Assignment3

Jordan Whitaker

6/6/2020

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

## Warning: package 'tidyverse' was built under R version 3.6.3

## Warning: package 'ggplot2' was built under R version 3.6.3

## Warning: package 'tidyr' was built under R version 3.6.3

## Warning: package 'readr' was built under R version 3.6.3

## Warning: package 'dplyr' was built under R version 3.6.3

## Warning: package 'stringr' was built under R version 3.6.3

library(caret)

## Warning: package 'caret' was built under R version 3.6.3

## 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 3.6.3

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()  
## )

head(blood)

## # A tibble: 6 x 5  
## Mnths\_Since\_Last TotalDonations Total\_Donated Mnths\_Since\_First DonatedMarch  
## <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 2 50 12500 98 1  
## 2 0 13 3250 28 1  
## 3 1 16 4000 35 1  
## 4 2 20 5000 45 1  
## 5 1 24 6000 77 0  
## 6 4 4 1000 4 0

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

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 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. Mnths\_Since\_Last = col\_double(),  
## .. TotalDonations = col\_double(),  
## .. Total\_Donated = col\_double(),  
## .. Mnths\_Since\_First = col\_double(),  
## .. DonatedMarch = col\_double()  
## .. )

set.seed(1234)  
train.rows = createDataPartition(y= blood$DonatedMarch, p=0.7, list= FALSE)  
  
train = blood[train.rows,]  
test = blood[-train.rows,]  
  
head(train)

## # A tibble: 6 x 5  
## Mnths\_Since\_Last TotalDonations Total\_Donated Mnths\_Since\_First DonatedMarch  
## <dbl> <dbl> <dbl> <dbl> <fct>   
## 1 0 13 3250 28 Yes   
## 2 1 16 4000 35 Yes   
## 3 2 20 5000 45 Yes   
## 4 1 24 6000 77 No   
## 5 4 4 1000 4 No   
## 6 2 9 2250 22 Yes

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   
##   
##   
##   
##

blood[,-5]

## # A tibble: 748 x 4  
## Mnths\_Since\_Last TotalDonations Total\_Donated Mnths\_Since\_First  
## <dbl> <dbl> <dbl> <dbl>  
## 1 2 50 12500 98  
## 2 0 13 3250 28  
## 3 1 16 4000 35  
## 4 2 20 5000 45  
## 5 1 24 6000 77  
## 6 4 4 1000 4  
## 7 2 7 1750 14  
## 8 1 12 3000 35  
## 9 2 9 2250 22  
## 10 5 46 11500 98  
## # ... with 738 more rows

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

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: The most important variable accoarding to varImp is TotalDonations while the least important is Mnths\_Since\_Last.

predRF = predict(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: The accuracy of the random forest model when applied to the training dataset is strong at .8989 and the sensitivity and specificty are .60 and .9925 respectively. The specificty is high at .9925, which shows the true negative rate. Correctly classifying the people who are predicted to not donate blood in March.

Task 6: The model that I generated has an accuracy of .9027 while the naive model sits at an accuracy of .7615, this shows that the model would do much better than simply classifying everyone into the majority class. The p-value shows that the model that I created is statistically significant when compared to the naive model.

Predictions on Test

predRF\_test = predict(rf\_fit, newdata = test)  
  
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: The model does not perform well on the testing dataset. The accuracy drops from 90% to 75%. The model is also not significantly better than the naive model.

Task 8: This model would struggle when it is fed “real world” data and I would not recommend it to be used in real-world applications. I would be concerned that it would misclassify people too often resulting in poor data driven decision making.