PRACTICAL WORK ON MACHINE LEARNING

Classification on bank marketing data

Martí Cardoso i Sabé
Meysam Zamani Forooshani

ML (MIRI - Data Science)

June, 2019

Table of contents

1 Introduction

In this document, we explain the problem that we want to tackle for the ML course project. Our project is called *Classification on bank marketing data* and it aims to predict if a client will subscribe a term deposit or not. In the following sections, we will explain all the technical details starting from where we obtained our dataset, what is our personal motivation for the topic, the main objective to achieve, pre-processing and exploratory analysis, the modeling part and the choice of the best model, and finally the conclusions of our work.

2 Description of the problem and available data

Our problem has data related with direct marketing campaigns of a Portuguese banking institution, campaigns that were based on phone calls. The main goal of the problem is to predict if a client will subscribe a term deposit or not, so it is a binary **classification** problem. Often, more than one call to the same client was required, in order to know if the bank term deposit would be subscribed (or not).

2.1 Data Source

The dataset of our problem is called **Bank Marketing Data Set** [7] and it can be found in the *UCI Machine Learning Repository*. It is available online on the following URL:

`https://archive.ics.uci.edu/ml/datasets/bank+marketing`

The Bank Marketing dataset has **20 explanatory** variables (10 of which are continuous, 7 categorical and 3 binary) and the binary response variable (called y). These explanatory variables are:

1. *age*: Age (cont.)
2. *job*: Type of job (cat.)
3. *marital*: Marital status (cat.)
4. *education*: Education (cat.)
5. $default$: Has credit in default? (binary)
6. *housing*: Has housing loan? (binary)
7. *loan*: Has personal loan? (binary)
8. *contract*: Contact communication type (cat.)
9. *month*: Last contact month of year (cat.)
10. day_of_week: Last contact day of the week (cat.)
11. *duration*: Last contact duration, in seconds (cont.)
12. *campaign*: Number of contacts performed during this campaign and for this client (cont.)
13. *pdays*: Number of days that passed by after the client was last contacted from a previous campaign (cont.)
14. *previous*: Number of contacts performed before this campaign and for this client (cont.)
15. *poutcome*: Outcome of the previous marketing campaign (cat.)
16. *emp.var.rate*: Employment variation rate (cont.)
17. *cons.price.idx*: Consumer price index (cont.)
18. $cons.conf.idx$: Consumer confidence index (cont.)
19. $euribor3m$: Euribor 3 month rate (cont.)
20. *nr.employed*: Number of employees (cont.)

We have **41.188 observations** and there are **no missing** values (although some of the categorical variables have, it has been created an *unknown* modality for them).

2.2 Motivation

We have chosen this dataset because it is an interesting classification problem related to business. The problem is not an easy one neither a very complex, so we think that with it, we will apply some of the machine learning techniques explained in class and learn how to develop an ML project. Also, the dataset comes from the UCI repository, so it is a well-studied problem and it is useful for academic purposes.

The dataset has enough explanatory variables (20) and the number of observations is not low for this type of problem (more than 40.000), so we think that we will not have any problem with this dataset.

2.3 Objective

As said before, the main goal of our project is to find the best model to predict whether the client subscribes a term deposit (the output variable will be called y). For this objective, we will have to apply some learning techniques to our dataset.

3 Related works and fundamental references

This dataset is available on the *UCI Machine Learning Repository*, so it is a well-studied one and many papers have used it.

For example, [2] compares multilayer perceptron neural network (MLPNN), tree augmented Naïve Bayes (TAN), logistic regression (LR), and Ross Quinlan new decision tree model (C5.0) in order to predict whether a client will subscribe a term deposit, it obtains testing accuracies near to 90%. [3] makes a comparison between J48-graft and LAD decision trees, radial basis function network and support vector machine, the best accuracy is obtained with SVM (87%). [4] uses the Naïve Bayes and the C4.5 decision tree algorithms and gets 85% and 93% of accuracy respectively.

As well, we can mention to papers referenced in the UCI Repository ([7]): [5] performs a semi-automatic feature selection and a comparison between several models (logistic regression, decision trees, neural network and support vector machine), and the classification accuracy achieved is 81%. Additionally [6] uses the CRISP-DM methodology and the best model is obtained using SVM.

Also, the OpenML webpage [1] has a task associated with this dataset and our classification purposes. In it, we can see a comparison between several methods and the results that they get. The best accuracy is near 90% and it is obtained with random forest.

4 Exploratory analysis

The first step to do in any ML project is to do an exploratory analysis on the dataset. In this section, it is explained the preprocess and visualization tasks applied to our project.

4.1 Pre-processing

Pre-processing is one of the most important parts of our work. It is important because it can have a deep impact on it. Let's see how we will deal with our dataset in the pre-processing section.

4.1.1 Basic inspection of the dataset

The first step to do is to load the data and do a basic inspection of it. As said before, the dataset contains 41.188 observations and 21 variables (20 explanatory variables and 1 response variable). Of these 20 explanatory variables, 10 are continuous and 10 categorical.

Firstly, we decided to **remove the *duration* variable** from the dataset because the duration is not known before a call is performed, so, it cannot be used for the predictive model. Now, our dataset has 19 explanatory variables (9 continuous and 10 factors).

Once this variable is deleted, we do a basic inspection of all variables by means of doing a summary of the dataset, creating boxplots and histograms of continuous variables (Appendix A) and barplots of categorical variables (Appendix B). By this inspection, we could understand the behavior of the variables and also look for outliers in the boxplot (we could not find any clear one that should be removed).

4.1.2 Missing values

Secondly, we should deal with the missing values. Our dataset does not have *NA* values, but some variables have values with the meaning of being missings. These variables are:

- Categorical variables. The missing values on the categorical variables were coded into a modality that group all these instances (called *unknow*). There are some variables with a lot of missings (e.g. $default$ has more than 8.000 missing observations), so we cannot remove these observations and it is convenient to keep this modality into our dataset because being *unknown* could be helpful for classification.

- *pdays* variable. The *pdays* is a continuous variable that has a lot of 999, the documentation says that 999 means that the client was not previously contacted, so we decided to set these values to NA. But now, more than 30.000 observations are NA ($> 70\%$), so we cannot impute these values either remove them. We decided to discretize this variable (without 999 values) into an ordered variable, and then add a new modality that means 'not previously contacted'.

4.1.3 Transformation of variables

As we can see in Appendix C, most of the continuous variables do not follow the normal distribution, so in this section we try to modify some existing variables to more friendly distributions for our statistical algorithms.

First, we try to apply **logarithms** to the variables that seem to be exponential (see Appendix C):

- *campaign*. After applying logarithms, there is a little of improvement but not much, the distribution does not seem to be Gaussian. But we thought that *log.campaign* should work better.

- *previous*. There is no improvement applying logs, but we noticed that it only has 7 different values, so we decided to discretize the variable into an ordered factor (called $previous.CAT$) and do not use the log.

Also, in Appendix C, we saw that some variables are very irregular (they have an irregular slope and many maximums). We thought that these variables could be **discretized** into an ordered factor, and in order to make a good analysis, we decided to create two datasets: one with the irregular variables and another with the variables discretized (more information about these datasets in following sections). So, these variables that will be discretized are: $emp.var.rate$, $cons.price.idx$, $cons.conf.idx$, $euribor3m$, $nr.employed$ and $campaign/log.campaign$.

4.1.4 Feature selection

In order to see if there is a relation between the explanatory variables and the output variable (y), we apply some tests of independence.

For the continuous variables, we apply the **Fisher's F test**. All them gives a p-value lower than 0.05 (except *pdays*), so we should keep all these continuous variables in the dataset. About the *pdays*, as said before, we have discretized it, so we will not use this continuous variable.

For the categorical variables (including the newly created ones), we apply the **Chi-squared test**. All variables except *loan* and *housing* get a p-value lower than 0.05, so we keep them. About the *loan* and *housing*, we cannot reject that y and these variables are independents (no relation), so we should remove them from the dataset.

4.1.5 Standardization

Finally, the continuous variables are standardized [1] because we want all variables to have the same mean and standard deviation. If not, some methods will give more importance to some variables than others and this is not what we want.

4.1.6 Feature extraction

It could be also interesting to apply some feature extraction techniques[2] to our dataset. We are going to try two approaches:

1. Apply PCA to the continuous variables and MCA to the categorical ones (selecting the number of significant dimensions for each one). Then, making the concatenation of both and applying again PCA and select the dimensions to use.
2. Apply MCA to the dataset with all continuous variables discretized.

In the following sections, it will be analyzed if these feature extraction methods are helpful for prediction tasks. Also, these methods will be used in the visualization section and are better explained there.

4.1.7 Summary

At this point, the pre-processing of our dataset is done. Let's summarize the different datasets that we have created:

- D1: Dataset containing all continuous variables (without discretization) and the categorical ones.
- D2: D1 but discretizing the continuous variables (except *age* that seems to be more or less gaussian).
- D3: The projections of PCA applied to the concatenation of PCA and MCA.
- D4: The projections of MCA applied to the dataset containing all variables discretized.

These four datasets will be tested in model selection, and the one that performs better will be used in the final model.

4.2 Visualization

In this section, we explore our data trying to make a visualization in 2d of our dataset and trying to understand how the different exploratory variables are related between them. We are going to use **Principal Components Analysis** and **Multiple Correspondence Analysis**.

Our dataset has mixed variables, so we decided to apply two different approaches: the first consists on keeping the continuous and categorical variables and make a visualization on this dataset (applying PCA to cont. and MCA to cat.), and the second consists on discretize all continuous variables and use MCA to make the visualization. We are only going to use the learning set, because the resulting dimensions of each approach will be used in model selection, so we cannot use the test set to compute them. Then, for the test set, we will apply the same transformation/projection that was applied to the train set. The two approaches are:

[1] The standardization is done using only the learning set and then we apply the same transformation to the test set (if not, the test set would have some influence on the training standardization).

[2] These feature extraction techniques only use the learning set to compute the projections and then the same transformation is applied to the test set.

Approach 1 (Creation of D3):

First, we applied **PCA to the numerical variables** of our dataset (it has 8 numerical variables). Looking at the scree plot (Appendix D - Figure 8), we see that with 6 dimensions we explain more than 95% of the variance, so they are enough. Moreover, the projection of the variables in the two first dimensions (Appendix D - Figure 9) shows us some interesting relationships between variables: first, the *emp.var.rate*, *cons.price.idx*, *euribor3m* and *nr.employed* seem to be highly correlated, and *previous* negative correlated with this group, and secondly *cons.conf.idx* and *age* seem to be correlated and be very poor correlated to the variables of the first group. Finally, we do the projection of individuals in the first two dimensions (Appendix D - Figure 10), the observations with target *yes* seems to have higher PC1 than *no*, but the observations are very mixed.

Secondly, we apply **MCA to the categorical variables** (our dataset has 9 variables). Then we select the dimensions to keep by selecting the ones that have more than the mean of all eigenvalues (we keep 20 dimensions, Appendix D - Figure 11). With the projection of the variables in the first dimensions (Appendix D - Figure 12), we found some relations between variables like that $pdays.CAT$ and *poutcome* are highly correlated or *education* and job are correlated too. Finally, we do the projection of the observations in 2D (Appendix D - Figure 13), and we see that the *no* output seems to be more concentrated in a region and the *yes* be more dispersed in the plane.

And finally, we apply **PCA to the concatenation of the projections of PCA and MCA** that we obtained in previous steps. The concatenation has 26 dimensions, but the new Scree plot suggests that just 6 dimensions are significant (Appendix D - Figure 14). Then, we make the projections of individuals in the first two dimensions (Appendix D - Figure 15), and again we see that the *no* seems to be in denser regions (more concentrated) and the *yes* more disperse.

These new 6 dimensions are saved as $D3$ and will be used in model selection.

Approach 2 (Creation of D4):

The second approach consists of, first **discretize all continuous variables** to have a dataset with only categorical variables, and secondly, **apply MCA** to this dataset. Again, we decide the number of dimensions to keep by selecting the ones that have more than the mean of all eigenvalues: we keep 29 dimensions (Appendix D - Figure 16). Then, we look at the projection of the variables in the two first dimensions (Appendix D- Figure 17), we found that $cons.price.idx.CAT$, $cons.conf.idx.CAT$, $euribor3m.CAT$, $nr.employed.CAT$ and $emp.var.rate.CAT$ are highly correlated and also that *poutcome* and $previous.CAT$ are correlated too (this information was also found in approach 1). Finally, we plot the projection of individuals in the first two dimensions (Appendix D - Figure 18), again, we see that *no* observations are in denser regions and *yes* are more dispersed.

These new 29 dimensions are saved as $D4$ and will be used in model selection.

5 Protocol of validation

The studied dataset has 41.188 observations, we think that this number is high enough to divide the data into one set for learning and another one for testing. So, we use the **holdout method**. We decided to choose 1/3 as the holdout ratio, so 66.6% of observations are used to select and create the best model and the remaining 33.3% to test it.

In order to select the observations to belongs to the learning or testing set, we do a **stratified random sampling**. We use random sampling because the data does not follow any sequential structure (e.g. ordered by time), so the best choice is to pick the observations randomly. Also, we decided to use stratified sampling because we want to keep the proportion of yes/no in both sets.

The learning set will be used for model selection and to create the final model. For the model selection step, we need to use a resampling method (we cannot use the same data for train and validate), so, we decided

to do a **10-fold cross-validation on the learning set**, and then we can compare the different models by means of the validation error (mean of all folds).

Once we have a model selected, we refit it using the full learning set, and we use the test set to get an estimation of how good or bad is the final model.

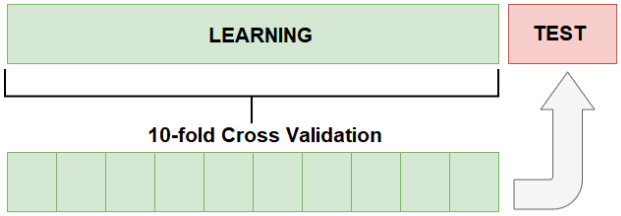

Figure 1: Validation protocol schema.

Performance metric

In order to decide how good or bad is a model, we need to choose the performance metric to be used. In our case, as we have a classification problem and the target variable is very unbalanced, we thought that the best choice was to use the **F1** score because it gives more weight to the *yes* modality than the accuracy.

Datasets

In the previous section (exploratory analysis), we found convenient to compare several modifications of the original dataset ($D1$,$D2$, $D3$ and $D4$). These four datasets will be used in the model selection step and the one that performs better will be chosen for the final model.

ML techniques

Finally, we have to decide which techniques will be used in model selection. They have to deal with a classification problem (with two levels) and four datasets: two mixed dataset ($D1$ and $D2$) and two numerical datasets ($D3$ and $D4$). So, the techniques that are suitable and will be used in this project are:

- Logistic Regression
- Ridge Regression (binomial)
- Lasso regression (binomial)
- Naïve Bayes
- LDA
- MLP
- SVM
- Decision tree
- Random forest

For each dataset, all these techniques are tried and the method that gives higher F1 score (result of the 10-fold cross-validation) will be selected as the final model. Also, the hyper-parameters of each method will be optimized (in the next sections it is explained with more details).

6 Model selection

In this section, it is explained the results of our model selection. Firstly, for each method, we explain the experiment, the set of hyper-parameters that will be tested and the results obtained, together with the best set of parameters for the specific method. And secondly, we do a comparison between these methods, explaining the ones that perform better, and the selected method (and hyper-parameters) that will be used for the final model. As explained before, we use a 10-fold cross-validation on the learning set and we will select the parameters and method that get the highest F1 value.

6.1 Methods

6.1.1 Logistic Regression

The logistic regression is the first method that we decided to test. It deals with binomial outputs and mixed data, so it is suitable for our problem. As we have an unbalanced dataset, we had to set the weights of each observation in order to give the same importance to both classes (if we do not set the weights, the predictions will be unbalanced to the *no* case, and thus, the F1 score will be very low). For this method and all the following ones, we will set the weights of each observation in order to solve this unbalanced issue. The results obtained are shown in the following table:

	D1	**D2**	**D3**	**D4**
F1 score	0.7293	0.7305	0.7174	0.7249

With this table we see that $D1$ and $D2$ performs better than $D3$ and $D4$, so the PCA and MCA transformations do not help to predict. Also, $D2$ gets a higher F1 than $D1$, so the discretization of the continuous variables that do not seem gaussian is useful.

6.1.2 Lasso and Ridge Regression (binomial)

Secondly, we try the L_1 and L_2 regularizations of the logistic regression. Again, we set the weights of each observation to give the same importance to both classes. These methods have one hyper-parameter: λ, so, we need to find the best value for it. For each method and dataset, we try 50 different λ from 10^{-4} to 10^2 distributed exponentially.

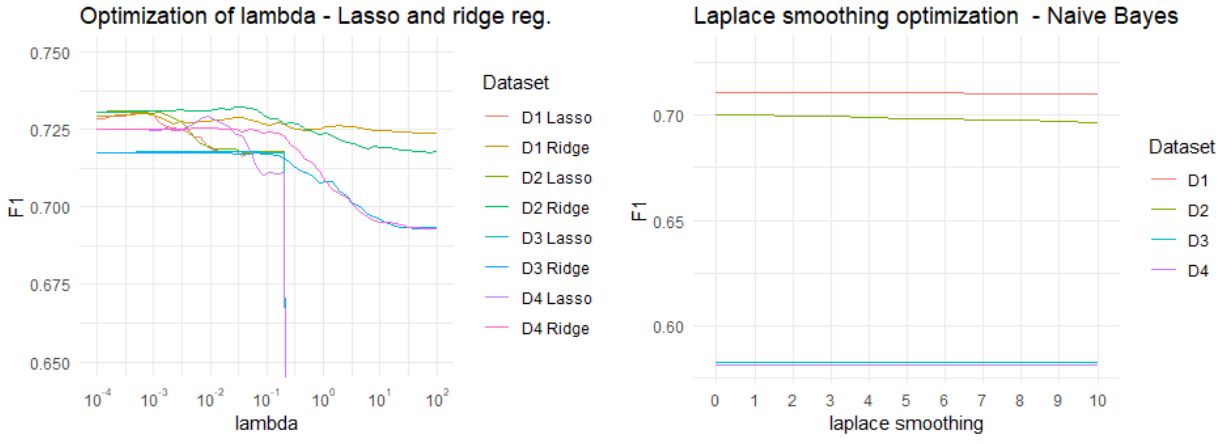

Figure 2: (Left) Optimization of λ for Lasso and Ridge regression, (Right) Optimization of the Laplace smoothing hyper-parameter for Naïve Bayes

Figure 2 (Left) shows the results of our experiments. First of all, we see that Lasso and Ridge get similar results when λ is low, but when increasing this value Lasso starts decreasing until a point where it goes to 0 (always predict the same class). Instead, the Ridge increases the value a little until 10^{-1} where it starts to decrease.

All four datasets seem to follow the same behavior, but again, $D3$ gets lower F1 than the others (blue lines) and also $D4$ is below $D1$ and $D2$. The $D2$ with Ridge regression and lambda near to 10^{-1} seems to be the clear winner. The following table shows the best results for each method and dataset:

	D1	**D2**	**D3**	**D4**
F1 (Lasso)	0.7304 ($\lambda = 0.0005$)	0.7310 ($\lambda = 0.0002$)	0.7178 ($\lambda = 0.0012$)	0.7294 ($\lambda = 0.0091$)
F1 (Ridge)	0.7302 ($\lambda = 0.0005$)	0.7323 ($\lambda = 0.0373$)	0.7174 ($\lambda = 0.0159$)	0.7255 ($\lambda = 0.0091$)

So, the regularization improves a little bit the logistic regression. The best performance ($F1$ of 0.7323) is obtained with Ridge regression and lambda $\lambda = 0.0373$.

6.1.3 LDA

The third method that is tested is LDA. This method does not have hyper-parameters to optimize, but we set the prior of both classes to be equal (in order to give the same importance to *yes* and *no*). The results obtained are:

	D1	**D2**	**D3**	**D4**
F1 score	0.7125	0.7093	0.6630	0.6532

Again $D1$ and $D2$ get higher performance than $D3$ and $D4$. But, comparing them with the previous methods, this method obtains lower F1 scores.

6.1.4 Naïve Bayes

The next method to test is the Naïve Bayes classifier. It has one hyper-parameter that could be optimized: the Laplace smoothing. So, we test 10 different values, from 0 to 10, being 0 when the correction is not used. Figure 2 (Right) shows the results of our experiments. It seems that the Laplace smoothing does not have a lot of influence on the F1 score: $D1$ increase a few until 5 and then decrease, $D2$ seems to decrease when increasing the hyper-parameter and $D3$ and $D4$ seems to be horizontal lines. The results obtained with this method are the following (l is the Laplace smoothing parameter):

	D1	**D2**	**D3**	**D4**
F1 score	0.7109 ($l = 5$)	0.7003 ($l = 1$)	0.5824 ($l = 0$)	0.5809 ($l = 0$)

$D1$ is the dataset that gets better results, followed by $D2$. $D3$ and $D4$ gets poorly F1 scores using this method. This method obtains similar results than LDA, so the F1 scores are below the logistic regression (regularized).

6.1.5 SVM

The next classifier that we thought it should be tried is the Support Vector Machine. For this method, we have to choose the C hyper-parameter and the kernel function. For C we are going to test six values exponentially distributed from 10^{-4} to 10. About the kernel function, we use the linear kernel, the polynomial kernel of degree 2 and 3 and the RBF. Also, the RBF has the γ hyper-parameter, so we do a grid search of C and γ (testing $\gamma = 2^{-3}$, 2^{-2} , 2^{-1} , 1 , 2^{1} and 2^{2}).

Figure 3 and Appendix E shows the graphical results of our experiments. For this method, we experienced a high learning time: for running all experiments we spend more than 48 hours. In Figure 3, we see that the linear and polynomial kernels have a similar behavior: with low C the $F1$ score is very bad, then increasing this value the $F1$ starts to increase until a point where it remains more or less stable. As larger is C the

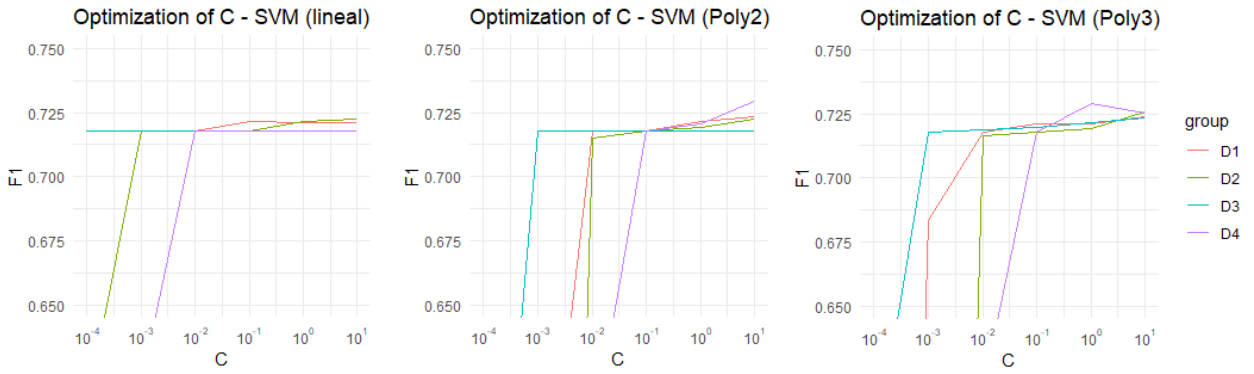

Figure 3: Optimization of C for linear (left), polynomial 2 (middle) and polynomial 3 (right)

training time is higher too, so we should take the lowest C that has high $F1$ score. We have also tried $C = 10^2$ for the polynomial cases, but it takes much more time to learn the model and there was no improvement on $F1$. In Appendix E, it is shown the grid search performed for the RBF kernel and the results obtained. With low or high value of C, the $F1$ is worse than with C near to 1. Also for the gamma, the best results are obtained when γ is small (2^{-2}).

The following table shows the best results for each dataset and kernel:

	D1	**D2**	**D3**	**D4**
F1 (Linear)	0.7215 ($C = 10^{-1}$)	0.7222 ($C = 10^{1}$)	0.7176 ($C = 10^{-3}$)	0.7177 ($C = 10^{-1}$)
F1 (Poly2)	0.7236 ($C = 10^{1}$)	0.7225 ($C = 10^{1}$)	0.7177 ($C = 10^{-3}$)	0.7295 ($C = 10^{1}$)
F1 (Poly3)	0.7236 ($C = 10^{1}$)	0.7255 ($C = 10^{1}$)	0.7231 ($C = 10^{1}$)	0.7289 ($C = 10^{0}$)
F1 (RBF)	0.7295 ($C = 10^{-1}$,$\gamma = 2^{-3}$)	0.7331 ($C = 10^{-1}$,$\gamma = 2^{-2}$)	0.7259 ($C = 1$,$\gamma = 2^{-2}$)	0.7303 ($C = 10^{-1}$,$\gamma = 2^{-2}$)

In this table, we can see that the four datasets obtain similar $F1$ scores (the difference is higher when using *linear* or *poly*2). For the kernels, the *RBF* is the one that better performs, but the others get $F1$ very near to this kernel. The result with $D2$ and *RBF* is the highest one amount all the methods seen until now, so the *SVM* works well for our problem (although it is the one that needs more time to train the model).

6.1.6 MLP

Now, we are going to test how our problem performs with a Multilayer Perceptron Neural Network. For this type of networks, we decide to set a large number of hidden neurons in one hidden layer and we explore different regularization values (*decay*). So, we set the MLP to have one hidden layer with 30 neurons (we thought it was enough, and if we increase this number, the training time will increase a lot). For the *decay* hyper-parameter we try the following values: 0 (no regularization), 10^{-3}, 10^{-2} ,... until 10^{5}.

Figure 4 (Left) shows the results of our experiments. Again, when running these experiments we experienced a high learning time, we need more than 6 hours to run all them. With this plot, we see that all datasets have a similar behavior: with low and high regularizations they obtain low $F1$ and the maximum is near to $decay = 1$. We also see that $D1$ and $D2$ works worse than the others datasets with low regularization, but they reach a similar maximum in all cases. The following table shows the best results, where we see that $D1$ and $D2$ get the maximum $F1$ score, but $D3$ and $D4$ are very near:

	D1	**D2**	**D3**	**D4**
F1 score	0.7309 ($decay = 10$)	0.7309 ($decay = 10$)	0.7284 ($decay = 1$)	0.7305 ($decay = 10$)

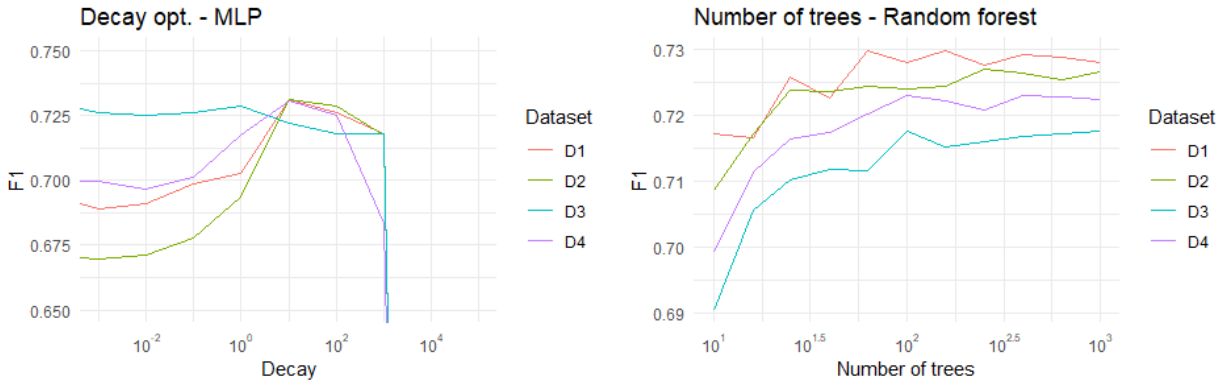

Figure 4: (Left) Selection of the decay hyper-parameter in MLP, (Right) Selection of the number of trees in Random forest

6.1.7 Decision tree

Another classification method are the decision trees. The first approach is to use a simple tree for the classification and see how it performs. So, we generate a decision tree for each dataset, and the 10-fold cross-validation $F1$ scores obtained are the following:

	D1	**D2**	**D3**	**D4**
F1 score	0.7186	0.7131	0.7172	0.7202

The results are quite good for using just one tree, but they are below the $F1$ obtained with SVM. $D4$ is the dataset that gets higher $F1$, but the other three are very near.

6.1.8 Random forest

Finally, the last method is an improvement of the previous one: the random forest. It has two hyper-parameters: the number of trees and the number of variables randomly sampled as candidates at each split. For the second one, we set it as the default value ($\sqrt{n}$ variables) that use to give good results. For the number of trees, we test 11 values exponentially distributed from 10 to 1000, and we will take the minimum number of trees that adding more trees to the forest do not add an improvement to the $F1$ score.

Figure 4 (Right) shows the results of our experiment. We can see that when the number of trees is small, then adding more trees has a high impact on the F1 score. But when it reaches a point, there is no big improvement, all datasets seem to stabilize when reaching 100 trees. For this method, $D1$ performs better than the other datasets.

The best results for each dataset can be seen in the following table ($nTree$ means the number of trees used):

	D1	**D2**	**D3**	**D4**
F1 score	0.7297 ($nTree = 63$)	0.7268 ($nTree = 251$)	0.7175 ($nTree = 100$)	0.7229 ($nTree = 100$)

This method obtains F1 scores a little higher than the ones obtained with decision trees, and lower than SVM (but near), so it seems that the random forest performs well for our problem.

6.2 Comparison

At this point, once we have explained all the experiments, we have to make a comparison between them and select the one that performs better.

The following table shows a summary of the best results:

	F1	**Dataset**	
Logistic regression	0.7305	D2	
Lasso	0.7310	D2	$\lambda = 0.0002$
Ridge	0.7323	D2	$\lambda = 0.0373$
LDA	0.7125	D1	
Naive Bayes	0.7109	D1	
SVM	0.7331	D2	kernel=RBF, $C = 10^{-1}$,$\gamma = 2^{-2}$
MLP	0.7309	D1/D2	$neurons = 30$, $decay = 10$
Decision tree	0.7202	D4	
Random forest	0.7297	D1	$nTree = 63$

LDA and *Naïve Bayes* are the two methods that obtains lowest results ($F1 \approx 0.71$), then the *Decision tree* ($F1 \approx 0.72$) and all others methods get an $F1$ score near to 0.73. In our experiments, mostly $D1$ and $D2$ performs better than $D3$ and $D4$, so the use of the projection of the PCA and MCA do not help on the prediction in our problem. Between $D1$ and $D2$, $D2$ seems to be a step above too, that means that our discretization of the irregular variables was useful. Also, we want to mention that SVM and MLP take a lot of time to train the models, instead all others seem to be much faster, but the results of these two methods are two of the best ones.

Summarizing, as we want to select the best method to predict if a client will subscribe or not, we decided to use the SVM model with the RBF kernel, $C = 0.1$, $\gamma = 2^{-2}$ and using the $D2$ dataset.

7 Final model

Finally, once we have selected the best method to use, we have to refit the model with all the training set ($D2$) and then do the prediction of the test set. With these predictions, we can estimate the generalization error of the model because this data was not used in the training stage.

So, we fit a SVM with the RBF kernel, $C = 0.1$ and $\gamma = 2^{-2}$ using the $D2$ (learning part). With the new model, we predict the $D2$ test set. This test set has 13.730 observations, so we have a large number of observations to compute the test error.

The confusion matrix of the test set is:

		Real labels	
		no	yes
predictions	no	10.027	540
	yes	2.156	1.007

And some interesting performance statistics are:

Accuracy	80.36%
Recall^{yes}	65.09%
Recall^{no}	82.30%
Precision^{yes}	31.83%
Precision^{no}	94.88%

The final model gets an accuracy not very high (80%) knowing that 90% of the observations are of type *no* (the model that predicts always *no* would have higher accuracy), but this is the price that we pay to give more importance to the *yes* class (minority class). For our problem, we want to have high recall on the *yes* case because we do not want to predict clients that will subscribe as a *no* (the bank will lose the client), and this recall on our model is not bad.

So, we think that this final model could be useful for the bank company because it would filter a lot of real *no* subscription users and it would keep most of the *yes*.

8 Scientific and personal conclusions

In this project, we have performed a full machine learning process on a real dataset related to bank marketing (the main goal of the problem is to classify if a client will subscribe a deposit or not). Doing it, we have learned how to develop a project of this type, starting by doing a pre-process of the data, then making the visualization of it, defining the protocol of validation, the model selection and finally creating the final model and getting an estimation of its generalization error.

At the beginning of the project, in order to do a correct analysis, we should understand how is our dataset. So, in the pre-processing stage, we did a first inspection of the dataset, then we deal with the missing values, we apply some transformation to the variables (logarithms and discretizations), we apply some feature selection techniques in order to remove unnecessary variables, we standardize all continuous variables and we apply some feature extraction techniques. After doing this pre-processing we have generated four different datasets: $D1$, $D2$, $D3$ and $D4$ (more details about these datasets in the Pre-processing Section)

Secondly, we did a visualization of our dataset in order to discover hidden information in it (and trying to reduce the dimensionality). As we have mixed data, we apply two different approaches: the first consists on applying PCA to the concatenation of PCA to continuous variables and MCA to categorical, and the second approach consists on applying MCA to the dataset with all variables discretized. With them, we discovered some relations between variables like that *cons.price.idx*, *euribor3m*, *nr.employed* and *emp.var.rate* are highly correlated between them.

Then, as a protocol of validation, we decided to use the holdout method (split the data into learning and test sets) because the test data cannot be used to create the classification models. Also, for the validation error, we decided to apply a 10-fold cross-validation on the learning set. As our data is unbalanced, we decided to use the F1 as a performance metric because we want to give more importance to the *yes* modality.

For the model selection part, we have tested several ML classification techniques (like logistic regression or SVM) and we have optimized most of their hyper-parameters. The one that obtains better results is the SVM using the RBF kernel, $C = 0.1$ and $\gamma = 2^{-2}$ applied to $D2$. About the four preprocessed datasets, we found that $D3$ and $D4$ do not improve the prediction tasks and that discretizing all continuous variables that do not seem Gaussian ($D2$) gets a little bit of improvement.

Once the model selection is done, we create the final model refitting it with the learning set and testing the predictions with the test sets. The results show us that the accuracy is not very high (80%) knowing that 90% of the observations are of type *no*. But the recall in the *yes* label is not bad, for our problem it is interesting that this recall is as higher as possible because we do not want to predict clients that will subscribe as a *no* (the bank will lose the client). So, we think that this final model could be useful for the bank company.

Overall, we found this project interesting to learn how to develop an ML project in a real dataset, and to understand and use in practice some of the methods explained in the lectures.

9 Possible extensions and known limitations

As future work, it remains to study other ML techniques not used in the model selection part of this project (like ensambles), or the use, for example, of other kernels to see if the predictions can be improved.

Moreover, the model could be also improved if we increase the number of observations in the dataset, especially for the case of *yes* subscriptions, or if more features related to the clients (columns) are include in the dataset.

References

[1] OpenML Supervised Classification on bank-marketing. `https://www.openml.org/t/9899`. [Online; accessed 03-04-2019].

[2] Hany A Elsalamony. Bank Direct Marketing Analysis of Data Mining Techniques. Technical Report 7, 2014.

[3] K. Wisaeng. A Comparison of Different Classification Techniques for Bank Direct Marketing. *International Journal of Soft Computing and Engineering (IJSCE)*, 2013.

[4] Masud Karim and Rashedur M Rahman. Decision Tree and Naïve Bayes Algorithm for Classification and Generation of Actionable Knowledge for Direct Marketing. *Journal of Software Engineering and Applications*, (6):196–206, 2013.

[5] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31. June 2014.

[6] S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology. In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimaraes, Portugal, October, 2011. EUROSIS. [bank.zip].

[7] UCI Machine Learning Repository. Bank Marketing Data Set. `https://archive.ics.uci.edu/ml/datasets/bank+marketing`. [Online; accessed 03-04-2019].