DT Scrum cycle Exercise

Phillip Llewellyn

7/19/2021

Decision Tree Scrum Cycle 7

Step 1

- Load data and get summaries

```
data <-read.csv("BankLoan_Dataset_2021-Clean.csv")  #HR.csv
#str(data)
data$lead <- as.factor(data$lead)
data$won <- as.factor(data$won)
summary(data)
```

```
##       X.1              X             RefNum          agerange
##  Min.   :   1.0   Min.   :   1.0   Min.   :10023467   Length:1123
##  1st Qu.: 281.5   1st Qu.: 281.5   1st Qu.:10023748   Class :character
##  Median : 562.0   Median : 562.0   Median :10024028   Mode  :character
##  Mean   : 562.0   Mean   : 562.0   Mean   :10024028
##  3rd Qu.: 842.5   3rd Qu.: 842.5   3rd Qu.:10024308
##  Max.   :1123.0   Max.   :1123.0   Max.   :10024589
##       age            job              marital           education
##  Min.   :22.00   Length:1123        Length:1123        Length:1123
##  1st Qu.:35.00   Class :character   Class :character   Class :character
##  Median :42.00   Mode  :character   Mode  :character   Mode  :character
##  Mean   :42.46
##  3rd Qu.:50.00
##  Max.   :61.00
##     balance          housing              loan              month
##  Min.   : -932.0   Length:1123        Length:1123        Length:1123
##  1st Qu.:   23.0   Class :character   Class :character   Class :character
##  Median :  167.0   Mode  :character   Mode  :character   Mode  :character
##  Mean   :  567.3
##  3rd Qu.:  446.0
##  Max.   :58544.0
##      date              duration         deposit       lead
##  Length:1123        Min.   :   2.0   Min.   :  1.80   0:350
##  Class :character   1st Qu.: 130.0   1st Qu.: 43.75   1:773
##  Mode  :character   Median : 203.0   Median : 52.94
##                     Mean   : 270.5   Mean   : 84.09
##                     3rd Qu.: 315.5   3rd Qu.: 92.71
##                     Max.   :2177.0   Max.   :388.68
```

```
##    product            qualified        contacted       won       loanvalue
##  Length:1123        Min.   :0.0000   Min.   :-1.0000   0:626   Min.   : 1526
##  Class :character   1st Qu.:0.0000   1st Qu.: 0.0000   1:497   1st Qu.: 3397
##  Mode  :character   Median :1.0000   Median : 0.0000           Median : 6530
##                     Mean   :0.6073   Mean   : 0.2787           Mean   : 5991
##                     3rd Qu.:1.0000   3rd Qu.: 1.0000           3rd Qu.: 7632
##                     Max.   :1.0000   Max.   : 1.0000           Max.   :12353
##       NPS         contacted_and_won qualified_and_contacted lead_and_qualified
##  Min.   : 3.000   Min.   :0.0000    Min.   :-1.0000         Min.   :0.000
##  1st Qu.: 7.000   1st Qu.:0.0000    1st Qu.: 0.0000         1st Qu.:1.000
##  Median : 7.000   Median :0.0000    Median : 0.0000         Median :1.000
##  Mean   : 7.874   Mean   :0.4426    Mean   : 0.3419         Mean   :0.919
##  3rd Qu.: 9.000   3rd Qu.:1.0000    3rd Qu.: 1.0000         3rd Qu.:1.000
##  Max.   :10.000   Max.   :1.0000    Max.   : 1.0000         Max.   :1.000
```

```
#str(data)
```

Step 2

- Split data into training and testing data

```
# separate the data for an equal split
data_lead_1 = data[data$lead == 1,]
data_lead_0 = data[data$lead == 0,]

#randomize the sampling
set.seed(1098765467)
newDataset_lead_1 <-sample.split(Y=data_lead_1$lead, SplitRatio = 0.35)
newDataset_lead_0 <-sample.split(Y=data_lead_0$lead, SplitRatio = 0.75)

trainData <-rbind(data_lead_1[newDataset_lead_1,], data_lead_0[newDataset_lead_0,])
testData <- rbind(data_lead_1[!newDataset_lead_1,][1:100,], data_lead_0[!newDataset_lead_0,])

summary(trainData$lead)
```

```
##   0   1
## 262 270
```

```
summary(testData$lead)
```

```
##   0   1
##  88 100
```

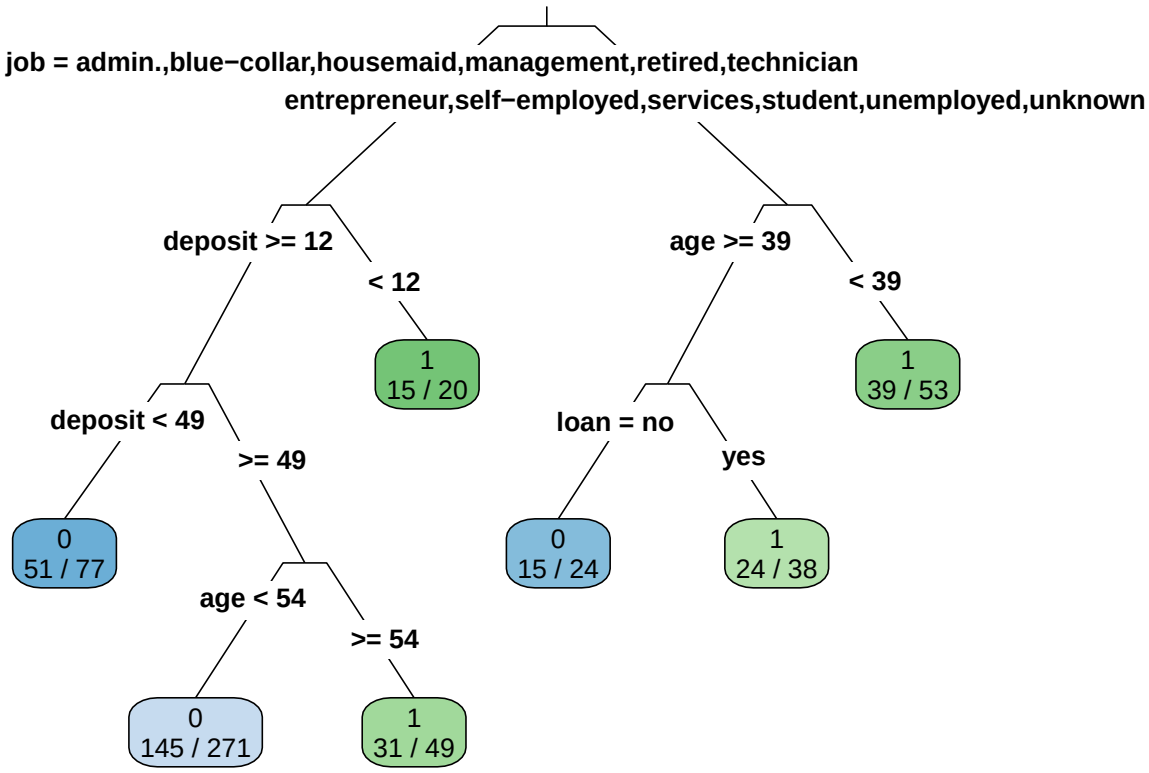
Step 3

- Fit a Decision Tree using training data

```
# The . specifies all other columns ( Class ~ . )
DTmodel <- rpart(lead ~ education + age + job + marital + deposit + balance + loan + housing, method="cl
```

- Target Variable = Class,
- Input Vaiables = All,
- split =gini or information gain
- control = rpart.control for prepruning DT minsplit- min records at node for split to occur, maxdepth - depth of the DT
- Fitting the model

```
rpart.plot(DTmodel, type=3, extra = 2, fallen.leaves = F, cex = 0.8)
```




```
#try extra with 2,8,4, 101
```

- Print out the information

```
#(DTmodel) # detailed summary of splits
DTmodel #prints the rules
```

```
## n= 532
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
```

```
##  1) root 532 262 1 (0.4924812 0.5075188)
##    2) job=admin.,blue-collar,housemaid,management,retired,technician 417 198 0 (0.5251799 0.4748201)
##      4) deposit>=11.7 397 183 0 (0.5390428 0.4609572)
##        8) deposit< 49.07 77  26 0 (0.6623377 0.3376623) *
##        9) deposit>=49.07 320 157 0 (0.5093750 0.4906250)
##         18) age< 53.5 271 126 0 (0.5350554 0.4649446) *
##         19) age>=53.5 49  18 1 (0.3673469 0.6326531) *
##      5) deposit< 11.7 20   5 1 (0.2500000 0.7500000) *
##    3) job=entrepreneur,self-employed,services,student,unemployed,unknown 115  43 1 (0.3739130 0.62608
##      6) age>=38.5 62  29 1 (0.4677419 0.5322581)
##       12) loan=no 24   9 0 (0.6250000 0.3750000) *
##       13) loan=yes 38  14 1 (0.3684211 0.6315789) *
##      7) age< 38.5 53  14 1 (0.2641509 0.7358491) *
```

- Run the second model

```
#DTmodel2 <- J48(as.factor(Class) ~., trainData, control = Weka_control(R = TRUE, M = round(NROW(trainD

#DTmodel2 <- J48(as.factor(left) ~., trainData, control = Weka_control(R = TRUE, M = 50))

#IGDT5model <- J48(as.factor(eReader_Adoption)~., trainData ,control = Weka_control(R = TRUE, M = round

#IGDT10model <- J48(as.factor(eReader_Adoption)~., trainData ,control = Weka_control(R = TRUE, M = roun
```

- Plot the model

```
#plot(DTmodel)
```

Step 4

- Use the fitted model to do predictions for the test data

```
predTest <- predict(DTmodel, testData, type="class")
probTest <- predict(DTmodel, testData, type="prob")

actualTest <- testData$lead
```

Step 5

- Create Confusion Matrix and compute the misclassification error

```
t1 <- table(predictions=predTest, actual = actualTest)
t1 # Confusion matrix
```

```
##            actual
## predictions  0  1
##           0 66 55
##           1 22 45
```

```
accuracy1 <- sum(diag(t1))/sum(t1) * 100
accuracy1
```

```
## [1] 59.04255
```

```
#calculatee sensitivity
sensitivity <- t1[2,2]/sum(t1[2,]) *100
sensitivity
```

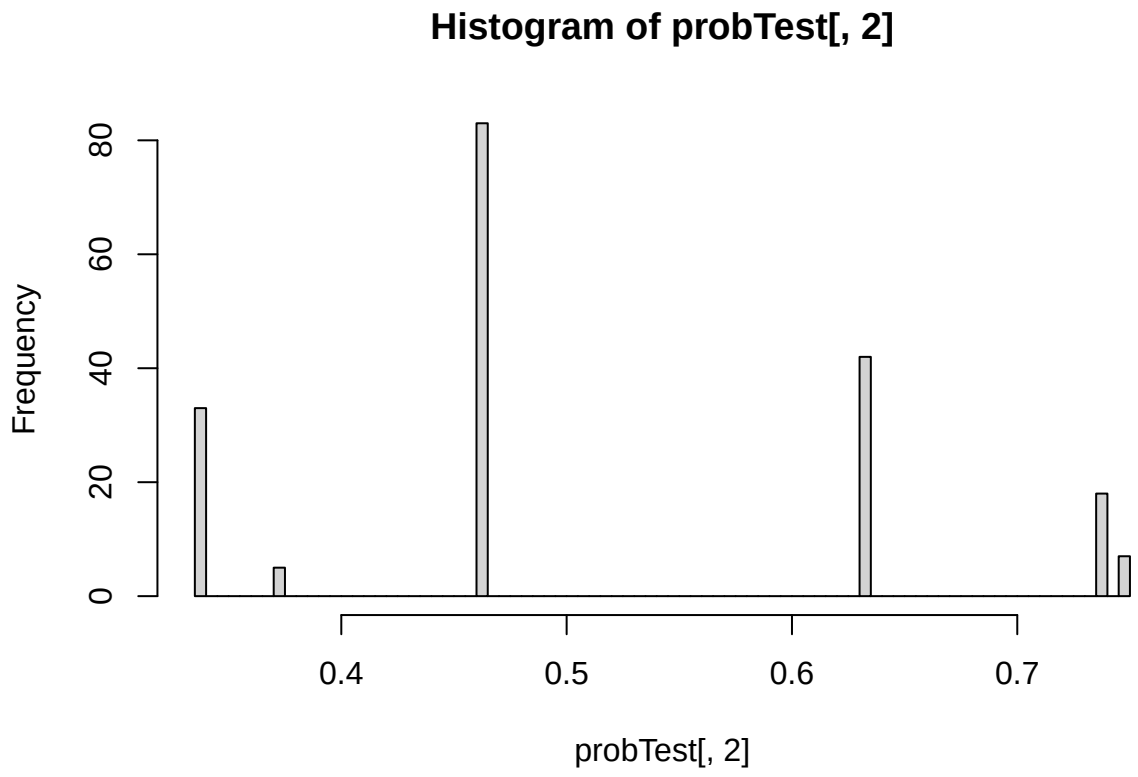
```
## [1] 67.16418
```

```
#calculate specificity
specificity <- t1[1,1]/sum(t1[1,]) *100
specificity
```

```
## [1] 54.54545
```

- Visualization of probabilities

```
hist(probTest[,2], breaks = 100)
```



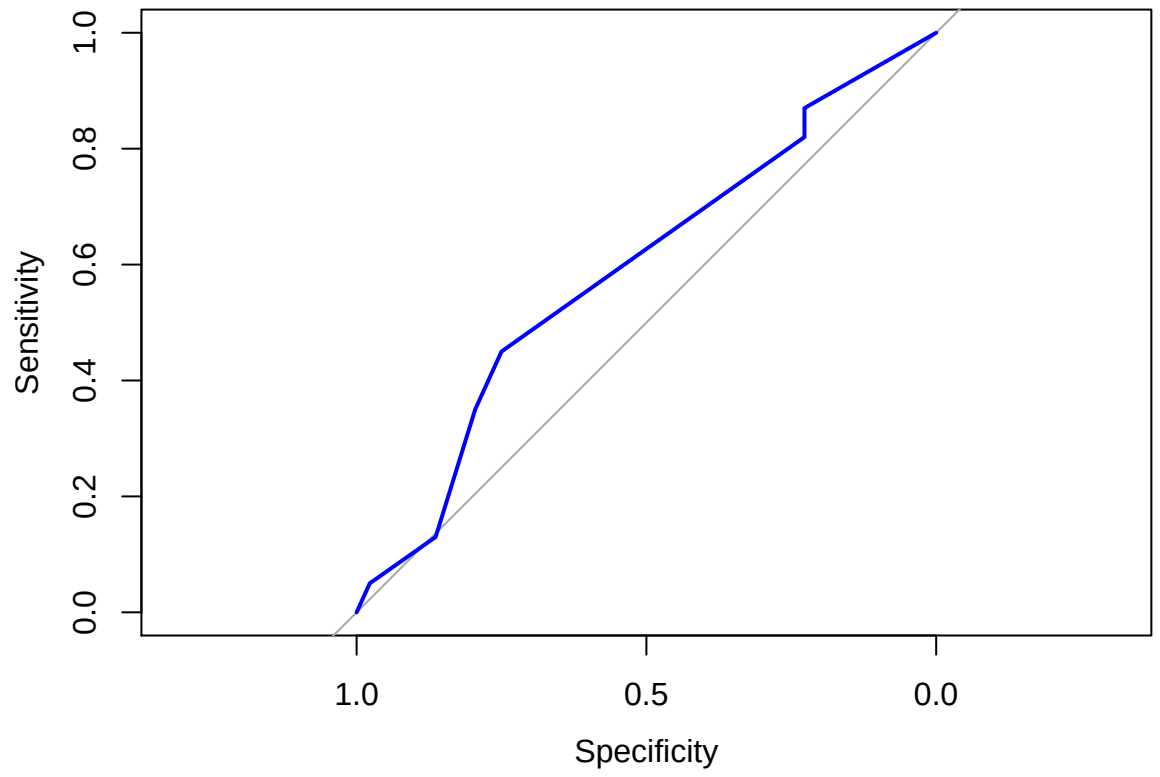
- ROC and Area Under the Curve

```
ROC <- roc(actualTest, probTest[,2])
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(ROC, col="blue")
```




```
AUC <- auc(ROC)
AUC
```

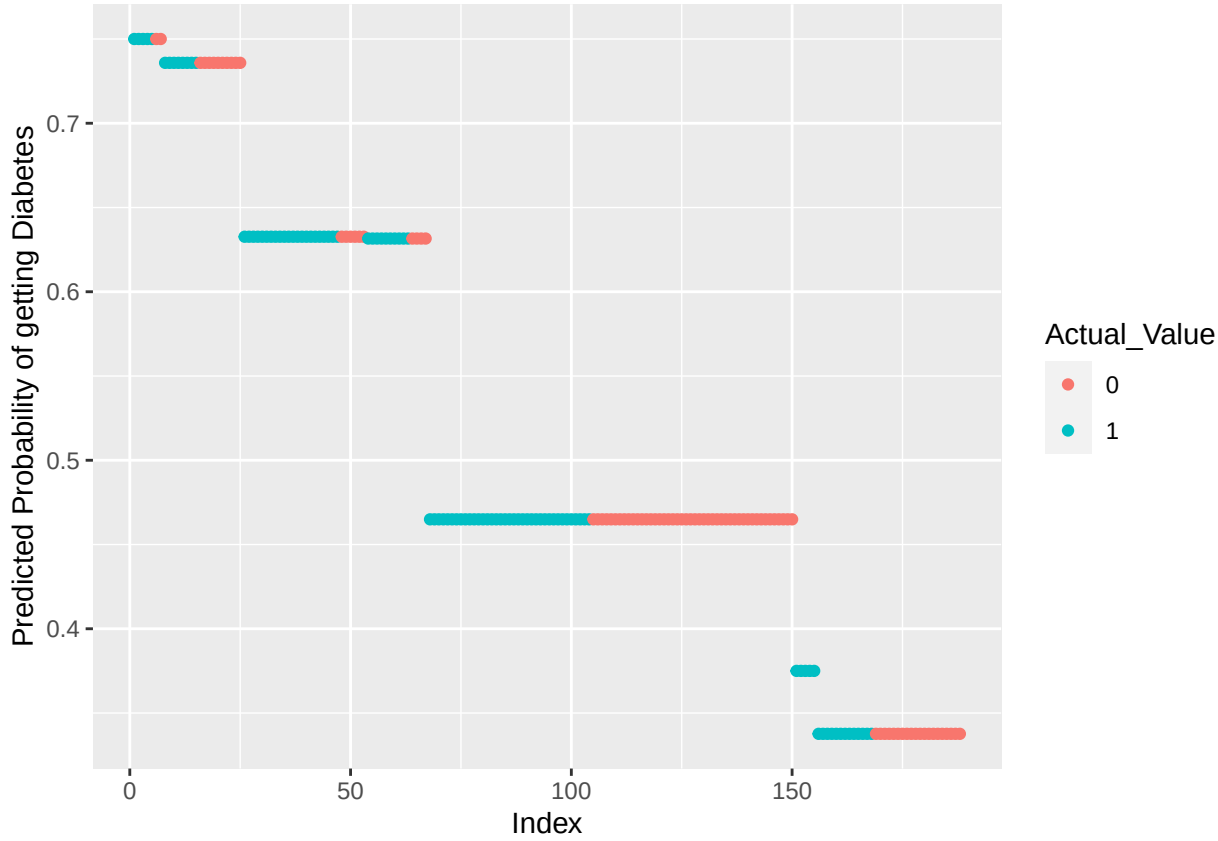
```
## Area under the curve: 0.5898
```

- A new dataframe with Predicted Prob, Actual Value and Predicted Value

```
predicted_data <- data.frame(Probs = probTest, Actual_Value= actualTest ,Predicted_Value = predTest )
  #predicted_data$Probs.0 <- Class 0 Probability
  #predicted_data$Probs.1 <-  Class 1 Probability
predicted_data <- predicted_data[order(predicted_data$Probs.1, decreasing=TRUE),] # Sort on Probabiliti
predicted_data$Rank <- 1:nrow(predicted_data) # Add a new variable rank
```

- plot the graph

```
ggplot(data=predicted_data, aes(x=Rank, y=Probs.1)) +
  geom_point(aes(color = Actual_Value)) + xlab("Index") + ylab("Predicted Probability of getting Diabet
```



Step 6

- Use model to make predictions on newdata. Note we can specify the newData as data.frame with one or many records

```
#newData <- data.frame(Nbr_Preg = 4 , Glucose_test = 100, Triceps_SF=40,BP =95, S_insulin = 150, BMI= 3

#predProbability <-predict(DTmodel, newData, type='prob')
#predProbability

## Performnce measures -
#setseed(1), gini
# Simplicity = 15 leaves
# Accuracy = 0.734
# AUC = 0.7627

#setseed(1), information
# Simplicity = 10 leaves
# Accuracy = 0.71
# AUC = 0.7834
```

Step 7

- EXAMINING STABILITY - Creating Decile Plots for Class 1 or 0 Sort

```
#-----Create empty df-------
#decileDF<- data.frame(matrix(ncol=3,nrow = 0))
#colnames(decileDF)<- c("Decile","per_correct_preds","No_correct_Preds","cum_preds")
#-----Initialize varables
#num_of_deciles=10
#Obs_per_decile<-nrow(predicted_data)/num_of_deciles
#decile_count=1
#start=1
#stop=(start-1) + Obs_per_decile
#prev_cum_pred<-0
#x=0
#-----Loop through DF and create deciles
#while (x < nrow(predicted_data)) {
#  subset<-predicted_data[c(start:stop),]
#  correct_count<- ifelse(subset$Actual_Value==subset$Predicted_Value,1,0)
#  no_correct_Preds<-sum(correct_count,na.rm = TRUE)
#  per_correct_Preds<-(no_correct_Preds/Obs_per_decile)*100
#  cum_preds<-no_correct_Preds+prev_cum_pred
#  addRow<-data.frame("Decile"=decile_count,"per_correct_preds"=per_correct_Preds,"No_correct_Preds"=no
#  decileDF<-rbind(decileDF,addRow)
#  prev_cum_pred<-prev_cum_pred+no_correct_Preds
#  start<-stop+1
#  stop=(start-1) + Obs_per_decile
#  x<-x+Obs_per_decile
#  decile_count<-decile_count+1
#}
#------Stability plot (correct preds per decile)
#plot(decileDF$Decile,decileDF$per_correct_preds,type = "l",xlab = "Decile",ylab = "Percentage of corre
```