Customers Churn for Banks

Florent Rudel Ndeffo

November 5, 2020

Abstract

In this project we aim at predicting whether a customer of a bank can leave or not using random forest with the following steps:

- Import a bank customer dataset.
- Display an overview of the bank dataset which has about 10000 rows.
- Setup Spark context for computations.
- Setup the random seed for sampling. The dataset will be split into 80% for training and 20% for testing.
- Use default R random forest algorithm and then the one coming from Spark ML libraries.
- Compare the computational time of the 2 procedure using a plot.
- Extract the important variables using Random Forest from Spark.
- Determine model effectivness and predicting power by computing training error and testing error.

Introduction

We live in an era where technology can more than ever, help us to compute and train complex model to predict outputs when having specific inputs for various dataset. However, the efficient of a predicting model is directly related to the type of output and the number of outcomes. For example, if you want to determine an output that has only 2 possible outcomes therefore you might want to use a logistic regression algorithm. We do not use a logistic regression algorithm in this analysis because usually the algorithm fails to capture when they are many parameters influencing an outcome.

Step 1: Read customers dataset' file and load the required libraries

```
# Load libraries
# dplyr is use to break complex formula into simpler piece of code
# more understandable
library(dplyr)
```

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

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
# Spark to leverage large datasets
library(sparklyr)
```

```
## Warning: package 'sparklyr' was built under R version 3.6.3
```

```r
# ggplot to plot graphs
library(ggplot2)
```

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

```r
library(cowplot)
```

```
## Warning: package 'cowplot' was built under R version 3.6.3
```

```
##
## ********************************************************
```

```
## Note: As of version 1.0.0, cowplot does not change the
```

```
##   default ggplot2 theme anymore. To recover the previous
```

```
##   behavior, execute:
##   theme_set(theme_cowplot())
```

```
## ********************************************************
```

```r
# Load random forest algorithm and other functions
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.3
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
# Caret is used to predict the customers in the testing set
library(caret)
```

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

```
## Loading required package: lattice
```

```
# DataExplorer creates a preview of the dataset
library(DataExplorer)
```

```
## Warning: package 'DataExplorer' was built under R version 3.6.3
```

```
# knitr is to display tables
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.6.3
```

```r
library(flextable)
```

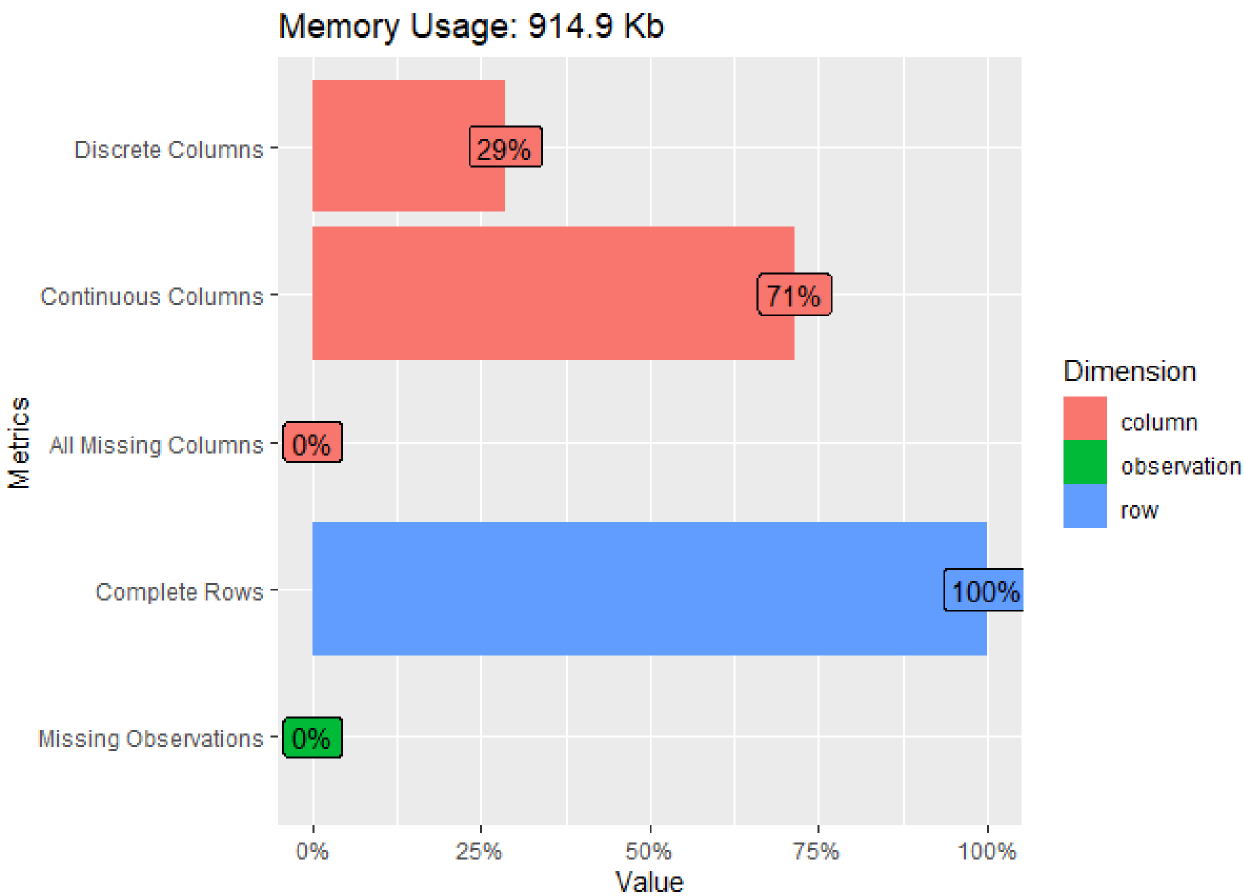
```
## Warning: package 'flextable' was built under R version 3.6.3
```

```r
# Read file
bank_customers <- read.csv("C:/Users/ndeff/OneDrive/Desktop/Research and online
          classes/Github/Data Science/Random Forest/Churn_Modelling.csv",
                       sep=",",stringsAsFactors = FALSE, header=T,na.strings=c(
         "#NUM!,","?"))
# Convert Exited column from continous to discrete value
bank_customers[,14]<- as.factor(bank_customers[,14])
# number of rows of car dataset
n=nrow(bank_customers)
```

Step 2: Show an overview and a sample of the dataset

Overview

```r
plot_intro(bank_customers)
```



Sample

```r
table = flextable(bank_customers[1:5,],col_keys = names(bank_customers),
          cwidth = 0.75, cheight = 1, theme_fun = theme_tron_legacy)
# Add color to text and the body of the table
table <- color(table,color = "black", part = "body")
table <- bg(table,bg = "white")
table
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

Step 3: Seting up Apache Spark

Setting up Spark follows the substeps below:

- Create a spark config variable that will contain all the details about how ressources we want Spark to use henceforth
- Since I am using Spark locally I am able to allocate the number of cores and how much RAM will be used

```
conf <- spark_config()
conf$`sparklyr.cores.local` <- 2
conf$`sparklyr.shell.driver-memory` <- "2G"
```

- Then create and connect a Spark cluster that will have the config above with this command

```
system.time( sc <- spark_connect(master = "local",version = "2.1.0",
                                 config = conf))
```

```
##    user  system elapsed
##    0.14    0.07    7.96
```

Step 4: Split the dataset into 80% training and 20% testing

```
#80% train sample
train_sample = sample(1:n, floor(n*0.80))
train_data = bank_customers[train_sample, ]
test_data = bank_customers[-train_sample, ]
```

Copy the training dataset into the Spark Cluster for faster computations and create a reference to it locally

```
customer_data <- copy_to(sc, train_data)
```

Step 5: Train the model using Spark and default R ML libraries

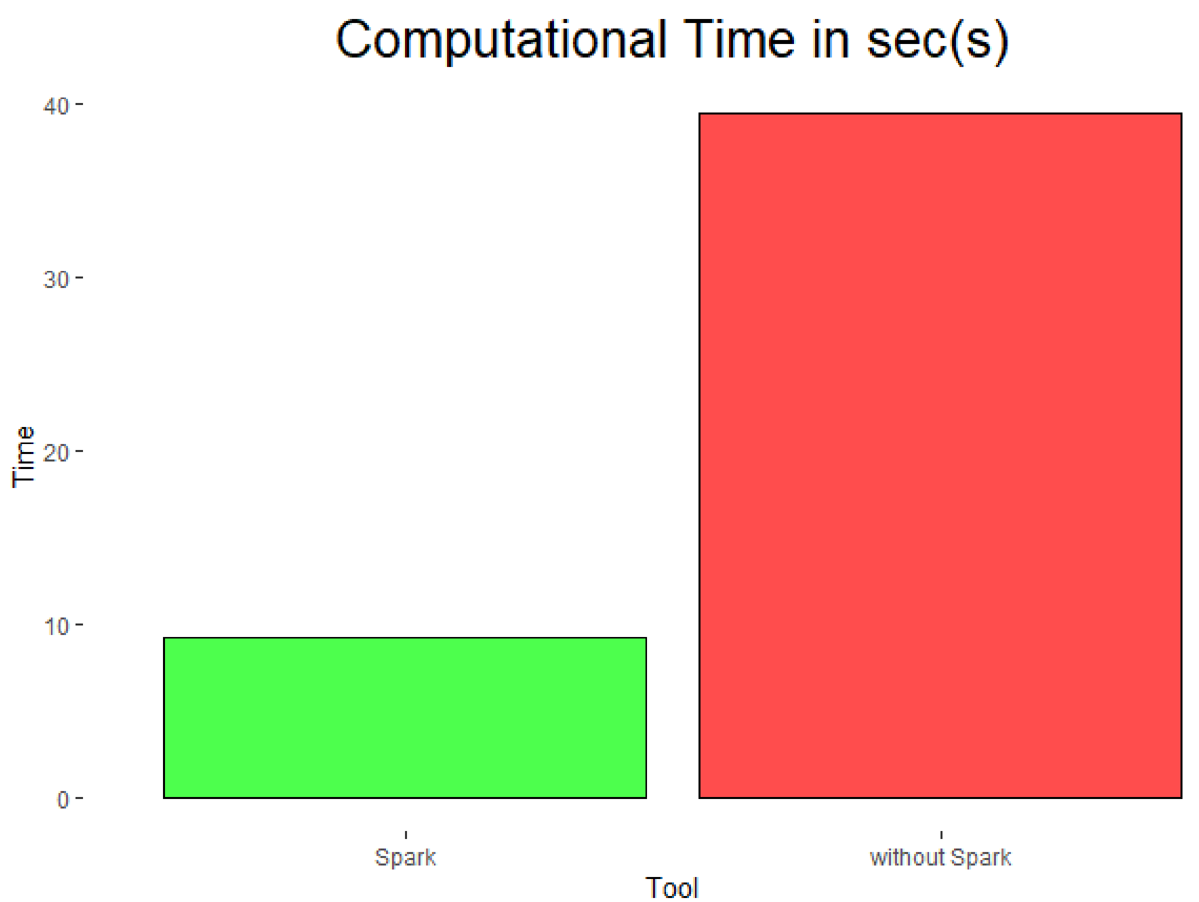
```
# Training time without using spark
training_time <- system.time(rf <- randomForest(Exited ~ CreditScore+ Age+ Tenure+ Balance+ NumOfProducts+
                HasCrCard+ IsActiveMember+ EstimatedSalary, data = train_data,
             type = classification, importance = TRUE, proximity = TRUE))

# Training model using randomForest from spark Machine learning library
spark_training_time <- system.time(rf_spark <- ml_random_forest_classifier(customer_data,Exited ~ CreditScore+ Age+ Tenure+ Balance+ NumOfProducts+ HasCrCard+ IsActiveMember+ EstimatedSalary, num_trees = 500))
```

Compare the computational time using Spark and without and plot the results

```
# Compare the training time when using spark and without using spark
names(training_time) <-c("User", "Sys","without Spark")
names(spark_training_time) <- c("User", "Sys","Spark")
df <- as.data.frame(c(training_time[3], spark_training_time[3]))
names(df) <- "Time"
Tool<- row.names(df)
ggplot(df)+ aes(x = Tool,y= Time)+
  geom_bar(colour = "black",  fill= c("green", "red"),
          alpha= 0.7, stat = "summary", show.legend = TRUE)+
```

```
  theme(plot.title = element_text(size = 20, hjust = 0.5), panel.background = el
        ement_rect("white"))+
  ggtitle("Computational Time in sec(s)")
```

```
## No summary function supplied, defaulting to `mean_se()`
```



Step 6: Determine important variables in predicting when a customer leave or does not leave the bank

```
ml_feature_importances(rf_spark)
```

```
##             feature  importance
## 1               Age 0.458723762
```

```
## 2   NumOfProducts 0.348665409
## 3  IsActiveMember 0.112492771
## 4         Balance 0.046852041
## 5     CreditScore 0.012131751
## 6 EstimatedSalary 0.010625361
## 7          Tenure 0.008894672
## 8       HasCrCard 0.001614232
```

3 Majors factors to accurately predict according to the model if a customer will leave the bank are:

- **Age** counts for about **44.5%**
- **Number of products** use by the customer counts for **35.87%**
- **Is the member active or not** influence the model for about **11.84%**

Steps 7: Determining the model effectiveness by computing the training and testing error

```
train_predict <- predict(rf, train_data)
test_predict  <- predict(rf, test_data)
```

Computing the training error

```
rdmFor_training_error <- sum(train_predict!= train_data$Exited)/nrow(train_data)
rdmFor_training_error
```

```
## [1] 0.056
```

The **training error** is about **5.125%**

Computing the testing error

```
rdmFor_testing_error <- sum(test_predict!= test_data$Exited)/nrow(test_data)
rdmFor_testing_error
```

```
## [1] 0.134
```

The **testing error** is about **14.6%**

Conclusion

In this analysis, we discover that the major factors to consider when trying to determine which costumers are more likely to leave. We found out that **age**, **number of products used** and **is the customer active** are extremely important. **Age, number of products used and is a customer active** influenced the model respectively with **44.5%, 35.8% and 11.8%**.