EdX_Final_Assessment_OwnProject

Vladimir Bousrez

2023-11-24

Own Project Bank Score

Part 1 - Introduction

In this project, we create a classification system leveraging on a dataset where we predict the event {y=0} and {y=1}. We will apply a machine learning algorithm to the training set of the data set named “bank_score” and test it on a final hold-out set, made of 10% of available data. The final hold-out test will not be used for training or selection of the algorithm, it will only be used at the end of the project on the final model. Instead, the 90% of data available for training from bank_score, called bank_score_set, should be split further into train and test. The criteria to assess the success of the algorithm is the Area Under the Curve (AUC) and the percentage of true positive on the test set. AUC measures how the model performs in distinguising the true and false positive. In order to proceed, we will:

- Prepare the work environment:
    - Download and load the libraries.
    - Download, unzip and consolidate the bank_score dataset.
    - Dedicate 10% of the “bank_score” file to final testing in the final_holdout_test. The remaining 90% goes to “bank_score_set”.
- Split “bank_score_set” between test and train.
- Analyse the data set.
- Calculate the AUC for four models.
    - glm (logistic regression) on several variables
    - knn (k-nearest neightboors classification model) on several variables
    - rf (random forest) on several variables
    - best rf model with 3 selected hyperparameters from a cross-validation and a grid search.
- Calculation of AUC on the final model.

Part 2 – Methods/analysis

Data preparation

The files have been downloaded (from kaggle, url=https://www.kaggle.com/datasets/kapturovalexander/bank-credit-scoring) on the computer and saved under “C:/Users/vladi/OneDrive/Documents/R/Own-project”

2.1 Data vizualisation

The completeness of the data set will be assessed here, to estimate the need for data cleaning. Then, each variable will be viewed to assess opportunities of adjustments and if we should keep them or not in the final model. We start with analyzing the structure of the data.

Structure of the Data

```
str(bank_score)
```

```
## 'data.frame':    4521 obs. of  17 variables:
##  $ age      : int  30 33 35 30 59 35 36 39 41 43 ...
##  $ job      : chr  "unemployed" "services" "management" "management" ...
##  $ marital  : chr  "married" "married" "single" "married" ...
##  $ education: chr  "primary" "secondary" "tertiary" "tertiary" ...
##  $ default  : chr  "no" "no" "no" "no" ...
##  $ balance  : int  1787 4789 1350 1476 0 747 307 147 221 -88 ...
##  $ housing  : chr  "no" "yes" "yes" "yes" ...
##  $ loan     : chr  "no" "yes" "no" "yes" ...
##  $ contact  : chr  "cellular" "cellular" "cellular" "unknown" ...
##  $ day      : int  19 11 16 3 5 23 14 6 14 17 ...
##  $ month    : chr  "oct" "may" "apr" "jun" ...
##  $ duration : int  79 220 185 199 226 141 341 151 57 313 ...
##  $ campaign : int  1 1 1 4 1 2 1 2 2 1 ...
##  $ pdays    : int  -1 339 330 -1 -1 176 330 -1 -1 147 ...
##  $ previous : int  0 4 1 0 0 3 2 0 0 2 ...
##  $ poutcome : chr  "unknown" "failure" "failure" "unknown" ...
##  $ y        : chr  "no" "no" "no" "no" ...
```

summary of the Data

```
summary(bank_score)
```

```
##       age            job              marital           education        
##  Min.   :19.00   Length:4521        Length:4521        Length:4521       
##  1st Qu.:33.00   Class :character   Class :character   Class :character  
##  Median :39.00   Mode  :character   Mode  :character   Mode  :character  
##  Mean   :41.17                                                           
##  3rd Qu.:49.00                                                           
##  Max.   :87.00                                                           
##    default             balance        housing              loan          
##  Length:4521        Min.   :-3313   Length:4521        Length:4521       
##  Class :character   1st Qu.:   69   Class :character   Class :character  
```

```
##  Mode  :character   Median :  444   Mode  :character   Mode  :character
##                     Mean   : 1423
##                     3rd Qu.: 1480
##                     Max.   :71188
##    contact               day           month             duration
##  Length:4521        Min.   : 1.00   Length:4521        Min.   :   4
##  Class :character   1st Qu.: 9.00   Class :character   1st Qu.: 104
##  Mode  :character   Median :16.00   Mode  :character   Median : 185
##                     Mean   :15.92                      Mean   : 264
##                     3rd Qu.:21.00                      3rd Qu.: 329
##                     Max.   :31.00                      Max.   :3025
##     campaign          pdays            previous          poutcome
##  Min.   : 1.000   Min.   : -1.00   Min.   : 0.0000   Length:4521
##  1st Qu.: 1.000   1st Qu.: -1.00   1st Qu.: 0.0000   Class :character
##  Median : 2.000   Median : -1.00   Median : 0.0000   Mode  :character
##  Mean   : 2.794   Mean   : 39.77   Mean   : 0.5426
##  3rd Qu.: 3.000   3rd Qu.: -1.00   3rd Qu.: 0.0000
##  Max.   :50.000   Max.   :871.00   Max.   :25.0000
##       y
##  Length:4521
##  Class :character
##  Mode  :character
##
##
##
```

We can now see the format of each variable, and what its content looks like. We also get the overview of the minimum and maximum value for each variable and a first understanding of which variable might need an adjustment.

We also see that there is no empty cell in the dataset when we look at the n/a in each column.

We will also look at the number of unique value per variable.

```
## age        67
## job        12
## marital    3
## education  4
## default    2
## balance    2353
## housing    2
## loan       2
## contact    3
## day        31
## month      12
## duration   875
## campaign   32
## pdays      292
## previous   24
```

```
## poutcome    4
## y           2
```

We can see that job, marital, education, default, housing, loan, contact, month, poutcome and y are categorical while age, balance, day, duration, campaign, pdays, previous are more continuous.

Variable vizualisation - categorical

We first establish a function which we will use to create a graph on each categorical variable.

Function

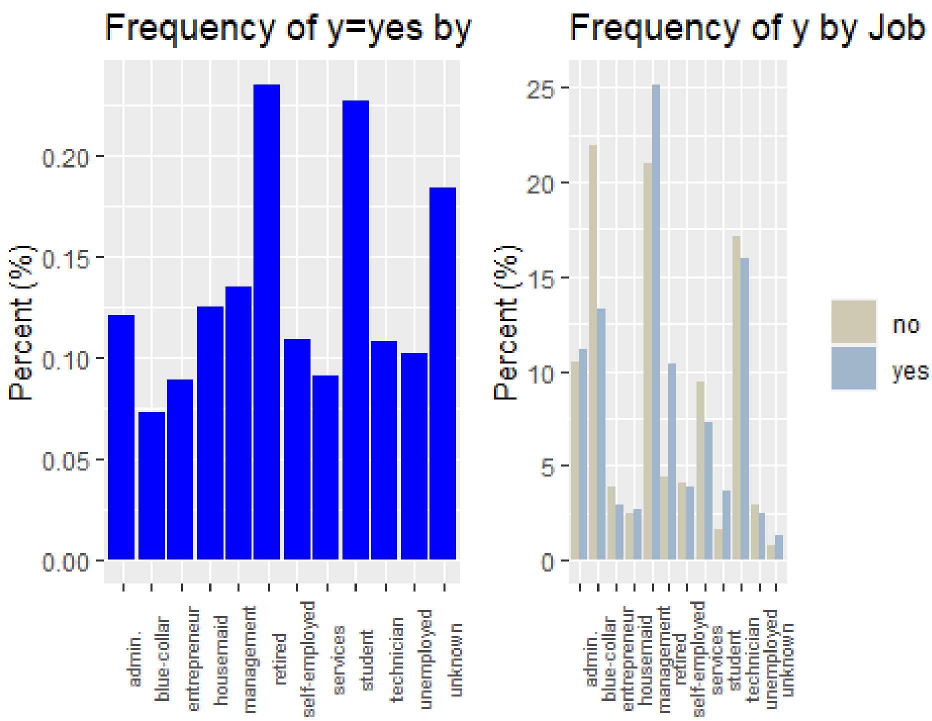
```
twograph <- function(namevar,labelvar,xfactors=NULL) {
  data_score=bank_score[,c(namevar,"y")]
  names(data_score) <- c("x","y")
#factor important for order of graphs
  if (!is.null(xfactors)) {
    data_score$x =
      factor(data_score$x,levels=xfactors)
  }
  Graph1 <- data_score %>%
    group_by(x) %>%
    summarize(avg = mean(y == "yes")) %>%
    ggplot(aes(x = x, y = avg)) +
    geom_bar(stat = "identity", fill = "blue") +
    labs(
      title = paste("Frequency of y=yes by ",
                    labelvar,sep = ""),
      #x = libvar,
      x = " ",
      y = "Percent (%)"
      #y = "Average 'Yes' Responses"
    ) +
    #theme(axis.text.x = element_text(angle = 45, hjust = 1))+
    theme(axis.text.x = element_text(angle = 90))+
    theme(axis.text.x = element_text(size = rel(0.8)))

  m = table(data_score$x, data_score$y)
  n = round(100*prop.table(m,2),2)

  no <- n[,1]
  yes <- n[,2]
  md <- row.names(n)

  df1 <- data.frame(no, yes, md)
  df2 <- melt(df1, id.vars='md')
  #print(df2)

  if (!is.null(xfactors)) {
```

```
    df2$md = factor(df2$md,levels=xfactors)
  }

  Graph2 <-ggplot(df2, aes(x=md, y=value, fill=variable)) +
    ggtitle(paste("Frequency of y by ", labelvar,sep = ""))+
    geom_bar(stat='identity', position='dodge') +
    #labs(y= "Percent of y", x = "Job type")+
    labs(y= "Percent (%)", x = " ")+
    labs(fill = " ") +
    scale_fill_manual(values=c("#CDC8B1","#9FB6CD"))+
    theme(axis.text.x = element_text(angle = 90))+
    theme(axis.text.x = element_text(size = rel(0.8)))

    #plot(Graph1)
    #plot(Graph2)
    cat("variable=",labelvar,"\n")
    print(t(n))
    grid.arrange(Graph1, Graph2, ncol = 2)

  # get in a list all the variables of the function
  return (list(graph1=Graph1,graph2=Graph2,
               m=m,n=n,yes=yes,no=no,md=md,
               df1=df1,df2=df2,namevar=namevar,
               labelvar=labelvar))

}
```

job

```
g1g2 = twograph("job","Job")
```

```
## variable= Job
##
##       admin. blue-collar entrepreneur housemaid management retired
##   no   10.50       21.92         3.82      2.45      20.95    4.40
##   yes  11.13       13.24         2.88      2.69      25.14   10.36
##
##       self-employed services student technician unemployed unknown
##   no           4.08     9.47    1.62      17.12       2.88    0.78
##   yes          3.84     7.29    3.65      15.93       2.50    1.34
```

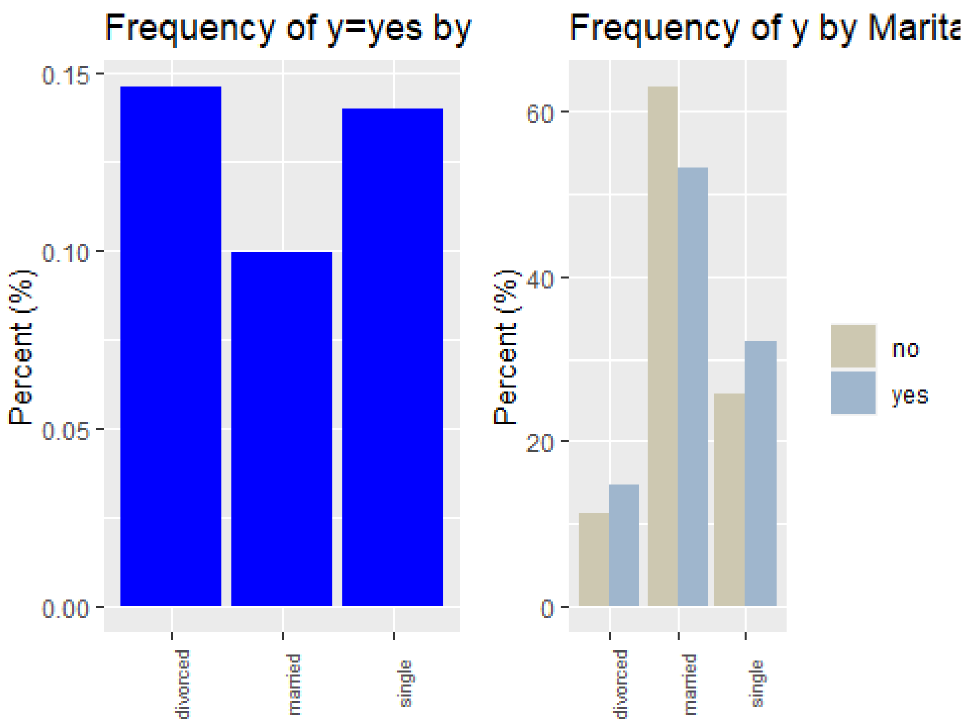



```
#Visual difference between yes and no: variable to keep
```

We can see strong difference in the distribution of yes and no for each position, we will definitely keep this variable in the algorithm.

marital

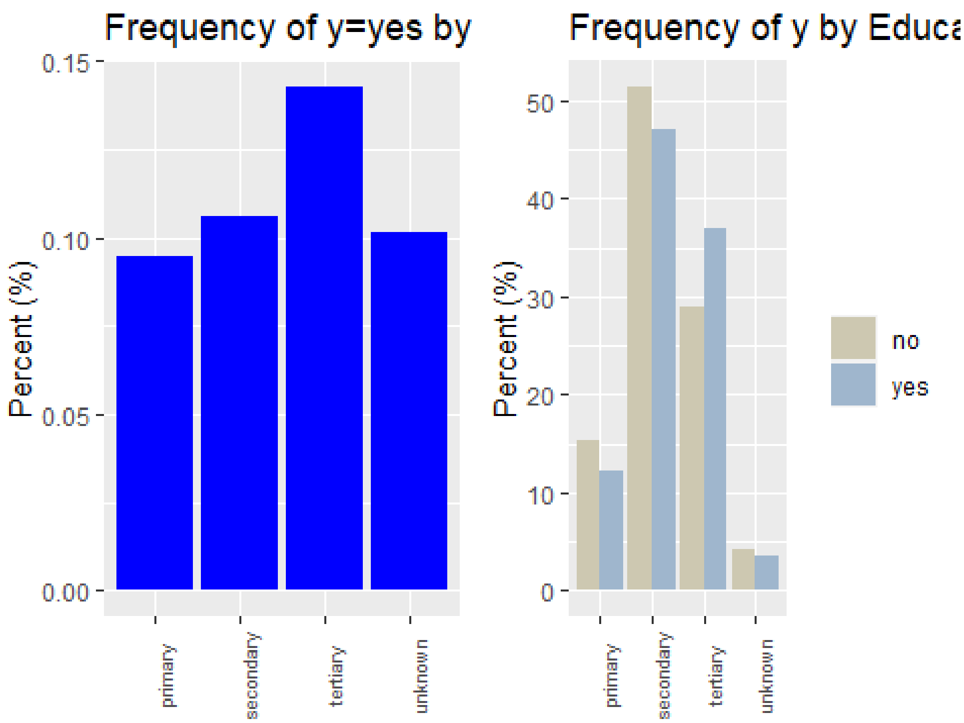
```
## variable= Marital
##
##         divorced married single
##    no      11.28   63.00  25.72
##    yes     14.78   53.17  32.05
```



We can see strong difference in the distribution of yes and no, we will definitely keep this variable in the algorithm.

education

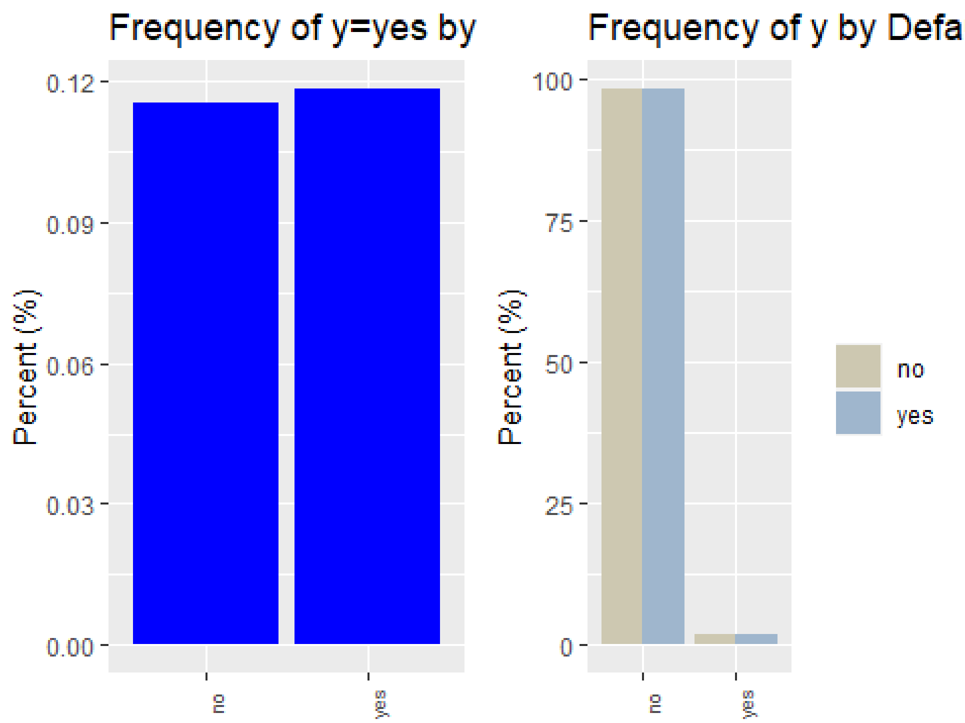
```
## variable= Education
##
##         primary secondary tertiary unknown
##    no    15.35     51.52    28.92    4.20
##    yes   12.28     47.02    37.04    3.65
```



We can see strong difference in the distribution of yes and no, we will definitely keep this variable in the algorithm.

default

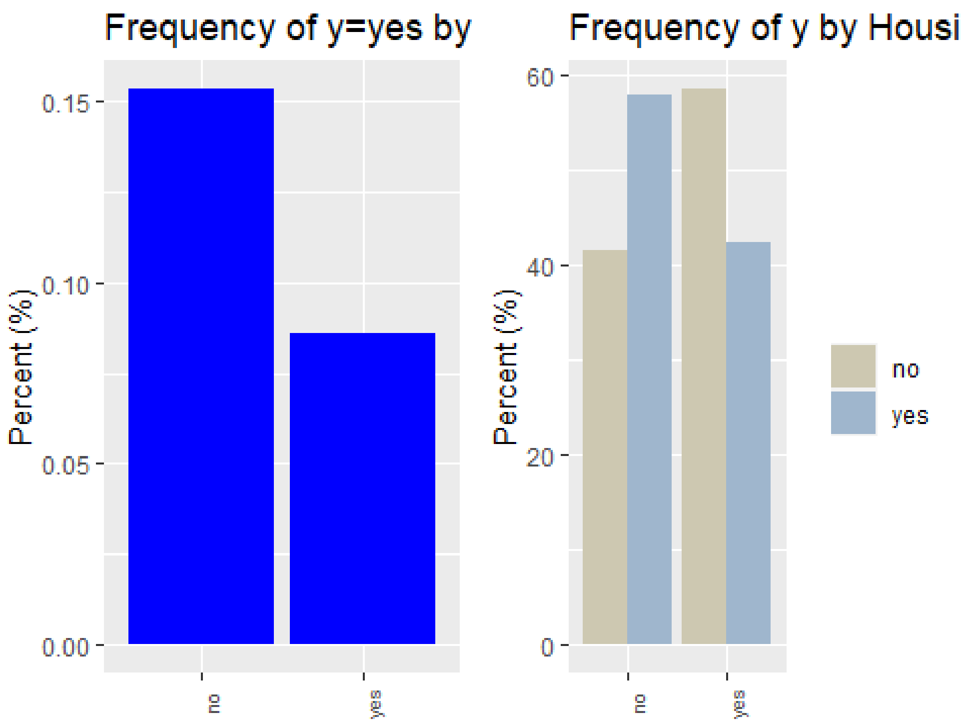
```
## variable= Default
##
##          no   yes
##   no  98.32  1.68
##   yes 98.27  1.73
```



The trend is not clear visually, we will need to check the variable further

housing

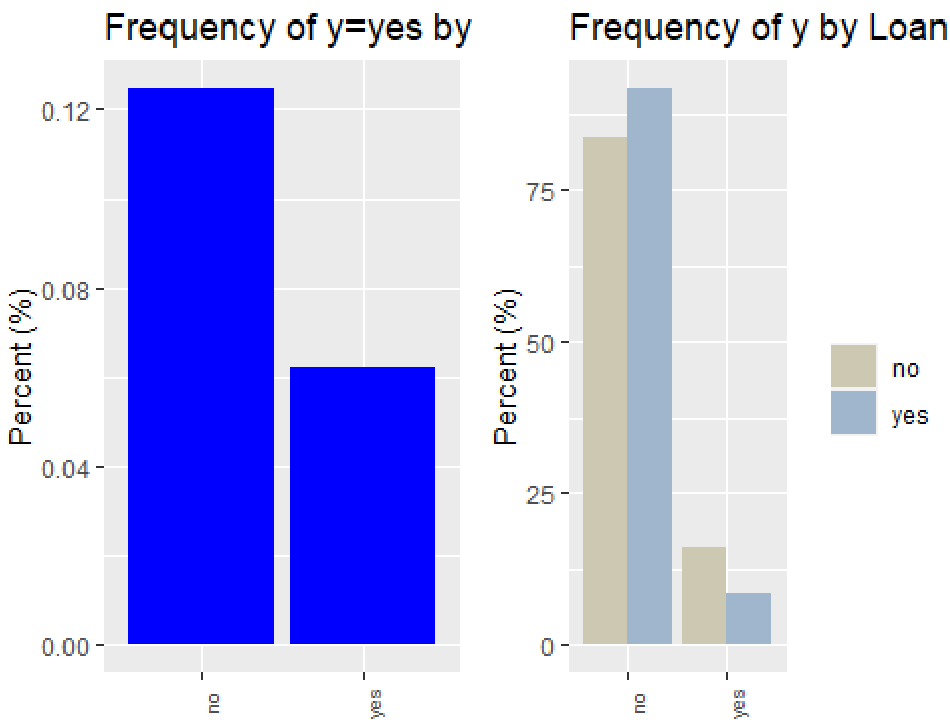
```
## variable= Housing
##
##          no   yes
##   no  41.52 58.48
##   yes 57.77 42.23
```



We can see strong difference in the distribution of yes and no, we will definitely keep this variable in the algorithm.

loan

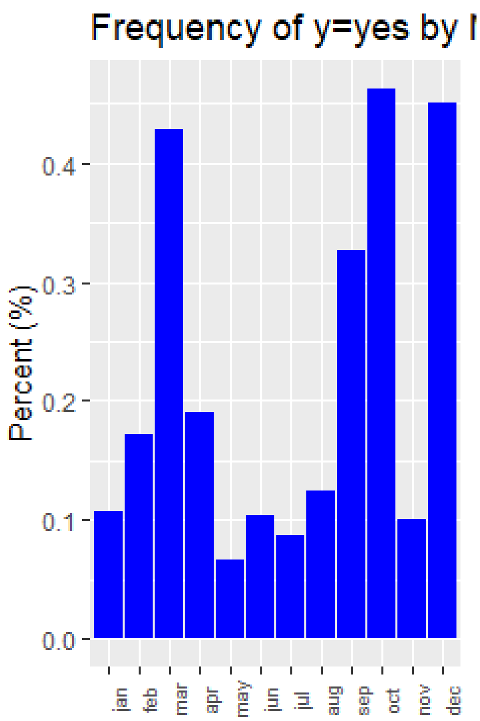
```
## variable= Loan
##
##          no   yes
##   no  83.80 16.20
##   yes 91.75  8.25
```

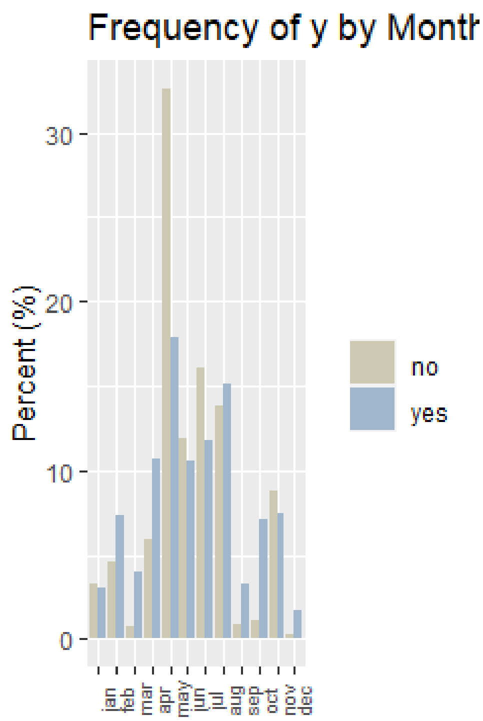


We can see strong difference in the distribution of yes and no, we will definitely keep this variable in the algorithm.

month

```
## variable= Month
##
##          jan   feb   mar   apr   may   jun   jul   aug   sep   oct   nov
dec
##    no   3.30  4.60  0.70  5.92 32.62 11.90 16.12 13.85  0.88  1.07  8.75
0.27
##    yes  3.07  7.29  4.03 10.75 17.85 10.56 11.71 15.16  3.26  7.10  7.49
1.73
```

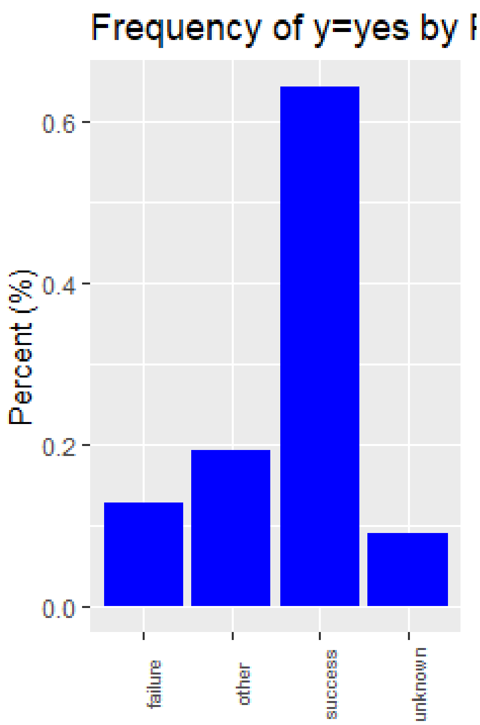


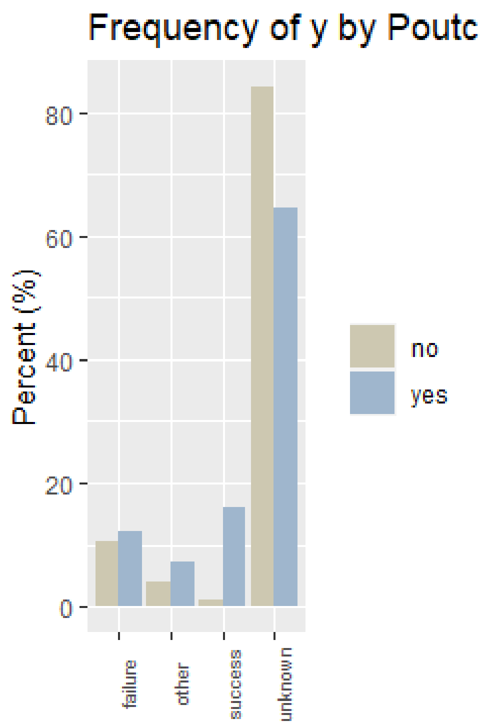


We can see strong difference in the distribution of yes and no, we will definitely keep this variable in the algorithm.

poutcome

```
## variable= Poutcome
##
##        failure other success unknown
##    no    10.67  3.98    1.15   84.20
##    yes   12.09  7.29   15.93   64.68
```

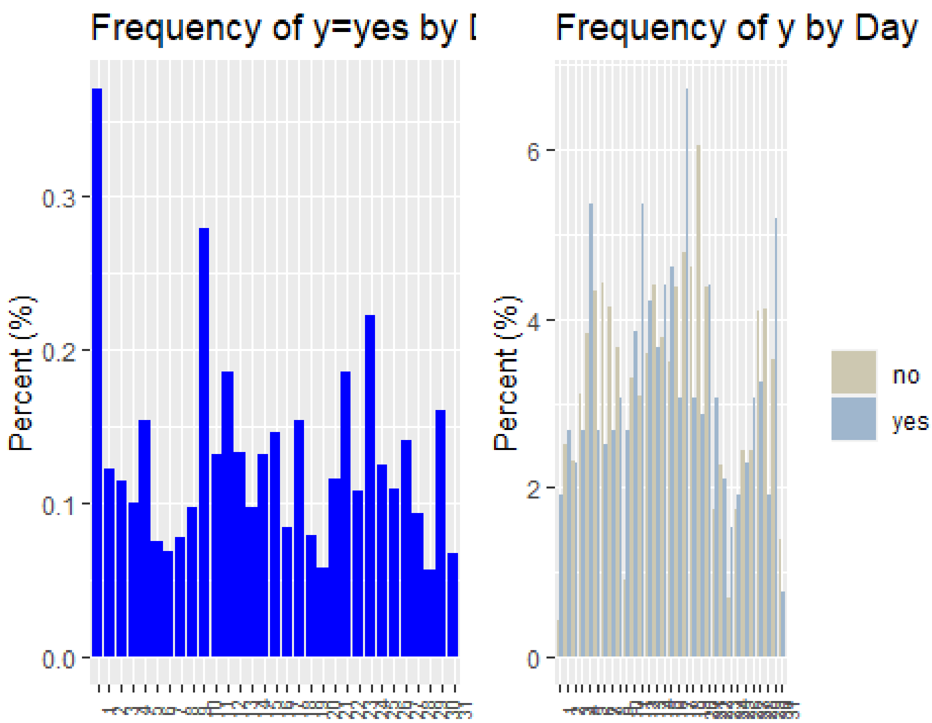




We can see difference in the distribution of yes and no, however, interpretation is not fully clear and further check will be required.

day

```
## variable= Day
##
##          1    2    3    4    5    6    7    8    9   10   11   12   13
14
##   no  0.43 2.50 2.33 3.12 3.82 4.32 4.42 4.15 3.67 0.90 3.30 3.08 3.60
4.40
##   yes 1.92 2.69 2.30 2.69 5.37 2.69 2.50 2.69 3.07 2.69 3.84 5.37 4.22
3.65
##
##         15   16   17   18   19   20   21   22   23   24   25   26   27
28
##   no  3.78 3.50 4.38 4.78 4.62 6.05 4.38 1.75 2.28 0.70 1.75 2.45 2.43
4.10
##   yes 4.41 4.61 3.07 6.72 3.07 2.88 4.41 3.07 2.11 1.54 1.92 2.30 3.07
3.26
##
##         29   30   31
##   no  4.12 3.52 1.38
##   yes 1.92 5.18 0.77
```

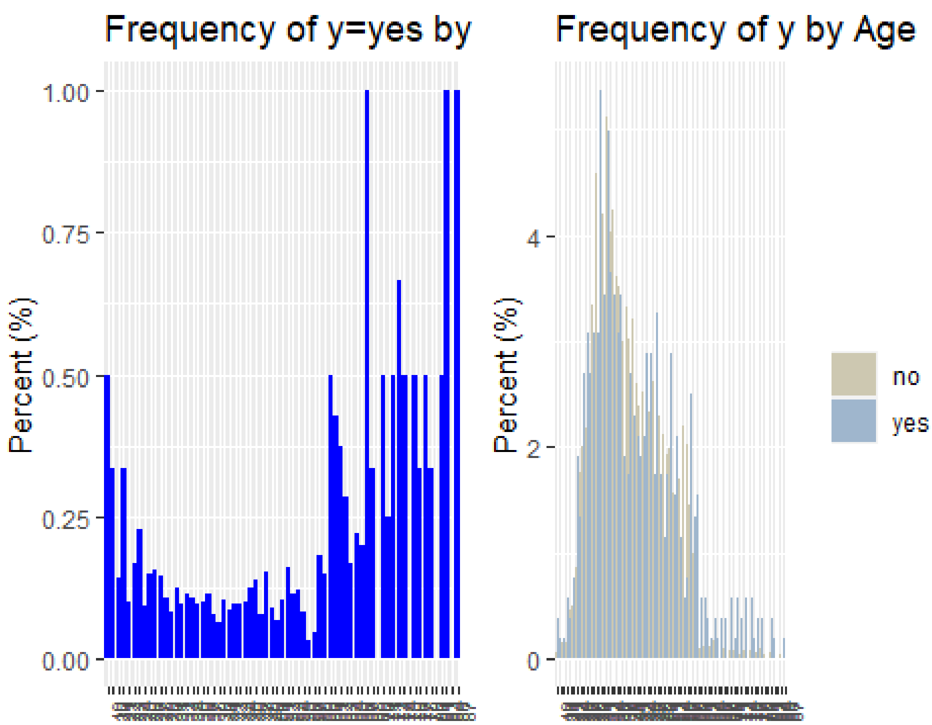


We can see difference in the distribution of yes and no, however, interpretation is not fully clear and further check will be required.

age

```
## variable= Age
##
##         19   20   21   22   23   24   25   26   27   28   29   30   31
32
##   no  0.05 0.05 0.15 0.15 0.45 0.50 0.85 1.75 2.00 2.17 2.08 3.35 4.58
4.90
##   yes 0.38 0.19 0.19 0.58 0.38 0.77 1.92 1.34 2.69 3.07 2.69 3.07 3.07
5.37
##
##         33   34   35   36   37   38   39   40   41   42   43   44   45
46
##   no  4.20 5.12 4.03 4.25 3.62 3.52 3.00 3.33 3.02 3.22 2.60 2.38 2.53
2.60
##   yes 3.45 4.99 3.65 3.45 3.07 3.45 1.92 1.73 2.69 2.30 2.11 1.92 2.11
2.88
##
##         47   48   49   50   51   52   53   54   55   56   57   58   59
60
##   no  2.33 2.62 2.38 2.30 2.12 1.93 1.98 1.57 1.98 1.70 2.20 2.02 1.45
1.00
##   yes 2.88 1.73 3.26 1.73 1.15 1.73 2.88 1.54 2.11 1.15 0.58 0.77 2.50
1.34
```

```
##
##         61   62   63   64   65   66   67   68   69   70   71   72   73
74
##   no  0.20 0.10 0.12 0.12 0.12 0.18 0.10 0.00 0.10 0.18 0.07 0.07 0.07
0.03
##   yes 1.54 0.58 0.58 0.38 0.19 0.38 0.19 0.38 0.38 0.00 0.58 0.19 0.58
0.38
##
##         75   76   77   78   79   80   81   82   83   84   85   86   87
##   no  0.07 0.05 0.07 0.05 0.05 0.10 0.03 0.00 0.05 0.00 0.00 0.03 0.00
##   yes 0.58 0.00 0.58 0.19 0.38 0.38 0.00 0.00 0.38 0.19 0.00 0.00 0.19
```

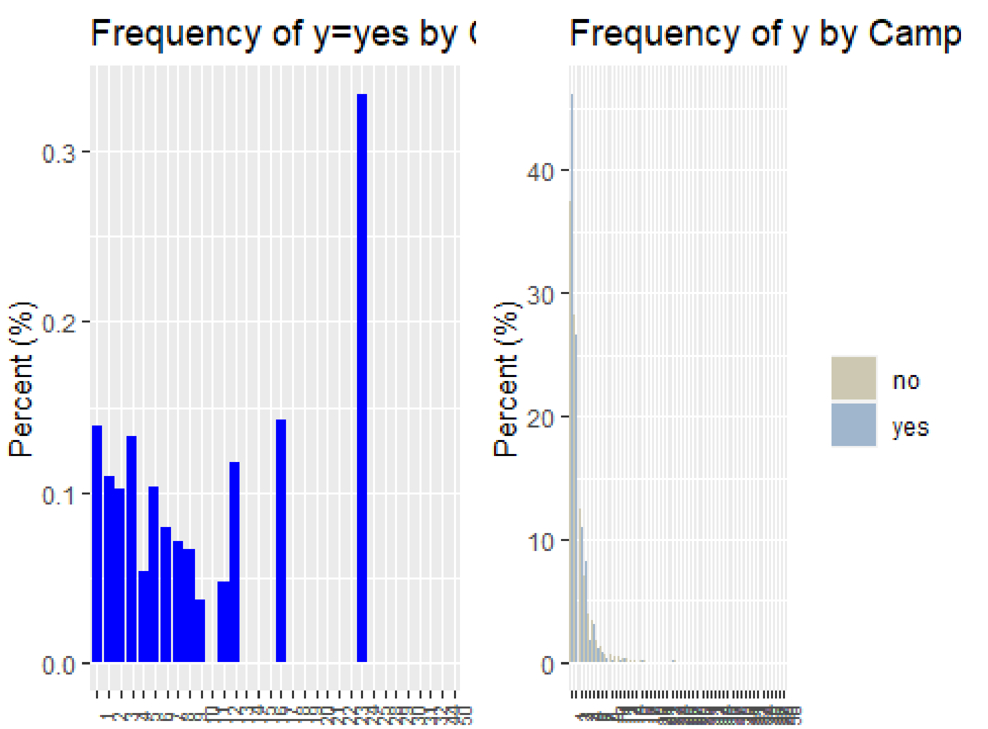


We can see difference in the distribution of yes and no, however, interpretation is not fully clear and further check will be required.

campaign

```
## variable= Campaign
##
##           1      2      3      4      5      6      7      8      9     10     11
12
##   no  37.35 28.15 12.53  7.05  3.95  3.48  1.73  1.30  0.70  0.65  0.55
0.50
##   yes 46.07 26.49 10.94  8.25  1.73  3.07  1.15  0.77  0.38  0.19  0.00
0.19
##
##          13     14     15     16     17     18     19     20     21     22     23
24
```

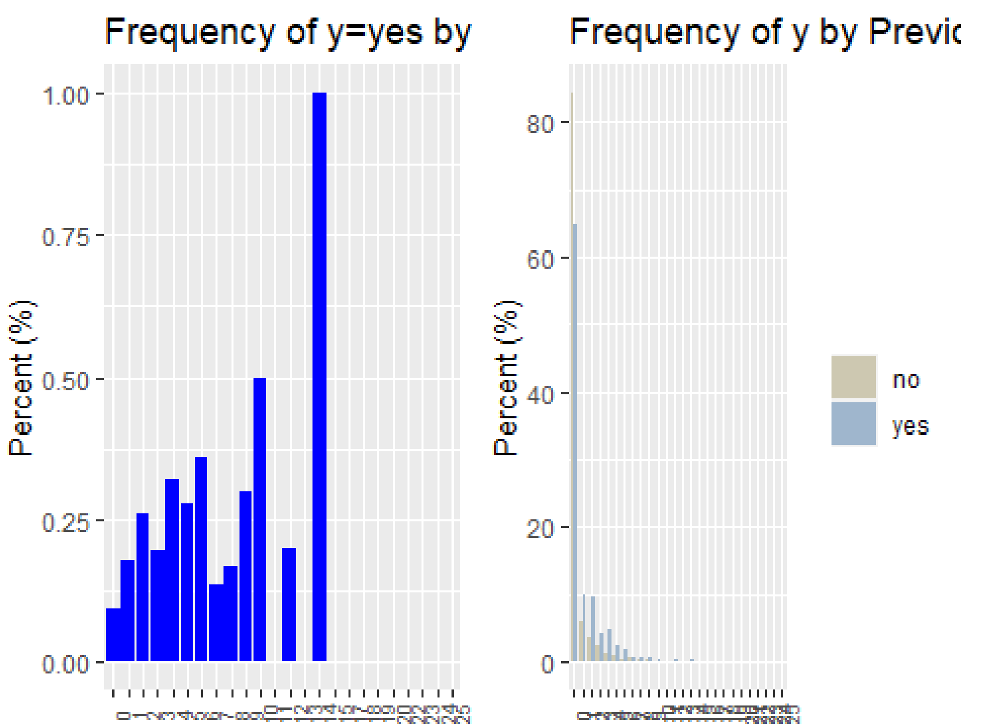
```
##    no    0.38  0.25  0.22  0.20  0.15  0.18  0.07  0.07  0.05  0.05  0.05
0.05
##    yes   0.38  0.00  0.00  0.00  0.19  0.00  0.00  0.00  0.00  0.00  0.00
0.19
##
##            25    26    27    28    29    30    31    32    33    34    35
36
##    no    0.10  0.00  0.00  0.07  0.03  0.03  0.03  0.05  0.00  0.00  0.00
0.00
##    yes   0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00
##
##            37    38    39    40    41    42    43    44    45    46    47
48
##    no    0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.03  0.00  0.00  0.00
0.00
##    yes   0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
0.00
##
##            49    50
##    no    0.00  0.03
##    yes   0.00  0.00
```



We can see difference in the distribution of yes and no, however, interpretation is not fully clear and further check will be required.

previous

```
## variable= Previous
##
##            0      1      2      3      4      5      6      7      8      9     10
11
##    no  84.20   5.88   3.57   2.28   1.32   0.85   0.40   0.48   0.38   0.18   0.05
0.07
##    yes 64.68   9.79   9.60   4.22   4.80   2.50   1.73   0.58   0.58   0.58   0.38
0.00
##
##           12     13     14     15     16     17     18     19     20     21     22
23
##    no   0.10   0.03   0.00   0.03   0.00   0.03   0.03   0.03   0.03   0.00   0.03
0.03
##    yes  0.19   0.00   0.38   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
0.00
##
##           24     25
##    no   0.03   0.03
##    yes  0.00   0.00
```



We can see difference in the distribution of yes and no, however, interpretation is not fully clear and further check will be required.

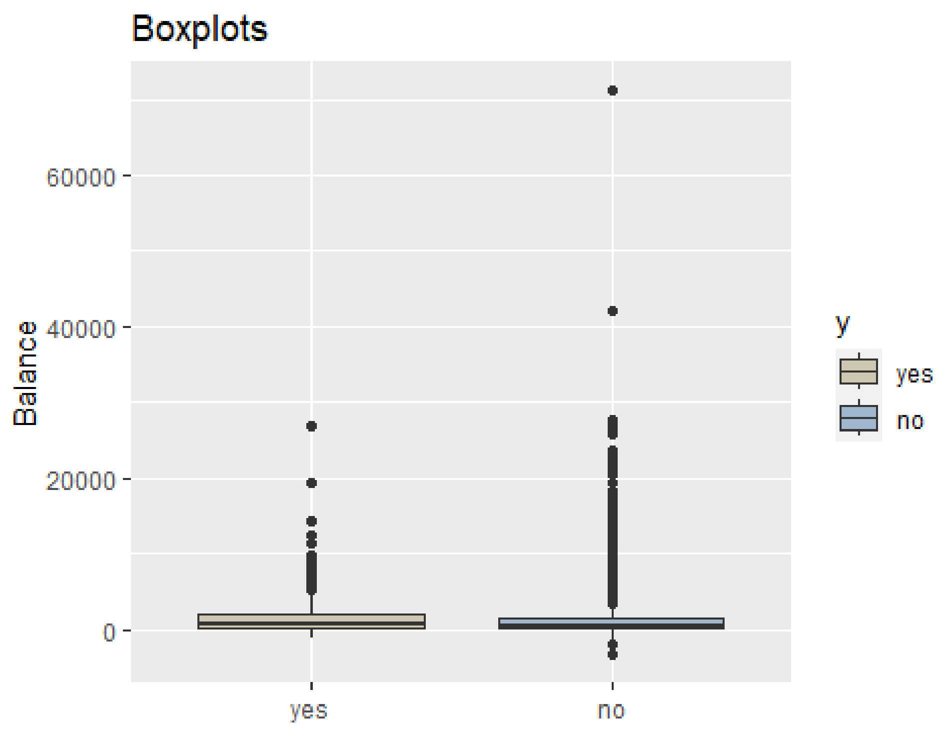
Variable vizualisation - continuous

Function

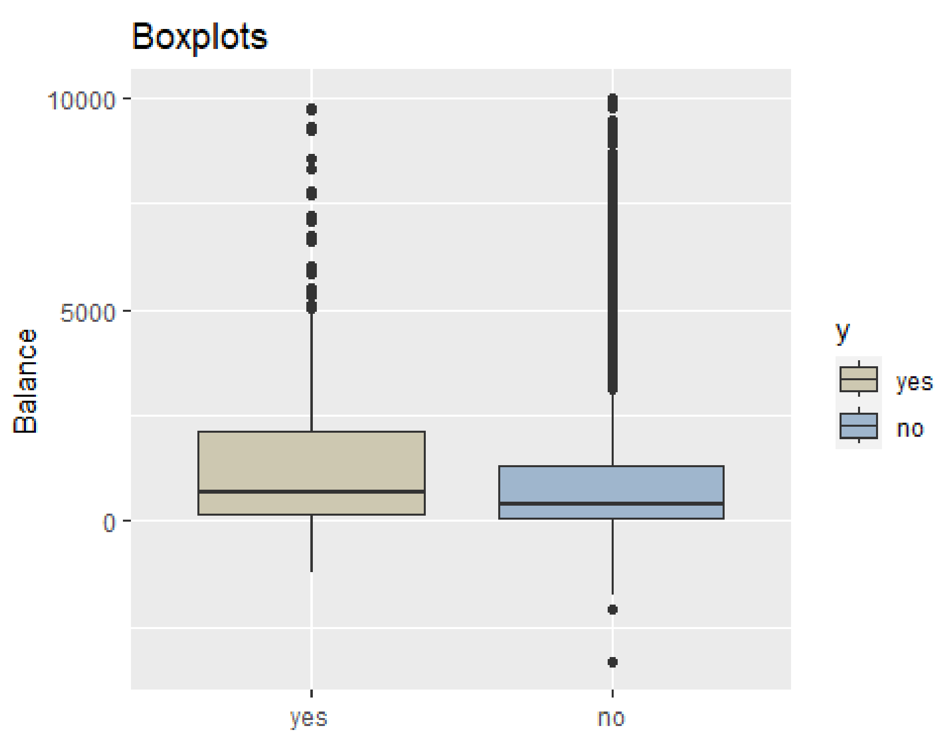
We elaborate a function for easier production of the graphs on continuous variable.

```
########## continuous variables : boxplot ###

onegraph_boxp <- function (namevar,labelvar, bank_score_) {
  data_score=bank_score_[,c(namevar,"y")]
  names(data_score) <- c("x","y")
  data_score$y <- factor(data_score$y,
                         levels=c("yes","no"))
  bp <- ggplot(data_score, aes(y, x))
  bp <- bp + geom_boxplot(fill = "#FFFFFF", color = "#FFFFFF")
  bp <- bp + geom_boxplot(aes(fill = y))
  bp <- bp + scale_fill_manual(values = c("#CDC8B1","#9FB6CD"))
  bp <- bp +labs(
    title = "Boxplots",
    x = " ",
    y = labelvar
  )
  bp <- bp + theme(legend.position = "right")
  # bp

  return (bp)
}

p <- onegraph_boxp("balance","Balance",bank_score)
plot(p)
```

Balance

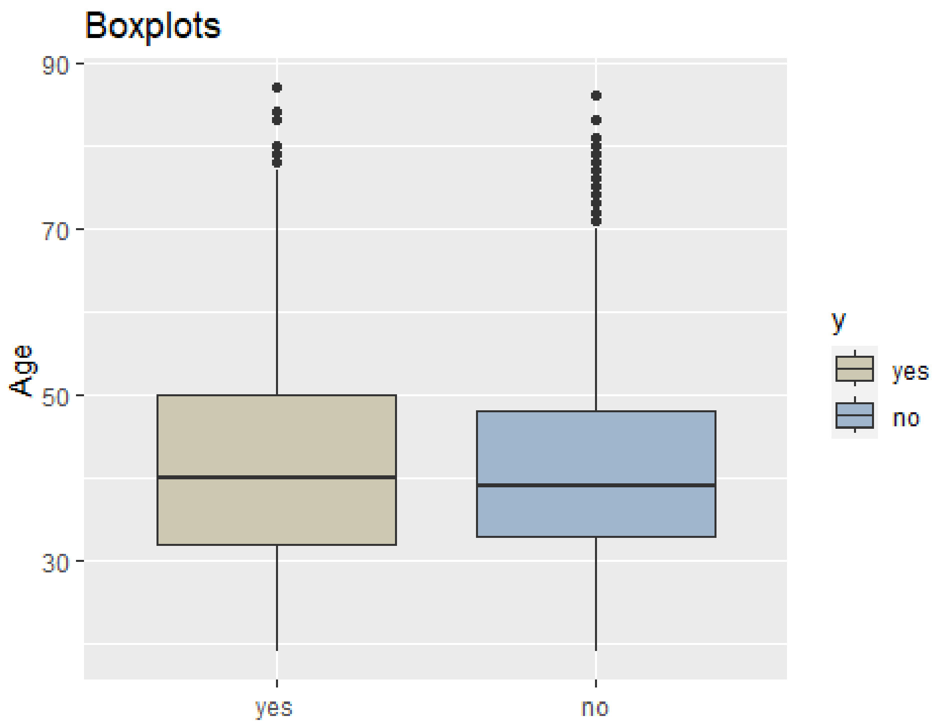


We should consider removing outliers.



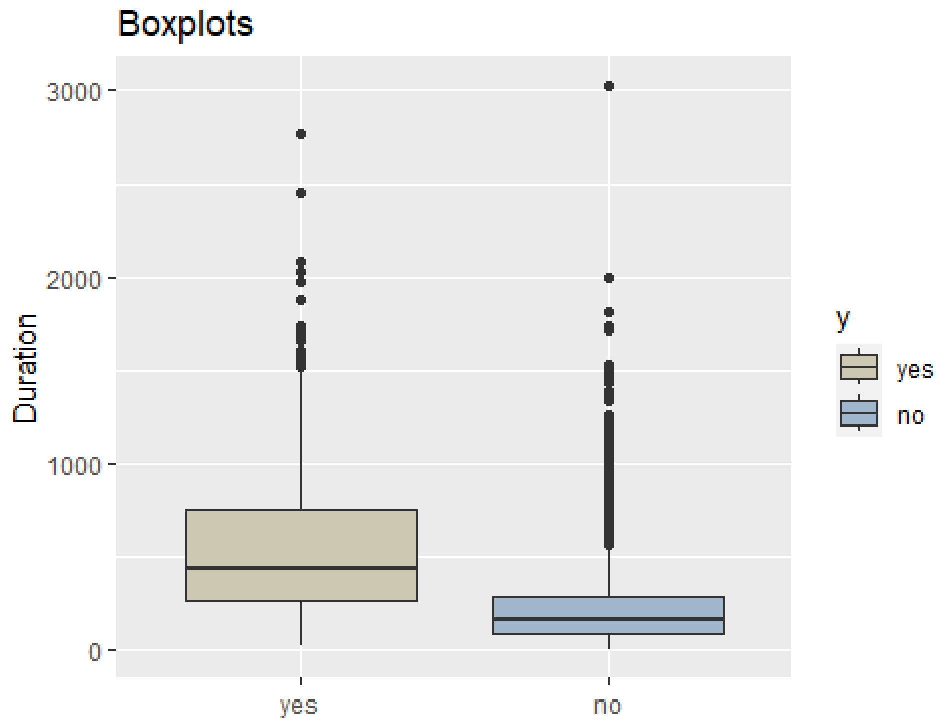
This variable should be tested further in the algorithms.

Age

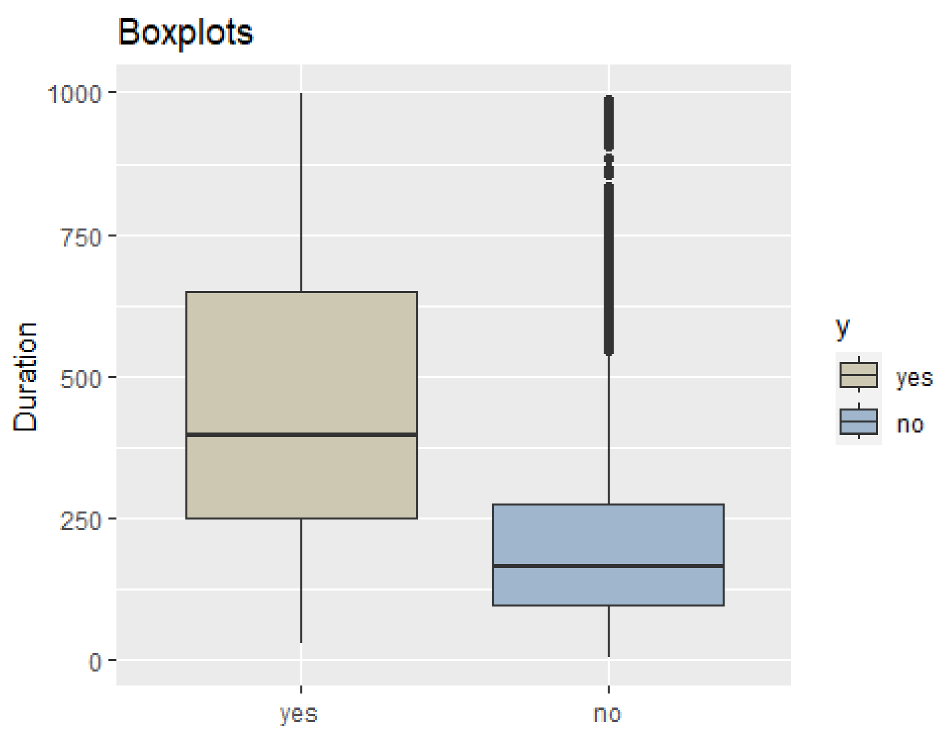


This variable should be tested further in the algorithms without outliers.

Duration

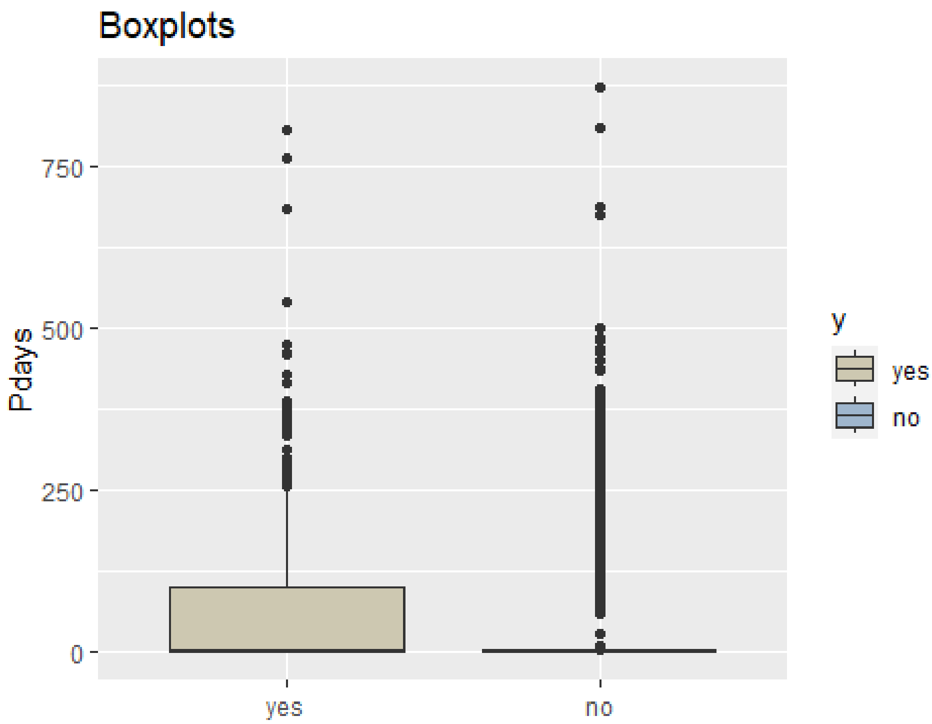


We should consider removing outliers beyond 1,000.

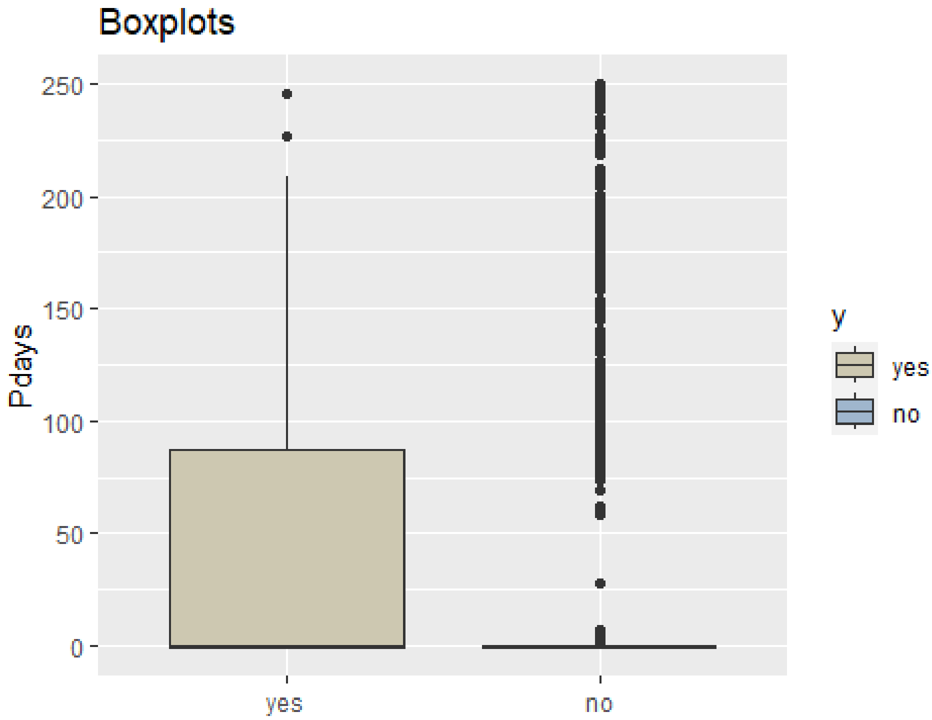


To be tested in the algorithms without outliers.

pdays

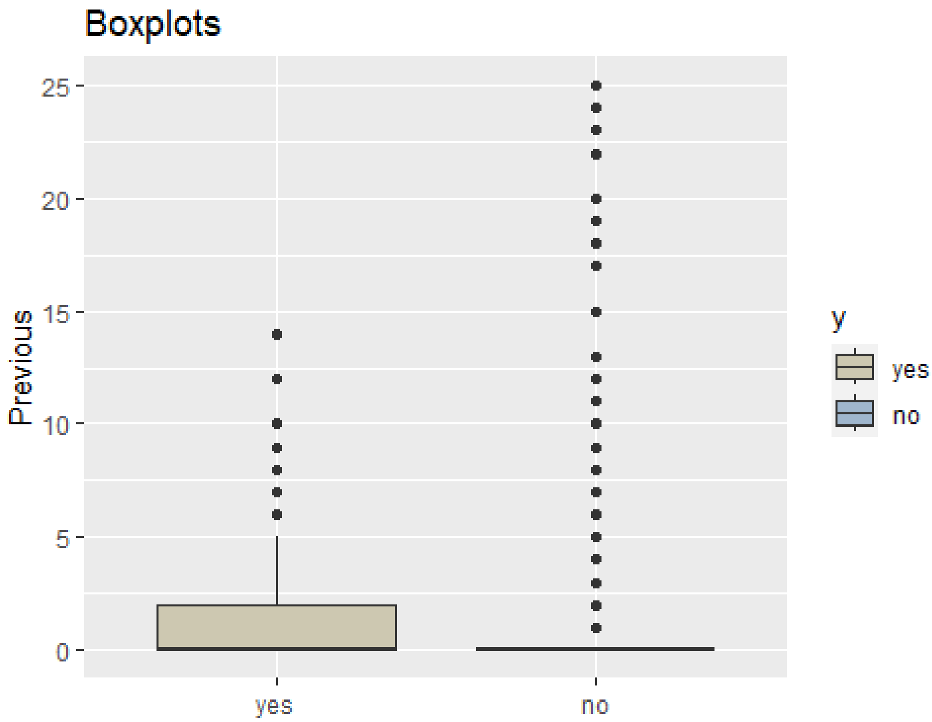


We test removing outliers beyond 250.

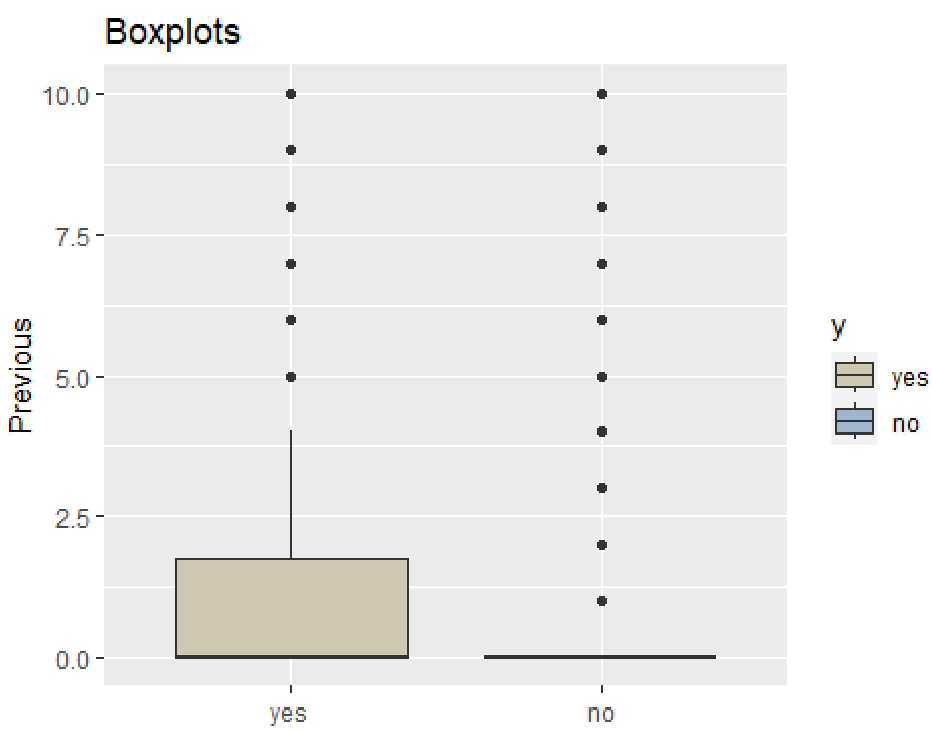


To be tested further in the algorithms without outliers.

previous



We test removing outliers beyond 250.



Variable to be tested further in the algorithms without outliers beyond 10.

2.2 – Preparing the sample

Split bank_score between bank_score_set and finalholdout_set

We split the bank_score set between a bank_score_set (90%) and a final_holdout_set (10%) in order to have a separate data set on which to confirm the results of the final model.

Number of row in bank_score_set + finalholdout - number of row in bank_score.

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

Split bank_score_set between train and test set

We need to test several models of algorithm, therefore we will train all the models on the train set and test them on the 10% put aside here.

Number of row in train_set + test_set - number of row in bank_score_set.

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

Build a balanced sample

Given that the proportion of yes is very small, building a model which correctly predict them will require oversampling or undersampling for constructing a balanced dataset and avoid bias training towards the larger class. For this data, undersampling has performed better than oversampling. We use the function "ovun.sample" from the package ROSE which gave the best result in comparison with a first direct implementation.

The number of "yes" and "no" in the train set are:

```
## [1] 469

## [1] 3600
```

The number of "yes" and "no" in the balanced set are:

```
##
##  no yes
## 469 469
```

2.3 – Training and testing the models

Removing the outliers from the dataset

We remove, on the train set and the balance set, all values of the variable "previous" above 10, value of "balance" above 10,000, value of "duration" above 1,000, values of "pdays" above 250. The number of rows in train_set and train_balanced (469*2) is thus now:

```
## [1] 4069

## [1] 938
```

Function for automatically training and testing the algorithms

In order to make training and testing of different algorithms easier, the following formula has been implemented to train the model and to test.

```
#Function to automatically compute accuracy, error and AUC
functionerror <- function(train_glm,train_set,test_set, ifcat=TRUE) {
  glm_pred_train <- predict(train_glm, train_set)
  glm_pred_test  <- predict(train_glm, test_set)
  if (ifcat) {
  cat("______________________________________\n")
  cat(paste(" ",train_glm$method,"  ",
            as.character(train_glm$call)[2],sep=""),"\n",
      paste(train_glm$method,"-> accuracy train = ",sep=""),
      round(mean(glm_pred_train==train_set$y),4),"\n",
      paste(train_glm$method,"-> accuracy test  = ",sep=""),
      round(mean(glm_pred_test==test_set$y),4),"\n")
  }
  tables2 = cbind(table(glm_pred_train,train_set$y),
                  table(glm_pred_test,test_set$y))
  if (ifcat) {
    cat("--------------------------------------\n")
    cat(paste(" ",train_glm$method,"-> confusion matrices
(train|test)\n",sep=""))
    print(tables2)
  }
  roc.out <- roc( as.integer(test_set$y=="yes"),
                  as.integer(as.character(glm_pred_test)=="yes"))
  auc.out = auc(roc.out)
  if (ifcat) {
    cat("AUC =",as.numeric(auc.out),"\n")
    cat("______________________________________\n")
  }
  return (list(glm_pred_train=glm_pred_train,glm_pred_test=glm_pred_test,
               train_glm=train_glm,train_set=train_set,test_set=test_set,
               auc.out=auc.out))
}
```

Model 1 - GLM model

variable job - education - marital - housing- loan – month

We start with the variables which were the most promising from the graphs.

```
## ______________________________________
##  glm  y ~ job + education + marital + housing + loan + month
##  glm-> accuracy train =  0.899
##  glm-> accuracy test  =  0.8761
## --------------------------------------
##  glm-> confusion matrices
## (train|test)
```

```
##       no yes  no yes
## no  3201 336 395  51
## yes   27  30   5   1

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.5033654
## ______________________________________
```

The model focuses on {y=0} because the imbalance.We will try with the balanced set to assess if the prediction of true positive can be improved.

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## ______________________________________
##  glm  y ~ job + education + marital + housing + loan + month
##  glm-> accuracy train =  0.6943
##  glm-> accuracy test  =  0.6792
## --------------------------------------
##  glm-> confusion matrices
## (train|test)
##      no yes  no yes
## no  323 137 276  21
## yes 106 229 124  31

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.6430769
## ______________________________________
```

Prediction for {y=1} is improved, we will thus focus on the balanced sample.

test day

We assess if adding the variable "day" improves the prediction.

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## ______________________________________
##  glm  y ~ job + education + marital + housing + loan + month + day
##  glm-> accuracy train =  0.6967
##  glm-> accuracy test  =  0.6836
## --------------------------------------
##  glm-> confusion matrices
## (train|test)
##       no yes  no yes
```

```
## no  2272 134 279  22
## yes  956 232 121  30

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.6372115
## ______________________________________
```

AUC decreases, we will not keep "day".

test poutcome

We assess if adding "poutcome" improves the prediction.

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## ______________________________________
##  glm  y ~ job + education + marital + housing + loan + month + poutcome
##  glm-> accuracy train =  0.749
##  glm-> accuracy test  =  0.7323
## --------------------------------------
##  glm-> confusion matrices
## (train|test)
##       no yes  no yes
## no  2471 145 299  20
## yes  757 221 101  32

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.6814423
## ______________________________________
```

AUC improves, we keep "poutcome" as a variable.

test age

We assess if adding "age" improves the prediction.

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## ______________________________________
##  glm  y ~ job + education + marital + housing + loan + month + poutcome +
age
```

```
##  glm-> accuracy train =  0.7371
##  glm-> accuracy test  =  0.7235
## --------------------------------------
##  glm-> confusion matrices
## (train|test)
##       no yes  no yes
## no  2424 141 295  20
## yes  804 225 105  32

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.6764423
## ______________________________________
```

AUC improves, we keep “age” as a variable.

test campaign

We assess if adding “campaign” improves the prediction.

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## ______________________________________
##  glm  y ~ job + education + marital + housing + loan + month + poutcome +
age + campaign
##  glm-> accuracy train =  0.7412
##  glm-> accuracy test  =  0.7212
## --------------------------------------
##  glm-> confusion matrices
## (train|test)
##       no yes  no yes
## no  2440 142 295  21
## yes  788 224 105  31

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.6668269
## ______________________________________
```

AUC decreases, “campaign” not kept as a variable.

test previous

We assess if adding previous improves the prediction.

```
## ______________________________________
##  glm  y ~ job + education + marital + housing + loan + month + poutcome +
age + previous
##  glm-> accuracy train =  0.7359
```

```
##  glm-> accuracy test  =  0.7168
## ------------------------------------
##  glm-> confusion matrices
## (train|test)
##       no yes  no yes
## no  2418 139 291  19
## yes  810 227 109  33

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.6810577
## ____________________________________
```

Accuracy increases, we keep the variable "previous".

test default

We assess if adding the variable "default" improves the prediction.

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in preProcess.default(thresh = 0.95, k = 5, freqCut = 19,
uniqueCut =
## 10, : These variables have zero variances: jobunknown

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful
cases

## ____________________________________
##  glm  y ~ job + education + marital + housing + loan + month + poutcome +
age + previous + default
##  glm-> accuracy train =  0.7329
##  glm-> accuracy test  =  0.7168
## ------------------------------------
##  glm-> confusion matrices
## (train|test)
##       no yes  no yes
## no  2408 140 292  20
## yes  820 226 108  32

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.6726923
## ____________________________________
```

The variable “default” increases AUC, we will keep this variable.

test balance

We assess if adding balance improves the prediction.

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## ______________________________________
##  glm  y ~ job + education + marital + housing + loan + month + poutcome +
age + previous + default + balance
##  glm-> accuracy train =  0.7457
##  glm-> accuracy test  =  0.7257
## --------------------------------------
##  glm-> confusion matrices
## (train|test)
##        no yes  no yes
## no   2456 142 297  21
## yes   772 224 103  31

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.6693269
## ______________________________________
```

The variable “balance” does not increase AUC, balance not kept.

test duration

We assess if adding the variable “duration” improves the prediction.

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## ______________________________________
##  glm  y ~ job + education + marital + housing + loan + month + poutcome +
age + previous + default + duration
##  glm-> accuracy train =  0.835
##  glm-> accuracy test  =  0.8053
## --------------------------------------
##  glm-> confusion matrices
## (train|test)
##        no yes  no yes
## no   2711  76 324  12
## yes   517 290  76  40
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.7896154
## ______________________________________

##
## Call:
## NULL
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          0.02419    6.59484   0.004 0.997073
## `jobblue-collar`    -0.37756    0.16728  -2.257 0.024008 *
## jobentrepreneur     -0.04508    0.11296  -0.399 0.689850
## jobhousemaid        -0.12681    0.12549  -1.011 0.312233
## jobmanagement       -0.26596    0.18379  -1.447 0.147878
## jobretired           0.15131    0.15285   0.990 0.322205
## `jobself-employed`  -0.15468    0.12354  -1.252 0.210530
## jobservices         -0.26188    0.13992  -1.872 0.061264 .
## jobstudent           0.03359    0.11657   0.288 0.773227
## jobtechnician       -0.29045    0.16196  -1.793 0.072926 .
## jobunemployed       -0.27493    0.13665  -2.012 0.044230 *
## jobunknown           0.07952    0.12777   0.622 0.533705
## educationsecondary   0.12284    0.19396   0.633 0.526514
## educationtertiary    0.27216    0.20450   1.331 0.183243
## educationunknown    -0.17885    0.13432  -1.332 0.183002
## maritalmarried      -0.19173    0.17254  -1.111 0.266472
## maritalsingle       -0.07819    0.18335  -0.426 0.669782
## housingyes          -0.30370    0.12154  -2.499 0.012462 *
## loanyes             -0.41663    0.12650  -3.293 0.000990 ***
## monthaug            -0.29786    0.14896  -2.000 0.045543 *
## monthdec             1.53384   65.44173   0.023 0.981301
## monthfeb             0.03766    0.12366   0.305 0.760700
## monthjan            -0.22110    0.11360  -1.946 0.051632 .
## monthjul            -0.48041    0.15199  -3.161 0.001574 **
## monthjun            -0.07116    0.12959  -0.549 0.582936
## monthmar             0.36331    0.16446   2.209 0.027172 *
## monthmay            -0.81065    0.17711  -4.577 4.72e-06 ***
## monthnov            -0.16891    0.13258  -1.274 0.202659
## monthoct             0.52813    0.17506   3.017 0.002554 **
## monthsep             0.09835    0.12763   0.771 0.440975
## poutcomeother        0.11135    0.12196   0.913 0.361247
## poutcomesuccess      0.79883    0.20695   3.860 0.000113 ***
## poutcomeunknown     -0.17842    0.21233  -0.840 0.400734
## age                 -0.25293    0.15649  -1.616 0.106039
## previous             0.03746    0.18770   0.200 0.841803
## defaultyes           0.17141    0.11006   1.557 0.119377
## duration             1.61758    0.13493  11.988  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1097.11  on 794  degrees of freedom
## Residual deviance:  593.15  on 758  degrees of freedom
## AIC: 667.15
##
## Number of Fisher Scoring iterations: 15
```

The variable "duration" improves AUC, duration is kept as a variable. This model will be kept as the best glm, hence the regression coefficients beta are shown.Data have been normalized. The coefficient "month dec" seens to be inconsistent due to the characteristics of this specific sample.

test pday

We assess if adding the variable "pday" improves the prediction.

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## ______________________________________
##  glm  y ~ job + education + marital + housing + loan + month + poutcome +
age + previous + default + duration + pdays
##  glm-> accuracy train =  0.8306
##  glm-> accuracy test  =  0.8119
## --------------------------------------
##  glm-> confusion matrices
## (train|test)
##       no yes  no yes
## no   2696  77 328  13
## yes   532 289  72  39

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.785
## ______________________________________
```

The variable "pday" decreases AUC and is not kept.

GLM conclusion

The best set of variable has been identified as job + education + marital + housing + loan + month+ poutcome+ age+ previous + default + duration.

Model 2 - knn model

We will also proceed with testing knn on the variable visually identified and test additional variable.

test job + education + marital + housing + loan + month+ poutcome+ age+ previous + default + duration

We try with the variable identified as best performing with glm

```
## ______________________________________
## knn y ~ job + education + marital + housing + loan + month + poutcome +
age + previous + default + duration
## knn-> accuracy train = 0.7593
## knn-> accuracy test = 0.7367
## --------------------------------------
## knn-> confusion matrices
## (train|test)
##        no yes  no yes
## no   2473 110 296  15
## yes   755 256 104  37

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.7257692
## ______________________________________
```

test day

Does the variable “day” helps improving the AUC with knn.

```
## ______________________________________
## knn y ~ job + education + marital + housing + loan + month + poutcome +
age + previous + default + duration + day
## knn-> accuracy train = 0.776
## knn-> accuracy test = 0.7478
## --------------------------------------
## knn-> confusion matrices
## (train|test)
##        no yes  no yes
## no   2538 115 301  15
## yes   690 251  99  37

## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases

## AUC = 0.7320192
## ______________________________________
```

The variable “day” decreases AUC.

test campaign

Does “campaign” help improve the AUC with knn?

```
## ______________________________________
##  knn  y ~ job + education + marital + housing + loan + month + poutcome + 
age + previous + default + duration + campaign
##  knn-> accuracy train =  0.7557
##  knn-> accuracy test  =  0.7301
## --------------------------------------
##  knn-> confusion matrices
## (train|test)
##       no yes  no yes
## no  2454 104 293  15
## yes  774 262 107  37

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.7220192
## ______________________________________
```

The variable “campaign” decreases AUC.

test balance

Does “balance” help improve the AUC with knn?

```
## ______________________________________
##  knn  y ~ job + education + marital + housing + loan + month + poutcome + 
age + previous + default + duration + balance
##  knn-> accuracy train =  0.7323
##  knn-> accuracy test  =  0.6969
## --------------------------------------
##  knn-> confusion matrices
## (train|test)
##       no yes  no yes
## no  2359  93 279  16
## yes  869 273 121  36

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```
## AUC = 0.6949038
## ______________________________________
```

balance decreases AUC

test pday

Does “pday” improve the AUC with knn?

```
## ______________________________________
##  knn  y ~ job + education + marital + housing + loan + month + poutcome + age + previous + default + duration + pdays
##  knn-> accuracy train =  0.7896
##  knn-> accuracy test  =  0.7633
## --------------------------------------
##  knn-> confusion matrices
## (train|test)
##       no yes  no yes
## no  2563  91 302   9
## yes  665 275  98  43

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.7909615
## ______________________________________
```

“pday” improves AUC

optimization of parameter with cross validation

With the best set of variable identified for knn (job + education + marital + housing + loan + month+ poutcome+ age+ previous + default + duration+pdays), we now optimize for the number of neighboors with a grid search and perform a more elaborate cross-validation procedure (average from 5 cv and 15 folds for each cv among the 5).

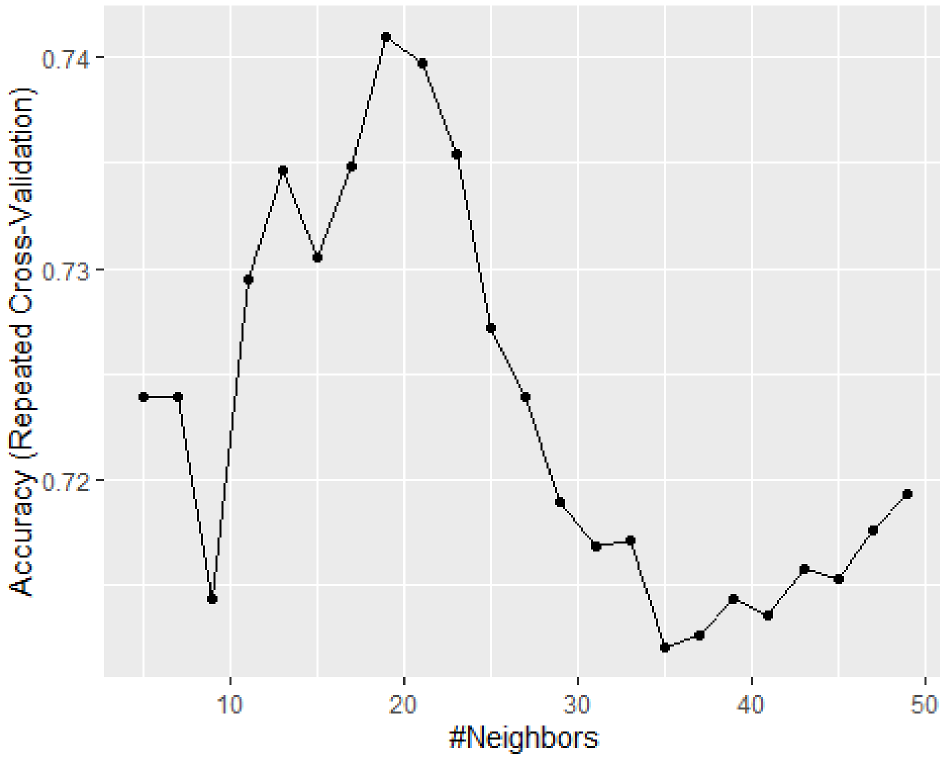
```
train_knn_cv_bal <- train(y ~ job + education + marital + housing + loan +
month+ poutcome+ age+ previous + default + duration+pdays,
                           method="knn",
                           data = train_balanced,
                           tuneGrid = data.frame(k=seq(5,50,2)),
                           trControl = trainControl(method="repeatedcv",
                                                    number = 15,
                                                    repeats = 5),
                           preProcess = c("center","scale"),
                           metric="Accuracy")
```

What is the best parameter of the knn?

```
##    k
## 8 19
```

```
## [1] 19
```

What are the results??




```
## ______________________________________
##  knn  y ~ job + education + marital + housing + loan + month + poutcome + age + previous + default + duration + pdays
##  knn-> accuracy train =  0.8322
##  knn-> accuracy test  =  0.8164
## --------------------------------------
##  knn-> confusion matrices
## (train|test)
##        no yes  no yes
## no  2762 137 329  12
## yes  466 229  71  40

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.7958654
## ______________________________________
```

conclusion knn

Optimizing variables and parameters helped improve the prediction, however predictive power is only slightly better than glm which reached AUC of 0.7896.

Model 3- random forest (rf)

test job + education + marital + housing + loan + month+ poutcome+ age+ previous + default + duration

We test with previously identified variable.

```
## _______________________________________
##  rf  y ~ job + education + marital + housing + loan + month + poutcome + age + previous + default + duration
##  rf-> accuracy train =  1
##  rf-> accuracy test  =  0.7942
## ---------------------------------------
##  rf-> confusion matrices
## (train|test)
##      no yes  no yes
## no  429   0 317  10
## yes   0 366  83  42

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.8000962
## _______________________________________
```

rf is promising

test day

```
## _______________________________________
##  rf  y ~ job + education + marital + housing + loan + month + poutcome + age + previous + default + duration + day
##  rf-> accuracy train =  1
##  rf-> accuracy test  =  0.8119
## ---------------------------------------
##  rf-> confusion matrices
## (train|test)
##      no yes  no yes
## no  429   0 321   6
## yes   0 366  79  46

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.8435577
## _______________________________________
```

The variable “day” improves AUC with rf.

test campaign

```
## ______________________________________
##  rf  y ~ job + education + marital + housing + loan + month + poutcome +
age + previous + default + duration + day + campaign
##  rf-> accuracy train =  1
##  rf-> accuracy test  =  0.8075
## --------------------------------------
##  rf-> confusion matrices
## (train|test)
##      no yes  no yes
## no  429   0 321   8
## yes   0 366  79  44

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.8243269
## ______________________________________
```

"campaign" decreases AUC.

test balance

```
## ______________________________________
##  rf  y ~ job + education + marital + housing + loan + month + poutcome +
age + previous + default + duration + day + balance
##  rf-> accuracy train =  1
##  rf-> accuracy test  =  0.8097
## --------------------------------------
##  rf-> confusion matrices
## (train|test)
##      no yes  no yes
## no  429   0 321   7
## yes   0 366  79  45

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.8339423
## ______________________________________
```

The variable "balance" decreases AUC.

test pday

```
## ______________________________________
##  rf  y ~ job + education + marital + housing + loan + month + poutcome +
age + previous + default + duration + day + pdays
##  rf-> accuracy train =  1
##  rf-> accuracy test  =  0.8186
## --------------------------------------
```

```
##  rf-> confusion matrices
## (train|test)
##      no yes  no yes
## no  429   0 324   6
## yes   0 366  76  46

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

## AUC = 0.8473077
## _______________________________________
```

The variable "pday" increases AUC

Model 4 optimized rf - loop to improve AUC of rf by optimization of the factor

In order to improve the tuning of the rf, we use the following code to cross validate with the rf and tune three hyperparameters. This is not run as part of this report in order to save time because it has already been run separately.

```
control <- trainControl(method='repeatedcv',
                        number=10,
                        repeats=10)
#Metric compare model is Accuracy
metric <- "Accuracy"
set.seed(num_seed)

nodesize.val.all  = c(1:5)
mtry.val.all      = c(1:7)
num.trees.val.all = c(50,100,150,200,250)
nb.models         = length(mtry.val.all)*
  length(num.trees.val.all)*
  length(nodesize.val.all)

AUC.all = matrix(0,nrow = nb.models, ncol=4)
colnames(AUC.all) <- c("mtry","num.trees","nodesize","auc")

model_all= list()

m= 0
sink("rf_boucle_ouput.txt")

for (nodesize.val in nodesize.val.all) {
  for (mtry.val in mtry.val.all) {
    for (num.trees.val in num.trees.val.all) {
      m = m+1
      cat("m=",m,"/",nb.models,"\n")
      cat("running random forest with mlty =",mtry.val,
          "  num.trees=",num.trees.val,"nodesize.val =", nodesize.val,"\n")
      tune.grid <- expand.grid(mtry=c(mtry.val))
```

```
      train_rf_try <- train(y~job + education + marital + housing + loan +
month+ poutcome+ age+ previous + default + duration + day+ pdays,
                            data=train_balanced,#[,c(list_vars,"y")],
                            method='rf',
                            metric='Accuracy',
                            tunegrid=tune.grid,
                            #num.trees = num.trees.val,
                            ntree = num.trees.val,
                            nodesize = nodesize.val,
                            trControl=control)
      model_all[[m]]=train_rf_try
      rf_try = functionerror(train_rf_try,train_balanced,test_set,
                             ifcat = TRUE)
      roc.out <- roc( as.integer(test_set$y=="yes"),
                      as.integer(as.character(rf_try$glm_pred_test)=="yes"))
      auc.out = auc(roc.out)
      cat("mlty =",mtry.val,"  num.trees =",num.trees.val,
          "nodesize.val =", nodesize.val,
          "   ", "AUC =",as.numeric(auc.out),"\n")
      #cat("--------------------n")
      cat("\n")

      AUC.all[m,] =
c(mtry.val,num.trees.val,nodesize.val,as.numeric(auc.out))

    }
  }
}
#close
sink()
```

From the output file, We see that the best parameter is the 117 run.

Part 3 - results

result of best rf tuning

from the file analysis, the best model is 117

```
#from the file analysis, the best model is 117
model_best_rf <- model_all[[117]]
model_best_rf
rf_best = functionerror(model_best_rf,train_balanced,test_set, ifcat = TRUE)
# AUC = 0.8579106
```

The result on the test set is an AUC of 0.8579.

result on final holdout

```
#what is the score on the holdout
rf_best_holdout =
```

```r
functionerror(model_best_rf,train_balanced,final_holdout_set, ifcat = TRUE)
#AUC = 0.7804808
```

the result on the final holdout is an AUC of 0.7804.

Part 4 - conclusion

The best prediction are is offered by the random forest after optimization of three hyperparameters from cross-validation. Then, the best AUC from knn was 0.7958 on the test. For glm it reached 0.7896. Hence, this is the tuned random forest which is used with the final dataset for validation, with finally an AUC equal to 0.7804.

The (small) difference of AUC between training and validation may come from the size of the training dataset from undersampling, due to imbalanced original dataset.The model might also be slightly overfit. Another reason may be that the chosen hyperparameters and set of kept variables may be only near the optimal choice. Another limit is the spliting which is unique, hence an averaging may be better but time consuming. A better model may be also possible by more recoding the variables, more tuning the outliers removal or changing to a neural network with deep learning available in r language.

Part 5 - reference

The dataset comes from Kaggle
https://www.kaggle.com/datasets/kapturovalexander/bank-credit-scoring

Under sampling was made thanks to the ovun.sample function from the ROSE library.

The AUC was calculated thanks to the pROC library.