Team Name: **MASK**

Team Members: Chinnasubbareddygari Mohan Reddy- A0163433L
Anusuya Manickavasagam- A0163300Y
Kesavan Sridharan- A0163207M

Model Building Workshop

Dataset used: Training : day.csv

Test : day.csv

Problem statement: To build predictive models(single and ensemble) to predict demand for a bicycle sharing scheme.

Model used: Random forest, Decision Tree, neural network and Generalised Linear Models(Poisson and Negative Binomial) were used to predict the demand for bicycle demand.

Tool used: **"**Rstudio" was used to build predictive models on the dataset

Summary:

1. **Data selection and pre-processing-** In order to find the outliers, we do the box plot of the different variables. In the variables, humidity and windspeed, we find the outliers. But, these outliers do not affect our dataset. So, we do not make any change. As a next step, we try to find the missing values and remove them.
2. **Handling Missing values-** We find the missing values and omit them from our dataset.
3. Creating the lag variable- Since yesterday's count of bikes is used to predict tomorrow's demand we use a lag of two days to arrive at a derived column called "lagcnt".
4. **Dividing the data into training and test data**- We then divide the dataset into training and test datasets with training data having 2011 data and test data having 2012 records.
5. The following **single and ensemble models** were implemented and the models were compared.
   a) Generalised Linear model-Poisson Model
   b) Generalised Linear model- Negative Binomial Model
   c) Decision Tree
   d) Ensemble model- Random forest
   e) Neural network
6. **Training the model on the test data**- After the model was trained, the test data was used to predict the predicted values.
7. **Calculating the errors on various models**- The various errors like, Root Mean Square Error and Mean Absolute errors were calculated. The model performance was calculated by subtracting the predicted value from the actual value and was plotted against time.
8. **Selection of best model**-The various models were compared and the best model was selected.
9. **Calculation of business problems**- The various business problems were calculated. The revenue was calculated by taking the minimum of predicted demand and actual demand. The costs were calculated and thus the profit was derived by subtracting the revenue from the cost.

So the **derived columns**, rent,cost,profit were used for model comparison between the default model and the predicted model.

Variables:

Variable	Description Code/VALUE	Code/Values	Names
1	Instant	continuous	Instant
2	dteday	date	dteday
3	season	1:springer, 2:summer, 3:fall, 4:winter	season
4	year	0: 2011, 1:2012	yr
5	holiday	weather day is holiday or not	holiday
6	weekday	day of the week	weekday
7	workingday	holiday=1,others=0	workingday
8	weathersit	- 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	weathersit
9	temp	continuous	temp
10	atemp	continuous	atemp
11	hum	Continuous	hum
12	windspeed	Continuous	windspeed
13	casual	Continuous	casual
14	registered	Continuous	registered
15	cnt	Continuous	cnt

Model steps

1) Data Selection and Pre-processing

The data is read from the day.csv and the dataset is checked for null data.

Reading the data

```
setwd("D:\Lecture notes_17\EB5102\Assignment1")

daydata<- day.csv(file="salary-train.csv",header=T,na.strings=c(""))

nrow(daydata)

##731
```

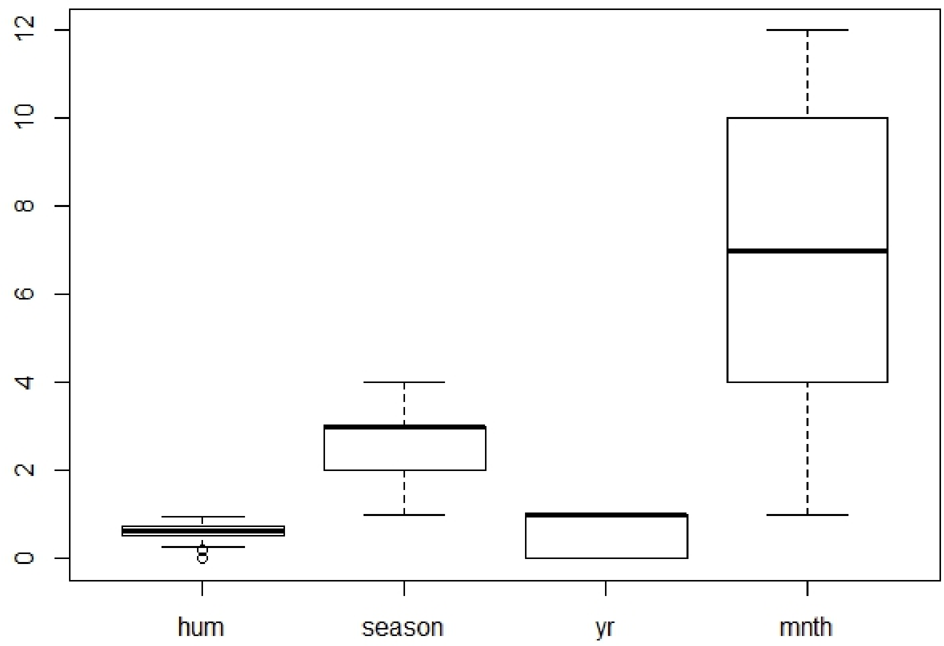
Checking for null data

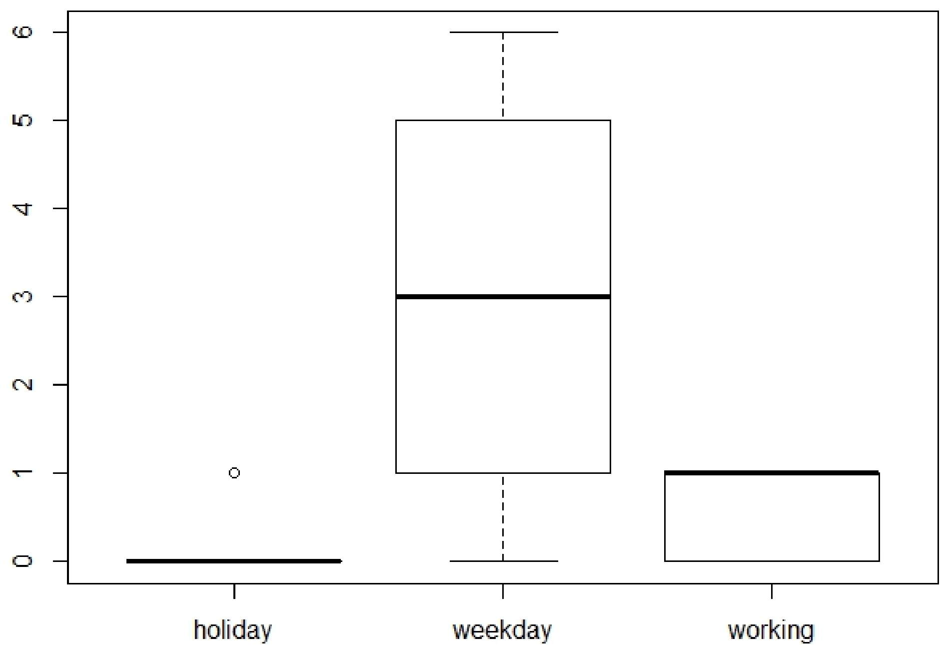
```
table(is.na(daydata))

#FALSE
```

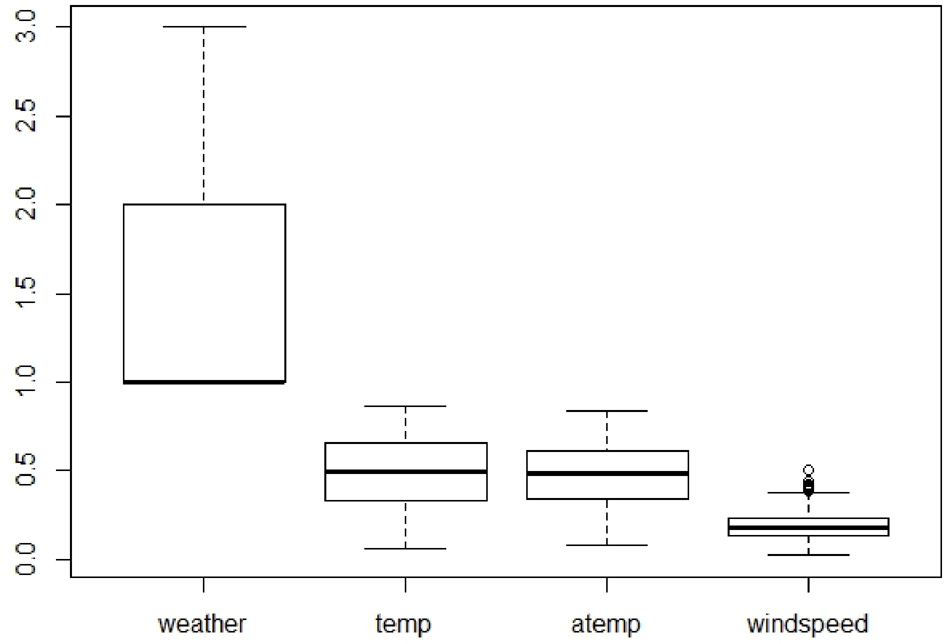
2) Feature Selection. Analysing the continuous variables by using boxplot

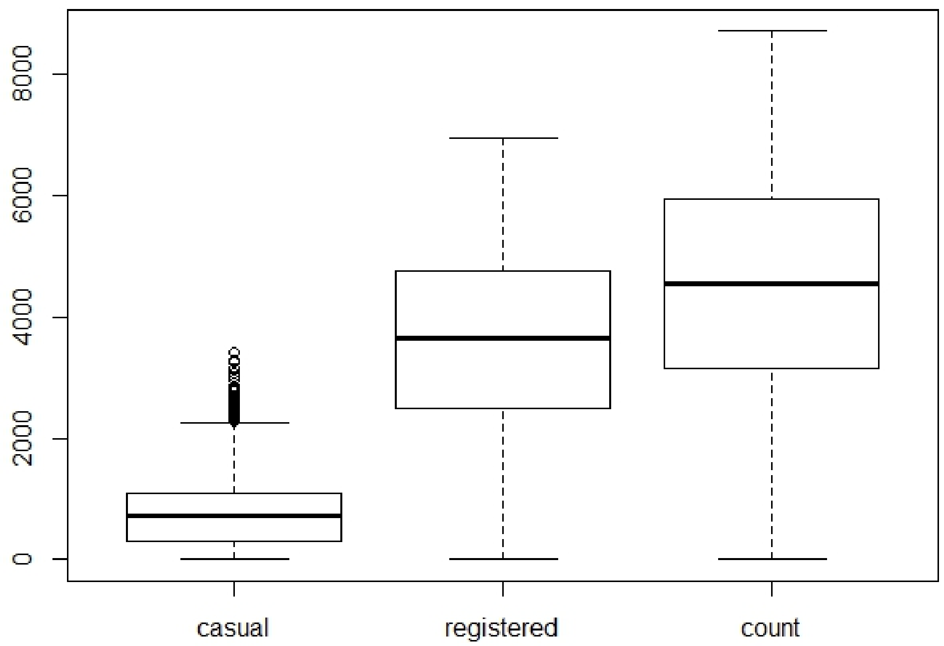
Next, we do a boxplot of all the continuous variables to analyse each variable and find out the outliers.

```
boxplot(daydata$hum,daydata$season,daydata$yr,daydata$mnth,names=c("hum","season","yr","mnth"))

boxplot(daydata$holiday,daydata$weekday,daydata$workingday,names=c("holiday","weekday","working"))

boxplot(daydata$weathersit,daydata$temp,daydata$atemp,daydata$windspeed,names=c("weather","temp","windspeed"))

boxplot(daydata$casual,daydata$registered,daydata$cnt,names=c("casual","registered","count"))
```


0
2
4
6
8
10
12
hum
season
yr
mnth


0
1
2
3
4
5
6
holiday
weekday
working





On analysing the boxplots for different variables we find that humidity and windspeed have few outliers. So, we examine the outliers of **humidity** and **windspeed**.

```
> boxplot.stats(daydata$hum)
$stats
[1] 0.2541670 0.5200000 0.6266670 0.7302085 0.9725000

$n
[1] 731

$conf
[1] 0.6143827 0.6389513

$out
[1] 0.187917 0.000000

> boxplot.stats(daydata$windspeed)
$stats
[1] 0.0223917 0.1349500 0.1809750 0.2332145 0.3781080

$n
[1] 731

$conf
[1] 0.1752326 0.1867174

$out
 [1] 0.417908 0.507463 0.385571 0.388067 0.422275 0.415429 0.409212 0.421642 0.441563 0.414800 0.386821 0.398008 0.407346

> |
```

But on further analysing the individual records it is found that these are single records and do not affect our model.

Removed duplicate variables:

i) Season and Month has are highly correlated, so we removed month from our model building
ii) Temp and atemp are highly correlated so, we removed atemp from our model building
iii) Removed "yr" as in our data has only one year for each train and test data.

#checking for correlation between temp and atemp

cor(traindata $temp, traindata $atemp)

0.9964765

#checking for correlation between month and season

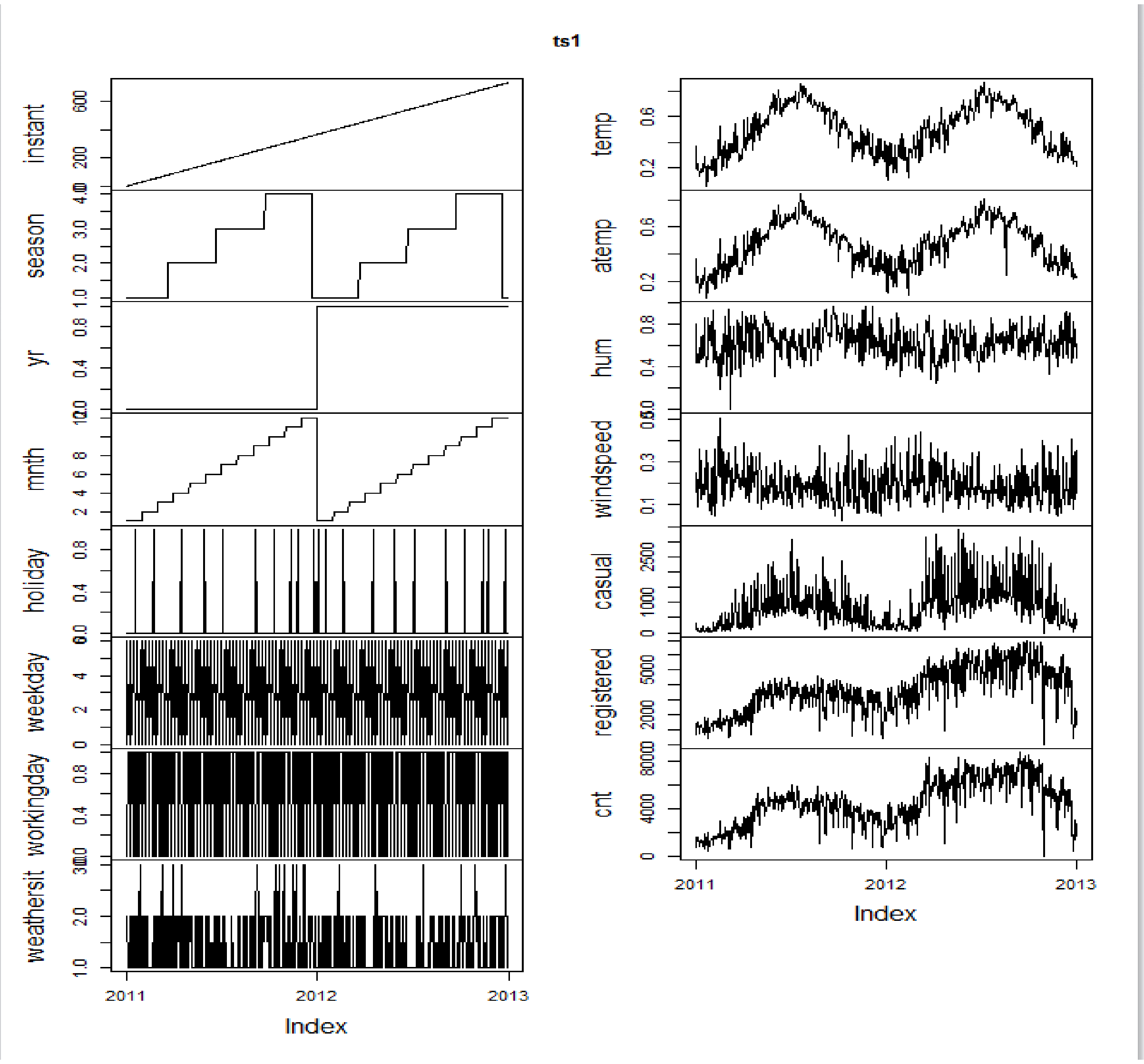
cor(traindata$season,traindata$mnth)

0.8310321

3) Creating the lag variable

In our problem statement since tomorrow's demand is predicted using yesterday's data we create a lag of 2 days,creating a variable called lagcnt.

```
library(zoo)

dates=daydata$dteday

daydata$dteday=NULL

ts1=zoo(daydata,as.Date(dates,"%Y-%m-%d"))

head(ts1,30)

plot(ts1)
```



```
#creating the lag variable by 2 days

lagcnt<- lag(ts1$cnt,k=-2)

ts2<-merge(ts1,lagcnt)
```

```
> head(ts2,7)
           instant season yr mnth holiday weekday workingday weathersit     temp    atemp      hum windspeed casual registered  cnt lagcnt
2011-01-01       1      1  0    1       0       6          0          2 0.344167 0.363625 0.805833 0.1604460    331        654  985     NA
2011-01-02       2      1  0    1       0       0          0          2 0.363478 0.353739 0.696087 0.2485390    131        670  801     NA
2011-01-03       3      1  0    1       0       1          1          1 0.196364 0.189405 0.437273 0.2483090    120       1229 1349    985
2011-01-04       4      1  0    1       0       2          1          1 0.200000 0.212122 0.590435 0.1602960    108       1454 1562    801
2011-01-05       5      1  0    1       0       3          1          1 0.226957 0.229270 0.436957 0.1869000     82       1518 1600   1349
2011-01-06       6      1  0    1       0       4          1          1 0.204348 0.233209 0.518261 0.0895652     88       1518 1606   1562
2011-01-07       7      1  0    1       0       5          1          2 0.196522 0.208839 0.498696 0.1687260    148       1362 1510   1600
> 
```

4) Dividing the data into training and test

Now, we divide the data into training and test sets.

```
#splitting the data into training and test

s1 = as.Date("01-JAN-2011", "%d-%b-%Y")

e1 = as.Date("31-DEC-2011", "%d-%b-%Y")

s2 = as.Date("01-JAN-2012", "%d-%b-%Y")

e2= as.Date("31-DEC-2012", "%d-%b-%Y")


traindata = window(ts2,start=s1, end=e1)

testdata = window(ts2, start=s2, end=e2)
```

From the boxplot we had done earlier for the different variables we have found that weather,season,holiday and working day are factors. So, we convert these integers to factors.

```
#converting weather,season,holiday,working day into factor

train_factor<- as.data.frame(traindata)

train_factor$weathersit<-factor(train_factor$weathersit)

train_factor$season<-factor(train_factor$season)

train_factor$holiday<-factor(train_factor$holiday)

train_factor$workingday<-factor(train_factor$workingday)
```

Next, we omit the null values present in the dataframe, train_factor.

```
#removing the null from the data

train_factor1<-na.omit(train_factor)
```

Finding out if the variables temp and atemp are correlated

```
cor(train_factor1$temp,train_factor1$atemp)

0.9964765
```

We find the value is very much close to 1 and highly **correlated**. So, we remove the variable **atemp** while fitting model.

5) Model building and testing-Poisson Model

Our response variable (y) is the cnt column and the various predictors are the other x's. Since the cnt is a count column first we try to fit **Poisson model.**

```
poi.mod <- glm(train_factor1$cnt ~  train_factor1$season + train_factor1$holiday +
train_factor1$workingday + train_factor1$weathersit + train_factor1$temp +
train_factor1$hum + train_factor1$windspeed, family = poisson, data = traindata)

summary(poi.mod)
```

```
> summary(poi.mod)

Call:
glm(formula = train_factor1$cnt ~ train_factor1$season + train_factor1$holiday +
    train_factor1$workingday + train_factor1$weathersit + train_factor1$temp +
    train_factor1$hum + train_factor1$windspeed, family = poisson,
    data = traindata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-46.475   -6.781    0.491    7.157   28.990

Coefficients:
                          Estimate Std. Error  z value Pr(>|z|)
(Intercept)               7.396347   0.006997 1057.044   <2e-16 ***
train_factor1$season2     0.487558   0.003940  123.750   <2e-16 ***
train_factor1$season3     0.433087   0.004776   90.685   <2e-16 ***
train_factor1$season4     0.616738   0.003517  175.357   <2e-16 ***
train_factor1$holiday1   -0.125734   0.006023  -20.876   <2e-16 ***
train_factor1$workingday1  0.002319   0.002014    1.151     0.25
train_factor1$weathersit2 -0.090145   0.002444  -36.880   <2e-16 ***
train_factor1$weathersit3 -0.692521   0.007045  -98.306   <2e-16 ***
train_factor1$temp        1.225957   0.008704  140.854   <2e-16 ***
train_factor1$hum        -0.227555   0.008812  -25.822   <2e-16 ***
train_factor1$windspeed  -0.616362   0.013432  -45.886   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 223583  on 362  degrees of freedom
Residual deviance:  47196  on 352  degrees of freedom
AIC: 50798

Number of Fisher Scoring iterations: 4

>
```

We, then fit the model to testdata

```
#converting weather,season,holiday,working day into factor in testdata
testdata$weathersit<-factor(testdata$weathersit)
testdata$season<-factor(testdata$season)
testdata$holiday<-factor(testdata$holiday)
testdata$workingday<-factor(testdata$workingday)
```

```
poi.test <- glm(testdata$cnt ~  testdata$season + testdata$holiday + testdata$workingday +
testdata$weathersit + testdata$temp +  testdata$hum + testdata$windspeed, family =
poisson, data = testdata)


summary(poi.test)
```

```
Console ~/
> summary(poi.test)

Call:
glm(formula = testdata$cnt ~ testdata$season + testdata$holiday +
    testdata$workingday + testdata$weathersit + testdata$temp +
    testdata$hum + testdata$windspeed, family = poisson, data = testdata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-87.819   -7.474    1.122    9.414   48.896

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)          8.4206368  0.0079907 1053.80   <2e-16 ***
testdata$season      0.0957334  0.0007095  134.94   <2e-16 ***
testdata$holiday    -0.1949791  0.0047568  -40.99   <2e-16 ***
testdata$workingday  0.0404701  0.0015731   25.73   <2e-16 ***
testdata$weathersit -0.1352329  0.0018435  -73.36   <2e-16 ***
testdata$temp        1.0660598  0.0043112  247.28   <2e-16 ***
testdata$hum        -0.2870769  0.0072199  -39.76   <2e-16 ***
testdata$windspeed  -0.5809375  0.0101462  -57.26   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 242557  on 365  degrees of freedom
Residual deviance:  95614  on 358  degrees of freedom
AIC: 99431

Number of Fisher Scoring iterations: 4

> |
```

```
model1<-predict(poi.test,testdata$cnt,type="response")
```

The model has residual deviance of 95614 and 358 degrees of freedom which accounts for 267 and has **over dispersion**.

```
#Checking for overdispersion using package in r

library(AER)

dispersiontest(poi.test,trafo=1)
```

```
Console ~/
> dispersiontest(poi.test,trafo=1)

        Overdispersion test

data:  poi.test
z = 10.448, p-value < 2.2e-16
alternative hypothesis: true alpha is greater than 0
sample estimates:
   alpha
233.8769

>
```

The **overdispersion** test value is 233.87 (c>1) which proves that there is overdispersion in the dataset.

Calculating model errors- Poisson Model

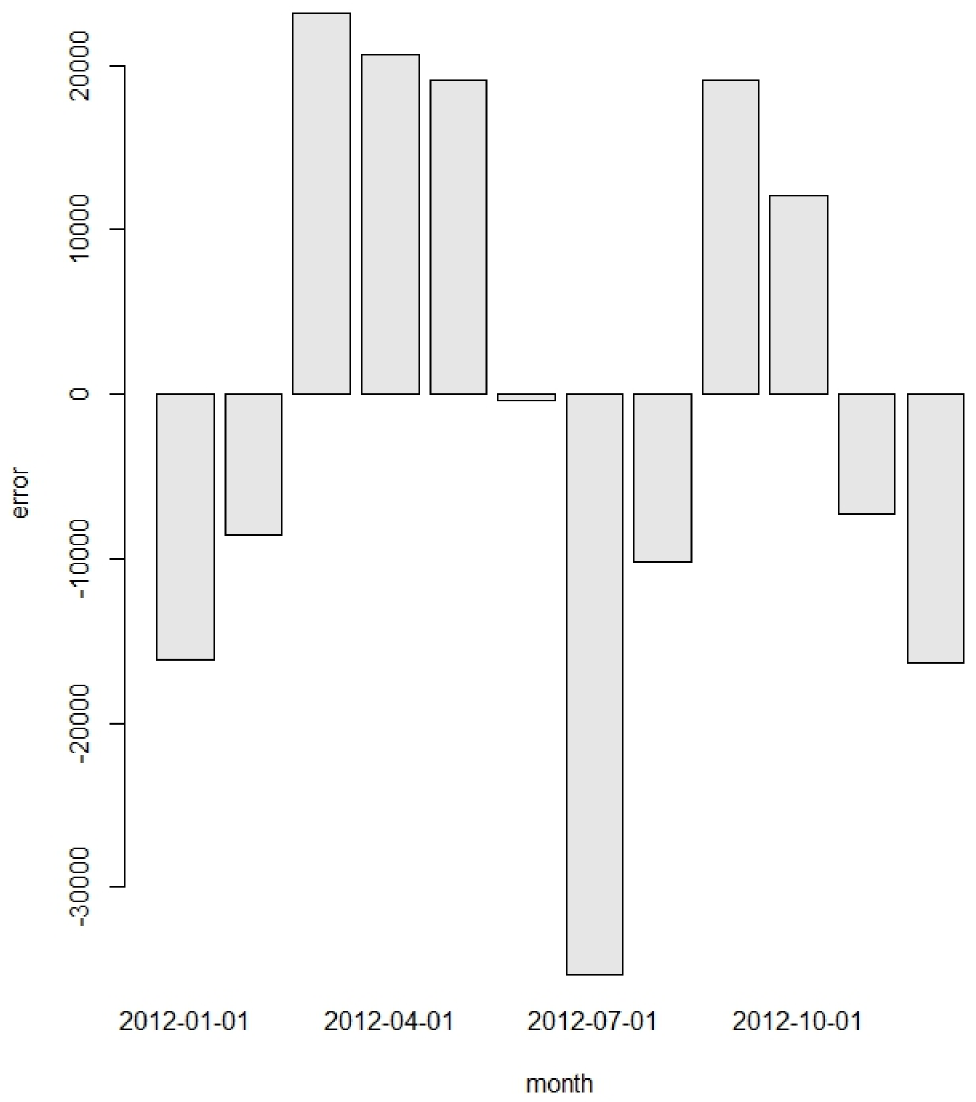
```
RMSE.glm <- sqrt((mean((as.numeric(Prediction)-
as.numeric(testdata$cnt))^2))/nrow(testdata))

RMSE.glm

#306.8119
```

```
MAD.glm <- sum(abs(as.numeric(Prediction)-
as.numeric(testdata$cnt)))/length(testdata$cnt)

MAD.glm

#5591.341
```

The **root mean square error** is **306.8119** and **mean absolute deviation** is **5591.341**

Visualising error over time

```
e1 <- ((testdata$cnt)-(model1))

t1<- aggregate(e1 ~ testdata$mnth, testdata, sum)

t1 <- do.call(cbind, t1)

barplot(t1,xlab="month",ylab="error")
```



On plotting error over the time period, we find that the model performance varies with respect to month.

6) Negative Binomial Model-Model Building and Testing

To overcome the over dispersion we now use the **negative binomial model,** since negative binomial model assumes that variance is quadratic function of the mean. We now fit negative binomial model using all the variables.

```
library(MASS)

glm.nb.mod <- glm.nb(train_factor1$cnt ~ train_factor1$season + train_factor1$holiday +
train_factor1$workingday + train_factor1$weathersit + train_factor1$temp +
train_factor1$atemp + train_factor1$hum+ train_factor1$windspeed, data =
train_factor1,link=log)


summary(glm.nb.mod)
```

```
Console ~/
> summary(glm.nb.mod)

Call:
glm.nb(formula = train_factor1$cnt ~ train_factor1$season + train_factor1$holiday +
    train_factor1$workingday + train_factor1$weathersit + train_factor1$temp +
    train_factor1$atemp + train_factor1$hum + train_factor1$windspeed,
    data = train_factor1, link = log, init.theta = 20.30911439)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.8473  -0.5715   0.0891   0.5234   2.7330

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)                7.27696    0.08837  82.348  < 2e-16 ***
train_factor1$season2      0.43122    0.04432   9.730  < 2e-16 ***
train_factor1$season3      0.37112    0.05824   6.372 1.86e-10 ***
train_factor1$season4      0.59260    0.03935  15.060  < 2e-16 ***
train_factor1$holiday1    -0.13842    0.07437  -1.861 0.062710 .
train_factor1$workingday1  0.03729    0.02637   1.414 0.157373
train_factor1$weathersit2 -0.08357    0.03130  -2.670 0.007586 **
train_factor1$weathersit3 -0.70457    0.06945 -10.145  < 2e-16 ***
train_factor1$temp         0.41340    0.80275   0.515 0.606563
train_factor1$atemp        1.20423    0.88924   1.354 0.175665
train_factor1$hum         -0.28314    0.10896  -2.599 0.009360 **
train_factor1$windspeed   -0.64614    0.17029  -3.794 0.000148 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(20.3091) family taken to be 1)

    Null deviance: 1630.20  on 362  degrees of freedom
Residual deviance:  367.01  on 351  degrees of freedom
AIC: 5799.7

Number of Fisher Scoring iterations: 1


              Theta:  20.31
          Std. Err.:  1.51

 2 x log-likelihood:  -5773.66
>
```

Removing **atemp** and applying the model.

```
glm.nb.mod2 <- glm.nb(train_factor1$cnt ~ train_factor1$season + train_factor1$holiday +
train_factor1$workingday + train_factor1$temp + train_factor1$hum +
train_factor1$weathersit + train_factor1$windspeed, data = train_factor1,link=log)

summary(glm.nb.mod2)
```

```
Console ~/
> summary(glm.nb.mod2)

Call:
glm.nb(formula = train_factor1$cnt ~ train_factor1$season + train_factor1$holiday +
    train_factor1$workingday + train_factor1$temp + train_factor1$hum +
    train_factor1$weathersit + train_factor1$windspeed, data = train_factor1,
    link = log, init.theta = 20.21817107)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.8089  -0.5792   0.0806   0.5136   2.8292

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)                7.31777    0.08303  88.137  < 2e-16 ***
train_factor1$season2      0.43239    0.04442   9.735  < 2e-16 ***
train_factor1$season3      0.36352    0.05812   6.255 3.97e-10 ***
train_factor1$season4      0.59748    0.03931  15.198  < 2e-16 ***
train_factor1$holiday1    -0.14861    0.07421  -2.002  0.04523 *
train_factor1$workingday1  0.03699    0.02643   1.400  0.16163
train_factor1$temp         1.48808    0.11202  13.284  < 2e-16 ***
train_factor1$hum         -0.26778    0.10875  -2.462  0.01380 *
train_factor1$weathersit2 -0.08747    0.03124  -2.800  0.00511 **
train_factor1$weathersit3 -0.71400    0.06925 -10.310  < 2e-16 ***
train_factor1$windspeed   -0.69221    0.16679  -4.150 3.32e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Negative Binomial(20.2182) family taken to be 1)

    Null deviance: 1622.96  on 362  degrees of freedom
Residual deviance:  367.03  on 352  degrees of freedom
AIC: 5799.3

Number of Fisher Scoring iterations: 1


              Theta:  20.22
          Std. Err.:  1.50

 2 x log-likelihood:  -5775.308
>
```

Next, we do a anova on both negative binomial models.

```
anova(glm.nb.mod,glm.nb.mod2)
```

```
> anova(glm.nb.mod,glm.nb.mod2)
Likelihood ratio tests of Negative Binomial Models

Response: train_factor1$cnt

                                 Model
1                        train_factor1$season + train_factor1$holiday + train_factor1$workingday + train_factor1$temp + train_factor1$hum + train_facto
r1$weathersit + train_factor1$windspeed
2 train_factor1$season + train_factor1$holiday + train_factor1$workingday + train_factor1$weathersit + train_factor1$temp + train_factor1$atemp + trai
n_factor1$hum + train_factor1$windspeed
     theta Resid. df    2 x log-lik.   Test    df LR stat.    Pr(Chi)
1 20.21817       352       -5775.308
2 20.30911       351       -5773.660 1 vs 2     1 1.647742 0.1992666
>
```

We get the p value as 0.199266 which is > 0.05. This means, we should not keep atemp in our model.

```
c(theta = summary(glm.nb.mod2)$theta, deviance = glm.nb.mod2$deviance, d.f =
glm.nb.mod2$df.residual)
```

```
Console ~/
> c(theta = summary(glm.nb.mod2)$theta, deviance = glm.nb.mod2$deviance, d.f = glm.nb.mod2$df.residual)
    theta  deviance       d.f
 20.21817 367.02553 352.00000
>
```

We get the value of **deviance** as **367.02** and **degree of freedom** as **352** so the dispersion is almost equal to 1. Hence, **negative binomial is better than Poisson model.**

Predicting for test data

```
glm.nb.mod2.test <- glm.nb(testdata$cnt ~  testdata$season + testdata$holiday +
testdata$workingday +  testdata$temp +  testdata$hum + testdata$weathersit
+ testdata$windspeed, data = testdata,link=log)


model2<-predict(glm.nb.mod2.test,testdata$cnt,type="response")
```

7) Calculating negative binomial model error

```
RMSE.nb <- sqrt((mean((as.numeric(model2)-
as.numeric(testdata$cnt))^2))/nrow(testdata))

RMSE.nb    #63.74
```

```
MAE.nb <- MAE.nb <- sum(abs(as.numeric(model2)-
as.numeric(testdata$cnt)))/length(testdata$cnt)

MAE.nb    #833.36
```

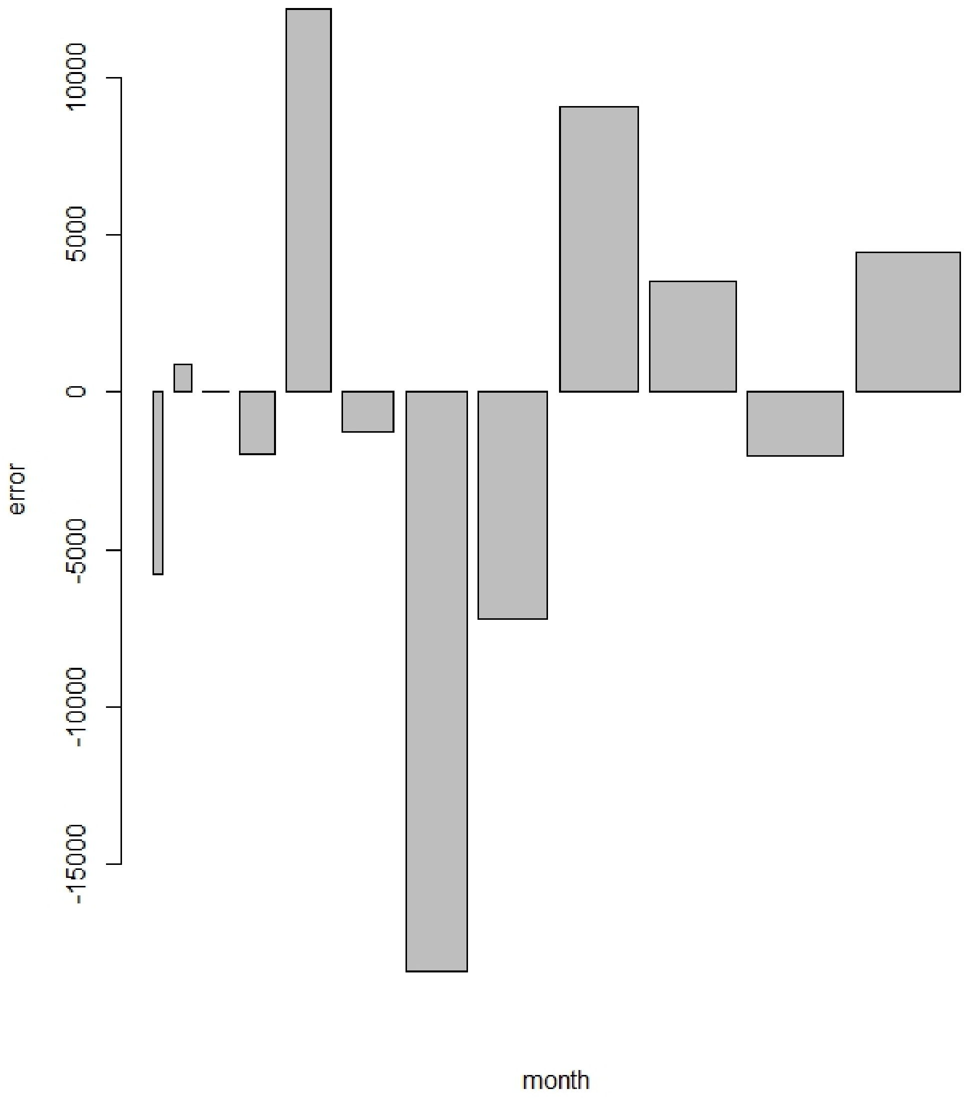
The **root mean square error** is **63.74** and **mean absolute deviation** is **833.36** which are less compared to the Poisson model errors.

Next, we calculate the **model error percentage.**

```
#calculating model error percentage

error=((sum(testdata$cnt)-sum(model2))/sum(testdata$cnt))*100

print(error)

#39.11
```

Next, we plot **error over time** using **negative binomial model**.

```
#plotting error over time

e2 <- ((testdata$cnt)-(model2))

t1<- aggregate(e2 ~ testdata$mnth, testdata, sum)

t1 <- do.call(cbind, t1)

barplot(t1,xlab="month",ylab="error")
```



We find that the error slowly increases over time and then decreases.

Business Performance using Negative Binomial Model

a) Using the given parameters: revenueperbike=$3 and loanperbike=$2 calculation of profit using existing model and negative binomial model

Default (Existing) model profit calculation:

```
dat<- pmin( testdata$cnt,testdata$lagcnt )
rentdf <- dat*3
costdf=testdata$lagcnt*2
profitdf=sum(rentdf)-sum(costdf)
tprofitdf=sum(profitdf)
profitdf
#1442972
```

```
#Default model profit as percentage of expenditure
costt<-sum(costdf)
expp=sum(tprofitdf/costt)*100
expp
#35.18
```

Negative binomial model profit calculation:

```
dat1<- pmin( testdata$cnt,model2 )
renttt<- dat1*3
costg=model2*2
profitg=renttt-costg
tprofitg=sum(profitg)
tprofitg
#1557088
```

- **What was your model profit for 2012 expressed as $ total?**
  The model profit for 2012 using negative binomial model is **$1557088.**

- **<u>What was your model profit expressed as a percentage of total expenditure?</u>**

```
#Negative Binomial model profit expressed as a percentage of total costs
PP=(tprofitg/sum(cost))*100
print(PP)
#62.39
```

The model profit expressed as a percentage of total expenditure is **<u>62.39%.</u>**

- **<u>What is the profit (total and percentage of expenditure) for the default prediction?</u>**

The profit for the default prediction is **<u>$1442972</u>** and as a percentage of expenditure is **<u>35.18%</u>**

- **<u>Under what conditions is your prediction model better than the default model? Is it</u>**

a) Always
b) Never
c) Only when revenue is high compared to costs(eg: $8 per rental vs $2 costs)
d) Only when revenue is low compared to costs (eg: $2.2 per rental vs $2 costs)
e) It's hard to say, you see no pattern.

<u>Calculation of profit for $8 rental and $2 costs</u>

```
Console ~/
> dat<- pmin( testdata$cnt,testdata$lagcnt )
> rentdf <- dat*8
> costdf=testdata$lagcnt*2
> profitdf=sum(rentdf)-sum(costdf)
> tprofitdf=sum(profitdf)
> profitdf
[1] 10683042
> dat1<- pmin( testdata$cnt,model2 )
> renttt<- dat1*8
> costg=model2*2
> profitg=renttt-costg
> tprofitg=sum(profitg)
> tprofitg
[1] 11034288
> 
```

Prediction model has **$11034288** and default model has **$10683042**

<u>Calculation of profit for $2.2 rental and $2 costs</u>

```
Console ~/
> dat<- pmin( testdata$cnt,testdata$lagcnt )
> rentdf <- dat*2.2
> costdf=testdata$lagcnt*2
> profitdf=sum(rentdf)-sum(costdf)
> tprofitdf=sum(profitdf)
> profitdf
[1] -35439.2
> dat1<- pmin( testdata$cnt,model2 )
> renttt<- dat1*2.2
> costg=model2*2
> profitg=renttt-costg
> tprofitg=sum(profitg)
> tprofitg
[1] 40736.52
>
```

Prediction model has profit of **$40736.52** and default model has **-$35439.2**

The prediction model has a profit at both $2.2 and $8 rental.

So the answer is a)always

- **<u>Did you find any evidence that model performance correlates with season or similar factor?</u>**

The coefficients for season2,season3 and season4 in the summary of the negative binomial model are positive indicating a positive correlation between season and the prediction variable.

- **<u>Did you find any evidence that your model performance decreases with age of the model?</u>**

  For negative Binomial model, the error varies with time. For a few months, the error is on the positive side and slightly high but for other months the error is less.

8) Decision Tree-Model Building and Testing

Our response variable (y) is the cnt column and the various predictors are the other x's. "tree" library is used to find Decision Trees.

```
library(rpart)

dt.fit = rpart(as.numeric(traindata$cnt)~ traindata$season + traindata$workingday +
traindata$weathersit  + traindata$temp + traindata$windspeed + traindata$casual +
traindata$registered,  data=daydata, method = "anova")pred=predict(fit,testdata)
```

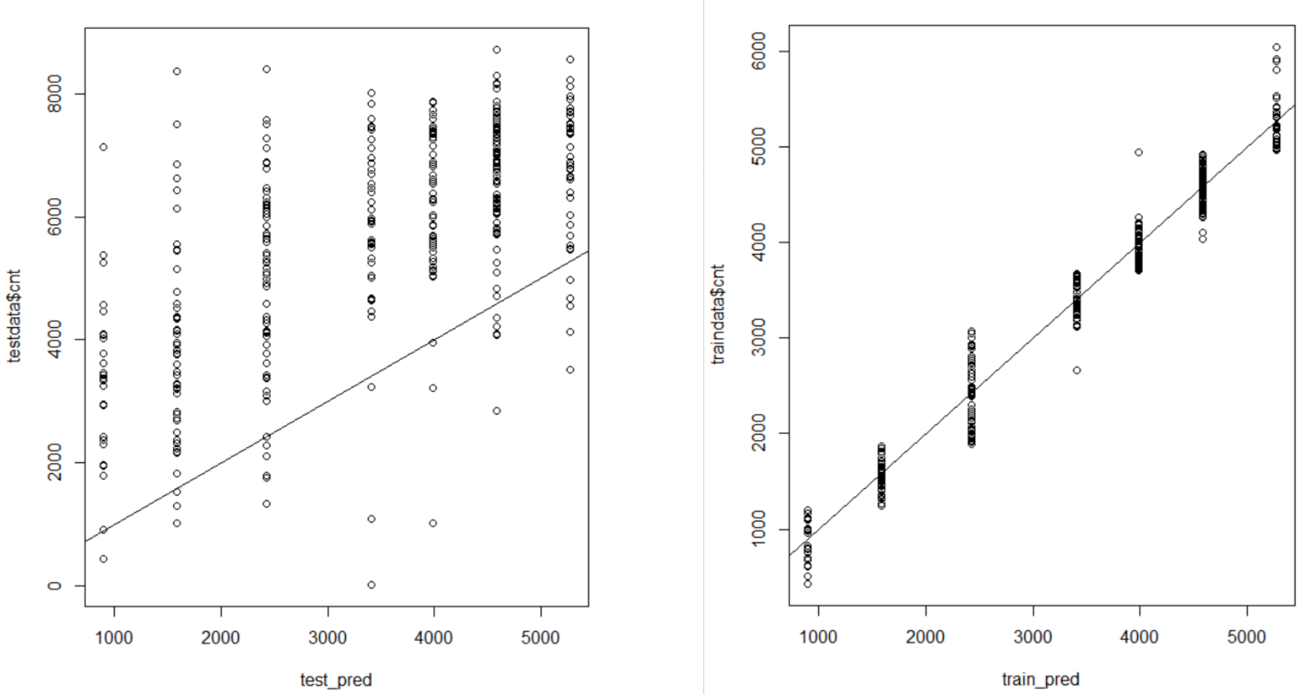
Analysing traindata using plot method.

```
train_pred=predict(dt.fit,traindata)

plot(train_pred,traindata$cnt)

abline(0,1)
```

Analysing testdata using plot method.

```
test_pred=predict(dt.fit,testdata)

testdata <- head(testdata,365)

plot(test_pred,testdata$cnt)

abline(0,1)pred=predict(fit,testdata)
```

TestData vs Train Data



Calculating model errors- Decision Tree

```
RMSE.dt <- sqrt((mean((as.numeric(test_pred)-as.numeric(testdata$cnt))^2))/nrow(testdata))

136.319

MAE.dt <- sum(abs(as.numeric(test_pred)-as.numeric(testdata$cnt)))/length(testdata$cnt)

2321.147
```

The **Root Mean Square Error** is **136.319** and **Mean Absolute Error** is **2321.147**

Business Performance using Decision Tree Model

b) Using the given parameters: revenueperbike=$3 and loanperbike=$2 calculation of profit using existing model and negative binomial model

Default (Existing) model profit calculation:

```
cycle_tmp =as.integer(testdata$cnt)

sum_of_cycle_tmp = sum(cycle_tmp)

lag_cycle_tmp =as.integer(testdata$lagcnt)

sum_of_lag_cycle = sum(lag_cycle_tmp)

total_rented_cycles_lag=0

for(i in 1:length(cycle_tmp)){

 if(cycle_tmp[i] < lag_cycle_tmp[i]){

  total_rented_cycles_lag = total_rented_cycles_lag + cycle_tmp[i]

 }else{

  total_rented_cycles_lag = total_rented_cycles_lag + lag_cycle_tmp[i];

 }}

sum_of_lag_cycle_tmp = (total_rented_cycles_lag*3 - sum_of_lag_cycle*2 );

print(paste("Total Existing Model Profit=====>: ", sum_of_lag_cycle_tmp))

#1442972
```

```
#Default model profit as percentage of expenditure

profit_Percent=(existing_model_profit/(sum_of_lag_cycle*2))*100

# 35.17556
```

```
> profit_Percent=(existing_model_profit/(sum_of_lag_cycle*2))*100
> print(paste("Total Profit Perccent=====>: ", profit_Percent))
[1] "Total Profit Perccent=====>:  35.1755617086523"
```

Decision Tree model profit calculation:

```
cycle_tmp =as.integer(testdata$cnt)
sum_of_cycle_tmp = sum(cycle_tmp)
predicted_cycle_tmp =as.integer(round(test_pred))
sum_of_predicted_cycle = sum(predicted_cycle_tmp)
total_rented_cycles=0
for(i in 1:length(cycle_tmp)){
 if(cycle_tmp[i] < predicted_cycle_tmp[i]){
  total_rented_cycles = total_rented_cycles + cycle_tmp[i]
 }else{
  total_rented_cycles = total_rented_cycles + predicted_cycle_tmp[i];
 }}
predicted_model_profit= (total_rented_cycles*3 - sum_of_predicted_cycle*2 );
# 1177823
```

- **What was your model profit for 2012 expressed as $ total?**
  The model profit for 2012 using negative binomial model is **$ 1177823.**

- **What was your model profit expressed as a percentage of total expenditure?**

```
predicted_model_profit = (total_rented_cycles*3 - sum_of_predicted_cycle*2 );
# 28.77
```

```
> profit_Percent=(predicted_model_profit/(sum_of_cycle_tmp*2))*100
> print(paste("Total Profit Perccent=====>: ", profit_Percent))
[1] "Total Profit Perccent=====>:  28.7716424334599"
```

The model profit expressed as a percentage of total expenditure is **28.77%.**

- **Under what conditions is your prediction model better than the default model? Is it**

f) Always
g) Never
h) Only when revenue is high compared to costs(eg: $8 per rental vs $2 costs)
i) Only when revenue is low compared to costs (eg: $2.2 per rental vs $2 costs)
j) It's hard to say, you see no pattern.

Calculation of profit for $8 rental and $2 costs

```
> predicted_model_profit = (total_rented_cycles*8 - sum_of_predicted_cycle*2 );
> print(paste("Total Predicted Values Profit=====>: ", predicted_model_profit))
[1] "Total Predicted Values Profit=====>:  7284258"
```

Prediction model has **$ 7284258** and default model has **$10683042**

Calculation of profit for $2.2 rental and $2 costs

```
> predicted_model_profit = (total_rented_cycles*2.2 - sum_of_predicted_cycle*2 );
> print(paste("Total Predicted Values Profit=====>: ", predicted_model_profit))
[1] "Total Predicted Values Profit=====>:  200793.4"
```

Prediction model has profit of **$200793.4** and default model has **-$35439.2**

The prediction model has a profit at both $2.2 and $8 rental.

So the answer is a)always

- **Did you find any evidence that model performance correlates with season or similar factor?**

  Compare to season, month variable getting better performance.

- **Did you find any evidence that your model performance decreases with age of the model?**

  Splitting data into one year for training and one year for test data is getting better results than 18 months data for training and 6 months data for testing.

- **18 Months Training data vs 12 Months Training data Results**

  12 Months Training data and 12 Months Test data given better results than 18 Month Training data.

  Only $8 per bicycle getting profit, other two ($3 per bicycle and $2.2 per bicycle) getting loss.

9) Model building and testing-Random Forest Model

Our response variable (y) is the cnt column and the various predictors are the other x's. "tree" library is used to find Decision Trees.

```
fit <- randomForest(as.numeric(traindata$cnt)~ traindata$season + traindata$workingday +
traindata$weathersit  + traindata$temp + traindata$windspeed + traindata$casual +
traindata$registere,   data=traindata, importance=TRUE, mtry=4, ntree=200)
```

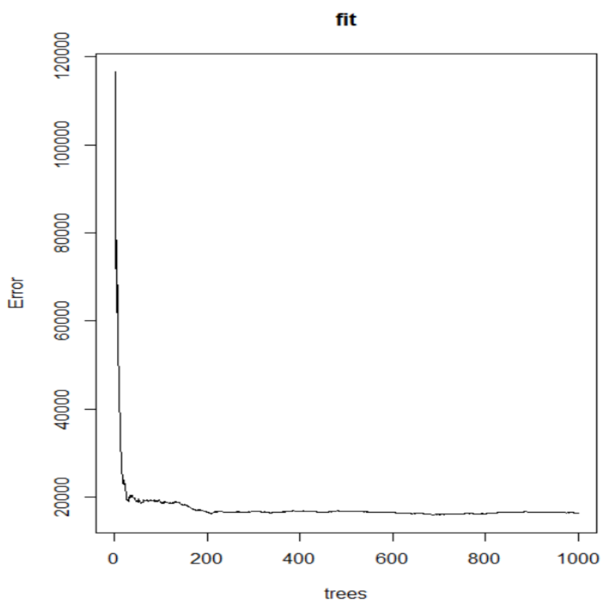
Find Importance of variable using importance() method: Based on below method removed less importance variables and tested with multiple combination of variables to get best results.

```
imp <- as.data.frame(sort(importance(fit)[,1],decreasing = TRUE),optional = T)

names(imp) <- "% Inc MSE"

imp
```

```
> imp <- as.data.frame(sort(importance(fit)[,1],decreasing = TRUE),optional = T)
> names(imp) <- "% Inc MSE"
> imp
                     % Inc MSE
traindata$registered 59.935267
traindata$casual     38.471795
traindata$atemp      23.445231
traindata$workingday 21.950158
traindata$temp       21.191277
traindata$season     18.447650
traindata$mnth       17.663962
traindata$weathersit 16.426534
traindata$weekday    15.405266
traindata$windspeed   8.886437
traindata$yr          0.000000
traindata$holiday    -2.677910
```

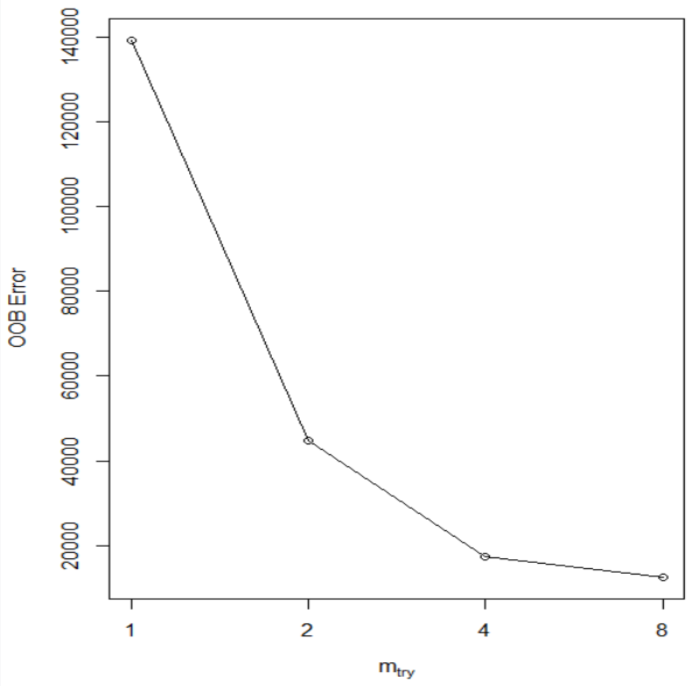
plot(fit) : To know best ntree attribute value in randomForest() method (shown below graph), As per the below graph more than 200 trees no effect, so used 200 trees in out random forest model.

Tuning RandomForest: Below code used to tune Randomforest model to find best mtry attribute value of random forest model.



TuneRF(): Starting with the default value of mtry, search for the optimal value (with respect to Out-of-Bag error estimate) of mtry for randomForest.

```
tunedata <- cbind(traindata$season , traindata$workingday , traindata$weathersit , traindata$temp ,
 traindata$windspeed , traindata$casual , traindata$registered)
t <- tuneRF(tunedata, as.numeric(traindata[,14]),  stepFactor = 0.5,  plot = TRUE, ntreeTry = 300,
        trace = TRUE,  improve = 0.05)
```



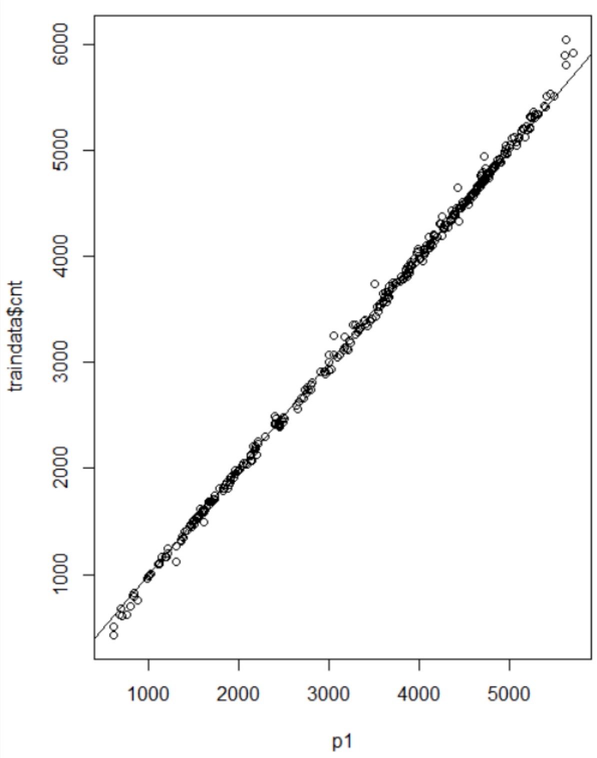
Analysing traindata using plot method.

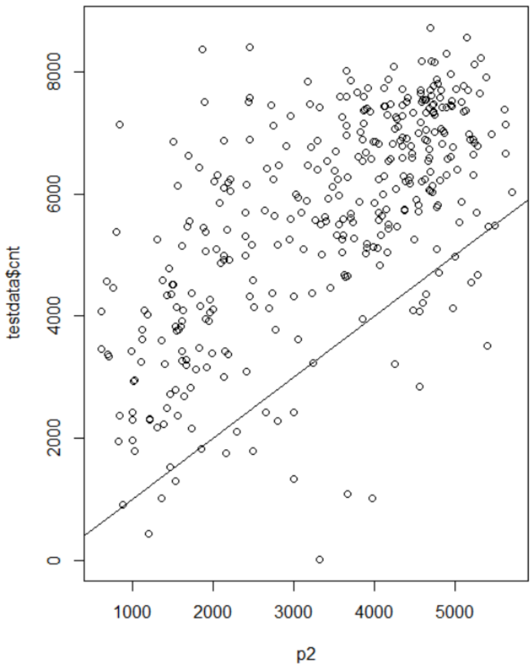
```
p1 <- predict(fit, traindata)
plot(p1,traindata$cnt)
abline(0,1)
```

Analysing testdata using plot method.

```
testdata<-head(testdata,365)
p2 <- predict(fit, testdata$cnt)
plot(p2,testdata$cnt)
abline(0,1)
```

Traindata vs testData





Calculating model errors- Decision Tree

```
RMSE.dt <- sqrt((mean((as.numeric(test_pred)-as.numeric(testdata$cnt))^2))/nrow(testdata))

136.47244

MAE.dt <- sum(abs(as.numeric(test_pred)-as.numeric(testdata$cnt)))/length(testdata$cnt)

2327.93439
```

The **Root Mean Square Error** is 136.47244 and **Mean Absolute Error** is 2327.93439

Business Performance using Decision Tree Model

c) **Using the given parameters: revenueperbike=$3 and loanperbike=$2 calculation of profit using existing model and negative binomial model**

Default (Existing) model profit calculation:

```
cycle_tmp =as.integer(testdata$cnt)
sum_of_cycle_tmp = sum(cycle_tmp)
lag_cycle_tmp =as.integer(testdata$lagcnt)
sum_of_lag_cycle = sum(lag_cycle_tmp)
total_rented_cycles_lag=0
for(i in 1:length(cycle_tmp)){
 if(cycle_tmp[i] < lag_cycle_tmp[i]){
  total_rented_cycles_lag = total_rented_cycles_lag + cycle_tmp[i]
 }else{
  total_rented_cycles_lag = total_rented_cycles_lag + lag_cycle_tmp[i];
 }}
existing_model_profit= (total_rented_cycles_lag*3 - sum_of_lag_cycle*2 );
print(paste("Total Existing Model Profit=====>: ", existing_model_profit))
#1442972
```

```
#Default model profit as percentage of expenditure
profit_Percent=(existing_model_profit/(sum_of_lag_cycle*2))*100
# 35.17556
```

```
> profit_Percent=(existing_model_profit/(sum_of_lag_cycle*2))*100
> print(paste("Total Profit Perccent=====>: ", profit_Percent))
[1] "Total Profit Perccent=====>:  35.1755617086523"
```

Random Forest model profit calculation:

```
cycle_tmp =as.integer(testdata$cnt)
sum_of_cycle_tmp = sum(cycle_tmp)
predicted_cycle_tmp =as.integer(round(test_pred))
sum_of_predicted_cycle = sum(predicted_cycle_tmp)
total_rented_cycles=0
for(i in 1:length(cycle_tmp)){
 if(cycle_tmp[i] < predicted_cycle_tmp[i]){
  total_rented_cycles = total_rented_cycles + cycle_tmp[i]
 }else{
  total_rented_cycles = total_rented_cycles + predicted_cycle_tmp[i];
 }}
predicted_model_profit= (total_rented_cycles*3 - sum_of_predicted_cycle*2 );
# 1174298
```

- **What was your model profit for 2012 expressed as $ total?**
  The model profit for 2012 using negative binomial model is **$ 1174298.**

- **What was your model profit expressed as a percentage of total expenditure?**

```
profit_Percent=(predicted_model_profit/(sum_of_lag_cycle*2))*100
# 28.65268%
```

```
> profit_Percent=(predicted_model_profit/(sum_of_lag_cycle*2))*100
> print(paste("Total Profit Perccent=====>: ", profit_Percent))
[1] "Total Profit Perccent=====>:  10.5415836665538"
```

The model profit expressed as a percentage of total expenditure is **28.65268%.**

- **<u>Under what conditions is your prediction model better than the default model? Is it</u>**

k) Always
l) Never
m) Only when revenue is high compared to costs(eg: $8 per rental vs $2 costs)
n) Only when revenue is low compared to costs (eg: $2.2 per rental vs $2 costs)
o) It's hard to say, you see no pattern.

<u>Calculation of profit for $8 rental and $2 costs</u>

```
> existing_model_profit = (total_rented_cycles_lag*8 - sum_of_lag_cycle*2 );
> print(paste("Total Existing Model Profit=====>: ", existing_model_profit))
[1] "Total Existing Model Profit=====>:  10683042"
```

```
> predicted_model_profit = (total_rented_cycles*8 - sum_of_predicted_cycle*2 );
> print(paste("Total Predicted Values Profit=====>: ", predicted_model_profit))
[1] "Total Predicted Values Profit=====>:  7273498"
```

Prediction model has **$7273498** and default model has **$10683042**

<u>Calculation of profit for $2.2 rental and $2 costs</u>

```
> existing_model_profit = (total_rented_cycles_lag*2.2 - sum_of_lag_cycle*2 );
> print(paste("Total Existing Model Profit=====>: ", existing_model_profit))
[1] "Total Existing Model Profit=====>:  -35439.1999999997"
```

```
> predicted_model_profit = (total_rented_cycles*2.2 - sum_of_predicted_cycle*2 );
> print(paste("Total Predicted Values Profit=====>: ", predicted_model_profit))
[1] "Total Predicted Values Profit=====>:  198281"
```

Prediction model has profit of **$198281** and default model has **-$35439.2**

The prediction model has a profit at both $2.2 and $8 rental.

So the answer is a)always

- **<u>Did you find any evidence that model performance correlates with season or similar factor?</u>**

  Compare to season, month variable getting better performance.

- **<u>Did you find any evidence that your model performance decreases with age of the model?</u>**

  Splitting data into one year for training and one year for test data is getting better results than 18 months data for training and 6 months data for testing.

- **<u>18 Months Training data vs 12 Months Training data Results</u>**

  12 Months Training data and 12 Months Test data given better results than 18 Month Training data.

  Only $8 per bicycle getting profit, other two ($3 per bicycle and $2.2 per bicycle) getting loss.

10) Neural Network-Model Building and testing

In Our **Neural Network model** we tried different response variables like trend variables (weekly Trend and tomorrow Trend) as (y) and the various predictors are the other x's. And to predict we used **rattle, nnet packages**.

```
library(rattle)

rattle()

preds<- predict(crs$nnet,newdata=testdata[,crs$input])

predpairs = cbind(testdata[crs$target],preds)

plot(predpairs)
```

Used below shown dependent and independent variables to predict bicycles. Tominc variable is (tomorrow trend) chosen as target variable.

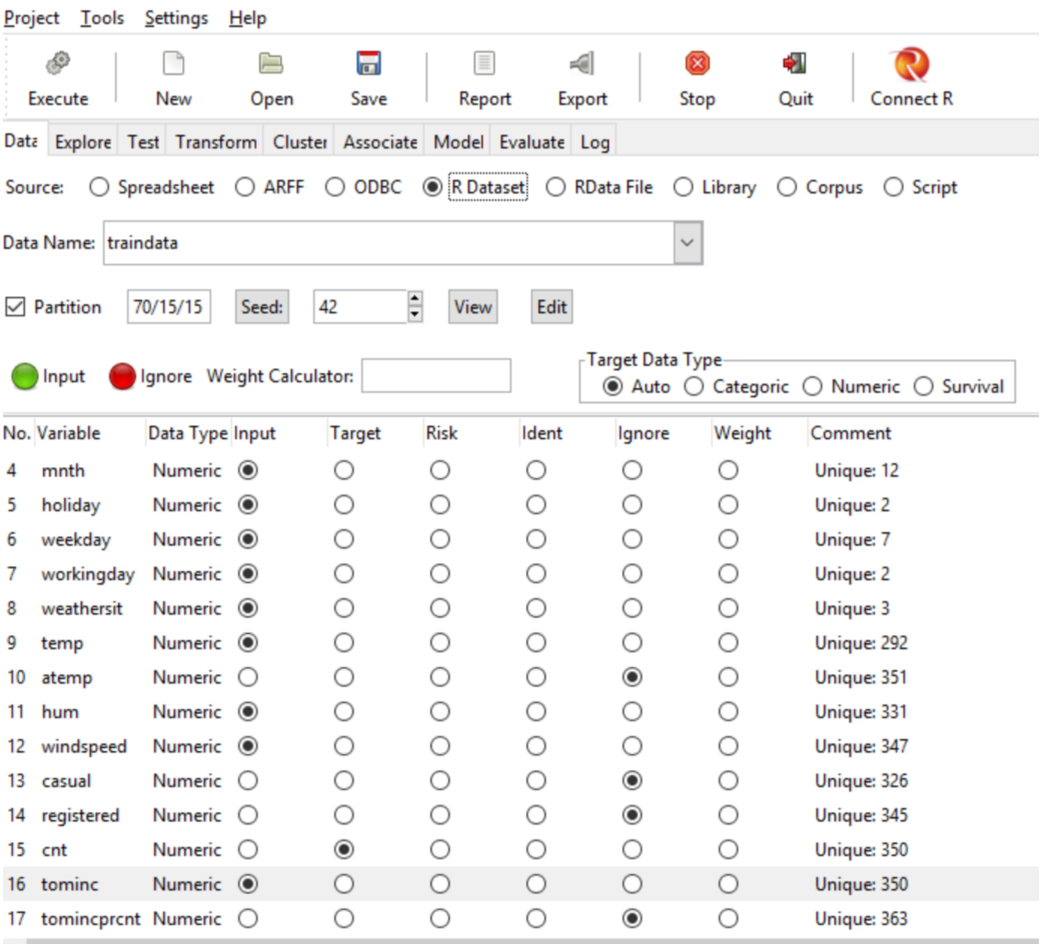
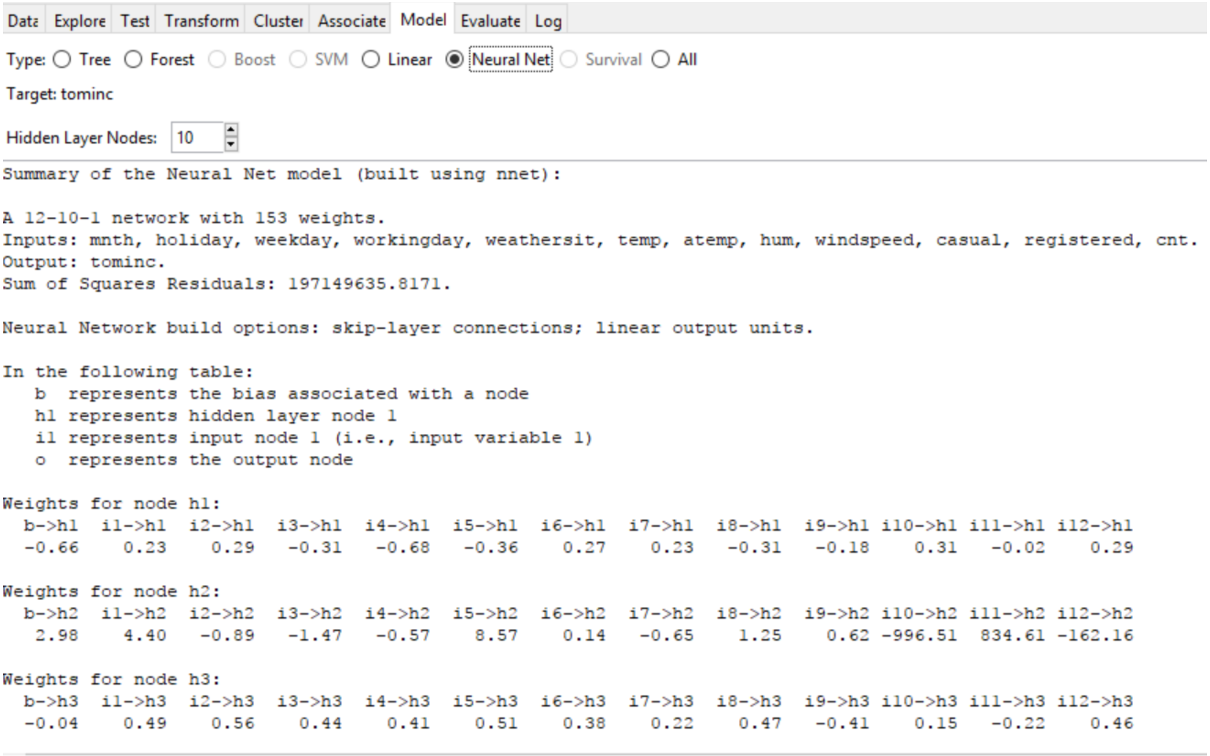


Fig: Rattle Data tab and variables selection.



Data Explore Test Transform Cluster Associate Model Evaluate Log

Type: Tree Forest Boost SVM Linear Neural Net Survival All

Target: tominc

Hidden Layer Nodes: 10

```
Summary of the Neural Net model (built using nnet):

A 12-10-1 network with 153 weights.
Inputs: mnth, holiday, weekday, workingday, weathersit, temp, atemp, hum, windspeed, casual, registered, cnt.
Output: tominc.
Sum of Squares Residuals: 197149635.8171.

Neural Network build options: skip-layer connections; linear output units.

In the following table:
   b  represents the bias associated with a node
   h1 represents hidden layer node 1
   i1 represents input node 1 (i.e., input variable 1)
   o  represents the output node

Weights for node h1:
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1 i10->h1 i11->h1 i12->h1
  -0.66    0.23    0.29   -0.31   -0.68   -0.36    0.27    0.23   -0.31   -0.18    0.31   -0.02    0.29

Weights for node h2:
  b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2  i8->h2  i9->h2 i10->h2 i11->h2 i12->h2
   2.98    4.40   -0.89   -1.47   -0.57    8.57    0.14   -0.65    1.25    0.62 -996.51  834.61 -162.16

Weights for node h3:
  b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3  i7->h3  i8->h3  i9->h3 i10->h3 i11->h3 i12->h3
  -0.04    0.49    0.56    0.44    0.41    0.51    0.38    0.22    0.47   -0.41    0.15   -0.22    0.46
```

Fig: Rattle Model selection tab and Summary of the Neural Net Model.

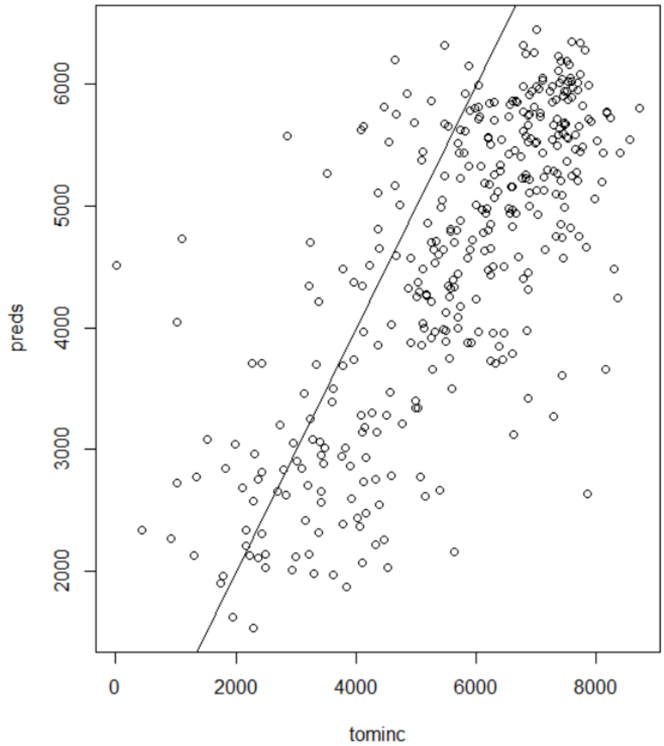


Fig: Plotting

Calculating model errors- Poisson Model

```
errors = apply(predpairs,1,function(row) abs(row[1]-row[2]))

cat(sprintf("means abs error = %f\n", mean(errors)))
```

```
> errors = apply(predpairs,1,function(row) abs(row[1]-row[2]))
> cat(sprintf("means abs error = %f\n", mean(errors)))
means abs error = 1358.287594
```

Neural Network model profit calculation:

```
lag_cycle_tmp =as.integer(testdata$tominc)

sum_of_lag_cycle = sum(lag_cycle_tmp)

cycle_tmp =as.integer(testdata$cnt)

sum_of_cycle_tmp = sum(cycle_tmp)

predicted_cycle_tmp =as.integer(round(preds))

sum_of_predicted_cycle = sum(predicted_cycle_tmp)

total_rented_cycles=0

for(i in 1:length(cycle_tmp)){

 if(cycle_tmp[i] < predicted_cycle_tmp[i]){

  total_rented_cycles = total_rented_cycles + cycle_tmp[i]

 }else{    total_rented_cycles = total_rented_cycles + predicted_cycle_tmp[i];

 }}
```

```
> predicted_model_profit = (total_rented_cycles*3 - sum_of_predicted_cycle*2 );
> print(paste("Total Predicted Values Profit=====>:  ", predicted_model_profit))
[1] "Total Predicted Values Profit=====>:  1468734"
> profit_Percent=(predicted_model_profit/(sum_of_lag_cycle*2))*100
> print(paste("Total Predicted Profit Perccent=====>:  ", profit_Percent))
[1] "Total Predicted Profit Perccent=====>:  35.8134340550149"
> 
```

- **What was your model profit for 2012 expressed as $ total?**
  The model profit for 2012 using negative binomial model is **$1468734.**

- **What was your model profit expressed as a percentage of total expenditure?**

  The model profit expressed as a percentage of total expenditure is **35.81%.**

- **<u>What is the profit (total and percentage of expenditure) for the default prediction?</u>**

The profit for the default prediction is **<u>$1442972</u>** and as a percentage of expenditure is **<u>35.18%</u>**

- **<u>Under what conditions is your prediction model better than the default model? Is it</u>**

p) Always
q) Never
r) Only when revenue is high compared to costs(eg: $8 per rental vs $2 costs)
s) Only when revenue is low compared to costs (eg: $2.2 per rental vs $2 costs)
t) It's hard to say, you see no pattern.

<u>Calculation of profit for $8 rental and $2 costs</u>

```
> predicted_model_profit = (total_rented_cycles*8 - sum_of_predicted_cycle*2 );
> print(paste("Total Predicted Values Profit=====>: ", predicted_model_profit))
[1] "Total Predicted Values Profit=====>:  9111744"
> profit_Percent=(predicted_model_profit/(sum_of_lag_cycle*2))*100
> print(paste("Total Predicted Profit Perccent=====>: ", profit_Percent))
[1] "Total Predicted Profit Perccent=====>:  222.179675060411"
>
```

Prediction model has $**9111744** and default model has $**10683042**

<u>Calculation of profit for $2.2 rental and $2 costs</u>

```
> predicted_model_profit = (total_rented_cycles*2.2 - sum_of_predicted_cycle*2 );
> print(paste("Total Predicted Values Profit=====>: ", predicted_model_profit))
[1] "Total Predicted Values Profit=====>:  245852.4"
> profit_Percent=(predicted_model_profit/(sum_of_lag_cycle*2))*100
> print(paste("Total Predicted Profit Perccent=====>: ", profit_Percent))
[1] "Total Predicted Profit Perccent=====>:  5.99483549415154"
>
```

Prediction model has profit of **$245852** and default model has **-$35439.2**

The prediction model has a profit at both $2.2 and $8 rental.

So the answer is a)always

- **<u>Did you find any evidence that model performance correlates with season or similar factor?</u>**

  Compare to season, month variable getting better performance.

- **<u>Did you find any evidence that your model performance decreases with age of the model?</u>**

  For Neural network, tomorrow trend variable is giving better performance then weekly trend variable.

- **<u>18 Months Training data vs 12 Months Training data Results</u>**

  12 Months Training data and 12 Months Test data given better results than 18 Months Training data.

  Always getting both models profit, but 12 Months Training model has more profit.

12) Conclusion

Among all 5 models Poisson Model, Negative binomial, Decision Tree, Random Forest, Neural Network we got more profit using Negative binomial model is 1557088 and default model profit is 1442972.