## Random Forests

### Anthony Sumter

**Libraries**

#install.packages("tidyverse","caret", "ranger" )  
options(tidyverse.quiet = TRUE)  
library(tidyverse)  
library(caret)

## Warning: package 'caret' was built under R version 3.6.2

## 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 3.6.2

**Dataset**

blood = read.csv("Blood.csv")  
blood = blood %>% mutate(DonatedMarch = as\_factor(as.character(DonatedMarch))) %>%  
mutate(DonatedMarch = fct\_recode(DonatedMarch,"No" = "0","Yes" = "1"))  
str(blood)

## 'data.frame': 748 obs. of 5 variables:  
## $ Mnths\_Since\_Last : int 2 0 1 2 1 4 2 1 2 5 ...  
## $ TotalDonations : int 50 13 16 20 24 4 7 12 9 46 ...  
## $ Total\_Donated : int 12500 3250 4000 5000 6000 1000 1750 3000 2250 11500 ...  
## $ Mnths\_Since\_First: int 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.

blood = blood %>% drop\_na()  
str(blood)

## 'data.frame': 748 obs. of 5 variables:  
## $ Mnths\_Since\_Last : int 2 0 1 2 1 4 2 1 2 5 ...  
## $ TotalDonations : int 50 13 16 20 24 4 7 12 9 46 ...  
## $ Total\_Donated : int 12500 3250 4000 5000 6000 1000 1750 3000 2250 11500 ...  
## $ Mnths\_Since\_First: int 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 ...

set.seed(1234)  
train.rows2 = createDataPartition(y = blood$DonatedMarch, p=0.7, list = FALSE)  
train3 = blood[train.rows2,]   
test3 = blood[-train.rows2,]

#### 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).

fit\_control = trainControl(method = "cv",number = 10)  
set.seed(123)   
rf\_fit = train(x=train3[,-1], y=train3$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?

The most important variable in the model is DonatedMarch while Total\_Donated is the least important.

varImp(rf\_fit)

## ranger variable importance  
##   
## Overall  
## DonatedMarch 100.00000  
## Mnths\_Since\_First 0.24350  
## TotalDonations 0.02558  
## Total\_Donated 0.00000

rf\_fit

## Random Forest   
##   
## 524 samples  
## 4 predictor  
## 2 classes: 'Yes', 'No'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 472, 472, 471, 471, 471, 472, ...   
## Resampling results across tuning parameters:  
##   
## mtry splitrule Accuracy Kappa  
## 2 gini 1 1   
## 2 extratrees 1 1   
## 3 gini 1 1   
## 3 extratrees 1 1   
## 4 gini 1 1   
## 4 extratrees 1 1   
##   
## 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 = gini  
## 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.train(rf\_fit, train3)  
head(predRF)

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

#### 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, sensitivity, and specificity of this model are all showing at 100%.

confusionMatrix(predRF, train3$DonatedMarch, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 125 0  
## No 0 399  
##   
## Accuracy : 1   
## 95% CI : (0.993, 1)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.2385   
## Detection Rate : 0.2385   
## Detection Prevalence : 0.2385   
## Balanced Accuracy : 1.0000   
##   
## '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 accuracy compared to the naive model is a better predictor when exposed to new data at 100% while the naive model is only at 76.1%.

#### 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?

The model looks to perform the for the most part the same way on the test data as the model constructed based on the train model. Both models have accuracy, sensitivity, and specificity at 100%. The naive model is slightly different with the train model coming in at 76.1% while for the test model the naive model came in slightly higher at 76.3%.

predRF\_test = predict(rf\_fit, newdata = test3)  
head(predRF\_test)

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

confusionMatrix(predRF\_test, test3$DonatedMarch, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 53 0  
## No 0 171  
##   
## Accuracy : 1   
## 95% CI : (0.9837, 1)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.2366   
## Detection Rate : 0.2366   
## Detection Prevalence : 0.2366   
## Balanced Accuracy : 1.0000   
##   
## '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?

This model could be used in the real-world with upcoming election to predict which party is likely to win. I would recommend this model if the data size is similar as the accuracy being 100% would make me confident in the results produced.Some concerns that could arise is if the data set becomes larger then it would take longer time to get a result. In addition future performance issues could arise within the specific application used to compute results.