Project.R

User

2019-10-29

library("dplyr")

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library("readxl")  
library("partykit")

## Loading required package: grid

## Loading required package: libcoin

## Loading required package: mvtnorm

library("rpart")  
library("caTools")  
library("caret")

## Loading required package: lattice

## Loading required package: ggplot2

Data <- read\_xlsx("C:/Users/User/Desktop/SEM-1/Data Analytics/ProjectData.xlsx")  
str(Data)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 296 obs. of 17 variables:  
## $ ID : num 1 2 3 4 5 6 7 8 9 10 ...  
## $ Response: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ Group : num 0 1 1 0 1 0 1 1 0 0 ...  
## $ X1 : num 460 74 58 39 15 47 23 14 56 40 ...  
## $ X2 : num 460 NA 0 NA 45 141 69 126 0 120 ...  
## $ X3 : num 460 NA 0 NA 60 188 92 224 0 160 ...  
## $ X4 : num 50.2 812.5 87.7 92.1 75.2 ...  
## $ X5 : num 9.15 0.88 0.39 26.79 16.6 ...  
## $ X6 : num 2.3 4.1 4.7 3.1 3.6 2.6 7.1 2.4 2.7 2.6 ...  
## $ X7 : num 274 407 946 535 1019 ...  
## $ Y1 : num 1 1 1 1 0 1 0 0 1 1 ...  
## $ Y2 : num 1 NA 0 NA 0 0 0 0 0 0 ...  
## $ Y3 : num 1 NA 0 NA 0 0 0 0 0 0 ...  
## $ Y4 : num 0 1 1 1 1 0 0 0 0 0 ...  
## $ Y5 : num 0 0 0 1 0 0 0 0 1 1 ...  
## $ Y6 : num 1 2 2 1 2 1 2 1 1 1 ...  
## $ Y7 : num 0 0 1 1 1 1 1 0 0 0 ...

df <- data.frame(Data[,-1]) #remove ID from dataframe  
Target=ifelse(df$Response==1,'Y','N')   
df <- data.frame(df, Target) #add Target to the dataframe  
df <- df[,-1] #remove Response  
df <- mutate(df,Y1=factor(Y1),Y2=factor(Y2),Y3=factor(Y3),Y4=factor(Y4),Y5=factor(Y5),Y6=factor(Y6),Y7=factor(Y7))  
str(df)

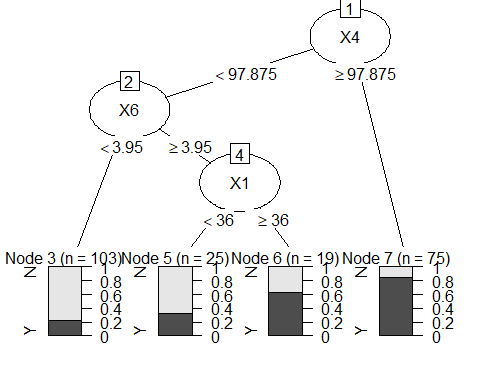
## 'data.frame': 296 obs. of 16 variables:  
## $ Group : num 0 1 1 0 1 0 1 1 0 0 ...  
## $ X1 : num 460 74 58 39 15 47 23 14 56 40 ...  
## $ X2 : num 460 NA 0 NA 45 141 69 126 0 120 ...  
## $ X3 : num 460 NA 0 NA 60 188 92 224 0 160 ...  
## $ X4 : num 50.2 812.5 87.7 92.1 75.2 ...  
## $ X5 : num 9.15 0.88 0.39 26.79 16.6 ...  
## $ X6 : num 2.3 4.1 4.7 3.1 3.6 2.6 7.1 2.4 2.7 2.6 ...  
## $ X7 : num 274 407 946 535 1019 ...  
## $ Y1 : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 1 2 2 ...  
## $ Y2 : Factor w/ 2 levels "0","1": 2 NA 1 NA 1 1 1 1 1 1 ...  
## $ Y3 : Factor w/ 2 levels "0","1": 2 NA 1 NA 1 1 1 1 1 1 ...  
## $ Y4 : Factor w/ 2 levels "0","1": 1 2 2 2 2 1 1 1 1 1 ...  
## $ Y5 : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 2 2 ...  
## $ Y6 : Factor w/ 3 levels "0","1","2": 2 3 3 2 3 2 3 2 2 2 ...  
## $ Y7 : Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 1 1 1 ...  
## $ Target: Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...

summary(df)

## Group X1 X2 X3   
## Min. :0.0000 Min. : 5.0 Min. : 0.00 Min. : 0   
## 1st Qu.:0.0000 1st Qu.: 16.0 1st Qu.: 30.25 1st Qu.: 40   
## Median :1.0000 Median : 38.0 Median : 126.00 Median : 192   
## Mean :0.6757 Mean : 301.3 Mean : 2908.60 Mean : 5015   
## 3rd Qu.:1.0000 3rd Qu.: 186.0 3rd Qu.: 558.75 3rd Qu.: 880   
## Max. :1.0000 Max. :9743.0 Max. :80919.00 Max. :143856   
## NA's :4 NA's :130 NA's :131   
## X4 X5 X6 X7   
## Min. : 21.82 Min. : 0.100 Min. :0.900 Min. : 110.3   
## 1st Qu.: 50.61 1st Qu.: 9.057 1st Qu.:3.100 1st Qu.: 368.1   
## Median : 71.83 Median :19.300 Median :3.600 Median : 653.2   
## Mean : 233.34 Mean :35.317 Mean :3.836 Mean :1353.1   
## 3rd Qu.: 132.38 3rd Qu.:61.970 3rd Qu.:4.300 3rd Qu.:1519.2   
## Max. :6864.00 Max. :99.800 Max. :9.700 Max. :8491.1   
## NA's :4 NA's :63 NA's :24   
## Y1 Y2 Y3 Y4 Y5 Y6 Y7   
## 0 :143 0 :105 0 : 89 0:144 0 :157 0 : 4 0 :112   
## 1 :149 1 : 61 1 : 76 1:152 1 :135 1 :104 1 :160   
## NA's: 4 NA's:130 NA's:131 NA's: 4 2 :125 NA's: 24   
## NA's: 63   
##   
##   
##   
## Target   
## N:154   
## Y:142   
##   
##   
##   
##   
##

#split into train and test sets  
set.seed(123)  
sample= sample.split(df$Target,SplitRatio= 0.75)  
train=subset(df, sample==TRUE)  
test= subset(df, sample==FALSE)

# All X's, Y's and groups  
df\_with\_all\_X\_and\_Y\_all\_groups <- train  
DT\_Model\_XY\_all\_groups <- rpart(Target~., data=df\_with\_all\_X\_and\_Y\_all\_groups,   
 control=rpart.control(minsplit=30,   
 minbucket=15,   
 maxdepth=4 ))   
plot(as.party(DT\_Model\_XY\_all\_groups))

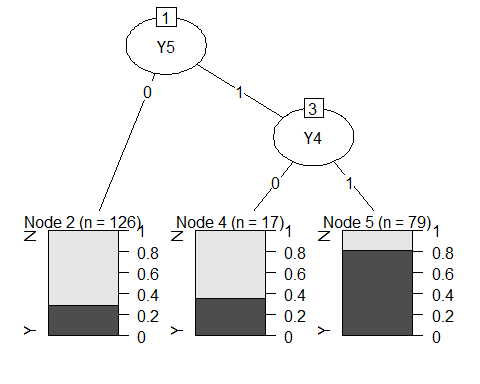


# checking accuracy using confusion matrix  
tree.predicted<- predict(DT\_Model\_XY\_all\_groups,test, type='class')  
confusionMatrix(tree.predicted , test$Target)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 32 10  
## Y 6 26  
##   
## Accuracy : 0.7838   
## 95% CI : (0.6728, 0.8711)  
## No Information Rate : 0.5135   
## P-Value [Acc > NIR] : 1.494e-06   
##   
## Kappa : 0.566   
##   
## Mcnemar's Test P-Value : 0.4533   
##   
## Sensitivity : 0.8421   
## Specificity : 0.7222   
## Pos Pred Value : 0.7619   
## Neg Pred Value : 0.8125   
## Prevalence : 0.5135   
## Detection Rate : 0.4324   
## Detection Prevalence : 0.5676   
## Balanced Accuracy : 0.7822   
##   
## 'Positive' Class : N   
##

**\*Note: This tree has the best accuracy out of all trees. Accuracy = 78.38%**

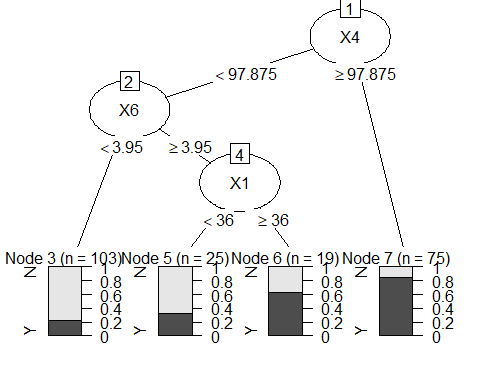
# all Y's - for all groups  
df\_excluding\_X <- within(train, rm(X1, X2, X3, X4, X5, X6, X7))  
DT\_Model\_excluding\_X <- rpart(Target~., data=df\_excluding\_X,   
 control=rpart.control(minsplit=30,   
 minbucket=15,   
 maxdepth=4 ))  
plot(as.party(DT\_Model\_excluding\_X))



# checking accuracy using confusion matrix  
tree.predicted<- predict(DT\_Model\_excluding\_X,test, type='class')  
confusionMatrix(tree.predicted , test$Target)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 31 13  
## Y 7 23  
##   
## Accuracy : 0.7297   
## 95% CI : (0.6139, 0.8265)  
## No Information Rate : 0.5135   
## P-Value [Acc > NIR] : 0.0001207   
##   
## Kappa : 0.4567   
##   
## Mcnemar's Test P-Value : 0.2635525   
##   
## Sensitivity : 0.8158   
## Specificity : 0.6389   
## Pos Pred Value : 0.7045   
## Neg Pred Value : 0.7667   
## Prevalence : 0.5135   
## Detection Rate : 0.4189   
## Detection Prevalence : 0.5946   
## Balanced Accuracy : 0.7273   
##   
## 'Positive' Class : N   
##

# all X's - for all groups  
df\_excluding\_Y <- within(train, rm(Y1, Y2, Y3, Y4, Y5, Y6, Y7))  
DT\_Model\_excluding\_Y <- rpart(Target~., data=df\_excluding\_Y,   
 control=rpart.control(minsplit=30,   
 minbucket=15,   
 maxdepth=4 ))  
plot(as.party(DT\_Model\_excluding\_Y))

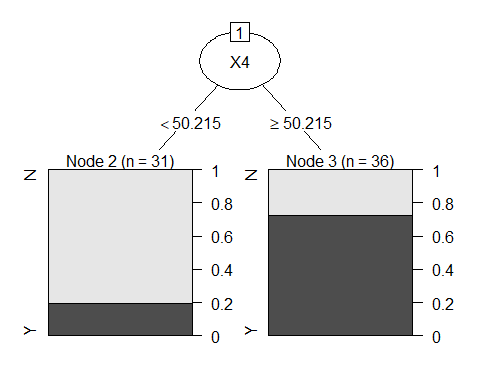


# checking accuracy using confusion matrix  
tree.predicted<- predict(DT\_Model\_excluding\_Y,test, type='class')  
confusionMatrix(tree.predicted , test$Target)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 32 10  
## Y 6 26  
##   
## Accuracy : 0.7838   
## 95% CI : (0.6728, 0.8711)  
## No Information Rate : 0.5135   
## P-Value [Acc > NIR] : 1.494e-06   
##   
## Kappa : 0.566   
##   
## Mcnemar's Test P-Value : 0.4533   
##   
## Sensitivity : 0.8421   
## Specificity : 0.7222   
## Pos Pred Value : 0.7619   
## Neg Pred Value : 0.8125   
## Prevalence : 0.5135   
## Detection Rate : 0.4324   
## Detection Prevalence : 0.5676   
## Balanced Accuracy : 0.7822   
##   
## 'Positive' Class : N   
##

**\*Note: This tree also has the best accuracy out of all trees. Accuracy = 78.38%. Therefore, we notice that the trees with best accuracy are the ones where we consider all both groups (0,1), with or without Y, as the splits are made on X’s.**

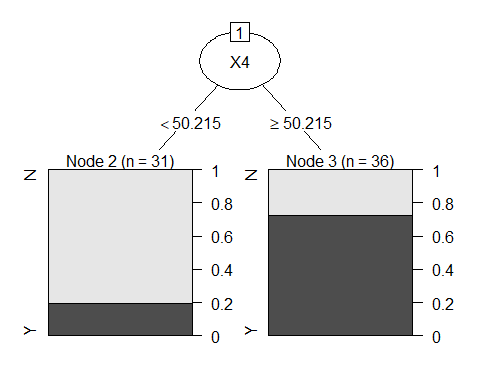
# All X,Y, group 0  
df\_all\_XY\_group0 <- train[train$Group == 0,]  
DT\_Model\_all\_XY\_group0 <- rpart(Target~., data=df\_all\_XY\_group0,   
 control=rpart.control(minsplit=30,   
 minbucket=15,   
 maxdepth=4 ))  
plot(as.party(DT\_Model\_all\_XY\_group0))



# checking accuracy using confusion matrix  
tree.predicted<- predict(DT\_Model\_all\_XY\_group0,test, type='class')  
confusionMatrix(tree.predicted , test$Target)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 10 7  
## Y 28 29  
##   
## Accuracy : 0.527   
## 95% CI : (0.4075, 0.6443)  
## No Information Rate : 0.5135   
## P-Value [Acc > NIR] : 0.4541756   
##   
## Kappa : 0.0677   
##   
## Mcnemar's Test P-Value : 0.0007232   
##   
## Sensitivity : 0.2632   
## Specificity : 0.8056   
## Pos Pred Value : 0.5882   
## Neg Pred Value : 0.5088   
## Prevalence : 0.5135   
## Detection Rate : 0.1351   
## Detection Prevalence : 0.2297   
## Balanced Accuracy : 0.5344   
##   
## 'Positive' Class : N   
##

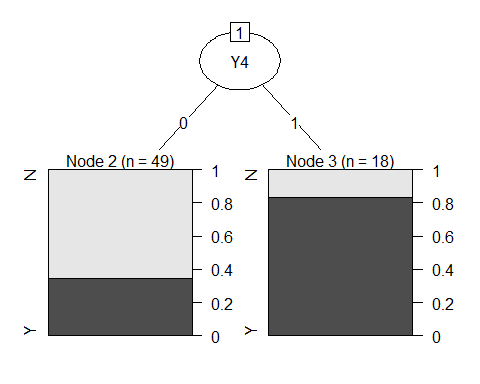
# all X, group 0  
df\_all\_X\_group0 <- train[train$Group == 0,]  
df\_all\_X\_group0 <- within(df\_all\_X\_group0, rm(Y1, Y2, Y3, Y4, Y5, Y6, Y7))  
DT\_Model\_all\_X\_group0 <- rpart(Target~., data=df\_all\_X\_group0,   
 control=rpart.control(minsplit=30,   
 minbucket=15,   
 maxdepth=4 ))  
plot(as.party(DT\_Model\_all\_X\_group0))



# checking accuracy using confusion matrix  
tree.predicted<- predict(DT\_Model\_all\_X\_group0,test, type='class')  
confusionMatrix(tree.predicted , test$Target)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 10 7  
## Y 28 29  
##   
## Accuracy : 0.527   
## 95% CI : (0.4075, 0.6443)  
## No Information Rate : 0.5135   
## P-Value [Acc > NIR] : 0.4541756   
##   
## Kappa : 0.0677   
##   
## Mcnemar's Test P-Value : 0.0007232   
##   
## Sensitivity : 0.2632   
## Specificity : 0.8056   
## Pos Pred Value : 0.5882   
## Neg Pred Value : 0.5088   
## Prevalence : 0.5135   
## Detection Rate : 0.1351   
## Detection Prevalence : 0.2297   
## Balanced Accuracy : 0.5344   
##   
## 'Positive' Class : N   
##

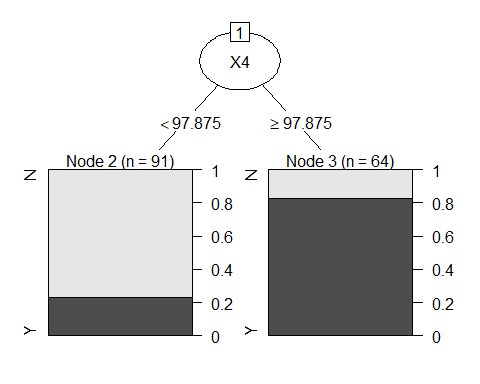
# all Y, group 0  
df\_all\_Y\_group0 <- train[train$Group == 0,]  
df\_all\_Y\_group0 <- within(df\_all\_Y\_group0, rm(X1, X2, X3, X4, X5, X6, X7))  
DT\_Model\_all\_Y\_group0 <- rpart(Target~., data=df\_all\_Y\_group0,   
 control=rpart.control(minsplit=30,   
 minbucket=15,   
 maxdepth=4 ))  
plot(as.party(DT\_Model\_all\_Y\_group0))



# checking accuracy using confusion matrix  
tree.predicted<- predict(DT\_Model\_all\_Y\_group0,test, type='class')  
confusionMatrix(tree.predicted , test$Target)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 25 12  
## Y 13 24  
##   
## Accuracy : 0.6622   
## 95% CI : (0.5428, 0.7681)  
## No Information Rate : 0.5135   
## P-Value [Acc > NIR] : 0.006916   
##   
## Kappa : 0.3243   
##   
## Mcnemar's Test P-Value : 1.000000   
##   
## Sensitivity : 0.6579   
## Specificity : 0.6667   
## Pos Pred Value : 0.6757   
## Neg Pred Value : 0.6486   
## Prevalence : 0.5135   
## Detection Rate : 0.3378   
## Detection Prevalence : 0.5000   
## Balanced Accuracy : 0.6623   
##   
## 'Positive' Class : N   
##

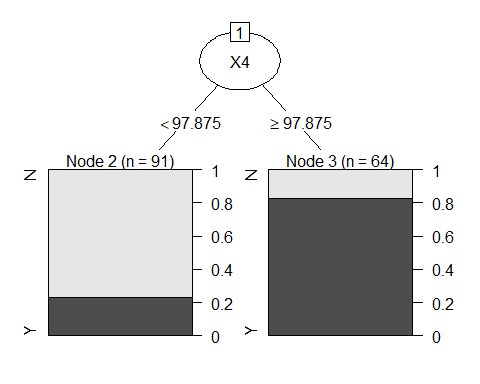
# All X,Y, group 1  
df\_all\_XY\_group1 <- train[train$Group == 1,]  
DT\_Model\_all\_XY\_group1 <- rpart(Target~., data=df\_all\_XY\_group1,   
 control=rpart.control(minsplit=30,   
 minbucket=15,   
 maxdepth=4 ))  
plot(as.party(DT\_Model\_all\_XY\_group1))



# checking accuracy using confusion matrix  
tree.predicted<- predict(DT\_Model\_excluding\_X,test, type='class')  
confusionMatrix(tree.predicted , test$Target)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 31 13  
## Y 7 23  
##   
## Accuracy : 0.7297   
## 95% CI : (0.6139, 0.8265)  
## No Information Rate : 0.5135   
## P-Value [Acc > NIR] : 0.0001207   
##   
## Kappa : 0.4567   
##   
## Mcnemar's Test P-Value : 0.2635525   
##   
## Sensitivity : 0.8158   
## Specificity : 0.6389   
## Pos Pred Value : 0.7045   
## Neg Pred Value : 0.7667   
## Prevalence : 0.5135   
## Detection Rate : 0.4189   
## Detection Prevalence : 0.5946   
## Balanced Accuracy : 0.7273   
##   
## 'Positive' Class : N   
##

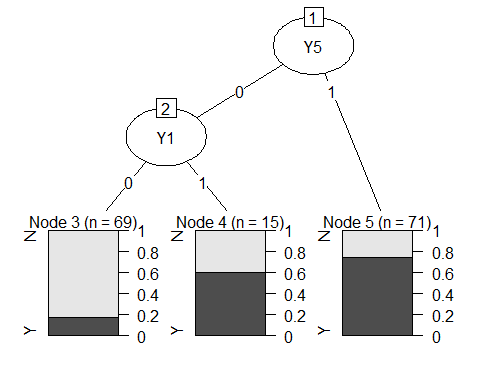
# all X, group 1  
df\_all\_X\_group1 <- train[train$Group == 1,]  
df\_all\_X\_group1 <- within(df\_all\_X\_group1, rm(Y1, Y2, Y3, Y4, Y5, Y6, Y7))  
DT\_Model\_all\_X\_group1 <- rpart(Target~., data=df\_all\_X\_group1,   
 control=rpart.control(minsplit=30,   
 minbucket=15,   
 maxdepth=4 ))  
plot(as.party(DT\_Model\_all\_X\_group1))



# checking accuracy using confusion matrix  
tree.predicted<- predict(DT\_Model\_all\_X\_group1,test, type='class')  
confusionMatrix(tree.predicted , test$Target)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 34 14  
## Y 4 22  
##   
## Accuracy : 0.7568   
## 95% CI : (0.6431, 0.849)  
## No Information Rate : 0.5135   
## P-Value [Acc > NIR] : 1.544e-05   
##   
## Kappa : 0.5096   
##   
## Mcnemar's Test P-Value : 0.03389   
##   
## Sensitivity : 0.8947   
## Specificity : 0.6111   
## Pos Pred Value : 0.7083   
## Neg Pred Value : 0.8462   
## Prevalence : 0.5135   
## Detection Rate : 0.4595   
## Detection Prevalence : 0.6486   
## Balanced Accuracy : 0.7529   
##   
## 'Positive' Class : N   
##

# all Y, group 1  
df\_all\_Y\_group1 <- train[train$Group == 1,]  
df\_all\_Y\_group1 <- within(df\_all\_Y\_group1, rm(X1, X2, X3, X4, X5, X6, X7))  
DT\_Model\_all\_Y\_group1 <- rpart(Target~., data=df\_all\_Y\_group1,   
 control=rpart.control(minsplit=30,   
 minbucket=15,   
 maxdepth=4 ))  
plot(as.party(DT\_Model\_all\_Y\_group1))



# checking accuracy using confusion matrix  
tree.predicted<- predict(DT\_Model\_all\_Y\_group1,test, type='class')  
confusionMatrix(tree.predicted , test$Target)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction N Y  
## N 22 6  
## Y 16 30  
##   
## Accuracy : 0.7027   
## 95% CI : (0.5852, 0.8034)  
## No Information Rate : 0.5135   
## P-Value [Acc > NIR] : 0.0007291   
##   
## Kappa : 0.4093   
##   
## Mcnemar's Test P-Value : 0.0550088   
##   
## Sensitivity : 0.5789   
## Specificity : 0.8333   
## Pos Pred Value : 0.7857   
## Neg Pred Value : 0.6522   
## Prevalence : 0.5135   
## Detection Rate : 0.2973   
## Detection Prevalence : 0.3784   
## Balanced Accuracy : 0.7061   
##   
## 'Positive' Class : N   
##