### Bank Customer Retention Course Project

*#install.packages(c("tidyverse","caret","randomForest","ranger","rpart","rattle","RColorBrewer","e1071","MASS","GGally","mice","VIM","ROCR"))*  
**library**(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──

## ✓ ggplot2 3.3.2 ✓ purrr 0.3.4  
## ✓ tibble 3.0.4 ✓ dplyr 1.0.2  
## ✓ tidyr 1.1.2 ✓ stringr 1.4.0  
## ✓ readr 1.4.0 ✓ forcats 0.5.0

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

**library**(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

**library**(randomForest)

## randomForest 4.6-14

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

##   
## Attaching package: 'randomForest'

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

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

**library**(ranger)

##   
## Attaching package: 'ranger'

## The following object is masked from 'package:randomForest':  
##   
## importance

**library**(rpart)  
**library**(rattle)

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.  
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

##   
## Attaching package: 'rattle'

## The following object is masked from 'package:ranger':  
##   
## importance

## The following object is masked from 'package:randomForest':  
##   
## importance

**library**(RColorBrewer)  
**library**(e1071)  
**library**(MASS)

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

**library**(GGally)

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

**library**(mice)

##   
## Attaching package: 'mice'

## The following objects are masked from 'package:base':  
##   
## cbind, rbind

**library**(VIM)

## Loading required package: colorspace

## Loading required package: grid

## VIM is ready to use.

## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues

##   
## Attaching package: 'VIM'

## The following object is masked from 'package:rattle':  
##   
## wine

## The following object is masked from 'package:datasets':  
##   
## sleep

**library**(ROCR)

Read in the data with the following code:

churn = **read\_csv**("Churn\_Modelling.csv")

##   
## ── Column specification ────────────────────────────────────────────────────────  
## cols(  
## RowNumber = col\_double(),  
## CustomerId = col\_double(),  
## Surname = col\_character(),  
## CreditScore = col\_double(),  
## Geography = col\_character(),  
## Gender = col\_character(),  
## Age = col\_double(),  
## Tenure = col\_double(),  
## Balance = col\_double(),  
## NumOfProducts = col\_double(),  
## HasCrCard = col\_double(),  
## IsActiveMember = col\_double(),  
## EstimatedSalary = col\_double(),  
## Exited = col\_double()  
## )

The variables in the dataset are:  
\* Row Number - Can be deleted  
\* CustomerId - Unique identifier for each customer  
\* Surname - Customer’s last name  
\* CreditScore - Customer’s credit score (higher is better)  
\* Geography - Customer’s country  
\* Gender - Customer’s gender  
\* Age - Customer’s age  
\* Tenure - Number of years that the customer has been with the bank  
\* Balance - Customer’s bank balance  
\* NumOfProducts - The number of bank products that the customer is using  
\* HasCrCard - Indicate whether the customer has (1) a credit card with the bank or (0) not  
\* IsActiveMember - Indicates the customer is (1) an active member of the bank or (0) not  
\* EstimatedSalary - The estimated salary of the customer  
\* Exited - An indicator that shows if the customer closed their account with the bank (1) or (0) not (This is the response variable)

Run your code from Phase 1 to clean and prepare the data. This code should complete the following tasks:

* Delete the RowNumber, CustomerId, and Surname columns (NOTE: The MASS package has been loaded so you will need to be careful if you choose to use the select function)

churn = churn **%>%** **drop\_na**()  
  
churn = churn **%>%** dplyr**::select**(**c**("CreditScore", "Geography", "Gender", "Age", "Balance", "Tenure", "NumOfProducts", "HasCrCard","IsActiveMember", "EstimatedSalary", "Exited"))  
  
**str**(churn)

## tibble [10,000 × 11] (S3: tbl\_df/tbl/data.frame)  
## $ CreditScore : num [1:10000] 619 608 502 699 850 645 822 376 501 684 ...  
## $ Geography : chr [1:10000] "France" "Spain" "France" "France" ...  
## $ Gender : chr [1:10000] "Female" "Female" "Female" "Female" ...  
## $ Age : num [1:10000] 42 41 42 39 43 44 50 29 44 27 ...  
## $ Balance : num [1:10000] 0 83808 159661 0 125511 ...  
## $ Tenure : num [1:10000] 2 1 8 1 2 8 7 4 4 2 ...  
## $ NumOfProducts : num [1:10000] 1 1 3 2 1 2 2 4 2 1 ...  
## $ HasCrCard : num [1:10000] 1 0 1 0 1 1 1 1 0 1 ...  
## $ IsActiveMember : num [1:10000] 1 1 0 0 1 0 1 0 1 1 ...  
## $ EstimatedSalary: num [1:10000] 101349 112543 113932 93827 79084 ...  
## $ Exited : num [1:10000] 1 0 1 0 0 1 0 1 0 0 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. RowNumber = col\_double(),  
## .. CustomerId = col\_double(),  
## .. Surname = col\_character(),  
## .. CreditScore = col\_double(),  
## .. Geography = col\_character(),  
## .. Gender = col\_character(),  
## .. Age = col\_double(),  
## .. Tenure = col\_double(),  
## .. Balance = col\_double(),  
## .. NumOfProducts = col\_double(),  
## .. HasCrCard = col\_double(),  
## .. IsActiveMember = col\_double(),  
## .. EstimatedSalary = col\_double(),  
## .. Exited = col\_double()  
## .. )

* Convert the Geography, Gender, HasCrCard, IsActiveMember, and Exited columns to factors

churn = churn **%>%** **mutate\_at**(**c**("Geography","Gender","HasCrCard","IsActiveMember","Exited"),as\_factor)  
  
**str**(churn)

## tibble [10,000 × 11] (S3: tbl\_df/tbl/data.frame)  
## $ CreditScore : num [1:10000] 619 608 502 699 850 645 822 376 501 684 ...  
## $ Geography : Factor w/ 3 levels "France","Spain",..: 1 2 1 1 2 2 1 3 1 1 ...  
## $ Gender : Factor w/ 2 levels "Female","Male": 1 1 1 1 1 2 2 1 2 2 ...  
## $ Age : num [1:10000] 42 41 42 39 43 44 50 29 44 27 ...  
## $ Balance : num [1:10000] 0 83808 159661 0 125511 ...  
## $ Tenure : num [1:10000] 2 1 8 1 2 8 7 4 4 2 ...  
## $ NumOfProducts : num [1:10000] 1 1 3 2 1 2 2 4 2 1 ...  
## $ HasCrCard : Factor w/ 2 levels "0","1": 2 1 2 1 2 2 2 2 1 2 ...  
## $ IsActiveMember : Factor w/ 2 levels "0","1": 2 2 1 1 2 1 2 1 2 2 ...  
## $ EstimatedSalary: num [1:10000] 101349 112543 113932 93827 79084 ...  
## $ Exited : Factor w/ 2 levels "0","1": 2 1 2 1 1 2 1 2 1 1 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. RowNumber = col\_double(),  
## .. CustomerId = col\_double(),  
## .. Surname = col\_character(),  
## .. CreditScore = col\_double(),  
## .. Geography = col\_character(),  
## .. Gender = col\_character(),  
## .. Age = col\_double(),  
## .. Tenure = col\_double(),  
## .. Balance = col\_double(),  
## .. NumOfProducts = col\_double(),  
## .. HasCrCard = col\_double(),  
## .. IsActiveMember = col\_double(),  
## .. EstimatedSalary = col\_double(),  
## .. Exited = col\_double()  
## .. )

* Rename the levels of the HasCrCard, IsActiveMember, and Exited columns from 0 and 1 to No and Yes (You may use more descriptive namings if you prefer)

churn = churn **%>%** **mutate** (HasCrCard = **as.factor**(HasCrCard))**%>%**  
 **mutate**(HasCrCard = **fct\_recode**(HasCrCard, "No"= "0", "Yes" = "1"))  
  
churn = churn **%>%** **mutate** (IsActiveMember = **as.factor**(IsActiveMember)) **%>%**  
 **mutate**(IsActiveMember = **fct\_recode**(IsActiveMember, "No"= "0", "Yes" = "1"))  
   
churn = churn **%>%** **mutate** (Exited = **as.factor**(Exited)) **%>%**  
 **mutate**(Exited = **fct\_recode**(Exited, "No" = "0", "Yes" = "1"))  
  
**summary**(churn)

## CreditScore Geography Gender Age Balance   
## Min. :350.0 France :5014 Female:4543 Min. :18.00 Min. : 0   
## 1st Qu.:584.0 Spain :2477 Male :5457 1st Qu.:32.00 1st Qu.: 0   
## Median :652.0 Germany:2509 Median :37.00 Median : 97199   
## Mean :650.5 Mean :38.92 Mean : 76486   
## 3rd Qu.:718.0 3rd Qu.:44.00 3rd Qu.:127644   
## Max. :850.0 Max. :92.00 Max. :250898   
## Tenure NumOfProducts HasCrCard IsActiveMember EstimatedSalary   
## Min. : 0.000 Min. :1.00 No :2945 No :4849 Min. : 11.58   
## 1st Qu.: 3.000 1st Qu.:1.00 Yes:7055 Yes:5151 1st Qu.: 51002.11   
## Median : 5.000 Median :1.00 Median :100193.91   
## Mean : 5.013 Mean :1.53 Mean :100090.24   
## 3rd Qu.: 7.000 3rd Qu.:2.00 3rd Qu.:149388.25   
## Max. :10.000 Max. :4.00 Max. :199992.48   
## Exited   
## No :7963   
## Yes:2037   
##   
##   
##   
##

* If there is any missing data, deal with this missingness is an appropriate manner
* You may also remove any data that you consider to be an outlier
* Split the data into training and testing sets (Use a 70/30 training/testing split with a random number seed of 1234)

**set.seed**(1234)  
train.rows = **createDataPartition**(y = churn**$**Exited, p=0.7, list = FALSE)  
train = **slice**(churn, train.rows)  
test = **slice**(churn, **-**train.rows)

For Phase 2 work, complete the following tasks:

* Compute the accuracy of a naive model on the training set

fit1 = **rpart**(Exited **~** CreditScore **+** Geography **+** Gender **+** Age **+** Balance **+** Tenure **+** NumOfProducts **+** HasCrCard **+** IsActiveMember **+** EstimatedSalary, data = train, method = "class")  
naivepred = **predict**(fit1, train, type = "class")  
**confusionMatrix**(naivepred, train**$**Exited)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 5446 852  
## Yes 129 574  
##   
## Accuracy : 0.8599   
## 95% CI : (0.8515, 0.8679)  
## No Information Rate : 0.7963   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4676   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9769   
## Specificity : 0.4025   
## Pos Pred Value : 0.8647   
## Neg Pred Value : 0.8165   
## Prevalence : 0.7963   
## Detection Rate : 0.7779   
## Detection Prevalence : 0.8996   
## Balanced Accuracy : 0.6897   
##   
## 'Positive' Class : No   
##

* Build the best logistic regression model that you can (including selection of an appropriate threshold)

**options**(scipen=999)  
   
allmod = **glm**(Exited **~**., train, family = "binomial")  
**summary**(allmod)

##   
## Call:  
## glm(formula = Exited ~ ., family = "binomial", data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.2950 -0.6604 -0.4605 -0.2759 2.8873   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.5797241423 0.2918268870 -12.267 < 0.0000000000000002 \*\*\*  
## CreditScore -0.0002540238 0.0003337158 -0.761 0.446539   
## GeographySpain -0.0261122244 0.0841900180 -0.310 0.756441   
## GeographyGermany 0.7266682538 0.0805863794 9.017 < 0.0000000000000002 \*\*\*  
## GenderMale -0.5032013437 0.0648807735 -7.756 0.00000000000000878 \*\*\*  
## Age 0.0726950585 0.0030753180 23.638 < 0.0000000000000002 \*\*\*  
## Balance 0.0000022951 0.0000006109 3.757 0.000172 \*\*\*  
## Tenure -0.0085468238 0.0111697429 -0.765 0.444166   
## NumOfProducts -0.1139994947 0.0566036052 -2.014 0.044010 \*   
## HasCrCardYes -0.1593351033 0.0697686272 -2.284 0.022385 \*   
## IsActiveMemberYes -1.0555711487 0.0686873659 -15.368 < 0.0000000000000002 \*\*\*  
## EstimatedSalary 0.0000006854 0.0000005664 1.210 0.226243   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 7077.6 on 7000 degrees of freedom  
## Residual deviance: 6033.0 on 6989 degrees of freedom  
## AIC: 6057  
##   
## Number of Fisher Scoring iterations: 5

emptymod = **glm**(Exited **~**1, train, family = "binomial")  
**summary**(emptymod)

##   
## Call:  
## glm(formula = Exited ~ 1, family = "binomial", data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.6749 -0.6749 -0.6749 -0.6749 1.7839   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.36342 0.02968 -45.94 <0.0000000000000002 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 7077.6 on 7000 degrees of freedom  
## Residual deviance: 7077.6 on 7000 degrees of freedom  
## AIC: 7079.6  
##   
## Number of Fisher Scoring iterations: 4

backmod = **stepAIC**(allmod, direction = "backward", trace = TRUE)

## Start: AIC=6057.03  
## Exited ~ CreditScore + Geography + Gender + Age + Balance + Tenure +   
## NumOfProducts + HasCrCard + IsActiveMember + EstimatedSalary  
##   
## Df Deviance AIC  
## - CreditScore 1 6033.6 6055.6  
## - Tenure 1 6033.6 6055.6  
## - EstimatedSalary 1 6034.5 6056.5  
## <none> 6033.0 6057.0  
## - NumOfProducts 1 6037.1 6059.1  
## - HasCrCard 1 6038.2 6060.2  
## - Balance 1 6047.1 6069.1  
## - Gender 1 6093.6 6115.6  
## - Geography 2 6127.5 6147.5  
## - IsActiveMember 1 6285.9 6307.9  
## - Age 1 6640.1 6662.1  
##   
## Step: AIC=6055.61  
## Exited ~ Geography + Gender + Age + Balance + Tenure + NumOfProducts +   
## HasCrCard + IsActiveMember + EstimatedSalary  
##   
## Df Deviance AIC  
## - Tenure 1 6034.2 6054.2  
## - EstimatedSalary 1 6035.1 6055.1  
## <none> 6033.6 6055.6  
## - NumOfProducts 1 6037.8 6057.8  
## - HasCrCard 1 6038.8 6058.8  
## - Balance 1 6047.7 6067.7  
## - Gender 1 6094.1 6114.1  
## - Geography 2 6128.2 6146.2  
## - IsActiveMember 1 6287.6 6307.6  
## - Age 1 6641.3 6661.3  
##   
## Step: AIC=6054.19  
## Exited ~ Geography + Gender + Age + Balance + NumOfProducts +   
## HasCrCard + IsActiveMember + EstimatedSalary  
##   
## Df Deviance AIC  
## - EstimatedSalary 1 6035.6 6053.6  
## <none> 6034.2 6054.2  
## - NumOfProducts 1 6038.3 6056.3  
## - HasCrCard 1 6039.5 6057.5  
## - Balance 1 6048.3 6066.3  
## - Gender 1 6094.9 6112.9  
## - Geography 2 6128.6 6144.6  
## - IsActiveMember 1 6287.6 6305.6  
## - Age 1 6642.0 6660.0  
##   
## Step: AIC=6053.65  
## Exited ~ Geography + Gender + Age + Balance + NumOfProducts +   
## HasCrCard + IsActiveMember  
##   
## Df Deviance AIC  
## <none> 6035.6 6053.6  
## - NumOfProducts 1 6039.7 6055.7  
## - HasCrCard 1 6040.9 6056.9  
## - Balance 1 6049.8 6065.8  
## - Gender 1 6096.5 6112.5  
## - Geography 2 6130.3 6144.3  
## - IsActiveMember 1 6289.7 6305.7  
## - Age 1 6643.0 6659.0

**summary**(backmod)

##   
## Call:  
## glm(formula = Exited ~ Geography + Gender + Age + Balance + NumOfProducts +   
## HasCrCard + IsActiveMember, family = "binomial", data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.3284 -0.6605 -0.4601 -0.2756 2.8962   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -3.7192137238 0.1811114401 -20.535 < 0.0000000000000002 \*\*\*  
## GeographySpain -0.0258825877 0.0841372530 -0.308 0.758369   
## GeographyGermany 0.7270423522 0.0805425509 9.027 < 0.0000000000000002 \*\*\*  
## GenderMale -0.5039179227 0.0648529181 -7.770 0.00000000000000784 \*\*\*  
## Age 0.0726784235 0.0030741370 23.642 < 0.0000000000000002 \*\*\*  
## Balance 0.0000022996 0.0000006107 3.765 0.000166 \*\*\*  
## NumOfProducts -0.1127566049 0.0565542048 -1.994 0.046176 \*   
## HasCrCardYes -0.1600027205 0.0697142089 -2.295 0.021726 \*   
## IsActiveMemberYes -1.0563105130 0.0685793173 -15.403 < 0.0000000000000002 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 7077.6 on 7000 degrees of freedom  
## Residual deviance: 6035.6 on 6992 degrees of freedom  
## AIC: 6053.6  
##   
## Number of Fisher Scoring iterations: 5

forwardmod = **stepAIC**(emptymod, direction = "forward", scope=**list**(upper=allmod,lower=emptymod),  
 trace = TRUE)

## Start: AIC=7079.58  
## Exited ~ 1  
##   
## Df Deviance AIC  
## + Age 1 6551.9 6555.9  
## + Geography 2 6907.6 6913.6  
## + IsActiveMember 1 6912.2 6916.2  
## + Balance 1 6992.0 6996.0  
## + Gender 1 7001.5 7005.5  
## + NumOfProducts 1 7061.4 7065.4  
## + HasCrCard 1 7072.8 7076.8  
## + CreditScore 1 7075.5 7079.5  
## <none> 7077.6 7079.6  
## + EstimatedSalary 1 7075.9 7079.9  
## + Tenure 1 7077.4 7081.4  
##   
## Step: AIC=6555.9  
## Exited ~ Age  
##   
## Df Deviance AIC  
## + IsActiveMember 1 6282.2 6288.2  
## + Geography 2 6383.7 6391.7  
## + Balance 1 6465.8 6471.8  
## + Gender 1 6483.9 6489.9  
## + NumOfProducts 1 6538.6 6544.6  
## + HasCrCard 1 6548.5 6554.5  
## + EstimatedSalary 1 6549.4 6555.4  
## <none> 6551.9 6555.9  
## + CreditScore 1 6550.0 6556.0  
## + Tenure 1 6551.9 6557.9  
##   
## Step: AIC=6288.19  
## Exited ~ Age + IsActiveMember  
##   
## Df Deviance AIC  
## + Geography 2 6123.6 6133.6  
## + Balance 1 6202.9 6210.9  
## + Gender 1 6218.2 6226.2  
## + NumOfProducts 1 6271.7 6279.7  
## + HasCrCard 1 6277.0 6285.0  
## <none> 6282.2 6288.2  
## + EstimatedSalary 1 6280.4 6288.4  
## + Tenure 1 6281.5 6289.5  
## + CreditScore 1 6281.5 6289.5  
##   
## Step: AIC=6133.59  
## Exited ~ Age + IsActiveMember + Geography  
##   
## Df Deviance AIC  
## + Gender 1 6065.5 6077.5  
## + Balance 1 6105.5 6117.5  
## + NumOfProducts 1 6114.7 6126.7  
## + HasCrCard 1 6117.7 6129.7  
## <none> 6123.6 6133.6  
## + EstimatedSalary 1 6122.2 6134.2  
## + Tenure 1 6122.6 6134.6  
## + CreditScore 1 6123.0 6135.0  
##   
## Step: AIC=6077.5  
## Exited ~ Age + IsActiveMember + Geography + Gender  
##   
## Df Deviance AIC  
## + Balance 1 6044.9 6058.9  
## + NumOfProducts 1 6055.2 6069.2  
## + HasCrCard 1 6059.9 6073.9  
## <none> 6065.5 6077.5  
## + EstimatedSalary 1 6064.2 6078.2  
## + Tenure 1 6064.8 6078.8  
## + CreditScore 1 6064.9 6078.9  
##   
## Step: AIC=6058.94  
## Exited ~ Age + IsActiveMember + Geography + Gender + Balance  
##   
## Df Deviance AIC  
## + HasCrCard 1 6039.7 6055.7  
## + NumOfProducts 1 6040.9 6056.9  
## <none> 6044.9 6058.9  
## + EstimatedSalary 1 6043.7 6059.7  
## + Tenure 1 6044.3 6060.3  
## + CreditScore 1 6044.3 6060.3  
##   
## Step: AIC=6055.65  
## Exited ~ Age + IsActiveMember + Geography + Gender + Balance +   
## HasCrCard  
##   
## Df Deviance AIC  
## + NumOfProducts 1 6035.6 6053.6  
## <none> 6039.7 6055.7  
## + EstimatedSalary 1 6038.3 6056.3  
## + CreditScore 1 6039.0 6057.0  
## + Tenure 1 6039.1 6057.1  
##   
## Step: AIC=6053.65  
## Exited ~ Age + IsActiveMember + Geography + Gender + Balance +   
## HasCrCard + NumOfProducts  
##   
## Df Deviance AIC  
## <none> 6035.6 6053.6  
## + EstimatedSalary 1 6034.2 6054.2  
## + CreditScore 1 6035.1 6055.1  
## + Tenure 1 6035.1 6055.1

*# Training Data Prediction*  
predicttrain = **predict**(forwardmod, type = "response")  
t1 = **table**(train**$**Exited, predicttrain **>** 0.5)  
t1

##   
## FALSE TRUE  
## No 5383 192  
## Yes 1136 290

(t1[1,1]**+**t1[2,2])**/nrow**(train)

## [1] 0.8103128

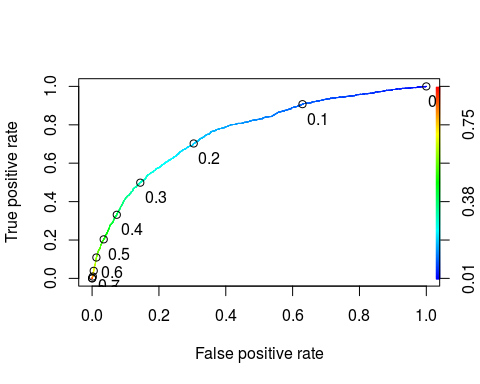
*#Naive Prediction*  
t1 = **table**(train**$**Exited, predicttrain **>** 1)  
t1

##   
## FALSE  
## No 5575  
## Yes 1426

(t1[1])**/nrow**(train)

## [1] 0.7963148

ROCRpred = **prediction**(predicttrain, train**$**Exited)   
  
ROCRperf = **performance**(ROCRpred, "tpr", "fpr")  
**plot**(ROCRperf, colorize=TRUE, print.cutoffs.at=**seq**(0,1,by=0.1), text.adj=**c**(**-**0.2,1.7))



opt.cut = **function**(perf, pred){  
 cut.ind = **mapply**(FUN=**function**(x, y, p){  
 d = (x **-** 0)**^**2 **+** (y-1)**^**2  
 ind = **which**(d **==** **min**(d))  
 **c**(sensitivity = y[[ind]], specificity = 1**-**x[[ind]],   
 cutoff = p[[ind]])  
 }, perf**@**x.values, perf**@**y.values, pred**@**cutoffs)  
}  
**print**(**opt.cut**(ROCRperf, ROCRpred))

## [,1]  
## sensitivity 0.7258065  
## specificity 0.6783857  
## cutoff 0.1924105

*# Training set performance*  
t1 = **table**(train**$**Exited, predicttrain **>** 0.1924105)   
t1

##   
## FALSE TRUE  
## No 3782 1793  
## Yes 392 1034

(t1[1,1]**+**t1[2,2])**/nrow**(train)

## [1] 0.6879017

*# Training set Sensitivity*   
1034**/**(1034**+**392)

## [1] 0.7251052

*# Training set Specificity*  
3782**/**(3782**+**1793)

## [1] 0.6783857

*# Testing set Performance*  
predicttest = **predict**(forwardmod, type="response", test)  
t2 = **table**(test**$**Exited, predicttest **>** 0.1924105)  
t2

##   
## FALSE TRUE  
## No 1642 746  
## Yes 174 437

*# Testing set Accuracy*  
(t2[1,1]**+**t2[2,2])**/**(**nrow**(test))

## [1] 0.6932311

*# Testing set Sensitivity*   
437**/**(437**+**174)

## [1] 0.7152209

*# Test set Specificity*   
1642**/**(1642**+**746)

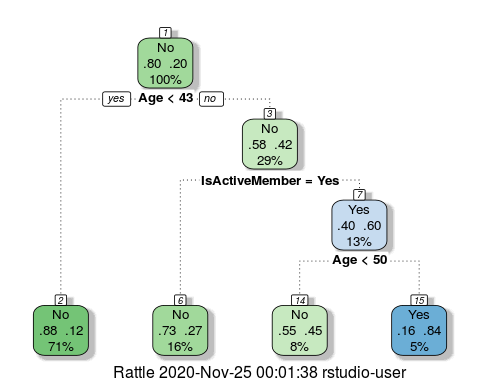
## [1] 0.6876047

* Build the best classification tree that you can

**set.seed**(999)  
ctrl <- **trainControl**(method = "cv",  
 number = 10)  
  
fit1 <- **train**(x=**as.data.frame**(train[,**-**11]), y=train**$**Exited,   
 method = "rpart",  
 trControl = ctrl)  
fit1

## CART   
##   
## 7001 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6300, 6300, 6300, 6302, 6302, 6300, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.03436185 0.8265973 0.2965793  
## 0.03856942 0.8237404 0.3124868  
## 0.06136045 0.8050330 0.1645371  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.03436185.

**fancyRpartPlot**(fit1**$**finalModel)



*# Classification Tree Training Set Predictions*  
predtrain2 = **predict**(fit1, train, type = "raw")  
  
**confusionMatrix**(predtrain2, train**$**Exited)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 5520 1141  
## Yes 55 285  
##   
## Accuracy : 0.8292   
## 95% CI : (0.8201, 0.8379)   
## No Information Rate : 0.7963   
## P-Value [Acc > NIR] : 0.000000000001763   
##   
## Kappa : 0.2651   
##   
## Mcnemar's Test P-Value : < 0.00000000000000022  
##   
## Sensitivity : 0.9901   
## Specificity : 0.1999   
## Pos Pred Value : 0.8287   
## Neg Pred Value : 0.8382   
## Prevalence : 0.7963   
## Detection Rate : 0.7885   
## Detection Prevalence : 0.9514   
## Balanced Accuracy : 0.5950   
##   
## 'Positive' Class : No   
##

*# Classification Tree Testing Set Predictions*  
predtest2 = **predict**(fit1, test, type = "raw")  
  
**confusionMatrix**(predtest2, test**$**Exited)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2357 498  
## Yes 31 113  
##   
## Accuracy : 0.8236   
## 95% CI : (0.8095, 0.8371)   
## No Information Rate : 0.7963   
## P-Value [Acc > NIR] : 0.00008645   
##   
## Kappa : 0.2403   
##   
## Mcnemar's Test P-Value : < 0.00000000000000022  
##   
## Sensitivity : 0.9870   
## Specificity : 0.1849   
## Pos Pred Value : 0.8256   
## Neg Pred Value : 0.7847   
## Prevalence : 0.7963   
## Detection Rate : 0.7859   
## Detection Prevalence : 0.9520   
## Balanced Accuracy : 0.5860   
##   
## 'Positive' Class : No   
##

* Build the best random forest that you can

**set.seed**(999)  
ctrl = **trainControl**(method = "cv",  
 number = 5) *#5 fold, k-fold cross-validation*  
  
tree1 = **train**(x=**as.data.frame**(train[,**-**11]),y=train**$**Exited,  
 method = "rpart",  
 tuneLength = 10,   
 trControl = ctrl)  
  
tree1

## CART   
##   
## 7001 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 5600, 5601, 5601, 5601, 5601   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.003038803 0.8571629 0.48411595  
## 0.004207574 0.8578773 0.48193907  
## 0.005610098 0.8583057 0.47790538  
## 0.018583450 0.8488790 0.43611519  
## 0.020336606 0.8475942 0.43012559  
## 0.026647966 0.8420240 0.39039862  
## 0.032258065 0.8350240 0.36309824  
## 0.034361851 0.8267382 0.32591741  
## 0.038569425 0.8237387 0.31759794  
## 0.061360449 0.8003149 0.06917791  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.005610098.

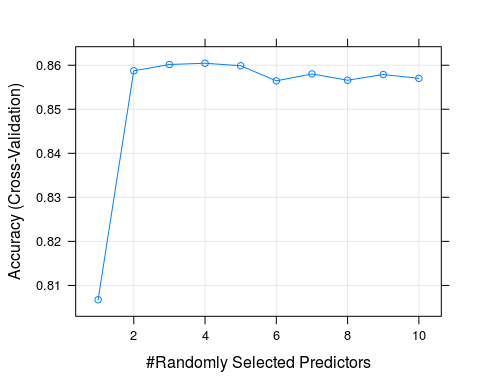
*#only parameter we can tune in the randomForest package is mtry*  
*#tgrid = expand.grid(*  
 *# mtry = 1:10 )*  
  
*#set.seed(999)*  
*#rf1 = train(x=as.data.frame(train[,-11]),y=train$Exited,*   
 *# method = "rf",*   
 *# tuneGrid = tgrid,*  
 *# trControl = ctrl) #same cross-validation set-up from before*

*#saveRDS(rf1, "rf1.rds")*

rf1 = **readRDS**("rf1.rds")  
**print**(rf1)

## Random Forest   
##   
## 7001 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 5600, 5601, 5601, 5601, 5601   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 1 0.8067418 0.07980577  
## 2 0.8587348 0.45965824  
## 3 0.8601628 0.49176151  
## 4 0.8604485 0.50185369  
## 5 0.8598765 0.50414538  
## 6 0.8564485 0.49462219  
## 7 0.8580199 0.49975642  
## 8 0.8565914 0.49616367  
## 9 0.8578770 0.50061776  
## 10 0.8570202 0.50058457  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 4.

**plot**(rf1)



**varImp**(rf1)

## rf variable importance  
##   
## Overall  
## Age 100.0000  
## Balance 57.6306  
## EstimatedSalary 57.3416  
## CreditScore 54.2132  
## NumOfProducts 48.9178  
## Tenure 26.4865  
## IsActiveMember 12.8548  
## Geography 9.6957  
## Gender 0.6093  
## HasCrCard 0.0000

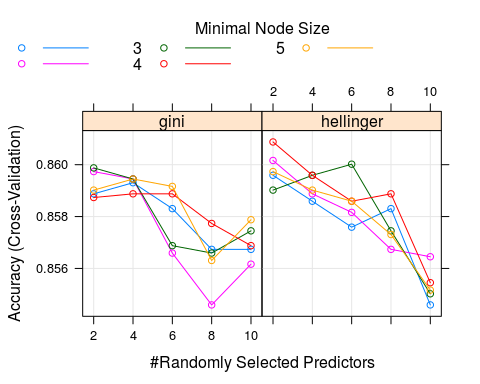
*#tgrid = expand.grid(*  
 *# mtry = c(2,4,6,8,10),*   
 *# splitrule = c("gini", "hellinger"),*  
 *# min.node.size = 1:5)*  
  
  
*#rf2 = train(x=as.data.frame(train[,-11]),y=train$Exited,*   
 *# method = "ranger",*   
 *# tuneGrid = tgrid,*  
 *# importance = "permutation",*   
 *# trControl = ctrl)*

*#saveRDS(rf2, "rf2.rds")*

rf2 = **readRDS**("rf2.rds")  
**print**(rf2)

## Random Forest   
##   
## 7001 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 5601, 5601, 5601, 5601, 5600   
## Resampling results across tuning parameters:  
##   
## mtry splitrule min.node.size Accuracy Kappa   
## 2 gini 1 0.8588768 0.4662702  
## 2 gini 2 0.8597341 0.4699154  
## 2 gini 3 0.8598768 0.4716304  
## 2 gini 4 0.8587341 0.4666561  
## 2 gini 5 0.8590196 0.4676944  
## 2 hellinger 1 0.8595910 0.4648994  
## 2 hellinger 2 0.8601625 0.4681978  
## 2 hellinger 3 0.8590200 0.4631725  
## 2 hellinger 4 0.8608764 0.4684483  
## 2 hellinger 5 0.8597337 0.4658182  
## 4 gini 1 0.8593048 0.4975524  
## 4 gini 2 0.8594481 0.4990659  
## 4 gini 3 0.8594477 0.4968865  
## 4 gini 4 0.8588762 0.4948506  
## 4 gini 5 0.8594473 0.4981952  
## 4 hellinger 1 0.8585909 0.4933504  
## 4 hellinger 2 0.8588763 0.4940331  
## 4 hellinger 3 0.8595904 0.4973836  
## 4 hellinger 4 0.8595908 0.4955099  
## 4 hellinger 5 0.8590196 0.4946561  
## 6 gini 1 0.8583051 0.4988463  
## 6 gini 2 0.8565911 0.4933659  
## 6 gini 3 0.8568769 0.4934157  
## 6 gini 4 0.8588766 0.5007910  
## 6 gini 5 0.8591623 0.5021361  
## 6 hellinger 1 0.8575911 0.4932195  
## 6 hellinger 2 0.8581621 0.4950329  
## 6 hellinger 3 0.8600193 0.5021250  
## 6 hellinger 4 0.8585910 0.4981951  
## 6 hellinger 5 0.8585914 0.4957934  
## 8 gini 1 0.8567346 0.4946543  
## 8 gini 2 0.8545919 0.4889231  
## 8 gini 3 0.8565918 0.4942351  
## 8 gini 4 0.8577338 0.4970535  
## 8 gini 5 0.8563055 0.4923718  
## 8 hellinger 1 0.8583055 0.4994462  
## 8 hellinger 2 0.8567345 0.4945179  
## 8 hellinger 3 0.8574487 0.4963538  
## 8 hellinger 4 0.8588771 0.4992591  
## 8 hellinger 5 0.8573063 0.4935846  
## 10 gini 1 0.8567348 0.4982121  
## 10 gini 2 0.8561635 0.4955742  
## 10 gini 3 0.8574488 0.5002887  
## 10 gini 4 0.8568775 0.4982216  
## 10 gini 5 0.8578771 0.5007182  
## 10 hellinger 1 0.8545921 0.4872025  
## 10 hellinger 2 0.8564492 0.4938266  
## 10 hellinger 3 0.8550206 0.4896782  
## 10 hellinger 4 0.8554491 0.4904560  
## 10 hellinger 5 0.8551634 0.4888441  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 2, splitrule = hellinger  
## and min.node.size = 4.

**plot**(rf2)



*#tgrid = expand.grid(*  
 *# mtry = 2,*   
 *# splitrule = "hellinger",*  
 *# min.node.size = 4)*  
  
*#rf3 = train(x=as.data.frame(train[,-11]),y=train$Exited,*   
 *# method = "ranger",*   
 *# max.depth = 5,*   
 *# tuneGrid = tgrid,*  
 *# importance = "permutation",*   
 *# trControl = ctrl)*

*#saveRDS(rf3, "rf3.rds")*

rf3 = **readRDS**("rf3.rds")  
**print**(rf3)

## Random Forest   
##   
## 7001 samples  
## 10 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 5601, 5601, 5601, 5601, 5600   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.831739 0.2641863  
##   
## Tuning parameter 'mtry' was held constant at a value of 2  
## Tuning  
## parameter 'splitrule' was held constant at a value of hellinger  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 4

*# Training set Predictions*  
trainpredrf3 = **predict**(rf3, type = "raw")   
**confusionMatrix**(trainpredrf3,train**$**Exited,  
 positive = "No")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 5560 1132  
## Yes 15 294  
##   
## Accuracy : 0.8362   
## 95% CI : (0.8273, 0.8448)   
## No Information Rate : 0.7963   
## P-Value [Acc > NIR] : < 0.00000000000000022  
##   
## Kappa : 0.2872   
##   
## Mcnemar's Test P-Value : < 0.00000000000000022  
##   
## Sensitivity : 0.9973   
## Specificity : 0.2062   
## Pos Pred Value : 0.8308   
## Neg Pred Value : 0.9515   
## Prevalence : 0.7963   
## Detection Rate : 0.7942   
## Detection Prevalence : 0.9559   
## Balanced Accuracy : 0.6017   
##   
## 'Positive' Class : No   
##

*# Testing set Predictions*  
testpredrf3 = **predict**(rf3, newdata = test, type="raw")  
**confusionMatrix**(testpredrf3,test**$**Exited,  
 positive = "No")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2372 477  
## Yes 16 134  
##   
## Accuracy : 0.8356   
## 95% CI : (0.8219, 0.8487)   
## No Information Rate : 0.7963   
## P-Value [Acc > NIR] : 0.00000002347   
##   
## Kappa : 0.2956   
##   
## Mcnemar's Test P-Value : < 0.00000000000000022  
##   
## Sensitivity : 0.9933   
## Specificity : 0.2193   
## Pos Pred Value : 0.8326   
## Neg Pred Value : 0.8933   
## Prevalence : 0.7963   
## Detection Rate : 0.7909   
## Detection Prevalence : 0.9500   
## Balanced Accuracy : 0.6063   
##   
## 'Positive' Class : No   
##

* Provide a summary of your three models’ performance versus the naive, on the training set, and on the testing set

**Naive Model**

The naive model produced 85.99% accuracy when used on the training set with a No Information Rate of 79.63%. The model produced .9769 Sensitivity and .4025 Specificity.

**Logistic Model**

The backmod and forwardmod produced an AIC of 6053.6 after removing several variables it deemed irrelevant. Age, IsActiveMember, Geography, Gender, Balance, HasCrCard, and NumOfProducts were determined to be relevant while EstimatedSalary, CreditScore and Tenure were removed. The three removed variables all had P-values less than 0.05.

The ROCR model determined .1924 to be the optimum threshold with a .7259 Sensitivity and .6784 Specificity. When the threshold was used on the training set, 68.79% accuracy was achieved with .7252 Sensitivity and .6784 Specificity.

The optimum threshold achieved 69.32% accuracy on the testing set with .7152 Sensitivity and .6876 Specificity

**Classification Tree Model**

This model produced a tree with 82.66% naive accuracy using a cp of .03436.

When used on the training set, 82.92% accuracy was achieved with a Sensitivity of .9901 and Specificity of .1999.

The model produced 82.36% accuracy on the testing set with a Sensitivity of .9870 and Specificity of .1849.

This model determined Age and IsActiveMember to be the 2 most important variables.

**Random Forest Model**

This model produced 83.17% naive accuracy. When used on the training set, 83.62% accuracy was achieved with .9973 Sensitivity and .2062 Specificity.

The testing set yielded 83.56% accuracy with .9933 Sensitivity and .2193 Specificity.

This model determined Age, NumOfProducts, and IsActiveMember to be the 3 most relevant variables.

* Discuss practical implications of your models including a discussion on accuracy versus sensitivity

The sensitivity is defined as the proportion of positive results out of the number of samples which were actually positive. When there are no positive results, sensitivity is not defined and a value of “NA” is returned. Similarly, when there are no negative results, specificity is not defined and a value of NA is returned.

While sensitivity is the proportion of true positives that are correctly identified, Accuracy is the proportion of true results, either true positive or true negative.

When determining the causes of customer exit, it is better to use the Accuracy metric rather than the Sensitivity metric due to the nature of the issue. Every model produced very high Sensitivity but low Specificity. This means the models cannot detect false alarms very well and are prone to classifying False Positives. This issue can be solved by using the Accuracy metric when presenting to stakeholders.

The Random Forest Model produced the highest accuracy of all models at 83. 62% when used on the training data set. It determined that Age, Number of Products Used and Member Activity are the most important predictors of customer exit.

We recommend that banks develop a marketing campaign to increase the number of products their customers use, this will simultaneously increase Member Activity.

Customers should be clustered by age. 20-30, 31-40, 41-50, 51-60, and 61+. This will allow banks to promote services to each segment that specifically appeal to that segment. For example, 20-30 year olds are much more interested in establishing credit than other segments and should have credit cards promoted to them. The 61+ segment will be more interested in accessing their retirement funds and should have financial advisory services promoted to them.

Implementing a marketing campaign will increase customer retention by causing them to use more bank products they are interested in. This makes leaving a much bigger hassle, making it less likely to occur.

As in Phase 1, your primary deliverable is a six slide PowerPoint presentation. This presentation should summarize your results. Recall that your audience for this deliverable is non-technical. Please focus on the practical use of your models. Please do not include any R code in your slides. You will also upload your knitted R work to Canvas.