# Random Forests Assignment

## BAN502 Predictive Analytics

### Susan Wiggins

Libraries needed for assignment:

library(tidyverse)

## -- Attaching packages ---------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.1 v purrr 0.3.2  
## v tibble 2.1.1 v dplyr 0.8.1  
## v tidyr 0.8.3 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

## -- 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(ranger)

Reading in the data set:

library(readr)  
Blood <- 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()  
## )

View(Blood)  
  
  
Blood = Blood %>% mutate(DonatedMarch = as\_factor(as.character(DonatedMarch)))%>%   
mutate(DonatedMarch = fct\_recode(DonatedMarch, "No" = "0", "Yes" = "1"))  
  
glimpse(Blood)

## Observations: 748  
## Variables: 5  
## $ Mnths\_Since\_Last <dbl> 2, 0, 1, 2, 1, 4, 2, 1, 2, 5, 4, 0, 2, 1, 2,...  
## $ TotalDonations <dbl> 50, 13, 16, 20, 24, 4, 7, 12, 9, 46, 23, 3, ...  
## $ Total\_Donated <dbl> 12500, 3250, 4000, 5000, 6000, 1000, 1750, 3...  
## $ Mnths\_Since\_First <dbl> 98, 28, 35, 45, 77, 4, 14, 35, 22, 98, 58, 4...  
## $ DonatedMarch <fct> Yes, Yes, Yes, Yes, No, No, Yes, No, Yes, Ye...

str(Blood)

## 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 ...

summary(Blood)

## Mnths\_Since\_Last TotalDonations Total\_Donated Mnths\_Since\_First  
## Min. : 0.000 Min. : 1.000 Min. : 250 Min. : 2.00   
## 1st Qu.: 2.750 1st Qu.: 2.000 1st Qu.: 500 1st Qu.:16.00   
## Median : 7.000 Median : 4.000 Median : 1000 Median :28.00   
## Mean : 9.507 Mean : 5.515 Mean : 1379 Mean :34.28   
## 3rd Qu.:14.000 3rd Qu.: 7.000 3rd Qu.: 1750 3rd Qu.:50.00   
## Max. :74.000 Max. :50.000 Max. :12500 Max. :98.00   
## DonatedMarch  
## Yes:178   
## No :570   
##   
##   
##   
##

### Task 1

Split the dataset into training (70%) and testing (30%) sets. Use set.seed of 1234.

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

### Task 2

Create a random forest model on the training set to predict DonatedMarch using all of the variables in the dataset. Use caret’s trainControl function to set up 10 fold cross-validation. Use a random number seed of 123. Use 100 trees (Note you can specify the number of trees by adding a line num.trees = 100 to the rf\_fit block of code).

Blood = Blood %>% select(c("Mnths\_Since\_Last","TotalDonations", "Total\_Donated","Mnths\_Since\_First", "DonatedMarch"))  
  
  
  
#blood = complete(Blood)   
#summary(blood)

fit\_control = trainControl(method = "cv",   
 number = 10) #set up 10 fold cross-validation  
  
#rf\_fit = train(DonatedMarch ~.,  
  
  
set.seed(123)   
rf\_fit = train(x=as.matrix(train[,-5]), y=as.matrix(train$DonatedMarch),  
 method = "ranger",   
 importance = "permutation",  
 trControl = fit\_control,  
 num.trees = 100)

### Task 3

Using varImp, what is the most important variable in the model, what is the least important?

Most Important variable is Total\_Donated.

Least Important variable is Mnths\_Since\_Last.

Check out random forest details

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.

### Task 4

Use the model to develop predictions on the training set. Use the “head” function to display the first six predictions.

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

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

### Task 5

Use the model to create a confusion matrix using caret’s confusionMatrix function for the training set. What is the accuracy, sensitivity, and specificity of the model?

The accuracy of the model is 0.9065, 90.65% accurate.

The sensitivity of the model is 0.6480, 64.80% sensitivity

The specificity of the model is 0.9875, 98.75% specificity.

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   
##

### Task 6

How does the accuracy of the model compare to a naive model that assumes that all observations are in the majority class?

The Random forest model shows an accuracy of 90.65% whereas the naive model that assumes that all observations are in the majority class is 76.15%. The difference of these two values is 14.50%. Therefore, a significance level in change for the random forest does exists. The p-value also is less than .05 indicating that the model is good.

### Task 7

Use the model to develop predictions on the test set. Develop a confusion matrix. How does the model perform on the testing set?

On the testing set the decreases on accuracy to 77.68%, and the naive model readings are about the same. Sensitivity also decrease some to 0.28302, 28.30%. The specificity is about a 6% decrease for the testing set also. The p-value is farther away from .05 in the training model than the testing model also. The p-value is greater than .05 in the testing model and is listed as 0.351547.

predRF\_test = predict(rf\_fit, newdata = test)  
head(predRF\_test)

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

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   
##

### Task 8

Comment on how this model might be used in the “real-world.” Would you recommend this model for real-world use? What if any concerns would you have about using the model?

Caret is trying to build the best model possible for performance on data the model has never seen, so we are trying to build the most robust model on the testing set even though the training to testing set decreased in the accuracy this is probably the best model to be used. I would be concerned about the value of the p-value has decreased significantly from training to testing the training model is less than .05 whereas the testing is greater than .05. I might also be concerned with the factor that there is around a 10% decrease in training to testing accuracy. The sensitvity also has a large decrease in difference from training to testing so ;therefore, I believe there are some concerns for using this in a real world situation.