options(tidyverse.quiet=TRUE)  
library(tidyverse)  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

library(ranger)

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()  
## )

Blood<-Blood %>% mutate(DonatedMarch = as\_factor(as.character(DonatedMarch))) %>% #1  
mutate(DonatedMarch = fct\_recode(DonatedMarch,  
"Yes"="1",  
"No"= "0"))  
#str(Blood)

set.seed(1234)  
train.rows<- createDataPartition(y = Blood$DonatedMarch, p=0.7, list = FALSE)  
train<- slice(Blood, train.rows)  
test<- slice(Blood,-train.rows)

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

saveRDS(rf\_fit, "rf\_fit.rds")

rf\_fit = readRDS("rf\_fit.rds")

varImp(rf\_fit)

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

rf\_fit

## Random Forest   
##   
## 524 samples  
## 4 predictor  
## 2 classes: 'No', 'Yes'   
##   
## 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 0.8035922 0.3828006  
## 2 extratrees 0.8074020 0.3812627  
## 3 gini 0.7787010 0.3328348  
## 3 extratrees 0.7997097 0.3719429  
## 4 gini 0.7902032 0.3585442  
## 4 extratrees 0.7785922 0.3266171  
##   
## 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 3** Using the varImp function, we can see the importance of each variable. Our most important variable for our predictor appears to be TotalDonations, while Mnths\_Since\_Last is the least important.

predRF = predict.train(rf\_fit, train)  
head(predRF)

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

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 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   
##

**Task 5** Below are the accuracy, sensitivity and specificity of our confusion matrix: Accuracy : 0.9027 Sensitivity : 0.6160 Specificity : 0.9925

**Task 6** The accuracy of our model is significantly better than the naive model, with more than a 14% increase in accuracy and a P-value of < 2.2e-16.

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

## [1] Yes Yes Yes No No Yes  
## 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 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   
##

**Task 7** Using the model we created, we can make predictions on our test set to validate our answers. My predictions on the test set are not that great in comparison to the training set. Our accuracy falls to 0.75, compared to the naive model which is .7634. Sensitivity and Specificity are also quite different between the test and training data. So based on the results, barring any errors made, the model does not perform that well on the testing set.

**Task8** This model could easily be used in a real world application. Using this model in healthcare could show which variables are most significant in a patient being readmitted to the hospital, or what the liklihood of a retail store having a return based on certain criteria. I would recommend this model for real world use because of the easily understandable output and the lower chance of overfitting the data. The only concern for the model seems to be if it is a large data set it may be very time consuming to run the model.