Cameroon: Machine Learning Script for the Predition of Approval

Adrien Ratsimbaharison

June 3, 2019

## Introduction

This machine learning script is based on the “Practical guide to implement machine learning with CARET in R” suggested by Saurav Kaushik in the Analytics Vidhya website: <https://www.analyticsvidhya.com/blog/2016/12/practical-guide-to-implement-machine-learning-with-caret-package-in-r-with-practice-problem/> (December 8, 2016).

The Caret package solves many problems in the implementation of machine learning process, and make it much easier to execute, even for beginners like the autors of this book. As Kaushik puts it, “This package [i.e., CARET package] alone is all you need to know for solve almost any supervised machine learning problem. It provides a uniform interface to several machine learning algorithms and standardizes various other tasks such as Data splitting, Pre-processing, Feature selection, Variable importance estimation, etc.”

Among the different steps suggested by Saurav Kaushik, we will focus on the following:

1. Getting started with loading the package and reading the data
2. Pre-processing the data using Caret
3. Splitting the data using Caret
4. Feature selection using Caret
5. Training models using Caret
6. Variable importance estimation using Caret
7. Making predictions using Caret

## 1. Getting started with loading the package and looking at the data

Installing and loading the Caret package and its dependencies:

Reading the data in R and looking at its structure:

Defining the problem:

In this problem we want to predict the classification of Cameroonians into three categories, according to their responses to Question Q58A in Afrobarometer Round 7 survey (“Do you approve or disapprove of the way that the President has performed his jobs over the past 12 months?”):

* “Disapprove the way the President has performed his job”
* “Approve the way the President has performed his job”
* “Don’t know whether to approve or not the way the President has performed his job”

## 2. Pre-processing the data using Caret

In this step, we only check for the missing values, and skipped the other pre-processing procedures, such as centering, scaling, principal component analysis, and creation of “one hot encoding”.

sum(is.na(approvalData\_transformed))

## [1] 0

Since there is no missing value to remove, we can move to the next step, which is splitting the data.

## 3. Splitting the data using Caret

Since the dataset was initially arranged in a country-year format, it is a good idea to randomize the rows before splitting the dataset into trainSet and testSet.

# Randomizing the dataset  
set.seed(333)  
rows <- sample(nrow(approvalData\_transformed))  
approvalData\_transformed <- approvalData\_transformed[rows,]  
  
#Spliting dataset into trainSet and testSet based on outcome with a ratio of 75% and 25%, using createDataPartition in Caret  
index <- createDataPartition(approvalData\_transformed$approval, p=0.75, list=FALSE)  
approvalTrainSet <- approvalData\_transformed[index,]  
approvalTestSet <- approvalData\_transformed[-index,]  
  
#Checking the structure of approvalTrainSet  
str(approvalTrainSet)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 905 obs. of 287 variables:  
## $ URBRUR : Factor w/ 2 levels "1","2": 2 2 2 1 1 1 2 1 1 1 ...  
## $ REGION : Factor w/ 12 levels "1220","1221",..: 10 6 6 2 2 6 8 1 8 3 ...  
## $ EA\_SVC\_A : Factor w/ 2 levels "0","1": 2 1 1 2 2 2 2 2 2 2 ...  
## $ EA\_SVC\_B : Factor w/ 3 levels "0","1","9": 2 1 1 2 2 1 2 2 2 2 ...  
## $ EA\_SVC\_C : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 2 1 2 ...  
## $ EA\_SVC\_D : Factor w/ 3 levels "0","1","9": 2 2 2 2 2 2 2 2 2 2 ...  
## $ EA\_FAC\_A : Factor w/ 3 levels "0","1","9": 1 1 1 1 1 2 1 2 1 1 ...  
## $ EA\_FAC\_B : Factor w/ 3 levels "0","1","9": 2 2 2 2 2 2 2 2 2 1 ...  
## $ EA\_FAC\_C : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 2 2 1 ...  
## $ EA\_FAC\_D : Factor w/ 2 levels "0","1": 2 1 2 2 2 2 2 2 2 1 ...  
## $ EA\_FAC\_E : Factor w/ 2 levels "0","1": 2 1 2 2 2 2 2 2 2 1 ...  
## $ EA\_FAC\_F : Factor w/ 3 levels "0","1","9": 1 1 1 2 2 1 2 2 2 1 ...  
## $ EA\_FAC\_G : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 1 ...  
## $ EA\_SEC\_A : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 2 1 1 ...  
## $ EA\_SEC\_B : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...  
## $ EA\_SEC\_C : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...  
## $ EA\_SEC\_D : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...  
## $ EA\_SEC\_E : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## $ EA\_ROAD\_A: Factor w/ 5 levels "1","2","3","5",..: 1 1 1 4 4 1 4 4 4 4 ...  
## $ EA\_ROAD\_B: Factor w/ 6 levels "1","2","3","4",..: 5 3 1 5 5 1 1 5 1 5 ...  
## $ EA\_ROAD\_C: Factor w/ 6 levels "1","2","3","4",..: 4 2 2 5 5 3 2 5 3 4 ...  
## $ Q1 : num 8 34 28 9 13 13 26 13 14 15 ...  
## $ Q2A : Factor w/ 52 levels "1","2","1220",..: 13 3 50 17 7 50 37 22 26 35 ...  
## $ Q2B : Factor w/ 43 levels "1","2","1220",..: 13 3 3 2 7 3 3 2 3 3 ...  
## $ Q3 : Factor w/ 4 levels "1","2","8","9": 2 1 4 4 4 4 1 1 2 1 ...  
## $ Q4A : Factor w/ 7 levels "1","2","3","4",..: 1 5 7 2 2 4 4 2 5 2 ...  
## $ Q4B : Factor w/ 7 levels "1","2","3","4",..: 1 4 1 3 1 2 4 3 3 4 ...  
## $ Q5 : Factor w/ 7 levels "1","2","3","4",..: 2 3 7 3 3 3 4 2 4 2 ...  
## $ Q6 : Factor w/ 7 levels "1","2","3","4",..: 2 2 2 2 2 4 2 2 4 3 ...  
## $ Q7 : Factor w/ 7 levels "1","2","3","4",..: 7 2 7 7 4 3 4 4 5 7 ...  
## $ Q8A : Factor w/ 7 levels "0","1","2","3",..: 3 1 3 1 1 3 1 4 3 5 ...  
## $ Q8B : Factor w/ 6 levels "0","1","2","3",..: 3 5 4 1 3 3 4 3 3 4 ...  
## $ Q8C : Factor w/ 7 levels "0","1","2","3",..: 3 5 3 1 1 3 3 4 3 4 ...  
## $ Q8D : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 3 1 3 1 5 3 4 ...  
## $ Q8E : Factor w/ 7 levels "0","1","2","3",..: 4 1 3 3 3 3 1 5 3 5 ...  
## $ Q8F : Factor w/ 8 levels "1","2","3","4",..: 3 6 6 4 3 4 5 5 3 2 ...  
## $ Q9 : Factor w/ 6 levels "0","1","2","3",..: 1 1 1 1 3 1 4 2 1 1 ...  
## $ Q10A : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 3 1 1 3 3 1 2 ...  
## $ Q10B : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 5 1 1 1 4 1 1 ...  
## $ Q11A : Factor w/ 5 levels "0","1","2","3",..: 1 4 1 4 4 1 1 1 1 2 ...  
## $ Q11B : Factor w/ 6 levels "0","1","2","3",..: 1 1 1 2 1 1 1 1 4 1 ...  
## $ Q12A : Factor w/ 6 levels "0","1","2","3",..: 4 1 1 4 1 4 1 5 3 1 ...  
## $ Q12B : Factor w/ 7 levels "0","1","2","3",..: 5 1 1 4 1 1 5 5 3 4 ...  
## $ Q12C : Factor w/ 7 levels "0","1","2","3",..: 4 1 1 1 1 1 5 4 1 4 ...  
## $ Q12D : Factor w/ 7 levels "0","1","2","3",..: 4 1 1 4 1 1 5 4 2 3 ...  
## $ Q12E : Factor w/ 7 levels "0","1","2","3",..: 4 1 1 4 1 1 5 4 1 3 ...  
## $ Q13 : Factor w/ 5 levels "0","1","2","8",..: 1 2 1 2 3 2 2 1 1 1 ...  
## $ Q14 : Factor w/ 6 levels "1","2","3","4",..: 3 1 3 2 4 3 3 6 3 3 ...  
## $ Q15 : Factor w/ 7 levels "1","2","3","4",..: 4 4 2 4 3 1 3 3 2 2 ...  
## $ Q16 : Factor w/ 7 levels "1","2","3","4",..: 4 1 2 4 4 4 3 4 4 1 ...  
## $ Q17 : Factor w/ 7 levels "1","2","3","4",..: 5 2 2 3 4 4 3 4 4 2 ...  
## $ Q18A : Factor w/ 6 levels "0","1","2","3",..: 6 1 2 4 1 1 1 1 3 1 ...  
## $ Q18B : Factor w/ 6 levels "0","1","2","3",..: 6 1 2 4 1 6 1 3 3 3 ...  
## $ Q18C : Factor w/ 6 levels "0","1","2","3",..: 6 1 1 3 1 1 1 1 4 1 ...  
## $ Q18D : Factor w/ 6 levels "0","1","2","3",..: 6 1 6 4 6 6 3 6 3 1 ...  
## $ Q19A : Factor w/ 7 levels "1","2","3","4",..: 5 5 3 2 5 4 2 7 4 2 ...  
## $ Q19B : Factor w/ 7 levels "1","2","3","4",..: 7 5 4 5 5 2 4 7 4 5 ...  
## $ Q19C : Factor w/ 7 levels "1","2","3","4",..: 7 1 5 2 5 7 2 5 2 3 ...  
## $ Q19D : Factor w/ 7 levels "1","2","3","4",..: 7 1 7 2 5 7 2 4 3 4 ...  
## $ Q19E : Factor w/ 7 levels "1","2","3","4",..: 7 1 7 2 5 4 2 5 2 3 ...  
## $ Q20A : Factor w/ 6 levels "0","1","2","3",..: 1 3 1 1 3 4 1 1 2 3 ...  
## $ Q20B : Factor w/ 6 levels "0","1","2","3",..: 1 1 1 1 4 4 3 1 1 1 ...  
## $ Q21A : Factor w/ 7 levels "0","1","2","3",..: 4 5 7 2 5 2 2 2 2 5 ...  
## $ Q21B : Factor w/ 7 levels "0","1","2","3",..: 3 5 2 4 5 4 4 5 2 3 ...  
## $ Q22 : Factor w/ 11 levels "0","1","2","3",..: 9 6 2 9 2 2 2 10 6 2 ...  
## $ Q23 : Factor w/ 7 levels "1","2","3","4",..: 6 2 4 5 4 4 4 6 1 3 ...  
## $ Q24A : Factor w/ 4 levels "0","1","8","9": 2 1 1 1 2 1 1 1 1 2 ...  
## $ Q24B : Factor w/ 4 levels "0","1","8","9": 1 1 1 1 2 1 1 1 1 1 ...  
## $ Q25A : Factor w/ 6 levels "0","1","2","3",..: 2 1 1 1 1 1 1 1 1 1 ...  
## $ Q25B : Factor w/ 6 levels "0","1","2","3",..: 2 1 1 1 1 1 1 1 1 1 ...  
## $ Q25C : Factor w/ 6 levels "0","1","2","3",..: 1 1 1 2 1 1 1 1 1 1 ...  
## $ Q25D : Factor w/ 6 levels "0","1","2","3",..: 1 1 1 1 4 1 1 1 1 1 ...  
## $ Q25E : Factor w/ 6 levels "0","1","2","3",..: 1 1 1 2 3 1 1 1 1 4 ...  
## $ Q25F : Factor w/ 5 levels "0","1","2","3",..: 1 1 1 1 4 1 1 1 1 4 ...  
## $ Q26A : Factor w/ 7 levels "0","1","2","3",..: 2 5 2 2 2 4 1 1 2 4 ...  
## $ Q26B : Factor w/ 7 levels "0","1","2","3",..: 2 2 7 2 2 2 1 1 2 3 ...  
## $ Q26C : Factor w/ 7 levels "0","1","2","3",..: 2 5 2 3 2 2 1 5 1 2 ...  
## $ Q26D : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 1 3 2 1 7 1 1 ...  
## $ Q26E : Factor w/ 7 levels "0","1","2","3",..: 1 1 1 3 3 1 1 7 2 1 ...  
## $ Q27A : Factor w/ 7 levels "1","2","3","4",..: 7 2 2 2 5 1 5 4 1 3 ...  
## $ Q27B : Factor w/ 7 levels "1","2","3","4",..: 2 2 7 1 1 1 4 7 5 2 ...  
## $ Q27C : Factor w/ 7 levels "1","2","3","4",..: 7 2 7 1 1 2 1 2 2 2 ...  
## $ Q28 : Factor w/ 5 levels "1","2","3","8",..: 5 3 5 3 1 3 3 1 3 3 ...  
## $ Q29 : Factor w/ 7 levels "1","2","3","4",..: 7 1 5 3 2 1 1 3 3 1 ...  
## $ Q30 : Factor w/ 7 levels "1","2","3","4",..: 7 1 2 2 3 1 4 2 2 2 ...  
## $ Q31 : Factor w/ 7 levels "1","2","3","4",..: 1 3 2 3 1 1 2 3 4 2 ...  
## $ Q32 : Factor w/ 7 levels "1","2","3","4",..: 1 1 3 1 2 2 3 2 4 3 ...  
## $ Q33 : Factor w/ 7 levels "1","2","3","4",..: 4 3 3 3 4 3 1 7 3 5 ...  
## $ Q34 : Factor w/ 7 levels "1","2","3","4",..: 3 2 4 4 3 1 4 7 3 2 ...  
## $ Q35 : Factor w/ 7 levels "1","2","3","4",..: 6 2 5 2 4 3 1 3 4 4 ...  
## $ Q36 : Factor w/ 7 levels "0","1","2","3",..: 2 3 7 3 3 3 2 4 4 3 ...  
## $ Q37 : Factor w/ 7 levels "1","2","3","4",..: 1 2 3 1 1 2 3 1 2 2 ...  
## $ Q38A : Factor w/ 7 levels "1","2","3","4",..: 4 4 2 2 4 2 4 1 4 4 ...  
## $ Q38B : Factor w/ 7 levels "1","2","3","4",..: 5 4 2 2 5 4 1 1 1 4 ...  
## $ Q38C : Factor w/ 7 levels "1","2","3","4",..: 4 4 4 5 3 2 1 2 4 4 ...  
## $ Q38D : Factor w/ 7 levels "1","2","3","4",..: 5 4 2 1 2 3 2 1 4 4 ...  
## $ Q38E : Factor w/ 7 levels "1","2","3","4",..: 2 2 2 1 5 5 5 4 4 4 ...  
## $ Q38F : Factor w/ 7 levels "1","2","3","4",..: 2 2 2 1 1 4 1 5 4 2 ...  
## $ Q38G : Factor w/ 7 levels "1","2","3","4",..: 2 7 7 2 5 4 1 1 2 4 ...  
## [list output truncated]

## 4. Feature selection using Caret

The feature selection, which is a crucial part of machine learning, is made easy by Caret. The “recursive feature elimination” or “rfe” function in Caret is used to find the best subset of features to be included in the models.

# Feature selection using rfe in caret  
ctrl <- rfeControl(functions = rfFuncs,  
 method = "repeatedcv",  
 repeats = 3,  
 verbose = FALSE)  
  
y <- approvalTrainSet$approval  
x <- select(approvalTrainSet, - approval)  
  
approvalProfile <- rfe(x, y,  
 rfeControl = ctrl)  
  
approvalProfile

##   
## Recursive feature selection  
##   
## Outer resampling method: Cross-Validated (10 fold, repeated 3 times)   
##   
## Resampling performance over subset size:  
##   
## Variables Accuracy Kappa AccuracySD KappaSD Selected  
## 4 0.7116 0.4720 0.04581 0.08449   
## 8 0.7351 0.5087 0.05213 0.10030   
## 16 0.7385 0.5117 0.03820 0.07527 \*  
## 286 0.7323 0.4747 0.04116 0.08523   
##   
## The top 5 variables (out of 16):  
## Q58B, Q43A, Q58C, REGION, Q99

Results:

Recursive feature selection

Outer resampling method: Cross-Validated (10 fold, repeated 3 times)

Resampling performance over subset size:

The top 5 variables (out of 16): Q58B, Q43A, Q58C, REGION, Q99

## 5. Training models using Caret

Caret provides a large number of algorithms with similar syntax. Following Kaushik’s practical guide, we will apply the following: GBM and Random forest.

fitControl <- trainControl(method = "repeatedcv",  
 number = 5,  
 repeats = 3)  
approvalModel\_gbm <- train(approval ~ ., data = approvalTrainSet,   
 method = "gbm",   
 trControl = fitControl,  
 verbose = FALSE)  
  
approvalModel\_rf <- train(approval ~ ., data = approvalTrainSet,   
 method = "rf",   
 trControl = fitControl,  
 verbose = FALSE,  
 importance=T)

## 6. Variable importance estimation using Caret

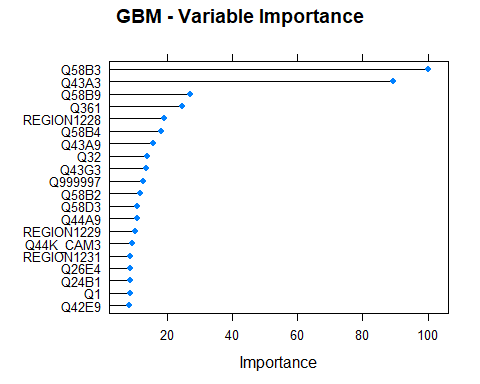
We can also check the variable importance estimates in Caret by using the “varImp” function" for any model.

### 6.1 Variable importance for GBM

# Checking variable importance for GBM  
# Variable Importance  
varImp(object=approvalModel\_gbm)

## gbm variable importance  
##   
## only 20 most important variables shown (out of 1741)  
##   
## Overall  
## Q58B3 100.000  
## Q43A3 89.277  
## Q58B9 27.196  
## Q361 24.680  
## REGION1228 19.054  
## Q58B4 18.310  
## Q43A9 15.744  
## Q32 13.854  
## Q43G3 13.661  
## Q999997 12.747  
## Q58B2 11.725  
## Q58D3 10.942  
## Q44A9 10.910  
## REGION1229 10.197  
## Q44K\_CAM3 9.452  
## REGION1231 8.850  
## Q26E4 8.719  
## Q24B1 8.716  
## Q1 8.600  
## Q42E9 8.570

#Plotting Variable importance for GBM model  
plot(varImp(object=approvalModel\_gbm),main="GBM - Variable Importance", top = 20)

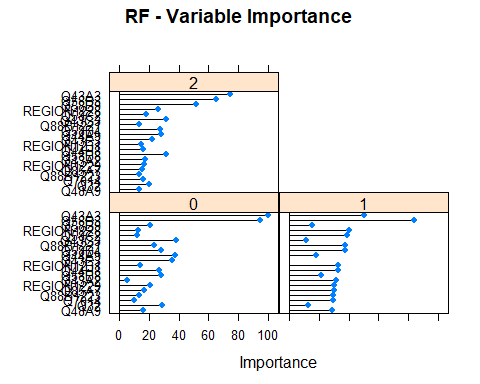


### 6.2 Variable importance for RF

#Checking variable importance for RF  
varImp(object=approvalModel\_rf)

## rf variable importance  
##   
## variables are sorted by maximum importance across the classes  
## only 20 most important variables shown (out of 1741)  
##   
## 0 1 2  
## Q43A3 100.000 50.06 74.63  
## Q58B3 94.594 83.67 65.23  
## Q58B9 20.618 15.38 51.90  
## REGION1228 12.894 40.46 26.29  
## Q58C2 11.978 39.07 18.17  
## Q43G3 38.196 11.30 31.48  
## Q88B1221 23.497 37.75 13.29  
## Q58B4 28.196 37.62 27.79  
## Q44A9 37.343 18.10 27.89  
## Q43E3 35.359 0.00 22.44  
## REGION1231 14.365 32.83 14.99  
## Q44H8 26.571 32.63 16.15  
## Q58B8 28.380 21.27 31.33  
## Q43A2 5.598 31.31 17.72  
## REGION1229 20.663 30.34 17.08  
## Q85A7 16.562 30.29 15.53  
## Q88B1223 13.479 29.56 13.30  
## Q7014 10.039 29.26 16.15  
## Q32 29.065 12.68 19.77  
## Q48A9 15.994 28.89 13.57

#rf variable importance  
  
  
#Plotting Varianle importance for Random Forest  
plot(varImp(object=approvalModel\_rf),main="RF - Variable Importance", top = 20)



## 7. Making predictions using Caret

#Predictions with gbm  
approvalPrediction\_gbm <-predict.train(object=approvalModel\_gbm,approvalTestSet,type="raw")  
table(approvalPrediction\_gbm)

## approvalPrediction\_gbm  
## 0 1 2   
## 110 173 16

confusionMatrix(approvalPrediction\_gbm,approvalTestSet$approval)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2  
## 0 72 31 7  
## 1 26 136 11  
## 2 2 1 13  
##   
## Overall Statistics  
##   
## Accuracy : 0.7391   
## 95% CI : (0.6855, 0.788)  
## No Information Rate : 0.5619   
## P-Value [Acc > NIR] : 1.666e-10   
##   
## Kappa : 0.5225   
##   
## Mcnemar's Test P-Value : 0.009096   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2  
## Sensitivity 0.7200 0.8095 0.41935  
## Specificity 0.8090 0.7176 0.98881  
## Pos Pred Value 0.6545 0.7861 0.81250  
## Neg Pred Value 0.8519 0.7460 0.93640  
## Prevalence 0.3344 0.5619 0.10368  
## Detection Rate 0.2408 0.4548 0.04348  
## Detection Prevalence 0.3679 0.5786 0.05351  
## Balanced Accuracy 0.7645 0.7635 0.70408

#Predictions with rf  
approvalPrediction\_rf <-predict.train(object=approvalModel\_rf,approvalTestSet,type="raw")  
table(approvalPrediction\_rf)

## approvalPrediction\_rf  
## 0 1 2   
## 121 162 16

confusionMatrix(approvalPrediction\_rf,approvalTestSet$approval)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2  
## 0 70 39 12  
## 1 29 127 6  
## 2 1 2 13  
##   
## Overall Statistics  
##   
## Accuracy : 0.7023   
## 95% CI : (0.647, 0.7536)  
## No Information Rate : 0.5619   
## P-Value [Acc > NIR] : 4.19e-07   
##   
## Kappa : 0.4634   
##   
## Mcnemar's Test P-Value : 0.005141   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2  
## Sensitivity 0.7000 0.7560 0.41935  
## Specificity 0.7437 0.7328 0.98881  
## Pos Pred Value 0.5785 0.7840 0.81250  
## Neg Pred Value 0.8315 0.7007 0.93640  
## Prevalence 0.3344 0.5619 0.10368  
## Detection Rate 0.2341 0.4247 0.04348  
## Detection Prevalence 0.4047 0.5418 0.05351  
## Balanced Accuracy 0.7219 0.7444 0.70408

## Conclusion