Assignment #4

Minghao Du

# Set-up

Load packages.

rm(list = ls())  
library(readxl)  
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.0.5

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.0.5 v dplyr 1.0.3  
## v tidyr 1.1.2 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(ggplot2)

For this and the next assignments (Assignment 3 and 4), you will use the dataset from <https://www.ibm.com/communities/analytics/watson-analytics-blog/it-help-desk/>. (the link is now invalid.) This data is from an Information Technology (IT) department interested in improving the satisfaction of customers.  
To start their analysis, they constructed this data set of 100,000 closed tickets that were filed at their help desk.

Load the dataset and save it as ithelp.

ithelp <- read\_excel("WA\_Fn-UseC\_-IT-Help-Desk.xlsx")

glimpse(ithelp)

## Rows: 100,000  
## Columns: 10  
## $ ticket <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ~  
## $ Requestor <dbl> 1929, 1587, 925, 413, 318, 858, 1978, 1209, 887, 17~  
## $ RequestorSeniority <chr> "1 - Junior", "2 - Regular", "2 - Regular", "4 - Ma~  
## $ ITOwner <dbl> 50, 15, 15, 22, 22, 38, 10, 1, 14, 46, 1, 50, 11, 2~  
## $ FiledAgainst <chr> "Systems", "Software", "Access/Login", "Systems", "~  
## $ TicketType <chr> "Issue", "Request", "Request", "Request", "Request"~  
## $ Severity <chr> "2 - Normal", "1 - Minor", "2 - Normal", "2 - Norma~  
## $ Priority <chr> "0 - Unassigned", "1 - Low", "0 - Unassigned", "0 -~  
## $ daysOpen <dbl> 3, 5, 0, 20, 1, 0, 9, 15, 6, 1, 7, 17, 10, 4, 7, 11~  
## $ Satisfaction <chr> "1 - Unsatisfied", "1 - Unsatisfied", "0 - Unknown"~

A total of 100,000 rows. Each row represent an individual request. The data includes:

* Requestor: employee who submitted the ticket
* RequestorSeniority: employee’s seniority within the company
* ITOwner: IT employee who serviced the ticket
* FileAgainst: functional area against which the ticket was filed (systems, software, hardware, access)
* TicketType: whether the ticket was a request for new services or an issue with existing services
* Severity: submitter-assigned severity of the ticket
* Priority: IT-assigned priority of the ticket
* daysOpen: number of days the ticket was open
* Satisfaction: satisfaction with the resolution of the ticket (reported by the submitter)

summary(ithelp)

## ticket Requestor RequestorSeniority ITOwner   
## Min. : 1 Min. : 1 Length:100000 Min. : 1.00   
## 1st Qu.: 25001 1st Qu.: 499 Class :character 1st Qu.:13.00   
## Median : 50001 Median : 999 Mode :character Median :26.00   
## Mean : 50001 Mean : 999 Mean :25.46   
## 3rd Qu.: 75000 3rd Qu.:1499 3rd Qu.:38.00   
## Max. :100000 Max. :2000 Max. :50.00   
## FiledAgainst TicketType Severity Priority   
## Length:100000 Length:100000 Length:100000 Length:100000   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
##   
##   
##   
## daysOpen Satisfaction   
## Min. : 0.000 Length:100000   
## 1st Qu.: 1.000 Class :character   
## Median : 5.000 Mode :character   
## Mean : 6.843   
## 3rd Qu.:10.000   
## Max. :54.000

Some ‘character’ variables behave unexpectedly. Let’s convert all character variables into factors, which indicate categorical variables in R.

ithelp<-ithelp%>%  
 mutate(ticket=as.factor(ticket),  
 Requestor=as.factor(Requestor),  
 RequestorSeniority=as.factor(RequestorSeniority),  
 ITOwner=as.factor(ITOwner),  
 FiledAgainst=as.factor(FiledAgainst),  
 TicketType=as.factor(TicketType),  
 Severity=as.factor(Severity),  
 Priority=as.factor(Priority),  
 Satisfaction=as.factor(Satisfaction))

Let’s check the summary again.

summary(ithelp)

## ticket Requestor RequestorSeniority ITOwner   
## 1 : 1 285 : 74 1 - Junior :20040 3 : 2084   
## 2 : 1 754 : 74 2 - Regular :41303 39 : 2080   
## 3 : 1 79 : 73 3 - Senior :18801 48 : 2072   
## 4 : 1 636 : 72 4 - Management:19856 35 : 2058   
## 5 : 1 1341 : 72 24 : 2045   
## 6 : 1 523 : 70 4 : 2040   
## (Other):99994 (Other):99565 (Other):87621   
## FiledAgainst TicketType Severity   
## Access/Login:29921 Issue :24926 0 - Unclassified: 367   
## Hardware : 9976 Request:75074 1 - Minor : 2317   
## Software :20068 2 - Normal :90912   
## Systems :40035 3 - Major : 4974   
## 4 - Critical : 1430   
##   
##   
## Priority daysOpen Satisfaction   
## 0 - Unassigned:30127 Min. : 0.000 0 - Unknown :30211   
## 1 - Low :17117 1st Qu.: 1.000 1 - Unsatisfied :21124   
## 2 - Medium :16258 Median : 5.000 2 - Satisfied :19602   
## 3 - High :36498 Mean : 6.843 3 - Highly satisfied:29063   
## 3rd Qu.:10.000   
## Max. :54.000   
##

It turns out that there are many cases with “Unknown” Satisfaction. Let’s exclude these cases from the analysis (Step 1: filter). Next, we may build a multi-class classification model (Unsatisfied, Satisfied, Highly satisfied), but let’s simplify it to a binary classifier and identify “Unsatisfied” cases, which are problematic. Create a new variable, “negative”, which indicates if a user’s feedback is negative (Step 2: mutate). The following code will do these jobs for you.

ithelp<-ithelp%>%  
 filter(Satisfaction!="0 - Unknown")%>%  
 mutate(negative=as.factor(ifelse(Satisfaction =="1 - Unsatisfied","Yes","No")))  
  
summary(ithelp)

## ticket Requestor RequestorSeniority ITOwner   
## 1 : 1 442 : 56 1 - Junior :13995 39 : 1470   
## 2 : 1 1489 : 55 2 - Regular :28963 24 : 1442   
## 5 : 1 98 : 53 3 - Senior :13027 19 : 1440   
## 9 : 1 248 : 53 4 - Management:13804 3 : 1437   
## 10 : 1 1391 : 53 48 : 1437   
## 12 : 1 1498 : 53 31 : 1435   
## (Other):69783 (Other):69466 (Other):61128   
## FiledAgainst TicketType Severity   
## Access/Login:20990 Issue :17454 0 - Unclassified: 263   
## Hardware : 6912 Request:52335 1 - Minor : 1620   
## Software :13972 2 - Normal :63459   
## Systems :27915 3 - Major : 3445   
## 4 - Critical : 1002   
##   
##   
## Priority daysOpen Satisfaction   
## 0 - Unassigned:20984 Min. : 0.00 0 - Unknown : 0   
## 1 - Low :12045 1st Qu.: 1.00 1 - Unsatisfied :21124   
## 2 - Medium :11409 Median : 5.00 2 - Satisfied :19602   
## 3 - High :25351 Mean : 6.83 3 - Highly satisfied:29063   
## 3rd Qu.:10.00   
## Max. :49.00   
##   
## negative   
## No :48665   
## Yes:21124   
##   
##   
##   
##   
##

# Set up for holdout validation

Let’s select 20% of dataset. Using these indices, we will create a test and a training dataset.

set.seed(1) # set a random seed   
index <- sample(nrow(ithelp), nrow(ithelp)\*0.2) # random selection of indices.   
test <- ithelp[index,] # save 20% as a test dataset  
training <-ithelp[-index,] # save the rest as a training set

# Tree model

library(rpart)  
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.0.4

Build the same model to predict “negative”, with the following variables. \* RequestorSeniority \* FiledAgainst \* TicketType \* Severity \* Priority \* daysOpen

But this time, we will make two changes. 1. Instead of the entire dataset for training (ithelp), you will use the training dataset (training). 2. Generate a bigger tree with cp=0. Set control=rpart.control(cp=0).

Save the model as ct\_model. We will skip plotting the tree. It may crash.

ct\_model<-rpart(negative~RequestorSeniority+FiledAgainst+TicketType+Severity+Priority+daysOpen, # model formula  
 data=training, # dataset  
 method="class", # "class" indicates a classification tree model   
 control=rpart.control(cp=0)) # tree control parameters.

Check the cross-validation result using printcp().

printcp(ct\_model)

##   
## Classification tree:  
## rpart(formula = negative ~ RequestorSeniority + FiledAgainst +   
## TicketType + Severity + Priority + daysOpen, data = training,   
## method = "class", control = rpart.control(cp = 0))  
##   
## Variables actually used in tree construction:  
## [1] daysOpen FiledAgainst Priority RequestorSeniority  
## [5] Severity TicketType   
##   
## Root node error: 16889/55832 = 0.3025  
##   
## n= 55832   
##   
## CP nsplit rel error xerror xstd  
## 1 4.1842e-03 0 1.00000 1.00000 0.0064265  
## 2 2.0131e-03 3 0.98745 0.98745 0.0064033  
## 3 5.9210e-04 4 0.98543 0.98549 0.0063997  
## 4 5.3289e-04 7 0.98366 0.98792 0.0064042  
## 5 4.7368e-04 8 0.98313 0.98757 0.0064036  
## 6 4.1447e-04 10 0.98218 0.98857 0.0064054  
## 7 3.9473e-04 19 0.97750 0.98887 0.0064060  
## 8 3.8487e-04 22 0.97632 0.98780 0.0064040  
## 9 3.5526e-04 29 0.97341 0.98721 0.0064029  
## 10 3.2566e-04 31 0.97270 0.98626 0.0064011  
## 11 3.1579e-04 33 0.97205 0.98632 0.0064013  
## 12 2.9605e-04 41 0.96945 0.98822 0.0064048  
## 13 2.7631e-04 44 0.96856 0.98893 0.0064061  
## 14 2.6645e-04 47 0.96773 0.98893 0.0064061  
## 15 2.6052e-04 50 0.96690 0.98893 0.0064061  
## 16 2.3684e-04 56 0.96518 0.98816 0.0064047  
## 17 1.9243e-04 60 0.96424 0.98691 0.0064024  
## 18 1.8947e-04 64 0.96347 0.98958 0.0064073  
## 19 1.7763e-04 73 0.96151 0.99076 0.0064095  
## 20 1.5789e-04 84 0.95956 0.99082 0.0064096  
## 21 1.5395e-04 87 0.95909 0.99029 0.0064086  
## 22 1.4803e-04 92 0.95832 0.98952 0.0064072  
## 23 1.4210e-04 105 0.95630 0.99023 0.0064085  
## 24 1.3816e-04 110 0.95559 0.99023 0.0064085  
## 25 1.1842e-04 114 0.95500 0.98999 0.0064081  
## 26 9.8684e-05 140 0.95180 0.99005 0.0064082  
## 27 8.8815e-05 162 0.94949 0.99195 0.0064117  
## 28 8.4586e-05 189 0.94695 0.99319 0.0064140  
## 29 7.8947e-05 196 0.94636 0.99355 0.0064146  
## 30 7.1052e-05 214 0.94488 0.99455 0.0064165  
## 31 6.9078e-05 224 0.94416 0.99432 0.0064161  
## 32 5.9210e-05 230 0.94375 0.99603 0.0064192  
## 33 5.4276e-05 319 0.93801 0.99663 0.0064203  
## 34 4.9342e-05 334 0.93706 0.99816 0.0064231  
## 35 4.7368e-05 349 0.93605 0.99852 0.0064238  
## 36 4.4408e-05 354 0.93582 0.99858 0.0064239  
## 37 3.9473e-05 362 0.93546 1.00047 0.0064273  
## 38 3.5526e-05 368 0.93522 1.00101 0.0064283  
## 39 2.9605e-05 378 0.93487 1.00184 0.0064298  
## 40 1.9737e-05 408 0.93398 1.00308 0.0064320  
## 41 1.4803e-05 439 0.93333 1.00349 0.0064328  
## 42 1.3158e-05 443 0.93327 1.00426 0.0064342  
## 43 1.1842e-05 452 0.93315 1.00557 0.0064365  
## 44 9.8684e-06 457 0.93309 1.00580 0.0064370  
## 45 0.0000e+00 469 0.93297 1.00586 0.0064371

Prune the tree using the cp value with the minimum xerror. Save the result as min\_xerror\_tree.

# prune tree with minimum cp value  
min\_xerror\_tree <-  
 prune(ct\_model, cp = ct\_model$cptable[which.min(ct\_model$cptable[, "xerror"]), ][1])

Apply this model to the test dataset to get the predicted probabilities.

test$predicted\_Prob <- predict(min\_xerror\_tree,test)[,2]

Using the 50% cut-off, generate class prediction.

test$predicted\_Prob\_YN <- ifelse(test$predicted\_Prob>0.5,"Yes","No")

## Question 1. What is the error rate of this model when we use the 50% cut-off?

table(test$negative == test$predicted\_Prob\_YN)

##   
## FALSE TRUE   
## 4173 9784

nrow(test[test$negative != test$predicted\_Prob\_YN,]) / nrow(test)

## [1] 0.2989898

## Question 2. Generate a confusion table of this model. What is the false positive rate of this model?

table(test$predicted\_Prob\_YN,  
 test$negative,  
 dnn = c("predicted", "actual"))

## actual  
## predicted No Yes  
## No 9522 3973  
## Yes 200 262

nrow(test %>% filter(test$predicted\_Prob\_YN == "Yes", test$negative == "No")) / nrow(test %>% filter(test$negative == "No"))

## [1] 0.0205719

# Logit Regression Model

Using the training dataset, build a logit regression model to predict “negative”, with the following variables. \* RequestorSeniority \* FiledAgainst \* TicketType \* Severity \* Priority \* daysOpen

Again, be sure to use `training’ dataset for model building.

logit\_model <-  
 glm(  
 negative ~ RequestorSeniority + FiledAgainst + TicketType + Severity + Priority +  
 daysOpen,  
 # model formula  
 data = training,  
 family = "binomial"  
 )  
summary(logit\_model)

##   
## Call:  
## glm(formula = negative ~ RequestorSeniority + FiledAgainst +   
## TicketType + Severity + Priority + daysOpen, family = "binomial",   
## data = training)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.8316 -0.8300 -0.7416 1.3139 2.1780   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.181140 0.147158 -1.231 0.218352   
## RequestorSeniority2 - Regular 0.031778 0.026162 1.215 0.224500   
## RequestorSeniority3 - Senior -0.075648 0.032761 -2.309 0.020938 \*   
## RequestorSeniority4 - Management -0.011964 0.032933 -0.363 0.716404   
## FiledAgainstHardware -1.516031 0.050537 -29.999 < 2e-16 \*\*\*  
## FiledAgainstSoftware -0.481388 0.029355 -16.399 < 2e-16 \*\*\*  
## FiledAgainstSystems -0.774230 0.029369 -26.362 < 2e-16 \*\*\*  
## TicketTypeRequest -0.405046 0.023264 -17.411 < 2e-16 \*\*\*  
## Severity1 - Minor 0.168235 0.155224 1.084 0.278444   
## Severity2 - Normal -0.389170 0.144768 -2.688 0.007183 \*\*   
## Severity3 - Major -0.994141 0.151898 -6.545 5.96e-11 \*\*\*  
## Severity4 - Critical -0.932085 0.168127 -5.544 2.96e-08 \*\*\*  
## Priority1 - Low -0.100680 0.029684 -3.392 0.000694 \*\*\*  
## Priority2 - Medium 0.036698 0.028956 1.267 0.205022   
## Priority3 - High 0.078909 0.023990 3.289 0.001004 \*\*   
## daysOpen 0.083506 0.001913 43.640 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 68446 on 55831 degrees of freedom  
## Residual deviance: 66201 on 55816 degrees of freedom  
## AIC: 66233  
##   
## Number of Fisher Scoring iterations: 4

Apply this model to the test dataset to get the predicted probabilities.

test$logit\_pred\_prob<-predict(logit\_model,test,type="response")  
test$logit\_pred\_class<-ifelse(test$logit\_pred\_prob>0.5,"Yes","No")

# Performance Visualization with ROC

Plot ROC curves of the tree model and logit regression mode you developed.

library(pROC)

## Warning: package 'pROC' was built under R version 4.0.4

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

tree\_roc <- roc(test$negative, test$predicted\_Prob, auc=TRUE)

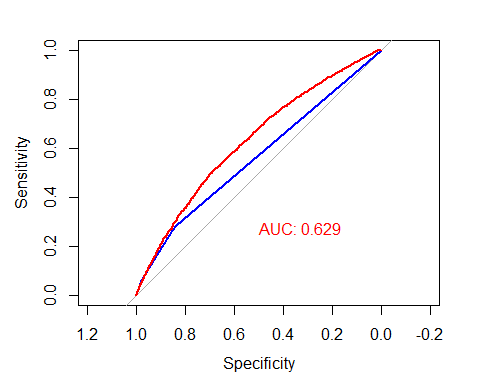
## Setting levels: control = No, case = Yes

## Setting direction: controls < cases

logit\_roc<-roc(test$negative,test$logit\_pred\_prob,auc=TRUE)

## Setting levels: control = No, case = Yes  
## Setting direction: controls < cases

plot(tree\_roc, pront.auc = TRUE,col="blue")  
plot(logit\_roc,print.auc=TRUE,print.auc.y=.3,add=TRUE,col="red")



## Question 3. What are AUCs of the classification tree model?

tree\_roc$auc

## Area under the curve: 0.564

## Question 4. What are AUCs of the logit regression model?

logit\_roc$auc

## Area under the curve: 0.6292

## Question 5

The same as the previous assignments, your last task is creating a report. Change the author name on the top of this R markdown file to yours. Compile this R markdown file into a Word document and submit it through the course Blackboard.