## Week4 Assignment 3

### DEY, Sankha

#### Random Forests

# message=FALSE done before last knit  
library(tidyverse)  
library(caret)  
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))) %>%   
mutate(DonatedMarch = fct\_recode(DonatedMarch,  
"No" = "0",  
"Yes" = "1"))  
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 ...

#### Task 1

set.seed(1234)   
trainrows43 = createDataPartition(y = Blood$DonatedMarch, p=0.7, list = FALSE) #70% in training  
train43 = slice(Blood,trainrows43)  
test43 = slice(Blood,-trainrows43)

Train set has 524 observations and test set has 224 observations.

#### Task 2

fit\_control = trainControl(method = "cv",   
 number = 10) #set up 10 fold cross-validation  
  
  
set.seed(123)   
rf\_fit = train(x=as.matrix(train43[,-5]), y=as.matrix(train43$DonatedMarch),   
 method = "ranger",   
 importance = "permutation",  
 trControl = fit\_control,  
 num.trees = 100)

#### Task 3

varImp(rf\_fit)

## ranger variable importance  
##   
## Overall  
## TotalDonations 100.00  
## Total\_Donated 61.02  
## Mnths\_Since\_First 43.18  
## Mnths\_Since\_Last 0.00

Using varImp function, we see TotalDonations is the most important variable in the model. Least important variable is Mnths\_Since\_Last.

#### Task 4

predRF = predict.train(rf\_fit, train43)  
head(predRF)

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

#### Task 5

confusionMatrix(predRF, train43$DonatedMarch, positive = "Yes")

## Warning in confusionMatrix.default(predRF, train43$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 78 4  
## No 47 395  
##   
## Accuracy : 0.9027   
## 95% CI : (0.874, 0.9267)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6962   
##   
## Mcnemar's Test P-Value : 4.074e-09   
##   
## Sensitivity : 0.6240   
## Specificity : 0.9900   
## Pos Pred Value : 0.9512   
## Neg Pred Value : 0.8937   
## Prevalence : 0.2385   
## Detection Rate : 0.1489   
## Detection Prevalence : 0.1565   
## Balanced Accuracy : 0.8070   
##   
## 'Positive' Class : Yes   
##

Accuracy is 90.27%  
Sensitivity is 62.40%  
Specificity is 99.00%

#### Task 6

Our training model accuracy is 0.9027. The no-information rate is 0.7615. This is the accuracy achievable by always predicting the majority class label. In this case if asked to predict whether a person will donate in March or not, by always choosing “No (Won’t Donate)” we can achive nearly 76.15% accuracy on the training set. So, our 90.27% accuracy is relatively better than the Naive model. A lower p-value (< 2.2e-16) supports our conclusion that our accuracy is better than the Naive model.

#### Task 7

Predictions on test

predRF\_test = predict(rf\_fit, newdata = test43)

Confusion matrix

confusionMatrix(predRF\_test, test43$DonatedMarch, positive = "Yes")

## Warning in confusionMatrix.default(predRF\_test, test43$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 21  
## No 40 150  
##   
## Accuracy : 0.7277   
## 95% CI : (0.6644, 0.7848)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.90769   
##   
## Kappa : 0.1398   
##   
## Mcnemar's Test P-Value : 0.02119   
##   
## Sensitivity : 0.24528   
## Specificity : 0.87719   
## Pos Pred Value : 0.38235   
## Neg Pred Value : 0.78947   
## Prevalence : 0.23661   
## Detection Rate : 0.05804   
## Detection Prevalence : 0.15179   
## Balanced Accuracy : 0.56124   
##   
## 'Positive' Class : Yes   
##

Now this model doesn’t look very impressive on test data with Accuracy only 0.7277 and No Information Rate is 0.7634. So, Accuracy is less than Naive model. Hence, the model doesn’t perform well on the test data.

#### Task 8

This model might be used to chech what variables have najor impacts on the response variable. However, I don’t recommend this model to be used in real-life prediction scenario because the Accuracy of the model is not better than the Naive model. Naive is the accuracy achievable by always predicting the majority class label. In other words, if asked to predict whether a person will donate blood in March or not, by always choosing “No (Won’t Donate)” we can achive nearly 76.15% accuracy on the testing set which is better than the model Accuracy (72.77%). A higher p-value (0.9) supports our conclusion. So, this model doesn’t have much significance on the accuracy of real-life predictions. A Naive can do better. Training data might have been overfitted.