# Random Forests

### Miguel Fernandez

### BAN-502

# Import libraries  
library(tidyverse)  
library(caret)  
library(ranger)

#### Task 1

# Read in data  
blood <- read\_csv("Blood.csv")  
# Create factor  
blood <- blood %>%  
 mutate(DonatedMarch = as\_factor(as.character(DonatedMarch))) %>%  
 mutate(DonatedMarch = fct\_recode(DonatedMarch,"Yes" = "1", "No" = "0"))  
# Set random seed  
set.seed(1234)  
# Split data for training and testing  
train.rows <- createDataPartition(y = blood$DonatedMarch,  
 p = 0.7,  
 list = FALSE)  
train <- slice(blood, train.rows)  
test <- slice(blood, -train.rows)

#### Task 2

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

#### Task 3

# Check variable importance  
varImp(rf\_fit)

## ranger variable importance  
##   
## Overall  
## TotalDonations 100.00  
## Mnths\_Since\_First 40.88  
## Total\_Donated 23.41  
## Mnths\_Since\_Last 0.00

The table reports measures of variable importance in the model created above. The variable TotalDonations received a perfect score of 100. It appears that Mnths\_Since\_Last is not important as it received a score of 0. These results are somewhat intuitive. If someone has a high number of total donations, there is a greater chance that one of those donations occurred in March.

#### Task 4

# Make predictions on training set  
pred.train = predict(rf\_fit, train)  
head(pred.train)

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

#### Task 5

# Create confusion matrix  
confusionMatrix(pred.train, train$DonatedMarch, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 77 3  
## No 48 396  
##   
## Accuracy : 0.9027   
## 95% CI : (0.874, 0.9267)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6943   
##   
## Mcnemar's Test P-Value : 7.218e-10   
##   
## Sensitivity : 0.6160   
## Specificity : 0.9925   
## Pos Pred Value : 0.9625   
## Neg Pred Value : 0.8919   
## Prevalence : 0.2385   
## Detection Rate : 0.1469   
## Detection Prevalence : 0.1527   
## Balanced Accuracy : 0.8042   
##   
## 'Positive' Class : Yes   
##

Analyzing the confusion matrix and statistics, we can see a fairly robust model. Model accuracy was just above 0.9. The sensitivity of the model is 0.62 indicating that the model correctly identified 62 percent of the true positive values. However, the specificity scored well with 0.99 meaning 99 percent of the true negatives were classified. If the purpose of the model is to identify those who have not donated, then the model performed very well. Next we will use the test data and see how the model performs on data it has never seen before.

#### Task 6

The model performed much better than if all values were predicted “No”. The no information rate, which measures the accuracy if all values were predicted as the majority class, is 0.76. This is approximately 14 points lower than the actual model performed. The statistics above also reveal a significant difference between the model accuracy and the no information rate as the p-value for this relationship is near zero.

#### Task 7

# Make predictions on testing set  
pred.test = predict(rf\_fit, test)  
head(pred.test)

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

# Create confusion matrix  
confusionMatrix(pred.test, test$DonatedMarch, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 13 17  
## No 40 154  
##   
## Accuracy : 0.7455   
## 95% CI : (0.6832, 0.8012)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.762475   
##   
## Kappa : 0.1716   
##   
## Mcnemar's Test P-Value : 0.003569   
##   
## Sensitivity : 0.24528   
## Specificity : 0.90058   
## Pos Pred Value : 0.43333   
## Neg Pred Value : 0.79381   
## Prevalence : 0.23661   
## Detection Rate : 0.05804   
## Detection Prevalence : 0.13393   
## Balanced Accuracy : 0.57293   
##   
## 'Positive' Class : Yes   
##

The model did not perform nearly as well on the testing data set as it did on the training data set. In fact, the no information rate is higher than the model accuracy which means that if the model predicted “No” for every observation, it would perform better than the actual model. Looking at the sensitivity, we can see that the model struggled to predict positive values. It correctly identified only a quarter of all true positive values. The specificity of the model on the testing set is still strong but not quite as strong as it was on the training set.

#### Task 8

This model has many real-world applications. Consider a hospital that has been collecting data on patients as they come in to donate blood. The hospital is running low on their blood supply which can be stored for over a month before it becomes unusable. The hospital has limited advertising resources and they would be interested in knowing which people have not donated to direct their advertising towards. The model created above would perform well given the high measures of specificity. Staff would now have a good list of patients to target for blood donation advertisements.

A few additional steps should be taken before this model is deployed. While the model performed well on the training set, measures of accuracy and performance were lower on the testing set. Perhaps the model could be trained using the entire data set and the k-folds cross-validation technique instead of splitting the data up into training and testing sets. This would ensure that the model was able to learn from as much data as possible. If the decision was made to go with the training/testing split, a larger data set might help improve model performance. For example, the original data set was 748 observations and a 70/30 split on the data left 524 observations for the training data set. If the data set were doubled in size to approximately 1,500 observations, the model could be trained on a larger data set.