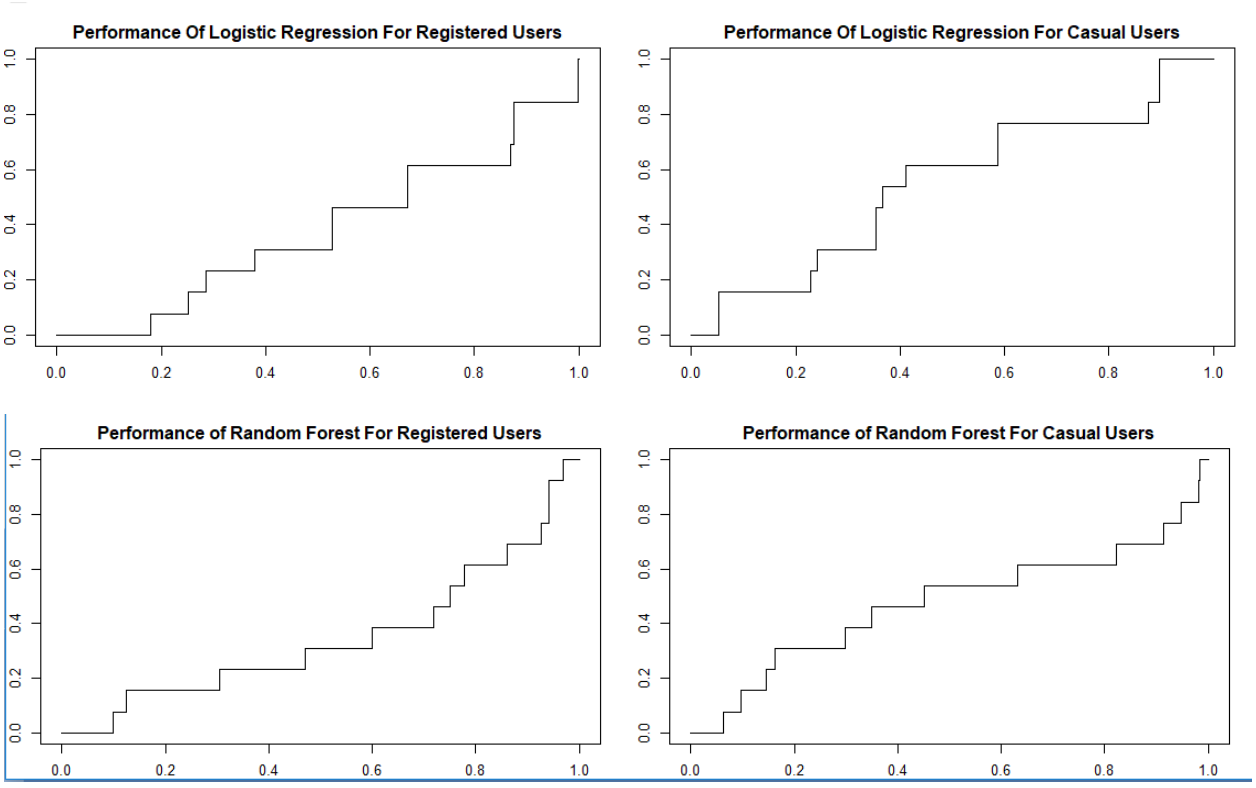
3.1.1 ROC Curve

ROC Curve model is sort of a balance between predicting the one's accurately or the zeroes accurately.In other words sensitivity and specificity.

This is nicely captured by the 'Receiver Operating Characteristics' curve, also called as the ROC curve. In fact, the area under the ROC curve can be used as an evaluation metric to compare the efficacy of the models.

Let's plot the curve and the area using the `plotROC` from `InformationValue` package.

```
# Performance Evaluation Of Logistic Regression for Registered Users
library(ROCR)
pred_logit_reg<- prediction(predict(logit_model_1), train$holiday)
perf_logit_reg <- performance(pred_logit_reg,"tpr","fpr")
plot(perf_logit_reg,main="Performance Of Logistic Regression For Registered Users")

#Performance Evaluation Of Logistic Regression for Casual Users
pred_logit_cas<- prediction(predict(logit_model_2), train$holiday)
perf_logit_cas <- performance(pred_logit_cas,"tpr","fpr")
plot(perf_logit_cas,main="Performance Of Logistic Regression For Casual Users")


# 2.Random Forest
#Performance Evaluation Of Random Forest for Registered Users
pred_randomforest_reg <-prediction(predict(fit1),train$holiday)
perf_randomforest_reg <-performance(pred_randomforest_reg,"tpr","fpr")
plot(perf_randomforest_reg,main="Performance of Random Forest For Registered Users")

#Performance Evaluation Of Random Forest for Casual Users
pred_randomforest_cas <-prediction(predict(fit2),train$holiday)
perf_randomforest_cas <-performance(pred_randomforest_cas,"tpr","fpr")
plot(perf_randomforest_cas,main="Performance of Random Forest For Casual Users")

#we have here is a line that traces the probability cutoff from 1 at the bottom-left to 0 in the top right.
#This is a way of analyzing how the sensitivity and specificity perform for the full range of probability cutoffs,
#that is from 0 to 1.
```



Instead, what we have here is a line that traces the probability cutoff from 1 at the bottom-left to 0 in the top right.

This is a way of analyzing how the sensitivity and specificity perform for the full range of probability cutoffs,that is from 0 to 1.

The ROC curve is the only metric that measures how well the model does for different values of prediction probability cutoffs. The `optimalCutoff` function from `InformationValue` can be used to know what cutoff gives the best sensitivity, specificity or both.

3.1.2 Concordance & Discordance

In an ideal model, the probability scores of all true 1's should be greater than the probability scores of ALL true 0's. Such a model is said to be perfectly concordant and this phenomenon can be measured by Concordance and Discordance.

For a perfect model, this will be 100%. So, the higher the concordance, the better is the quality of the model. This can be computed using the `Concordance` function in `InformationValue` package.

```
#Evaluation Using Concordance & Discordance
install.packages("InformationValue")  # For stable CRAN version

#Performance Evaluation of Registered Users Using Concordance $ Discordance
InformationValue::Concordance(test$logreg, test$holiday)

##Performance Evaluation of Casual Users Using Concordance $ Discordance
InformationValue::Concordance(test$logcas,test$holiday)

#we take all possible combinations of true events and non-events.
#Concordance is the percentage of pairs,
#where true event's probability scores are greater than the scores of true non-events.
#Here no pairing is given for calculating the concordance.
```

```
> InformationValue::Concordance(test$logreg, test$holiday)
$`Concordance`
[1] NaN

$Discordance
[1] NaN

$Tied
[1] NaN

$Pairs
[1] 0
```

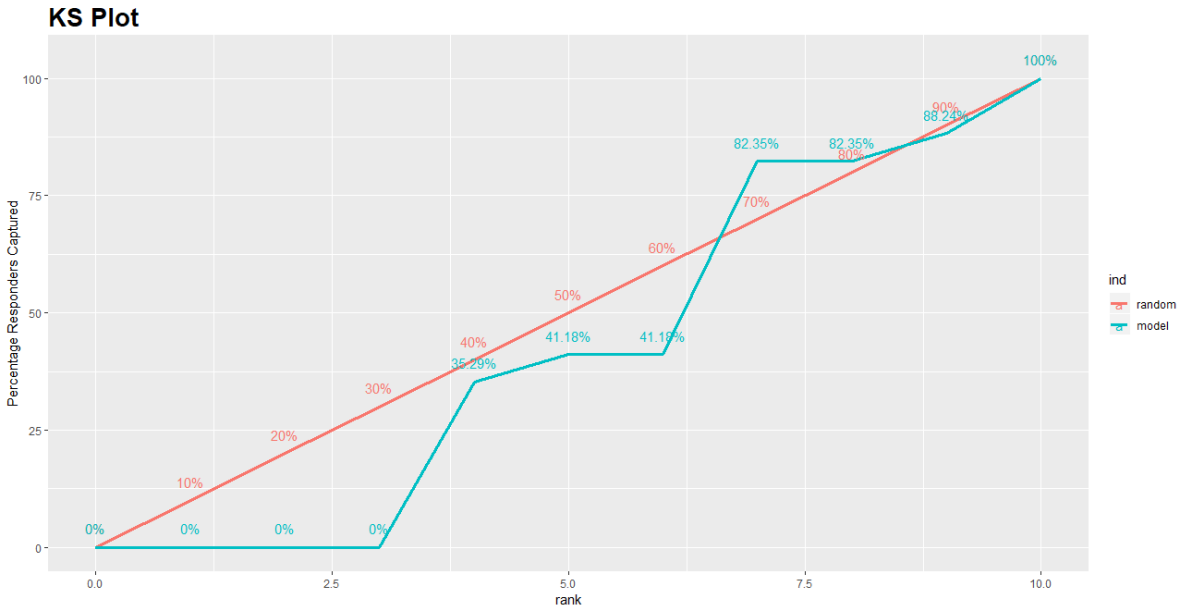
Here no paring is between two variables so their respective concordance and discordance is null.

3.1.3 ks – plot

The KS Chart is particularly useful in marketing campaigns and ads click predictions where you want to know the right population size to target to get the maximum response rate.

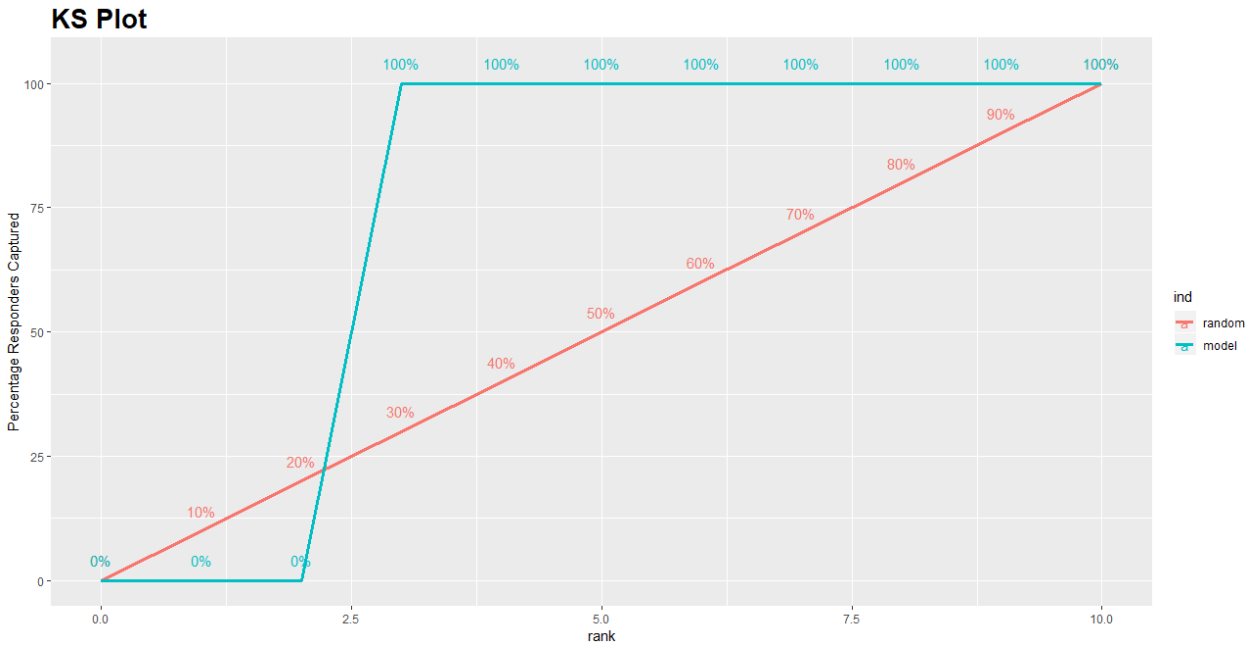
The KS chart below shows how this might look like. The length of the vertical dashed red line indicates the KS Statistic

```
#Evaluation Using ks-plot of Registered Users
InformationValue::ks_plot(test$logreg,test$holiday)

#Evaluation Using ks-plot of Casual Users
InformationValue::ks_plot(test$logcas,test$holiday)

#The KS chart and statistic that is widely used in credit scoring scenarios
#and for selecting the optimal population size of target users for marketing campaigns.
```



ks – plot for Registered Users

By targeting the top 65% of the population, the model is able to cover 82.35% of responders (1's).



ks –plot for Casual Users

By targeting the top 20% of the population, the model is able to cover 100 % of responders (1's).

3.1.4 ks – stat

Ks-statistics and ks-chart are used to make decision like how to predict the bike rental count on the daily basis.

The significance of KS statistic is, it helps to understand, what portion of the population should be targeted to get the highest response rate (1's).

The KS statistic can be computed using the `ks_stat` function in `InformationValue` package. By setting the `returnKSTable = T`, you can retrieve the table that contains the detailed decile level splits.

+

```
#Evaluation Using ks-stat of Registered Users
InformationValue::ks_stat(test$logreg,test$holiday)

InformationValue::ks_stat(test$logreg,test$holiday,returnKSTable = T)

#Evaluation Using ks-stat of Casual Users
InformationValue::ks_stat(test$logcas,test$holiday)

InformationValue::ks_stat(test$logcas,test$holiday,returnKSTable = T)

#The significance of KS statistic is, it helps to understand,
#what portion of the population should be targeted to get the highest response rate (1's).
```

```
> InformationValue::ks_stat(test$logreg,test$holiday)
[1] 0.3529
```

```
> InformationValue::ks_stat(test$logcas,test$holiday)
[1] 1
```

3.1.5 Confusion Matrix

The `caret` package provides the awesome `confusionMatrix` function for this. It takes in the predicted and actual values. And to avoid confusion, always specify the `positive` argument.

Otherwise, it is possible for '0' to be taken as 'positive' or the 'event', and will cause a big mistake which may go unnoticed.

```
#Evalution Performance Using Confusion Matrix
library(caret)
caret::confusionMatrix(test$workingday, test$holiday, positive="1", mode="everything")

#The rows in the confusion matrix are the count of predicted 0's and 1's (from y_pred),
#while, the columns are the actuals (from y_act).
```

```
> caret::confusionMatrix(test$workingday, test$holiday, positive="1", mode="everything")
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  78   8
         1 189   0

               Accuracy : 0.2836
                 95% CI : (0.2311, 0.3409)
    No Information Rate : 0.9709
    P-Value [Acc > NIR] : 1

                  Kappa : -0.0591
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.00000
            Specificity : 0.29213
         Pos Pred Value : 0.00000
         Neg Pred Value : 0.90698
              Precision : 0.00000
                 Recall : 0.00000
                     F1 :     NaN
             Prevalence : 0.02909
         Detection Rate : 0.00000
   Detection Prevalence : 0.68727
      Balanced Accuracy : 0.14607

       'Positive' Class : 1
```

3.2 Model Selection

We can see that out of three models one models perform great accuracy work therefore we can select those models without any loss of information.

Appendix A - Extra Figures

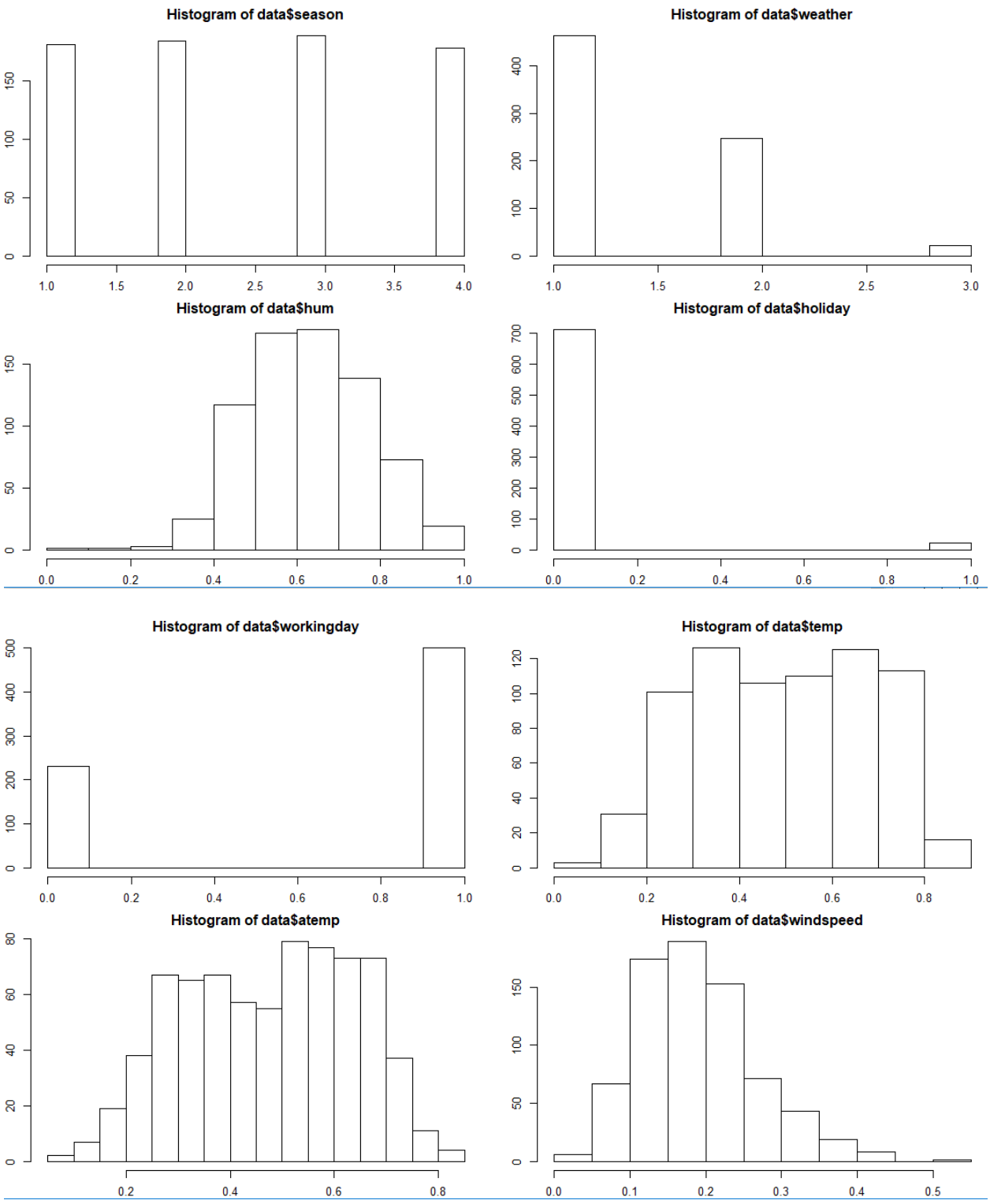


Figure 1.1 Histogram For All Numerical Variables

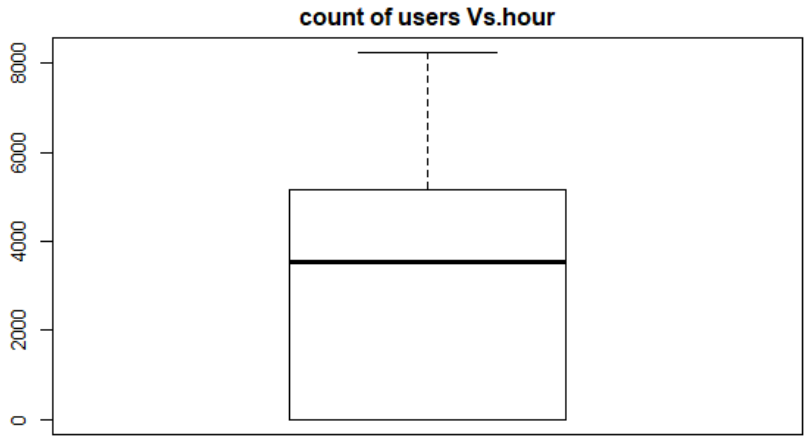


Figure 2.1 Boxplot for Count of users Vs. Hour

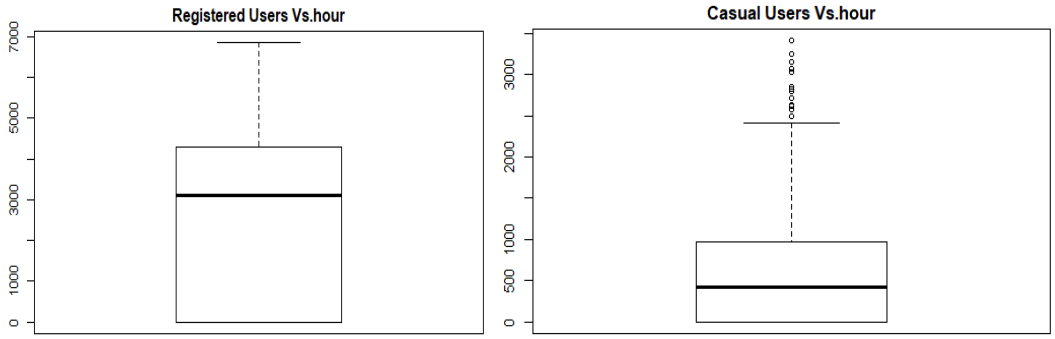


Figure 2.2 Boxplot for Registered & Casual Users Vs. Hour

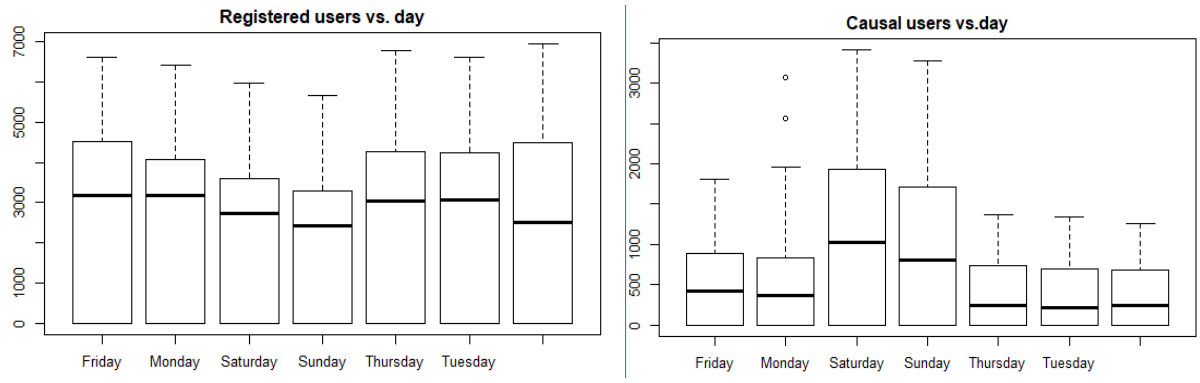


Figure 2.3 Boxplot for Registered & Casual Users Vs. Day

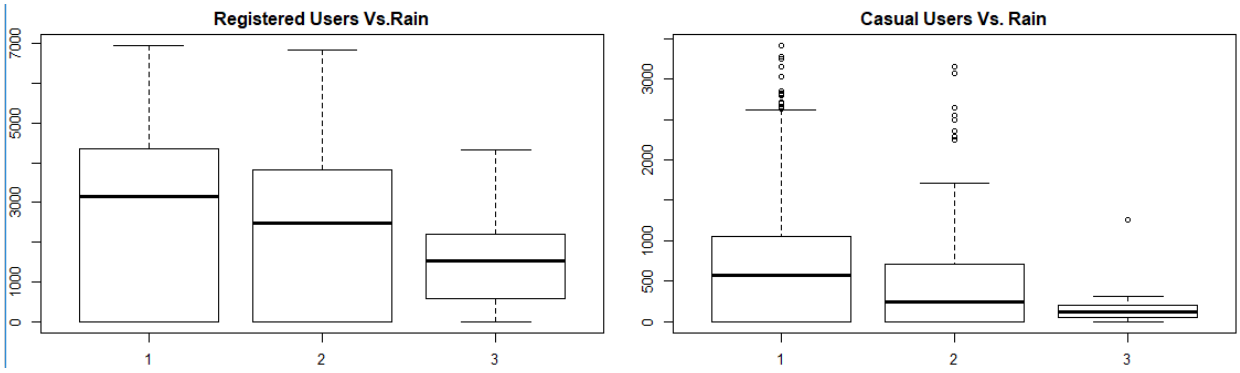


Figure 2.4 Boxplot for Registered & Casual Users Vs.Rain

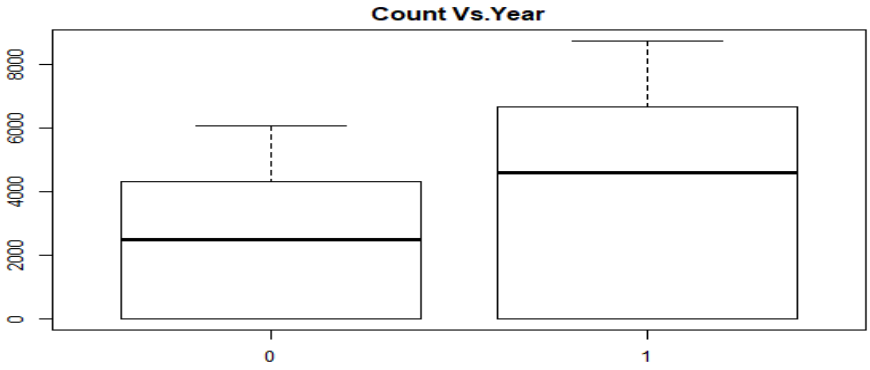


Figure 2.5 Boxplot for Registered & Casual Users Vs. Year

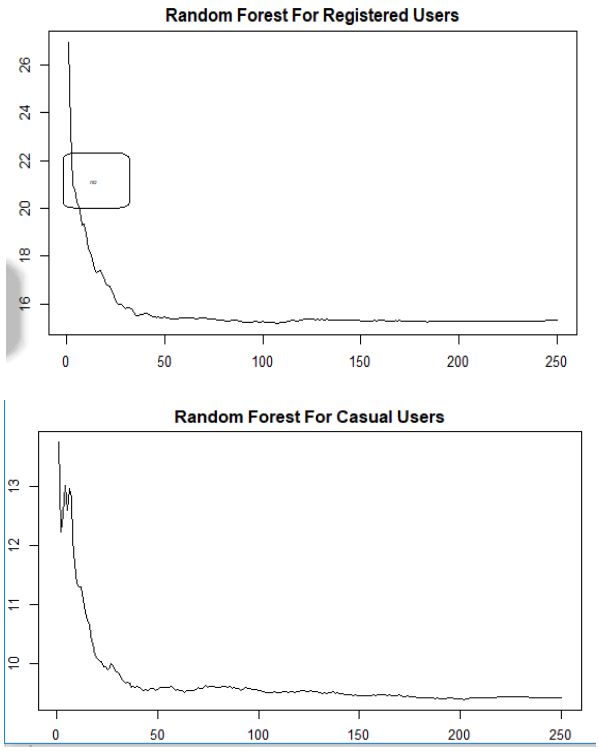
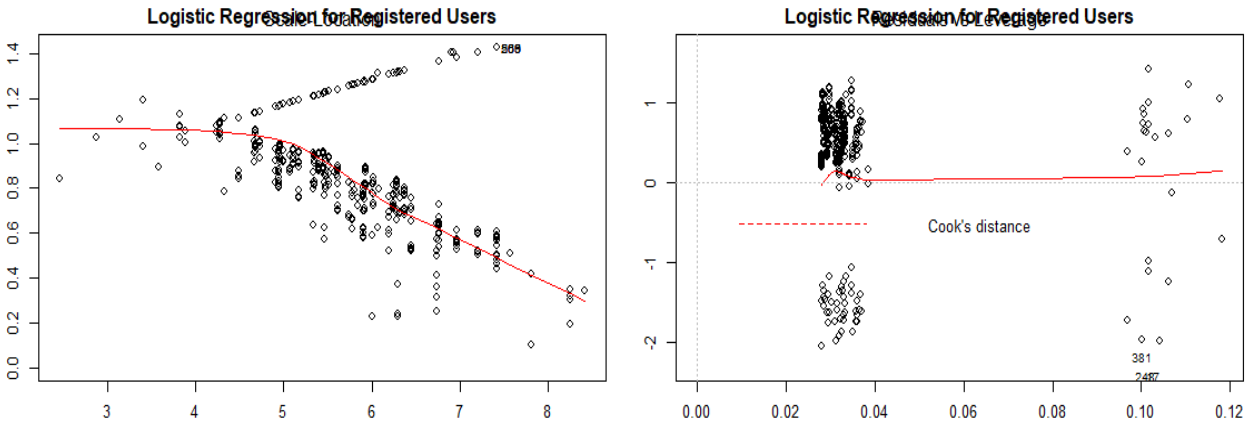


Figure 2.6 Random Forest For Registered & Casual Users



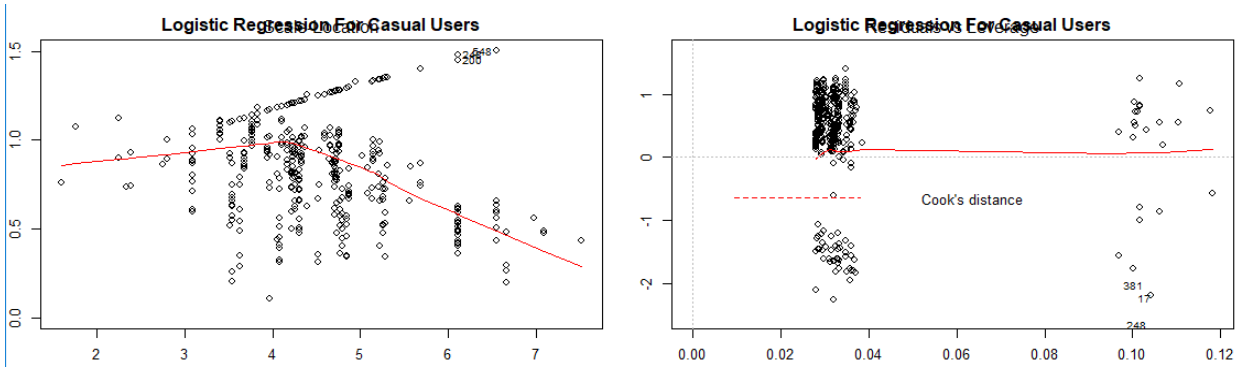


Figure 2.7 Logistic Regression For Registered & Casual Users

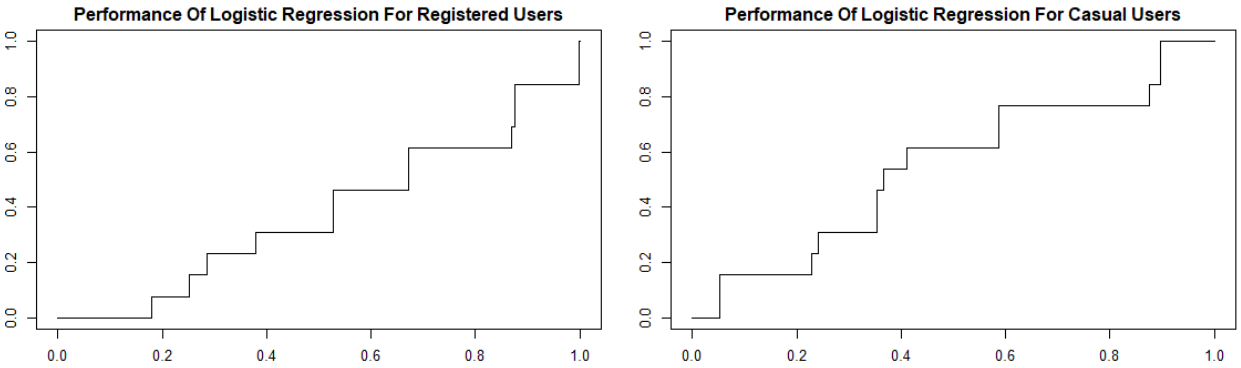


Figure 3.1 Performance Of Logistic Regression Using ROC Curve

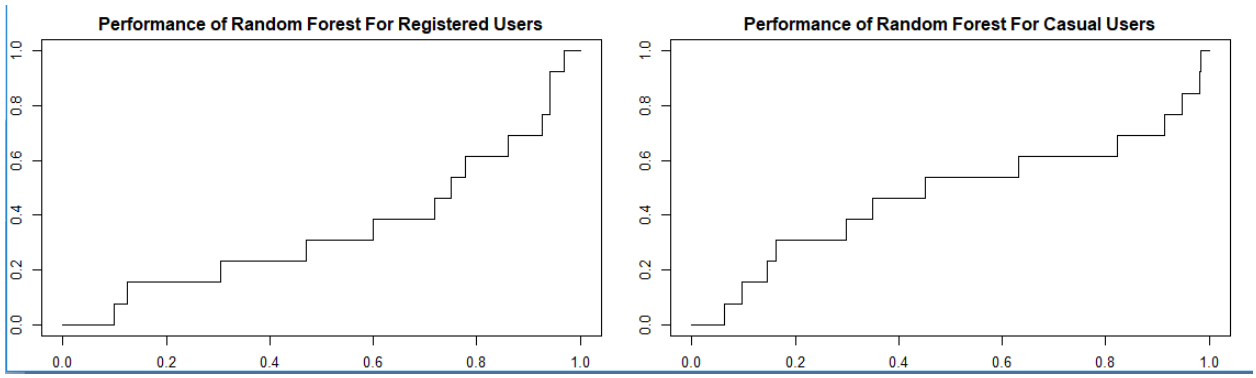


Figure 3.2 Performance Of Random Forest Using ROC Curve

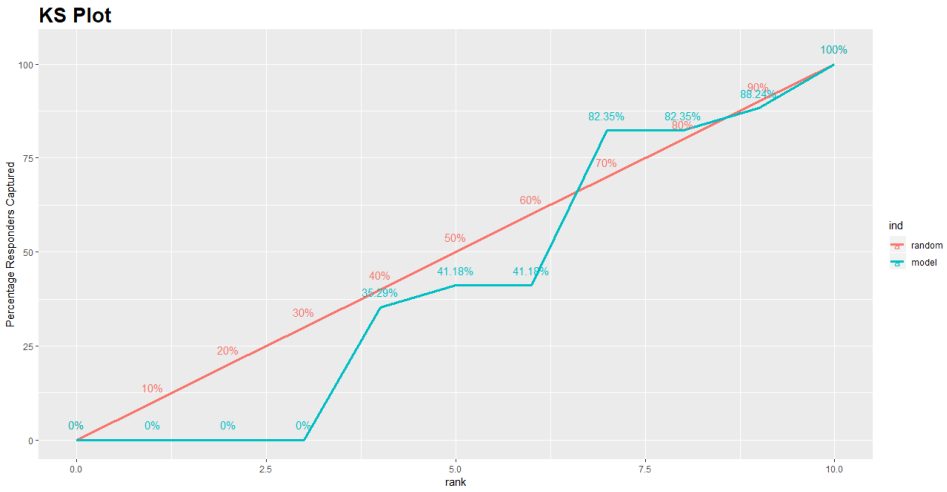


Figure 3.3 ks-plot For Registered Users

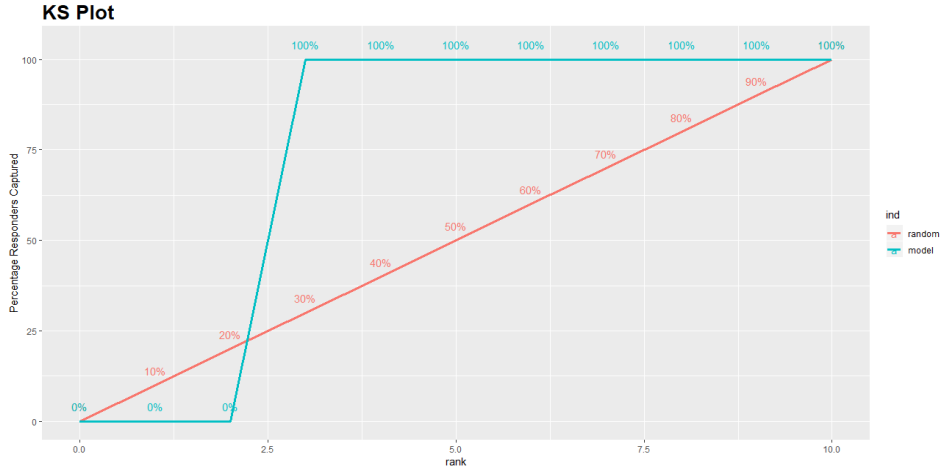


Figure 3.4 ks-plot For Casual Users

Appendix B - R Code

Complete R File

```
#Remove The Variable & Values In Environment
rm(list=ls())

#Set The Directory
setwd("F:/Eddwisor/Task Program/Projects/Second Project Works")

#Get The Directory
getwd()

#------Load the dataset-------#
df_data=read.csv("day.csv",encoding = 'ISO-8859-1')

#-----Check Dimension-------#
dim(df_data)              #rows=731 and column=16

#Get the Column Names
names(df_data)

#Get the structure of the dataset
str(df_data)
head(df_data,4)

#The DataSet contain 12 independent variables and 3 dependent variables
#Split the dataset into train and test dataset
library(caret)
intrain<-createDataPartition(y=df_data$cnt,p=0.7,list=FALSE)
train<-df_data[intrain,]
test<-df_data[-intrain,]
```

```
#------Check Dimension of Train & Test Dataset ----------#
dim(train)              #rows=515  and column=16

dim(test)               #rows=216  and column=16

#Get The Column Names of Train & Test Dataset
names(train)
names(test)

#-------------------------------------------------------------------
#No need of instant for Bike Renting
#get the index of column instant and remove it from the train dataset

instant_index=match("instant",names(train))
instant_index
train=train[-instant_index]

#After Removing The columns Check The train Dataset
str(train)
#Train dataset have 4 num variables, 10 int and 1 factor
#-------------------------------------------------------------------

#-------------------------------------------------------------------
#No need of instant for Bike Renting
#get the index of column instant and remove it from the test dataset

instant_index=match("instant",names(test))
instant_index
test=test[-instant_index]
```

```
#In test dataset no need of dependent variables
#So we get the index of columns and remove it from the test dataset

casual_index=match("casual",names(test))
casual_index
test=test[-casual_index]

registered_index=match("registered",names(test))
registered_index
test=test[-registered_index]

cnt_index=match("cnt",names(test))
cnt_index
test=test[-cnt_index]

#After Removing The columns Check The test Dataset
str(test)
#Train dataset have 4 num variables, 7 int and 1 factor
#-------------------------------------------------------------------

#Combine both Train and Test Data set(to understand the distribution of independent varia

test$registered=0
test$casual=0
test$cnt=0
data=rbind(train,test)

#Get the Structure Of Dataset
str(data)
```

```r
#Check for the Missing Value
sum(is.na(data))                          #no missing value is avaliable

#Plot the histogram of each numerical variables and analyze the distribution
par(mfrow=c(2,2))
par(mar = rep(2, 4))
hist(data$season)
hist(data$weather)
hist(data$hum)
hist(data$holiday)
hist(data$workingday)
hist(data$temp)
hist(data$atemp)
hist(data$windspeed)

prop.table(table(data$weathersit))

#Convert discrete variables into factor (season, weather, holiday, workingday)

data$season=as.factor(data$season)
data$weathersit=as.factor(data$weathersit)
data$holiday=as.factor(data$holiday)
data$workingday=as.factor(data$workingday)

#Outliers Detection using Multivariate Analysis
#Here I have added some additional hypothesis from the dataset. Let's test the
```

```r
# 1. Hourly Trend:
#We don't have variable 'hour' with us right now.But we can extract it using datetime(dteday)column.

data$hour=substr(data$dteday,12,13)
data$hour=as.factor(data$hour)

#Let's plot the hourly trend of count over hours and check if our hypothesis is correct or not.
#We will separate train and test data set from combined one.

train=data[as.integer(substr(data$dteday,9,10))<20,]      #training dataset is for the first 19 days of each month
test=data[as.integer(substr(data$dteday,9,10))>19,]       #Test dataset is from 20th days to month ends.

boxplot(train$cnt ~ train$hour,xlab="hour",ylab="count of users",main="count of users Vs.hour")

#Distribution Of Registered and Casual Users Seperately
boxplot(train$registered ~ train$hour,xlab="hour",ylab="Registered users",main="Registered Users Vs.hour")
boxplot(train$casual ~ train$hour,xlab="hour",ylab="Casual users",main="Casual Users Vs.hour")
#Above you can see that registered users have similar trend as count. Whereas, casual users have different trend.
#Thus, we can say that 'hour' is significant variable and our hypothesis is 'true'.


#########################################################################
# 2. Daily Trend:
# Like Hour,we will generate a variable for day from datetime(dteday)variable and after that we'll plot it.

date=substr(data$dteday,1,10)
days<-weekdays(as.Date(date))
data$day=days

#Plot the Registered and casual Users seperately
boxplot(data$registered ~ data$day,xlab="day",ylab="registered users",main="Registered users vs. day")
boxplot(data$casual ~ data$day,xlab="day",ylab="casual users",main="Causal users vs.day")

#While looking at the plot, I can say that the demand of causal users increases over weekend.
#########################################################################

.........................................................................
# 3.Rain :
#We don't have the 'rain' variable with us but have 'weathersit' which is sufficient to test our hypothesis

data$rain=substr(data$weathersit,12,13)
data$rain=as.factor(data$rain)

#train=data[as.integer(substr(data$weathersit,9,10))<20,]
#test=data[as.integer(substr(data$weathersit,9,10))>19,]

boxplot(data$registered ~ data$rain,xlab="rain",ylab="registered users",main="Registered Users Vs.Rain")
boxplot(data$casual ~ data$rain,xlab="rain",ylab="casual users",main="Casual Users Vs. Rain")

#It is clearly satisfying our hypothesis.
#########################################################################

# 4.Temperature, Windspeed and Humidity :
# These are continuous variables so we can look at the correlation factor to validate hypothesis.

sub=data.frame(train$registered,train$casual,train$cnt,train$temp,train$hum,train$atemp,train$windspeed)
cor(sub)

#Variable temp is positively correlated with dependent variables (casual is more compare to registered)
#Variable atemp is highly correlated with temp.
#Windspeed has lower correlation as compared to temp and humidity.
#########################################################################

# 5.Year :
#We have a yr variable to hypothesis the varaibles.
boxplot(data$cnt ~ data$yr,xlab="year",ylab="count",main="Count Vs.Year")

#Here 0 represent 2011 and 1 represent 2012
#You can see that 2012 has higher bike demand as compared to 2011.
#########################################################################
```

```
# 6.Pollution & Traffic :
#We don't have the variable related with these metrics in our data set so we cannot test this hypothesis.
```

```
#########################Feature Engineering############################
train$hour = as.integer(train$hour)           #Convert hour to integer
test$hour = as.integer(test$hour)             #Modifying in both train and test data set

#We use the library rpart for decision tree algorithm.
library(rpart)
library(rattle)                 #These libraries will be used to get a good visual plot for the decision model
library(rpart.plot)
library(RColorBrewer)

hour_reg=rpart(registered~hour,data=train)
summary(hour_reg)
fancyRpartPlot(hour_reg)

hour_cas=rpart(casual~hour,data=train)
summary(hour_cas)
fancyRpartPlot(hour_cas)

#We have created bins for temperature for both registered and casual users

temp_reg=rpart(registered~temp,data=train)
summary(temp_reg)
fancyRpartPlot(temp_reg)

temp_cas=rpart(casual~temp,data=train)
summary(temp_cas)
fancyRpartPlot(temp_cas)
```

```
#We have created bins for temperature for both registered and casual users

temp_reg=rpart(registered~temp,data=train)
summary(temp_reg)
fancyRpartPlot(temp_reg)

temp_cas=rpart(casual~temp,data=train)
summary(temp_cas)
fancyRpartPlot(temp_cas)

#We have created bins for year for both registered and casual users

year_reg=rpart(registered~yr,data = train)
summary(year_reg)
fancyRpartPlot(year_reg)

year_cas=rpart(casual~yr,data = train)
summary(year_cas)
fancyRpartPlot(year_cas)

#We have created bins for day for both registered and casual users
day_reg=rpart(registered~day,data = data)
summary(day_reg)
fancyRpartPlot(day_reg)

day_cas=rpart(casual~day,data = data)
summary(day_cas)
fancyRpartPlot(day_cas)
```

```r
#Before executing random forest,execute following steps
#Convert discrete variables into factor(weathersit,season,hour,holiday,workingday,mnth)

train$hour = as.factor(train$hour)
test$hour = as.factor(test$hour)

train$weathersit = as.factor(train$weathersit)
test$weathersit  = as.factor(test$weathersit)

train$season = as.factor(train$season)
test$season = as.factor(test$season)

train$holiday = as.factor(train$holiday)
test$holiday = as.factor(test$holiday)

train$workingday = as.factor(train$workingday)
test$workingday = as.factor(test$workingday)

train$mnth = as.factor(train$mnth)
test$mnth = as.factor(test$mnth)


#log transformation for some skewed variables, which can be seen from their distribution.
train$reg1=train$registered+1
train$cas1=train$casual+1
train$logcas=log(train$cas1)
train$logreg=log(train$reg1)
test$logreg=0
test$logcas=0
```

```r
#predicting the log of registered users.
library(randomForest)
set.seed(415)
fit1 <- randomForest(logreg ~ hour +workingday+holiday+hum+atemp+windspeed+season+weathersit+yr, data=train
pred1=predict(fit1,test)
test$logreg=pred1
print(fit1)
plot(fit1,main="Random Forest For Registered Users")


#predicting the log of casual users.
set.seed(415)
fit2 <- randomForest(logcas ~hour +hum+atemp+windspeed+season+weathersit+holiday+workingday+yr, data=train,
pred2=predict(fit2,test)
test$logcas=pred2
print(fit2)
plot(fit2,main="Random Forest For Casual Users")

#Re-transforming the predicted variables and then writing the output of count to the file submit.csv
#creating the final submission file
test$registered=exp(test$logreg)-1
test$casual=exp(test$logcas)-1
test$cnt=test$casual+test$registered
s<-data.frame(dteday=test$dteday,count=test$cnt)
write.csv(s,file="submit.csv",row.names=FALSE)
```

```
#########################Logistic Regression#############################
#Perform Logistic Regression for Registered Users
logit_model_1 <-glm(logreg ~ season+mnth+holiday+workingday+weathersit, data=train)
pred_model_1 <-predict(logit_model_1,test)
test$logreg=pred_model_1
print(logit_model_1)
plot(logit_model_1,main="Logistic Regression for Registered Users")

#Perform Logistic Regression For Casual Users
logit_model_2 <-glm(logcas ~season+mnth+holiday+workingday+weathersit,data=train)
pred_model_2 <-predict(logit_model_2,test)
test$logcas=pred_model_2
print(logit_model_2)
plot(logit_model_2,main="Logistic Regression For Casual Users")

y_pred <- ifelse(pred_model_1 > 0.5, 1, 0)
y_act <- test$logreg
```

```
#######################Evaluation Of Model#############################
###Computing a simple ROC curve (x-axis: fpr, y-axis: tpr)
# 1.Logistic Regression
# Performance Evaluation Of Logistic Regression for Registered Users
library(ROCR)
pred_logit_reg<- prediction(predict(logit_model_1), train$holiday)
perf_logit_reg <- performance(pred_logit_reg,"tpr","fpr")
plot(perf_logit_reg,main="Performance Of Logistic Regression For Registered Users")

#Performance Evaluation Of Logistic Regression for Casual Users
pred_logit_cas<- prediction(predict(logit_model_2), train$holiday)
perf_logit_cas <- performance(pred_logit_cas,"tpr","fpr")
plot(perf_logit_cas,main="Performance Of Logistic Regression For Casual Users")
```

```
# 2.Random Forest
#Performance Evaluation Of Random Forest for Registered Users
pred_randomforest_reg <-prediction(predict(fit1),train$holiday)
perf_randomforest_reg <-performance(pred_randomforest_reg,"tpr","fpr")
plot(perf_randomforest_reg,main="Performance of Random Forest For Registered Users")

#Performance Evaluation Of Random Forest for Casual Users
pred_randomforest_cas <-prediction(predict(fit2),train$holiday)
perf_randomforest_cas <-performance(pred_randomforest_cas,"tpr","fpr")
plot(perf_randomforest_cas,main="Performance of Random Forest For Casual Users")

#we have here is a line that traces the probability cutoff from 1 at the bottom-left to 0 in the top right.
#This is a way of analyzing how the sensitivity and specificity perform for the full range of probability cutoffs,
#that is from 0 to 1.
```

```r
#Evaluation Using Concordance & Discordance
install.packages("InformationValue")  # For stable CRAN version

#Performance Evaluation of Registered Users Using Concordance $ Discordance
InformationValue::Concordance(test$logreg, test$holiday)

##Performance Evaluation of Casual Users Using Concordance $ Discordance
InformationValue::Concordance(test$logcas,test$holiday)

#we take all possible combinations of true events and non-events.
#Concordance is the percentage of pairs,
#where true event's probability scores are greater than the scores of true non-events.
#Here no pairing is given for calculating the concordance.
```

```r
#Evaluation Using ks-plot of Registered Users
InformationValue::ks_plot(test$logreg,test$holiday)

#Evaluation Using ks-plot of Casual Users
InformationValue::ks_plot(test$logcas,test$holiday)

#The KS chart and statistic that is widely used in credit scoring scenarios
#and for selecting the optimal population size of target users for marketing campaigns.

#-------------------------------------------------------------------------------------------

#Evaluation Using ks-stat of Registered Users
InformationValue::ks_stat(test$logreg,test$holiday)

InformationValue::ks_stat(test$logreg,test$holiday,returnKSTable = T)

#Evaluation Using ks-stat of Casual Users
InformationValue::ks_stat(test$logcas,test$holiday)

InformationValue::ks_stat(test$logcas,test$holiday,returnKSTable = T)

#The significance of KS statistic is, it helps to understand,
#what portion of the population should be targeted to get the highest response rate (1's).
```

```r
#For Recall,Precision and fscore we have to convert variables into factor
test$workingday  =  as.factor(test$workingday)
test$holiday = as.factor(test$holiday)

#Evaluation Using Recall,Precision and F-score
recall(test$workingday,test$holiday)                #recall = 29.21
precision(test$workingday,test$holiday)             #precision = 90.69

#F1 Score = (2 * Precision * Recall) / (Precision + Recall)
f1_score = (2 * 90.69 * 29.21)/(90.69 + 29.21)      #f1_score = 44.18
f1_score

#You have an F1 Score of 44.18 percent.That's not so good.
#A good model should have a good precision as well as a high recall.
#So ideally, I want to have a measure that combines both these aspects in one single metric – the F1 Score.

#-------------------------------------------------------------------------------------------

#Evalution Performance Using Confusion Matrix
library(caret)
caret::confusionMatrix(test$workingday, test$holiday, positive="1", mode="everything")

#The rows in the confusion matrix are the count of predicted 0's and 1's (from y_pred),
#while, the columns are the actuals (from y_act).
```