Universitat Politécnica de Catalunya
Facultat d'Informàtica de Barcelona

COMPUTING & INTELLIGENT SYSTEMS

Lab 3 - Machine Learning

Robert Carausu
Marc Vila Gómez

January 17, 2018

Contents

1 Introduction **1**

1.1 Problem statement 1

1.2 Summary 2

2 Data analysis & preprocessing **3**

2.1 Continuous variables 3

2.2 Categorical variables 5

3 Modelling **7**

3.1 Logistic Regression 7

3.2 LDA, QDA & Naive Bayes 10

3.3 Random Forest 10

4 Conclusions **12**

5 Annex **13**

5.1 Code 13

1 Introduction

1.1 Problem statement

A Portuguese banking institution has made a **direct marketing** campaign (by means of phone calls) in order to make clients subscribe to certain financial product (a term deposit). The data has the following characteristics:

- Often, more than one contact to the same client was required, in order to assess if the product would be subscribed.
- Number of examples: 45,211; 16 predictors, of very different nature and type, including factors, '999' and 'unknown
- The **target** variable is whether a term deposit was subscribed ('yes') or not ('no')

The **input** (predictive variables) are:

- bank client data:
  1. age (numeric)
  2. job : type of job ("admin.", "unknown", "unemployed", "management", "housemaid", "entrepreneur", "student", "blue-collar", "self-employed", "retired", "technician", "services")
  3. marital : marital status ("married","divorced","single")
  4. education ("unknown", "secondary", "primary", "tertiary")
  5. default: has credit in default? ("yes", "no")
  6. balance: average yearly balance, in euros (numeric)
  7. housing: has housing loan? ("yes", "no")
  8. loan: has personal loan? ("yes", "no")
- related with the last contact of the current campaign:
  1. contact: contact communication type ("unknown", "telephone", "cellular")
  2. day: last contact day of the month (numeric)
  3. month: last contact month of year ("jan", "feb", "mar", ..., "nov", "dec")
  4. duration: last contact duration, in seconds (numeric)
- other variables:
  1. campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

2. pdays: number of days passed after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)
3. previous: number of contacts performed before this campaign and for this client (numeric)
4. poutcome: outcome of the previous marketing campaign (”unknown”, ”other”, ”failure”, ”success”)

The bank institution wants to predict the probability that a client will subscribe or no the deposit based on the previous data, so they segment the market and center their resources on the clients with a higher probability of subscribing it.

1.2 Summary

In this document we will perform the analysis and selection of the best classification method in order to solve the problem stated in the previous section, which is to correctly predict the class of the target variable *deposit subscribed* (**yes**), or not (**no**).

We will first perform an analysis of the data that we have available from the bank and do any processing that may be necessary to it in order to increase the efficiency of the methods that we'll test.

Then, we will test five different classifiers: **Logistic Regression**, **LDA**, **QDA**, **Naive Bayes** and **Random Forest**, and select the best one that fits our model based on its **error** and **precision**. The last one, **Random Forest**, is especially important since as we will find out the data is highly unbalanced, with just 11.70% of the clients subscribing to the deposit after the marketing campaign.

2 Data analysis & preprocessing

In this section we will discuss and analyze the data that was given to the students and also perform some preprocessing that may be necessary in order to perform the modelling.

2.1 Continuous variables

In Figure 1 we can see the continuous variables from the data before processing. The age variable seems to be ok, but the other variables are highly skewed and also need to be scaled. In order to fix these issues we'll be applying the *log* and *scale* functions wherever we can:

- balance: it has negative values so we can't apply log to fix the skewness without removing data, so we will just scale it.
- duration: we can fix both skewness and scale.
- campaign : if we apply scale and log it has some undesired effects (as seen in Figure 2) so we leave it as it is.
- previous: if we apply scale and log it has some undesired effects (as seen in Figure 2) so we leave it as it is.
- pdays: most of its values are -1 (which means that the client was not contacted in a previous campaign) so we decide to create a categorical variable with *not_contacted* and *contacted* as categories.

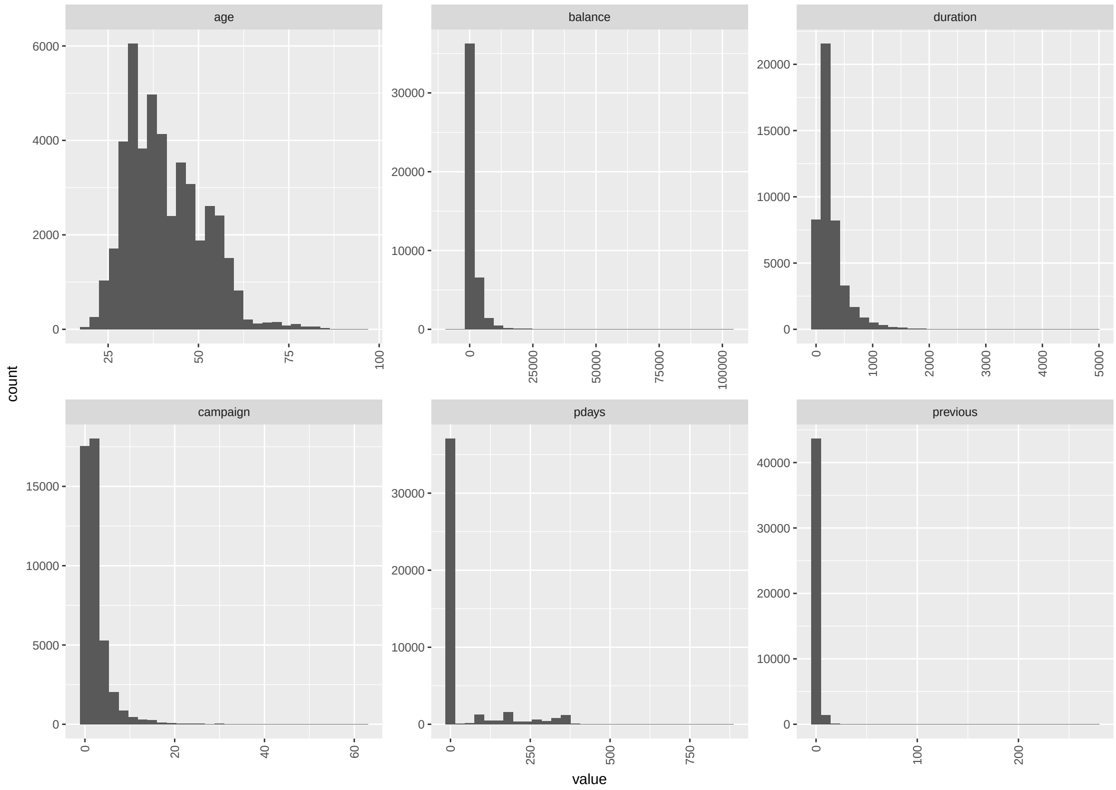

Figure 1: Continuous variables before preprocessing

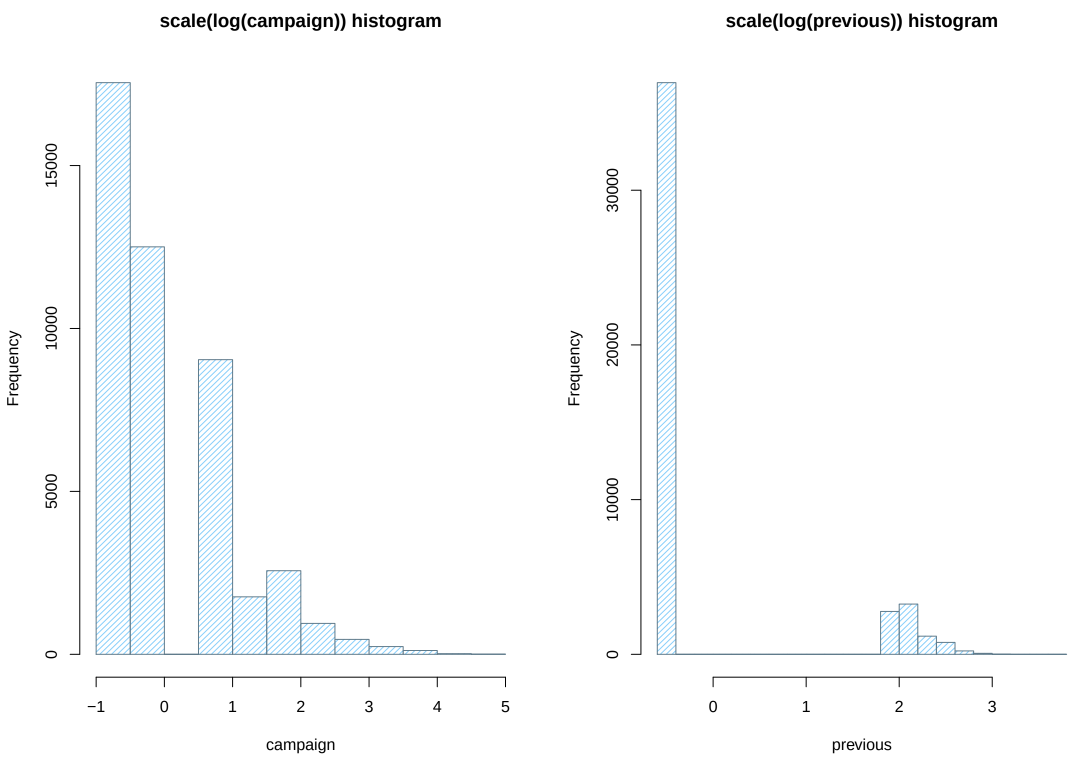

Figure 2: Applying log and scale to campaign and previous variables

After applying all the changes mentioned in the previous paragraphs, the continuous variables remain as seen in Figure 3. The *pdays* variable is shown in Figure 4 with the rest of the categorical variables.

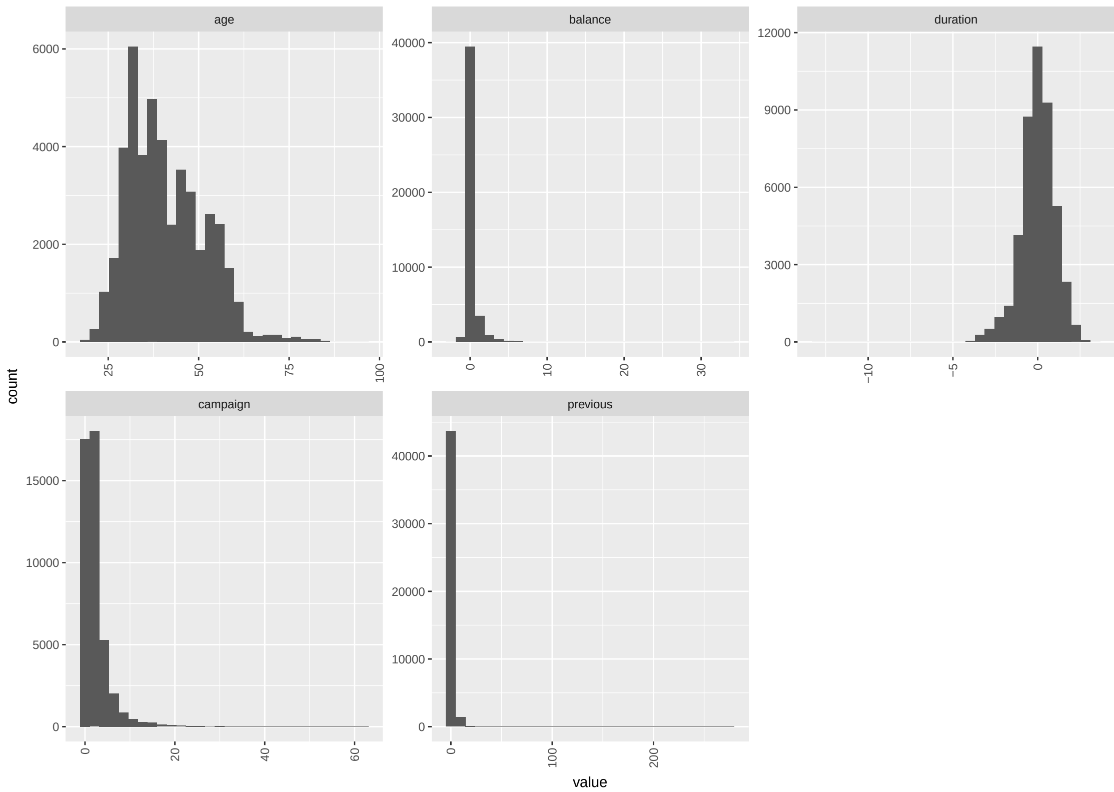

Figure 3: Continuous variables after preprocessing

2.2 Categorical variables

In Figure 4 we can see the categorical variables. As we can see, most of the variables seem ok except for *job*, *education*, *contact*, and *poutcome*, which the missing observations are labeled as *unknown*. Since we don't want do remove any of the data we will leave the categorical variables as they are. We could replace them for the **NA** value in R, but since only Random Forest can deal with this kind of variables we choose to not modify anything.

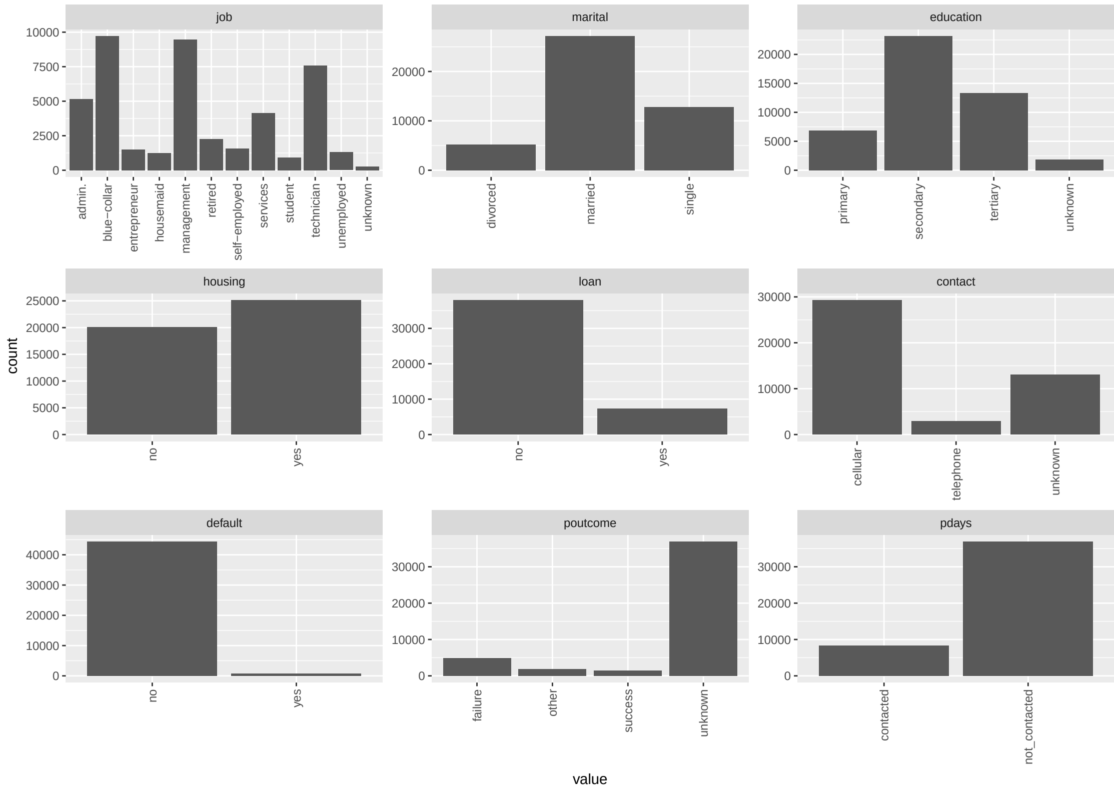

Figure 4: Categorical variables

3 Modelling

In this section we will be discussing the different classifiers that we will use to fit our data and make predictions. We will use five different classifiers as stated before: Logistic Regression, LDA, QDA, Naive Bayes and Random Forest. We will compare them by two different parameters, **error** and **precision**, this last one being of special importance since the bank is more interested in the people that may want to subscribe to their deposit, so the resources of the marketing campaign can be dedicated to contacting these clients.

Because of the nature of the data, we expect Logistic Regression and Random Forest to be the ones that perform better. In order to fit our models, we will be dividing our data in two different sets: **learning** and **testing**, the first one being used to fit the models and the second one to test it and make predictions, in a 2/3 and 1/3 proportion, respectively.

The details of the implementation can be seen in the code provided in the annex or in the *.R* file included in the folder with this document.

3.1 Logistic Regression

After fitting the first model with the original data, we inspect the *p-values* and observe that there are a few variables that have a really low impact in the target variable (the higher the *p-value* the lower the impact), these are: *age*, *job*, *marital*, *default*, *balance*, *pdays* and *previous*.

So we decide to do four different fits:

- Original data, with all the variables.
- Original data, without the variables mentioned.
- Pre-processed data, with all the variables.
- Pre-processed data, without the variables mentioned.

After fitting the first model, we find that the error is **9.93%** and the precision is **34.15%**. This is a pretty low precision, it makes us miss most of the clients that would said yes to subscribing the deposit. In order to improve this value and balance the error we lower the threshold for a positive subscription from 0.5 to 0.3, now the error is **9.97%** (a very small increase), but the precision is **53.78%**, almost doubled, so we decide to make all the fits with this new value for the threshold.

The results can be observed in Table 1, we see that the fact that we fit without the discarded variables doesn't improve the error and precision of the model (in fact it seems to lower it), so we will be using all of the variables from now on. But there is an important improvement in using the pre-processed data, we can see that the precision has improved by more than 4 points, which is quite good.

Variables	**Data**	**Error**	**Precision**
All	Original	9.97%	53.78%
Less	Original	10.07%	53.32%
All	Pre-processed	10.60%	57.95%
Less	Pre-processed	10.53%	57.44%

Table 1: Error and precision for logistic regression

In order to get a better grasp of the performance of the classifier we performed *k-fold Cross Validation* dividing the data in 10 equal divisions, using 1 for validation and 9 for training, and performing it 100 times since we observed that the computation time wasn't too high. The results can be observed in Table 2 which confirms our previous suppositions. In Figure 5 and Figure 6 there are the histograms and box plots for the accuracy, error and precision for logistic regression with all the variables after the cross validation

Variables	**Data**	**Error**	**Precision**
All	Pre-processed	10.31%	58.80%
Less	Pre-processed	10.26%	58.37%

Table 2: Mean values for error and precision for logistic regression with cross validation

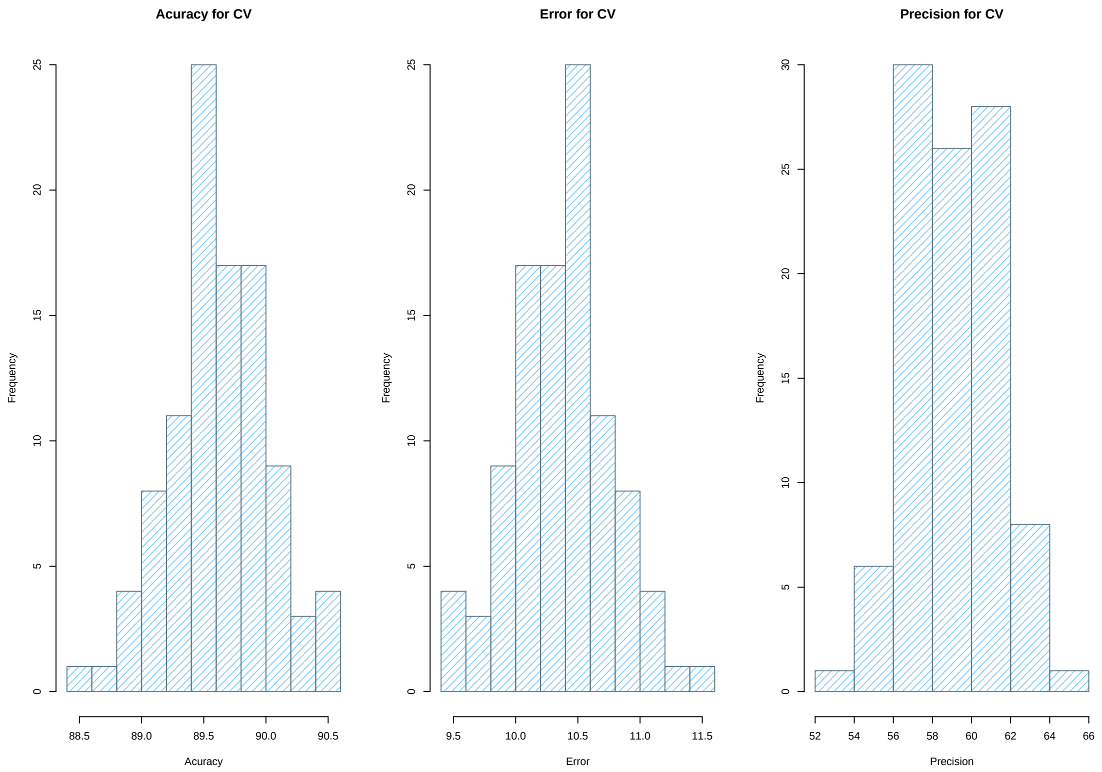

Figure 5: Histograms of accuracy, error and precision for logistic regression

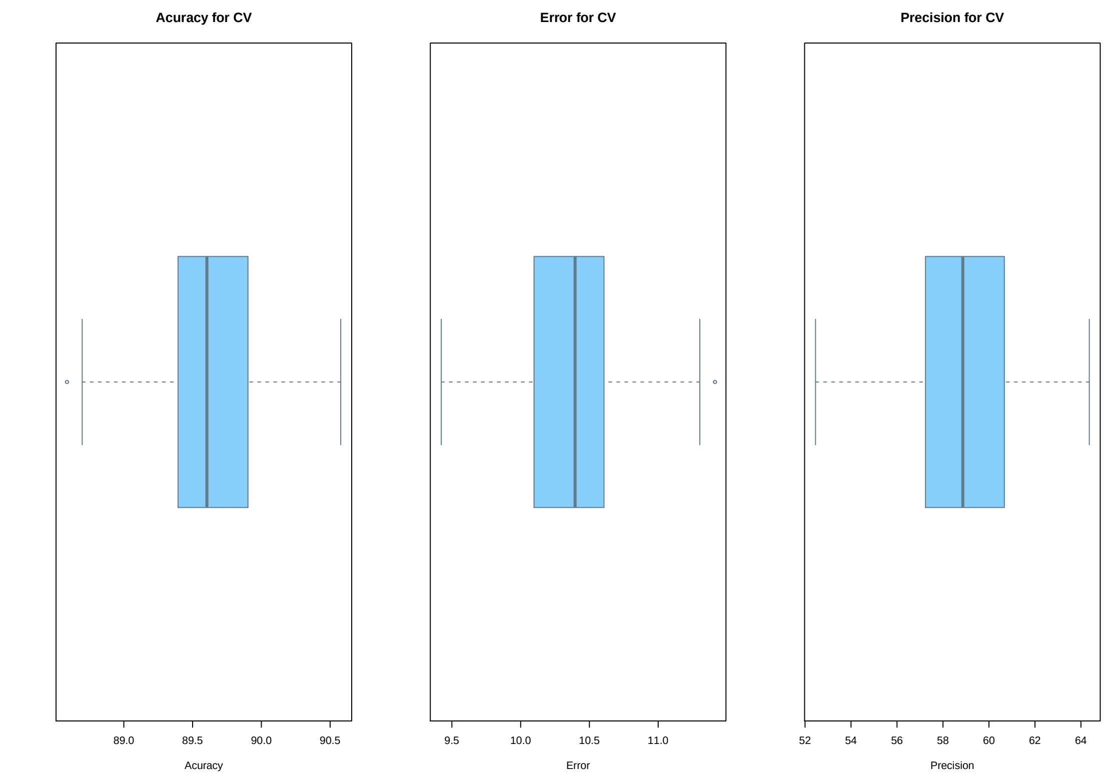

Figure 6: Box plots of accuracy, error and precision for logistic regression

3.2 LDA, QDA & Naive Bayes

We also performed the fitting for three other classifiers, LDA, QDA and Naive Bayes. The results can be observed in Table 3. We can see that as predicted, LDA and QDA performed rather poorly, both with a precision way lower than 50%, the limit with which consider a prediction better than randomly assigning the observations to subscribed or not subscribed. The only one that got close to this value is Naive Bayes but it still has a high error, so we consider these three classifiers not good for our model.

Model	Error	Precision
LDA	10.73%	28.51%
QDA	13.43%	37.99%
Naive Bayes	12.57%	43.89%

Table 3: Results for LDA, QDA and Naive Bayes

3.3 Random Forest

The last classifier that we tested is Random Forest. The results of it can be observed in Table 4. Because of the nature of this classifier, we used the original untreated data with a number of trees of 300, and we got the results in the first row of the table mentioned before. So far this is the classifier with the best results, which is the one that we will decide for, since it has the lowest error and highest precision. We decided to do a 10-fold Cross Validation to make sure the values we were getting were consistent, this wasn't really necessary because of how this classifier works, but the results we got were pretty much the same as the first iteration.

In Figure 7 and Figure 8 we can observe the histograms and box plots for the accuracy, error and precision for this method. **Because of the better results of random forest, we would decide for this method to model the data for the marketing campaign of the bank.**

Model	Error	Precision
Random Forest	9.30%	64.66%
Random Forest (10-fold CV)	9.17%	64.66%

Table 4: Results for Random Forest classifier

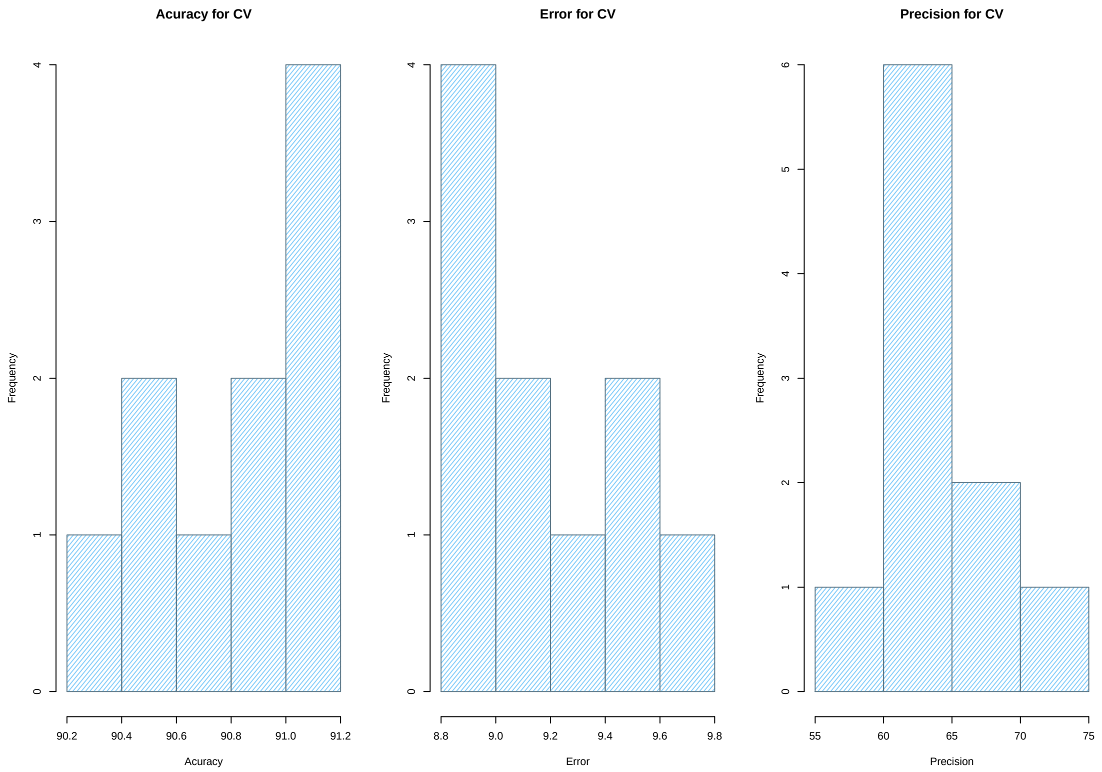

Figure 7: Histograms of accuracy, error and precision for random forest

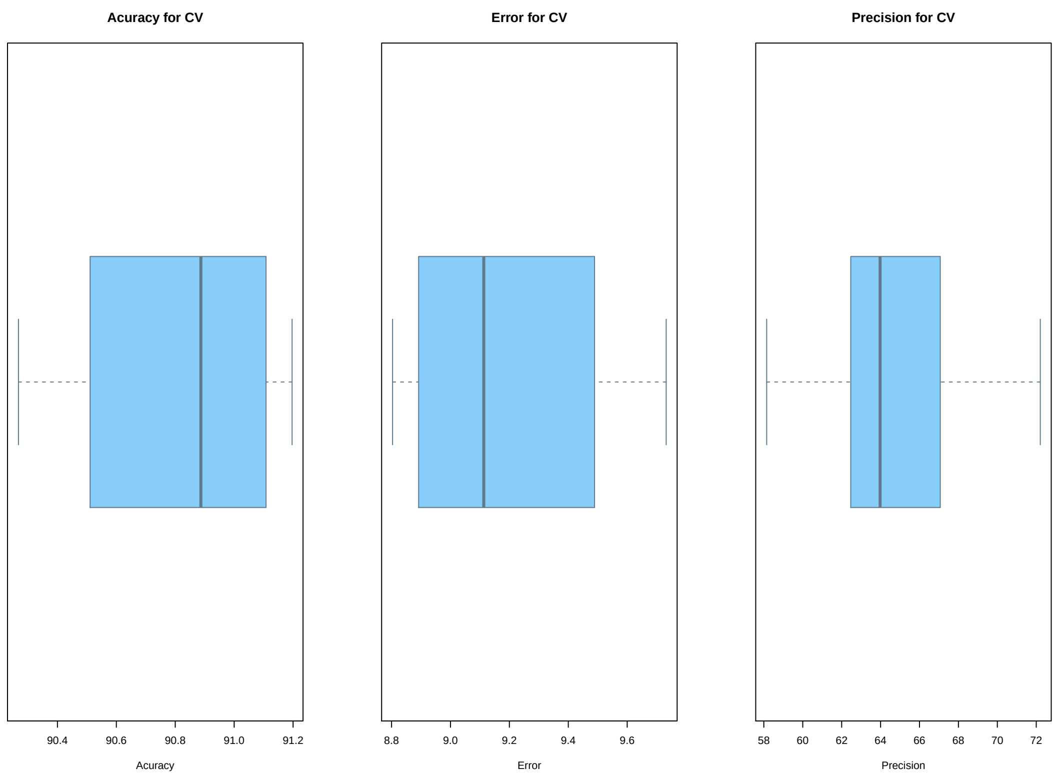

Figure 8: Box plots of accuracy, error and precision for random forest

4 Conclusions

In this document we made the analysis and comparison of five different classifiers for the problem stated in the introduction. We got the opportunity to observe how important is the preparation of the data in order to get the best results for our model, and also how choosing the appropriate classifier can mean success or failure in addressing the problem correctly.

In this particular case, we observed that the best method was Random Forest. This is because we were working with a particular set of data, which was very unbalanced and has a lot of different variables that make it more difficult to find a correlation between them and the target variable.

In general we are satisfied with the results obtained, since a precision of more than 60% means that the bank is not making random guessing in what type of client to invest time in selling its products. This values of course can probably be improved with more dedication and knowledge from our part.

5 Annex

5.1 Code

```r
#####################################################################
#################### Project 1: the Bank Marketing Data Set
#################### Robert Carausu & Marc Vila, CSI - MEI 2017-2018
#####################################################################
set.seed (6046)
library(reshape2)
library(ggplot2)

## Direct marketing campaigns (phone calls) of a Portuguese banking institution.
## The classification goal is to predict if the client will subscribe a term deposit

## Getting the dataset
deposit <- read.table(file="./data/bank.csv", header=TRUE, stringsAsFactors=TRUE, sep=";")
# We rename the target variable
colnames(deposit)[ncol(deposit)] <- "subscribed"
original_data = deposit # We make a copy to compare it later with our pre-processed data

# 45211 observations and 17 different variables
# (9 categorical: job, marital, education, default, housing, loan, contact, month, poutcome and y)
dim(deposit)
summary(deposit)
# 11.70% of subscribed, so our model sholdn't have a higher error than this
# Data is very unbalanced so some models will adjust worse than others
sum(deposit$subscribed=="yes")/sum(length(deposit$subscribed))*100

## Let's have a visual inspection of the continuous variables before pre-processing
# Age seems ok
```

```r
# The other variables are highly skewed so we will try to scale and apply log where we can
# We can do it for duration, not for balance since it has negative values and we don't want to lose data
# pdays can be converted to categorical: "not_contacted" (in previous campaign) and "contacted"
d.cont <- melt(deposit[, c("age", "balance", "duration", "campaign", "pdays", "previous")])
ggplot(d.cont, aes(x = value)) + facet_wrap(~variable, scales = "free") + geom_histogram() + theme(axis.text.x=element_text(angle=90, hjust=1, vjust=0.5))

## Let's have a visual inspection of the factor variables before pre-processing
# They seem ok so we won't be touching these variables
d.categ <- melt(deposit, measure.vars=c("job", "marital", "education", "housing", "loan", "contact", "default", "poutcome"))
ggplot(d.categ, aes(x = value)) + facet_wrap(~variable, scales = "free") + geom_bar() + theme(axis.text.x=element_text(angle=90, hjust=1, vjust=0.5))

# This dataset needs a lot of pre-processing ... also it displays a good mixture of categorical and numeric variables
# In conclusion LDA/QDA may not the best choice, a good choice may be Logistic Regression. We will test also Naive Bayes and Random Forest and
# choose the best model that fits our problem

#### PRE-PROCESSING ####
#### Fixing skewness and scaling continuous variables
# The balance has negative values, so we can only scale it
hist(deposit$balance, col='lightskyblue', border='lightskyblue4', xlab='balance', main='balance histogram', density=50)
# There are 3766 for negative balance
# The only way to fix it is to delete this observations so we choose to leave it as it is since we don't want to lose data
sum(deposit$balance<0)
deposit$balance = scale(deposit$balance)
```

```r
# Scaled balance
hist(scale(deposit$balance), col='lightskyblue', border='lightskyblue4', xlab='balance', main='balance histogram', density=50)

# duration, campaign and previous are all skewed, so we apply log and scale
hist(deposit$duration, col='lightskyblue', border='lightskyblue4', xlab='duration', main='duration histogram', density=50)
deposit$duration = log(deposit$duration+0.001) # +0.001 to avoid -Inf
hist(deposit$duration, col='lightskyblue', border='lightskyblue4', xlab='duration', main='duration histogram', density=50)
deposit$duration = scale(deposit$duration)
hist(deposit$duration, col='lightskyblue', border='lightskyblue4', xlab='duration', main='duration histogram', density=50)

# Applying log and scale to campaign and previous has some undesired effects, so we will leave them as they are
hist(scale(log(deposit$campaign + 0.001)), col='lightskyblue', border='lightskyblue4', xlab='campaign', main='scale(log(campaign)) histogram', density=50)
hist(scale(log(deposit$previous + 0.001)), col='lightskyblue', border='lightskyblue4', xlab='previous', main='scale(log(previous)) histogram', density=50)

# pdays has most of values -1 (not contacted previously).
# We make a categorical value with "contacted" for pdays!=-1 and "not contacted" previously for pdays=-1
hist(deposit$pdays, col='lightskyblue', border='lightskyblue4', xlab='pdays', main='pdays histogram', density=50)
deposit$pdays = cut(deposit$pdays, breaks=c(-Inf, 0.0, Inf), labels=c("not_contacted", "contacted"))
table(deposit$pdays)
plot(deposit$pdays)

#### Fixing "unknown" values

# There are 288 subscriptions for unknown job, we leave it as it is since we don't want to delete this data
```

```r
summary(deposit[deposit$job=="unknown",])

# We could change the unknown values to NA (as well as the 0 previous contacts variables), this is useful if we use a Random Forest algorythm,
# but since it is not the case we leave it as it is

# We plot again after pre-processing
d.cont <- melt(deposit[, c("age", "balance", "duration", "campaign", "previous")])
ggplot(d.cont, aes(x = value)) + facet_wrap(~variable, scales = "free") + geom_histogram() + theme(axis.text.x=element_text(angle=90, hjust=1, vjust=0.5))

# Now pdays is categorical
d.categ <- melt(deposit, measure.vars=c("job", "marital", "education", "housing", "loan", "contact", "default", "poutcome", "pdays"))
ggplot(d.categ, aes(x = value)) + facet_wrap(~variable, scales = "free") + geom_bar() + theme(axis.text.x=element_text(angle=90, hjust=1, vjust=0.5))

library(caret)
library(MASS)
library(e1071)
library(randomForest)

# PREPARING THE TRAINING AND TEST DATA
## Since we want to use different methods, we need CV and a separate test set:

N <- nrow(deposit)
all.indexes <- 1:N

learn.indexes <- sample(1:N, round(2*N/3))
test.indexes <- all.indexes[-learn.indexes]

learn.data <- deposit[learn.indexes,]
original.learn.data <- original_data[learn.indexes,]
```

```r
test.data <- deposit[test.indexes,]
original.test.data <- original_data[test.indexes,]

nlearn <- length(learn.indexes)
ntest <- N - nlearn

#### MODELLING ####
########### LOGISTIC REGRESSION ##########
# We use Logistic Regression as recommended since it doesn't need a lot of preprocessing of the data and we also have a lot of categorical variables

# ORIGINAL DATA
# First aproximation with the original unchanged data & all variables
glm.fit = glm(subscribed~., data=original.learn.data, family="binomial")

# Observing the p-values, we can have an idea of the variables that have more importance in predicting our model,
# a low p-value indicates that we can reject the null hipotesis, thus that variable has an importance on our model,
# a higher p-value means that we can discard that variable
# so we can fit the mode again with just the variable that actually have an influence on our model
# We can discard the following since they affect our model less: age, job, marital, default, balance, pdays and previous
summary(glm.fit)

# We calculate the prediction with and without the discarded variables and compare the errors
glm.probs = predict(glm.fit, original.test.data, type="response")
glm.pred = rep("no", length(glm.probs))
glm.pred[glm.probs>.5] = "yes"

# We choose 3 values to represent our model performance: accuracy, error and precision, the last one is important
# because the bank wants to contact only those clients that are more probable to subscribe to the loan
```

```r
# We can see that accuracy is high (90.07 %), but precission is low (34.15%), to solve this we lower the threshold
# for which a client may subscribe a loan (the probability) compare the values again, since for the bank clients
# with 30% probability of subscribing is probably worth to spend it's ressources contacting them
res.performance = table(glm.pred, original.test.data$subscribed)
res.accuracy = (res.performance[2,2] + res.performance[1,1])/sum(res.performance)*100
res.error = 100 - res.accuracy
res.precision = (res.performance[2,2])/(res.performance[2,2] + res.performance[1,2])*100

# Accuracy is slightly lower (90.03%) but precission has almost doubled (53.78%)
glm.pred[glm.probs>.3] = "yes"
res.performance = table(glm.pred, original.test.data$subscribed)
res.accuracy = (res.performance[2,2] + res.performance[1,1])/sum(res.performance)*100
res.error = 100 - res.accuracy
res.precision = (res.performance[2,2])/(res.performance[2,2] + res.performance[1,2])*100

# Now we fit the model without the variables that had less of an impact
glm.fit = glm(subscribed~.-age-job-marital-default-balance-pdays-previous, data=original.learn.data, family="binomial")
glm.probs = predict(glm.fit, original.test.data, type="response")
glm.pred = rep("no", length(glm.probs))

# The total accuracy decreases to 89.93% and precission to 53.32%, so using less variables makes our model a bit less accurate
# but the difference is really small so it's not really important to discard those variables
# If we have too many variables and computation time is important,
# we can also see that removing the ones we selected won't affect so much our model prediction
glm.pred[glm.probs>.3] = "yes"
res.performance = table(glm.pred, original.test.data$subscribed)
```

```r
res.accuracy = (res.performance[2,2] + res.performance[1,1])/sum(res.performance)*100
res.error = 100 - res.accuracy
res.precision = (res.performance[2,2])/(res.performance[2,2] + res.performance[1,2])*100

# PREPROCESSED DATA
# We will fit with all the variables and also removing the ones that we mentioned before
glm.fit = glm(subscribed~., data=learn.data, family="binomial")
glm.probs = predict(glm.fit, test.data, type="response")
glm.pred = rep("no", length(glm.probs))

# Accuracy is 89.40% and precission is 57.95%, so our model is much more precise detecting clients
# that will probably buy the finantial product of the bank with the preprocessed data
glm.pred[glm.probs>.3] = "yes"
res.performance = table(glm.pred, original.test.data$subscribed)
res.accuracy = (res.performance[2,2] + res.performance[1,1])/sum(res.performance)*100
res.error = 100 - res.accuracy
res.precision = (res.performance[2,2])/(res.performance[2,2] + res.performance[1,2])*100

# Now we fit the model without the variables that had less of an impact
glm.fit = glm(subscribed~.-age-job-marital-default-balance-pdays-previous, data=learn.data, family="binomial")
glm.probs = predict(glm.fit, test.data, type="response")
glm.pred = rep("no", length(glm.probs))

# Accuracy: 89.47%, precision: 57.44%
# As before, there is a small reduction in accuracy and precision but the results with preprocessed data are better
glm.pred[glm.probs>.3] = "yes"
res.performance = table(glm.pred, original.test.data$subscribed)
res.accuracy = (res.performance[2,2] + res.performance[1,1])/sum(res.performance)*100
```

```
res.error = 100 - res.accuracy
res.precision = (res.performance[2,2])/(res.performance[2,2] + res.performance[1,2])*100

#### To get a better grasp at the performance of our model, we do k-fold cross validation
precision <- NULL
accuracy <- NULL
error <- NULL
k <- 100

# It may take a while to compute
for (i in 1:k)
{
  N <- nrow(deposit)
  all.indexes <- 1:N

  # we choose 9/10s of the data as training data and the rest as test data
  learn.indexes <- sample(1:N, round(9*N/10))
  test.indexes <- all.indexes[-learn.indexes]

  learn.data <- deposit[learn.indexes,]
  test.data <- deposit[test.indexes,]

  nlearn <- length(learn.indexes)
  ntest <- N - nlearn
  glm.fit = glm(subscribed ~ ., data=learn.data, family="binomial")
  #glm.fit = glm(subscribed ~ .-age-job-marital-default-balance-pdays-previous, data=learn.data, family="binomial")
  glm.probs = predict(glm.fit, test.data, type="response")

  glm.pred = rep("no", length(glm.probs))
  glm.pred[glm.probs>.3] = "yes"

  res.performance = table(glm.pred, test.data$subscribed)
```

```
  accuracy[i] <- (res.performance[2,2] + res.performance[1,1])/sum(res.performance)*100
  error[i] <- 100 - accuracy[i]
  precision[i] <- (res.performance[2,2])/(res.performance[2,2] + res.performance[1,2])*100
}

# We can see that our model performs pretty well, even though the data is highly unbalanced
# Mean values with all the variables
# accuracy: 89.69%
# error: 10.31%
# precision: 58.80 %

# Mean values without the variables that influence less our model (swap the commented code in the previous bucle)
# accuracy: 89.74%
# error: 10.26%
# precision: 58.37 %
mean(accuracy)
mean(error)
mean(precision)

par(mfrow=c(1,3))
hist(accuracy, col='lightskyblue', border='lightskyblue4', xlab='Acuracy', main='Acuracy for CV', density=50)
hist(error, col='lightskyblue', border='lightskyblue4', xlab='Error', main='Error for CV', density=50)
hist(precision, col='lightskyblue', border='lightskyblue4', xlab='Precision', main='Precision for CV', density=50)

boxplot(accuracy, horizontal=T, col='lightskyblue', border='lightskyblue4', xlab='Acuracy', main='Acuracy for CV')
boxplot(error, horizontal=T, col='lightskyblue', border='lightskyblue4', xlab='Error', main='Error for CV')
boxplot(precision, horizontal=T, col='lightskyblue', border='lightskyblue4', xlab='Precision', main='Precision for CV')
dev.off()
```

```

# To compare the performance of our model we will also model with LDA and QDA and analyze their performances.
# Also we will test NaiveBayes and RandomForest
N <- nrow(deposit)
all.indexes <- 1:N

learn.indexes <- sample(1:N, round(2*N/3))
test.indexes <- all.indexes[-learn.indexes]

learn.data <- deposit[learn.indexes,]
test.data <- deposit[test.indexes,]

nlearn <- length(learn.indexes)
ntest <- N - nlearn

################### LDA #################
# With LDA the precision is much lower, so we won't be using this model
# 10.73% error, 89.27% accuracy, 28.51% precision
lda.fit = lda(subscribed ~ ., data=learn.data)
lda.pred = predict(lda.fit, test.data)
lda.class = lda.pred$class

res.performance = table(lda.class, test.data$subscribed)
res.accuracy = (res.performance[2,2] + res.performance[1,1])/sum(res.performance)*100
res.error = 100 - res.accuracy
res.precision = (res.performance[2,2])/(res.performance[2,2] + res.performance[1,2])*100

################### QDA #################
# Performs worse than LDA, but the precision is a bit higher so it detects better the subscriptions
# We confirm that both LDA and QDA are not suitable models to fit our problem
# 13.43% error, 86.57% accuracy, 37.99% precision
qda.fit <- qda(subscribed ~ ., data=learn.data)
qda.pred = predict(qda.fit, test.data)
```

```
qda.class = qda.pred$class

res.performance = table(qda.class, test.data$subscribed)
res.accuracy = (res.performance[2,2] + res.performance[1,1])/sum(res.performance)*100
res.error = 100 - res.accuracy
res.precision = (res.performance[2,2])/(res.performance[2,2] + res.performance[1,2])*100

################# NAIVE BAYES ##################
# It performs better than LDA and QDA, but worse than logistic regression
# 12.57% error, 87.43% accuracy, 43.89% precision
bayes.fit <- naiveBayes(subscribed ~ ., data = learn.data)
bayes.pred <- predict(bayes.fit, test.data)

res.performance = table(bayes.pred, test.data$subscribed)
res.accuracy = (res.performance[2,2] + res.performance[1,1])/sum(res.performance)*100
res.error = 100 - res.accuracy
res.precision = (res.performance[2,2])/(res.performance[2,2] + res.performance[1,2])*100

#################### Random Forest #################
# 9.30% error, 90.70% accuracy, 64.66% precision, so far the best method
rf <- randomForest(subscribed ~ ., data = original_data[learn.indexes,], ntree=300, proximity=FALSE)
rf.pred <- predict(rf, newdata=original_data[-learn.indexes,])

res.performance = table(Truth=original_data[-learn.indexes,]$subscribe,Pred=rf.pred)
res.accuracy = (res.performance[2,2] + res.performance[1,1])/sum(res.performance)*100
res.error = 100 - res.accuracy
res.precision = (res.performance[2,2])/(res.performance[2,2] + res.performance[1,2])*100

#### As with logistic regresion we do k-fold CV to confirm our model accuracy and precision
```

```r
# We choose k=10 since randomForest has a high computation time
precision <- NULL
accuracy <- NULL
error <- NULL
k <- 10

# It may take quite a while to compute
for (i in 1:k)
{
  N <- nrow(deposit)
  all.indexes <- 1:N

  # we choose 9/10s of the data as training data and the rest as test data
  learn.indexes <- sample(1:N, round(9*N/10))
  test.indexes <- all.indexes[-learn.indexes]

  nlearn <- length(learn.indexes)
  ntest <- N - nlearn
  #glm.fit = glm(subscribed ~ ., data=learn.data, family="binomial")
  rf <- randomForest(subscribed ~ ., data = original_data[learn.indexes,], ntree=300, proximity=FALSE)
  rf.pred <- predict(rf, newdata=original_data[-learn.indexes,])

  res.performance = table(Truth=original_data[-learn.indexes,]$subscribe,Pred=rf.pred)
  accuracy[i] <- (res.performance[2,2] + res.performance[1,1])/sum(res.performance)*100
  error[i] <- 100 - accuracy[i]
  precision[i] <- (res.performance[2,2])/(res.performance[2,2] + res.performance[1,2])*100
}
# Mean values
# accuracy: 90.83%
# error: 9.17 %
# precision: 64.66%
mean(accuracy)
mean(error)
```

```r
mean(precision)

par(mfrow=c(1,3))
hist(accuracy, col='lightskyblue', border='lightskyblue4', xlab='Acuracy', main='Acuracy for CV', density=50)
hist(error, col='lightskyblue', border='lightskyblue4', xlab='Error', main='Error for CV', density=50)
hist(precision, col='lightskyblue', border='lightskyblue4', xlab='Precision', main='Precision for CV', density=50)

boxplot(accuracy, horizontal=T, col='lightskyblue', border='lightskyblue4', xlab='Acuracy', main='Acuracy for CV')
boxplot(error, horizontal=T, col='lightskyblue', border='lightskyblue4', xlab='Error', main='Error for CV')
boxplot(precision, horizontal=T, col='lightskyblue', border='lightskyblue4', xlab='Precision', main='Precision for CV')
dev.off()

# To conclude, random forest is the best method, followed by logistic regression, according to the results
```