

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    18-30 : 97
## 1st Qu.: 281.5    1st Qu.: 281.5    1st Qu.:10023748    31-40 :430
## Median : 562.0    Median : 562.0    Median :10024028    41-60 :594
## Mean    : 562.0    Mean    : 562.0    Mean    :10024028    over 60: 2
## 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    blue-collar:348    divorced:165    primary :198
## 1st Qu.:35.00    technician :167    married :709    secondary:626
## Median :42.00    admin.      :153    single  :249    tertiary :201
## Mean     :42.46    management :148                unknown  : 98
## 3rd Qu.:50.00    services   :145
## Max.     :61.00    retired    : 49
##           (Other) :113
##           balance    housing    loan           month           date
## Min.      : -932.0    no :129    no :624    april      : 94    1/1/2018: 94
## 1st Qu.:   23.0    yes:994    yes:499    february: 94    2/1/2018: 94
## Median :  167.0                january : 94    3/1/2018: 94
## Mean     :  567.3                july    : 94    4/1/2018: 94
## 3rd Qu.:  446.0                june    : 94    5/1/2018: 94
## Max.     :58544.0                march   : 94    6/1/2018: 94
##           (Other) :559    (Other) :559
##           duration    deposit    lead           product    qualified
## Min.      : 2.0    Min.      : 1.80    0:350    auto      :217    Min.      :0.0000
## 1st Qu.: 130.0    1st Qu.: 43.75    1:773    business  : 25    1st Qu.:0.0000
## Median : 203.0    Median : 52.94                mortgage :312    Median :1.0000
```

```
## Mean : 270.5 Mean : 84.09 no product:350 Mean :0.6073
## 3rd Qu.: 315.5 3rd Qu.: 92.71 unsecured :219 3rd Qu.:1.0000
## Max. :2177.0 Max. :388.68 Max. :1.0000
##
## contacted won loanvalue NPS contacted_and_won
## Min. :-1.0000 0:626 Min. : 1526 Min. : 3.000 Min. :0.0000
## 1st Qu.: 0.0000 1:497 1st Qu.: 3397 1st Qu.: 7.000 1st Qu.:0.0000
## Median : 0.0000 Median : 6530 Median : 7.000 Median :0.0000
## Mean : 0.2787 Mean : 5991 Mean : 7.874 Mean :0.4426
## 3rd Qu.: 1.0000 3rd Qu.: 7632 3rd Qu.: 9.000 3rd Qu.:1.0000
## Max. : 1.0000 Max. :12353 Max. :10.000 Max. :1.0000
##
## qualified_and_contacted lead_and_qualified
## Min. :-1.0000 Min. :0.000
## 1st Qu.: 0.0000 1st Qu.:1.000
## Median : 0.0000 Median :1.000
## Mean : 0.3419 Mean :0.919
## 3rd Qu.: 1.0000 3rd Qu.:1.000
## 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(1)
newDataset_lead_1 <-sample.split(Y=data_lead_1$lead, SplitRatio = 0.28)
newDataset_lead_0 <-sample.split(Y=data_lead_0$lead, SplitRatio = 0.6)

trainData <-rbind(data_lead_1[newDataset_lead_1,], data_lead_0[newDataset_lead_0,])
testData <- rbind(data_lead_1[!newDataset_lead_1,][1:150,], data_lead_0[!newDataset_lead_0,])

summary(trainData$lead)
```

```
## 0 1
## 210 216
```

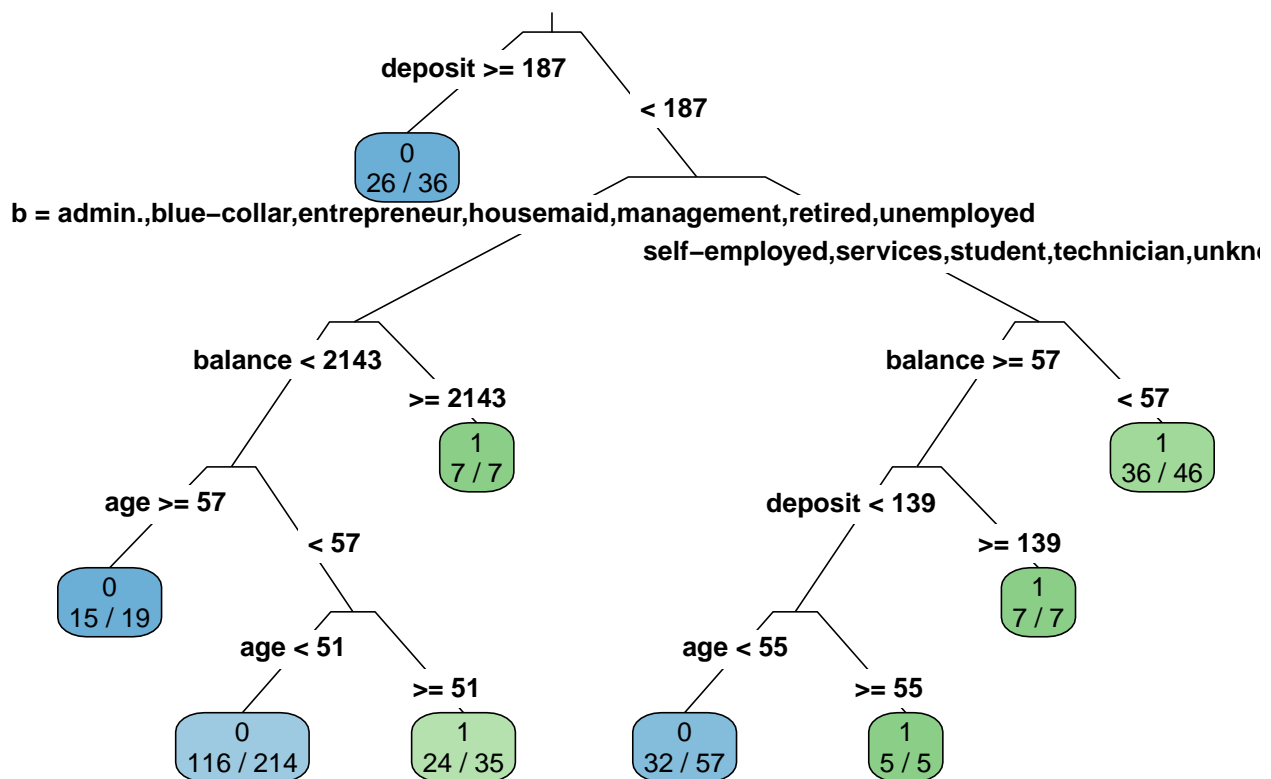
```
summary(testData$lead)
```

```
## 0 1
## 140 150
```

- Fit a Decision Tree using training data

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

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



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

- 3

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

```
## n= 426
##
## node), split, n, loss, yval, (yprob)
##      * denotes terminal node
##
## 1) root 426 210 1 (0.4929577 0.5070423)
##    2) deposit>=186.655 36 10 0 (0.7222222 0.2777778) *
##    3) deposit< 186.655 390 184 1 (0.4717949 0.5282051)
##      6) job=admin.,blue-collar,entrepreneur,housemaid,management,retired,unemployed 275 133 0 (0.516
##      12) balance< 2142.5 268 126 0 (0.5298507 0.4701493)
##        24) age>=56.5 19 4 0 (0.7894737 0.2105263) *
##        25) age< 56.5 249 122 0 (0.5100402 0.4899598)
##          50) age< 50.5 214 98 0 (0.5420561 0.4579439) *
##          51) age>=50.5 35 11 1 (0.3142857 0.6857143) *
##        13) balance>=2142.5 7 0 1 (0.0000000 1.0000000) *
##      7) job=self-employed,services,student,technician,unknown 115 42 1 (0.3652174 0.6347826)
##        14) balance>=56.5 69 32 1 (0.4637681 0.5362319)
##          28) deposit< 139.04 62 30 0 (0.5161290 0.4838710)
##            56) age< 54.5 57 25 0 (0.5614035 0.4385965) *
##            57) age>=54.5 5 0 1 (0.0000000 1.0000000) *
##          29) deposit>=139.04 7 0 1 (0.0000000 1.0000000) *
##        15) balance< 56.5 46 10 1 (0.2173913 0.7826087) *
```

- 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$won
```

Step 5

- Create Confusion Matrix and compute the misclassification error

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

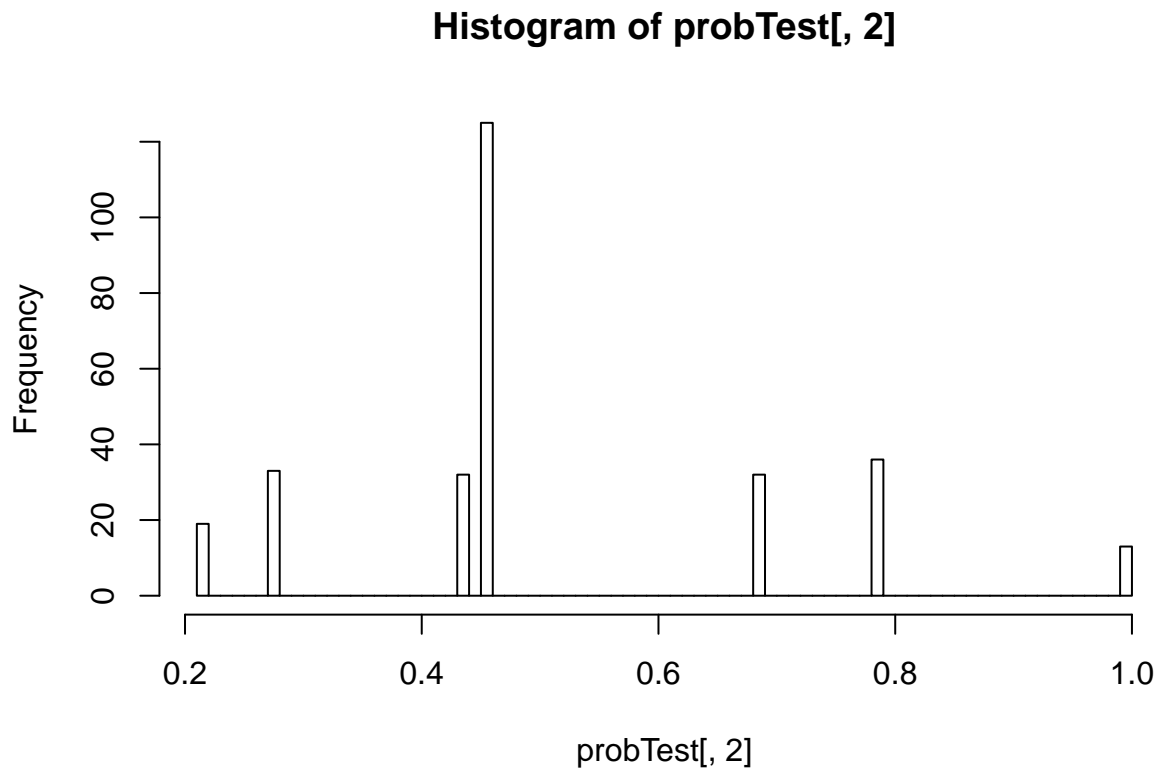
```
##           actual
## predictions  0   1
##           0 177  32
##           1  72   9
```

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

```
## [1] 0.6413793
```

- Visualization of probabilities

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



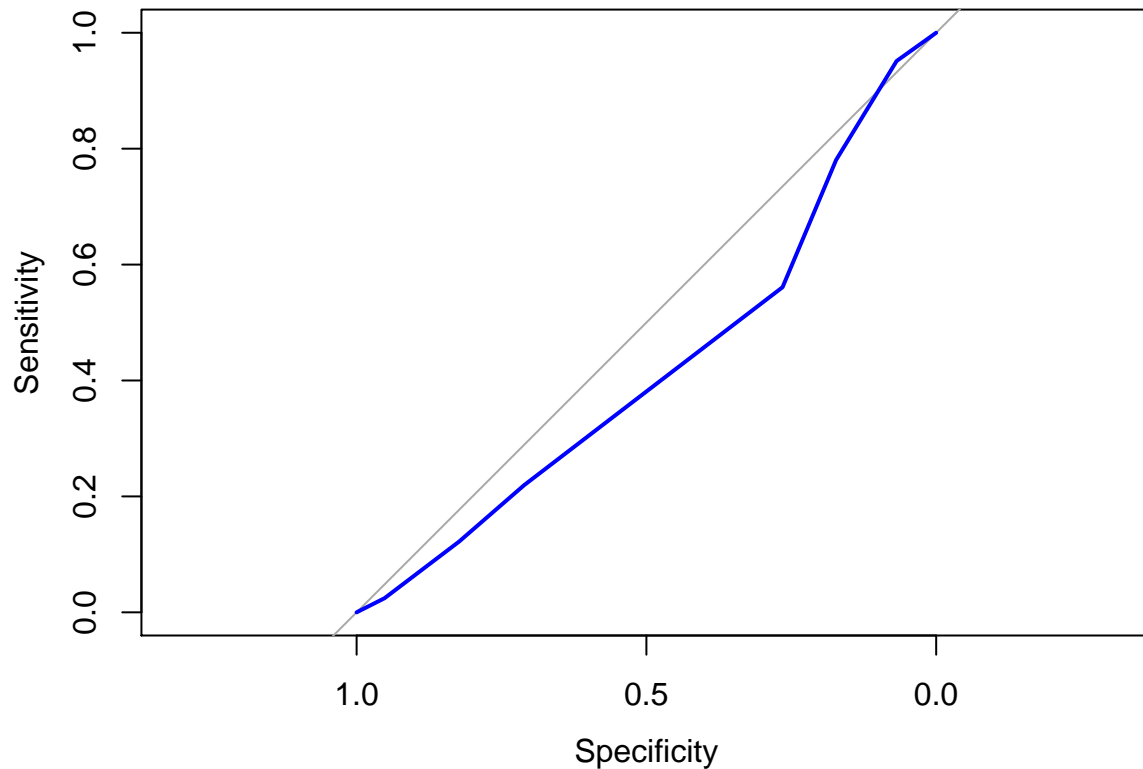
- ROC and Area Under the Curve

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

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

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

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



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

```
AUC
```

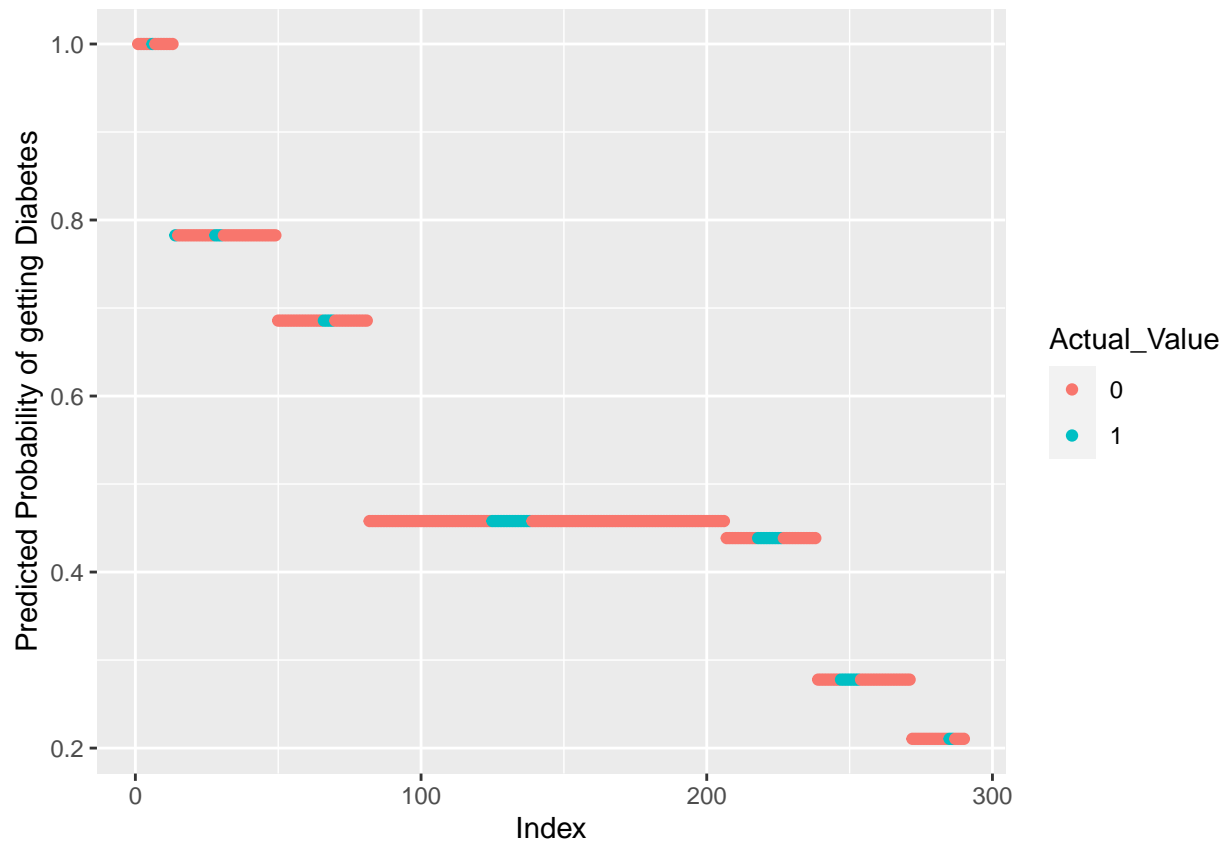
```
## Area under the curve: 0.4221
```

- 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$Probs.1, decreasing=TRUE),] # Sort on Probability
predicted_data$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 Diabetes")
```



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 variables
#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
```