SAPIENZA
UNIVERSITÀ DI ROMA

Predicting customers complaints in an energy company

Faculty of Information Engineering, Computer Science and Statistics

Master's degree in Data Science

Candidate

Laura Giuliano

ID number 1602745

Thesis Advisor

prof. Anagnostopoulos Aristidis

External Advisor

Antonio Olivieri

Academic Year 2019/2020

Thesis defended on 23 January 2020
in front of a Board of Examiners composed by:

prof. Anagnostopoulos Aristidis (chairman)
prof. Chatzigiannakis Ioannis
prof. Cuomo Francesca
prof. Brunero Liseo
prof. Maggino Filomena
prof. Palagi Laura
prof. Scardapane Simone

Predicting customers complaints in an energy company
Master's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: June 16, 2020

Author's email: laurettagiuliano@hotmail.it

Abstract

Attracting a new customer to a different business company has today become more expensive than retaining an existing one. Companies must therefore identify those customers who are about to churn to another business and improve their customer satisfaction with Targeted Marketing. One promising way to identify such customers is predicting customer complaints and dissatisfaction by analysing large volumes of available data.

The goal of this thesis is the development of accurate and reliable models for predicting customer complaints. By predicting such complaints, business companies can reduce the cost of dealing with a complaint, improve the customer satisfaction, and therefore reduce customer churn.

We tackle the above problem using Machine Learning tools and reproducing the three main phases of a data science life cycle: Data, Discovery, and Deployment. The Data phase refers to the collection, preparation, and exploration of data. The Discovery phase concerns building and refining multiple models with the goal of selecting the best one. The Deployment phase entails applying the selected model to the new data and evaluate the results. Before performing the above three steps, we run an ex-ante analysis that aims to understand the business ecosystem, their specific problem, and to define our detailed objectives. Our final results indicates that the best model is the *random forest*, which achieves on average 16% of precision out of 5000 customers per month. These results were compared to the model previously used by the analysed company (called *Logistic*) that only achieved 5% of precision per month. We leave the challenging problem of further improving the precision of the predictive model (*e.g.*, 20% precision) as an interesting future work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Company

I have been working as a consultant for NTT Data, one of the largest IT services company worldwide. I was assigned to a project in one of the most important gas and energy company in Italy. I worked mainly on two different business problems: churn and complaint prediction. This thesis will focus on complaint prediction.

1.2 Method of work

There are two main methods for project management: V-cycle and Scrum.

V-cycle: is based on an end-to-end development, so it is a total delegation from the manufacturing of the application to the final delivery. A development cycle takes several months.

It is a simple method used mostly for small projects. The drawback of this method consists to plan everything in one shot, hence risks are detected at the end of the project because the test phase of the product can be carry out only at the end of the development.

Below in Figure 1.1 a schema of the development process:

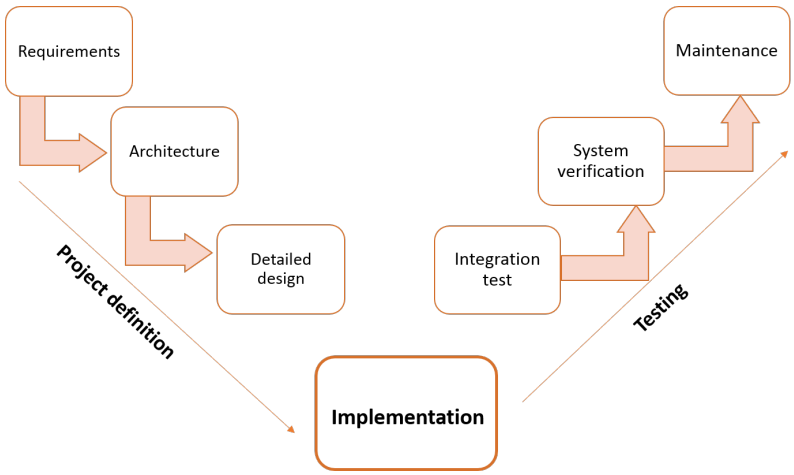

Figure 1.1. V-cycle

Scrum methods are iterative, this means that analysts develop small pieces of application that bring value to the work, which has to be validated by the company before developing it. Consequently, the project is divided into iterations with the client instead of everything in one shot as the V-cycle, knowing that unforeseen events will happen along the way.

The iterative process has the advantage to have a product that is very close to the client's needs, taking into account changes, based on feedback, in the latter. It also facilitates communication between all project stakeholders and empowers the entire team.

On the other hand, an Agile/Scrum approach can be challenging because it may also require some level of organisational transformation and collaborative work.

The project management method used in my work was the Scrum method applied to a team work. Each iteration is called a "sprint" and it last two weeks.

The description of the typical steps of the "sprint" is the following:

- At the start of each sprint the product owner decides the tasks to be accomplished with a sequence order of priority.
- Each task is weighted democratically by the team.
- Before each sprint the team gives a velocity that is the total weight of task that can be done.
- The product owner assigns a selection of task to be done during the two weeks.
- Each day the team has a morning progress meeting.
- At the end of the two weeks, results of what has been done during the sprint are presented.
- After the review there is a retrospective with the scrum master, concerning problems and improvements of the sprint.

Below Figure 1.2 a representation of the steps:

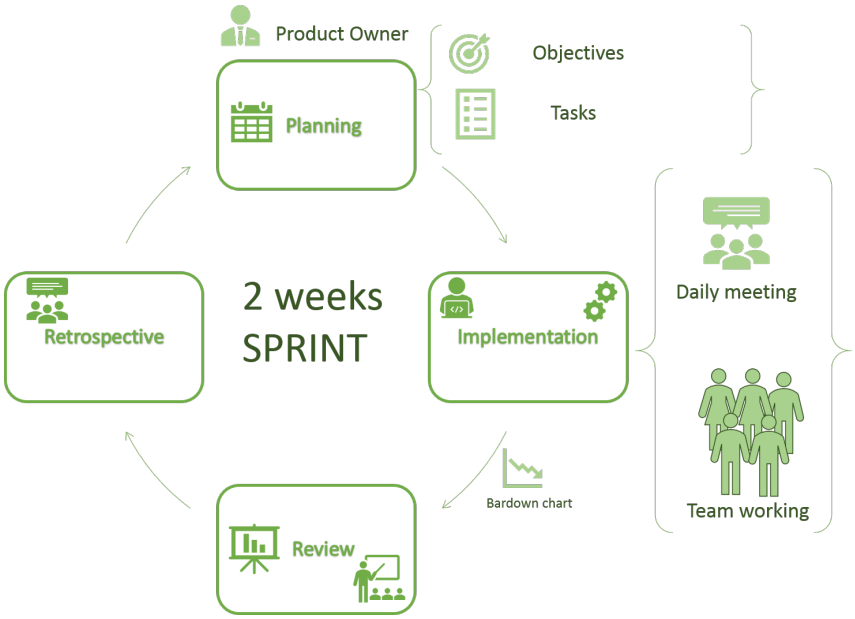

Figure 1.2. Agile working

1.3 Development framework: Apache Spark

In the project, we worked and had to deal with big data; there were about 12 millions of observations per month and 300 fields. The company client uses the Hadoop software combined with Apache Spark to analyse their data sets.

These tools were chosen to maintain high speed in processing large datasets in terms of waiting time between queries and waiting time to run the program.

Apache Spark is an in-memory distributed computing framework that supports ETL, analytics, machine learning, and also the processing of graphs on large volumes of data with different formats in batches or pseudo-real time.

When you launch a process on the Spark framework, you go through the Driver which is the master, it self communicates with the cluster manager, the latter one manages the resources of the workers (the workers execute the actual processing).

We can develop Spark treatments with several different languages, the most used is Scala (the framework itself being developed in Scala), Python, Java, SQL and R language. In particular we used the MLlib tools to code machine learning in pySpark.

The success of the Spark framework in the implementation of MapReduce on Hadoop is mainly due to less expensive and quicker Shuffle steps. In fact, MapReduce performs several reads/writes using the disk while Spark limits these operations (and the intermediate data) in memory (RAM).

1.4 Business challenge

"*Even if you have a good business, customers complaint are unavoidable*"

As usual in data science the analytic process always starts with a business challenge with precise and measurable analysis goals. As already anticipated, our challenge is to predict customer complaints because these are time consuming, frustrating, and expensive processes for a company.

Furthermore, by ignoring or dismissing complaints, a company will drop off customer satisfaction by not valuing their opinions.

Moreover, complaints are the first advice of churn, that is another huge problem for a company.

When we talk about customer's complaint, we refer to a problem that a customer had, that has been formally reported to the company by email, telephone, or letter.

1.4.1 Goal

The goal of this study is to improve a base model (*Logistic*), already used by the company, that manages to achieve an average monthly precision between 4%-5% over 5000 customers, and to reach a precision of 20% in order to increase the ROI and reduce the total cost of dealing with complaints. We want to achieve this by minimising the number of caring call to not let people complain, while at the same time maximising the number of people stopped before complain by reaching only customers that will actually complain.

1.4.2 Dataset

As we previously mentioned, observations were taken from one of the biggest Italian energy company.

We have five months of historical data of customers that formally complained related to:

- Demographic information: basic information of customers. For example: sex, age, region,city of utility service.
- Behaviour: how the service is used. For example: usage and type of the commodity,offer type, billing
- Interactions between customers and company: contacts outbound and inbound by different media (telephone, web,chat,documents,app)

The dataset used to train the model is composed by about 30 million observations (row/services) during three months (January, February, March 2019) and 300 columns/fields. We reduced the dataset with the goal of reducing the processing time without sacrificing precision. The other three months, *i.e., April, May, and June* were used as validation.

1.4.3 Target

Our target variable is a binary variable that tells us if a customer will soon complain or not. We refer to *Class 0* to customers who do not complain and to *Class 1* to customers who complain.

In Figure 1.3, we see the evolution of the customer base compared to the number of customers that complain over six months.

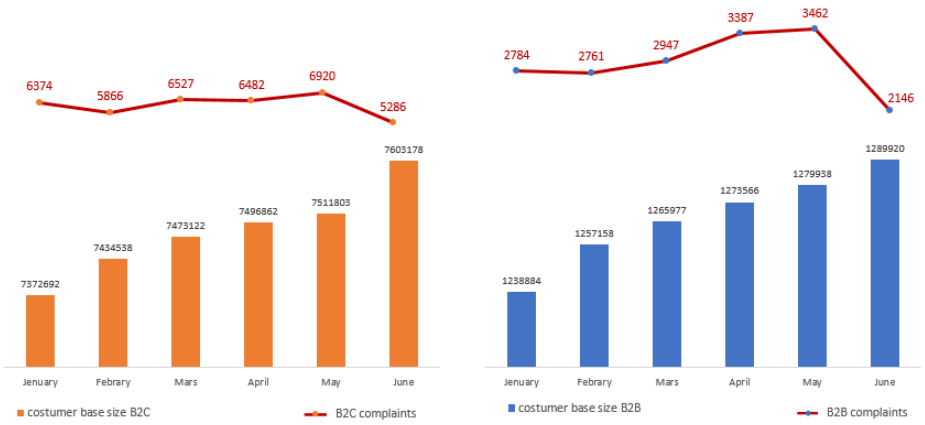

Figure 1.3. Target construction

We see a decrease of complaints in June because the target data about that month was not completely available.

The company needs a reaction time to prevent customers' complaints, so we need a constraint on the time window before the critical event occurs. In our case this reaction time is 7 days. However, we cannot see the data in this time window because, in reality, we would need to consider the time for transmitting the list of customers likely to complain to the operational unit and the time to contact the client.

By not setting this constraint, we might have better performance but this would be useless for the company because it will not have the time to contact the customer before he/she complains.

Therefore, we set the target variable depending on the reaction time of the company: the target variable will be in Class 1 if the customer is likely to complain in the next 30 days with respect to a reference date Dta_{t0}.

Figure 1.4 summarises all the time frames explained above.

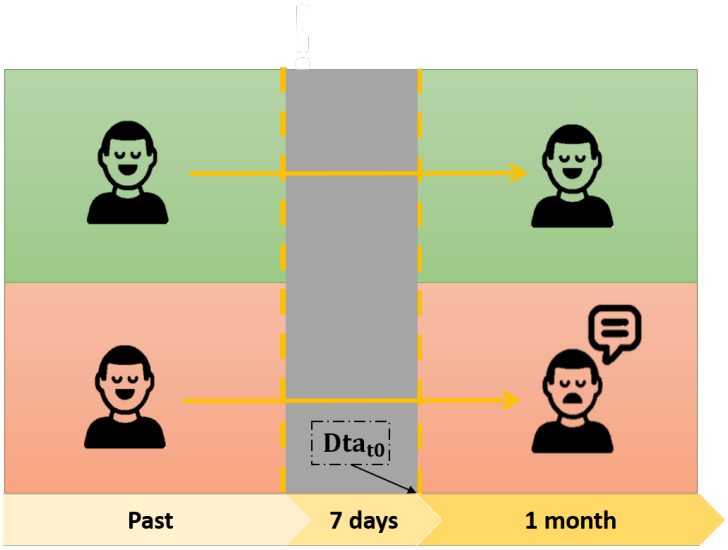

Figure 1.4. Target construction

Chapter 2

Data exploration

In order to have good model performance the most important thing is to check the quality of data; for that reason all data points are presented using the same logic, and the overall data set is free of inconsistencies.

2.1 Descriptive statistical analysis

In order to find out inconsistencies in the dataset, we decided to analyse minimum, maximum, standard deviation, mean, kurtosis and Skewnees of the numerical variables. This is a way to inspect if the frequency distribution of a numerical variable has more or less accentuated weight on the tails. We measured the variability of data by checking the range and the standard deviation. By doing that we found out which variables have outliers that lead also to an important skewness. The solution to this problem is explained in section 2.2.1 .

Thanks to this analysis, we also noticed that there were some constant variables, therefore having a standard deviation equal to zero; this was produced by some errors in the feature engineering step that were than solved (section 3.1).

2.2 Cleaning Data

2.2.1 Outliers

We noticed that for some variables there were some extreme values (really far from the mean). The presence of outliers can alter the predictions of some models like the logistic regression. We consider an observation as an outlier if:

$$x_i \in x_{outlier} \begin{cases} \text{if } x_i < Q_{0.25} - 1.5 \cdot \text{IQR} \\ \text{if } x_i > Q_{0.75} + 1.5 \cdot \text{IQR} \end{cases}$$

where $IQR = Q_{0.75} - Q_{0.25}$

We decided to impute them with this rule:

if $Q_{0.75} \neq 0$:

$$x_i = \begin{cases} Q_{0.25} - 1.5 \cdot \text{IQR} & \text{if } x_i < Q_{0.25} - 1.5 \cdot \text{IQR} \\ Q_{0.75} + 1.5 \cdot \text{IQR} & \text{if } x_i > Q_{0.75} + 1.5 \cdot \text{IQR} \end{cases}$$

(2.1)

if $Q_{0.75} = 0$:

$$x_i = \begin{cases} Q_{0.25}[x > 0] - 1.5 \cdot \text{IQR}[x > 0] & \text{if } x_i < Q_{0.25}[x > 0] - 1.5 \cdot \text{IQR}[x > 0] \\ Q_{0.75}[x > 0] + 1.5 \cdot \text{IQR}[x > 0] & \text{if } x_i > Q_{0.75}[x > 0] + 1.5 \cdot \text{IQR}[x > 0] \end{cases}$$

(2.2)

where $Q_{0.25}[x > 0]$ means that the quantile distribution is calculated without considering the $x_I = 0$

2.2.2 Missing values

It is important to deal with missing data because such observations will be omitted from the analysis, thus biasing the result.

Some variables had more then 90 % of null values. This is due to the fact that these variables were not properly built in the feature engineering so we removed them from the model or fixed them.

The other missing values, only a few, have been replaced with 0 for numerical variables, and with a new modality "***" for the categorical variables. We decided to apply another approach to the new variable $val_{popolazione}$, this will be explained in subsection 3.4

This was right to do because in most of the cases the null value means that the consumer does not have that modality (not applicable). For example, type of electric deal: if someone does not have an electric contract but has just a gas one, he cannot have any electrical deal. This value therefore has a meaning and can form a class apart.

In the case of a random forest model we were not forced to do that , missing values will be treated as another category and will probably be scored automatically according to the splitting rules. Instead, the logistic regression model does not accept missing values.

Chapter 3

Data preparation

3.1 Feature engineering

We perform Feature engineering through a python file that enables to construct the customer base with all the indicators we need.

For example all the variables about consumption and payment bills average, minimum, and maximum are computed over 1,2,3,6,9, months before the reference date, in order to deal with changes over time.

Further, more variables about contacts and actions, in different channels and for different reasons, that the customer has made (inbound) or the company has made (outbound) are computed over 1,2,3,6,9, montsh before the reference date.

All these indicatosr will be useful to represent various behaviours of customers about commitment level with the service.

3.2 Undersampling

Fortunately but unfortunately we had a huge difference in the number of observations between *Class 0* and *Class 1*, so our dataset was unbalanced. This extreme imbalance in the data has to be taken in account in fitting the model. This is a problem for the model because it can learn how to predict people who does not complain but not people that will complain, leading to a high precision for the *Class 0* but really low for the opposite *Class 1*.

In addiction we had to deal with big data (about 11.000.000 of observations per month) so under-sampling is also useful to reduce computational costs and to obtain better or the same performance.

We used the technique to provide a more balanced distribution and build our model on that. In our case since *Class 1* is rare, all events are selected and only some non event cases are selected.

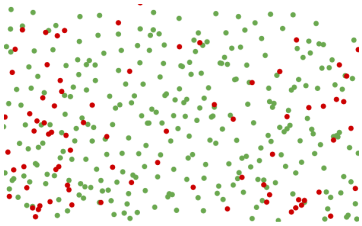

Figure 3.1. Before undersampling

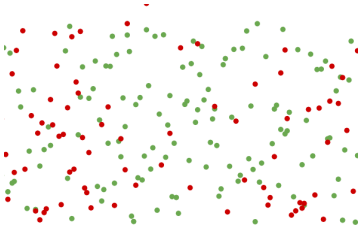

Figure 3.2. After undersampling

We focus on 2 under-sampling techniques:

- **Random**: we took randomly 0.2% from the *Class 0* and 100% from *Class 1*. By doing this we reduce the unbalancing of the dataset.
- **Stratified under-sampling**: we decided to construct a k-means cluster (A.2) of 8 groups over *Class 0* and than take randomly a fixed percentage (10%) of customers from each cluster. In this way we reduced the size of *Class 0* by not taking into consideration similar customers and keeping same proportion.

3.3 Features selection

Due to the fact that we had lot of information (311 variables and about 11 million observation per month) we preferred to reduce the number of input of the model and choose the one that gives us useful information about the target, all those decision were approved by a business translator.

3.3.1 Random forest features importance

To make a stronger decision we decided to run a basic random forest model with max depth=2 then we decided to keep the variable that contribute till 95% of the cumulative variable importance. The technique is shown in the Figure 3.3 below :

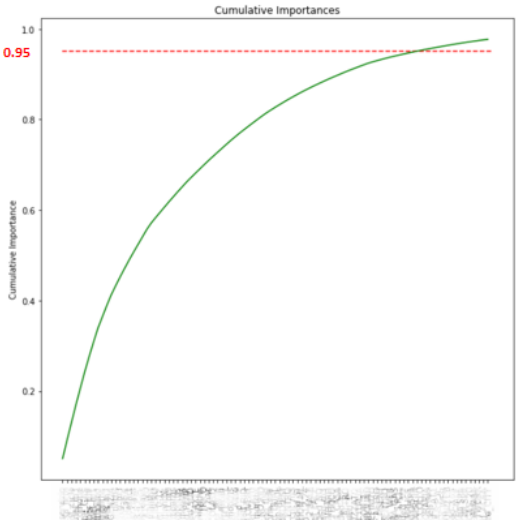

Figure 3.3. Cumulative variable Importance of a Random Forest model

The variables above the red line were cut from the final train. The categorical variable were cut just if any modality of the variable was present in the top 50 features.

We will see in subsection4.1.3 how the model define a score for each variable.

3.3.2 Correlation analysis

The correlation coefficient gives us information about the existence of a linear relation (in the form of a straight line) between the two quantities considered (X, Y)

The quality of the correlation can be measured by a correlation coefficient r.

$$r = \frac{\sum(X - \bar{X})(Y - \bar{Y})}{\sqrt{\sum(X - \bar{X})^2} \times \sqrt{\sum(Y - \bar{Y})^2}} = \frac{cov(X, Y)}{\sigma_x \sigma_y}$$

$-1 \leq r \leq 1$ If r is closer to 1 means that variables are positively correlated, instead if r is closer to -1 means negatively correlated. A coefficient of zero correlation does not mean the absence of any relationship between the two quantities, but there may be a non-linear relationship between them.

Analysing the correlation between variable allow us to reduce the curse of dimensionality.

By analysing the correlation between groups of variables we manage to cut variables where $|r| > 0.75$ and keep just one to train models.

In Figure 3.4 below we can see the correlation matrix of a group of variables:

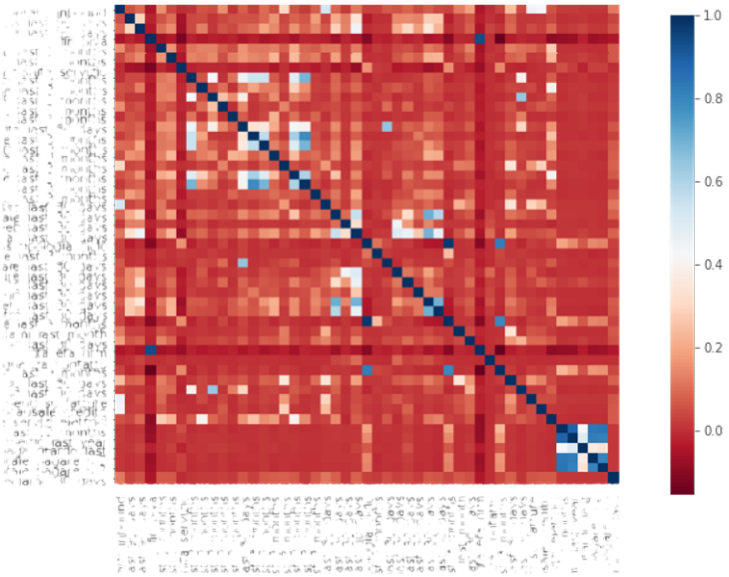

Figure 3.4. Example of a Correlation matrix

We had may correlated variables since many variables were historically computed. For example the number of outbound contacts were computed 3,6,9,12 months before the t_0.

Correlated variables in a logistic model can lead to misleading results, instead boosted trees algorithms are immune to those effects. But it is always useful to

reduce dimension, without loosing information, to reduce the computation time of the algorithm.

3.4 Variable conversion

In order to synthesise the information provided by a variable, make its interpretation easier and in some cases improve computational time, it is possible to convert categorical variable to continuous ones or viceversa (discretization).

For example, a variable for population has been created with a mapping of the variable "comune". The categorical variable corresponding to the house location of a customer has been processed in order to become a numerical variable since there were too many modalities. We found in the Istituto nazionale di statistica (ISTAT) website the number of people living in a specific area and mapped it. So we finally created the field of the number of people in a specific area that is more interesting than the name of the area , since we have the information if it's a big or small city.

We encountered the problem of missing data that we discussed in section 2.2.2 , in this case to deal with this issue we decided to impute missing values with the mean of the ṕrovincia; a bigger area than ćomune.

3.4.1 WOE & IV

Weight of Evidence (WOE) and Information Value (IV) were used to reduce the number of modalities of categorical variables that have a lot of modalities and, in particular to describe the relationship between a predictive variable and a binary target variable. We can formulate it as follows:

$$WOE_i = log\left(\frac{\text{Distribution of } Class\ 1_i}{\text{Distribution of } Class 0_i}\right) = log\left(\frac{\%_i \text{ of } class\ 1_i}{\%_i \text{ of } Class\ 0_i}\right) = log(\frac{f(X_i|Y=1)}{f(X_i|Y=0)}$$

Where the distribution of *Class* 1/0 is the conditional probability density function or a discrete probability distribution, therefore is just the relative frequency of *Class* 1 over the relative frequency of *Class* 0 for each modality of the variable.

In case we want to compute WOE for a continuous variable we just need to create buckets and compute the tables (3.2 & 3.1).

Information Value measures the strength of the relationship described by WOE:

$$IV = \sum_{i=1}^{d} (\%_i \text{ of } class\ 0 - \%_i \text{ of } \ class\ 1) * WOE_i$$

In the table below we can see the steps to compute both of these indicators:

In the tables below we can see the steps needed to compute IV and WOE, where N_i is the number of observations in $bucket_i$, $E_i(y=1)$ is the number of observations in $bucket_i$ where the target belongs to *class* 0, $\overline{E}_i(y=0)$ is the number of observations in $bucket_i$ where the target belongs to *class* 0 , $\%E_i$ is the distribution of $Class\ 1_i$, and $\%\overline{E}_i$ is the distribution of $Class\ 0_i$.

Variable	count	$E_i(y=1)$	$\overline{E}_i(y=0)$	$\%E_i$	$\%\overline{E}_i$
$bucket_1$	N_1	E_1	$\overline{E}_1$	E_1/N_E	$\overline{E}_1/N_{\overline{E}}$
$bucket_2$	N_2	E_2	$\overline{E}_2$	E_2/N_E	$\overline{E}_2/N_{\overline{E}}$
$bucket_3$	N_3	E_3	$\overline{E}_3$	E_3/N_E	$\overline{E}_3/N_{\overline{E}}$
$bucket_4$	N_4	E_4	$\overline{E}_4$	E_4/N_E	$\overline{E}_4/N_{\overline{E}}$
$bucket_5$	N_5	E_5	$\overline{E}_5$	E_5/N_E	$\overline{E}_5/N_{\overline{E}}$
Total	N	N_E	$N_{\overline{E}}$		

Table 3.1. Example: first steps to compute WOE & IV

Variable	WOE_i	$\%E_i - \%\overline{E}_i$	IV_i
$bucket_1$	$log((\%E_1)/(\%\overline{E}_1))$	$(\%E_1) - (\overline{E}_1)$	$(\%E_1 - \%\overline{E}_1) * WOE_1$
$bucket_2$	$log((\%E_2)/(\%\overline{E}_2)$	$(\%E_2) - (\overline{E}_2)$	$(\%E_2 - \%\overline{E}_2) * WOE_2$
$bucket_3$	$log((\%E_3)/(\%\overline{E}_3))$	$(\%E_3 - (\overline{E}_3)$	$(\%E_3) - \%\overline{E}_3) * WOE_3$
$bucket_4$	$log((\%E_4)/(\%\overline{E}_4))$	$(\%E_4) - (\overline{E}_4)$	$(\%E_4 - \%\overline{E}_4) * WOE_4$
$bucket_5$	$log((\%E_5)/(\%\overline{E}_5))$	$(\%E_5) - (\overline{E}_5)$	$(\%E_5 - \%\overline{E}_5) * WOE_5$
Total		**IV**	$\sum_i^5 (IV_i)$

Table 3.2. Example: Final steps to compute WOE & IV

In the table 3.3 below we can see the predictive power of the variable compared to the IV value.

IV	Predictive Power
(-∞,0.02]	not useful for prediction
(0.02,0.1]	weak predictor
(0.1,0.3]	medium predictor
(0.3,0.5]	strong
(0.5, ∞)	suspicious

Table 3.3. Predictive power of IV

3.5 Partitioning data set

The model we are going to talk about will be trained over 2 months and tested over 2 months. Further more, we will implement all of this procedure on 2 different data set one concerning the business customer (B2B) and one concerning residential one (B2C), whereas from history business customers have different behaviour from residential.

3.6 Pipeline

To automatise the process of creating dummies for the categorical variables and scaling numerical variables, we will use a pipeline. A pipeline is a structured work-flow of analytic actions that is represented as a diagram. In the pipeline diagram, a node represents each action. The pipeline allows us to make the same action in the validation test. It is also really useful to put another parameter (`handle_unknown='ignore'`) in the one hot encoding function in order to not have problems if we have new features in the validation set not already seen in the train.

Those process are performed inside the pipeline

One hot encoder get the dummies for the categorical variables we need to apply this because vector assembler need that all variables are numeric types or boolean type.

Scaling We used the min-max scaling rule:

$$Scale_{min_{\&}max}(x_i) = \frac{x_i - x_{min}}{x_{max} - x_{min}} * (max - min) + min$$

where the output of $Scale_{min_{\&}max}(x_i) \in [0, 1]$

This was applied just for the logistic model, because the presence of outliers can bias predictions.

vector assembler Before training a model it is important we construct a single vector column from all columns for each row, because the model takes as input a vector of features

Chapter 4

Proposed models

After completed all these preprocessing steps on the study base, we are going to create a statistical model, that is a function which will allow us to best explain our target variable.

Since the company does not only want to predict complaints, in order to estimate the budget to allocate for the next year campaign to prevent complaints, but also to understand the reasons why customers complain, we need an interpretable model.

Further more it will be useful to have a score as a prediction to create a priority list, so that actions about a customer can be done in a sequential order, from most urgent to the least one.

We decided to try the LASSO logistic, Random Forest and Gradient Boosting classifiers, that will be formally presented in the section 4.1.

In chapter 5 we will compare the three models using the main metrics and decide which one is the best.

4.1 Models

4.1.1 Logistic

The first model we built is a logistic regression. This model is very common in classification tasks because it is easily interpretable. We can formalise it as follows: given $\mathbf{Y}$, a binary target variable , and $X_i = (X_{i1},...,X_{in})$ as explanatory variables for each customer i, and β, the vector of coefficients of the logistic regression, the model can be written like that:

$$Y_i = \begin{cases} 1 \text{ with probability : } \pi(X_i) \\ 0 \text{ with probability: } 1 - \pi(X_i) \end{cases} \qquad with \qquad \pi(X_i) = \frac{e^{\beta X_i}}{1 + e^{\beta X_i}} = \frac{1}{1 + e^{-\beta X_i}}$$

In Figure 4.1 below we can see the representation of $\pi(X_i)$:

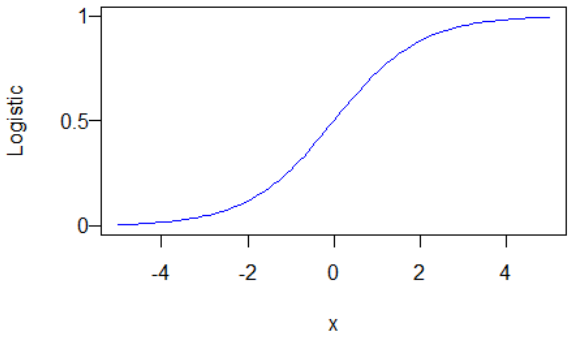

Figure 4.1. Logistic function

The logistic function $\pi(X_i)$ takes real values and transforms them to probabilities. The vector of coefficients β is linked to $\pi(X_i)$ via the Logit transformation:

$$logit(\pi(X_i)) = \ln(\frac{\pi(X_i)}{1-\pi(X_i)}) = \beta X_i \qquad \forall i$$

The logit Link transforms probabilities (between 0 and 1) to logit scores (between negative infinity and positive infinity). The logit function corresponds to the log Odds

By applying the exponential to the logit transformation we obtain:

$$\frac{\pi(X_i)}{1-\pi(X_i)} = e^{\beta X_i}$$

This is the multiplicative model for the odds: β represents the change when a feature passes from x_i to $x_i + 1$ holding all other features constant.

This form of the model makes it possible to interpret the coefficients as a measure of association between the target variable and the features.

The method used to find the vector of regression coefficients β is the Maximum likelihood estimation (MLE) That is, finding the solution to this equation:

$$\frac{\mathrm{d}}{\mathrm{d}\beta} \prod \pi(X_i)^{y_i}(1-\pi(X_i))^{1-y_i} = 0$$

where $\pi(X_i)^{y_i}(1-\pi(X_i))^{1-y_i}$ is the probability density function of Y.

The main advantage of this model is that you can interpreter coefficients as probabilities, so you can rank your predictions and than decide boundaries to classify customers.

There are two things to be careful with: first we should not train the logistic regression with a feature that perfectly distinguish the binary target, otherwise the weight of the feature would be infinite, leading to no convergence; Secondly the explanatory variables must be scaled so that the weights of variables can be compared with each other.

4.1.2 Improvement of logistic

The standard Logistic regression was optimised by changing of some default parameters in the logistic classifier in PySpark.

First of all, the weights were added to the model. For each observation in the train set we assigned a non negative value following this rule:

$$weight_i = \begin{cases} \frac{N}{n_0 \cdot 2} & \text{if } y_i = 0 \\ \frac{N}{n_1 \cdot 2} & \text{if } y_i = 1 \end{cases} \tag{4.1}$$

where N is the length of the train set, and n_0 & n_i are the number of observations belonging to $Class_0$ and $Class_1$ respectively.

By adding weights we manage to have relatively large weights for $Class_1$, giving them more influence during the analysis than observations that have smaller weights.

In addition, in the case of the logistic model we decided to use the feature normalised created with the pipeline

Lasso

We tried also to set the elasticNetParam to 1 to transform our logistic classifier into a logistic Lasso classifier

So the L_1 penalty was added to the logistic model. This penalty forces the sum of the absolute values of the model parameters to be less than a fixed upper limit, therefore forcing some of the coefficients towards zero. We end up with only the most informative variables in the final model.

Lasso penalisation can be defined as follows:

$$\hat{\beta}^L = argmin_\beta \|y - X\beta\|_2^2 + \lambda \sum_{j=1}^{P} |\beta_j|$$

where $\lambda > 0$ is the control parameter of the power of penalty. The larger λ, the more coefficients are forced to be exactly zero. The optimal value for λ can be chosen via cross validation.

One of the advantages of LASSO is that, by having some of the coefficients equal to zero, the variance of the model is decreased while the bias is not altered, thus reducing overfitting.

4.1.3 Random Forest

Random Forest is an ensemble model, so it is a combination of Decision Tree (A.1) classifiers. The combination of simpler model is called bagging.

The basic idea behind this method is to reduce the high variance of the base model (without increasing the bias) averaging the simple base classifiers.In order to do that we need independence of the components we are averaging. To recover some independence, each tree is built on different rows and columns of the dataset.

In fact, in the standard Random Forest model, the Decision Tree are trained on a random sub-sample of the available training dataset (bootstrap) whose features, at each branch split, are also a random sub-sample of the original set of features.

This additional variability in the features used leads to better predictive accuracy, improving the variance term of the error of the model.

In a binary classification problem, each tree casts a vote for one of the two classes: the class receiving the majority of the votes is the one predicted by the model.

In Figure 4.2 we can observe the schema of Random forest model:

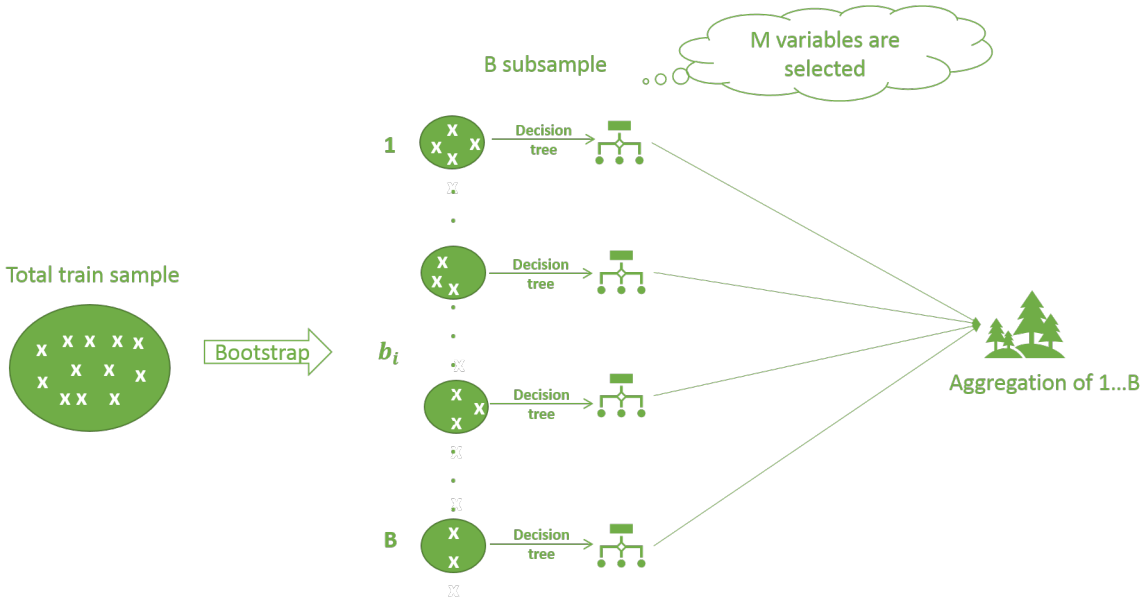

Figure 4.2. Random Forest schema

Parameters

Some parameters have to be set before training the model in order to improve performance and avoid overfitting. One way to to this is via a grid search: selecting the combination of parameters that leads to the lowest error on the training set.

- **max depht**: is the maximum length of the longest path from the first (root) node to a leaf node (`maxDepth=16`).
- **Feature subset strategy**: M, the number of variable used to build each tree. (`featureSubsetStrategy='sqrt'`).
- **Number of trees**: B, the number of different trees that are built and then aggregated. An high number can make the computation slow but makes prediction stable (decreasing variance) (`numTrees=150`).
- **Minimum instances per node**: the minimum number of observations that a node must have. If small the model can be too detailed and give too much importance to noise. (`minInstancesPerNode=50`).
- **Subsampling rate**: is a fraction of the train sample, thus the size of each b_i tree in the forest.(`subsamplingRate=0.8`) .

Feature importance

One useful aspect of random forest is that it automatically returns the relative importance of the features of the data. In the case of classification trees, a first summary of the importance of a variable can be obtained using entropy as Gini's index. We calculate the number of times the Gini's index has decreased by partitioning with respect to the variable j. We get this number for each classification tree b_i. The average value of this indicator on the B bootstrap samples gives us the importance of variable j.

Advantages:

- easy to interpret.
- can handle categorical variables without the need to create dummy variables.
- automatically handles missing values and variable reduction.
- not effected by outliers.
- trees are independent: this make the process parallelizable.

Disadvantages:

- tends to overfit if the number of leafs is high , we have to be careful in setting parameters.
- a lot of parameters to be tuned.

4.1.4 Gradient Boosted Tree classifier

Boosting is another example of ensemble method: we combine many weak learners, improving their performance by lowering the variance term of the model's error. The basic idea of boosting is fundamentally different from that of random forest: this algorithm follows a recursive process; the rule built in step k depends on the rule built in step k–1. In random forest, the weak learners where independent from each other, while in boosting they are strongly dependent.

When applying boosting to decision trees by fitting the negative gradient of the residuals, we obtain the Gradient Boosting Decision Tree Classifier.

In the Figure 4.3 below we can see a schema of the process:

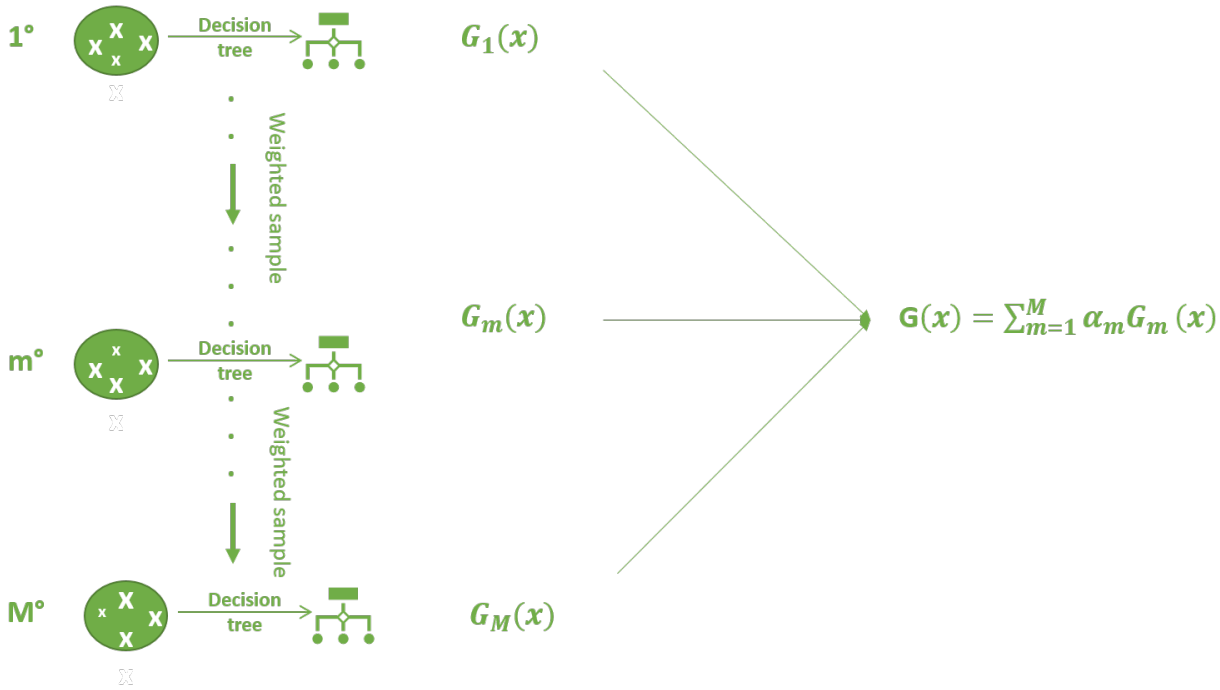

Figure 4.3. Schema of GBT

The goal of the algorithm is to learn from is mistakes and weight in a meaningful way instances before the next iteration. Instances that are misclassified at iteration k receive a higher weight at iteration $k+1$, and the opposite is true for instances correctly classified. So, across iterations, the algorithm keeps track of the residuals to weight each sample: more weight is assigned repeatedly to instances which are difficult to classify.

This process is repeated until a previously specified number of trees is reached, or the loss is reduced below a certain threshold.

We can describe the main steps of the algorithm as follows:

1. Initialisation : $g_0(\cdot) = \text{argmin}_c \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, c)$, where ℓ is the loss function that we try to minimise empirically.

2. For $m = 1$ to M: Compute the opposite of the gradient $-\frac{\partial}{\partial g(x_i)} \ell(y_i, g_m(x_i))$ and evaluate it at the points $g_{m-1}(x_i)$:

$$U_i = -\frac{\partial}{\partial g(x_i)} \ell(y_i, g_m(x_i))\Big|_{g(x_i)=g_{m-1}(x_i)}, \quad i = 1, \ldots, n$$

Fit a tree to K terminal nodes on the sample $(x_1, U_1), \ldots, (x_n, U_n)$ Compute the optimal prediction for each node $\rho_1, \ldots, \rho_K$; update:

$$\rho_k = \underset{\rho}{\text{argmin}} \sum_{x \in R_k} \ell(y_i, g_{m-1}(x_i) + \rho)$$

where R_k contains the ensemble of x_i that belongs to k^{th} node of the tree. Update : $g_m(x) = g_{m-1}(x) + \lambda\rho_{k(x)}, k(x)$ indicating the number of the node which contains x, and λ is the learning rate between 0 and 1.

3. Output : $\hat{g}_M(x)$

Parameters

- **Max number of iteration** : M, trees in the sequence. (`maxIter=40`)
- **Max depth** of each decision tree (`maxDepth=10`)
- **Learning Rate** controls the size of the weight changes. It ranges from 0 (exclusive) to 1. The default initial value is 0.1. The default initial value for the learning rate is 0.1. The default for the range is from 0.01 to 1.(`learningrate=0.01`)

Advantages

The main advantage of boosting is that the algorithm focuses on the misclassified cases. This improves the performance of the model.

Disavantages

It leads to overfitting because tries to recover mistakes in the train.

4.2 Prediction

As output our model will give us a rank that represents the best guess of our model. Each model will produce a list of customers ranked from highest to lowest priority. In this case priority means treat him before others because he is more likely to complain. Rank values are meaningless, just the relative order is important. We can convert the rank prediction into binary decision by classifying all ranks above a threshold t as *Class 0* and all ranks below the t as *Class 1*.

4.3 Metrics

To compare the models and choose the best one, many different metrics were used :

Confusion matrix is optimal to summarise performance and retrieve many other useful metrics. In the table below we have its notation:

		predicted	
		1	0
actual	1	TP	FN
	0	FP	TN

Table 4.1. Confusion Matrix notation

In our case P will be events of *class 1* and N events in *class 0*. Our model will assign a prediction score, of belonging to *class 1*, to each row of the validation set, in order to compute the confusion matrix. First we have to convert our scores to a binary prediction. We cannot assume that the scores are probabilities since they are uncalibrated, but they can be mapped into an interval between 0 and 1.

To make this conversion we will need to set a good threshold (T_h) and apply the definition below:

$$\hat{y}_i = \begin{cases} 1 \text{ if } & \hat{y}_i > T_h \\ 0 & \text{otherwise} \end{cases}$$

The way to decide the best threshold will be described next.

Since some of the predictions of the model will be wrong and some good we will have False Negative (FN) and False Positive (FP).

We can retrieve some measures from the table 4.1:

Precision: is the proportion of well predicted of *class 1* over all the predicted as 1:

$$\frac{TP}{TP + FP}$$

Recall(sensity): is the True Positive Rate, so the proportion of actual positives that are correctly identified:

$$TPR = \frac{TP}{TP + FN}$$

where $TP + FN$ the total number of actual positives (P)

F1: summarises the precision and recall metrics in one single score:

$$2 \cdot \frac{precision * recall}{precision + recall}$$

Specificity: is the true negative rate: fraction of true negative that are correctly classified:

$$\frac{TN}{TN + FP}$$

False positive Rate: proportion of actual negatives that are not correctly identified:

$$FPR = \frac{FP}{FP + TN} = 1 - \text{specificity}$$

with FP the number of false positives, TN the number of true negatives and $FP + TN$ the total number of actual negatives N.

Accuracy: is the proportion of correctly predicted samples over all.

$$\frac{TP + TN}{P + N}$$

Precision 5000: It is the precision calculated over the first 5000 customers ordered by the score provided by the model. This metric will be critical to choose the best model because this work aims to be optimal on 5000 customers. If the model works well they can decide to increase this number, thus the investments.

Selection of the threshold: In order to decide the best cut-off we used the Precision & recall scores as a function of the decision threshold

An example of the precision recall curve is shown below in Figure 4.4 :

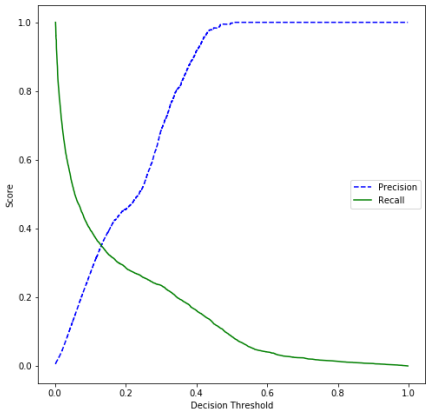

Figure 4.4. Precision recall scores curve

The x-axis shows the threshold instead the y-axis represents the score of recall for the green curve and precision for the blue one.

Thus every point represents the precision and the recall that we will get when we choose a determined threshold.

A high precision relates to a low false positive rate, and high recall relates to a low False Negative Rate.

Since we would like to have both of this high we chose as threshold the intersection between the 2 curves.

Choosing the right threshold is hugely important because the dataset is unbalanced, so the standard 0.5 would be bad for sure.

ROC curve: is a tool for evaluating and comparing models. An example of the curve is shown below in Figure 4.5 :

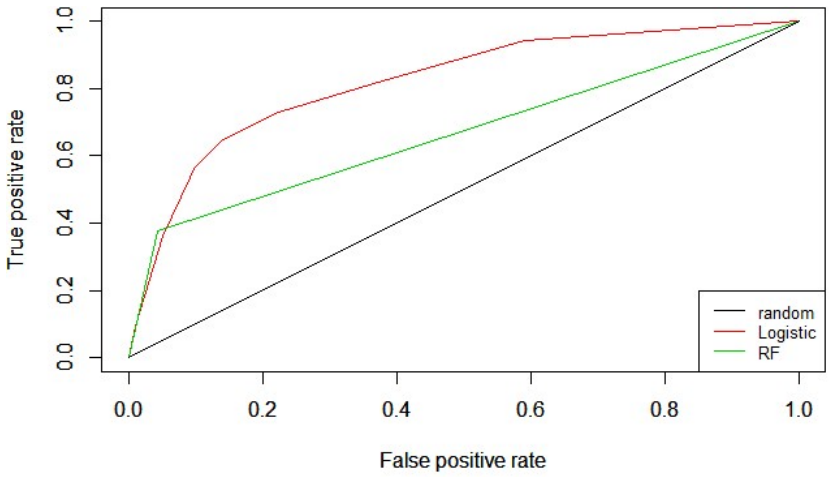

Figure 4.5. An example of a ROC curve

We have on the x axis the False Positive Rate (FPR) and on the y axis True Positive Rate (TPR). The curve shows how TPR and FPR change by varying the threshold from 1 to 0.

In Figure 4.5 we show the ROC curve for 3 different models: the black one is just an example of the worst case scenario,random guessing; the red curve represents a logistic regression model and the green one is a Random forest model.

The best model according to the ROC metric is the one with the Area Under the Curve (AUC) closer to 1; in this case the red one.

Decreasing the value of the classification threshold leads to classify more elements as positive, causing an increase in the number of both false and true positives.

ROC curve can be useful also to compare performance between train and validation to check whether the model is overfitting. Furthermore it can help deciding the best threshold , to gain the best accuracy, by finding the maximum of the curve.

Lift: The Lift chart is the main metric we used because it allows us to express the dependency between costs and expected benefit. It corresponds to the concentration of individuals who actually complained (TP) according to the number of individuals on the top part (n) of a model-score ranked list.

$$Lift(n) = \frac{\sum_{j=1}^{n}(y_j)}{\frac{n}{N}\sum_{i=1}^{n}(y_i)} = \frac{TP_n/n}{(TP+FN)/N}$$

Pay attention that it can happen that a model with lower accuracy or AUC, has a larger lift or viceversa.

In the plot below we can observe a zoom of the lift at $\frac{n}{N} = 0.01$:

Figure 4.6 :

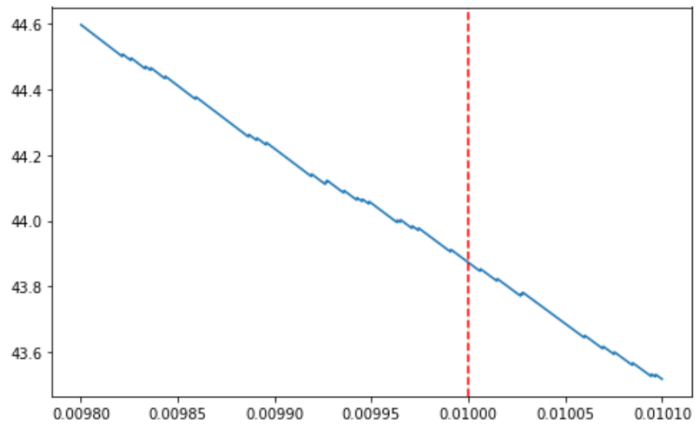

Figure 4.6. An example of a zoom of a Lift chart

A lift of 1 is equivalent to no gain compared with contacting a customer at random, whereas a lift of 2 corresponds to the situation where there are twice the number of complaints reached compared to the number you would expect by contacting the same number of customer at random

Lift can be seen also as the ratio of the percentage of captured responses within each percentile bin to the average percent of responses for the model.

The higher the lift in the lower percentiles of the chart, the better the model is.

Chapter 5

Results

An optimal model has been chosen using the model assessment measured on the validation data. The results in the validation are reported as an upper bound on the performance expected when the model is deployed.

Lift, precision and miss-classification will be examined in the validation to assess the performance of the models.

All the metrics we are going to present are computed over customers. So the predictions for the services are aggregated and the maximum score is kept .

As already said in section 3.5, in order to improve our performance we built two different models: one for business and one for residential customers. In fact they have different variables that leads a customer to complain, as we will see in the feature importance analysis of the two models.

It will be described later, in section 5.2, the strategy to combine both results to increase the final performance.

5.1 Model selection

In order to choose the best model we compared the main metrics presented in section 4.3 .

The analysis presented below will refer to the business data set.

First we check how were the performance in the train set, but to choose the best model we will base our decision on the results of the validation set.

In table 5.1 we find the result of the train data set. It seems that the best model is the GBT because of the highest lift, but we have to check the performance on the validation to asses that we are not overfitting,

Models	**LASSO**	**RF**	**GBT**
lift	21.6	64.4	76
threshold	0.99	0.2	0.4
$\boldsymbol{F_1}$	0.13	0.61	0.74
$\boldsymbol{service_{100k}}$	9%	16%	19.3%
$\boldsymbol{custumer_{100k}}$	1.2%	2.5%	4.1%

Table 5.1. Result comparison train BUS

Lift is high in both GBT and RF but not in LASSO. The threshold proposed is low for the ensemble models but high in LASSO due to uncalibrated score . Precision_{100} is computed standardly and over customers. It is not high in none of them but precision over $100k$ observations is still acceptable.

We see that the GBT was overfitting from the validation results (section 5.2): we tried by changing the parameters of the GBT classifier to reduce overfitting but this was the best result.

In this case the Random forest model is the best one; we choose this model for the business.

For completeness we show in the table 5.2 all of the metrics considered alongside the ones we actually used to choose the model. The metrics more determining are *Lift* and $customers_{5k}$; Random Forest has the highest in both months of validation.

With this model we gain in performance as we have already seen, in interpretability and computational time.

	Aprile			Maggio		
Models	**LASSO**	**RF**	**GBT**	**LASSO**	**RF**	**GBT**
Lift	21.8	32.53	24.2%	30.16	38.6	24.4
$\boldsymbol{max_{prob}}$	1	0.57	0.99	1	0.51	0.99
F_1	0.16	0.21	0.16	0.02	0.09	0.19
$\boldsymbol{service_{100k}}$	4.4%	5%	4.6%	7.1%	7.8%	6.1%
$\boldsymbol{customer_{100k}}$	1.6%	2%	1.98%	1.6%	2%	1.86%
$\boldsymbol{service_{5k}}$	7.8%	15.9%	13.26%	9.3%	14.2%	8.7%
$\boldsymbol{customer_{5k}}$	22.6%	33.9%	20.3%	19.4%	22.8%	10%

Table 5.2. Validation results

We apply the same procedure to choose the best model for the residential customers;Random Forest is again the best one.

We present in the next sections the results of the best models, separately business and residential, by exploring feature importance and the metrics presented in section 4.3.

5.1.1 Business (B2B)

In the plot below we can see the variable importance of the random forest model.

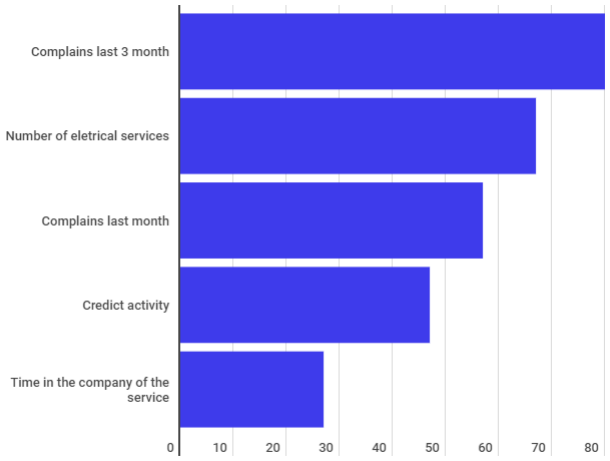

Figure 5.1. Feature importance B2B

These are the variables that define the splitting rules that manage to distinguish better complainers and non-complainers.

One of the most interesting and strong in importance is how many complaints you have done before the reference date. This can help the company in making decision about dealing with a complaint.

Further more we can observe the main metrics in the table 5.3 below .

Month	April	May	June
F1	0,17	0,15	0,12
Precision 3k	22%	19%	13%
Precision 5k	16%	15%	10%
Lift 0.01	30	27	26
Max (Pr)	0,7	0,7	0,7

Table 5.3. Result Business

We do not achieve the goal (20%) but we are really close to it. We will see in the next section how combining the models will improve this result and will get us closer to the goal.

Precision over 5k customers are stable in the first 2 months, it decreases little in June but we expected this cause the total number of complaints in the month of June was smaller. 1.4.3

5.1.2 Residential (B2C)

As we can observe from table 5.4 metrics are quite stable but they are greater in April because is the closer month to the train.

Month	April	May	June
F1: 1	0,12	0,11	0,08
Precision 3k	20%	18%	13%
Precision 5k	16%	15%	10%
Lift 0.01	30	27	26
Max (Pr)	0,7	0,7	0,7

Table 5.4. Result resident

It can be that in June results look worse because there are less complaints in that month (1.4.3)

It is important to notice that precision over 3000 customers is higher than over 5000: this means that the model is working well and putting priority on true positives. This was an hint to decide the process to combine the results between B2C and B2B (5.2).

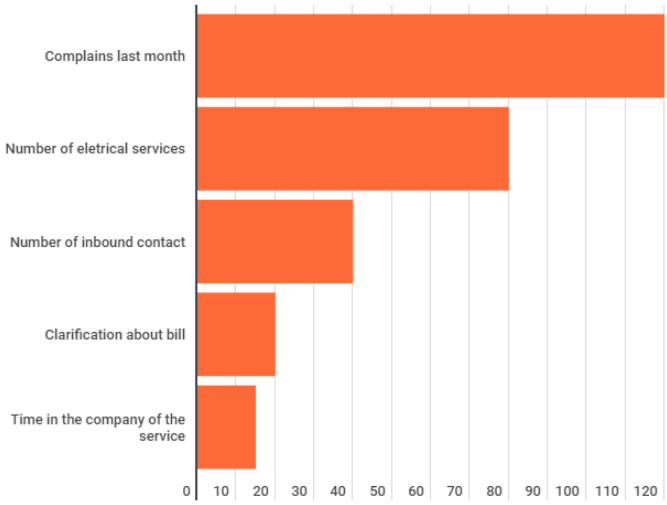

Figure 5.2. Feature importance B2C

As we can see from the figure above 5.2 the features that contribute more to this model are different from the business one, but the main ones are the same.

Feature importance helps the company to learn about products or pricing plan weak points, operation issues, as well as customer preferences and expectations to proactively reduce reasons of complaints.

5.2 Combine results

In order to combine the two list of customer and achieve the best performance we decided to combine the result following the schema below:

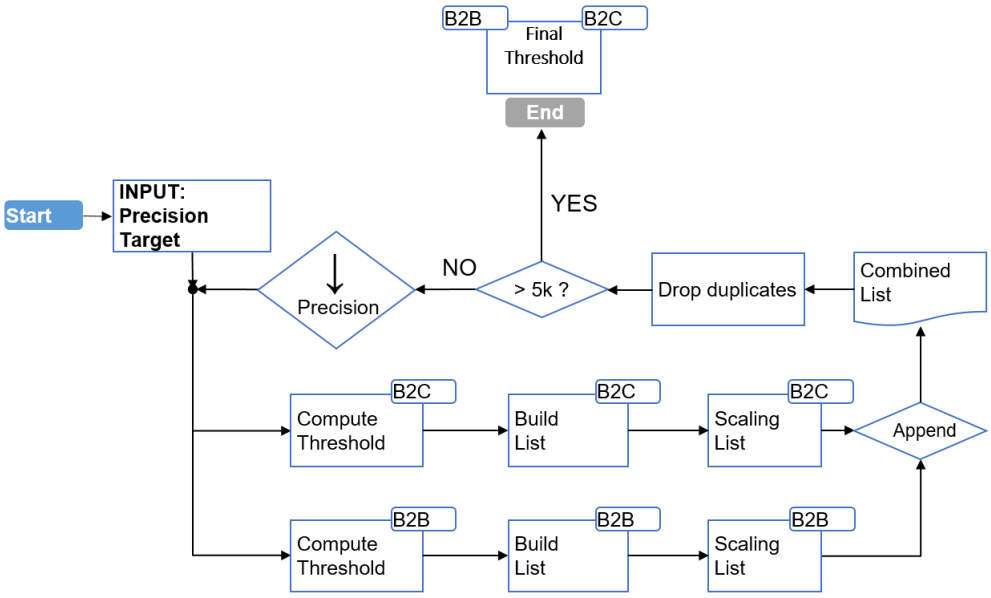

Figure 5.3. Logic to set threshold before combine models

We start by selecting the precision and the threshold in April, that will be used in the next prediction months , May and June. Those steps are explained in Figure 5.3:

- First of all we set an overall precision that we want to have for both models
- we calculate the threshold for each model in order to achieve the precision desired
- we generate the list of complainers over this threshold for each model
- we scale the score of the list generated before, between 0 and 1
- we append the 2 different list generated by the model and than order by scaled score
- we check if there are enough (not less than 5000) customers displayed in the combined list
  - if not, we reduce a little the precision and repeat the steps above till we reach 5000
  - if yes, it's done we have the desired threshold to reach the best precision

Now that we know the right threshold to chose to have a good length list , we have to follow the next figure 5.4:

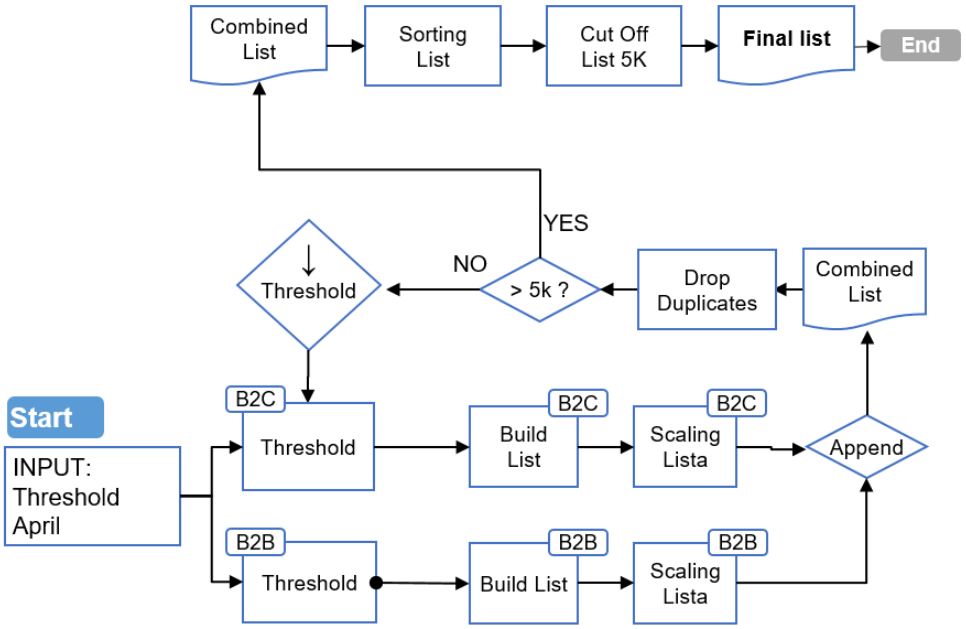

Figure 5.4. Process to combine list

- we generate one list, from each different model, of customer over the threshold set with the previous steps

- we scale the score of the list generated before, between 0 and 1

- we append the 2 different list generated by the models and than order by scaled score

- we check if there are enough (not less than 5000) customers displayed in the combined list

  – if not, we reduce a little the thresholds and repeat the steps above till we reach 5000

  – if yes, finally we have the list of customer that will likely complain

In the figure 5.5 below we can see how the precision of the combined models is better than the one of the single models.

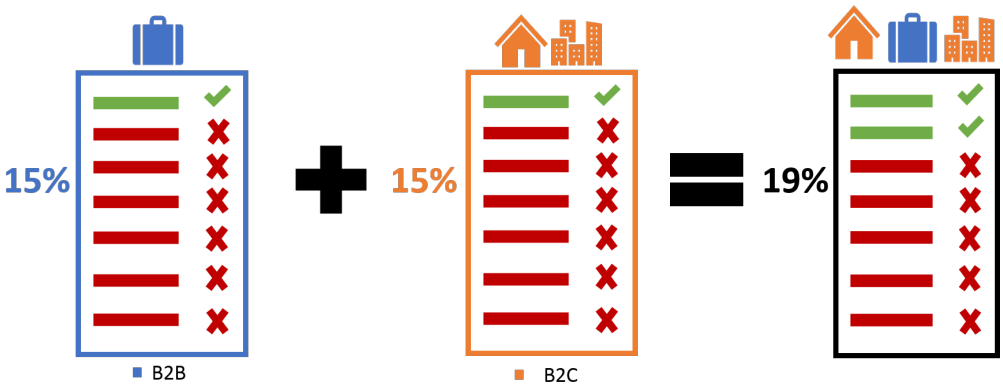

Figure 5.5. Precision 5k of May

This is a great result because the model that combines the outputs, allows to increase the precision by 3-4% compared to the performance of the separate single model. Even though the performance is lower in the month of June , we achieve always greater performance in the combined list.

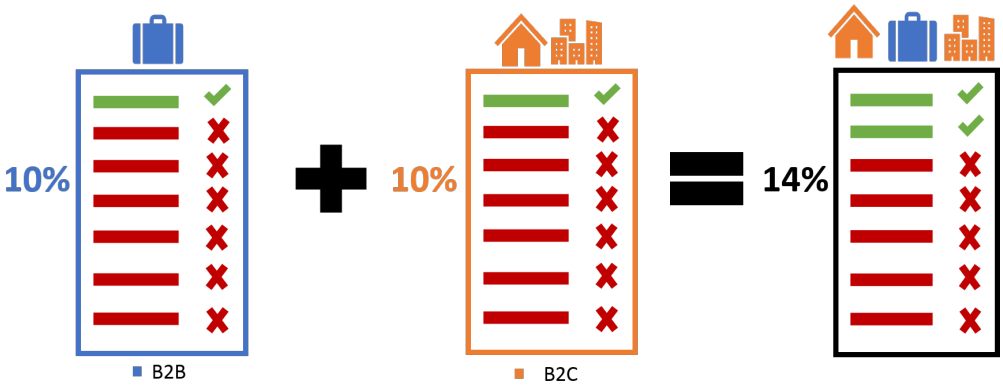

Figure 5.6. Precision 5k of June

In figure 5.7 we can observe in dark blue and light blue the proportion of business and residents that we are going to predict as potential complainers in the month of May.

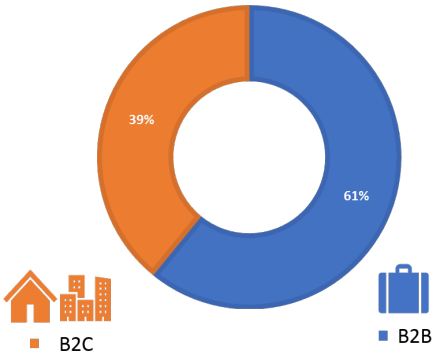

Figure 5.7. Percentage of business and resident in the list of 5k of May

As we expected there are more business customers: this is because we have better performance in predicting them.

The best model has been put into production; of course it is still monitored by data scientists to maintain the best level of prediction accuracy. Monitoring the model is important because the scenario changes over time.

Chapter 6

Conclusion

In this thesis, we studied the problem of predicting customer complaints at one of the largest energy business companies in Italy. We developed a model of prediction based on machine learning, and more specifically, we identified "*random forest*" classification as the most precise technique to address the above problem. To assess the precision of our model, we evaluated our techniques over a large amount of data comprising a list of 5000 customers in the month of September 2019. We achieved a precision of 12%. This was in line with our initial expectations that were based on an upper bound of 14% precision computed during our analysis. The results of our predictions have been used by the company to contact the set of identified customers during the month of October, thus anticipating possible complaints.

To demonstrate the usefulness of our model within the company business operation, we will now analyse costs and benefits. All these analyses have been performed assuming the best case scenario, *i.e.*, when a customer is successfully contacted by the company and his/her issue is resolved within the contact process.

We first compare the costs for the company with and without our developed model. As a first step, we need to calculate the cost of contacting a customer ($Cost_{contact}$), the cost of dealing with a complaint ($Cost_{complaint}$), the total number of customers contactable in a month ($N_{contact}$), the average number of complaints in a month ($\bar{N}_{complaint}$), and the average number of correctly predicted customers' complaints ($\bar{N}_{predicted_1}$). The overall cost can be formulated as follows:

$$C_{tot} = (Cost_{contact} \cdot N_{contact}) + (Cost_{complaint} \cdot \bar{N}_{complaint}) - (\bar{N}_{predicted_1} \cdot N_{contact})$$

The expected total cost of the complaints for the company, using our model, is therefore equal to the sum of the cost of contacting all the 5000 customers of the list given by our model and the average cost of dealing with customers that will complain within a month, minus the benefits the company obtain by successfully predicting one customer's complaint before the company has to deal with him/her.

The average cost per month without the model would instead be equal to $Cost_{complaint} * \bar{N}_{complaint}$, a simplified approximation.

In Figure 6.1, we can see the results drawn from the two equations explained above. Our findings show that the company would save up to 11% of the costs when adopting our model.

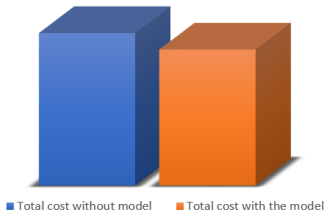

Figure 6.1. Cost comparison.

We note that the ultimate objective of the company is achieving 20% of precision per month. This means reducing the number of customers who complain, who may also churn, by 12000 in a year.

Reaching this level of precision is however not enough since the company also needs to successfully help the customers with their issues beyond identifying such customers.

In order to gather additional information for solving the problem of identifying customers that are likely to complain, we have started developing a multi class model to predict the motivation of the claims (*e.g.*, bill clarification, offer clarification, credit, execution of work, activation problems, selling claims, infrastructure and network, delay). A comprehensive evaluation of the results of this extended model is left as future work.

Glossary

Area Under the Curve (AUC) is computed as :

$$\int_0^1 \frac{TP}{P}\, d\frac{FP}{N}$$

B2B Business to Business

B2C Business to Consumer

Big Data are characterised by the 4 V: huge Volume, comes from Variety of sources, Variety of formats and Velocity of flows.

Business translator is the link between scientific and business workers. They guarantee focus on the business value, and finding efficient and interpretable solutions.

Churn is when existing customers end a relationship with a company.

Ensemble model In this class of algorithms, a complex model is build starting from a, usually high, number of simpler models, and the final prediction of the model is a combination of the predictions from the smaller component.

Entropy is

$$\text{Entropy}= \sum_i^C f_i * log(f_i)$$

where f_i is the frequency of label i at a node and C is the number of unique labels (2 in case of binary target).

ETL Extract, Transform, Load.

Euclidian distance Given 2 vectors $X = x_1, ...x_n$ and $Y = y_1, ...y_n$, The euclidian distance is the shortest distance between X and Y:

$$D(X,Y) = ||X - Z|| = [\sum_{i=1}^{n}(x_i - y_i)^2]^2$$

False Negative Rate proportion of actual positives that are not correctly identified:

$$FNR = \frac{FN}{FN + TP}$$

with FN the number of false negatives, TP the number of true positives and $FN + TP$ the number of actual positives

False Negative (FN) Actually positive but classified as negative

False Positive (FP) Actually negative but classified as positive.

False Positive Rate (FPR) proportion of actual negatives that are not correctly identified:

$$FPR = \frac{FP}{FP + TN}$$

with FP the number of false positives, TN the number of true negatives and $FP + TN$ the number of actual negatives.

Gini is

$$Gini = \sum_{i}^{C} f_i(1-f_i)$$

where f_i is the frequency of label i at a node and C is the number of unique labels (2 in case of binary target).

Hadoop is open source software used to process Big Data, in a scalable and flexible way thanks to the use of distributed computing environment.

ISTAT Istituto nazionale di statistica

Kurtosis is the mean to the power of 4 of a normalised variable Y:

$$K = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \bar{Y}}{\sigma}\right)^4$$

· If k = 3 it is like a standard normal distribution (symmetric)
· If k>3it indicates a heavy-tailed distribution,oppositely
· If k<3 indicates flatter distribution

Link is a function that maps real values between 0 and 1.

Machine Learning is a branch of artificial intelligence that builds automated process that learn iteratively from data and make prediction with minimal human intervention.

MapReduce is a programme model developed by Google; it can be divided into 3 main steps:

- **Map** This step allow to apply a function separately to input pair $key, value$ and produces a set of intermediate $key^{new}, value^{new}$ pairs

- **Shuffle** The algorithm will sort on each server the data that go together with respect to their key . Shuffling is basically sending map output partitions to the corresponding reduce tasks.

- **Reduce** Finally, it will group all the data of each server by key and return the result.

Maximum likelihood estimation (MLE) is an approach for estimating the parameters (θ) of a probability distribution by maximising a likelihood function. In order to maximise the likelihood, this equation must be solved:

$$\frac{\mathrm{d}}{\mathrm{d}\theta} \; L(X, \theta) \mid_{\theta=\hat{\theta}_{MLE}}$$

MLlib Machine Learning Library is a Distributed Machine Learning structure above Spark in view of the distributed memory-based Spark architecture.

NTT Nippon Telegraph and Telephone

Odds is the probability that an event occurs divided by probability of the no event

$$\frac{P(E=1)}{P(E=0)}$$

Outlier is a value that is well away from the other observation, abnormally small or high.

Overfitting issue in a model when predictions correspond too closely to the training data and may therefore fail to predict unseen observations. It leads to small bias but large variance.

ROC Receiver Operating Characteristic

ROI Return On investments

Skewnees is a normalised measure of asymmetry:

$$A = \frac{\mu - Me}{\sigma} \in [-1, +1]$$

- Positive if $A > 0 \rightarrow Me < \mu$
- Negative if $A < 0 \rightarrow \mu < Me$
- Symmetric if $A == 0$

Targeted Marketing involves first identifying individuals who are most likely to respond to a marketing action, and then designing a marketing campaign that targets those individuals. The goal of targeted marketing is to acquire new customers and generate additional revenue from existing customers.

True Positive Rate (TPR) proportion of actual positive that are correctly identified:

$$TPR = \frac{TP}{TP + FN}$$

with FP the number of false positives, TN the number of true negatives and $TP + FN$ the number of actual positives.

Unbalanced data set occurs when the response variable is a factor variable where the classes are not represented equally, that means that the number of data points available for one class is extremely different from the others.

Appendix A

A.1 Decision tree classifier

Decision Trees take decisions based on rules concerned with the values of the input variables. The rules are expressed in Boolean logic and can be represented hierarchically as nodes in a tree structure, where a node is the location where branches split according to the value of a variable. Nodes with only one connection are leaf nodes. Each tree leaf provides a prediction by choosing the class that has the largest posterior probability (most frequent class) in that leaf.

To define the best split (rule) for each input variable, recursive partitioning is used. This algorithm makes the optimal choice at each split depending on a splitting criterion that maximise the ; the splitting criterion measures the reduction in variability of the target distribution in the child nodes. There are two types of splitting criterion Gini impurity and Entropy. The criterion can be chosen as a parameter of the model. The goal is to reduce variability and thus increase purity in the child nodes.

This partitioning is repeated in each child node till some stopping rules determine the end of the process and the final depth of the tree. Stopping rules are parameters of the model:

- Maximum depth (`maxDepth`) of a tree. Deeper trees are more expressive (potentially allowing higher accuracy), but they are also more costly to train and are more likely to overfit.
- No split candidate leads to an information gain greater than a minimum information gain (`minInfoGain`). For a node to be split further, the split must improve at least this much (in terms of information gain)
- No split candidate produces child nodes which each have at least a given number of instances in a node (`minInstancesPerNode`)

Advantages: Easy to interpret can handle categorical variables without the need of creation of dummy variables Trees automatically handle missing values and variable reduction. Therefore, the input data requires less preparation and are not effected by outliers.

Disadvantages: It can be challenging deciding the maximum number of leaves because too many partitions lead to overfitting, and too few leaves lead to

simple model and large bias.

A.2 K-means

K-means is a clustering algorithm that aims to find K partitions of a data set that minimise the average squared Euclidian distance between each data point x_i and the cluster's center it belongs to, $\frac{1}{N}\sum_{i=1}^{N}[min_j D(x_i, z_j)^2]$. The algorithm implemented in pyspark is an optimised version of the k-means, k-means++. The difference is just in how the centers are chosen at the beginning of the algorithm. Below we see the steps to perform the k-means++ algorithm:

- Given $X = x_1, .., x_i, .., x_N$ we first need to find the K centers $Z = z_1, .., z_j, .., z_K$ following this steps:
  - Randomly choose one among $X : z_1 = x_j$
  - While $Z < K$,
    - $\forall x_i \in X - Z$, compute the Euclidian distance, $D(x_i, z_j) = ||x_i - z_j||$, and chose the next z with probability proportional to $D(x_i, z_j)^2$
- According to the rule of minimising distance we assign a point x_i to the closest cluster center z_j:
$$x_i \in C_j \quad if \ min_j ||x_i - z_j||$$
- We compute the average squared mean distance of each cluster and repeat the step until no change, convergence.

  Some disadvantages are that the algorithm can be effected by outliers and there is no rule to set K; the problem of initial cluster choice was solved with the improvement stated above.

Bibliography

[1] Agile project management. Available from: https://managedagile.com/what-are-the-advantages-and-disadvantages-of-agile-scrum/.

[2] Decision trees - rdd-based api. Available from: https://spark.apache.org/docs/2.1.0/mllib-decision-tree.html.

[3] Ensembles - rdd-based api. Available from: https://spark.apache.org/docs/2.1.0/mllib-ensembles.html#random-forests.

[4] Gradient boosting classifiers in python with scikit-learn. Available from: https://stackabuse.com/gradient-boosting-classifiers-in-python-with-scikit-learn/.

[5] Introduction to software engineering/process/v-model. Available from: https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Process/V-Model.

[6] Minmaxscaler. Available from: https://spark.apache.org/docs/2.1.0/ml-features.html#minmaxscaler.

[7] Precision-recall. Available from: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html.

[8] Vectorassembler. Available from: https://spark.apache.org/docs/latest/ml-features#vectorassembler.

[9] *Statistica*. Il Mulino (2010).

[10] *Weight of Evidence Module*. STATISTICA (2013). Available from: https://documentation.statsoft.com/portals/0/formula%20guide/Weight%20of%20Evidence%20Formula%20Guide.pdf.

[11] *Measures of Skewness and Kurtosis*. NIST/SEMATECH (April, 2012). Available from: https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm.

[12] Beakta, R. Big data and hadoop: A review paper. *Spl. Issue*, **2** (2015), 1694. Available from: https://pdfs.semanticscholar.org/ba3d/3f95465aacbece737322aaf74611b0611f93.pdf.

[13] Breiman, L. Random forests. *Machine learning*, **45** (2001), 5.

[14] Deng, K. *OMEGA: ON-LINE MEMORY-BASED GENERAL PURPOSE SYSTEM CLASSIFIER*. Ph.D. thesis, Carnegie Mellon University (1998). Available from: "https://www.cs.cmu.edu/~kdeng/thesis/logistic.pdf".

[15] Fonti, V. Feature selection using lasso. *Research Paper in Business Analytics*, (2017). Available from: https://beta.vu.nl/nl/Images/werkstuk-fonti_tcm235-836234.pdf.

[16] Isabelle Guyon, A. E. An introduction to variable and feature selection. *Journal of Machine Learning Research*, (2003), 1157. Available from: http://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf.

[17] ISTAT. Dati popolazione per comune. Available from: hhttp://dati.istat.it/Index.aspx?DataSetCode=DCIS_POPRES1.

[18] Jose Manuel Pereira, A. F. d. S., Mario Basto. The logistic lasso and ridge regression in predicting corporate failure. *Procedia Economics and Finance*, **39** (2016), 634. doi:https://doi.org/10.1016/S2212-5671(16)30310-0.

[19] Jyoti Yadav, M. S. A review of k-mean algorithm. *Physics Procedia*, **4** (july 2013), 2972 – 2975. Available from: https://pdfs.semanticscholar.org/65f1/232434c5eeddd9e658db7ae0dd5c47b6e20d.pdf.

[20] Littler, S. Cumulative gains and lift curves: Measuring the performance of a marketing campaign. Available from: https://select-statistics.co.uk/blog/cumulative-gains-and-lift-curves-measuring-the-performance-of-a.

[21] Miha Vuk, T. C. Roc curve, lift chart and calibration plot. **3** (2006), 89. Available from: http://www.stat.wvu.edu/~jharner/courses/dsci503/docs/vuk.pdf.

[22] Molnar, C. Interpretable machine learning. *Physics Procedia*, **4.2** (November 2019). Available from: https://christophm.github.io/interpretable-ml-book/.

[23] MUKHERJEE, S. Information value (iv) and weight of evidence (woe). (2018). Available from: https://stepupanalytics.com/information-value-iv-and-weight-of-evidence-woe/.

[24] Murat KORKMAZ1, Y. Y., Selami GÜNEY2. The importance of logistic regression implementation in the turkish livestock sector and logistic regression implementations/fields. *Research Article*, **16** (2012), 25. Available from: https://dergipark.org.tr/tr/download/article-file/172263.

[25] What is v-model- advantages, disadvantages and when to use it? Available from: http://tryqa.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/.

[26] Oyelade, O., Oladipupo. Application of k-means clustering algorithm for prediction of students' academic performance. *(IJCSIS) International Journal*

of Computer Science and Information Security, **7** (2010). Available from: https://arxiv.org/ftp/arxiv/papers/1002/1002.2425.pdf.

[27] SAS. Interactive machine learning course notes.

[28] S.Gopal Krishna Patro, K. K. s. Normalization: A preprocessing stage. (2015). Available from: https://www.researchgate.net/publication/274012376_Normalization_A_Preprocessing_Stage.

[29] Shmueli, G. Lift up and act! classifier performance in resource-constrained applications. (2019). Available from: https://arxiv.org/pdf/1906.03374.pdf.

[30] Youguo Li, H. W. A clustering method based on k-means algorithm. *Physics Procedia*, **25** (2012), 1104 – 1109. Available from: https://www.sciencedirect.com/science/article/pii/S1875389212006220.