Capstone

Nelson Levy K F Macedo

08/03/2022

Introduction

This report is part of the EDX Course HarvardX: PH129.9X Data Science: Capstone. In this part, the objective is to create machine learning algorithm to work in any choosed dataset.

The dataset selected is about frauds in proccess Brazilian Social Security System. To limit the scope of the project, I choosed only one type of benefit translated for some like “age social benefit”. For this species of benefit, the person with 65 years or more who does not have minimal profit receives a monthly minimal paycheck (in Brazil in 2022 it’s about U$200,00) and there are around 820.000 active benefits in the dataset provided.

The experience in this kind of fraud (working about 15 years in Brazilian Federal Police, and now being Head of Division for Frauds against Social Security System) teaches that there are some kind of circunstances that goes toward a fraudulent proccess. Those circunstances does not show peremptorily there is a fraud but surely are not typical features. We will call those circunstances as “trails” of fraud. For instance, let’s say one person have never been in any governmental database and, from nowhere, it appears in front of a Social Security officer and asks for the benefit. It is not a fraud by itself (it can be a farmer worker who never had documents, for example), but it is a uncommon situation. Unfortunately, we can not provide all the details about the origin of the trails, but in the dataset available there 28 trails (two categorical and twenty six boolean).

Method - Dataset definitions

Among those proccess, there are a lot of proccess that were audited by human analysis. In the available data, were identified about 15.000 thousand proccess with human analysis (fraud or not fraud). The initial tests showed a problem of prevalence: in this proccess, only around 5-10% were detected as fraud.

To better balance the dataset, were included informations about benefits suspended or cancelled by fraud (and its trails). So now, the dataset has around 41.000 benefits, within 28 trail informations and a field named “Status” with the conclusion “fraud” or “not_fraud”.

For the challenge porpoused, the approach used is the ensembled. Using the knowledge provided in the course, this 41.000 proccess dataset where divided in train_set and test_set, in proportion of 0.9 to 0.1 each. Using the models available in the “caret package”, a lot of tries were made, including almost all availabe in the list here (https://rdrr.io/cran/caret/man/models.html (https://rdrr.io/cran/caret/man/models.html)).

The models used in the final version were “gamLoess”, “multinom”, “lda” and “glm”. The models “knn” and “qda” did not work with the data set. The models “adaboost”, “multinom”, “adabag”, “bglmAjCat”, “svmLinear” and “rf”, for instance, took a very long time to complete the train and did not show results much better than the already mentioned models. The naive_bayes model predicted all the results as fraud (it has accuracy based on the prevalence of the dataset) and will not be used from now on.

Initially, check if is necessary to install and/or load libraries:

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Carregando pacotes exigidos: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Carregando pacotes exigidos: caret
```

```
## Carregando pacotes exigidos: lattice
```

```
## 
## Attaching package: 'caret'
```

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

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Carregando pacotes exigidos: data.table
```

```
## 
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
## 
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
## 
##     transpose
```

```
if(!require(readxl)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Carregando pacotes exigidos: readxl
```

```
if(!require(writexl)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Carregando pacotes exigidos: writexl
```

```
if(!require(dslabs)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Carregando pacotes exigidos: dslabs
```

```
library(tidyverse)
library(caret)
library(data.table)
library(readxl)
library(writexl)
library(dslabs)
```

Analysis

The first step is to load the data. The dataset is available in my GitHub account (https://github.com/nelsonlevy/Capstone (https://github.com/nelsonlevy/Capstone)), file “Fraud_base.xlsx”. After that, data should be divided in train_set and test_set. Here, I divided in 0.9/0.1 as follow:

```
#Read initial dataset as provided
data <- as.data.frame(read_xlsx(file.choose(), sheet=1, progress = readxl_progress()))

#alternative source for data file
#data <- tempfile()
#download.file("https://github.com/nelsonlevy/Capstone/blob/main/Fraud_base.xlsx", data)



#define set.seed
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
#divides in train_set and test_set
sample_size  <- floor(0.9 * nrow(data))
train_index <- sample(seq_len(nrow(data)), size=sample_size)
train_set <- data[train_index,]  #90% of observations
test_set <- data[-train_index,]  #10% of observations
```

Then I define the models to be trained and fit. After the fit, the columns receive the name of the models:

```
#define the models for training
models <- c("gamLoess", "multinom", "lda", "glm")

#Fitting the models
fits <- lapply(models, function(model){
  print(model)
  train(Status ~ ., method = model, data = train_set)
})
```

```
## [1] "gamLoess"
```

```
## Carregando pacotes exigidos: gam
```

```
## Carregando pacotes exigidos: splines
```

```
## Carregando pacotes exigidos: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
names(fits) <- models
```

To check, we can see the dimensions of the predictions and the accuracy of the models:

```
#Register the predictions
pred <- sapply(fits, function(object)
  predict(object, newdata = test_set))
dim(pred)
```

```
## [1] 4154    4
```

```
#Accuracy analysis
acc <- colMeans(pred == test_set$Status)
acc
```

```
##  gamLoess  multinom       lda       glm
## 0.8781897 0.8921521 0.8868560 0.8921521
```

```
mean(acc)
```

```
## [1] 0.8873375
```

After that, I compute the votes for each model. After tests, the ensembled with the best results where found when there are at least two votes for “Fraud”, with better accuracy than every isolated model:

```
#compute the votes
votes <- rowMeans(pred == "Fraud")

#if there are at least two votes for "Fraud", compute as "Fraud"; else, compute as "Not_Fraud"
y_hat <- ifelse(votes > 0.49, "Fraud", "Not_Fraud")
mean(y_hat == test_set$Status)
```

```
## [1] 0.8923929
```

The results of the ensembled can be organized as follows:

```
# organize the data in a matrix
table(predicted = y_hat, actual = test_set$Status)
```

```
##            actual
## predicted   Fraud Not_Fraud
##   Fraud      2658       142
##   Not_Fraud   305      1049
```

```r
# ensembled accuracy
test_set %>%
  mutate(y_hat = y_hat) %>%
  summarize(accuracy = mean(y_hat == Status))
```

accuracy <dbl>
0.8923929

1 row

The parameters can be described like this:

```r
# recall
recall <- 2658/(2658+305)
recall
```

```
## [1] 0.8970638
```

```r
#TNR
TNR <- 1049/(1049+142)
TNR
```

```
## [1] 0.8807725
```

```r
#Precision
precision <- 2658/(2658+142)
precision
```

```
## [1] 0.9492857
```

```r
#F1 Score
F1 <- 2/((precision+recall)/(precision*recall))
F1
```

```
## [1] 0.9224362
```

```r
# ensembled accuracy
test_set %>%
  mutate(y_hat = y_hat) %>%
  summarize(accuracy = mean(y_hat == Status))
```

accuracy <dbl>
0.8923929

1 row

Results

The machine learning algorithm here described has around 89.2% of accuracy, but with 94.9% of precision and 92.2% of F1 Score.

It is not part of the original project, but there is a plus: with the models fit, I can make prediction for all the rest of data (820.000 total minus 41.000 in dataset). To make easier to calculate, I create a dataset with the unique combinations per row (available in my GitHub account, file "distinct.xlsx"):

```
#Testing in the rest of the dataset
data_all <- as.data.frame(read_xlsx(file.choose(), sheet=1, progress = readxl_progress()))

#alternative source for data_all file
#data <- tempfile()
#download.file("https://github.com/nelsonlevy/Capstone/blob/main/distinct.xlsx", data)



pred2 <- sapply(fits, function(object)
  predict(object, newdata = data_all))
dim(pred2)
```

```
## [1] 21819     4
```

The prediction for the rest of the dataset provided a list of around 217.000 benefits predicted as "Fraud". The list with the predicted Fraud is now with the Brazilian Federal Police authorities for further investigation. If it is confirmed in the real proccesses that there are 89.7% of recall, that will mean a loss of about US$ 5 billion dollars in fraud along the last 10 years.

Conclusion

The objective was to create a ensembled algorithm to predict fraud in brazilian social security system benefits. The algorithm designed was able to detect fraud with 94.9% of precision. In the name of Brazilian people I am very grateful to the EDX and HarvardX team wich provided knowledge to perform so usefull research

References

Irizary, R. (2022) Introduction to Data Science, Data analysis and prediction algorithms with R. Available in https://rafalab.github.io/dsbook/ (https://rafalab.github.io/dsbook/).