# Portugese Bank Marketing Calls

Le Thuc Anh

6/26/2022

## Contents

# 1 Packages installation & dataset import

Requring packages and setting options in R

```
# Require packages
Packages <- c("data.table", "tidyverse", "ggplot2", "caret", "gridExtra",
              "rpart", "randomForest", "rattle", "AUC")
# Print non-scientific number with maximum 4 signifcant digits
options("scipen" = 999, "digits" = 4)
```

```
# Marketing Portugese Bank dataset:
# https://archive.ics.uci.edu/ml/datasets/bank+marketing
# https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank.zip
# https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip
```

# 2 Introduction

The data is related with direct marketing campaigns of a **Portuguese banking institution**. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be (**yes**) subscribed or not (**no**) subscribed.

The classification goal is to predict if the client will **subscribe a term deposit** (variable $y$).

Let's see the details of **Portugese bank dataset** by using functions below:

```r
str(bank_full)
```

```
## Classes 'data.table' and 'data.frame':   41188 obs. of  21 variables:
##  $ age           : int  56 57 37 40 56 45 59 41 24 25 ...
##  $ job           : chr  "housemaid" "services" "services" "admin." ...
##  $ marital       : chr  "married" "married" "married" "married" ...
##  $ education     : chr  "basic.4y" "high.school" "high.school" "basic.6y" ...
##  $ default       : chr  "no" "unknown" "no" "no" ...
##  $ housing       : chr  "no" "no" "yes" "no" ...
##  $ loan          : chr  "no" "no" "no" "no" ...
##  $ contact       : chr  "telephone" "telephone" "telephone" "telephone" ...
##  $ month         : chr  "may" "may" "may" "may" ...
##  $ day_of_week   : chr  "mon" "mon" "mon" "mon" ...
##  $ duration      : int  261 149 226 151 307 198 139 217 380 50 ...
##  $ campaign      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays         : int  999 999 999 999 999 999 999 999 999 999 ...
##  $ previous      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome      : chr  "nonexistent" "nonexistent" "nonexistent" "nonexistent" ...
##  $ emp_var_rate  : num  1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...
##  $ cons_price_idx: num  94 94 94 94 94 ...
##  $ cons_conf_idx : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...
##  $ euribor3m     : num  4.86 4.86 4.86 4.86 4.86 ...
##  $ nr_employed   : num  5191 5191 5191 5191 5191 ...
##  $ y             : chr  "no" "no" "no" "no" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

The **Portugese bank dataset** has 21 columns and 41188 rows. There are *41188 contacted times* and *20 features of the customers*:
- 11 client-related features

1 - age (numeric)

2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')

3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)

4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate', 'professional.course','university.degree','unknown')

5 - default: has credit in default? (categorical: 'no','yes','unknown')

6 - housing: has housing loan? (categorical: 'no','yes','unknown')

7 - loan: has personal loan? (categorical: 'no','yes','unknown')

Related with the last contact of the current campaign:

8 - contact: contact communication type (categorical: 'cellular','telephone')

9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', …, 'nov', 'dec')

10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')

11 - duration: last contact duration, in seconds (numeric).

- 4 other features

12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)

14 - previous: number of contacts performed before this campaign and for this client (numeric)

15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

- 5 social and economic features

16 - emp_var_rate: employment variation rate - quarterly indicator (numeric)
17 - cons_price_idx: consumer price index - monthly indicator (numeric)

18 - cons_conf_idx: consumer confidence index - monthly indicator (numeric)
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)

20 - nr_employed: number of employees - quarterly indicator (numeric)

So, this project will focus on **predict whether one contact in the dataset resulted in a bank term deposit or not**.

# 3 Data cleaning

Let's evaluate how clean the dataset is:

```
sum(is.na(bank_full)) # NA or null values
```

```
## [1] 0
```

There are no missing values in the dataset.

Let's standardize the class of each features in the dataset.

```
fn.change.class <- function(dt, datecol=NULL, factorcol=NULL){
        strTmp <- which(sapply(dt, class) != "character")
        dt[, (strTmp):=lapply(.SD, as.numeric), .SDcols=strTmp]
        if (length(datecol)){
                dt[, (datecol):=lapply(.SD,
                                       function(x){
                                         as.Date(trunc(as.POSIXct(x, origin="1970-01-01"), "day"))}),
                   .SDcols=datecol]
        }
        if (length(factorcol)){
                dt[, (factorcol):=lapply(.SD, as.factor), .SDcols=factorcol]
        }
        return(dt)
}
bank_full = fn.change.class(bank_full)
str(bank_full)
```

```
## Classes 'data.table' and 'data.frame':   41188 obs. of  21 variables:
##  $ age           : num  56 57 37 40 56 45 59 41 24 25 ...
```

```
##  $ job           : chr  "housemaid" "services" "services" "admin." ...
##  $ marital       : chr  "married" "married" "married" "married" ...
##  $ education     : chr  "basic.4y" "high.school" "high.school" "basic.6y" ...
##  $ default       : chr  "no" "unknown" "no" "no" ...
##  $ housing       : chr  "no" "no" "yes" "no" ...
##  $ loan          : chr  "no" "no" "no" "no" ...
##  $ contact       : chr  "telephone" "telephone" "telephone" "telephone" ...
##  $ month         : chr  "may" "may" "may" "may" ...
##  $ day_of_week   : chr  "mon" "mon" "mon" "mon" ...
##  $ duration      : num  261 149 226 151 307 198 139 217 380 50 ...
##  $ campaign      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays         : num  999 999 999 999 999 999 999 999 999 999 ...
##  $ previous      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome      : chr  "nonexistent" "nonexistent" "nonexistent" "nonexistent" ...
##  $ emp_var_rate  : num  1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...
##  $ cons_price_idx: num  94 94 94 94 94 ...
##  $ cons_conf_idx : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...
##  $ euribor3m     : num  4.86 4.86 4.86 4.86 4.86 ...
##  $ nr_employed   : num  5191 5191 5191 5191 5191 ...
##  $ y             : chr  "no" "no" "no" "no" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Let's check the correlation between numeric features of the dataset.

```
strTmp <- which(sapply(bank_full, class) == "numeric")
bank_full_num = bank_full[, .SD, .SDcols=strTmp]
cor_bankfull = cor(bank_full_num)
cor_bankfull>= 0.8
```

```
##                  age duration campaign pdays previous emp_var_rate
## age             TRUE    FALSE    FALSE FALSE    FALSE        FALSE
## duration       FALSE     TRUE    FALSE FALSE    FALSE        FALSE
## campaign       FALSE    FALSE     TRUE FALSE    FALSE        FALSE
## pdays          FALSE    FALSE    FALSE  TRUE    FALSE        FALSE
## previous       FALSE    FALSE    FALSE FALSE     TRUE        FALSE
## emp_var_rate   FALSE    FALSE    FALSE FALSE    FALSE         TRUE
## cons_price_idx FALSE    FALSE    FALSE FALSE    FALSE        FALSE
## cons_conf_idx  FALSE    FALSE    FALSE FALSE    FALSE        FALSE
## euribor3m      FALSE    FALSE    FALSE FALSE    FALSE         TRUE
## nr_employed    FALSE    FALSE    FALSE FALSE    FALSE         TRUE
##                cons_price_idx cons_conf_idx euribor3m nr_employed
## age                     FALSE         FALSE     FALSE       FALSE
## duration                FALSE         FALSE     FALSE       FALSE
## campaign                FALSE         FALSE     FALSE       FALSE
## pdays                   FALSE         FALSE     FALSE       FALSE
## previous                FALSE         FALSE     FALSE       FALSE
## emp_var_rate            FALSE         FALSE      TRUE        TRUE
## cons_price_idx           TRUE         FALSE     FALSE       FALSE
## cons_conf_idx           FALSE          TRUE     FALSE       FALSE
## euribor3m               FALSE         FALSE      TRUE        TRUE
## nr_employed             FALSE         FALSE      TRUE        TRUE
```

We see that *euribor3m* and *nr_employed* is highly correlated. So that, we will remove *euribor3m*.

```r
bank_full$euribor3m = NULL
```

# 4 Data Exploration

Once again, here are the summary of the *Portugese Bank Dataset*:

```r
summary(bank_full)
```

```
##       age            job              marital            education
##  Min.   :17   Length:41188       Length:41188       Length:41188
##  1st Qu.:32   Class :character   Class :character   Class :character
##  Median :38   Mode  :character   Mode  :character   Mode  :character
##  Mean   :40
##  3rd Qu.:47
##  Max.   :98
##    default            housing              loan              contact
##  Length:41188       Length:41188       Length:41188       Length:41188
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##     month            day_of_week          duration       campaign
##  Length:41188       Length:41188       Min.   :   0   Min.   : 1.00
##  Class :character   Class :character   1st Qu.: 102   1st Qu.: 1.00
##  Mode  :character   Mode  :character   Median : 180   Median : 2.00
##                                        Mean   : 258   Mean   : 2.57
##                                        3rd Qu.: 319   3rd Qu.: 3.00
##                                        Max.   :4918   Max.   :56.00
##      pdays        previous       poutcome          emp_var_rate
##  Min.   :  0   Min.   :0.000   Length:41188       Min.   :-3.400
##  1st Qu.:999   1st Qu.:0.000   Class :character   1st Qu.:-1.800
##  Median :999   Median :0.000   Mode  :character   Median : 1.100
##  Mean   :962   Mean   :0.173                      Mean   : 0.082
##  3rd Qu.:999   3rd Qu.:0.000                      3rd Qu.: 1.400
##  Max.   :999   Max.   :7.000                      Max.   : 1.400
##  cons_price_idx cons_conf_idx    nr_employed         y
##  Min.   :92.2   Min.   :-50.8   Min.   :4964   Length:41188
##  1st Qu.:93.1   1st Qu.:-42.7   1st Qu.:5099   Class :character
##  Median :93.8   Median :-41.8   Median :5191   Mode  :character
##  Mean   :93.6   Mean   :-40.5   Mean   :5167
##  3rd Qu.:94.0   3rd Qu.:-36.4   3rd Qu.:5228
##  Max.   :94.8   Max.   :-26.9   Max.   :5228
```

And, let's check how imbalance the dataset is:

```r
no_y = bank_full[, .N , by = y]
no_y[, perc_N := (N/ sum(N))*100]
no_y
```

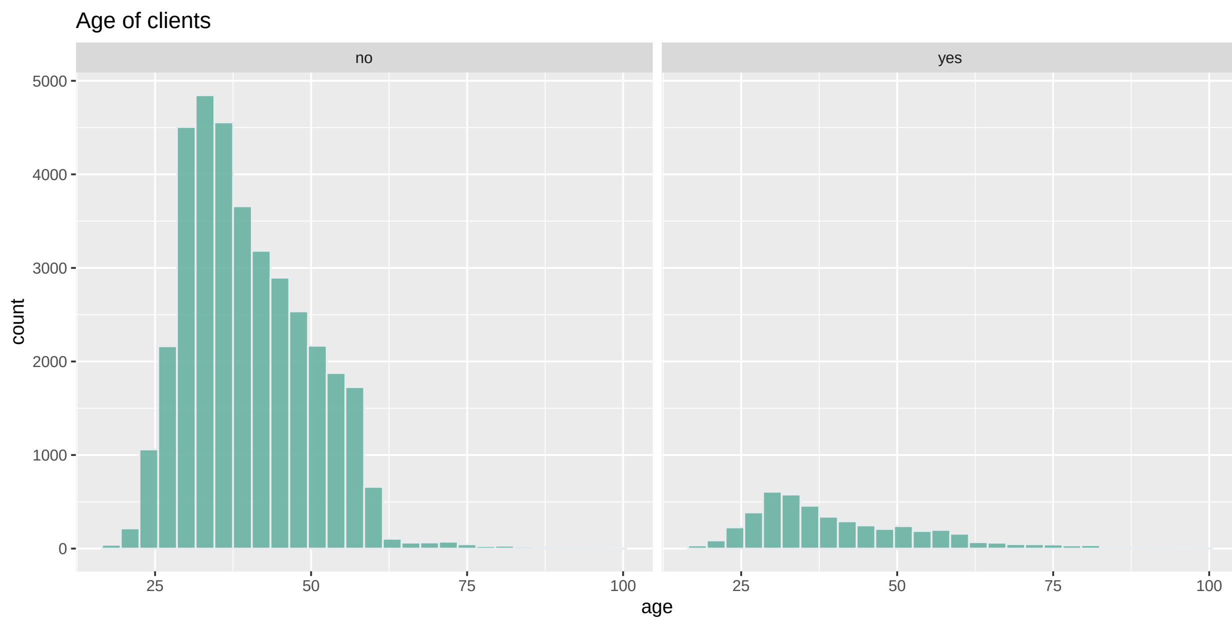
```
##      y     N perc_N
## 1:  no 36548  88.73
## 2: yes  4640  11.27
```

The dataset is skewed towards **no** with 88.73% of the contacts are resulted in **no** answer to bank term deposit. Which resulted in a ratio of **yes**:**no** as 1:7.8.

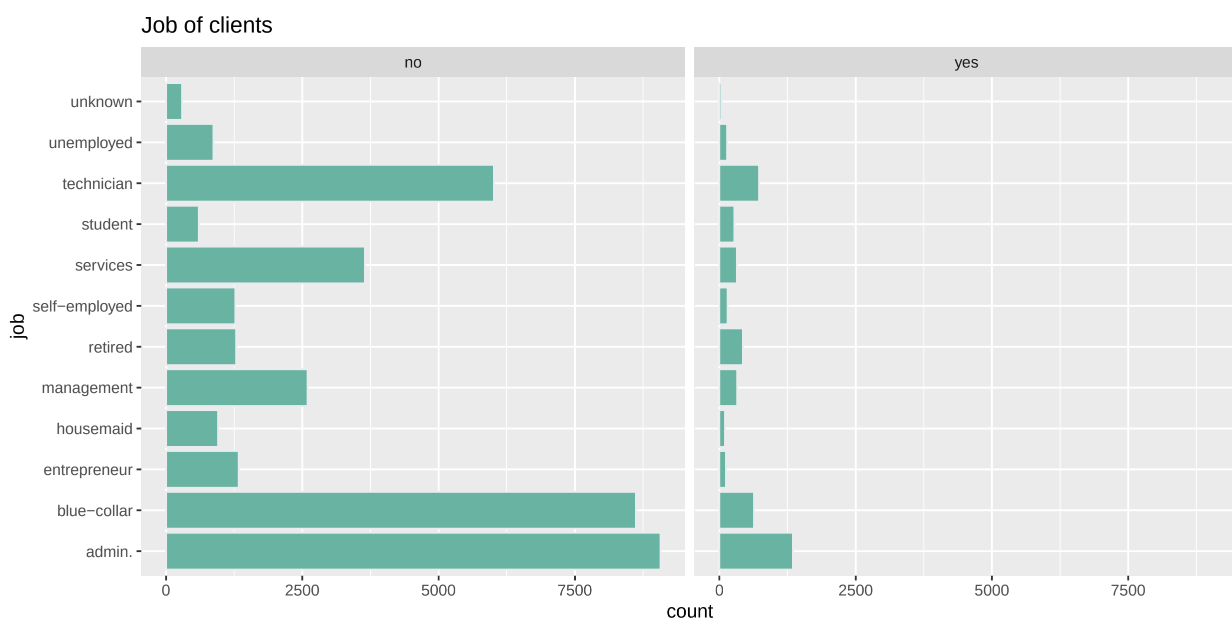
So, there is no need of further fixing to leverage the lesser group.

## 4.1 Demographic

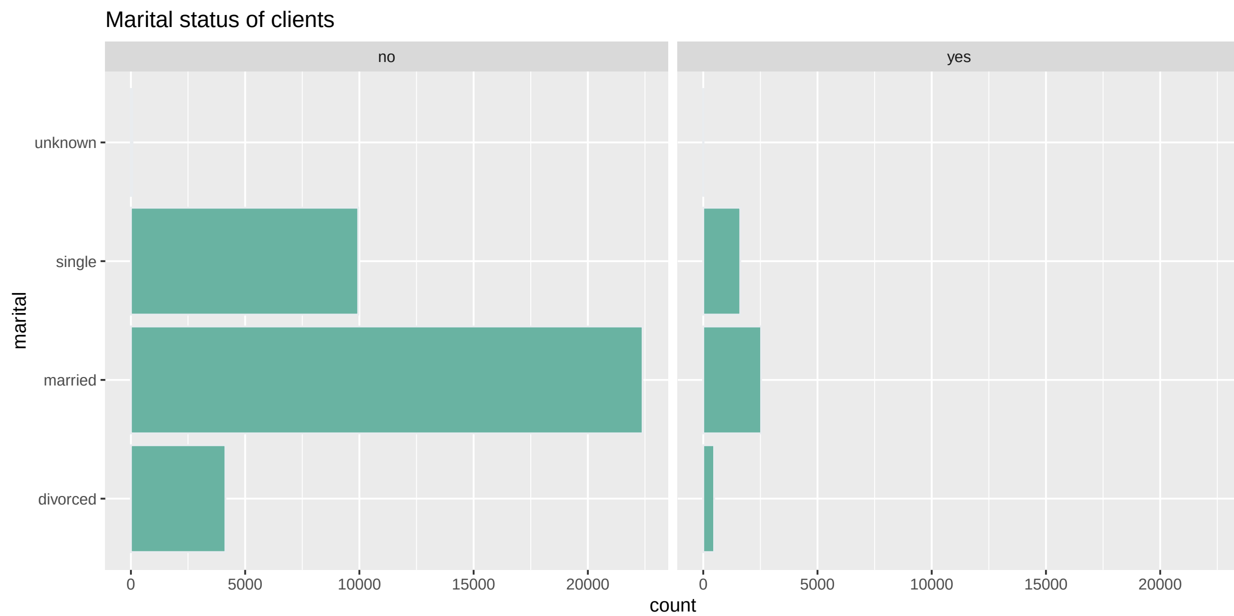
Let's explore how different in demographic for each group of subcription **yes** and **no**



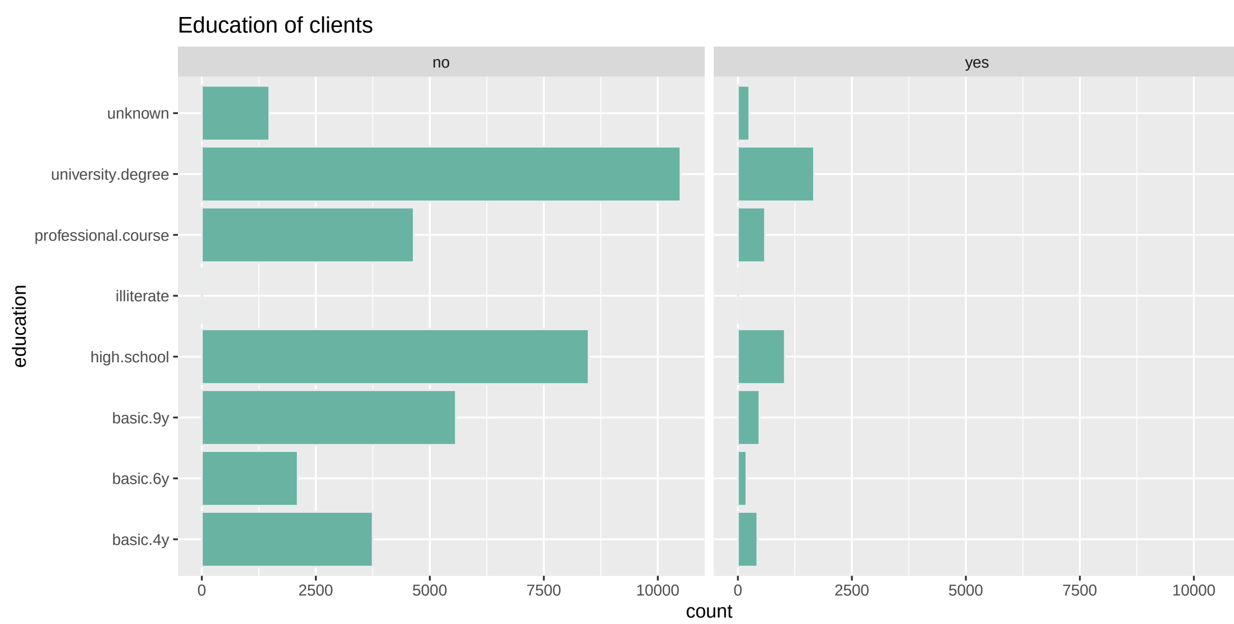

By observing this barplot, we could see that the age for both groups are quite similar with the highest number at around 25 to 40 years old.



For **no** group, the two dominant job title is *blue-collar* and *admin.*. While for **yes** group, the dominants also see in *technician* and following up is *retired* group.

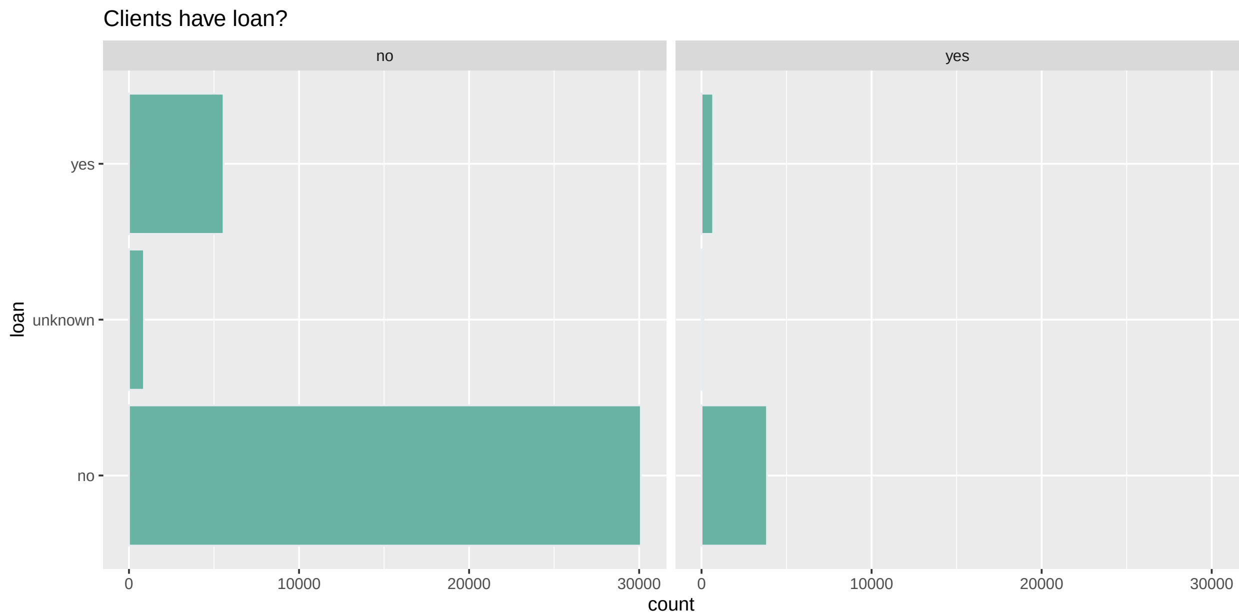



The class *unknown* is extremely small indicate that our data completeness is quite good and that this data could be trusted.
The group *married* is the most popular for both **yes** and **no** group.

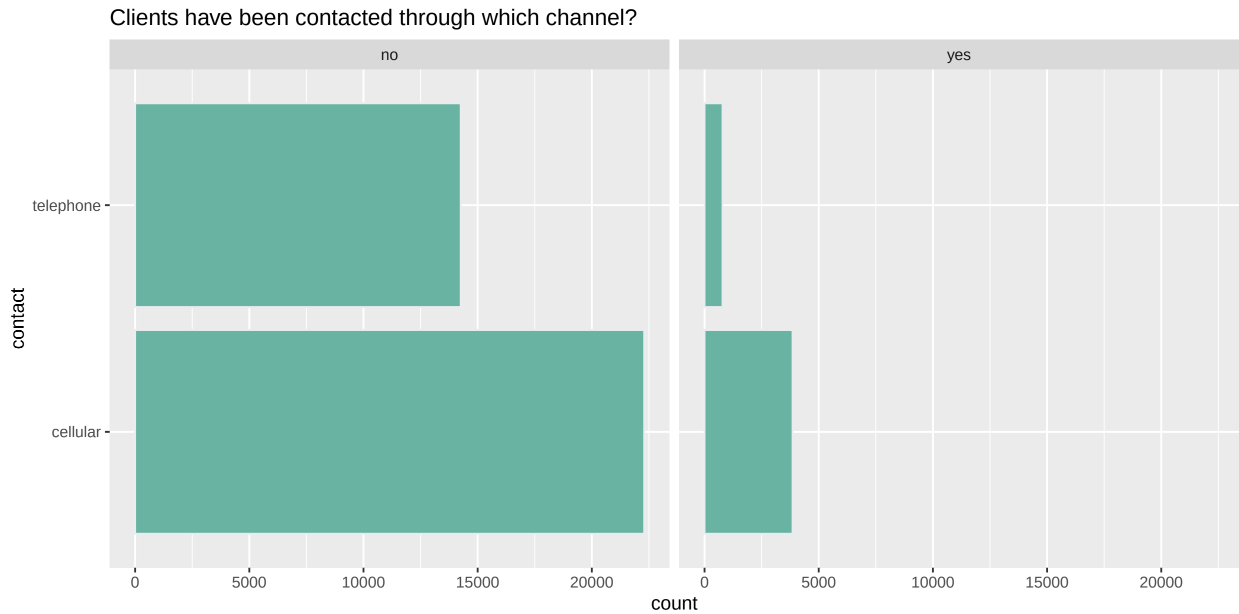



The most popular are *university.degree* and *high.school* for both **yes** and **no**. However, in **yes** group, we could see the rising of *basic.4y* as compared to the second popular group *professional.course* and *basic.9y*.

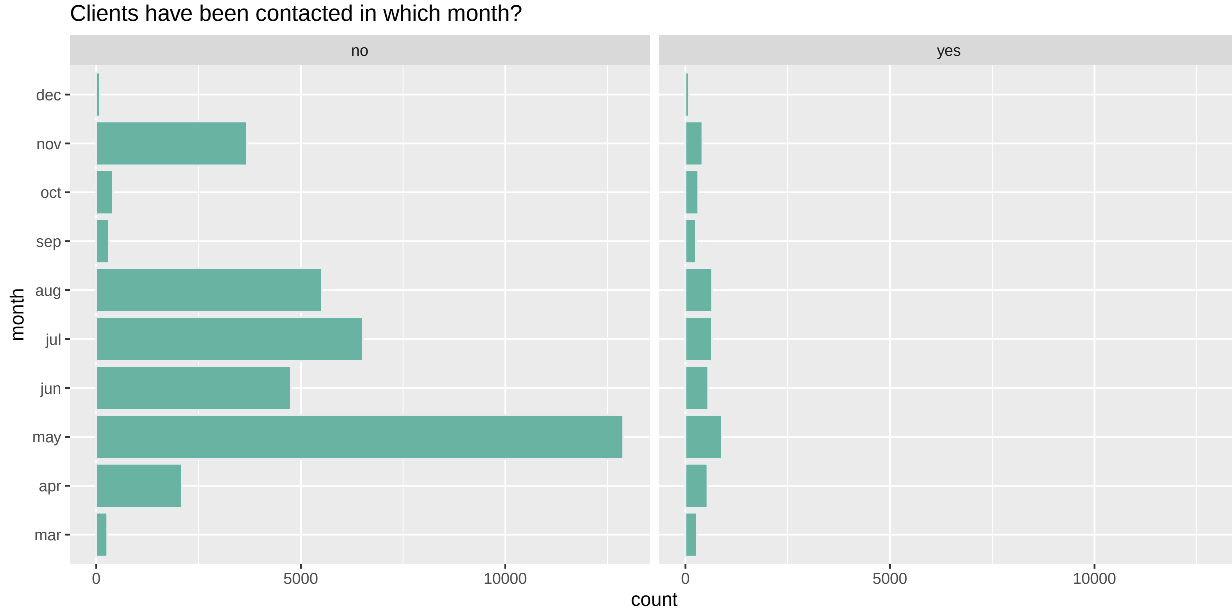
## 4.2 Bank-related features



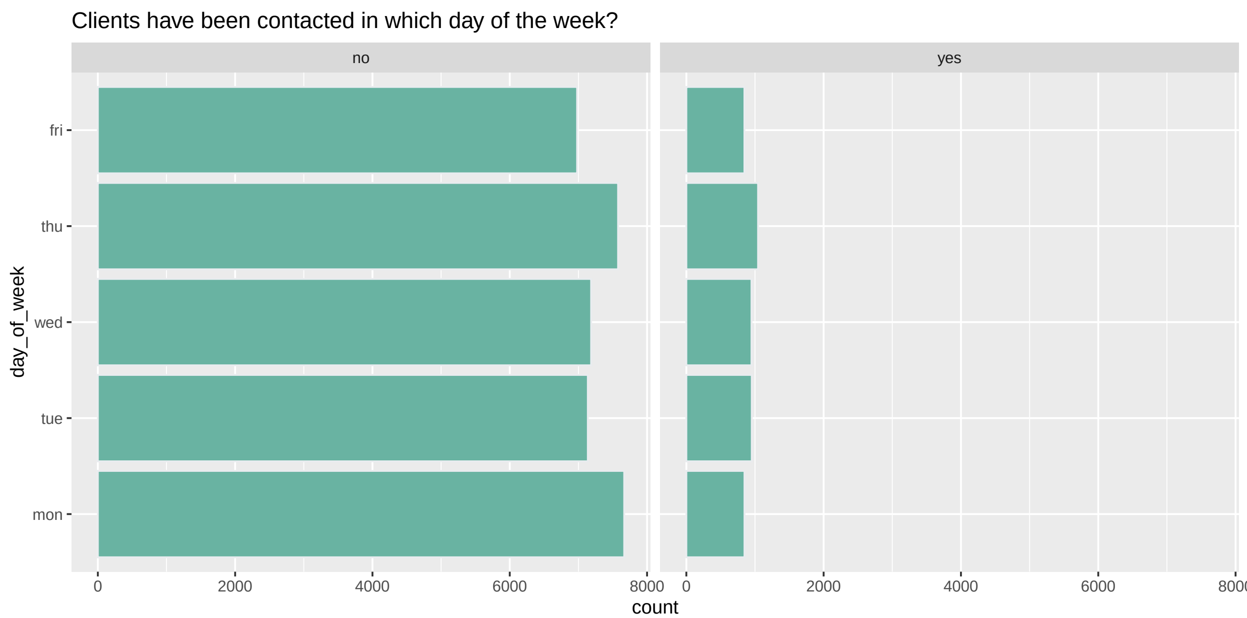

The proportion of having loan and not having loan for **yes** group and **no** group are similar.



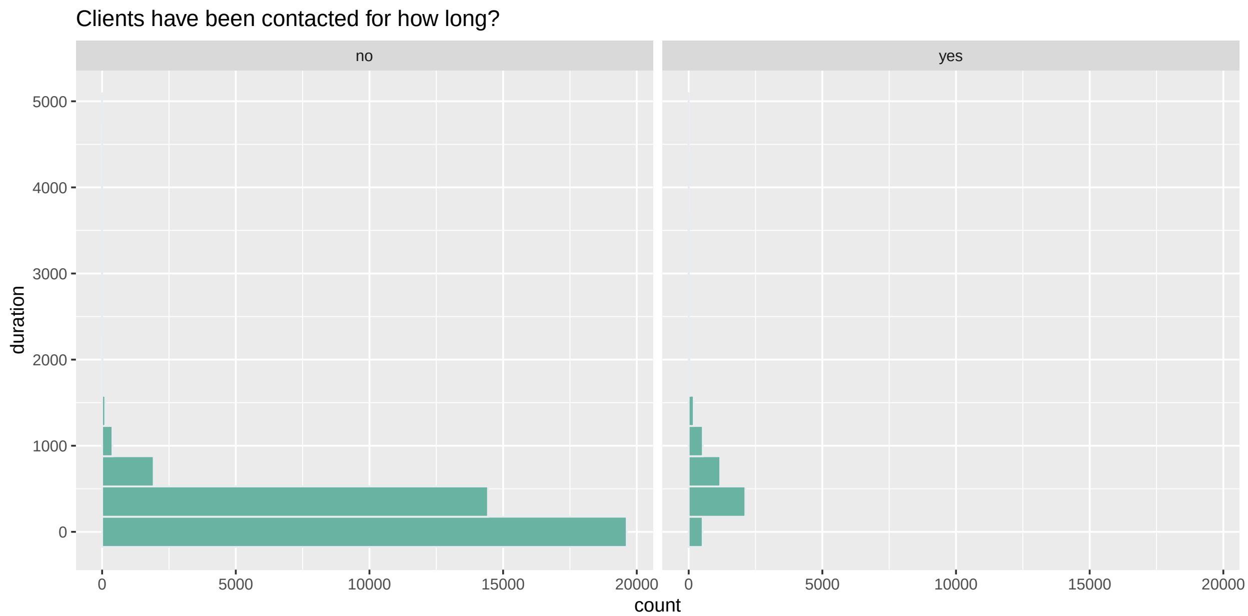

Clients are being contacted through *cellular* more than *telephone*.



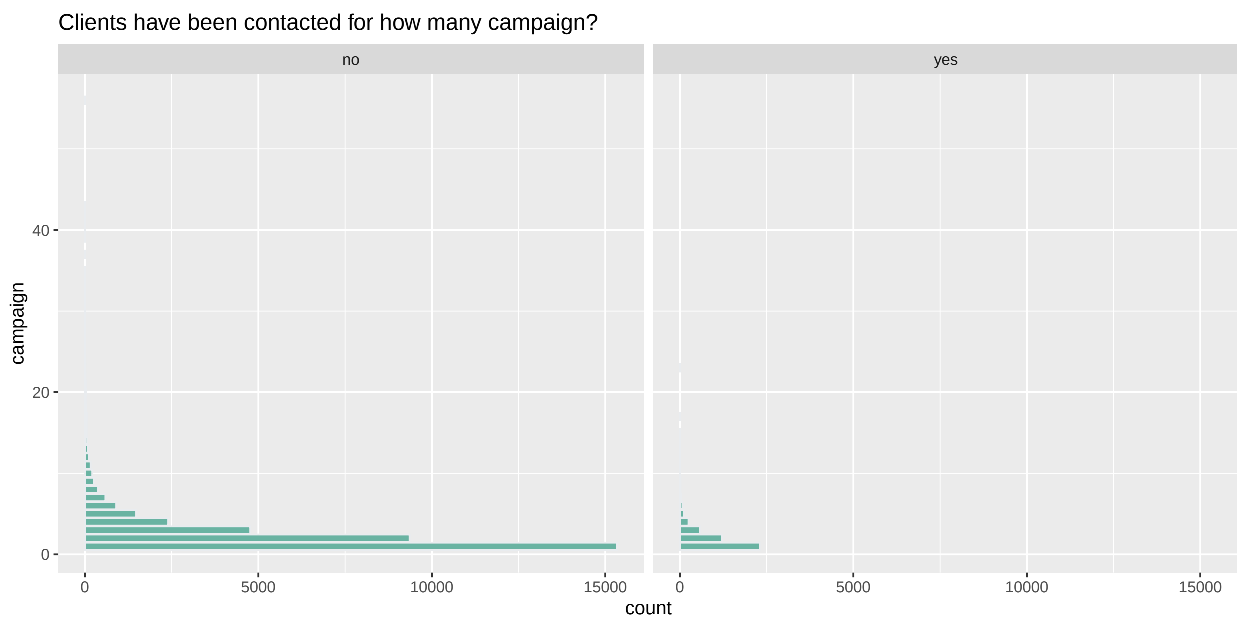

Clients are contacted a lot in *may* but in *mar*, *sep*, *oct* and *dec* has the number of **yes** and **no** clients in equal.



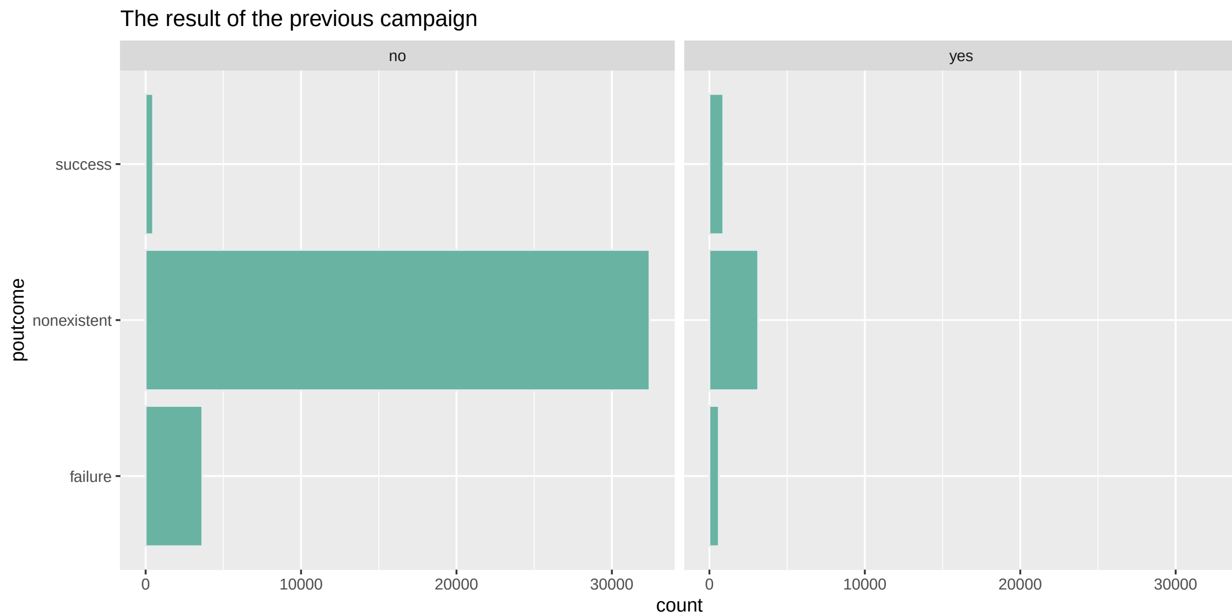

Clients are contacted similarly throughout the week, except 2 days of weekends (none contact was made in *sat* and *sun*).



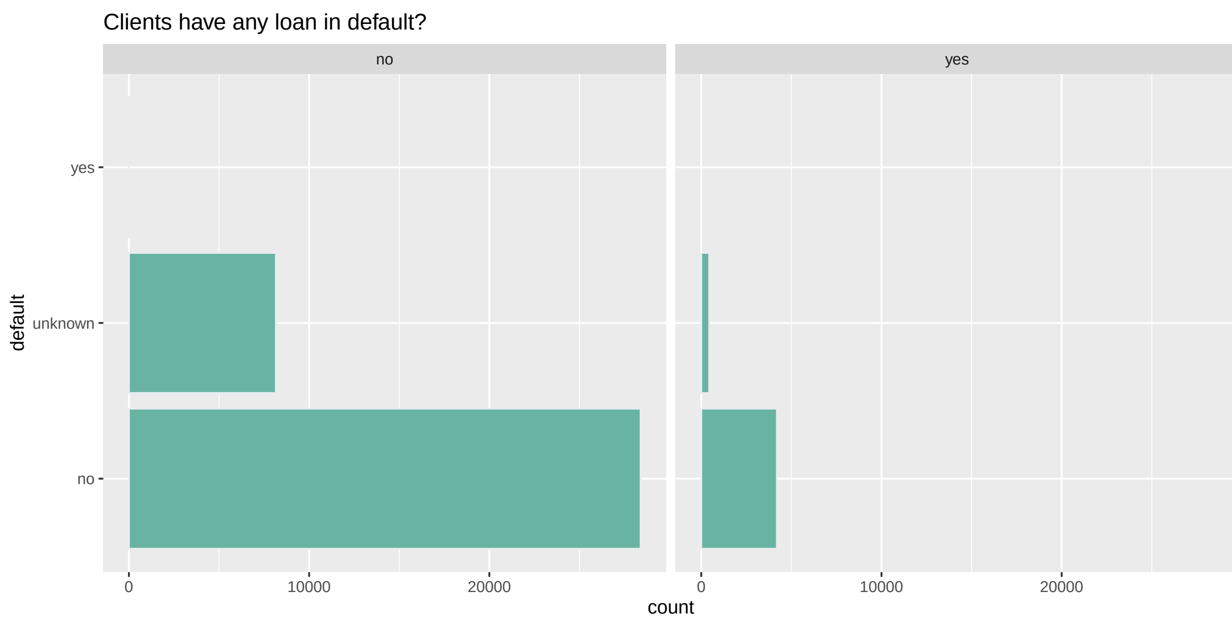

It seems that clients that said **yes** seems to be contacted longer than the **no** group.




It is similar between both groups in the number of historic campaigns.




The *success* of the previous campaign has a greater portions in **yes** group than in **no** group.




Most clients in **yes** group do not have a loan in default.

# 5 Split train/validation/test data

As this project will run several models, the **Portugese Bank dataset** will be divided into 3 parts: Train(60%), Test(20%) and Validation(20%).

Firstly, let's transform column y to factors for our models to do classification correctly.

```
# Transform y into factors
bank_full$y = as.factor(bank_full$y)
str(bank_full)
```

```
## Classes 'data.table' and 'data.frame':   41188 obs. of  20 variables:
##  $ age           : num  56 57 37 40 56 45 59 41 24 25 ...
##  $ job           : chr  "housemaid" "services" "services" "admin." ...
##  $ marital       : chr  "married" "married" "married" "married" ...
##  $ education     : chr  "basic.4y" "high.school" "high.school" "basic.6y" ...
##  $ default       : chr  "no" "unknown" "no" "no" ...
##  $ housing       : chr  "no" "no" "yes" "no" ...
##  $ loan          : chr  "no" "no" "no" "no" ...
##  $ contact       : chr  "telephone" "telephone" "telephone" "telephone" ...
##  $ month         : Factor w/ 12 levels "jan","feb","mar",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ day_of_week   : Factor w/ 7 levels "mon","tue","wed",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ duration      : num  261 149 226 151 307 198 139 217 380 50 ...
##  $ campaign      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays         : num  999 999 999 999 999 999 999 999 999 999 ...
##  $ previous      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome      : chr  "nonexistent" "nonexistent" "nonexistent" "nonexistent" ...
##  $ emp_var_rate  : num  1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...
##  $ cons_price_idx: num  94 94 94 94 94 ...
##  $ cons_conf_idx : num  -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...
##  $ nr_employed   : num  5191 5191 5191 5191 5191 ...
##  $ y             : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Secondly, split the dataset:

```
set.seed(123)
spec = c(train = .6, validation = .2, test = .2)
set.seed(123)
g = sample(cut(
  seq(nrow(bank_full)),
  nrow(bank_full)*cumsum(c(0,spec)),
  labels = names(spec)
))

# Split into train, test, validation dataset
res = split(bank_full, g)
# Move all dataset to global environment
list2env(res,globalenv())
```

Now, we are having 3 different parts of the dataset. Let's check again our 3 datasets

```
print(train)
```

```
##        age         job marital           education default housing loan
##     1:  56   housemaid married            basic.4y      no      no   no
##     2:  56    services married         high.school      no      no  yes
##     3:  59      admin. married professional.course      no      no   no
##     4:  24  technician  single professional.course      no     yes   no
##     5:  25    services  single         high.school      no     yes   no
##    ---
## 24708:  29  unemployed  single            basic.4y      no     yes   no
## 24709:  46 blue-collar married professional.course      no      no   no
## 24710:  56     retired married   university.degree      no     yes   no
```

```
## 24711:  44  technician married professional.course      no      no   no
## 24712:  74     retired married professional.course      no     yes   no
##          contact month day_of_week duration campaign pdays previous    poutcome
##     1: telephone   may         mon      261        1   999        0 nonexistent
##     2: telephone   may         mon      307        1   999        0 nonexistent
##     3: telephone   may         mon      139        1   999        0 nonexistent
##     4: telephone   may         mon      380        1   999        0 nonexistent
##     5: telephone   may         mon       50        1   999        0 nonexistent
##    ---
## 24708:  cellular   nov         fri      112        1     9        1     success
## 24709:  cellular   nov         fri      383        1   999        0 nonexistent
## 24710:  cellular   nov         fri      189        2   999        0 nonexistent
## 24711:  cellular   nov         fri      442        1   999        0 nonexistent
## 24712:  cellular   nov         fri      239        3   999        1     failure
##        emp_var_rate cons_price_idx cons_conf_idx nr_employed   y
##     1:          1.1          93.99         -36.4        5191  no
##     2:          1.1          93.99         -36.4        5191  no
##     3:          1.1          93.99         -36.4        5191  no
##     4:          1.1          93.99         -36.4        5191  no
##     5:          1.1          93.99         -36.4        5191  no
##    ---
## 24708:         -1.1          94.77         -50.8        4964  no
## 24709:         -1.1          94.77         -50.8        4964  no
## 24710:         -1.1          94.77         -50.8        4964  no
## 24711:         -1.1          94.77         -50.8        4964 yes
## 24712:         -1.1          94.77         -50.8        4964  no
```

Our train dataset has 24,712 rows and has the same number of columns

```
print(validation)
```

```
##       age          job  marital         education default housing loan
##    1:  57     services  married       high.school unknown      no   no
##    2:  37     services  married       high.school      no     yes   no
##    3:  41  blue-collar  married           unknown unknown      no   no
##    4:  41  blue-collar  married           unknown unknown      no   no
##    5:  57    housemaid divorced          basic.4y      no     yes   no
##   ---
## 8234:  35   technician divorced          basic.4y      no      no   no
## 8235:  38    housemaid divorced university.degree      no      no   no
## 8236:  38 entrepreneur  married university.degree      no      no   no
## 8237:  40   management divorced university.degree      no     yes   no
## 8238:  62      retired  married university.degree      no     yes   no
##         contact month day_of_week duration campaign pdays previous    poutcome
##    1: telephone   may         mon      149        1   999        0 nonexistent
##    2: telephone   may         mon      226        1   999        0 nonexistent
##    3: telephone   may         mon      217        1   999        0 nonexistent
##    4: telephone   may         mon       55        1   999        0 nonexistent
##    5: telephone   may         mon      293        1   999        0 nonexistent
##   ---
## 8234:  cellular   nov         tue      363        1   999        0 nonexistent
## 8235:  cellular   nov         wed      403        2   999        0 nonexistent
## 8236:  cellular   nov         wed      144        2   999        0 nonexistent
```

```
## 8237:  cellular   nov         wed      293        2   999        4     failure
## 8238:  cellular   nov         thu      208        1     1        6     success
##       emp_var_rate cons_price_idx cons_conf_idx nr_employed   y
##    1:          1.1          93.99         -36.4        5191  no
##    2:          1.1          93.99         -36.4        5191  no
##    3:          1.1          93.99         -36.4        5191  no
##    4:          1.1          93.99         -36.4        5191  no
##    5:          1.1          93.99         -36.4        5191  no
##   ---
## 8234:         -1.1          94.77         -50.8        4964 yes
## 8235:         -1.1          94.77         -50.8        4964 yes
## 8236:         -1.1          94.77         -50.8        4964  no
## 8237:         -1.1          94.77         -50.8        4964  no
## 8238:         -1.1          94.77         -50.8        4964 yes
```

Our validation dataset has 8,238 rows and has the same number of columns.

```
print(test)
```

```
##       age         job  marital           education default housing loan
##    1:  40      admin.  married            basic.6y      no      no   no
##    2:  45    services  married            basic.9y unknown      no   no
##    3:  37      admin.  married         high.school      no     yes   no
##    4:  46      admin.  married             unknown      no      no   no
##    5:  49 blue-collar  married             unknown      no      no   no
##   ---
## 8234:  31   housemaid   single   university.degree      no      no   no
## 8235:  31      admin.   single   university.degree      no     yes   no
## 8236:  34     student   single             unknown      no     yes   no
## 8237:  64     retired divorced professional.course      no     yes   no
## 8238:  73     retired  married professional.course      no     yes   no
##         contact month day_of_week duration campaign pdays previous    poutcome
##    1: telephone   may         mon      151        1   999        0 nonexistent
##    2: telephone   may         mon      198        1   999        0 nonexistent
##    3: telephone   may         mon      172        1   999        0 nonexistent
##    4: telephone   may         mon      348        1   999        0 nonexistent
##    5: telephone   may         mon       73        1   999        0 nonexistent
##   ---
## 8234:  cellular   nov         mon      159        4   999        0 nonexistent
## 8235:  cellular   nov         thu      353        1   999        0 nonexistent
## 8236:  cellular   nov         thu      180        1   999        2     failure
## 8237:  cellular   nov         fri      151        3   999        0 nonexistent
## 8238:  cellular   nov         fri      334        1   999        0 nonexistent
##       emp_var_rate cons_price_idx cons_conf_idx nr_employed   y
##    1:          1.1          93.99         -36.4        5191  no
##    2:          1.1          93.99         -36.4        5191  no
##    3:          1.1          93.99         -36.4        5191  no
##    4:          1.1          93.99         -36.4        5191  no
##    5:          1.1          93.99         -36.4        5191  no
##   ---
## 8234:         -1.1          94.77         -50.8        4964  no
## 8235:         -1.1          94.77         -50.8        4964 yes
## 8236:         -1.1          94.77         -50.8        4964  no
```

```
## 8237:          -1.1          94.77          -50.8          4964  no
## 8238:          -1.1          94.77          -50.8          4964 yes
```

Our test dataset has 8,238 rows and has the same number of columns.

# 6 Methods & Analysis

This project will use 3 different algorithms to predict the **yes** and **no** label in the dataset.
1 - Decision Tree
2 - Random Forest
3 - Logistic Regression
These 3 models will be trained on *train* set and validate on *validation* set of data to evaluate the model.

To select the best performing model, AUC and F1-score will both be used (our dataset are quite imbalanced so that F1_score is used). The best model will be tested on the *test* set of data for final results.

# 7 Model 1 - Decision Tree classification

## 7.1 Decision Tree model training

Our first model - The *Decision Tree* model will be used through the function below:

```
decis_tree = rpart(formula = y ~ ., data = train)
```

Let's see the importance of **Portugese bank dataset** features

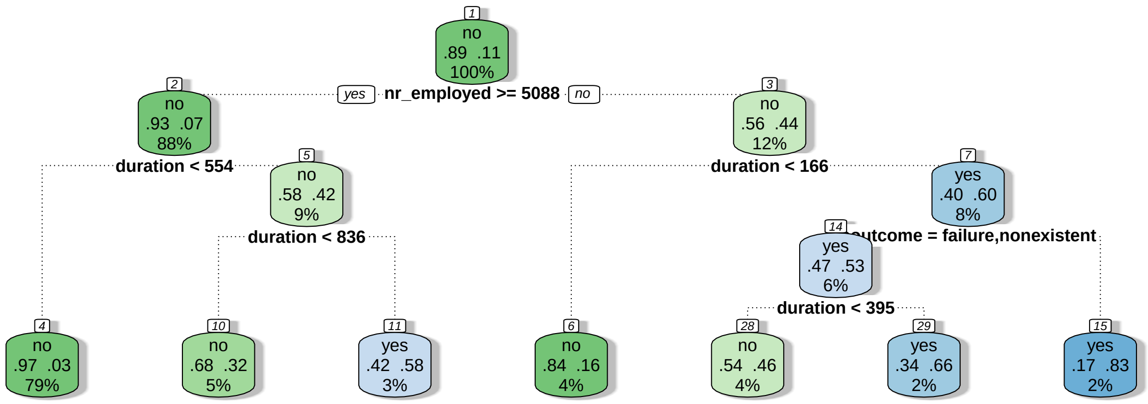
```
varImp(decis_tree)
```

```
##                Overall
## cons_conf_idx    78.10
## cons_price_idx   85.90
## contact          27.15
## duration       1624.63
## emp_var_rate    513.41
## month            77.97
## nr_employed     781.66
## pdays           656.75
## poutcome        641.99
## age               0.00
## job               0.00
## marital           0.00
## education         0.00
## default           0.00
## housing           0.00
## loan              0.00
## day_of_week       0.00
## campaign          0.00
## previous          0.00
```

The model stated that feature *Number of employees* is the most important feature when it classify between **yes** or **no**. Follow up is *euribor3m* in the second place and *nr_employed* in the third place.

To see the detailed result of our model, let's plot the result:

```
fancyRpartPlot(decis_tree, cex = 0.8, caption = "Decision Tree model results")
```




Decision Tree model results

The cutoff for feature *nr_employed* is *5088* and later on feature *duration* have several cutoffs to determine the value of *y*

## 7.2 Decision Tree model validation

Now, let's test this decision tree model in the *validation* set:

```
decis_pred = predict(decis_tree, validation, type = 'class')
```

## 7.3 F1 Score of Decision tree model

To see how good our model is doing, Confusion Matrix is a simple tool for quick evaluation.

```
confusionMatrix(data = decis_pred, reference = validation$y, positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  7150  505
##        yes  204  379
##
##                Accuracy : 0.914
##                  95% CI : (0.908, 0.92)
##     No Information Rate : 0.893
```

```
##     P-Value [Acc > NIR] : 0.0000000000756
##
##                   Kappa : 0.472
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.4287
##             Specificity : 0.9723
##          Pos Pred Value : 0.6501
##          Neg Pred Value : 0.9340
##              Prevalence : 0.1073
##          Detection Rate : 0.0460
##    Detection Prevalence : 0.0708
##       Balanced Accuracy : 0.7005
##
##        'Positive' Class : yes
##
```

Calculated the Confusion Matrix, we have:
The *sensitivity* is

```
dtree_sense = caret::sensitivity(decis_pred, validation$y)
print(dtree_sense)
```

```
## [1] 0.9723
```

The *specificity* is

```
dtree_spec = caret::specificity(decis_pred, validation$y)
print(dtree_spec)
```

```
## [1] 0.4287
```

The *F1 score* is

```
decis_f1 = 2*(dtree_sense * dtree_spec)/(dtree_sense + dtree_spec)
print(decis_f1)
```

```
## [1] 0.5951
```

We know that increasing precision will decrease recall, and vice versa. So that our *F1 score* is quite in the middle in the range from 0 to 1.

## 7.4 AUC of Decision Tree model

The *AUC* of *Decision Tree* model is

```
decis_AUC = round(AUC::auc(roc(decis_pred, validation$y)),2)
print(decis_AUC)
```

```
## [1] 0.7
```

So that, Decision Tree model has a better-than-average AUC.

# 8 Model 2 - Random Forest

The second model we are trying to classify our dataset is *Random Forest*

## 8.1 Random Forest model training

Let's train the *Random Forest* model using the train dataset.

## 8.2 Random Forest model validation

Let's test the *Random Forest* model in the *validation* dataset.

```
rf_pred = predict(rf_fit, validation)
head(rf_pred)
```

```
##  1  2  3  4  5  6
## no no no no no no
## Levels: no yes
```

Before tuning, the AUC of *Random Forest* model is

```
rf_AUC = round(AUC::auc(roc(rf_pred, validation$y)),2)
print(rf_AUC)
```

```
## [1] 0.73
```

## 8.3 Tuning the model

One parameter of *Random Forest* could be tuning is the number of trees. So that, we will fit the model with a sequence of different trees's numbers to see how the model turns out.

```
rf_tuned_trees <- seq(from=1, to=200, by=10)
print(rf_tuned_trees)
```

```
##  [1]   1  11  21  31  41  51  61  71  81  91 101 111 121 131 141 151 161 171 181
## [20] 191
```

```
rf_tuned = data.frame(matrix(ncol = 2, nrow = 0))

for (i in rf_tuned_trees) {
  fit = randomForest(y~. , train, ntree = i, importance = TRUE)
  pred =  predict(fit, validation)
  auc = AUC::auc(roc(pred, validation$y))
  result = cbind(i, auc)
  rf_tuned = rbind(rf_tuned, result)
}
rf_tuned_results = rf_tuned[rf_tuned$auc == max(rf_tuned$auc),]
rf_tuned_results
```

```
##    i   auc
## 6 51 0.738
```

So that, the best performance number of trees is

```r
rf_tuned_results$i
```

```
## [1] 51
```

So, let's predict the *validation* dataset:

```r
rf_tuned_fit = randomForest(y~. , train, ntree = rf_tuned_results$i, importance = TRUE)
rf_tuned_pred = predict(fit, validation, type = 'class')
```

## 8.4 F1 score of Random Forest model

Here is the confusion matrix of the tuned *Random Forest* model:

```r
confusionMatrix(data = rf_tuned_pred, reference = validation$y, positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  7135  451
##        yes  219  433
##
##                Accuracy : 0.919
##                  95% CI : (0.913, 0.924)
##     No Information Rate : 0.893
##     P-Value [Acc > NIR] : 0.00000000000000143
##
##                   Kappa : 0.52
##
##  Mcnemar's Test P-Value : < 0.0000000000000002
##
##             Sensitivity : 0.4898
##             Specificity : 0.9702
##          Pos Pred Value : 0.6641
##          Neg Pred Value : 0.9405
##              Prevalence : 0.1073
##          Detection Rate : 0.0526
##    Detection Prevalence : 0.0791
##       Balanced Accuracy : 0.7300
##
##        'Positive' Class : yes
##
```

Calculated the Confusion Matrix, we have:

The *sensitivity* is

```r
rf_sense = caret::sensitivity(rf_tuned_pred, validation$y)
print(rf_sense)
```

```
## [1] 0.9702
```

The *specificity* is

```r
rf_spec = caret::specificity(rf_tuned_pred, validation$y)
print(rf_spec)
```

```
## [1] 0.4898
```

The *F1 score* of *Random Forest* model is

```r
rf_f1 = 2*(rf_sense * rf_spec)/(rf_sense + rf_spec)
print(rf_f1)
```

```
## [1] 0.651
```

The *F1 score* of *Random Forest* model is better than *Decision Tree* model *F1 score*.

## 8.5 AUC of Random Forest model

The *Random Forest* model AUC is

```r
rf_tuned_AUC = round(AUC::auc(roc(rf_pred, validation$y)),2)
print(rf_tuned_AUC)
```

```
## [1] 0.73
```

# 9 Model 3 - Logistic Regression

The third model we are trying to classify our dataset is *Logistic Regression*.

## 9.1 Logistic Regression model training

```r
glm_fit = glm(y ~ ., family = "binomial", data = train)
```

```r
varImp(glm_fit)
```

```
##                                Overall
## age                           0.650835
## jobblue-collar                3.032989
## jobentrepreneur               0.377886
## jobhousemaid                  0.310976
```

```
## jobmanagement                  0.422543
## jobretired                     2.390358
## jobself-employed               0.890042
## jobservices                    1.139917
## jobstudent                     0.863064
## jobtechnician                  0.106329
## jobunemployed                  0.521187
## jobunknown                     0.309663
## maritalmarried                 0.959067
## maritalsingle                  0.717519
## maritalunknown                 0.726166
## educationbasic.6y              0.795419
## educationbasic.9y              0.411873
## educationhigh.school           0.006519
## educationilliterate            0.289566
## educationprofessional.course   0.232492
## educationuniversity.degree     1.336654
## educationunknown               0.227358
## defaultunknown                 3.222377
## defaultyes                     0.063922
## housingunknown                 0.891420
## housingyes                     0.025467
## loanyes                        0.254798
## contacttelephone               6.253070
## monthapr                      11.169391
## monthmay                      15.389068
## monthjun                       9.185682
## monthjul                       9.556987
## monthaug                       7.047740
## monthsep                       7.414594
## monthoct                       8.912861
## monthnov                      12.530548
## monthdec                       6.058824
## day_of_weektue                 3.484525
## day_of_weekwed                 4.227478
## day_of_weekthu                 1.955356
## day_of_weekfri                 2.236124
## duration                      49.081169
## campaign                       2.400477
## pdays                          2.190277
## previous                       0.510292
## poutcomenonexistent            3.407010
## poutcomesuccess                4.006100
## emp_var_rate                   8.598899
## cons_price_idx                 7.684995
## cons_conf_idx                  5.732782
## nr_employed                    3.342935
```

The model stated that feature *duration* is the most important feature when it classify between *yes* or *no*. Follow up is *monthmay* (May in month) in the second place and *monthnov* (Nov in month) in the third place.

## 9.2 Logistic Regression model validation

```
glm_res = predict(glm_fit, validation, type = 'response')
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
head(glm_res)
```

```
##        1        2        3        4        5        6
## 0.005427 0.010800 0.006667 0.003071 0.014009 0.006945
```

After predict the y in the *test set*, we could see that the predict function for *Logistic Resgression* model returns a vector of probabilities.
Our next task is to find the best threshold to determine whether one probability is *yes* or *no*.

## 9.3 Calculating threshold value

Good threshold value is the number classify *yes* and *no* with the minimum number of error class.
So that, we will try different threshold to find the best one:

```
prob = seq(0, 1, length.out = 10)
error = data.frame(matrix(ncol = 3, nrow = 0))
for (i in prob) {
  split = ifelse(glm_res >= i, 'yes', 'no')
  error_yes = 100*sum(ifelse(split == 'yes' & validation$y=='no', 1,0))/length(split)
  error_no = 100*sum(ifelse(split == 'no' & validation$y=='yes', 1,0))/length(split)
  auc = round(AUC::auc(roc(split, validation$y)),2)
  col = cbind(i,auc, error_yes, error_no, tot_error = sum(error_yes, error_no))
  error = rbind(error,col)

}
error_results = error[error$auc == max(error$auc),]
print(error_results)
```

```
##        i  auc error_yes error_no tot_error
## 2 0.1111 0.87     12.65    1.262     13.91
```

So that, the threshold *0.1111* is the threshold with the highest AUC with only 13% guessing wrong for **yes**.
We will use this to evaluate the model.

The code below will determine either a client in *validation* dataset is **yes** or **no**.

```
glm_pred = factor(ifelse(glm_res >= error_results$i , "yes", "no"), levels = levels(validation$y))
```

## 9.4 F1 Score of Logistic Regression model

Here is the confusion matrix of the *Logistic Regression* model:

```r
confusionMatrix(data = glm_pred, reference = validation$y, positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  6312  104
##        yes 1042  780
##
##                Accuracy : 0.861
##                  95% CI : (0.853, 0.868)
##     No Information Rate : 0.893
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.505
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.8824
##             Specificity : 0.8583
##          Pos Pred Value : 0.4281
##          Neg Pred Value : 0.9838
##              Prevalence : 0.1073
##          Detection Rate : 0.0947
##    Detection Prevalence : 0.2212
##       Balanced Accuracy : 0.8703
##
##        'Positive' Class : yes
##
```

Calculated the Confusion Matrix, we have:
The *sensitivity* is

```r
glm_sense = caret::sensitivity(glm_pred, validation$y)
print(glm_sense)
```

```
## [1] 0.8583
```

The *specificity* is

```r
glm_spec = caret::specificity(glm_pred, validation$y)
print(glm_spec)
```

```
## [1] 0.8824
```

The *F1 score* is

```r
glm_f1 = 2*(glm_sense * glm_spec)/(glm_sense + glm_spec)
print(glm_f1)
```

```
## [1] 0.8702
```

So that our *F1 score* is a very good score in the range from 0 to 1.

### 9.5 AUC of Logistic Regression model

The AUC of *Logistic Regression* model is

```
glm_AUC = round(AUC::auc(roc(glm_pred, validation$y)),2)
print(glm_AUC)
```

```
## [1] 0.87
```

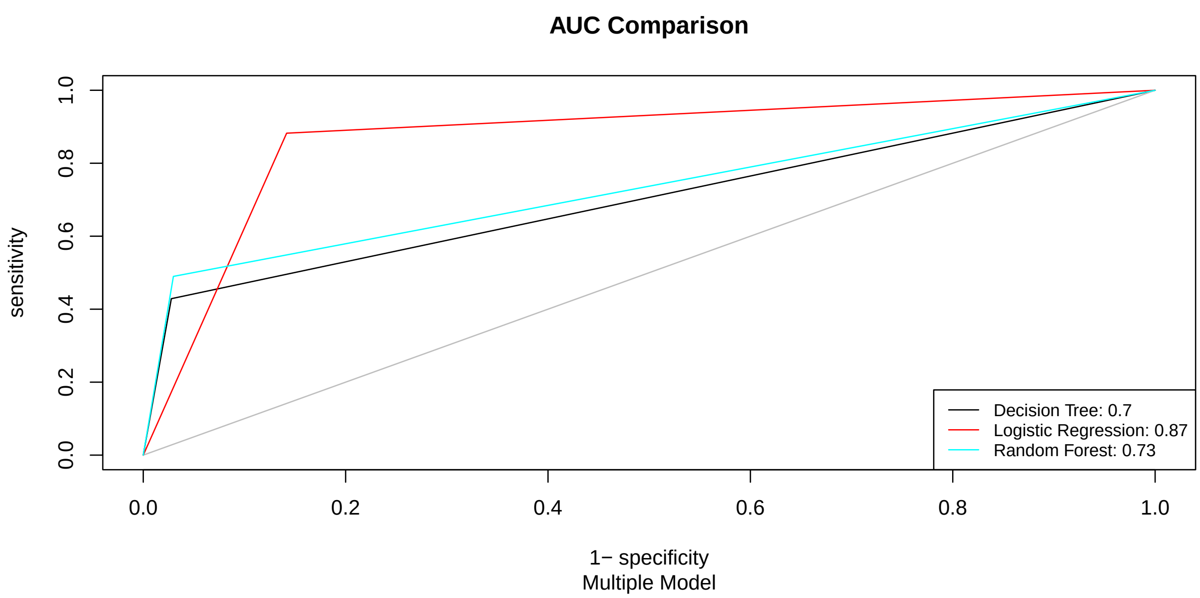
The AUC of *Logistic Regression* model is better than the Decision Tree and Random Forest.

## 10 Comparing the models performance

Here is the performance of our models:

```
##                Models F1_score  AUC
## 1       Decision Tree   0.5951 0.70
## 2       Random Forest   0.6510 0.73
## 3 Logistic Regresion   0.8702 0.87
```

Let's also plot the models ROC curve to visualize the performance:




So that, *Logistic Regression* is the best performing models in this project.

## 11 Testing on test set

Let's test *Logistic Regression* model on the *test* dataset.

### 11.1 Predict the test dataset

```r
glm_test_pred = predict(glm_fit, test, type = 'response')
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```r
head(glm_test_pred)
```

```
##        1        2        3        4        5        6
## 0.009633 0.006691 0.009480 0.022181 0.004356 0.013276
```

### 11.2 Calculating threshold value for test dataset

Again, we will try different threshold to find the best one:

```r
prob_test = seq(0, 1, length.out = 10)
error_test = data.frame(matrix(ncol = 3, nrow = 0))
for (i in prob_test) {
  split = ifelse(glm_test_pred >= i, 'yes', 'no')
  error_yes = 100*sum(ifelse(split == 'yes' & test$y=='no', 1,0))/length(split)
  error_no = 100*sum(ifelse(split == 'no' & test$y=='yes', 1,0))/length(split)
  auc = round(AUC::auc(roc(split, test$y)),2)
  col = cbind(i,auc, error_yes, error_no, tot_error = sum(error_yes, error_no))
  error_test = rbind(error_test,col)

}
error_test_results = error_test[error_test$auc == max(error_test$auc),]
print(error_test_results)
```

```
##        i  auc error_yes error_no tot_error
## 2 0.1111 0.87     12.93    1.384     14.31
```

So that, the threshold *0.1111* is the threshold with the highest AUC with only 13% guessing wrong for **yes**. We will use this to evaluate the model on the *test* dataset.

The code below will determine either a client in *test* dataset is **yes** or **no**.

```r
glm_test_pred_res = factor(ifelse(glm_test_pred >= error_test_results$i, "yes", "no"),
                           levels = levels(test$y))
```

### 11.3 F1 Score of Logistic Regression model

Here is the confusion matrix of the *Logistic Regression* model:
The *sensitivity* is

```r
confusionMatrix(data = glm_test_pred_res, reference = test$y, positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction   no  yes
##        no  6166  114
##        yes 1065  893
##
##                Accuracy : 0.857
##                  95% CI : (0.849, 0.864)
##     No Information Rate : 0.878
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.526
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.887
##             Specificity : 0.853
##          Pos Pred Value : 0.456
##          Neg Pred Value : 0.982
##              Prevalence : 0.122
##          Detection Rate : 0.108
##    Detection Prevalence : 0.238
##       Balanced Accuracy : 0.870
##
##        'Positive' Class : yes
##
```

```
glm_test_sense = caret::sensitivity(glm_test_pred_res, test$y)
print(glm_sense)
```

```
## [1] 0.8583
```

The *specificity* is

```
glm_test_spec = caret::specificity(glm_test_pred_res, test$y)
print(glm_spec)
```

```
## [1] 0.8824
```

The *F1 score* is

```
glm_test_f1 = 2*(glm_test_sense * glm_test_spec)/(glm_test_sense + glm_test_spec)
print(glm_f1)
```

```
## [1] 0.8702
```

## 11.4 AUC of Logistic Regression model

The AUC of *Logistic Regression* model is

```
glm_test_AUC = round(AUC::auc(roc(glm_test_pred_res, test$y)),2)
print(glm_test_AUC)
```

```
## [1] 0.87
```

# 12 Results

So that, the best performing model is *Logistic Regression* model with the F1 score on test set is 0.87 and AUC on test set is 0.87 (with *threshold at 0.1111*).
On the range from 0 to 1, this model F1 score is considered very good nearly 0.9 out of 1. And the AUC is scored at the similar score in the same range. This high AUC means this model have a very good performance at distinguishing between the positive and negative classes.

# 13 Conclusion

In this reports, we used three models *Decision Tree*, *Random Forest* and *Logistic Regression* to **predict whether one contact in the dataset resulted in a bank term deposit or not**.

After training, validating and testing on 3 datasets *train*, *validation* and *test*. The best performing model is *Logistic Regression* with **AUC of 0.87** and **F1-score of 0.87** on the *test* dataset.

The project has not make use of any robust or boosted version of these models and other more advanced algorithm to predict. So that, in future work, boosted versions and more advanced algorithm would be used to predict and have a better performance score.