randomforestM4cohen

#necessary packages  
options(tidyverse.quiet = TRUE)  
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 3.5.3

## Warning: package 'ggplot2' was built under R version 3.5.3

## Warning: package 'tibble' was built under R version 3.5.3

## Warning: package 'tidyr' was built under R version 3.5.3

## Warning: package 'readr' was built under R version 3.5.3

## Warning: package 'purrr' was built under R version 3.5.3

## Warning: package 'dplyr' was built under R version 3.5.3

## Warning: package 'stringr' was built under R version 3.5.3

## Warning: package 'forcats' was built under R version 3.5.3

library(caret)

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

library(ranger)

## Warning: package 'ranger' was built under R version 3.5.3

#read in blood.csv, rename variables per specifications and convert select variables to factors  
blooddata = read\_csv("blood.csv")

## Parsed with column specification:  
## cols(  
## Mnths\_Since\_Last = col\_double(),  
## TotalDonations = col\_double(),  
## Total\_Donated = col\_double(),  
## Mnths\_Since\_First = col\_double(),  
## DonatedMarch = col\_double()  
## )

#DonatedMarch comes in as double, below convert to factor-interpreting numbers as characters  
blooddata = blooddata %>% mutate(DonatedMarch = as\_factor(as.character(DonatedMarch)))%>%  
 mutate(DonatedMarch = fct\_recode(DonatedMarch, "Yes" = "1", "No" = "0"))  
str(blooddata)

## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 748 obs. of 5 variables:  
## $ Mnths\_Since\_Last : num 2 0 1 2 1 4 2 1 2 5 ...  
## $ TotalDonations : num 50 13 16 20 24 4 7 12 9 46 ...  
## $ Total\_Donated : num 12500 3250 4000 5000 6000 1000 1750 3000 2250 11500 ...  
## $ Mnths\_Since\_First: num 98 28 35 45 77 4 14 35 22 98 ...  
## $ DonatedMarch : Factor w/ 2 levels "Yes","No": 1 1 1 1 2 2 1 2 1 1 ...

# Task 1

Split the dataset into train and test datasets (70%,30%)

set.seed(1234)  
train.rows = createDataPartition(y = blooddata$DonatedMarch, p=0.7, list = FALSE) #70% in training  
train = blooddata[train.rows,]   
test = blooddata[-train.rows,]

# Task 2

Create random forest model on training dataset to specification.

fit\_control = trainControl(method = "cv",  
 number = 10) #set up 10 fold cross-validation  
  
set.seed(123)  
rf\_fit = train(x=as.matrix(train[,-5]), y=as.matrix(train$DonatedMarch), #using as.matrix function to pass tibble which does not play nicely with train function...  
 method = "ranger",   
 importance = "permutation",  
 trControl = fit\_control,  
 num.trees = 100)

# Task 3

View details and discuss variables importance in predicting DonatedMarch.

varImp(rf\_fit)

## ranger variable importance  
##   
## Overall  
## Total\_Donated 100.000  
## TotalDonations 38.494  
## Mnths\_Since\_First 7.657  
## Mnths\_Since\_Last 0.000

rf\_fit

## Random Forest   
##   
## 524 samples  
## 4 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 471, 471, 472, 472, 471, 472, ...   
## Resampling results across tuning parameters:  
##   
## mtry splitrule Accuracy Kappa   
## 2 gini 0.7804790 0.3105144  
## 2 extratrees 0.7880987 0.3133046  
## 3 gini 0.7804790 0.3284588  
## 3 extratrees 0.7747097 0.2923162  
## 4 gini 0.7689768 0.2939497  
## 4 extratrees 0.7727504 0.2903873  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 1  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 2, splitrule =  
## extratrees and min.node.size = 1.

The most important variable in the dataset is Total\_Donated and the least important is Mnths\_Since\_Last.

# Task 4

Develop and display first 6 predictions on training dataset.

predRF = predict(rf\_fit)  
head(predRF)

## [1] Yes Yes No No Yes Yes  
## Levels: No Yes

# Task 5 & 6

Create confusion matrix on training dataset and comment.

confusionMatrix(predRF, train$DonatedMarch, positive = "Yes")

## Warning in confusionMatrix.default(predRF, train$DonatedMarch, positive  
## = "Yes"): Levels are not in the same order for reference and data.  
## Refactoring data to match.

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 81 5  
## No 44 394  
##   
## Accuracy : 0.9065   
## 95% CI : (0.8783, 0.93)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7117   
##   
## Mcnemar's Test P-Value : 5.681e-08   
##   
## Sensitivity : 0.6480   
## Specificity : 0.9875   
## Pos Pred Value : 0.9419   
## Neg Pred Value : 0.8995   
## Prevalence : 0.2385   
## Detection Rate : 0.1546   
## Detection Prevalence : 0.1641   
## Balanced Accuracy : 0.8177   
##   
## 'Positive' Class : Yes   
##

The accuracy of this model is 91% which is an improvement from the naive model which is 76% accurate. The sensitivity of predicting DonatedMarch Yes’s is 65% and the sensitivity of predicting DonatedMarch No’s is 99%. The model is significantly different from the naive model in accuracy.

# Task 7

Use model to develop predictions, create confusion matrix and comment on performance on testing dataset.

predRF\_test = predict(rf\_fit, newdata = test)  
  
confusionMatrix(predRF\_test, test$DonatedMarch, positive = "Yes")

## Warning in confusionMatrix.default(predRF\_test, test$DonatedMarch,  
## positive = "Yes"): Levels are not in the same order for reference and data.  
## Refactoring data to match.

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 15 12  
## No 38 159  
##   
## Accuracy : 0.7768   
## 95% CI : (0.7165, 0.8296)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.351547   
##   
## Kappa : 0.2562   
##   
## Mcnemar's Test P-Value : 0.000407   
##   
## Sensitivity : 0.28302   
## Specificity : 0.92982   
## Pos Pred Value : 0.55556   
## Neg Pred Value : 0.80711   
## Prevalence : 0.23661   
## Detection Rate : 0.06696   
## Detection Prevalence : 0.12054   
## Balanced Accuracy : 0.60642   
##   
## 'Positive' Class : Yes   
##

Using the model on the testing dataset, the accuracy of this model is 78%. The sensitivity of predicting DonatedMarch Yes’s is 28% and the sensitivity of predicting DonatedMarch No’s is 92%. I would have concerns about using this model on unseen data. While sensitivity of correctly predicting those who didn’t donate, it is doing pretty poorly at predicting who donated in March. On the training data, the model was predicting 65% but on the testing data it is predicting less than 50%. The consequences of not marketing or soliciting to these likely donors could cost the organization unnecessarily. I might try to run the model again but maybe without the restrictions on the number of trees and see how that model performs on the testing data.