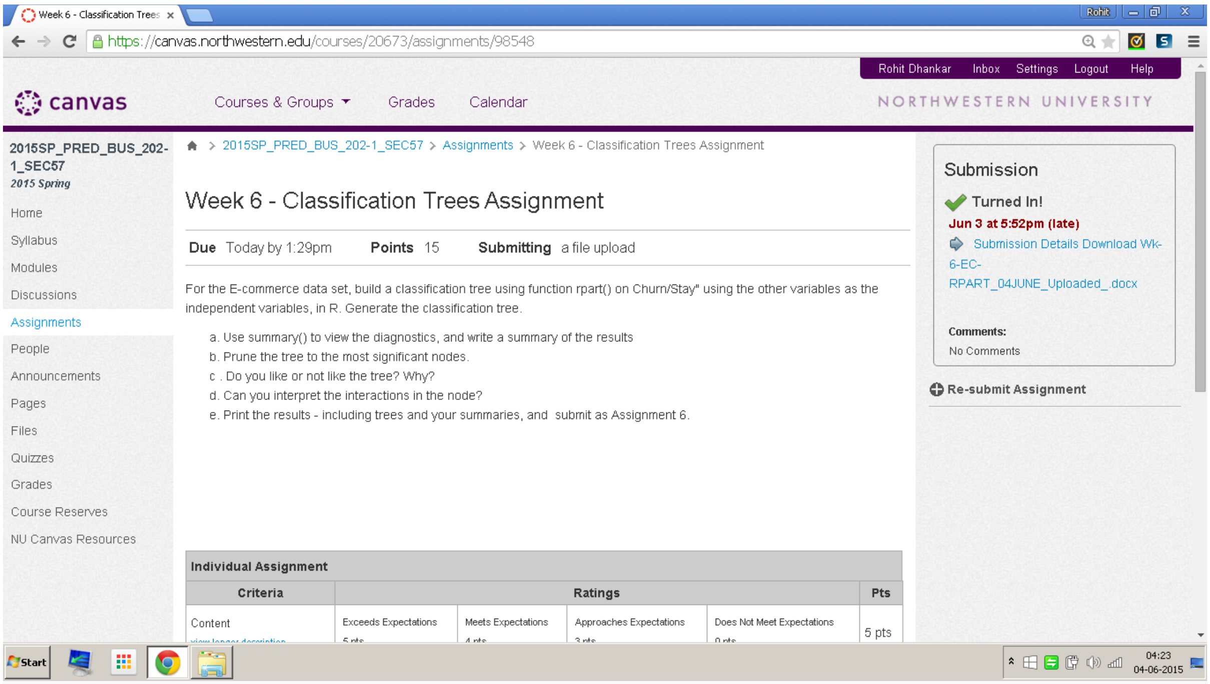

Week 6 - Classification Trees

https://canvas.northwestern.edu/courses/20673/assignments/98548

Rohit Dhankar Inbox Settings Logout Help

canvas Courses & Groups Grades Calendar NORTHWESTERN UNIVERSITY

2015SP_PRED_BUS_202-1_SEC57
2015 Spring

Home
Syllabus
Modules
Discussions
Assignments
People
Announcements
Pages
Files
Quizzes
Grades
Course Reserves
NU Canvas Resources

> 2015SP_PRED_BUS_202-1_SEC57 > Assignments > Week 6 - Classification Trees Assignment

Week 6 - Classification Trees Assignment

Due Today by 1:29pm **Points** 15 **Submitting** a file upload

For the E-commerce data set, build a classification tree using function rpart() on Churn/Stay" using the other variables as the independent variables, in R. Generate the classification tree.

a. Use summary() to view the diagnostics, and write a summary of the results
b. Prune the tree to the most significant nodes.
c . Do you like or not like the tree? Why?
d. Can you interpret the interactions in the node?
e. Print the results - including trees and your summaries, and submit as Assignment 6.

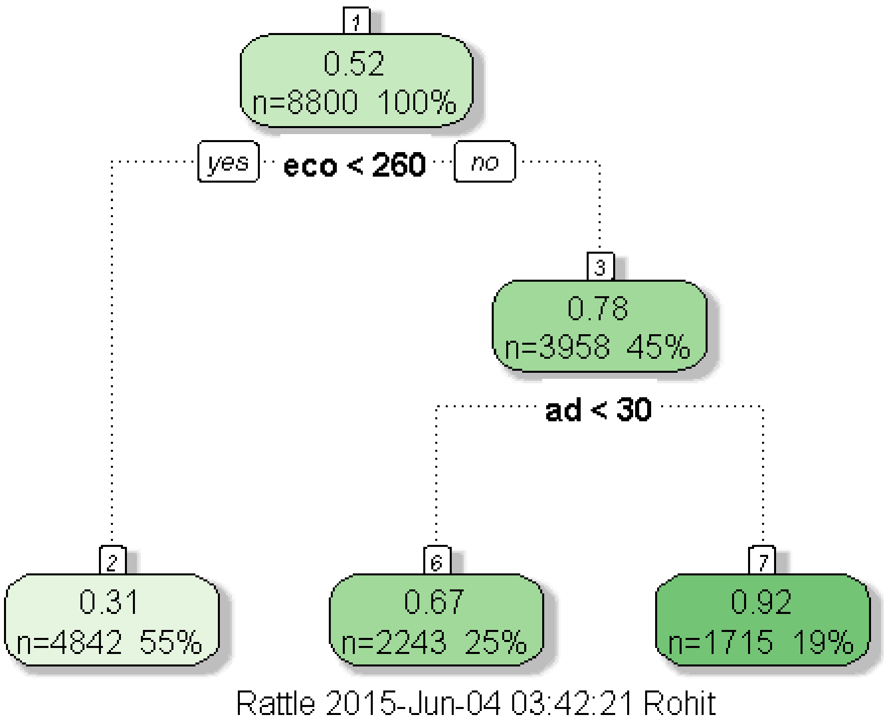
Individual Assignment					
Criteria	Ratings				Pts
Content	Exceeds Expectations	Meets Expectations	Approaches Expectations	Does Not Meet Expectations	5 pts

Submission
Turned In!
Jun 3 at 5:52pm (late)
Submission Details Download Wk-6-EC-RPART_04JUNE_Uploaded_.docx
Comments:
No Comments
Re-submit Assignment

Start 04:23 04-06-2015

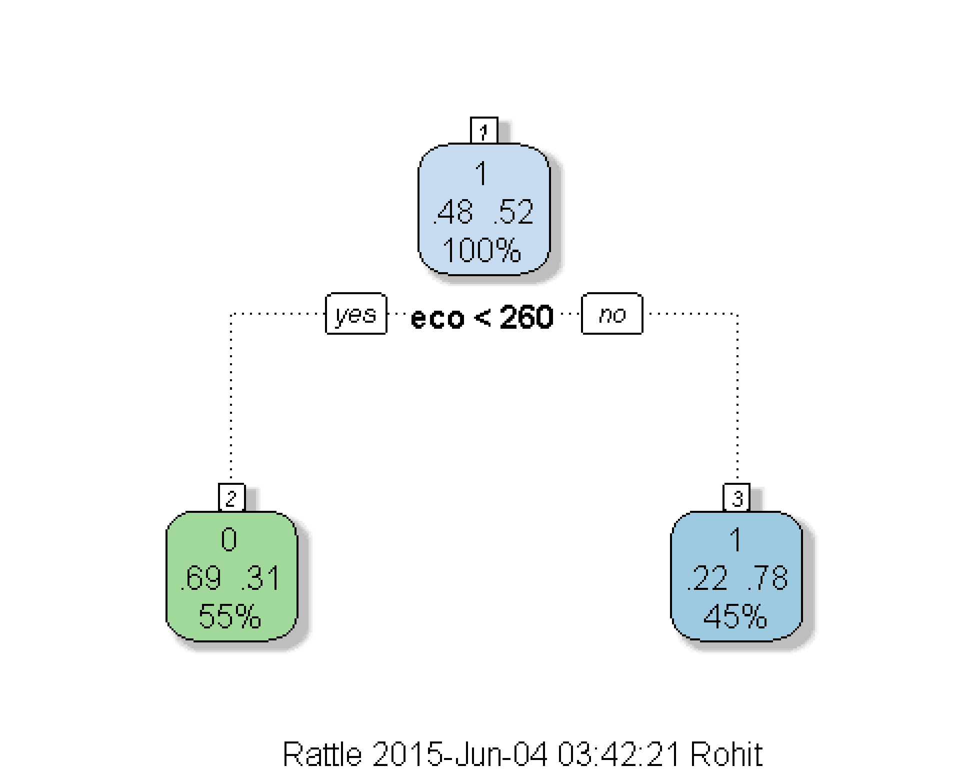
Week 6 - Classification Trees Assignment

```
ec <- read.csv("C:/STAT/wk-6-EC-RPART/ec.csv")
attach(ec)
#Creation of Training and Test Data
s<-c(sample(1:500,400), sample(501:1000,400), sample(1001:10000,8000))
trn<-ec[s,]
tst<-ec[-s,]
library(rpart)
library(caret)

## Loading required package: lattice
## Loading required package: ggplot2

# Building the Classification Tree. As Dependent Variable is Binomial.
# Churned ==0 / Stayed ==1
# Model - mT created from Training Data Sample.
# Same model tested with "Test" data for Prediction Accuracy .
mT<-train(st~.,method="rpart",data =trn)

library(rattle)

fancyRpartPlot(mT$finalModel)
```

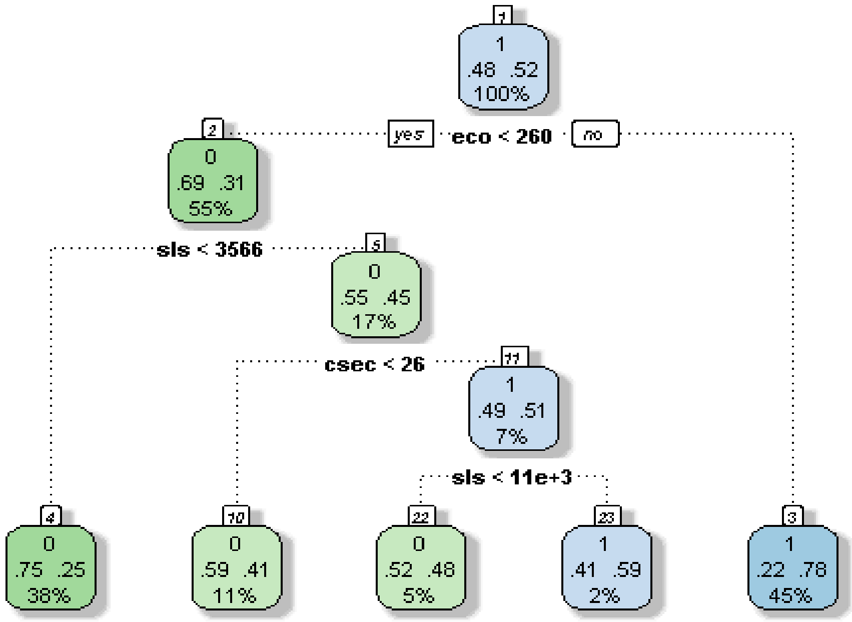



```
#Prediction for Training Data
TrnPred<-predict(mT, newdata=trn)
head(TrnPred)

## [1] 0.3118546 0.6705305 0.3118546 0.9236152 0.9236152 0.3118546

# Transferring the Data Type - TrnPred from Numeric Vector to FACTOR
# As seen below we do not get the 0 and 1 classification as we did with the "iris"
data.
# The table() seen below is the Confusion Matrix ...
TrnPred<-as.factor(TrnPred)
head(TrnPred)

## [1] 0.311854605534903 0.670530539456086 0.311854605534903
0.923615160349854
## [5] 0.923615160349854 0.311854605534903
## Levels: 0.311854605534903 0.670530539456086 0.923615160349854

# Confusion Matrix ....

table(trn$st,TrnPred)

##    TrnPred
##     0.311854605534903 0.670530539456086 0.923615160349854
##   0              3332               739               131
##   1              1510              1504              1584

TstPred<-predict(mT, newdata=tst)
head(TstPred)

## [1] 0.6705305 0.9236152 0.6705305 0.6705305 0.9236152 0.9236152

# Confusion Matrix ....
```

```
table(tst$st,TstPred)

##    TstPred
##     0.311854605534903 0.670530539456086 0.923615160349854
##   0               446               107                18
##   1               201               219               209

#Building the Tree with Function - rpart and method ="class"
mRp<-rpart(st~.,data=trn, method="class")
fancyRpartPlot(mRp)
```

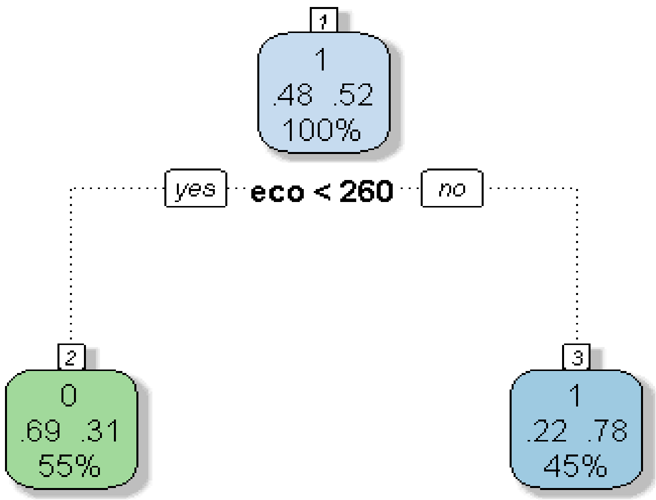



```
# using Control -- minsplit==500 and cp==0.001
mRp_minsplit500cp0.001<-rpart(st~.,data=trn,
method="class",control=rpart.control(minsplit=500, cp=0.001))

# Creating fancyRpartPlot of the Classification Tree of data=trn [ Training Data]
fancyRpartPlot(mRp_minsplit500cp0.001)
```



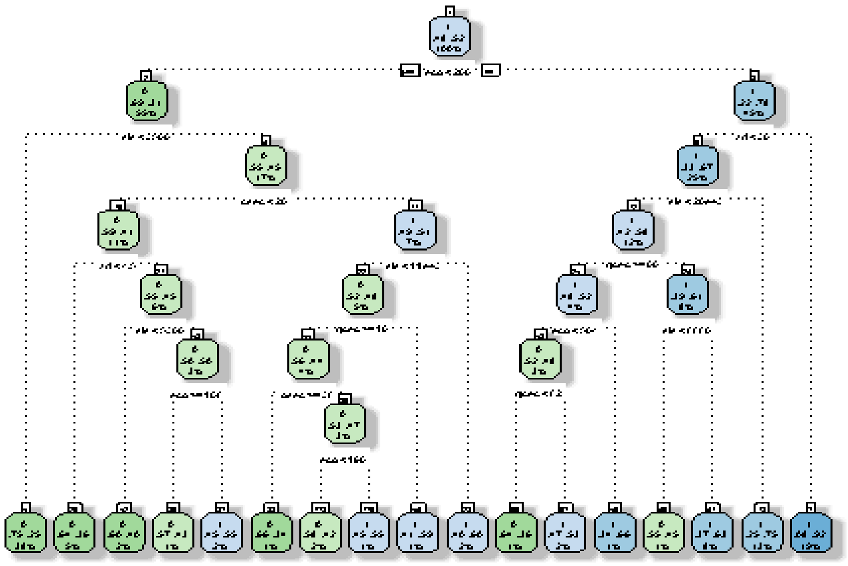
Rattle 2015-Jun-04 03:42:22 Rohit

```
# using Control -- minsplit==1000 and cp==0.05
mRp_minsplit100cp0.005<-rpart(st~.,data=trn,
method="class",control=rpart.control(minsplit=500, cp=0.005))
fancyRpartPlot(mRp_minsplit100cp0.005)
```



Rattle 2015-Jun-04 03:42:23 Rohit

```
# using Control -- minsplit==200 and cp==0.0005
mRp_minsplit200cp0.0005<-rpart(st~.,data=trn,
method="class",control=rpart.control(minsplit=200, cp=0.0005))
fancyRpartPlot(mRp_minsplit200cp0.0005)
```



Rattle 2015-Jun-04 03:42:23 Rohit

```
#Display Results
print(mRp)

## n= 8800
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 8800 4202 1 (0.4775000 0.5225000)
##   2) eco< 259.5 4842 1510 0 (0.6881454 0.3118546) *
##   3) eco>=259.5 3958  870 1 (0.2198080 0.7801920) *

print(mRp_minsplit200cp0.0005)

## n= 8800
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 8800 4202 1 (0.47750000 0.52250000)
##     2) eco< 259.5 4842 1510 0 (0.68814539 0.31185461)
##       4) sls< 3566.5 3304  821 0 (0.75151332 0.24848668) *
##       5) sls>=3566.5 1538  689 0 (0.55201560 0.44798440)
##        10) csec< 25.5 941  384 0 (0.59192349 0.40807651)
##          20) ad< 4.5 452  162 0 (0.64159292 0.35840708) *
##          21) ad>=4.5 489  222 0 (0.54601227 0.45398773)
##            42) sls< 5366 216   86 0 (0.60185185 0.39814815) *
##            43) sls>=5366 273  136 0 (0.50183150 0.49816850)
##              86) eco>=168.5 116   50 0 (0.56896552 0.43103448) *
##              87) eco< 168.5 157   71 1 (0.45222930 0.54777070) *
##        11) csec>=25.5 597  292 1 (0.48911223 0.51088777)
##          22) sls< 11098.5 438  210 0 (0.52054795 0.47945205)
##            44) qcec>=10.5 319  140 0 (0.56112853 0.43887147)
```

```
##             88) osec>=38.5 82   28 0 (0.65853659 0.34146341) *
##             89) osec< 38.5 237  112 0 (0.52742616 0.47257384)
##              178) eco< 196.5 141   59 0 (0.58156028 0.41843972) *
##              179) eco>=196.5 96   43 1 (0.44791667 0.55208333) *
##           45) qcec< 10.5 119   49 1 (0.41176471 0.58823529) *
##         23) sls>=11098.5 159   64 1 (0.40251572 0.59748428) *
##     3) eco>=259.5 3958  870 1 (0.21980798 0.78019202)
##       6) ad< 29.5 2243  739 1 (0.32946946 0.67053054)
##        12) sls< 26358.5 1080  451 1 (0.41759259 0.58240741)
##          24) qcec>=65.5 346  166 1 (0.47976879 0.52023121)
##            48) eco< 564.5 261  124 0 (0.52490421 0.47509579)
##              96) qcec< 82.5 81   29 0 (0.64197531 0.35802469) *
##              97) qcec>=82.5 180   85 1 (0.47222222 0.52777778) *
##            49) eco>=564.5 85   29 1 (0.34117647 0.65882353) *
##          25) qcec< 65.5 734  285 1 (0.38828338 0.61171662)
##            50) sls< 8886 71   32 0 (0.54929577 0.45070423) *
##            51) sls>=8886 663  246 1 (0.37104072 0.62895928) *
##        13) sls>=26358.5 1163  288 1 (0.24763543 0.75236457) *
##       7) ad>=29.5 1715  131 1 (0.07638484 0.92361516) *
```

```r
# plot cp() - complexity parameter value
plotcp(mRp)
```

```r
plotcp(mRp_minsplit200cp0.0005)
```

```
printcp(mRp)
```

```
##
## Classification tree:
## rpart(formula = st ~ ., data = trn, method = "class")
##
## Variables actually used in tree construction:
## [1] eco
##
## Root node error: 4202/8800 = 0.4775
##
## n= 8800
##
##       CP nsplit rel error  xerror      xstd
## 1 0.4336      0    1.0000 1.00000 0.0111510
## 2 0.0100      1    0.5664 0.57877 0.0099836
```

```
printcp(mRp_minsplit200cp0.0005)
```

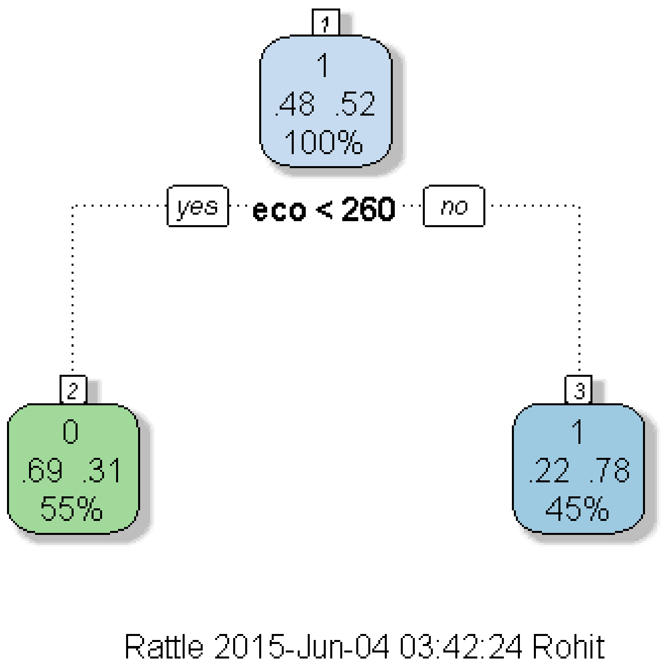
```
##
## Classification tree:
## rpart(formula = st ~ ., data = trn, method = "class", control =
rpart.control(minsplit = 200,
##     cp = 5e-04))
##
## Variables actually used in tree construction:
## [1] ad   csec eco  osec qcec sls
##
```

```
## Root node error: 4202/8800 = 0.4775
##
## n= 8800
##
##           CP nsplit rel error  xerror      xstd
## 1 0.43360305      0   1.00000 1.00000 0.0111510
## 2 0.00309376      1   0.56640 0.57330 0.0099542
## 3 0.00118991      5   0.55402 0.57235 0.0099490
## 4 0.00095193     10   0.54807 0.57473 0.0099619
## 5 0.00050000     16   0.54093 0.58591 0.0100213
```

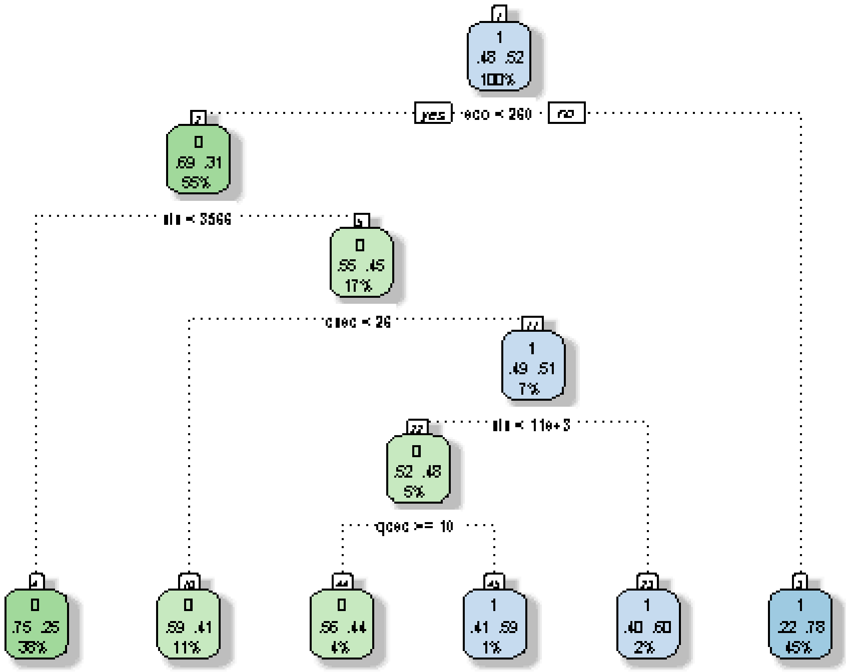
```
mRp$cptable[which.min(mRp$cptable[,"xerror"]),"CP"]
```

```
## [1] 0.01
```

```
# as seen from the CP table and the "which.min" formula...
# CP ==0.01  is the Min CP corresponding to the Minimum "xerror".
# Combining the Prune command and the which.min commands...
# Pruning the Tree basis the CP values.

pmRp<- prune(mRp, cp= mRp$cptable[which.min(mRp$cptable[,"xerror"]),"CP"])
pmRp_minsplit200cp0.0005<- prune(mRp_minsplit200cp0.0005, cp=
mRp_minsplit200cp0.0005$cptable[which.min(mRp_minsplit200cp0.0005$cptabl
e[,"xerror"]),"CP"])



# plot the Pruned tree using - #fancyRpartPlot(pmRp)
fancyRpartPlot(pmRp)
```




```
fancyRpartPlot(pmRp_minsplit200cp0.0005)
```



Rattle 2015-Jun-04 03:42:24 Rohit

```
# Bagging -- Bagging will have bias similar to the individual models
# but a reduced variance as we are averaging over individual models.

library(caret)
m_bag<-train(st~.,method="treebag",data =trn)

## Loading required package: ipred
## Loading required package: plyr

print(m_bag)

## Bagged CART
##
## 8800 samples
##    8 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 8800, 8800, 8800, 8800, 8800, 8800, ...
##
## Resampling results
##
##   RMSE       Rsquared   RMSE SD      Rsquared SD
##   0.4277007  0.2678247  0.003554656  0.01313884
##
##
```

```
# Creating a Random Forest ...

library(randomForest)

## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.

rf_model_1<-randomForest(st~.,data =trn,ntree = 10)

## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?

rf_model_1

##
## Call:
##  randomForest(formula = st ~ ., data = trn, ntree = 10)
##                Type of random forest: regression
##                      Number of trees: 10
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.2241656
##                     % Var explained: 10.15

# This code - has created a Random Forest -- but its a Type :- REGRESSION we
need a Type :- Classification .
# Also we gort the below mentioned warning message ....
# Warning message:
#   In randomForest.default(m, y, ...) :
#   The response has five or fewer unique values.  Are you sure you want to do
regression?

# Checking the head - of the 2nd Tree in the Random Forest
head(getTree(rf_model_1,k=2))

##   left daughter right daughter split var split point status prediction
## 1             2              3         3       293.5     -3  0.5225000
## 2             4              5         4        12.5     -3  0.3225112
## 3             6              7         8        35.5     -3  0.7998373
## 4             8              9         2         2.5     -3  0.2445492
## 5            10             11         3       188.5     -3  0.4764398
## 6            12             13         4       118.5     -3  0.7091141

# Checking the head - of the 3rd Tree in the Random Forest
head(getTree(rf_model_1,k=3))

##   left daughter right daughter split var split point status prediction
## 1             2              3         2        33.5     -3  0.5218182
## 2             4              5         3       142.5     -3  0.3363862
## 3             6              7         3       773.5     -3  0.7948862
## 4             8              9         5        12.5     -3  0.2783902
## 5            10             11         1     11135.0     -3  0.5052239
## 6            12             13         4        62.5     -3  0.6756565

# From the Two - Head , values as seen here
# - the "left daughter" and  "right daughter" , values
# are the same for both the Trees .
```

```
# Values of "split var" , "split point" and "prediction" are different.
```