Clasification

Oluwadamilola Owolabi

2024-04-20

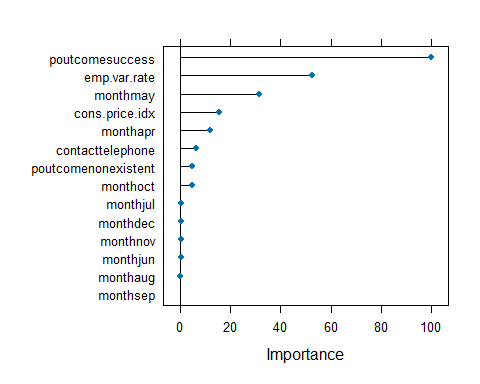
## RANDOM FOREST MODEL 1: SIMPLE LOGISTIC MODEL

For our non-parametric model, we plan on using Random forest on our Simple Logistic Model. It is an ensemble learning method that combines the predictions of multiple individual decision trees to improve the overall performance and robustness of the model.

#### SLM : CARET PACKAGE

We plan on using the caret package, which iterates through different mtry and ntree values, to find the optimum one

set.seed(1234) #setting the seed  
library(caret) # Loading the caret package  
  
# Running Random Forest  
# Specify the training control parameters  
train\_control <- trainControl(method = "cv", # Cross-validation method  
 number = 5,) # Number of folds  
  
# Define the random forest model  
fitted\_rf <- train(y ~ month + poutcome + emp.var.rate + contact + cons.price.idx, # formulae  
 data = train\_data, # Response variable  
 method = "rf", # Random forest method  
 trControl = train\_control) # Training control parameters  
  
#fitted\_rf  
  
#plotting the importance plot  
plot(varImp(fitted\_rf, horizontal = TRUE))

 From the plot above, we can see that the 3 most impactful variables in the SLM are poutcome, emp.var.rate and cons.price.idx.

#### Making predictions and getting the metrics

# Getting the threshold  
metrics = data.frame(thresh=seq(0, 1, by = 0.0001))   
num\_thresh <- nrow(metrics)  
  
#initializing the new metrics to 0  
metrics$sensitivity <- 0  
metrics$specificity <- 0  
metrics$ppv <- 0  
metrics$npv <- 0  
metrics$accuracy <- 0  
metrics$f1 <- 0  
predicted <- predict(fitted\_rf, newdata = train\_data, type = "prob")['yes']  
#Getting the threshold  
  
#Running a for loop to find the optimum threshold  
for (i in 1:num\_thresh){  
 if(i %% 100 == 0) {  
 #print(paste(i,'/',num\_thresh,sep=''))  
 }  
   
 # Confusion Matrix  
 predicted\_classes <- ifelse(predicted[, "yes"] > metrics$thresh[i], 'yes', 'no')  
 predicted\_classes\_factor <- factor(predicted\_classes, levels = levels(train\_data$y))  
 confusion\_matrix <- confusionMatrix(predicted\_classes\_factor, train\_data$y)  
   
   
 # Metrics  
 metrics$sensitivity[i] <- as.numeric(confusion\_matrix$byClass['Sensitivity'])  
 metrics$specificity[i] <- as.numeric(confusion\_matrix$byClass['Specificity'])  
 metrics$ppv[i] <- as.numeric(confusion\_matrix$byClass['Pos Pred Value'])  
 metrics$npv[i] <- as.numeric(confusion\_matrix$byClass['Neg Pred Value'])  
 metrics$accuracy[i] <- as.numeric(confusion\_matrix$overall['Accuracy'])  
 metrics$f1[i] <- as.numeric(confusion\_matrix$byClass['F1'])  
}  
  
# Get threshold value that maximizes F1  
# Get F1 thresholds  
maxF1 <- max(metrics$f1, na.rm = TRUE) #   
maxF1 # 0.4593

## [1] 0.9468008

theshF1 <- metrics$thresh[which.max(metrics$f1)]   
theshF1 # 0.004

## [1] 0.534

# Test data  
predicted <- predict(fitted\_rf, newdata = test\_data, type = "prob")['yes']  
predicted\_classes <- ifelse(predicted[, "yes"] > theshF1, 'yes', 'no')  
CM <- confusionMatrix(as.factor(predicted\_classes), as.factor(test\_data$y))

## Warning in confusionMatrix.default(as.factor(predicted\_classes),  
## as.factor(test\_data$y)): Levels are not in the same order for reference and  
## data. Refactoring data to match.

CM

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction yes no  
## yes 215 125  
## no 696 7202  
##   
## Accuracy : 0.9003   
## 95% CI : (0.8937, 0.9067)  
## No Information Rate : 0.8894   
## P-Value [Acc > NIR] : 0.0007196   
##   
## Kappa : 0.3018   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.23600   
## Specificity : 0.98294   
## Pos Pred Value : 0.63235   
## Neg Pred Value : 0.91188   
## Prevalence : 0.11059   
## Detection Rate : 0.02610   
## Detection Prevalence : 0.04127   
## Balanced Accuracy : 0.60947   
##   
## 'Positive' Class : yes   
##

#Printing out the metric  
Sensitivity <- CM$byClass["Sensitivity"]  
Specificity <- CM$byClass["Specificity"]  
Prevalence <- CM$byClass["Prevalence"]  
PPV <- CM$byClass["Pos Pred Value"]  
NPV <- CM$byClass["Neg Pred Value"]  
F1 <- (2 \* Sensitivity \* PPV)/(Sensitivity + PPV)  
  
#AUROC  
predicted\_classes <- factor(predicted\_classes, levels = c("yes", "no"))  
roc\_rf <- roc(response=test\_data$y,predictor= as.numeric(predicted\_classes),levels=c("no","yes"),direction = ">")  
auroc <- auc(roc\_rf)  
  
#printing merics  
cat("F1: ", F1, "\n") # 0.4380

## F1: 0.343725

cat("Sensitivity: ", Sensitivity, "\n") #0.3820

## Sensitivity: 0.2360044

cat("Specificity: ", Specificity, "\n") #0.9550

## Specificity: 0.9829398

cat("Prevalence: ", Prevalence, "\n") #0.1106

## Prevalence: 0.1105851

cat("PPV: ", PPV, "\n") #0.5133

## PPV: 0.6323529

cat("NPV: ", NPV, "\n") #0.9255

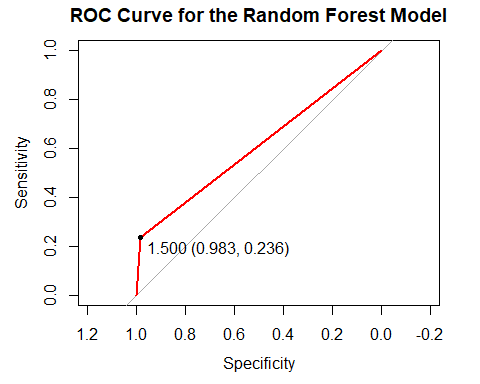
## NPV: 0.9118764

cat("AUROC: ", auroc, "\n") #0.6685

## AUROC: 0.6094721

#### GETTING THE ROC CURVE

# Print the AUROC  
auroc <- auc(roc\_rf) # 0.6685  
  
plot(roc\_rf,print.thres="best",col="red")  
title(main = 'ROC Curve for the Random Forest Model', line = 3)



The graphs looks different. It looks like the Random Forest model is confident of its predictions. Its most likely due to overfitting due to the high complexity of the Random Forest model.

#### SLM : RANDOM FOREST PACKAGE

Running the random forest again using the Random Forest Package this time. I am looking for the one to produce the best AUC value. For my hyperparameters, I chose mtry = 2 and ntree = 6000.We plan on using the caret package, which iterates through different mtry and ntree values, to find the optimum one

set.seed(1234) #setting the seed  
  
# Convert the binary outcome to a factor  
train\_data$y <- as.factor(train\_data$y)

# Define the random forest model  
fitted\_rf1.2 <-randomForest(y ~ month + poutcome + emp.var.rate + contact + cons.price.idx, data=train\_data, ntree=5000, importance = TRUE, keep.forest=TRUE, mtry=3)

#### CONTRIBUTION PLOTS FROM THE RANDOM FOREST

Looking at the contribution plots of our RF results to visualize the data to see which variables contributed the most

importance\_data <- as.data.frame(importance(fitted\_rf1.2))

plot\_data <- data.frame(

Variable = row.names(importance\_data),

no = importance\_data$no,

yes = importance\_data$yes,

accuracy = importance\_data$MeanDecreaseAccuracy, #impact of each variable on the overall accuracy of the model

Impurity = importance\_data$MeanDecreaseGini # reduction in impurity (how well a variable separates the classes) achieved by each variable.

)

plot\_data <- plot\_data[order(plot\_data$accuracy, decreasing = TRUE), ]

*# Make a contribution plot*

ggplot(plot\_data, aes(x = Variable, y = accuracy)) +

geom\_bar(stat = "identity", fill = "skyblue", width = 0.7) +

labs(title = "Accuracy Contribution Plot - Random Forest",

x = "Variable",

y = "accuracy") +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

ggplot(plot\_data, aes(x = Variable, y = Impurity)) +

geom\_bar(stat = "identity", fill = "skyblue", width = 0.7) +

labs(title = "Impurity Contribution Plot - Random Forest",

x = "Variable",

y = "Impurity") +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 45, hjust = 1))

A graph with blue rectangles

Description automatically generated

A graph with blue rectangles

Description automatically generated

Based on the contribution plot, there is evidence that the 3 most influential variables within the dataset are Poutcome, cons.price.idx & emp.var.rate, which is similar to the previous caret package.

#### MAKING PREDICTIOND AND GETTONG METRICS

# Get threshold

metrics = data.frame(thresh=seq(0, 1, by = 0.0001))

num\_thresh <- nrow(metrics)

metrics$sensitivity <- 0

metrics$specificity <- 0

metrics$ppv <- 0

metrics$npv <- 0

metrics$accuracy <- 0

metrics$f1 <- 0

predicted <- data.frame(predict(fitted\_rf1.2, newdata = train\_data, type = "prob"))['yes']

#Getting the threshold

for (i in 1:num\_thresh){

if(i %% 100 == 0) {

#print(paste(i,'/',num\_thresh,sep=''))

}

# Confusion Matrix

predicted\_classes <- ifelse(predicted[, "yes"] > metrics$thresh[i], 'yes', 'no')

predicted\_classes\_factor <- factor(predicted\_classes, levels = levels(train\_data$y))

confusion\_matrix <- confusionMatrix(predicted\_classes\_factor, train\_data$y)

# Metrics

metrics$sensitivity[i] <- as.numeric(confusion\_matrix$byClass['Sensitivity'])

metrics$specificity[i] <- as.numeric(confusion\_matrix$byClass['Specificity'])

metrics$ppv[i] <- as.numeric(confusion\_matrix$byClass['Pos Pred Value'])

metrics$npv[i] <- as.numeric(confusion\_matrix$byClass['Neg Pred Value'])

metrics$accuracy[i] <- as.numeric(confusion\_matrix$overall['Accuracy'])

metrics$f1[i] <- as.numeric(confusion\_matrix$byClass['F1'])

}

# Get threshold value that maximizes F1

# Get F1 thresholds

maxF1 <- max(metrics$f1, na.rm = TRUE) #

maxF1 #0.4605

theshF1 <- metrics$thresh[which.max(metrics$f1)]

theshF1 # 0.0034

# Test data

predicted <- data.frame(predict(fitted\_rf1.2, newdata = test\_data, type = "prob"))['yes']

predicted\_classes <- ifelse(predicted[, "yes"] > theshF1, 'yes', 'no')

CM <- confusionMatrix(as.factor(predicted\_classes), as.factor(test\_data$y))

CM

#Printing out the metric

Sensitivity <- CM$byClass["Sensitivity"]

Specificity <- CM$byClass["Specificity"]

Prevalence <- CM$byClass["Prevalence"]

PPV <- CM$byClass["Pos Pred Value"]

NPV <- CM$byClass["Neg Pred Value"]

F1 <- (2 \* Sensitivity \* PPV)/(Sensitivity + PPV)

#AUROC

predicted\_classes <- factor(predicted\_classes, levels = c("yes", "no"))

roc\_rf <- roc(response=test\_data$y,predictor= as.numeric(predicted\_classes),levels=c("no","yes"),direction = ">")

auroc <- auc(roc\_rf)

cat("F1: ", F1, "\n")

cat("Sensitivity: ", Sensitivity, "\n")

cat("Specificity: ", Specificity, "\n")

cat("Prevalence: ", Prevalence, "\n")

cat("PPV: ", PPV, "\n")

cat("NPV: ", NPV, "\n")

cat("AUROC: ", auroc, "\n")

# Results:

# F1: 0.4405507

# Sensitivity: 0.3863886

# Specificity: 0.9542787

# Prevalence: 0.1105851

# PPV: 0.5123726

# NPV: 0.9259701

# AUROC: 0.6703336

The RF model for the randomForest package is greater than the caret package by 0.5%. Hence i will go ahead with this model

#### GETTING THE ROC CURVE

# Print the AUROC

auroc <- auc(roc\_rf) # 0.7044

plot(roc\_rf,print.thres="best",col="red")

title(main = 'ROC Curve for the Random Forest Model', line = 3)

A graph with a red line

Description automatically generated

The ROC curve is is similar to the previous model. Hence i can infer that this is the roc curve of what a random forest model would predict. Most likely due to the complexity.