Bank Marketing Campaign

Miguel Ángel Jiménez Sánchez

May the 30th, 2019

1. Overview

1.1 Objective

Our goal on this project is to study a dataset generated in a marketing campaign. This campaign was performed by a Portuguese bank between 2008 and 2010 to offer a term deposit.

We will try to obtain insights to optimize the resources of the marketing department. We will construct as well a predictive model to predict which contacts or calls (I'll use both terms) may result on the subscription to the product that we are offering.

1.2 The dataset

The dataset used on this project is based on "Bank Marketing" UCI dataset and is publicly available for research. It can be downloaded from http://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip.

The zip file contains two csv files. The first (bank-additional-full.csv) contains the full data set. The second file (bank-additional.csv) contains a sample of 10% of the former. The zip file contains as well a txt file with information about the dataset and a description of the fields.

We will use only the file "bank-additional-full.csv". Each row represents a contact with a customer. The dataset contains 41188 observations and 21 variables, including our target variable 'y'. This variable is categorical with two possible values "yes" or "not", depending on whether the customer subscribes the term deposit or not.

The rest of the variables are described on the txt file. These variables can be divided in four groups:

1. Bank client data

- age (numeric)
- job: Type of job (categorical)
- marital: marital status (categorical)
- education (categorical)
- default: has credit in default? (categorical)
- housing: has housing loan? (categorical)
- loan: has personal loan? (categorical)

2. Last contact of the current campaign

- contact: contact communication type (categorical: "cellular","telephone")
- month (categorical)
- day_of_week (categorical)
- duration: duration of call in seconds (numerical)

3. Social and economic context attributes

- emp.var.rate: employment variation rate - quarterly indicator (numeric)
- cons.price.idx: consumer price index - monthly indicator (numeric)

- cons.conf.idx: consumer confidence index - monthly indicator (numeric)

- euribor3m: euribor 3 month rate - daily indicator (numeric)
- nr.employed: number of employees - quarterly indicator (numeric)

4. Other attributes regarding previous contacts with the client

- campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
- 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)
- previous: number of contacts performed before this campaign and for this client (numeric)
- poutcome: outcome of the previous marketing campaign (categorical

5. The output variable y: has the client subscribed a term deposit? (binary: "yes","no")

As we see on this dataset we have both categorical and numeric variables.

1.3 Main steps

The first step will be an exploratory data analysis. From this data analysis we'll get our first insights on this data set and we'll determine if some preprocessing is needed.

After that, we'll train 6 different algorithms with this algorithm: glm, glmnet, Naive Bayes, Knn, Rborist and SVM radial. Finally, we'll ensemble the probabilities obtained from these 6 algorithms into our final model.

2. Analysis/ Methods

The intent in this chapter is not doing an exhaustive review of all the variables on the dataset. It will be centered on the variables with the higher (or lower) predictive power and the variables that may need some preprocessing.

2.1 Duration variable

One of the first decisions on this dataset is removing the column 'Duration', which describes the duration of the call. Actually some tests that I've done result that this is a variable that helps greatly in predicting our target variable y.

But it has a big inconvenient: we know the duration of the call at the same time than our target variable (indeed, if our call operators are polite enough to say "thanks" and "goodbye", the duration is known later than y). So this is not a variable that we can use for our objective to predict the subscriptions, as it is warned on the txt file included beside the csv file.

2.2 Summary of the data

With the next command we'll see a summary of our data:

```
summary(calls)
```

```
##       age                 job            marital
##  Min.   :17.00   admin.     :10422   divorced: 4612
##  1st Qu.:32.00   blue-collar: 9254   married :24928
##  Median :38.00   technician : 6743   single  :11568
##  Mean   :40.02   services   : 3969   unknown :   80
##  3rd Qu.:47.00   management : 2924
##  Max.   :98.00   retired    : 1720
##                  (Other)    : 6156
##                education        default         housing
##  university.degree  :12168   no     :32588   no     :18622
##  high.school        : 9515   unknown: 8597   unknown:  990
##  basic.9y           : 6045   yes    :    3   yes    :21576
##  professional.course: 5243
##  basic.4y           : 4176
##  basic.6y           : 2292
##  (Other)            : 1749
##       loan            contact          month       day_of_week
##  no     :33950   cellular :26144   may    :13769   fri:7827
##  unknown:  990   telephone:15044   jul    : 7174   mon:8514
##  yes    : 6248                     aug    : 6178   thu:8623
##                                    jun    : 5318   tue:8090
##                                    nov    : 4101   wed:8134
##                                    apr    : 2632
##                                    (Other): 2016
##     duration         campaign          pdays          previous
##  Min.   :   0.0   Min.   : 1.000   Min.   :  0.0   Min.   :0.000
##  1st Qu.: 102.0   1st Qu.: 1.000   1st Qu.:999.0   1st Qu.:0.000
##  Median : 180.0   Median : 2.000   Median :999.0   Median :0.000
##  Mean   : 258.3   Mean   : 2.568   Mean   :962.5   Mean   :0.173
##  3rd Qu.: 319.0   3rd Qu.: 3.000   3rd Qu.:999.0   3rd Qu.:0.000
##  Max.   :4918.0   Max.   :56.000   Max.   :999.0   Max.   :7.000
##
##         poutcome      emp.var.rate      cons.price.idx  cons.conf.idx
##  failure    : 4252   Min.   :-3.40000   Min.   :92.20   Min.   :-50.8
##  nonexistent:35563   1st Qu.:-1.80000   1st Qu.:93.08   1st Qu.:-42.7
##  success    : 1373   Median : 1.10000   Median :93.75   Median :-41.8
##                      Mean   : 0.08189   Mean   :93.58   Mean   :-40.5
##                      3rd Qu.: 1.40000   3rd Qu.:93.99   3rd Qu.:-36.4
##                      Max.   : 1.40000   Max.   :94.77   Max.   :-26.9
##
##    euribor3m      nr.employed     y
##  Min.   :0.634   Min.   :4964   no :36548
##  1st Qu.:1.344   1st Qu.:5099   yes: 4640
##  Median :4.857   Median :5191
##  Mean   :3.621   Mean   :5167
##  3rd Qu.:4.961   3rd Qu.:5228
##  Max.   :5.045   Max.   :5228
##
```

From here we can see several things:

- We have no NA values in our data. But some categorical variables have values as 'unknown' or 'nonexistent'. We'll analyse these variables later.
- The results of our target variable y aren't balanced. We only have roughly a 11% of 'yes'.
- Variable "default": in our data set of more than 40.000 calls we have only three calls to people with a previous default. The rest are unknown or have no previous default. It seems that this variable won't have a high predictive power.

2.3 Correlation on numeric columns

Now let's focus on the correlation of the numeric columns:

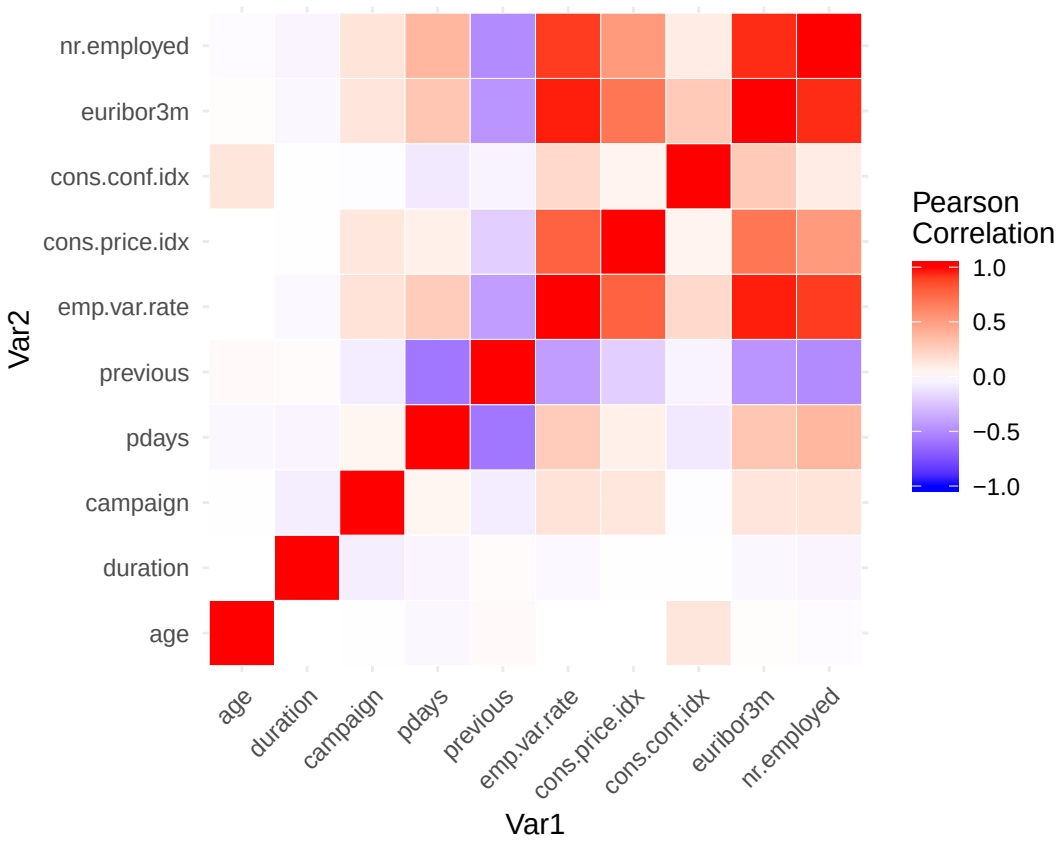
```
# Get correlation matrix
(cormat <- Filter(is.numeric,calls) %>%
    cor(method="pearson"))
```

```
##                          age     duration    campaign       pdays
## age             1.0000000000 -0.000865705  0.00459358 -0.03436895
## duration       -0.0008657050  1.000000000 -0.07169923 -0.04757702
## campaign        0.0045935805 -0.071699226  1.00000000  0.05258357
## pdays          -0.0343689512 -0.047577015  0.05258357  1.00000000
## previous        0.0243647409  0.020640351 -0.07914147 -0.58751386
## emp.var.rate   -0.0003706855 -0.027967884  0.15075381  0.27100417
## cons.price.idx  0.0008567150  0.005312268  0.12783591  0.07888911
## cons.conf.idx   0.1293716142 -0.008172873 -0.01373310 -0.09134235
## euribor3m       0.0107674295 -0.032896656  0.13513251  0.29689911
## nr.employed    -0.0177251319 -0.044703223  0.14409489  0.37260474
##                    previous  emp.var.rate cons.price.idx cons.conf.idx
## age              0.02436474 -0.0003706855    0.000856715   0.129371614
## duration         0.02064035 -0.0279678845    0.005312268  -0.008172873
## campaign        -0.07914147  0.1507538056    0.127835912  -0.013733099
## pdays           -0.58751386  0.2710041743    0.078889109  -0.091342354
## previous         1.00000000 -0.4204891094   -0.203129967  -0.050936351
## emp.var.rate    -0.42048911  1.0000000000    0.775334171   0.196041268
## cons.price.idx  -0.20312997  0.7753341708    1.000000000   0.058986182
## cons.conf.idx   -0.05093635  0.1960412681    0.058986182   1.000000000
## euribor3m       -0.45449365  0.9722446712    0.688230107   0.277686220
## nr.employed     -0.50133293  0.9069701013    0.522033977   0.100513432
##                   euribor3m nr.employed
## age              0.01076743 -0.01772513
## duration        -0.03289666 -0.04470322
## campaign         0.13513251  0.14409489
## pdays            0.29689911  0.37260474
## previous        -0.45449365 -0.50133293
## emp.var.rate     0.97224467  0.90697010
## cons.price.idx   0.68823011  0.52203398
## cons.conf.idx    0.27768622  0.10051343
## euribor3m        1.00000000  0.94515443
## nr.employed      0.94515443  1.00000000
```

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following objects are masked from 'package:data.table':
##
##     dcast, melt

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
melted_cormat <- melt(cormat)

melted_cormat %>%
  ggplot(aes(x=Var1, y=Var2, fill=value)) +
  geom_tile(color="white") +
  scale_fill_gradient2(low="blue", high = "red", mid="white",
                       midpoint=0, limit= c(-1,1), space="Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle=45, vjust = 1, hjust=1)) +
  coord_fixed()
```

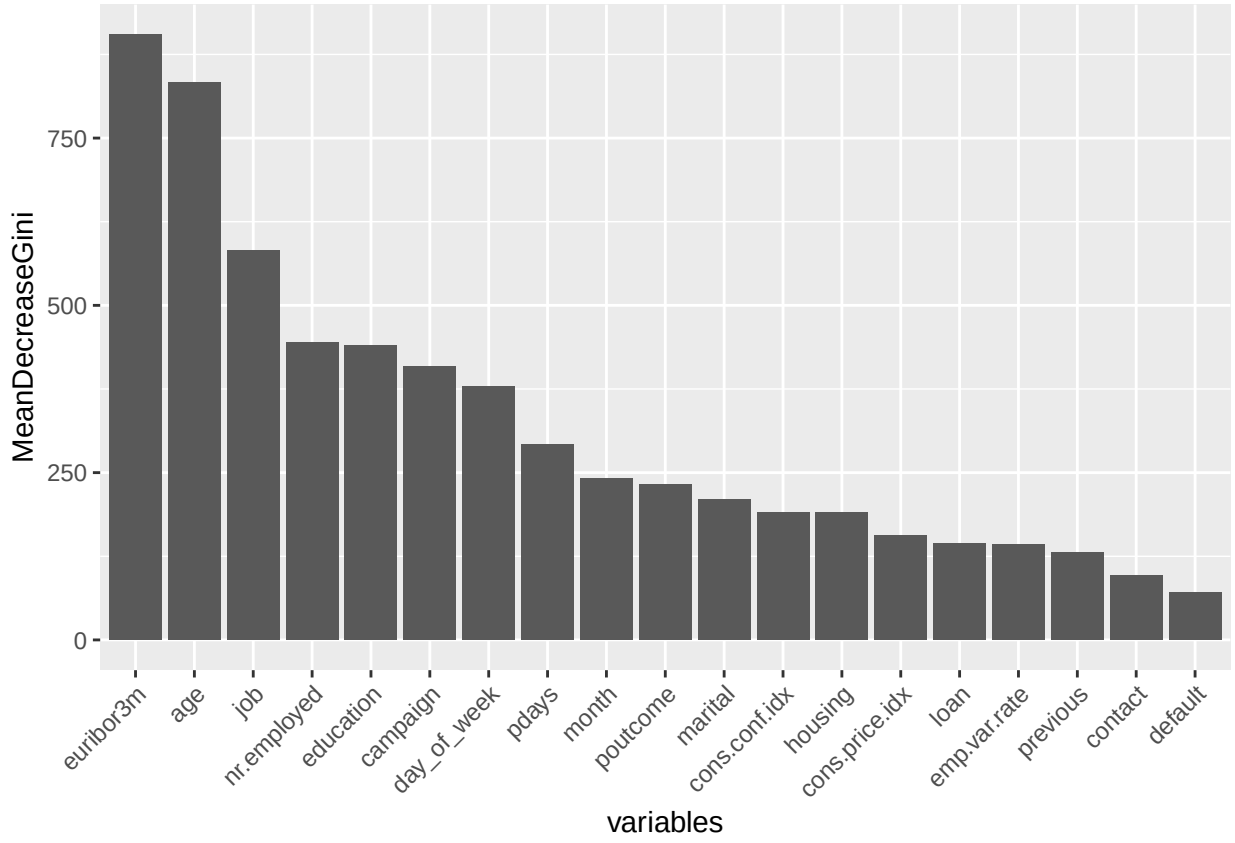


On this figure we can see strong correlations on several economic variables. Namely we have three variables strongly correlated: The euribor3m, which is an interest rate, and the two variables regarding employment (nr.employed and emp.var.rate). The correlation between the euribor3m and emp.var.rate is 0.9722, so one of these variables can be removed.

2.4 Variable importance

Now we will study the variable importance on predicting our target value. To do this we'll use the availability of this functionality on the randomforest package. Let's generate 100 trees and see which are the most important variables:

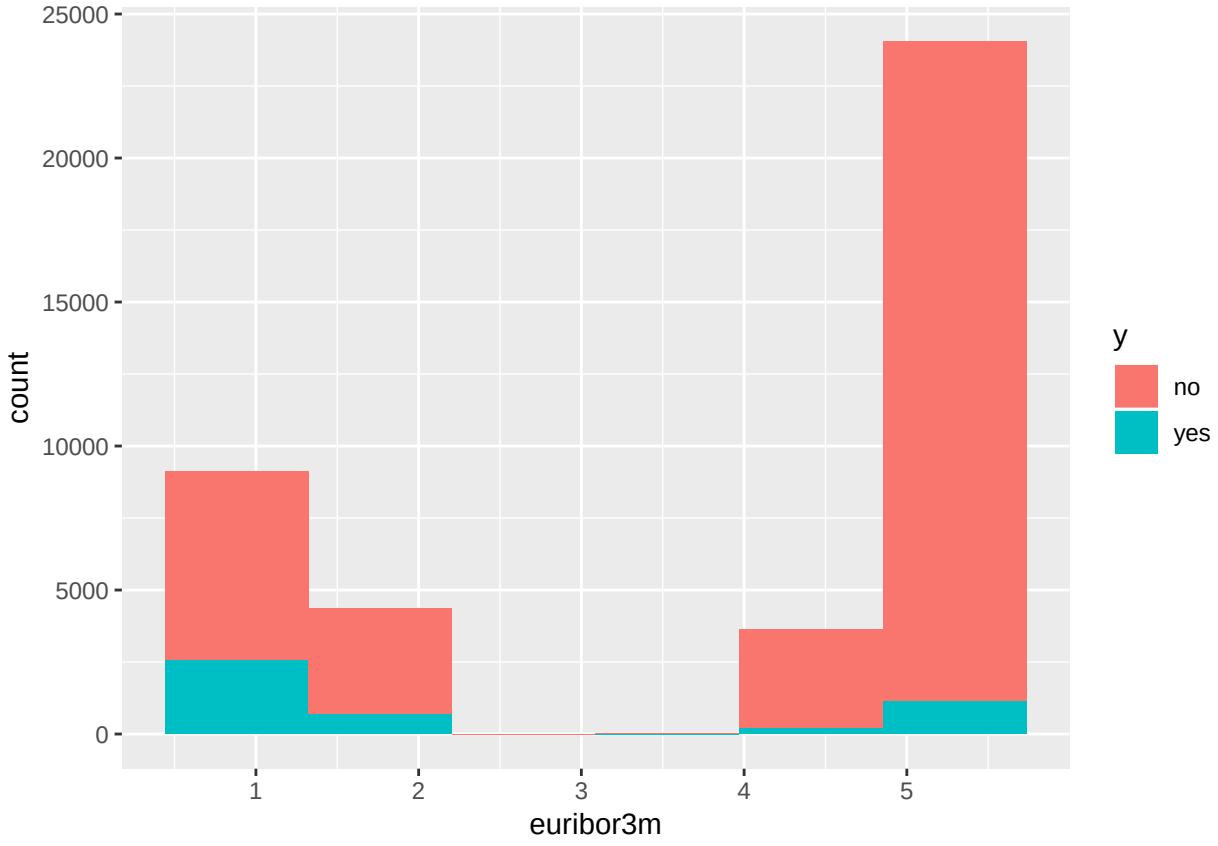
```
library(randomForest)
calls_y = calls$y
calls_x <- subset(calls, select=-c(duration,y))

rf <- randomForest(calls_x, calls_y, ntree = 100)
imp <- as.data.frame(importance(rf))
variables <- names(calls_x)
imp <- cbind(imp,variables) %>%
  arrange(desc(MeanDecreaseGini))
# set variables as factor to maintain orden in the plot
imp$variables <- factor(imp$variables, levels=imp$variables)
ggplot(imp, aes(x=variables,y=MeanDecreaseGini)) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle=45, vjust = 1, hjust=1))
```



2.4.1 euribor 3m variable

One of the most important variables is an interest rate indicator as the 3 month euribor. It seems logic, as the product we are offering is a term deposit. Let's see the distribution of results regarding this indicator:

```
calls %>%
  ggplot(aes(x=euribor3m, fill=y)) +
    geom_histogram(bins=6)
```

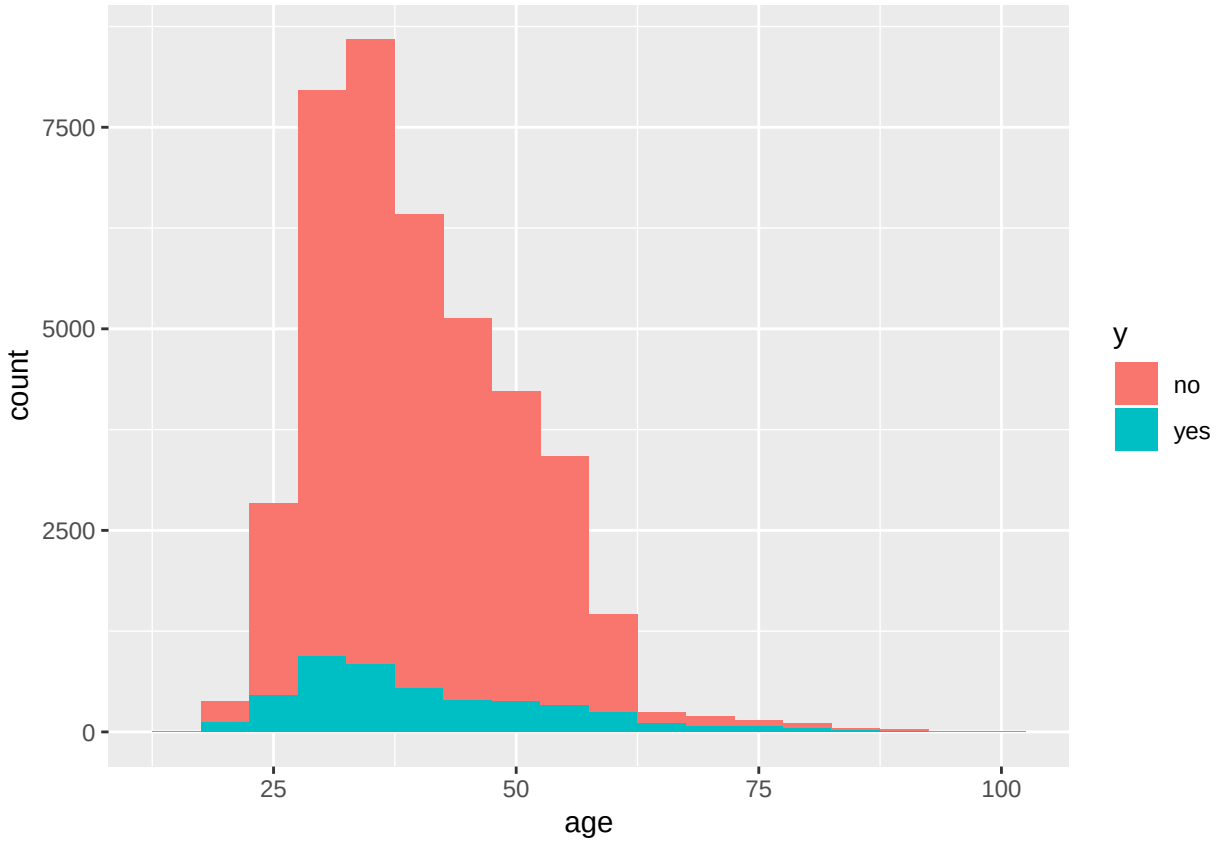


Seems that our campaign was more successful with a low interest rate. And we seldom have results with euribor around 3%. Why? Our data set register calls from may 2008 to November 2010. If we look at historical data for years 2008, 2009 and 2010 at https://www.euribor-rates.eu, we can see that interest rates collapsed quite abruptly at the end of 2008. Seems than some characteristic of our term deposit was more attractive with a low interest rate. Maybe our interest rate was competitive or maybe a relatively safe investment as a term deposit is preferred on crisis times.

2.4.2 age variable

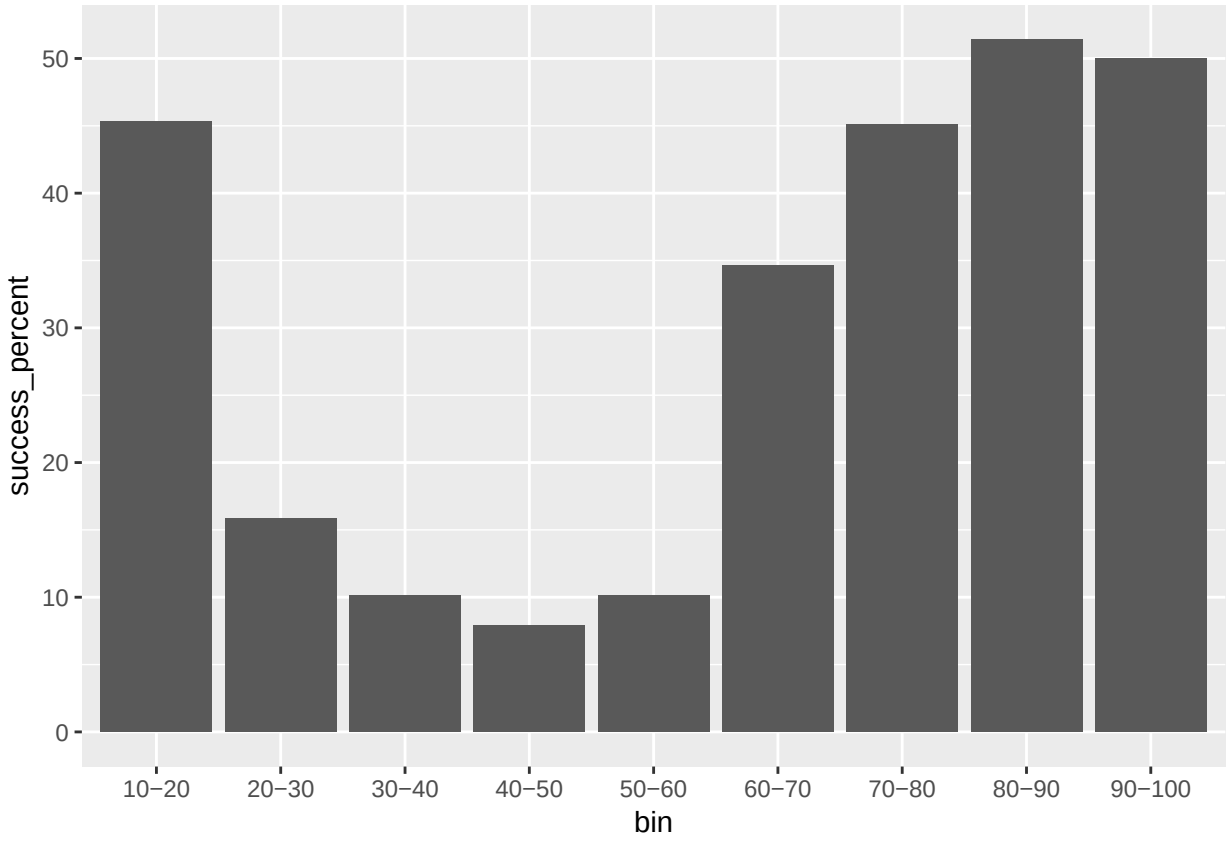
The next figure shows the distribution of the calls by age, in bins of five years.

```
calls %>%
  ggplot(aes(x=age, fill=y)) +
  geom_histogram(binwidth=5)
```



We can see that the age ranges have different success rates. Seems that our campaign was less successful with people between 35 and 45 years. The next figure shows the success rate by age:

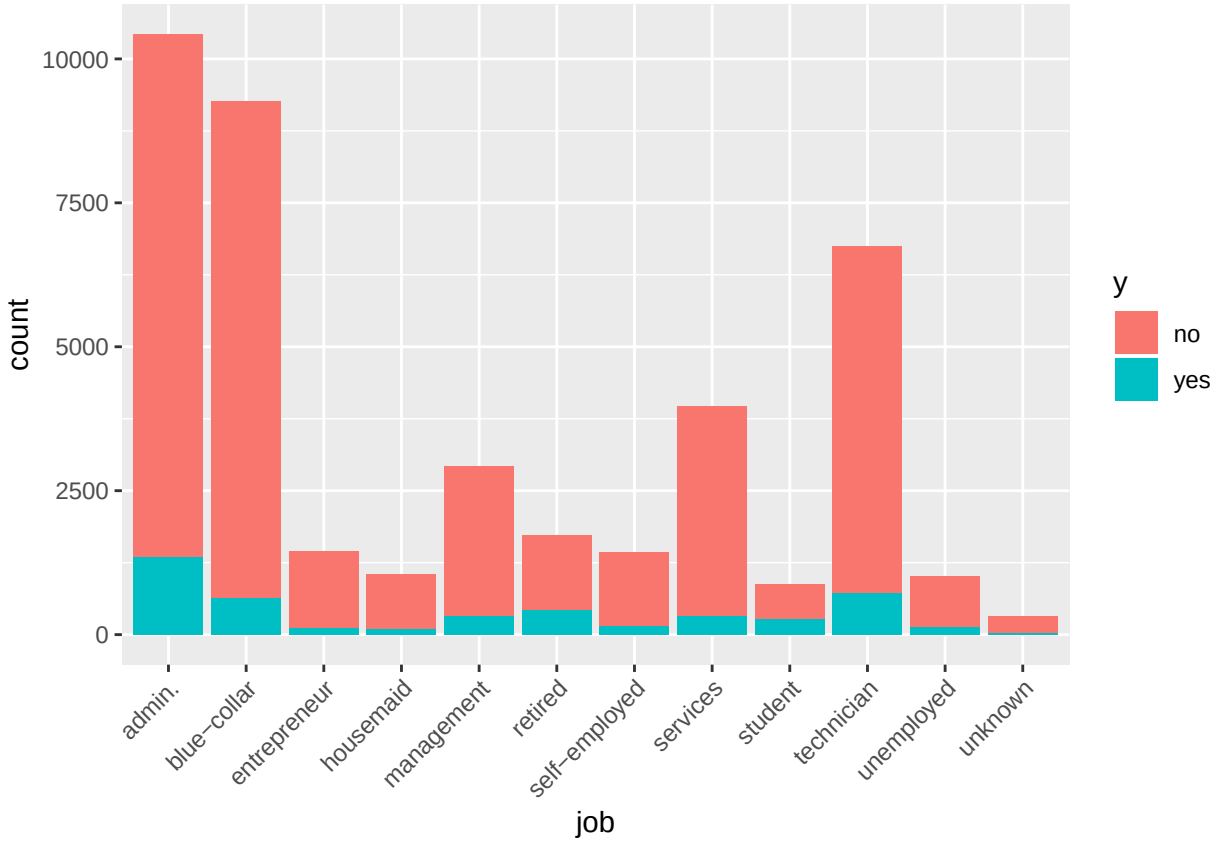
```
age_success <- calls %>%
  mutate(age_round = floor(age/10)*10, bin=paste(age_round, age_round+10,sep="-")) %>%
  group_by(bin,y) %>%
  summarize(n=n()) %>%
  ungroup() %>%
  group_by(bin) %>%
  spread(y,n, fill=0) %>%
  mutate(success_percent = (yes /(no+yes))*100)

age_success %>%
  ggplot(aes(x=bin, y=success_percent)) +
  geom_bar(stat="identity")
```



2.4.3 job variable

Now we'll analyse the job variable. In the next figure we classify the calls by jobs, and show the successes and failures by color.

```
calls %>%
  ggplot(aes(x=job, fill=y)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle=45, vjust = 1, hjust=1))
```



Now the success rates by job are calculated. We'll show the best and the worse rates.

```
job_success <- calls %>%
  group_by(job,y) %>%
  summarize(n=n()) %>%
  ungroup() %>%
  group_by(job) %>%
  spread(y,n) %>%
  mutate(success_percent = (yes /(no+yes))*100) %>%
  arrange(desc(success_percent))
```

The best rates:

```
head(job_success,5)
```

```
## # A tibble: 5 x 4
## # Groups:   job [12]
##   job           no   yes success_percent
##   <fct>      <int> <int>           <dbl>
## 1 student      600   275            31.4
## 2 retired     1286   434            25.2
## 3 unemployed   870   144            14.2
## 4 admin.      9070  1352            13.0
## 5 management  2596   328            11.2
```

Seems that the campaign was quite successful on students and retired people. Let's see the worse rates:

```
tail(job_success,5)
```

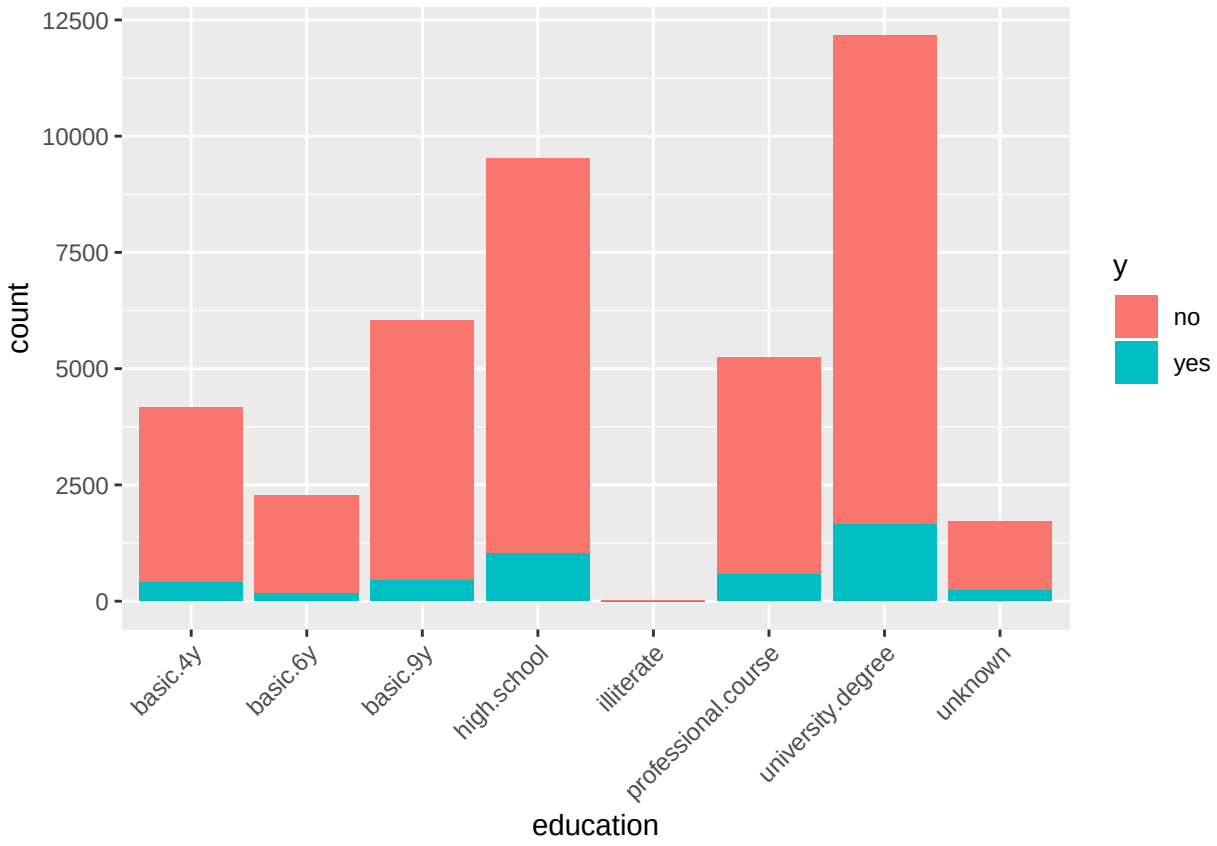
```
## # A tibble: 5 x 4
## # Groups:   job [12]
##   job              no   yes success_percent
##   <fct>         <int> <int>           <dbl>
## 1 self-employed  1272   149            10.5
## 2 housemaid       954   106            10
## 3 entrepreneur   1332   124             8.52
## 4 services       3646   323             8.14
## 5 blue-collar    8616   638             6.89
```

The worse results were with blue-collar employees.

2.4.4 Education

The next figure shows the success ratios by education level:

```
calls %>%
  ggplot(aes(x=education, fill=y)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle=45, vjust = 1, hjust=1))
```

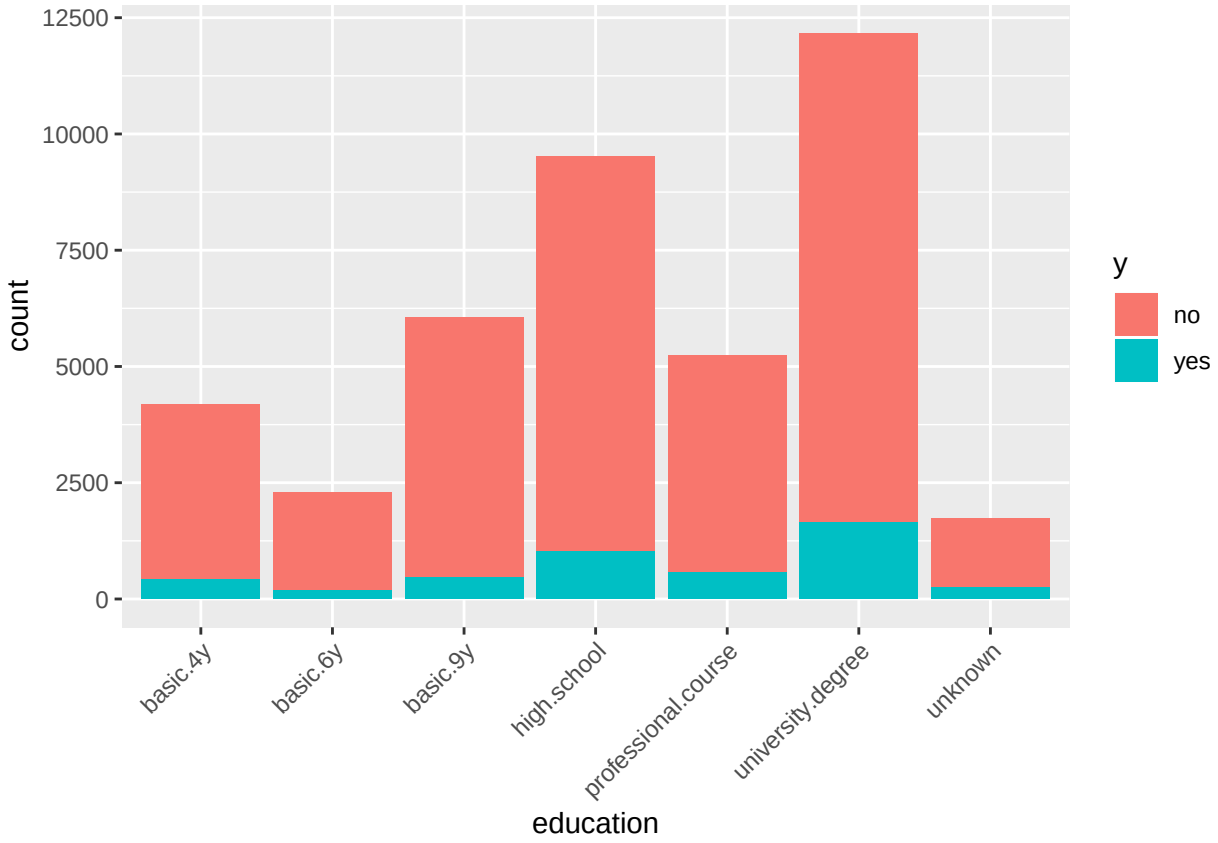


We can see here that we have a category (illiterate) with very few observations. A way to address this may be to fusion this category with the lower education level, basic.4y, that now will be this category or lower. We'll do it this way:

```
# Fusion of illiterate and basic.4y
levels_ed <- levels(calls$education)
levels(calls$education) <- ifelse(levels_ed=="illiterate","basic.4y",levels_ed)
rm(levels_ed)
```

And now the figure is:

```
calls %>%
  ggplot(aes(x=education, fill=y)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle=45, vjust = 1, hjust=1))
```



The next table shows the success ratios by education level

```
education_success <- calls %>%
  group_by(education,y) %>%
  summarize(n=n()) %>%
  ungroup() %>%
  group_by(education) %>%
  spread(y,n) %>%
  mutate(success_percent = (yes /(no+yes))*100) %>%
```

```
  arrange(desc(success_percent))

education_success
```

```
## # A tibble: 7 x 4
## # Groups:   education [7]
##   education              no   yes success_percent
##   <fct>               <int> <int>           <dbl>
## 1 unknown              1480   251            14.5
## 2 university.degree   10498  1670            13.7
## 3 professional.course  4648   595            11.3
## 4 high.school          8484  1031            10.8
## 5 basic.4y             3762   432            10.3
## 6 basic.6y             2104   188             8.20
## 7 basic.9y             5572   473             7.82
```

Apart from the 'unknown' category, the customers with the highest education levels seems to be a little more likely to subscribe the term deposit.

Regarding the 'unknown' category, seems that has a reasonable rate of success. Indeed it is the category with the higher rate of success. It doesn't seem reasonable to consider not knowing the education level of our customers as a success factor in our campaign, so we'll need to apply here some imputation techniques. We'll review this in the preprocessing chapter.

2.5 Variables with near Zero variance or near Zero entropy

The caret package has a function that detects features with near zero variance, and hence candidates to be removed. Let's see its recommendation.

```
nzv <- nearZeroVar(calls)
names(calls)[nzv]
```

```
## [1] "pdays"
```

The recommendation is to remove the 'pdays' variable which is the number of days passed since the last contact with the same client. But If we we look at the bar plot on the previous point, this variable has a worthy ninth place of nineteen. Let's have a closer look to this variable.

2.5.1 variable 'pdays'

The next table shows how many rows we have for each value in pdays:

```
table(calls$pdays)
```

```
##
##     0     1     2     3     4     5     6     7     8     9    10    11
##    15    26    61   439   118    46   412    60    18    64    52    28
##    12    13    14    15    16    17    18    19    20    21    22    25
##    58    36    20    24    11     8     7     3     1     2     3     1
##    26    27   999
##     1     1 39673
```

If we calculate the frequency of the value '999' (no previous contact):

```
calls %>%
  group_by(pdays) %>%
  summarize(n = n()) %>%
  mutate(freq = (n / sum(n)) * 100) %>%
  filter(pdays==999) %>% .$freq
```

```
## [1] 96.32174
```

We obtain that 96 % of the calls have no previous contact. But remember that our target variable is asymmetric as well. Let's the relationship between this variable with other values than 999 and our target variable.

```
calls %>%
  filter(pdays != 999) %>%
  group_by(y) %>%
  summarise(n=n())
```

```
## # A tibble: 2 x 2
##   y         n
##   <fct> <int>
## 1 no      548
## 2 yes     967
```

As we see, roughly two thirds of the calls with a recent contact have a positive result. Indeed, comparing with the total number of positive results in our data set we see that approximately a 20% of the positive results have previous contacts in the lasts days.

Given these results, we'll keep the pdays variable, and recommend our marketing department to place resources on following up some customers.

2.5.2 variable 'default'

Now we'll analyse the 'default' variable. As we saw with the summary only three calls corresponded to customers with a previous default. We saw as well that in the first bar plot on the variable importance point this variable had the last position.

This is a categorical variable. Just to give a number to discard this variable, we'll use the entropy formula defined by Shannon:

$$H(x) = -\sum_{x_i} p(x_i) log_2 p(x_i)$$

In r code:

```
entropy = function(vector) {
  px = table(vector) / length(vector)
  lpx = log(px, base=2)
  -sum(px * lpx)
}
```

Our entropy is:

```r
entropy(calls$default)
```

```
## [1] 0.7401218
```

But after removing the unknown values we have near zero entropy:

```r
entropy(droplevels(calls$default[calls$default != "unknown"]))
```

```
## [1] 0.001366928
```

So this is clearly a variable candidate to be not considered in our model.

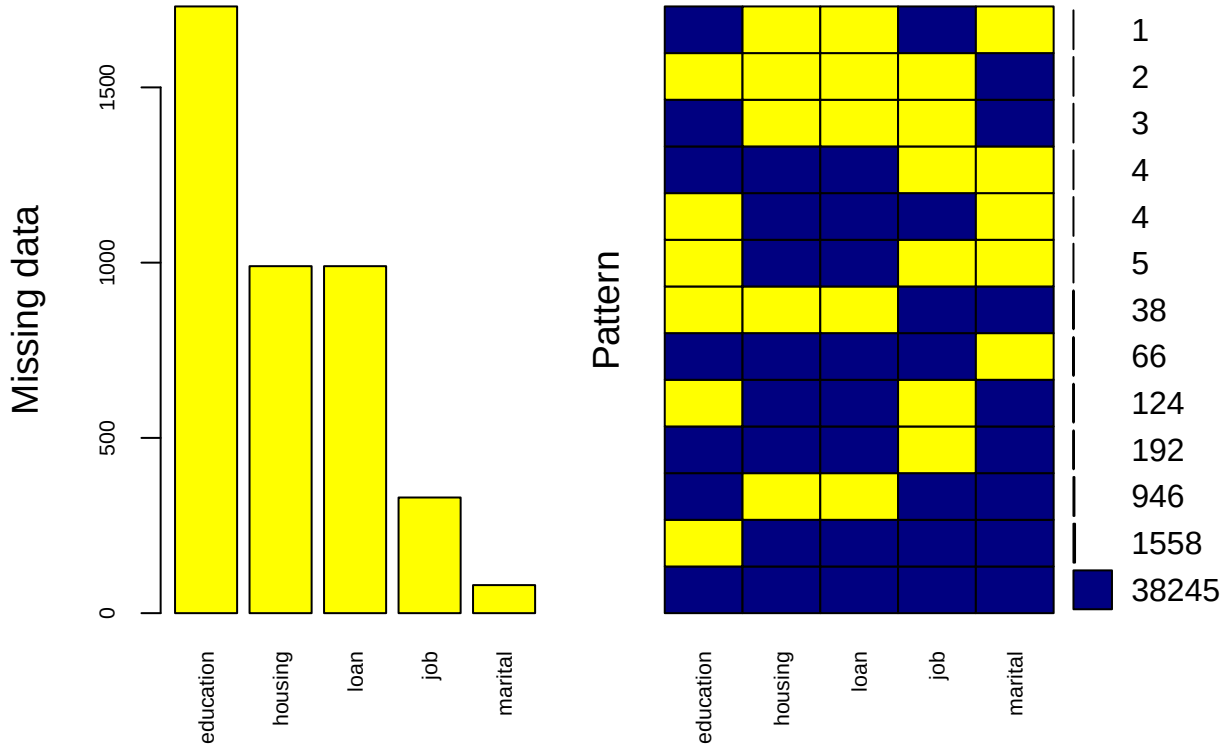
2.6 Missing values

As we have seen in previous steps, some categorical variables have values labeled as 'unknown' or 'nonexistent'.

The only variable that has the value 'nonexistent' is the variable 'poutcome' (previous outcome). By my understanding, these aren't actually missing values so I won't do any imputation on them.

The next five categorical variables have value labeled as 'unknown': education, housing, loan, job and marital. The next figures shows some patterns regarding unknown data on these data:

```r
## Missing values
# replace unknown by NA

calls$job <- as.character(calls$job)
calls$job <- ifelse(calls$job == "unknown", NA, calls$job)
calls$job <- factor(calls$job)

calls$marital <- as.character(calls$marital)
calls$marital <- ifelse(calls$marital == "unknown", NA, calls$marital)
calls$marital <- factor(calls$marital)

calls$education <- as.character(calls$education)
calls$education <- ifelse(calls$education == "unknown", NA, calls$education)
calls$education <- factor(calls$education)

calls$housing <- as.character(calls$housing)
calls$housing <- ifelse(calls$housing == "unknown", NA, calls$housing)
calls$housing <- factor(calls$housing)

calls$loan <- as.character(calls$loan)
calls$loan <- ifelse(calls$loan == "unknown", NA, calls$loan)
calls$loan <- factor(calls$loan)


aggr_plot <- aggr(calls[,c("job","marital","education", "housing", "loan")], col=c('navyblue','yellow')
                  numbers=TRUE, sortVars=TRUE, prop=FALSE,
                  labels=c("job","marital","education", "housing", "loan"), cex.axis=.7,
                  gap=3, ylab=c("Missing data", "Pattern"))
```




```
##
##  Variables sorted by number of missings:
##   Variable Count
##  education  1731
##    housing   990
##       loan   990
##        job   330
##    marital    80
```

These figures give us some insights on missing data:

1. 38245 observations out of 41188 have all 5 variables informed
2. There aren't observations with all 5 variables 'unknown'
3. The variable with more unknown values is 'education' (1731 observations)
4. If we look at the pattern table, we see that when 'housing' is unknown, 'loan' is unknown as well and vice versa

2.7 Preprocessing

Here starts the preprocessing of the data. If first place we'll remove the variables 'duration','default' and 'emp.var.rate' as in our data analysis we have seen that they won't be useful for our model.

```
calls <- subset(calls, select=-c(duration, default, emp.var.rate))
```

The next step is the imputation of the missing values. To do this, I will use the MICE package (Multivariate Imputation by Chained Equations). You can see the documentation here. First we'll do the initialization:

```
init = mice(calls, maxit=0)
meth = init$method
predM = init$predictorMatrix
```

The five variables are related with personal data. It is pointless to consider certain data to deduct personal data, as the interest rates. We can neither consider our target variable as a predictor for the missing values.

```
# Remove some variables as predictors
predM[, c("contact","month","day_of_week", "campaign", "pdays",
          "previous","poutcome", "cons.price.idx","cons.conf.idx",
          "euribor3m","y")]=0
```

Now we define the methods for prediction. The variables 'job', 'marital' and 'education' have several categories. The variables 'housing' and 'loan' have only two categories.

```
# set methods for prediction
meth[c("job", "marital","education")] = "polyreg"
meth[c("housing","loan")] = "logreg"
```

And now starts the imputation:

```
set.seed(103)
imputed = mice(calls, method=meth, predictorMatrix=predM, m=5)

imputed <- complete(imputed)
```

The next step is not part of the imputation. The results of some of the algorithms used in the next chapters have improved when we define the education variable as an ordered factor:

```
## ordered factors
imputed$education <- factor(imputed$education, ordered=TRUE, levels = c("basic.4y","basic.6y","basic.9y
```

The summary of the dataset resulting of the imputation is the next:

```
summary(imputed)
```

```
##       age                job           marital
##  Min.   :17.00   admin.     :10487   divorced: 4615
##  1st Qu.:32.00   blue-collar: 9343   married :24978
##  Median :38.00   technician : 6781   single  :11595
##  Mean   :40.02   services   : 3995
##  3rd Qu.:47.00   management : 2953
##  Max.   :98.00   retired    : 1751
##                  (Other)    : 5878
##                education     housing      loan            contact
##  basic.4y           : 4478   no :19076   no :34789   cellular :26144
##  basic.6y           : 2407   yes:22112   yes: 6399   telephone:15044
##  basic.9y           : 6334
```

```
##  high.school        : 9909
##  professional.course: 5465
##  university.degree  :12595
##
##      month       day_of_week    campaign          pdays
##  may    :13769   fri:7827    Min.   : 1.000   Min.   :  0.0
##  jul    : 7174   mon:8514    1st Qu.: 1.000   1st Qu.:999.0
##  aug    : 6178   thu:8623    Median : 2.000   Median :999.0
##  jun    : 5318   tue:8090    Mean   : 2.568   Mean   :962.5
##  nov    : 4101   wed:8134    3rd Qu.: 3.000   3rd Qu.:999.0
##  apr    : 2632               Max.   :56.000   Max.   :999.0
##  (Other): 2016
##     previous            poutcome     cons.price.idx  cons.conf.idx
##  Min.   :0.000   failure    : 4252   Min.   :92.20   Min.   :-50.8
##  1st Qu.:0.000   nonexistent:35563   1st Qu.:93.08   1st Qu.:-42.7
##  Median :0.000   success    : 1373   Median :93.75   Median :-41.8
##  Mean   :0.173                       Mean   :93.58   Mean   :-40.5
##  3rd Qu.:0.000                       3rd Qu.:93.99   3rd Qu.:-36.4
##  Max.   :7.000                       Max.   :94.77   Max.   :-26.9
##
##    euribor3m      nr.employed     y
##  Min.   :0.634   Min.   :4964   no :36548
##  1st Qu.:1.344   1st Qu.:5099   yes: 4640
##  Median :4.857   Median :5191
##  Mean   :3.621   Mean   :5167
##  3rd Qu.:4.961   3rd Qu.:5228
##  Max.   :5.045   Max.   :5228
##
```

2.8 Division of train and test set

For the train set we will select a 90% of the observations and the other 10% will be the test set.

```
#############################################################################
## divide in 90% train set and 10% test set
set.seed(204)
test_index <- createDataPartition(y = imputed$y, times = 1, p = 0.1, list = FALSE)

train_set <- imputed[-test_index,]
test_set <- imputed[test_index,]


rm(test_index)
```

We will define a subset of the training set to select the hyperparameters. We'll use it for the algorithms that need more computation power. When possible, we'll use the hole data set to select hyperparameters.

```
## This is a sample that will be used to train the hiperparameters of several algorithms
n <- 6500
index <- sample(nrow(train_set), n)
```

2.9 Algorithms

In the next sections we'll train several algorithms in order to predict which contacts will result in a subscription. The intent of this section is not doing a detailed description of each algorithm, as it can be found in internet.

For each algorithm we may do the next steps:

1. Preprocessing of data: For some algorithms we'll do some operations on the data, as centering and scaling.
2. hyperparameter tuning
3. train the model
4. Evaluate with the test set.

To compare the performance of these models the Area Under the Curve (AUC) metric will be used. To calculate it, the pROC package is used.

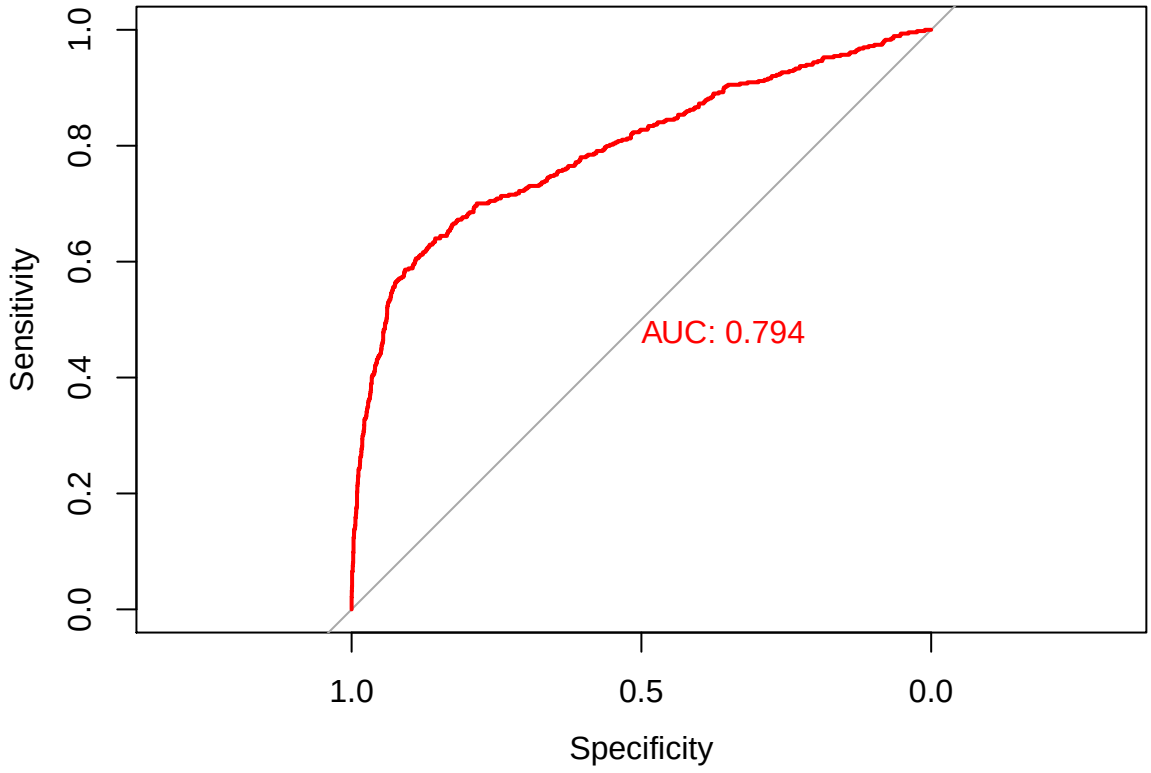
2.9.1 Generalized Linear Model (glm)

This model has no hyperparameters, so we'll train directly the train set.

```
#####################################
## glm
fit_glm <- glm(y ~ . ,
               data=train_set,
               family="binomial")
p_glm <- predict(fit_glm, test_set, type="response")
```

And the result:

```
ROC <- roc(test_set$y, p_glm)
plot(ROC, col = "red", print.auc=TRUE)
```




```
auc_results <- tibble(method = "glm", AUC = pROC::auc(ROC))
```

2.9.2 GLM with lasso or elasticnet regularization (glmnet)

On the next model we fit two hyperparamenters: alpha and lambda. As this algorithm is quick enough, we'll search the hyperparameters with the whole train set.

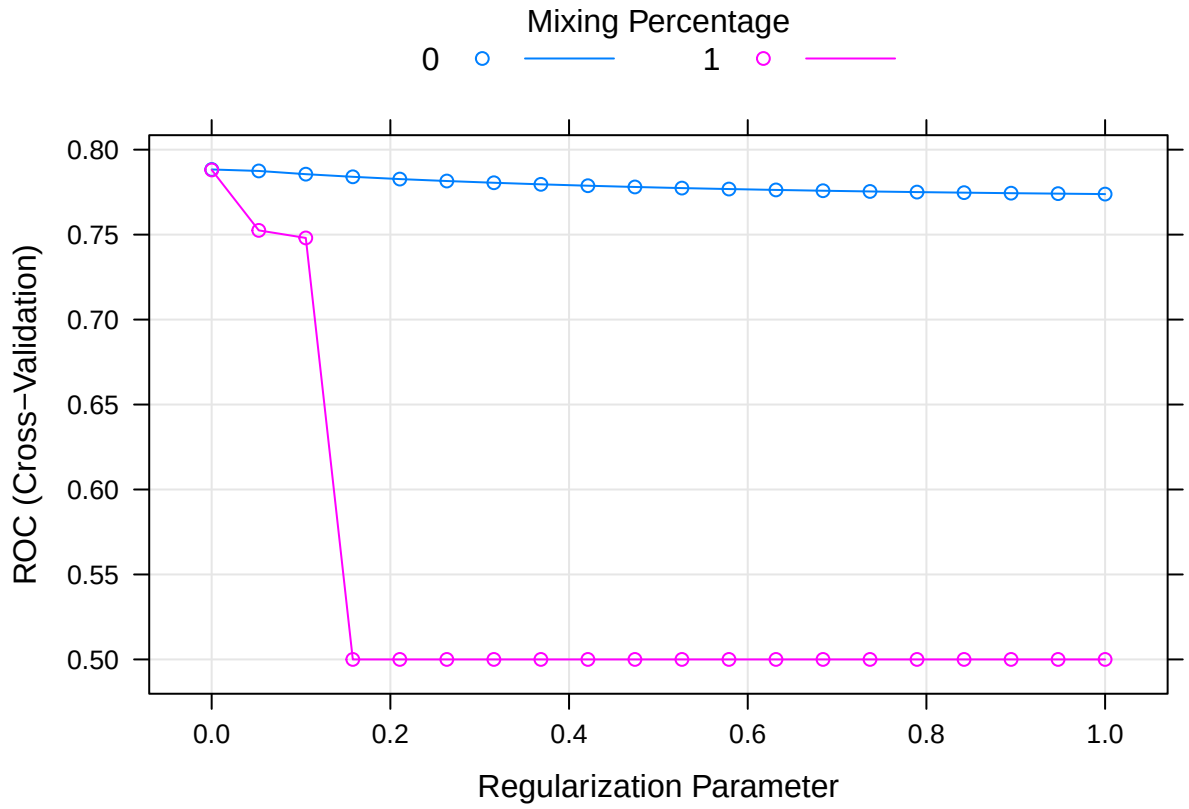
The hyperparameters for this algorithm are two: * alpha is the elasticnet mixing parameter. "1" for lasso regression, "0" for ridge regression * lambda: the penalization used for regularization

```
#######################################
## glmnet
glmnet_control <- trainControl(
  method = "cv", number=5,
  summaryFunction = twoClassSummary,
  classProbs = TRUE,
  verboseIter = TRUE
)

glmnet_grid <- expand.grid(alpha = 0:1, lambda = seq(0.0001, 1, length = 20))

train_glmnet <- train(
  y ~ ., train_set,
  tuneGrid = glmnet_grid,
  method = "glmnet",
```

```
  trControl = glmnet_control,
  metric="ROC"
)
```

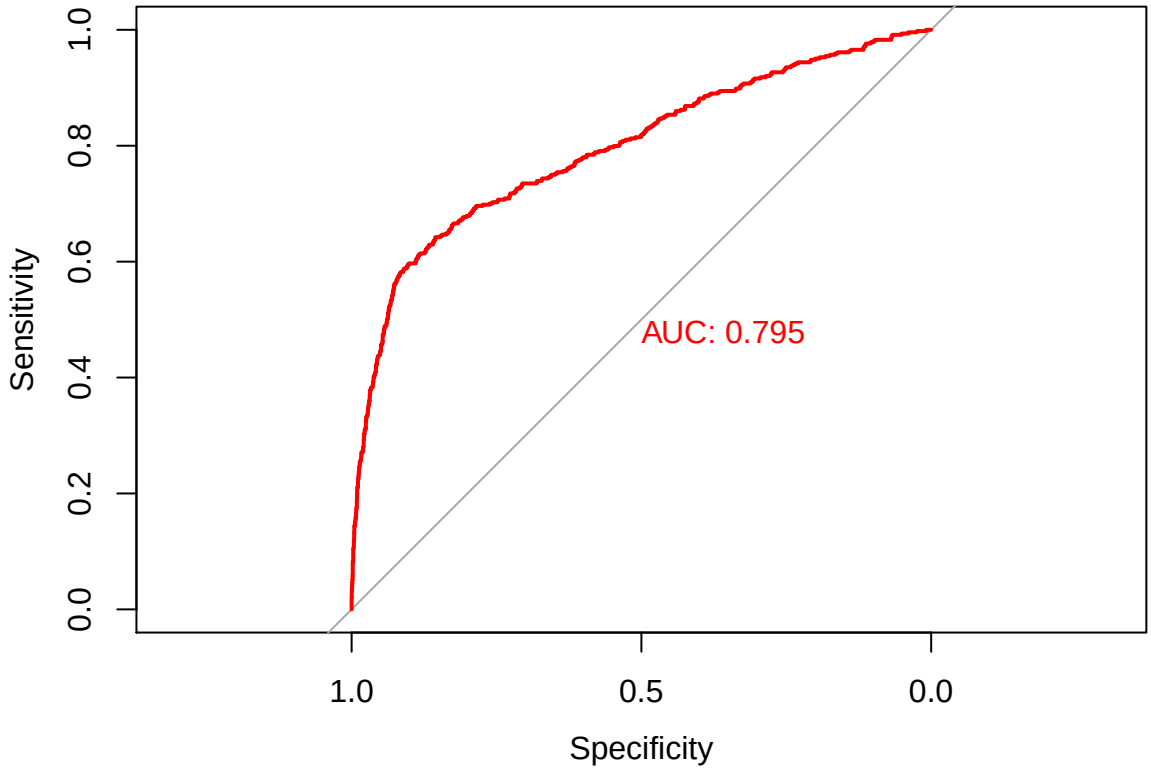
The results of our hyperparameter search:

```
# plot the model
plot(train_glmnet)
```



And now we do the predictions on the test set and calculate AUC:

```
p_glmnet <- predict(train_glmnet, test_set, type="prob")


ROC <- roc(test_set$y, p_glmnet[,"yes"])
plot(ROC, col = "red", print.auc=TRUE)
```




```
auc_results <- rbind(auc_results,
                           tibble(method="glmnet",
                                      AUC = pROC::auc(ROC) ))
```

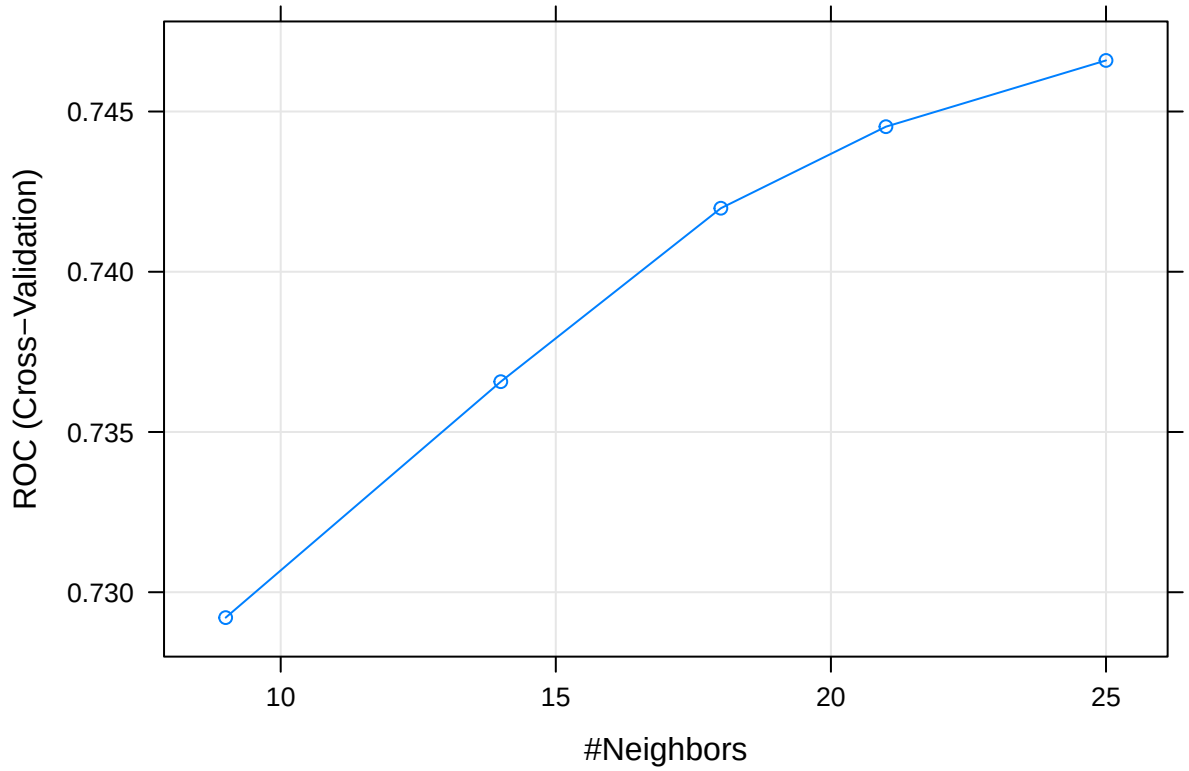
2.9.3 k-Nearest Neighbours (KNN)

For this algorithm, the hyperparameter to search is the number of neighbors k. we'll do the hyperparameter search with a sample of the training set.

```
#######################################
## knn

control_knn <- trainControl(
  method="cv", number = 5,
  summaryFunction = twoClassSummary,
  classProbs = TRUE,
  verboseIter = TRUE)


train_knn <- train(y ~ .,
                   data=train_set[index,],
                   method = "knn",
                   tuneGrid = data.frame(k = c(9,14,18,21,25)),
                   trControl = control_knn,
                   metric="ROC")
```

The results of our search:

```
# plot the model
plot(train_knn)
```

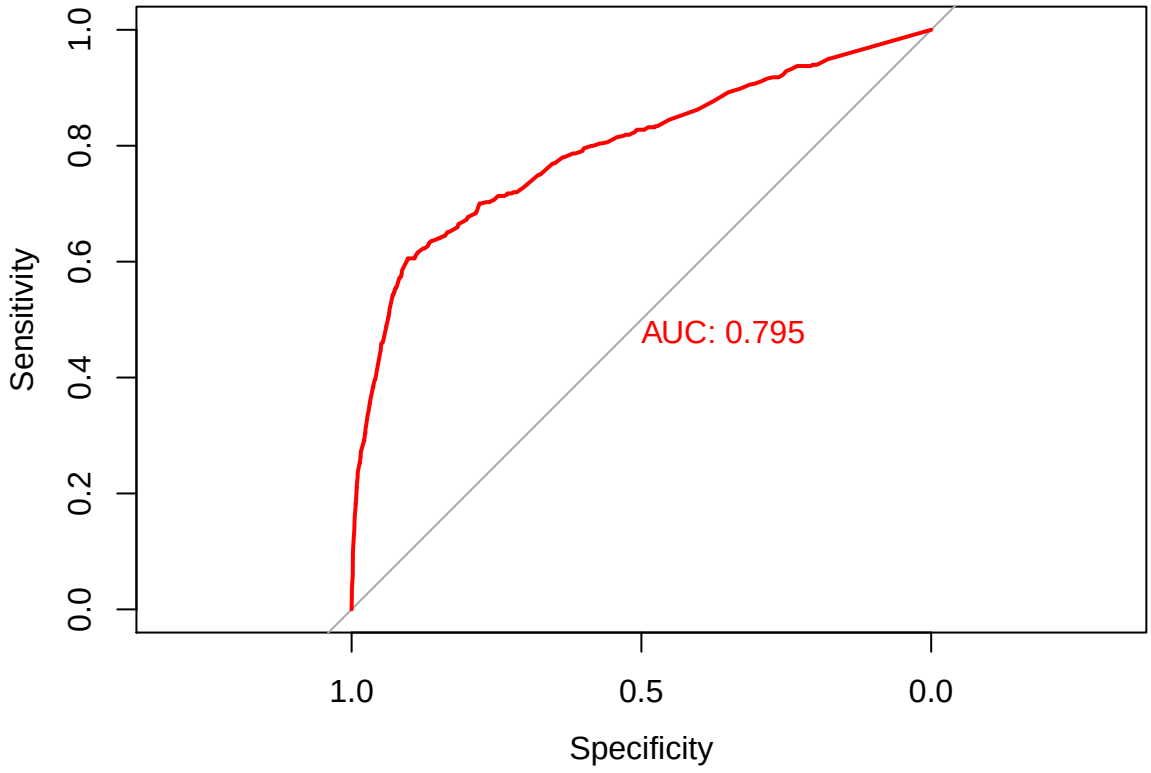


Now train the algorithm with the best tune found on the whole train set.

```
fit_knn <- knn3(y ~., train_set,  k = train_knn$bestTune$k)
```

Finally do the predictions on the test set and calculate AUC.

```
p_knn  <- predict(fit_knn, test_set)

ROC <- roc(test_set$y, p_knn[,"yes"])
plot(ROC, col = "red", print.auc=TRUE)
```




```
auc_results <- rbind(auc_results,
                          tibble(method="knn",
                                     AUC = pROC::auc(ROC) ))
```

2.9.4 Naive Bayes

Our next algorithm is Naive Bayes. This algorithm will produce a lot of warning messages, but we can ignore them.

Our search grid includes three hyperparameters: usekernel (true or false), fl (Laplace correction) and adjust (a factor correction to multiply with the smoothing bandwidth; adjust = 1 means no adjust)

```
#####################################
## Naive Bayes

control_nb <- trainControl(
  method="cv", number = 5,
  summaryFunction = twoClassSummary,
  classProbs = TRUE,
  verboseIter = TRUE)


nb_grid <- expand.grid(
  usekernel = c(TRUE, FALSE),
  fL=0:3,
```

```
  adjust = c(0.5, 1, 1.5, 2)
)
```

With this algorithm we'll apply three transformations on the data: center (center numeric values around zero), scale (scale numeric values) and Independent component Analysis (ica):

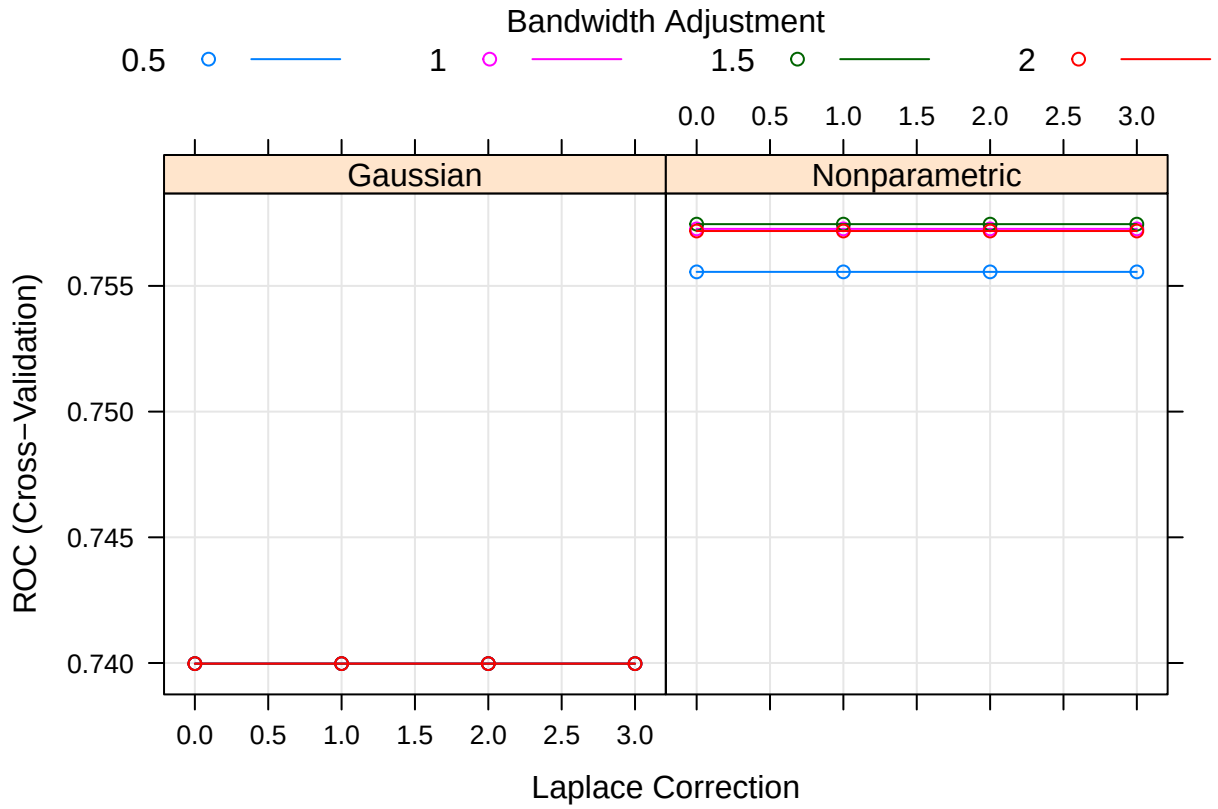
```
preProc_nb <- preProcess(subset(train_set, select=-y), method=c("center", "scale","ica"),n.comp=5)
preProc_train_set <- predict(preProc_nb,train_set)
preProc_test_set <- predict(preProc_nb,test_set)
```

And now, search the best tune of the hyperparameters in a subset of the training set:

```
train_nb <- train(y ~.,
                  data=preProc_train_set[index,],
                  method="nb",
                  trControl = control_nb,
                  tuneGrid = nb_grid,
                  metric="ROC")
```

The results of our search:

```
#Plot the model
plot(train_nb)
```

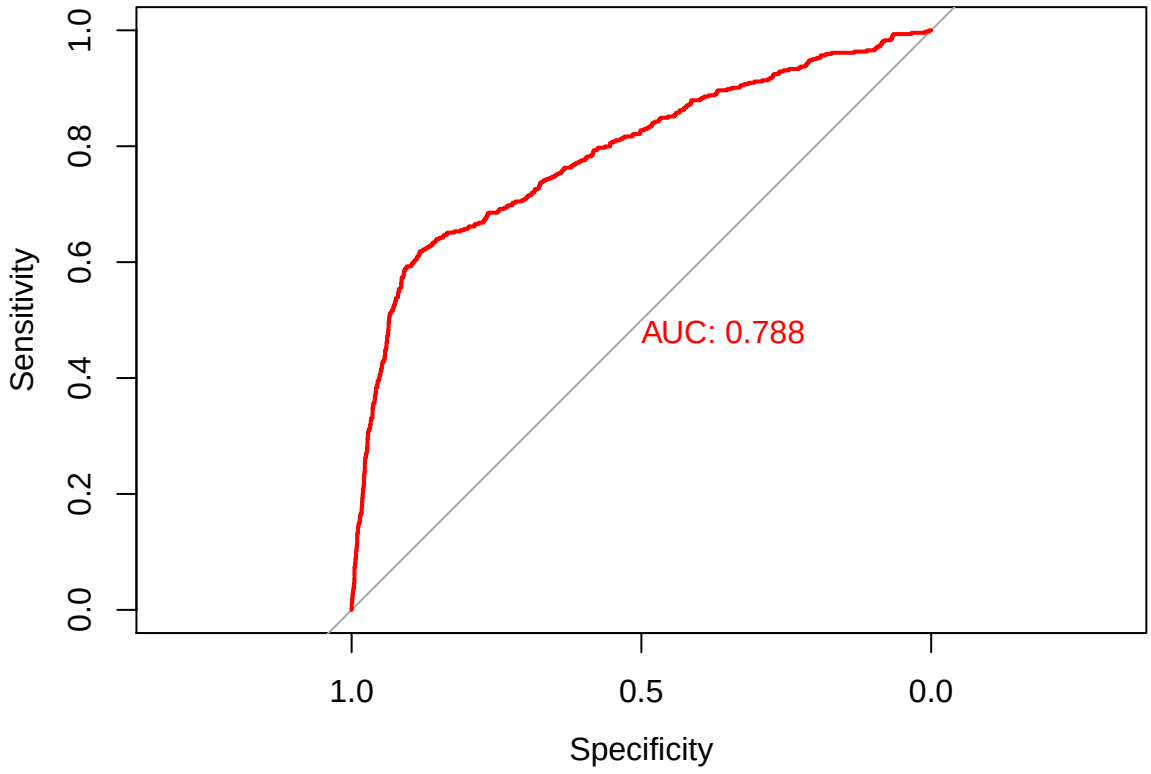


Now we train the algorithm with the whole train set with the best tune:

```
fit_nb <- NaiveBayes(y ~ .,
                     data=preProc_train_set,
                     fL = train_nb$bestTune$fL,
                     usekernel = train_nb$bestTune$usekernel,
                     adjust=train_nb$bestTune$adjust)
```

Finally do the predictions on the test set and calculate AUC.

```
p_nb <- predict(fit_nb, preProc_test_set)$posterior

ROC <- roc(test_set$y, p_nb[,"yes"])
plot(ROC, col = "red", print.auc=TRUE)
```




```
auc_results <- rbind(auc_results,
                         tibble(method="Naive Bayes",
                                    AUC = pROC::auc(ROC) ))
```

2.9.5 Random Forest (RBorist)

With the RBorist function we found a problem with the ordered factor, so just for this algorithm I'll undo this transformation:

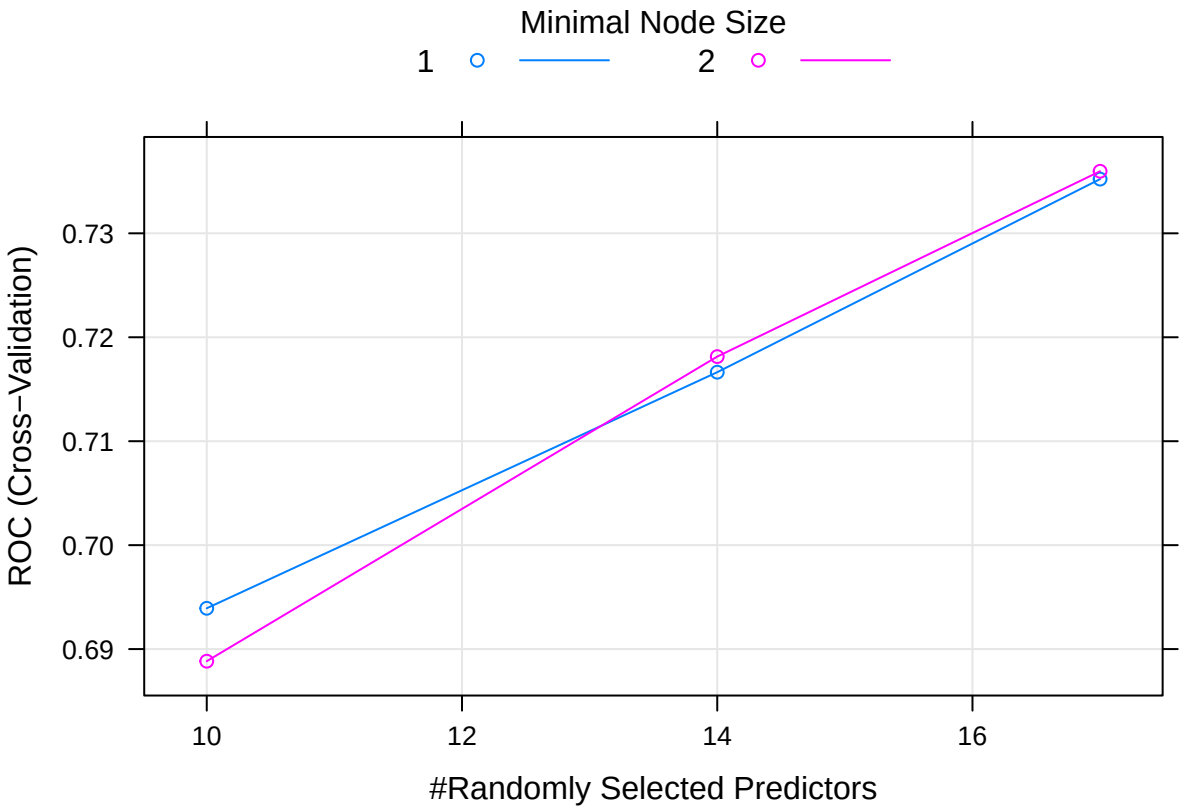
```
# Preprocessing: Rborist returns an error of unsopported type where there are ordered factors
preProc_train_set <- train_set
preProc_train_set$education <- factor(preProc_train_set$education, ordered=FALSE)

preProc_test_set <- test_set
preProc_test_set$education <- factor(preProc_test_set$education, ordered=FALSE)
```

For this algorithm we'll search the best tune through two hyperparameters: minNode (minimal index width over which to attempt splitting) and predFixed (number of trial predictors for a split).

```
# hiperparameter tuning
control_rf <- trainControl(
  method="cv", number = 5,
  summaryFunction = twoClassSummary,
  classProbs = TRUE,
  verboseIter = TRUE)

grid_rf <- expand.grid(minNode = c(1,2) , predFixed = c(10, 14, 17))

train_rf <-  train(y ~ .,
                   data = preProc_train_set[index,],
                   method = "Rborist",
                   nTree = 50,
                   trControl = control_rf,
                   tuneGrid = grid_rf,
                   nSamp = 5000,
                   metric="ROC")
```

The next figure shows the results of our search:

```
## plot the model
plot(train_rf)
```

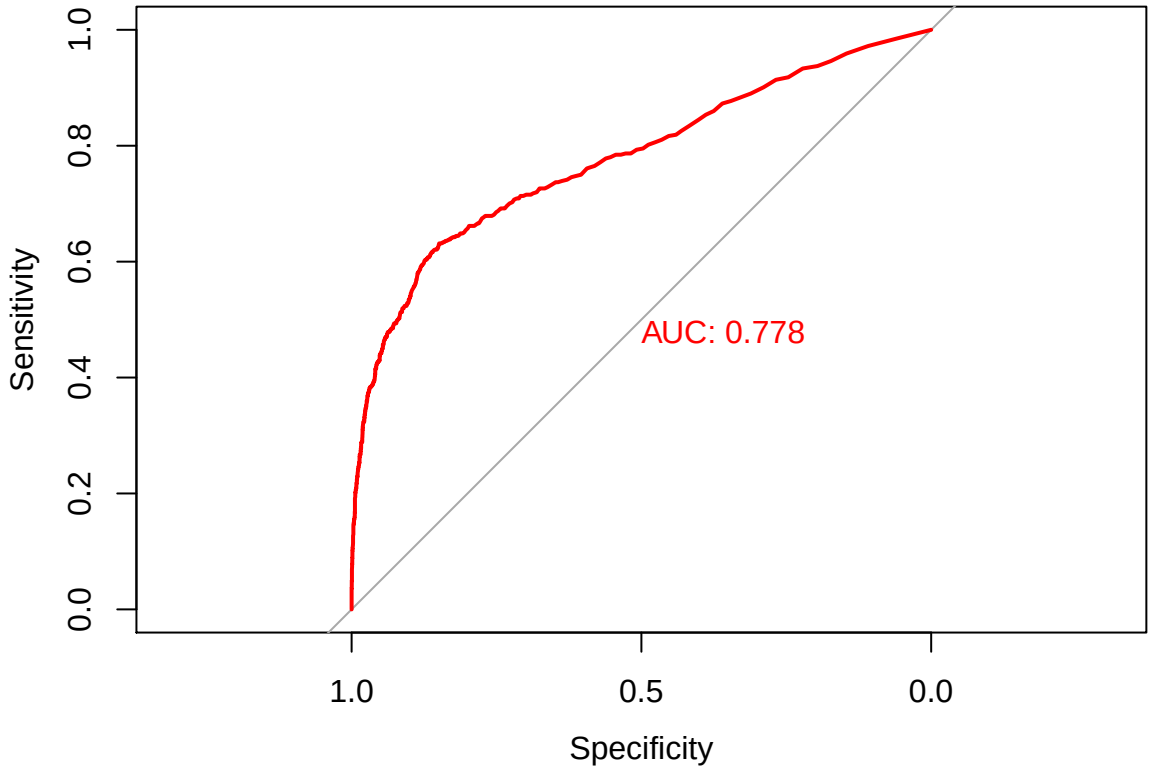


Now we train the algorithm with the whole train set with the best tune:

```
fit_rf <- Rborist(subset(preProc_train_set, select=-y),
                  train_set$y,
                  nTree = 1000,
                  minNode = train_rf$bestTune$minNode,
                  predFixed = train_rf$bestTune$predFixed,
                  verbose=TRUE)
```

Finally do the predictions on the test set and calculate AUC:

```
p_rf <- predict(fit_rf, subset(preProc_test_set, select=-y))$census
p_rf<- p_rf / rowSums(p_rf)


ROC <- roc(test_set$y, p_rf[,"yes"])
plot(ROC, col = "red", print.auc=TRUE)
```




```
auc_results <- rbind(auc_results,
                          tibble(method="Rborist",
                                     AUC = pROC::auc(ROC) ))
```

2.9.6 Support Vector Machines (SVM) Radial

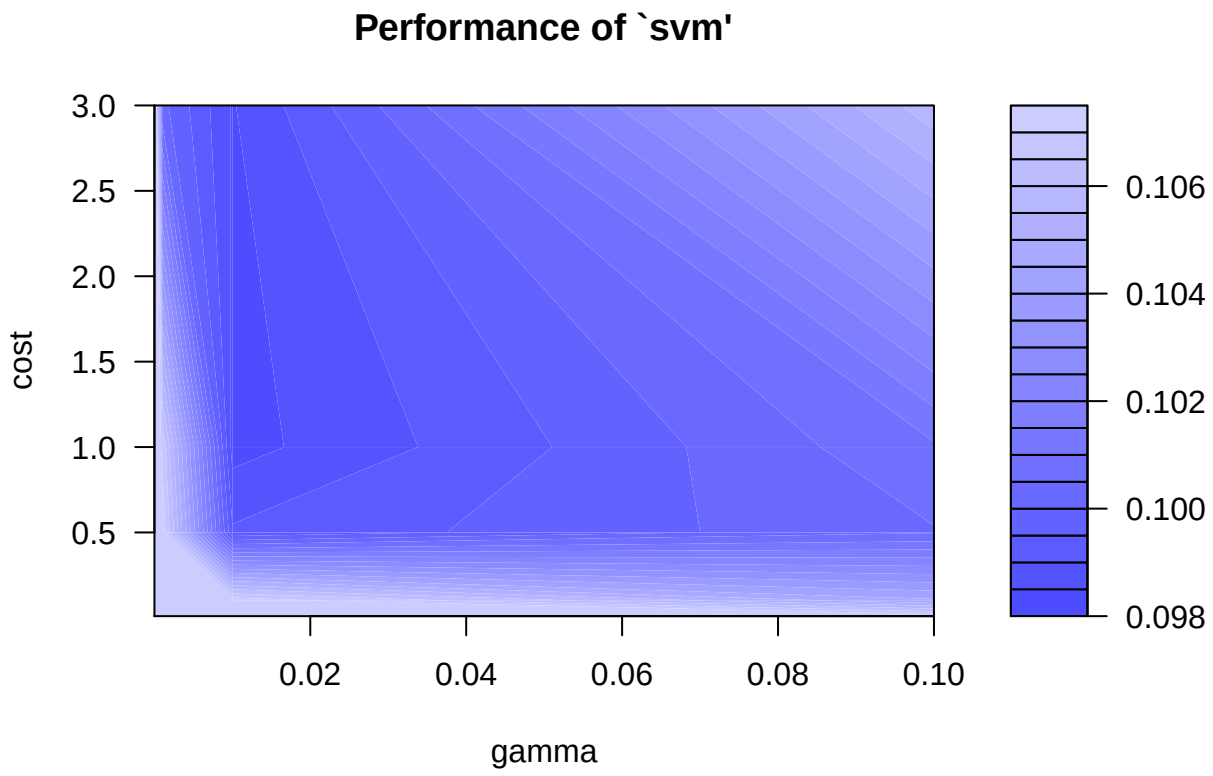
The last algorithm that we will train on this project is the Support Vector Machines radial. The execution of this algorithm can take around 30 minutes. Here we'll tune two hyperparameters: gamma and C (cost of constraints violation)

First we'll apply two preprocessing transformations on the data. Center and scale

```
preProc_svmRadial <- preProcess(subset(train_set, select=-y), method=c("center", "scale"))
preProc_train_set <- predict(preProc_nb,train_set)
preProc_test_set <- predict(preProc_nb,test_set)
```

And now we search the best hyperparameters with a subset of the train set.

```
tuned_parameters <- tune.svm(y~.,
                             data = preProc_train_set[index,],
                             gamma = 10^(-5:-1),
                             cost = c(0.01, 0.1, 0.5, 1, 3))


plot(tuned_parameters)
```

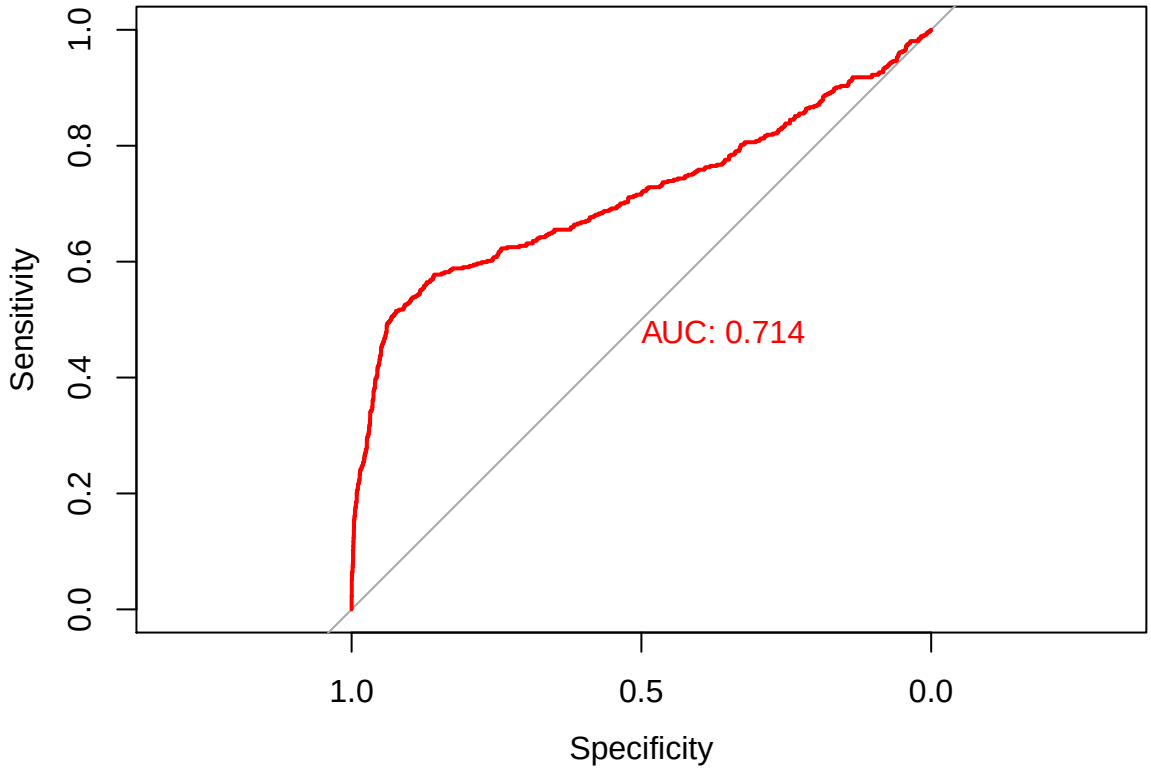


Next the model is trained with the best tune and the whole train set:

```
fit_svmRadial <- svm(y ~ .,
                     data=preProc_train_set,
                     kernel="radial",
                     cost=tuned_parameters$best.parameters$cost,
                     gamma=tuned_parameters$best.parameters$gamma,
                     probability=TRUE)
```

Finally get the predictions on the test set and calculate AUC:

```
pred <- predict(fit_svmRadial, preProc_test_set, probability=TRUE)

p_svmRadial <- attr(pred,"probabilities")

ROC <- roc(test_set$y, p_svmRadial[,"yes"])
plot(ROC, col = "red", print.auc=TRUE)
```

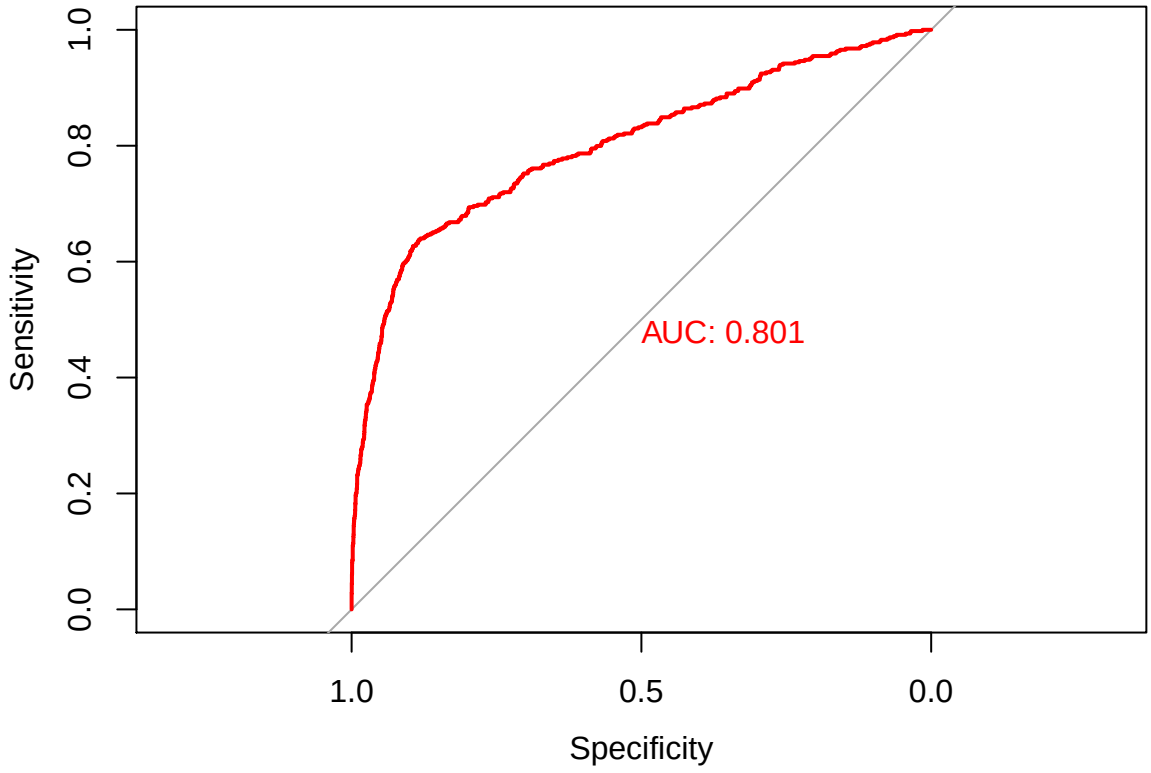



```
auc_results <- rbind(auc_results,
                          tibble(method="svmRadial",
                                     AUC = pROC::auc(ROC) ))
```

2.9.7 Ensemble of previous models

Finally we'll calculate the mean of the probabilities obtained from the previous models. The intent of this last step is to combine the output of the previous models to obtain better predictions:

```
#####################################
## Ensemble of previous models

p_ensemble <- (p_glm + p_glmnet + p_knn + p_nb + p_rf + p_svmRadial) / 6

ROC <- roc(test_set$y, p_ensemble[,"yes"])
plot(ROC, col = "red", print.auc=TRUE)
```




```
auc_results <- rbind(auc_results,
                          tibble(method="ensemble",
                                     AUC = pROC::auc(ROC) ))
```

3. Results

As we can see the ensemble gives the best AUC:

```
auc_results
```

```
## # A tibble: 7 x 2
##   method        AUC
##   <chr>       <dbl>
## 1 glm         0.794
## 2 glmnet      0.795
## 3 knn         0.795
## 4 Naive Bayes 0.788
## 5 Rborist     0.778
## 6 svmRadial   0.714
## 7 ensemble    0.801
```

The confusion matrix that we get with the ensemble of the five models:

```r
y_pred <- factor(ifelse(apply(p_ensemble, 1, which.max)-1==1, "yes", "no"))

cm_ensemble <- confusionMatrix(y_pred, test_set$y, positive="yes")

cm_ensemble
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  3597  346
##        yes   58  118
##
##                Accuracy : 0.9019
##                  95% CI : (0.8924, 0.9108)
##     No Information Rate : 0.8874
##     P-Value [Acc > NIR] : 0.001424
##
##                   Kappa : 0.3271
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.25431
##             Specificity : 0.98413
##          Pos Pred Value : 0.67045
##          Neg Pred Value : 0.91225
##              Prevalence : 0.11265
##          Detection Rate : 0.02865
##    Detection Prevalence : 0.04273
##       Balanced Accuracy : 0.61922
##
##        'Positive' Class : yes
##
```

3.2 Tradeof between accuracy and sensitivity

The results have an accuracy of 90%, but our sentivity is quite low.

But we can still do something. Remember that our objective is to optimize the resources of our marketing department. Imagine that the department has a budget to do 1000 calls.

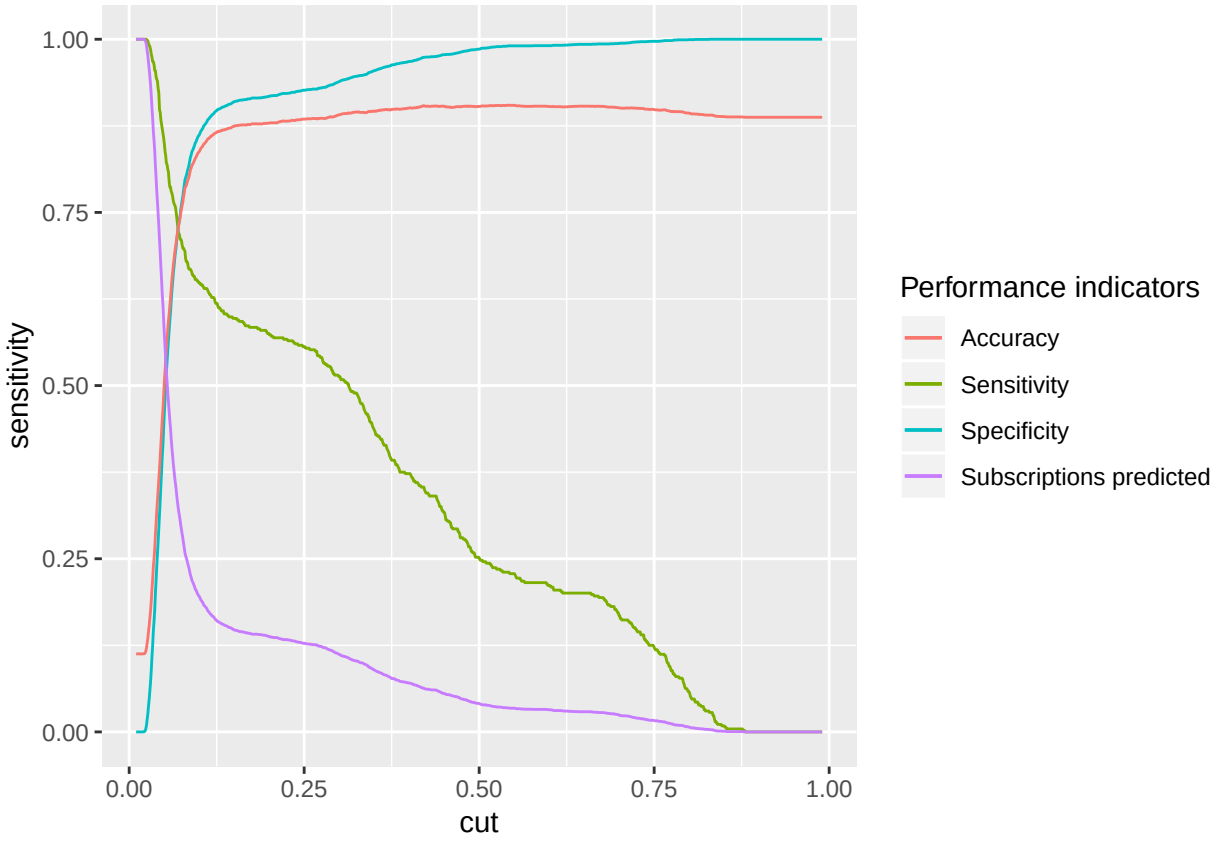
The variable 'p_ensemble' contains a structure with two columns "yes" and "no". The column "yes" contains the probabilities for each call to obtain the subscription. If we had budget for 1000 calls, we can simply get the probabilities obtained by our ensemble model and select the 1000 calls with the greatest probability to subscribe.

```r
selection <- test_set[order(p_ensemble$yes, decreasing=TRUE)[1:1000],]
sel_subscriptors <- sum(selection$y=="yes")
tot_subscriptors <- sum(test_set$y=="yes")
```

By selecting the 1000 out of 4119 best probabilities, we get 312 subscriptors. Not bad when the total number of possible subscriptors are 464.

Other approach can be trying different thresholds with our vector of probabilities. We consider y="yes" when the probability of being "yes" is greater than the threshold. This way we sacrifice some accuracy to get more sensitivity. On the next figure we show the effect of selecting different thresholds (x axis) on several performance indicators.

```
#############################################################################
## Accuracy - sensitivity trade off
#############################################################################

## performance indicators function
performance_indicators <- function(cut, probs, y) {
  y_hat = factor(ifelse(probs > cut, "yes", "no"), levels=c("yes","no"))
  w = which(y=="yes")
  sensitivity = mean( y_hat[w] == "yes" )
  specificity = mean( y_hat[-w] == "no" )
  accuracy = mean( y==y_hat )
  selection_rate = length(which(y_hat == "yes")) / length(y_hat)
  out = t(as.matrix(c(cut, sensitivity, specificity, accuracy, selection_rate)))
  colnames(out) = c("cut","sensitivity", "specificity", "accuracy", "pred_yes")
  return(out)

}

cuts <- seq(.01,.99,length=1000)

list_cut <- lapply(cuts, function(x) {performance_indicators(x,p_ensemble[,"yes"],test_set$y)})
df_cut <- data.frame(do.call(rbind, list_cut))

df_cut %>%
  ggplot() +
  geom_line(aes(x=cut,y=sensitivity, color="Sensitivity")) +
  geom_line(aes(x=cut,y=specificity, color="Specificity")) +
  geom_line(aes(x=cut,y=accuracy, color="Accuracy")) +
  geom_line(aes(x=cut,y=pred_yes, color="Subscriptions predicted")) +
  labs(color = 'Performance indicators')
```



Suppose that we call only to the rows predicted as subscriptions. Observe that the curve of subscriptions predicted has an elbow around a threshold of 0.11, which is the ratio of "yes" on the data set. This seems to be the optimal threshold.

4. Conclusions

With our Exploratory Data Analysis we have discarded some variables to be considered in our model, as they do not contribute much to predict subscriptions. In the other hand we have done a review of the variables with the most predictive power and we have gained some insights from them.

We have created a predictive model, that allow us to assign probabilities of success on a list of calls. This allows us to select the calls with the most chances of success if we have a limited budget.

We can improve this model by adding more algorithms to our ensemble. In the other hand we can improve the model as well by including new data or adding other variables.

5. Citations

[Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, In press, http://dx.doi.org/10.1016/j.dss.2014.03.001