Decision Tree & Random Forest

Elizabeth Jones

5/11/2022

#Import Dataset

```
#install.packages("Rattle")
#install.packages("Rose")
#install.packages("randomForest")
#install.packages("mlbench")
#install.packages("smotefamily")
#library(mlbench)
#remotes::install_github("cran/DMwR")
#library(DMwR)
#library(smotefamily)
#library(ROSE)
#library(rattle)
#library(readr)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(rpart)
library(rpart.plot)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:purrr':
##
##     cross
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```r
library(e1071)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(randomForest)
```

```
## randomForest 4.7-1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
car_insurance_data <- read.csv("C:/Users/eljon/OneDrive/Desktop/IST 707/Final Project/car insura
nce data.csv")
```

#Review data and remove NA’s

```
#Check the number of variables for the outcome variable
table(car_insurance_data$OUTCOME)
```

```
##
##    0    1
## 6867 3133
```

```
#Check for NA's
sum(is.na(car_insurance_data))
```

```
## [1] 1939
```

```
#remove rows with NAs
insured<-na.omit(car_insurance_data)

#verify NAs are removed
sum(is.na(insured))
```

```
## [1] 0
```

```
#Verify number of variables
table(insured$OUTCOME)
```

```
##
##    0    1
## 5613 2536
```

#Add column for rank

```
#Good customers = No Accidents
#OK customers = 1 Accident
#Bad customers = More than 1 Accident

r.insured<-insured %>% mutate (RANKED = case_when(PAST_ACCIDENTS == 0 ~ "good", PAST_ACCIDENTS =
= 1 ~"ok", PAST_ACCIDENTS > 1 ~ "bad"))
```

##Remove columns for gender, race & id & past accidents

```
d.insured<-r.insured[,-c(1,3,4,8)]
```

#Discrentize columns

```
# DUI = 0 equals no and 1 equals yes
# SPEEDING_VIOLATIONS = 0 equal no and 1 equals yes

d.insured<-d.insured %>% mutate (DUIS = case_when(DUIS == 0 ~ "0", DUIS >= 1 ~ "1"),
                                 SPEEDING_VIOLATIONS = case_when(SPEEDING_VIOLATIONS == 0 ~ "0",
SPEEDING_VIOLATIONS >= 1 ~ "1"))
```

#Factor columns for decision tree

```
f.insured = d.insured |> mutate_if(is.character, as.factor)
f.insured = f.insured |> mutate_if(is.numeric, as.factor)
```

#Split dataset into train and test set

```
set.seed(111)

trainList <-
  createDataPartition(y=f.insured$OUTCOME,p=0.4,list=FALSE)

insured.trainSet <-f.insured[trainList,]
insured.testSet <- f.insured[-trainList,]

#Check ratio of variables
table(insured.testSet$OUTCOME)
```

```
##
##    0    1
## 3367 1521
```

```
table(insured.trainSet$OUTCOME)
```

```
##
##    0    1
## 2246 1015
```

#Create first decision tree model

```
first.model<-rpart(OUTCOME ~.,data = insured.trainSet, method = 'class')

printcp(first.model)
```

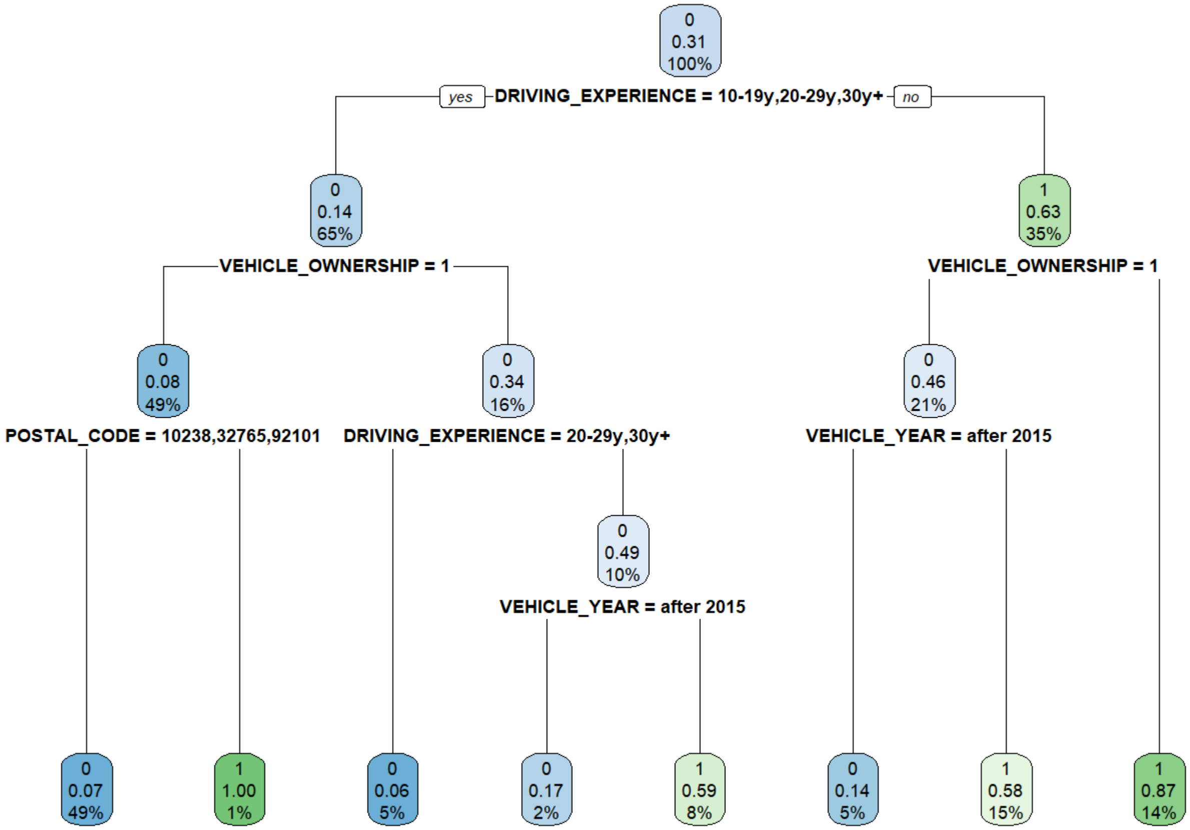
```
##
## Classification tree:
## rpart(formula = OUTCOME ~ ., data = insured.trainSet, method = "class")
##
## Variables actually used in tree construction:
## [1] DRIVING_EXPERIENCE POSTAL_CODE        VEHICLE_OWNERSHIP  VEHICLE_YEAR
##
## Root node error: 1015/3261 = 0.31125
##
## n= 3261
##
##         CP nsplit rel error  xerror     xstd
## 1 0.288670      0   1.00000 1.00000 0.026049
## 2 0.063054      1   0.71133 0.71133 0.023359
## 3 0.016502      3   0.58522 0.58522 0.021715
## 4 0.010000      7   0.51921 0.53005 0.020882
```

```
varImp(first.model)
```

```
##                        Overall
## AGE                 387.488726
## ANNUAL_MILEAGE       23.843284
## DRIVING_EXPERIENCE  440.709228
## EDUCATION             1.993395
## INCOME              350.857791
## MARRIED              40.143460
## PAST_ACCIDENTS      208.593842
## POSTAL_CODE          75.678772
## VEHICLE_OWNERSHIP   363.556663
## VEHICLE_YEAR        173.930907
## CHILDREN              0.000000
## VEHICLE_TYPE          0.000000
## SPEEDING_VIOLATIONS   0.000000
## DUIS                  0.000000
## RANKED                0.000000
```

#Plot first decision tree model

```
rpart.plot(first.model)
```



#Check model accuracy

```
prediction <- predict(first.model,newdata=insured.testSet, type = "class")
confusionMatrix(prediction,insured.testSet$OUTCOME)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2801  255
##          1  566 1266
##
##                Accuracy : 0.832
##                  95% CI : (0.8213, 0.8424)
##     No Information Rate : 0.6888
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.629
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8319
##             Specificity : 0.8323
##          Pos Pred Value : 0.9166
##          Neg Pred Value : 0.6910
##              Prevalence : 0.6888
##          Detection Rate : 0.5730
##    Detection Prevalence : 0.6252
##       Balanced Accuracy : 0.8321
##
##        'Positive' Class : 0
##
```

#Create second decision tree model with loss matrix

```
loss<-matrix(c(0,10,1,0),ncol=2)
second.model<-rpart(OUTCOME ~.,data = insured.trainSet, method = 'class', parms = list(loss=los
s))
```

#Check second model accuracy

```
prediction2<-predict(second.model,newdata=insured.testSet, type = "class")
confusionMatrix(prediction2,insured.testSet$OUTCOME)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2297  102
##          1 1070 1419
##
##                Accuracy : 0.7602
##                  95% CI : (0.748, 0.7721)
##     No Information Rate : 0.6888
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5238
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6822
##             Specificity : 0.9329
##          Pos Pred Value : 0.9575
##          Neg Pred Value : 0.5701
##              Prevalence : 0.6888
##          Detection Rate : 0.4699
##    Detection Prevalence : 0.4908
##       Balanced Accuracy : 0.8076
##
##        'Positive' Class : 0
##
```

#Create third decision tree model with loss matrix and balanced training set

```
#balance training set
s.insured<-DMwR::SMOTE(OUTCOME ~., insured.trainSet, perc.over = 100, perc.under = 200)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
#verify ratio
prop.table(table(s.insured$OUTCOME))
```

```
##
##   0   1
## 0.5 0.5
```

```
#update loss matrix
loss2<-matrix(c(0,2,1,0),ncol=2)

#make model
third.model<-rpart(OUTCOME ~.,data = s.insured, method = 'class', parms = list(loss=loss2))
```

#Check third model accuracy

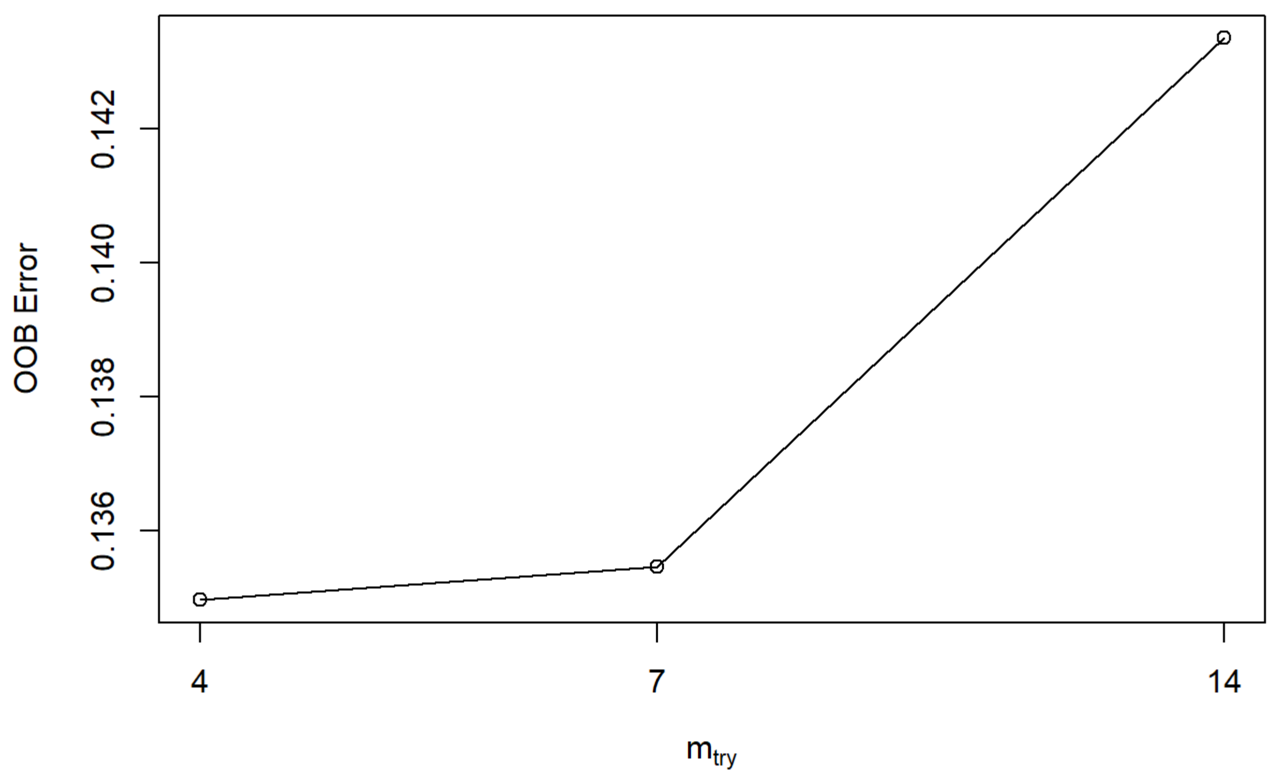
```
prediction3<-predict(third.model,newdata=insured.testSet, type = "class")
confusionMatrix(prediction3,insured.testSet$OUTCOME)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2422  161
##          1  945 1360
##
##                Accuracy : 0.7737
##                  95% CI : (0.7617, 0.7854)
##     No Information Rate : 0.6888
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5375
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7193
##             Specificity : 0.8941
##          Pos Pred Value : 0.9377
##          Neg Pred Value : 0.5900
##              Prevalence : 0.6888
##          Detection Rate : 0.4955
##    Detection Prevalence : 0.5284
##       Balanced Accuracy : 0.8067
##
##        'Positive' Class : 0
##
```

#Create Random Forest Model with balanced training set

```
#tune model
tuneRF(x = s.insured[,1:14],
       y = s.insured$OUTCOME,
       ntreeTry = 500,
       mtryStart = 14,
       trace = FALSE)
```

```
## 0.05498282 0.05
## 0.003636364 0.05
```




```
##        mtry  OOBError
## 4.OOB     4 0.1349754
## 7.OOB     7 0.1354680
## 14.OOB   14 0.1433498
```

```
#make model
fourth.model<- randomForest(formula = OUTCOME ~., data = s.insured, mtry = 7, ntree = 500)
```

#Check fourth model accuracy

```
prediction4<-predict(fourth.model,newdata=insured.testSet, type = "class")
confusionMatrix(prediction4,insured.testSet$OUTCOME)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    0    1
##          0 2584  267
##          1  783 1254
##                                           
##                Accuracy : 0.7852          
##                  95% CI : (0.7734, 0.7966)
##     No Information Rate : 0.6888          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.5415          
##                                           
##  Mcnemar's Test P-Value : < 2.2e-16       
##                                           
##             Sensitivity : 0.7674          
##             Specificity : 0.8245          
##          Pos Pred Value : 0.9063          
##          Neg Pred Value : 0.6156          
##              Prevalence : 0.6888          
##          Detection Rate : 0.5286          
##    Detection Prevalence : 0.5833          
##       Balanced Accuracy : 0.7960          
##                                           
##        'Positive' Class : 0               
## 
```