UNIVERSITAT DE BARCELONA

FUNDAMENTAL PRINCIPLES OF DATA SCIENCE MASTER'S THESIS

Apply Machine Learning in the Company to Predict the Quality of Sales Leads

---

Author:
Jordi SOLÉ CASARAMONA

Supervisors:
Xavier LLORENS LATORRE
Mariano YAGÜEZ INSA

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

August 27, 2020

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

Apply Machine Learning in the Company to Predict the Quality of Sales Leads

by Jordi SOLÉ CASARAMONA

Many organizations are still driven by intuition and experience-based decision making. With this type of decision-making, problems such as human bias, loss of experienced workers, and the reluctance to use more sophisticated information systems can be a severe problem. With the arrival of the era of data, companies have at their disposal more information than never before, but not many know how to use this resource to its full potential. In this work, we are going to develop a data science pipeline to predict the quality of the sales leads for the EMEA 3D sales department in HP, a project that aims to enhance the transition to a data-driven decision-making organization.

In order to solve this problem, the developed pipeline was focused on two tasks. The first, involved developing a web scraping tool to obtain information not previously available on the company database or that was very time consuming to acquire due to the size of the database, of more than 40,000 leads. And second, the training of a machine learning algorithm to predict a score quality together with an explainability of the main features of the decision for every lead.

The result of this process greatly impacted the business, all the knowledge was kept always in the company inside the machine learning model, and the explanations of each decision are making gain confidence in the model. Furthermore, the sales team used the score to make more data-driven decisions and save time by prioritizing the best quality leads. The accuracy of the trained Extreme Gradient Boosting algorithm to do the predictions proved to be a 13.45% improvement over the baseline model with a total accuracy of 0.94282 when tested on the test set.

Lastly, all these tasks were put together as a pipeline and uploaded to a server inside HP to execute the process automatically every day with minimal human intervention. The pipeline developed proved to give very positive results for the organization and further developments are being made to enhance the results.

Acknowledgements

First of all, I would like to thank my tutor inside HP, Xavier Llorens Latorre, that has guided me through the whole process of understanding the problem and building a solution for it with his wide domain knowledge of the sales process. Also, thank the EMEA 3D ISR sales team for their feedback to improve the feature engineering and predictions, especially Thibault Rames. Likewise, I would like to thank the EMEA 3D head of sales, Emilio C. Juarez to let me use the organization data to build the machine learning pipeline for this project.

Lastly, thank professor Mariano Yagüez Insa for his advice at the beginning of the process.

Contents

Chapter 1

Introduction

1.1 Motivation

The sales process has always been a human to human interaction for many centuries. Where the seller tries to capture the buyer's needs and attempts to solve them with the product offered while capturing the maximum value for the company. Nowadays not much has changed, this process is done mostly through the internet but most of the time it boils down to human to human interaction, especially when dealing with large transactions. The main difference now is that companies track every little movement that the potential customer makes. This entails the creation of abundant information that many companies don't know how to use. The usage of this data could enhance decision making and minimize the costs of the company, gaining a competitive advantage in the process.

The use of Data Science tools in non-technical departments, especially in big and well-established companies, is difficult. Most of their sales process is still guided by the intuition of the salesmen on a particular trace of information from the potential customer. This information is called lead, and they usually contain personal details about the customer, such as, what company is he working for, what role is he in, among others. The intuition gained through experience is used by the salesman to determine what leads have a high probability to go further in the sales funnel. Because this intuition is not stored and it is difficult to pass to others, every time the salesman is replaced this knowledge gets lost. Even more when considering that the average contract length for this position in HP (called Inside Sales Representatives or ISR's for short) is less than 2 years for this group.

Furthermore, the salesman of the given company is not the only one that is after the customer, other salesmen from competing companies are usually after the same deal too. Hence, it is important to go through the different leads and prioritize those that have a higher chance of success before the competition.

Due to all the above, it is clear that losing information and knowledge is not an option, and that companies should embrace ways to use the full potential of the data they have. Subsequently, a bigger investment in data-related profiles could lead to an overall improvement in the sales and marketing process. This will support the transition from individual small intuition-based decisions, to an algorithmic decision making using previous information from every other ISR in the company. Creating an organization-wide knowledge that even with the replacement of the salesmen, could never be lost. This will make the expertise reside always inside the organization.

1.2 Problem Statement

The main problems in sales teams are: the high rotation of personnel, their decisions based on intuition, and their division in multiple sub-regions that have different needs, creating information silos between regions.

In this project, we aim to focus on the prediction of the status for each one of the leads from HP's 3D Multi Jet Fusion (MJF) printers in the European market, as well as to minimize the salesman's decisions errors.

There are 4 possible status of leads:

- **Qualified**: When the lead is good to pass the filter and go further in the sales funnel. The customer shows potential buying interest in the product. Approximately 2,500 leads (6.17%).
- **Nurture**: When the lead is good but it is not ready to go to the next sale stage just yet. This could be due to customer budget constrains or because they are waiting for new material developments. Nurture leads are set to be recontacted after a period of time that ranges from 6 months to 2 years. Approximately 3,500 leads (8.64%).
- **Closed**: This is where the leads that didn't have a good quality nor entered any of the two previous states reside. The customer just requested information without any intention of buying the machine. Approximately 30,000 leads (74.08%).
- **New**: In this state, we find the newest leads that have not been classified in any of the definitive above states just yet. These are the leads that the ISR has to decide in what state they belong in by getting information about the company and their interest in HP's product. These "New" leads are the leads that the developed machine learning algorithm will tell us in what category they will most likely wind up in. Approximately 4,500 leads (11.11%).

The size of the dataset is currently more than 40,500 leads from the fiscal year 2016 quarter 1 to the fiscal year 2020 quarter 4. On average, every week 257 new leads are entered into the system. In the following sections, we will see the description of the information from every lead.

Note that these MJF printers have a range of prices from 150,000 to 450,000 USD. And thus, due to it being a high priced product and only B2B, the number of leads in the system is not as high as in other company divisions with cheaper products such as personal computers. Also, the 3D Printing division started in 2016.

It is in these "New" leads, that had not been assigned to any state, where the machine learning algorithm is going to be used. Because we want the output to be easily understood by anyone, regardless of their background, the output score must be trouble-free to understand. To do so, the distinction between Good and Bad leads was made. Where good leads are leads that have been assigned to Qualified or Nurture states, and bad leads are those that belong in Closed state due to their null capacity of generating a deal.

Also important is the fact that the score for every lead won't be just a binary score, but instead, a score that ranges from 0 to 1 in order to differentiate the leads with a

score of 0.9 with the ones with 0.6. Thereby, we could use this more detailed score to build a priority list of leads to be contacted based on their score.

Another problem that we will have to face is that because there are only 6,000 leads in either Qualified and Nurture states (Good leads) versus the 30,000 Closed leads (Bad leads) in the system, an imbalanced classification problem arises.

Moreover, and due to the manager's request, the developed algorithm has to have the ability to explain the decisions for each individual lead. This is done to help the team better understand the algorithm's decisions and give the capability to the salesman to learn from the algorithm and improve their decision making and trust to the model.

In addition, due to this information being dynamic, especially the activities performed to the leads, this pipeline needs to be executed in an automated daily. This is because the score can change after performing, for example, an activity such as a phone call with the customer.

And lastly, while the pipeline was developed, the 3D department in HP changed its CRM (Customer Relationship Management) from Dynamics to Salesforce. This meant that the column names were changed, some fields used by the algorithm where deleted, and new ones created. Thus, the implementation had to be revised again to adjust the pipeline to the change of CRM.

1.3 Related Work

Applying machine learning in sales is becoming an important topic, especially sales prediction inside the companies. These enterprises are transitioning from traditional forecasting techniques to more sophisticated machine learning algorithms to improve their predictive analytics and sales forecasting (Cheriyan et al., 2018). In other analysis (Korolev and Ruegg, 2015) authors claimed that Boosted Tree algorithms like Extreme Gradient Boosting is state-of-the-art for many sales prediction problems. Moreover, other publications declare that research has shown that companies perform better when they apply data-driven decision-making instead of intuition-based decisions, and drives decisions away from human biases (Provost and Fawcett, 2013). Some lines of work investigate how machine learning explainability on business is used to help domain experts to iteratively evaluate and update their beliefs using methods such as EXPLAIN and IME (Bohanec, Borstnar, and Robnik-Sikonja, 2017).

1.4 Objectives

The aim of this work is to successfully predict a score for each lead to help the salesman know if the lead will become good (Nurture or Qualified) or bad (Closed). This will involve getting extra information from companies by using web scraping techniques to have a better knowledge of their company products and size. Furthermore, understand the main reason behind the good leads and, at the same time, yield an explanation for the ISR's to understand the score outputted by the algorithm. Lastly, automate the processes by building a pipeline so it can be executed every day with minor human intervention.

1.5 Report layout

To easily follow this paper, the report layout has been divided in 6 chapters:

- Chapter 2: Where we will explain in detail some of the methodologies used in this project.
- Chapter 3: In which the data processing will be explained. From the features that compose the different datasets to the cleaning done to them, as well as the feature engineering developed and encoding used.
- Chapter 4: On this section we will go over the experiments and results obtained with the different machine learning algorithms used. By looking at the validation metrics, we will choose the best algorithm to perform the predictions.
- Chapter 5: We will go over the details on how the pipeline works, their inner processes, and its automation. Moreover, we will expose the benefits that this project has bring to HP.
- Chapter 6: And lastly, the conclusions of the future work that could be done to further enhance the project.

Chapter 2

Background information

In this chapter, we aim to provide the background information to achieve a better understanding of the concepts and techniques used in the project.

2.1 Web scraping

Web scraping is a tool used to extract information from public websites using a web browser. These processes consist of a spider or web crawler designed to obtain data automatically without any human interaction. This code retrieves an HTML of the web page or from the desired part of the page where the information resides. Then, using dedicated packages like *BeautifulSoup* the wanted information can be cut out from all the HTML web page code.

One of the main purposes of web scraping is to build a database and to perform data analysis with the retrieved information for better decision-making with extra information that we didn't have before. While this practice is not illegal it is not considered legal, and many companies such as LinkedIn try to protect their most valuable asset, their data, using advanced techniques. The company even went as far as suing 100 anonymous scrapers, but the US court's decision was that data scraping of publicly available information does not constitute a violation of the CFAA (Computer Fraud and Abuse Act), and hence, it is not considered a criminal offense nor crime.

2.2 Tree based algorithms

Tree-based algorithms are a type of supervised learning techniques that are very popular due to its high performance and interpretability. Because their building blocks are decision trees, we can understand each of their decisions by looking at their nodes and can perform both classification and regression tasks. In this paper, we are going to focus on two types of tree-based algorithms to better understand how they work.

2.2.1 Random Forest

Random Forest algorithm is a type of Bootstrapping aggregation or Parallel Ensamble Learning. In this approach, we grow multiple decision trees independently from each other and decide the output decision by performing the average of all the resulting decision trees when we are facing a regression problem, or takes the majority vote when the problem is a classification.

If the number of the trees in the forest, N, is large, it makes the overall forest more

robust to the variability of the output, giving the effects of K-fold cross-validation. Its performance improvement is due to the reduction of the variance of the classifier while maintaining its bias. The algorithm is described as follows:

1. Samples from the training set, X, are taken randomly but with replacement in smaller subsets x_n.
2. Grow a full tree using, for example, the ID3 algorithm (Iterative Dichotomiser 3) from (Quinlan, 1986) algorithm with no pruning. When splitting, we select $d < D$, where D is the total input variables and d a subset of D.
3. Compute the Information Gain and split the node to reduce the entropy within the random subset of samples x_n.
4. Repeat the process for all the subsets x_n to build the forest.

This algorithm is widely used because it has many benefits such as:

- It is interpretable. Humans can understand the decisions made by the algorithm because the variables and the values of the split are visible in the nodes. The model also outputs the importance of the variables by counting the attribute used to do the splitting as being it more important.
- Can perform both classification and regression tasks.
- Easily handles irrelevant attributes by imposing Gain=0 to them.
- Can handle missing data.
- Very fast at testing time: O(depth).

While the main drawback of Random Forest would be that because its greedy approach, trees may not find the best configuration to fit the data.

2.2.2 Extreme Gradient Boosting

Extreme Gradient Boosting algorithm or XGBoost algorithm is a type of Boosting or Sequential Ensamble Learning where each component in the aggregation depends on all the others. By using this method, the algorithm is capable of correcting the errors made by the model in the previous iteration. The XGBoost algorithm was developed by (Chen and Guestrin, 2016) and, as well as the Random Forest algorithm, XGBoost works for regression and classification problems.

But what makes Extreme Gradient Boosting one of the most used algorithms in Kaggle competitions and state-of-the-art in some problems is its features:

- Extreme Gradient Boosting improves the performance of Gradient Boosting because it uses second-order gradients and regularization techniques like Lasso and Ridge.
- It uses parallelization for sequential tree building. Its training time is far lower than algorithms like Random Forest, this is one of the keys to why it has become such a popular model.
- The algorithm is built to optimize the resources of the hardware, it has cache awareness and "out-of-core" computing.
- It can naturally handle sparsity.

2.3 Interpretability and Explainability

Many times interpretability and explainability are confused, nevertheless, they are different concepts. Interpretability can be defined to be the degree of white box, meaning in which extend we can explain the mechanism used to go from input to output, and how will the output change when the input is tweaked. Whereas explainability is how you can explain the model in human terms, even if the model is not interpretable.

One of the main benefits of using decision trees family algorithms is that the decisions made by these models can be explained and understood by a human, due to its splits using axis-orthogonal hyperplanes. And this is very important because now you could explain the algorithm to non-technical profiles and since the model is capable of being explained, the trust in it grows as now we can explain why did the algorithm output a certain score. Furthermore, with these explanations, we are capable of enhancing human decision making and even correct errors or biases of the models as we can see the decisions of the algorithm.

Libraries like LIME (Local Interpretable Model-Agnostic Explanations) from (Ribeiro, Singh, and Guestrin, 2016) help us better understand the decision of the algorithm, and because it is model-agnostic, it is independent of the type of method used. LIME provides what its called local model interpretability, meaning it explains individual predictions by approximating the model from the neighborhood of the prediction. It tweaks a single data sample by changing the feature values and observes the resulting impact on the output, developing a model approximations of the black box that is the algorithm. Hence, LIME helps grow trust in the model since for every prediction we can explain the "reason why" of the prediction.

Chapter 3

Data Prepossessing

In this chapter, we are going to firstly explain the data sources behind this project to fully understand the features of the final dataset. Secondly, the processes to clean user-entered data. And lastly, the feature engineering developed using domain knowledge aiming to improve the performance of machine learning algorithms.

Disclaimer: The data used for this project is highly sensitive and as part of the confidentiality agreement signed with HP, only minimal amounts of data with no customer details can be taken out of the company nor be externally used. As a result, no data exploration can be publicly shown.

3.1 Data sets

The databases used in this project to build the final dataset fed to the algorithm can be divided in two groups:

3.1.1 System original information

This data is extracted manually with csv files from the corporate *Salesforce* CRM from 3 different tables:

- **Lead database:** Here we can find personal information of the customer: name, country, company, the position of the contact inside the company, when the lead was created or modified, what type of machine are they interested in, etc.

  Every lead has a unique primary key called Response-Id that is a sequential number in ascending order, thus, it is related with the time when the lead was created and with the similar leads created on a given campaign. In this table of the database we have the label to predict, this is the status (Good or Bad lead) in which the lead will most probably wind up (see section 1.2).

  We can also find information related to what sector the company is in, email, and phone of the contact. But, because this is information that has to be introduced manually, it is sometimes missing or liable to have some errors.

- **Activities database:** It shows the tasks that the sales team performed to a given lead. These can be a call, custom e-mail, massive e-mails, etc. With this table we are able to track how many touches did we have with each customer, giving us a good indication of customer engagement. Furthermore, the date when those activities were performed is also stored together with some comments that the ISR introduces manually.

- **Campaigns database:** In what trade shows, events, webinars or other marketing campaigns have the lead been present. All the events where the customer went are kept, in conjunction with the date.

3.1.2 Scraping information

This is the extra information that the web scraping algorithm gets from the web pages of *LinkedIn* and *Glassdoor*. By doing this, we are saving salesmen time looking for detailed information about the new companies in those same portals. Furthermore, with extra information about the companies, the salesman can have better decision making and also the machine learning algorithm can use this data to improve its outcome. More details on the scraping process in the Pipeline description in chapter 5, where we will explain how this was achieved.

- **LinkedIn scraping:** LinkedIn could find approximately 79.5% of the companies listed in the Lead database. In those companies found on the web, relevant information such as the number of employees, headquarters, foundation year, contact phone, products, and vertical of the company was obtained. In addition, 45% of them had extra commentaries about their main products, but those descriptions were not always in English and needed to be translated. The scraping also takes the LinkedIn company name to later check if the found information was from the correct company.

- **Glassdoor scraping:** From Glassdoor the information scraped was: the reported income of the companies, as well as the number of employees and their company vertical. Fewer companies were found using this web and the details were not as accurate as the ones found in LinkedIn. Thus, if the information of the company was found in both web pages, the data from LinkedIn was the one used.

Using the scraped information for more than 40,000 leads, the total number of leads segmented, meaning the industry of the companies scraped, went from a 31.83% to a 84.26%. This was achieved by combining the results from HP's CRM, and the scrapings of LinkedIn, and Glassdoor. Because of the huge number of lead that needed to be scraped at the beginning, multiple scrapings were run in parallel for 3 weeks to obtain the past lead information.

These scraping information gave HP very important findings. We improved the knowledge that both Sales and Marketing teams can use for better targeting the industries with the highest chance of having good leads and better analyze the market size and status. Regarding the 16.14% of leads whose segment not found using web scraping nor in the CRM data, are thought to be companies that are not big enough or not technological enough to be on the web. This also gives us the information that they might not have the technology or the size to buy a 3D printer.

Regarding the full dataset used for machine learning, the data was joined together in a first normal form with the Response-Id as the primary key for joining the data sets from the system original information of HP's CRM. Because of what we were looking for in the scraped web pages were companies, the company name act as the primary key to join both tables obtained from the scraping information of LinkedIn and Glassdoor. Lastly, the resulting tables from the CRM and the extra scraped information were joined using the company name as the primary key.

3.2 Data Cleaning

Because we are dealing with data introduced manually by sales personnel, it is keen on some errors. Furthermore, the scraping process had some cleaning to be made. In this section, we unfold the data cleaning processes that take place in the pipeline.

The first modification is to generalize the different nine states that the Response can have from the company CRM to the 4 states we described in section 1.2. This is done because there is fine detail coming from the CRM lead state that it is not needed for the machine learning prediction. Recall that we just want to know for every new lead, the probability in which it will be Qualified or Nurture (good lead), or a Close lead (bad lead).

Another important cleaning performed to the data was in the "Phone number" field. This is because these data was introduced manually by the ISR's and because we are dealing with all EMEA countries (Europe, Middle East, and Africa) the telephone numbers can be very different, with mismatching digit length and with sometimes a plus sign in the front, or dashes between numbers. In order to standardize the phones, all punctuation mark was removed.

From the scraped LinkedIn and Glassdoor information, we obtained the segment in which the company operates in. But, these web pages use different terminology than HP's CRM. For example, SEAT S.A. is in the segment of "Mobility and Transportation" for HP, "Automotive" for LinkedIn, and "Manufacturer of transportation equipment" for Glassdoor. Hence, a cleaning transforming the scraped segments to HP's had to be made. This was done by duplicating the columns of their respective segment information and replacing it by and approximation of HP's from an excel file to translate the segments to HP segments. By doing this, we kept both the original LinkedIn and Glassdoor segment and its HP's equivalent, standardizing the data. The HP segments are more general, non-detailed segment.

Also from the scraped information, we obtained the two columns that needed cleaning because they were intervals. These are company income and company size. Because the algorithm can not work with interval due to it being strings, these had to be transformed to numbers without altering their ordinal nature. To carry out this task, the upper bound was assigned to the interval, and in this way, the distances between intervals were preserved and not hashed to random numbers. For example, a company with "1001 - 5000 employees" was assigned to 5000.

3.3 Feature Engineering

Using the domain knowledge gained with more than one year of experience in the sales team, many feature engineering fields were developed to help the machine learning algorithm in their classification task. The extra fields created were:

- **Count of campaigns:** This field captures how many times has a lead come to different campaigns with the same Response-Id. This is done by a groupby count by the Response-Id.
- **Count of companies:** Shows how many times has a company entered the CRM even with different Response-Ids. Another groupby count is used but now on the lower case company name field.

- **Phone or Landline:** It is very important to know if the phone that we received from the customer is actually a mobile phone or a landline number, this can show the customer engagement in the product. To do this, the Wikipedia page of "List of mobile telephone prefixes by country" (Wikipedia contributors, 2020) was used to identify the two types of lines using the previously cleaned phone number field.

- **3D company:** Using the products section from the LinkedIn scraped information, we created a boolean field to check if "3D" was among one of the company products. Adding to this, the field is also positive if the name of the company incorporates the string "3D" in the company name.

- **HP + LinkedIn + Glassdoor Vertical:** Because we have the information of which vertical the company operates from 3 different sources in the best of the cases we need to merge these fields. Because the data entered by the salesman is said to be the most accurate, it has the highest priority. Then, LinkedIn and Glassdoor equivalent HP segment, by this priority order. If we follow the example in 3.2, SEAT S.A. would have the value of "Mobility and Transportation".

- **LinkedIn + Glassdoor Company Size:** The same goes this time for the company size. This is not an original field from the corporate CRM and hence, the sources are LinkedIn and Glassdoor, with the first one having the priority when both are found.

- **Phone or Email present:** This created field informs if there is any contact phone or email with the customer in a boolean fashion. As we will see later on, this proved to be one of the most important fields of them all, because if you cannot contact your customer, the lead will, most certainly, be a bad lead.

- **OHE for Activities:** A One Hot Encoding (OHE) was developed for every activity performed to a lead because a lead can have multiple activities assigned. Thus, if there were multiple activities with the same Response-Id, the One Hot Encoding of the different encoding would be merged and summed. By doing these, we could inform the model of the number of activities of every type performed to a certain Response-Id, even when activities are performed more than once.

- **OHE for Camapigns:** The same as the previous field was done but this time with the different types of campaigns. Since a lead can come to different campaigns, a One Hot Encoding was created for every campaign and then merged and summed by the primary key of the Response-Id.

- **Appearances of important words:** From the LinkedIn web page we scraped 45% of the times the products or services the company was providing to their customers in the form of a list. To retrieve the most important products from these lists, we used TF (Term Frequency) count to see the most repeated words in the corpus of the combined lists. The following step was selecting the most frequent product from the list of products for each company. In doing so, the list of products was transformed into the most frequent (in the corpus) product for every row.

- **Fiscal Year Quarter:** With the creation date of the Lead, we translated it to Fiscal Year. This is widely used in sales, where bonuses are assigned in a quarterly manner and hence, some correlation could exist since the salesman is more prone to pass a regular lead as a Qualified lead (good lead) to increase their numbers. It is also during the last month of the quarter when the sales personnel push harder to close the deals.

3.4 Label Encoding

To deal with fields that contained strings, the *Sklearn Label Encoder* function was used. This encoder transforms strings into integers, starting from 1 with the first string encountered in a given column, 2 for the second one, and so on. To maintain the encoding constant, the whole data was ordered in descending creation date. By doing so, we make sure that the encoders were always the same for previously seen data, and when new strings arrive, the encode would assign a non used integer to encode the new strings. This approach was selected because, in this way, the encoder never changes, unlike with hashing functions. This will be essential when using machine learning to predict daily the score for the "New" status leads.

Chapter 4

Experiments & Results

In this section, the different experiments and results from the pipeline will be exposed. The chosen machine learning algorithms to be tested in the experiments were the Random Forest and the Extreme Gradient Boosting models. This models where selected because of the restriction set by the team manager, stating that he wanted an algorithm in which their decisions could be explained to the sales team in human non-technical terms (see section 1.2). Hence, tree-based algorithms were chosen to develop this task.

The dataset was divided into three different sets because the performance of both Random Forest and XGBoost algorithms had to be tested and the best algorithm selected. The sets only contain Good and Bad leads. The leads still on "New" status had been left out of this set because those leads are the ones that have to be predicted with the algorithm trained on the hole previous dataset of good and bad leads.

- Training set size: (25340, 58)
- Validation set size: (7276, 58)
- Testing set size: (3585, 58)

Note that these numbers change due to leads being introduced in the CRM daily. Furthermore, the status of leads is dynamic and leads can go, for example, from Nurture to Closed.

4.1 Validation Metrics

Let's remember what were the main problems for our dataset. First, because the Good leads (Nurture and Qualified status) are just 16.9% of the leads with a total number of more than 40,000, we are facing an unbalanced dataset. And second, because of the previous point, leads that are good are very valuable and we can not afford to classify a good lead as a bad one. Hence, the Recall or True Positive Rate (TPR) formula 4.1 is a high priority for this project.

$$TPR = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{4.1}$$

When facing a binary classification problem, the most used validation metrics are commonly Accuracy and the Area Under Curve or AUC. Since AUC works best for skewed distributions, this metric was chosen to be the main validation metric. Furthermore, AUC incorporates the True Positive Rate that we said was very important for this problem.

The AUC depends on the ROC curve or Receiver Operating Characteristic curve, in fact, the AUC is the integral from [0,1] or "area under the curve" of the ROC. This curve is generated by plotting the True Positive Rate (formula 4.1) against the False Positive Rate (formula 4.2) on different operating points or threshold value.

$$FPR = \frac{\text{False Positive}}{\text{False Positive} + \text{True Negative}} \tag{4.2}$$

Hence, the AUC can be understood as a way to tell to what degree the model is capable of distinguishing between classes. A perfect model with an AUC of 1 is a model that can perfectly distinguish between the two classes, and this is what we want our machine learning algorithm to optimize.

Because it was considered that the Accuracy metric was also important to optimize, we ended up using a multi-metric evaluation from *sklearn* package *make_scorer* that was used in the GridSearchCV and refitted on the AUC to achieve the best cross-validated AUC score.

Apart from the AUC and Accuracy metrics, we are going to use the F1 Score metric that is the harmonic mean of the precision and recall (see formula 4.3) as a validation metric to test the performance of the modes. This metric takes into account both false positives and false negatives, or the Type I and Type II errors, from the second diagonal of the confusion matrix. Thus, F1 Score is very useful when classes don't have the same number of samples, and that is the case for our problem.

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})} \tag{4.3}$$

4.2 Baseline

The baseline for the machine learning was established to be the ratio of bad lead of the dataset, that is 83.1%. This means that if we build a naive classifier that every lead was assigned to be a bad lead, the estimated classifier accuracy would be 83.1%. Hence, this is the baseline accuracy that our model will have to beat.

4.3 Random Forest

For the Random Forest algorithms (RF) two approaches were tried: First, the Balanced Random Forest Classifier algorithm from the *imbalanced-learn* package as we are dealing with an unbalanced dataset. According to the documentation it "*randomly under-samples each bootstrap sample to balance it*" (G. Lemaitre, 2017). And second, we used the standard Random Forest Classifier from *sklearn* to test the effectiveness of the Balanced version. Both algorithms had the same configuration, with a Cross-Validation of 5 fold and the following Search Grid parameters:

- 'max_depth': [5, 25, 50]
- 'n_estimators': [1000, 2000]

After training the algorithms for almost an hour with the train set (25,340 leads), they were tested on the validation set (7,276 leads) using SearchGridCV with 5 fold, the best configurations were selected, and both models were testes against different evaluation metrics.

TABLE 4.1: Evaluation metrics comparing the performance of Balanced and standard Random Forest on the validation set.

Algorithm	Accuracy	AUC	F1 Score
Balanced Random Forest	0.88414	0.956787	0.739574
Random Forest	0.92207	0.961829	0.759236

As we can see in table 4.1, for all the different metrics the Balanced Random Forest performed worst than the Random Forest, especially in the Accuracy and F1 Score. The standard random forest achieved a 4.29% higher accuracy and a 2.66% better F1 SCore with respect the balanced version of the model. Regarding the AUC score, both algorithms achieved a similar score.

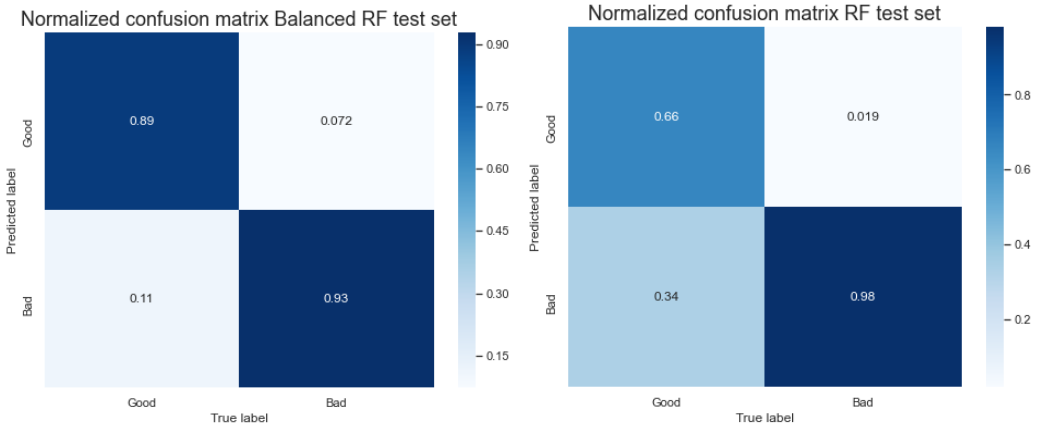

FIGURE 4.1: Column normalized confusion matrices yield by the Balanced Random Forest (left) and Random Forest (right) on the validation set.

Looking at figure 4.1 we can see that the confusion matrix yield by the Balanced version of the Random Forest model (left) obtained very good results, as most of the values are on the first diagonal of the matrix. Because it is column normalized, we can instantaneously see that the True Positive Rate is 89%, and that the False Positive Rate is just 7.2%.

When looking at the confusion matrix for the standard Random Forest (right), we see that the first diagonal is not perfectly distinguishable as the Balanced version, but the False Positive Rate rises to a 98%. Whereas, the True Positive Rate is 66%, far lower than the balanced version. The high accuracy achieved by the standard Random Forest model can be explained because the majority class, with 83.1% of the leads being bad leads, was classified 98% of the time correctly. When comparing the two matrices, the non-balanced version achieves a worst True Positive Rate at 66% and a False Negative Rate of 34%, 23 points worst in both metrics than in the balanced version. As previously said, this is explained because the standard version didn't have any measure of dealing with the unbalanced dataset and performed very good on the bad leads classification.

4.4 Extreme Gradient Boosting

For the Extreme Gradient Boosting a Search Grid was also used with a Cross-Validation of 5 fold with the following parameters:

- 'objective': [binary:logistic]
- 'learning_rate': [0.05, 0.1, 0.2]
- 'n_estimators': [1000, 2000]
- 'scale_pos_weight': [2, 5]

Notice the parameter 'scale_pos_weight'. This is the ratio of the number of negative class to the positive class used to balance imbalanced datasets like the one we are facing. This proportion of our problem is around 4.6931. Hence, this hyperparameter being at 5 for example, gives 5 times more weight to the positive class, meaning that the misclassification error of a positive example has a 5 fold higher cost. Same as with the Random Forest, the XGBoost was tested with the three different evaluation metrics.

TABLE 4.2: Evaluation metrics comparing for the Extreme Gradient Boosting algorithm on the validation set.

Algorithm	Accuracy	AUC	F1 Score
Extreme Gradient Boosting	0.93664	0.970280	0.831567

As shown in the table 4.2, the Extreme Gradient Boosting seem to perform better than the previous Random Forest algorithms even with the selected model having 'scale_pos_weight'=5 as a hyperparameter.

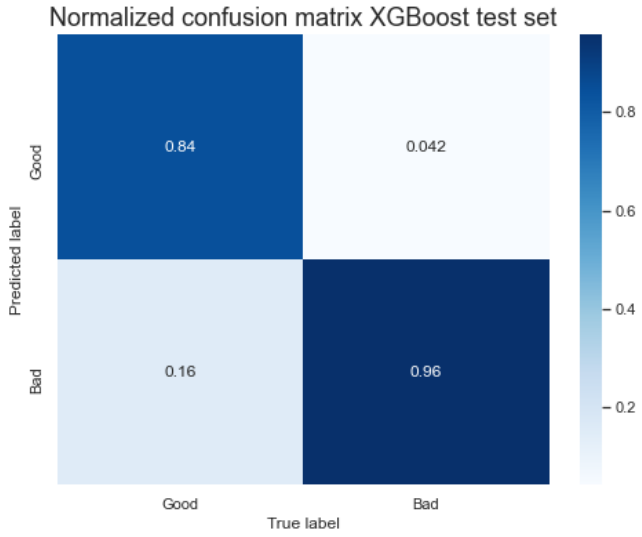

FIGURE 4.2: Column normalized confusion matrix yield by the Extreme Gradient Boosting on the validation set.

From the above figure 4.2 we can see that the confusion matrix has a True Positive Rate of 84% and a True Positive Rate of 96%. The type I and type II errors, especially

the first one, are very low. The False Positive Rate yield for this model is 4.2%. This gives us a better understanding of the high AUC score obtained in table 4.2. When comparing this confusion matrix with the previous models, we can see that the XGBoost classified correctly the vast majority of the negative examples and performed almost as well as the Balanced Random Forest with the classification of the positive class. Because of this, this algorithm has the highest accuracy of all models tested.

4.5 Model selection

After using the training and validation sets to train the model and test it against different validation metrics, we have to pick the best one. As a result, we need to compare the three previous classifiers, starting for the main validation metric of the scoring, the AUC.

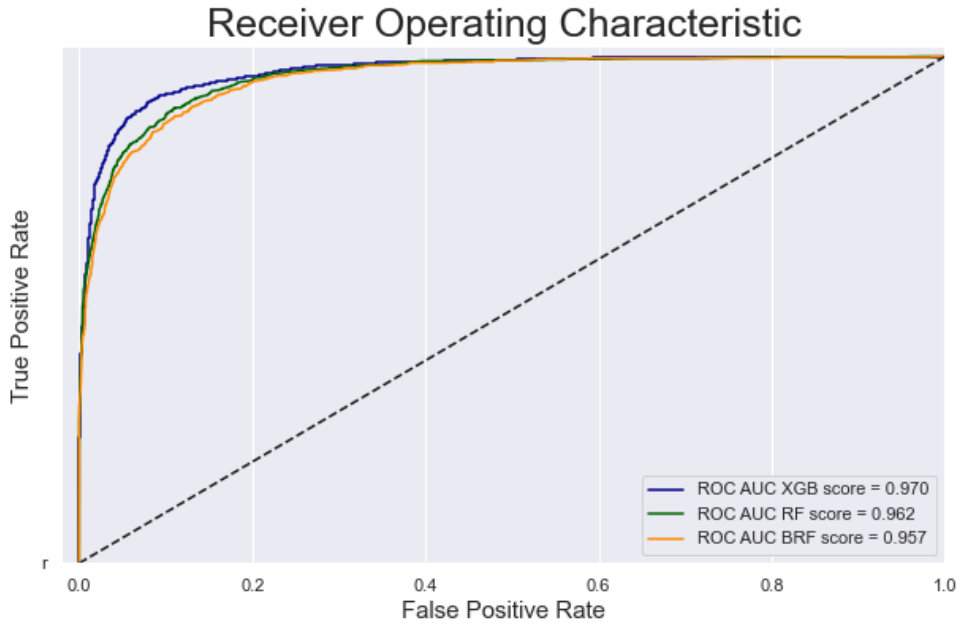

FIGURE 4.3: Reciver Operating Characteristic curve for Balanced Random Forest, Random Forest, and XGBoost clasifiers.

From the above figure 4.3 we can see the different ROC curves for the three previous models tested. Clearly, the line in blue is the ROC that has a bigger Area Under the Curve, this line belongs to the XGBoost classifier that obtains an AUC of 0.97028. This score is not far from 1, that would be the perfect classifier, and is a big improvement over the 0.5 line in black that is the random choice. Almost a point below is the AUC's of both Random Forest classifiers.

Now let's compare all the validation metrics used with all the three previous models.

TABLE 4.3: Evaluation metrics comparing the performance of the baseline model, the Balanced Random Forest, standard Random Forest, and Extreme Gradient Boosting on the validation set.

Algorithm	Accuracy	AUC	F1 Score
Baseline model	0.83100	-	-
Balanced Random Forest	0.88414	0.956787	0.739574
Random Forest	0.92207	0.961829	0.759236
Extreme Gradient Boosting	**0.93664**	**0.970280**	**0.831567**

Following the table 4.3, it can clearly be seen at first sight that the XGBoost algorithm outperforms all of the two types of Random Forest algorithms used in this work and the baseline model. Let's go over each of the validation metrics.

- **Accuracy:** The highest accuracy is achieved by the XGBoost classifier and it is 0.01457 points higher or an increase of 1.58% with respect to the Random Forest accuracy, that is the second-best model in this aspect. When looking back to the baseline accuracy in section 4.2, the system accuracy was set to be 83.1%. With XGBoost we have increased the accuracy by 12.71% or 0.10564 points. The accuracy of the balanced version of the Random Forest falls far behind at 0.88318.

- **AUC:** The AUC's of the different classifiers are very similar. But again, the Extreme Gradient Boosting classifier has the highest score with an AUC of 0.97028, a 0.9% increase respect the second-best AUC score of the standard Random Forest. This means that XGBoost is the classifier that can better separate the two different classes of the problem.

- **F1 Score:** Here is where the XGBoost algorithm exceeds all the other algorithms by far. The F1 Score of 0.831567 can be explained by the small Type I and Type II errors present in the confusion matrix in figure 4.2 while having a high number of True Positive examples. XGBoost outperforms the second-best F1 Score by a 9.33%.

Hence, it is clear that the best machine learning algorithm to use against the test set is the Extreme Gradient Boosting algorithm, which has outperformed the Random Forest models. After validating the model, now with a bigger data set, we will use it with an end goal of outputting the score for every lead in "New" status.

4.6 Test set performance

Now that we have selected the algorithm that the pipeline is going to use, we will train the Extreme Gradient Boosting with both training and validation sets and tested its performance in the test set. By doing this, we will give more samples with which the machine learning algorithm can be trained. Hence, the sets are now:

- Training set size: (32616, 58)

- Test set size: (3585, 58)

The hyperparameters to optimize are the same as the previous Extreme Gradient Boosting in section 4.4. After fitting the model for half an hour, we obtain the following results summarized in table 4.4:

TABLE 4.4: Evaluation metrics for the Extreme Gradient Boosting algorithm on the test set.

Algorithm	Accuracy	AUC	F1 Score
Validation Extreme Gradient Boosting	0.94282	0.973318	0.833469

These results are a slightly improvement from the previous XGBoost tested with the test data. The XGBoost tested against the validation data has achieved an extra 0.00618 better Accuracy, 0.00304 improved AUC and a 0.0016 enhanced F1 Score.

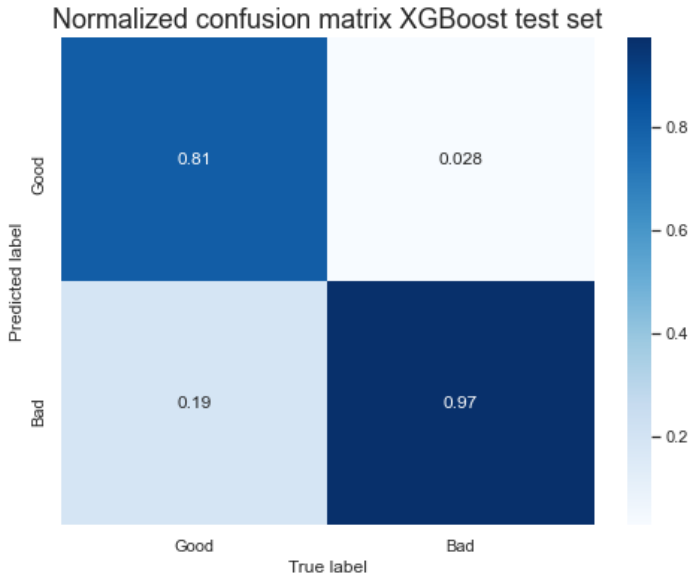

FIGURE 4.4: Column normalized confusion matrix yield by the Extreme Gradient Boosting on the test set.

From the above confusion matrix in figure 4.4 we can see that the increase in accuracy of this last model is due to the True Negative Rate has grown from 0.96 to 0.97, with 2867 of the 2950 negative entries classified correctly. This might not seem much, but because the majority class (bad lead) is 5 times bigger than the minority class (good leads) this increase has made the accuracy grow, even with a decrease of the True Positive Rate from 84% to 81%. Regarding the False Negatives and False Positives rate, this last one has been reduced slightly. Fact that could explain the small increase on F1 Score.

Thus, we can say that with new leads with non defined state, the XGBoost will have a similar performance as the one seen in table 4.4. We say similar because the XGBoost deployed in the pipeline has been trained with the whole dataset, and hence, these metrics from the above table are just an approximation of the real performance. Regardless, the algorithm accuracy is 13.45% higher than the model baseline.

Chapter 5

Pipeline

In this chapter, we will review all the pipeline created after selecting the machine learning model that is going to give us the scoring for each new lead entering the system. Here is where all the bits and pieces explained in earlier sections come together.

5.1 Pipeline description

In the image in 5.1 we can see the full pipeline developed to solve this problem. In it, there are two main parts, the processes, and data from the local computer (green outline) and the processes done in HP's Z8 workstation server with 187 GB of RAM (dark blue outline). Different options were considered to host the pipeline, but since the data that we are dealing with is very confidential and has many customer details, the team wanted the data to be hosted inside HP. That is why the Z8 server was chosen over other platforms such as AWS or Azure.

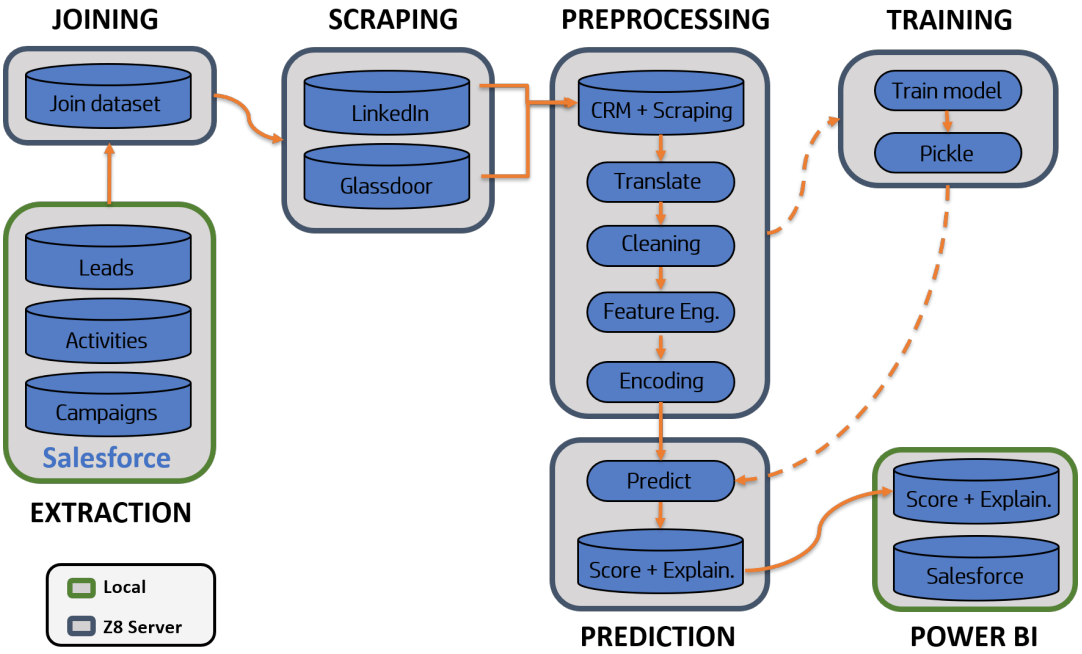

FIGURE 5.1: Schema of the developed pipeline for the project.

Extraction

The process starts with the extraction process in the local computer, where the data

from the corporate CRM has to be downloaded locally and uploaded to the server. This process can not be automated due to Salesforce constraints.

Joining

Inside the Z8 server, a python file with all the code for the pipeline is run every midnight. The code that deals with the joining process resides at "Dataset_Creation.py", receives the different tables of Activities, Leads and Campaigns from Salesforce and joins them together using as the primary key the Response-Id. Once this is done, a new xlsx is generated called "NewData.xlsx" that replaces the previous file. In addition, one backup copy is saved with the same file name followed by a timestamp. This process takes a mean time of 3 minutes and 43 seconds.

Scraping

From the "NewData.xlsx", the LinkedIn and the Glassdoor Scraping compare the company names of the new excel file with the company names they have in their internal files. Those files are called "Linkedin_Sraping.xlsx" for LinkedIn, and "Glassdoor_Sraping.xlsx" for Glassdoor information.

The python code used are "Linkedin_Scraping.py" and "Glassdoor_Scraping.py" to obtain the data described in 3.1.2. After finding the new companies that need to be scraped, a chromedriver is open and navigates to the desired websites to access the requested information. This process is different for the two web scrapers:

- **LinkedIn Scraper:** First, the web crawler looks for the company on the search bar and if the company is found, it navigates to the "about" section of the company to obtain the information. If the company is not found, the scraping continues to the next ones, leaving the not found company with null values. This process takes approximately 30 seconds to scrape one lead, hence it can be a slow process when a new campaign is entered into the system with more than 200 leads. Moreover, you have to consider that LinkedIn has mechanisms to catch web scraping, and hence, the algorithm has to go at a slower pace to resemble human behavior.

  But when massive campaigns take place, introducing more than 500 leads into the system, LinkedIn scraping stops functioning because there is a maximum of searches per day estimated to be around 500. Once this maximum was reached, the user is logged out and captchas would appear in the next login. The workaround for this issue was to create 7 different LinkedIn accounts and, by randomly changing the user account once 300 leads were scraped, we avoided the LinkedIn maximum day searches. In addition to this change of profile, the code also saves the scraped information until the moment in case the code crashed or the internet connection fails when dealing with high amounts of new leads.

- **Glassdoor Scraper:** The scraping performed in this web site consisted of for each company, go to the "Company" section, and search the desired lead there. If the company is present in Glassdoor, the information of the company is displayed right at the top of the company page. Regarding Glassdoor measures to stop web scraping from their portal, we didn't found any issues in more than 5

months. Hence, the web scraping can be much faster, at around 20 seconds per company without such restrictions. Glassdoor scraping also has a mechanism to save the information scraped in case of a rising error.

Once the process is ended, both scraping files are updated with the newfound information, and a backup file for both excels is created.

	Name	Web_Name	Website	Phone	Industry	Company size	Headquarters	Type	Founded	Specialties
0	university of limerick	University of Limerick	http://www.ul.ie/	+353-(0)61-202700\n \n\n Phone...	Higher Education	1,001-5,000 employees	Limerick, munster	Educational Institution	1972	Training, Research, and Education
1	gkn sinter metals gmbh	GKN Powder Metallurgy	http://www.gknpm.com/	None	Mechanical Or Industrial Engineering	5,001-10,000 employees	Auburn Hills, Michigan	Public Company	None	Engineering, Automotive, Industrial, Powder Me...
2	1024colourtech	None	None	None	None	None	None	None	None	None
3	moduleworks	ModuleWorks	http://www.moduleworks.com	None	Computer Software	201-500 employees	None	Privately Held	2003	CAD/CAM Software, 5-Axis, 4-Axis, 3-Axis, Simu...
4	zddc ltd.	Langfeng CEC Dacheng Electronics Co., Ltd.	http://zddc.com.cn	None	None	None	None	None	None	None

FIGURE 5.2: Resulting information from the LinkedIn scraping.

In figure 5.2 we can see the outcome of 5 scraped companies done by the LinkedIn scraper. Here we have the company name in lower case and in the second column the name of the company on LinkedIn. This column is also used to determine if LinkedIn didn't show us the correct company or a company with a similar name.

As seen wit the company "1024colourtech", LinkedIn was not able to find the company, and thus, the fields are filled with "None". In this example, non of the companies had a Speciality in another language different from English, and hence, no translation needed to be done.

In addition, observe the fifth row, where the CRM name and the company name and the LinkedIn company name are very different. But when looking at the company web, it finds the correct company. Because of this, no implementation to compare the similarity of words such as Jacquard distance was developed, trusting that the LinkedIn algorithm will show the best results possible. The total execution time of both scraping processes depends on how many leads are entered into the system.

Preprocessing

Once the extra information from LinkedIn and Glassdoor is stored, they are joined together with the systems CRM information by the company name using the "Create_file_ml.py". Because some of the scraped companies can have strings in non-English languages, a program is run to detect the language of the strings and if they are not in English, it translates them to this language.

This process takes around 10 seconds with a timesleep of 1 seconds to not overload the GoogleTranslator API used to translate. By doing this, we are achieving a more homogeneous and higher integrity data that can be now be used to develop feature engineering in English.

After the translation, the data preprocessing explained in section 3.2 to section 3.4 takes place. The preprocessing, feature engineering, and encoding for the hole dataset take place ins this process. The resulting file named "Results_Verticals_Salesforce.xlsx" is saved along with its buck up file. This process ensures that the data can be feed to the machine learning algorithm to be trained with good and bad leads or to predict the leads in the new status.

Train

The next step is to decide if we want to retrain the algorithm, otherwise, it will use the pickle from previously trained data to predict the probability of new leads to become good or bad leads. Let's say we want to retrain the algorithm. We can do that by setting the parameters "train_xgb" of the pipeline to True. This retraining is done approximately once every week. Now the model will be retrained on all the 36,000 leads, composed by good leads (Nurture and Qualified) and bad leads (Closed).

Because the data is growing by 257 new leads every week, the training data keeps growing and the algorithm improving. As we saw in section 4.5, the XGBoost algorithm is the chosen algorithm to be trained with all the data. After it finishes, it saves the model into a pickle file to be used to do faster predictions, and can be later used as a parameter in the clean_dataset_ml function in "Predict_Qualified_Salesforce.py" to specify the model to use to output the predictions.

Prediction

The predictions are made with leads that have the status "New", hence, these new data have to be preprocessed to be fed to the machine learning algorithm in the same way we did with the good and bad leads. These involve the cleaning, feature engineering, and encoding. Once the process has been completed, we use the pickle file to load the desired XGBoost algorithm that is going to be used to output the score for every lead. After that is done, we use the LIME package to give us the 5 most important attributes to understand why the machine learning output a certain score for that lead.

	ResponseId	Pred Prob	Explain_1	Explain_2	Explain_3	Explain_4	Explain_5
0	3DMCR-R2133833.0	0.556919	('Sub Segment <= 0.00', -0.179191803705065)	('Activ_Reached Calls <= 0.00', -0.17129491289...	('Products > 7.00', 0.16545944511035335)	('Title <= 7.00', -0.12281930359273814)	('Created By <= 0.00', 0.1059972886281962)
1	3DMCR-R2133802.0	0.233707	('Activ_Reached Calls <= 0.00', -0.25980019736...	('Segment <= 0.00', -0.17294034750054657)	('LnkdIn_and_Glsdr_Employees <= 0.00', 0.14235...	('Response Creation Date <= 391.00', 0.1415168...	('Sub Segment <= 0.00', -0.1353097813011282)
2	3DMCR-R2141791.0	0.013039	('Segment <= 0.00', -0.1814711725318605)	('Activ_Email <= 0.00', -0.169479449370327)	('Sub Segment <= 0.00', -0.15862661581594303)	('End Date <= 15.00', 0.1413019847755207)	('Month <= 3.00', 0.13564347097783083)
3	3DMCR-R2173577.0	0.012628	('Activ_Reached Calls <= 0.00', -0.24846848802...	('Sub Segment <= 0.00', -0.18739708326161716)	('Activ_Email <= 0.00', -0.16923402241794677)	('Count_account <= 1.00', 0.14057126597635383)	('Response Creation Date <= 391.00', 0.1129165...
4	3DMCR-R2159328.0	0.007059	('Sub Segment > 11.00', 0.12220823555084215)	('Created By > 8.00', -0.119088548535101)	('Activ_Reached Calls <= 0.00', -0.10775275676...	('Segment > 3.00', 0.10567817570613468)	('Products <= 0.00', -0.09746243245742488)
...	...	...	...	...	...	...	...
41493	3DMCR-R2186951.0	0.005365	NaN	NaN	NaN	NaN	NaN

FIGURE 5.3: Output score of the XGBoost algorithm and the 5 LIME feature explainabilities for different Response-Ids.

From figure 5.3 above we can see the output file of the pipeline. This information is stored in the "Lead_Scores_Machine_Learning.xlsx" file along with a backup. In

this data, we can see the primary key, the Response-Ids, the predicted score, and the 5 explainability columns for each lead.

Notice that columns generated by LIME have positive or negative scores. The explainability features show the top 5 most relevant attributes for the given decision ordered by the absolute score of the given attribute. For example, for the first row, the subsegment was found to be the most important attribute for the decision. When looking at the importance of the field we see that the "Sub Segment" field found was zero or less than zero, meaning that there was not any subsegment assigned to the lead. As a result, the explainability shows a negative score. This is key to understand in the factor that have a positive or a negative impact on the machine learning score.

For the better user experience and to make the team trust the algorithm, the scores for every lead are calculated. This included the leads in good or bad states, that are also predicted. But because the algorithm has been trained with this data, it outputs near 0 results when the lead is in Closed state (bad lead) and 1 when the lead is in Qualified or Nurture state (good lead). With this, all the leads have a score and no nulls will be shown in the system. In practical terms, only the score of the new leads are used for the ISR decision making. The process of generating the score takes less than 20 seconds due to the high speed of the XGBoost.

This whole process, depends largely on the number of data points with which you want to do the approximation, can take up to 4 hours with a good estimation of 500 data points with 5 explainability attributes. This is why for giving the explainability of the 5000 "New" leads, the process needs to be in a server where the procedure is executed every midnight and be ready every morning. Notice that the last row of the figure 5.3 has no explainability features. This is because it is from a lead that is already in a good or bad state. Calculating the explainability for the hole 40,000 leads would take a non-feasible amount of time.

PowerBi

The last step is to retrieve the excel with the score and explainability information that has been executing at midnight in the Z8 server and upload it to PowerBI. Hence, now all the organization can have the quality of the leads using the classification score, the explainability of the score, and the scraped information from every new lead.

5.2 Pipeline automation

For the automation of this pipeline the libraries *schedule* and *time* were used. The code at *main.py* imports all the dependencies needed for the pipeline to work. These are executed as follows:

1. Dataset_Creation.py
2. Linkedin_Scraping.py
3. Glassdoor_Scraping.py

4. Create_file_ml.py
5. Clean_dataset_ml.py

To execute all the process in a scheduled manner, the main.py has a function called "ml_pipeline" that executes all the above functions. The parameters of this function are the:

- train_xgb: Boolean. Used to perform a retrain to the model. When set to true it retrains the XGBoost algorithm with all the data from good and bad leads, leaving out of the training set the leads with the "New" status. Furthermore, if True, the model predictions will be automatically made with the newly retrained model, even if a pickle file is given as a parameter to model_used function.
- model_used: Pickle file that is given as a parameter the decide which pickle model to use for the predictions. By default it is the last file created.
- number_samples: Number of samples to do the linear approximation with LIME. As more samples used, better predictions but the time to do this process for every lead grows.

Using the schedule function, we can specify at what time of the day the process needs to be executed automatically. The code will be executed in a while loop and will only execute when the conditions specified on the schedule function are true.

The time that the process takes to execute depends on the number of leads in "New" status and the new leads that entered the system in which web scraping needs to be performed. Overall, the mean execution time during one week has 2 hours and 17 minutes with a number_samples variable set at 250 and no retraining. This execution time is expected to grow when more points are added to the LIME approximation and when higher numbers of leads will be entering the system due to summer vacations end.

5.3 Achieved improvements

One of the most measurable improvements of this work is the cost reduction that the web scraping gives to the salesman, reducing the time they spend looking for the company online and finding their products. It is estimated that on average, the team members spend less than 3 minutes researching the web when a new previously unknown company enters the system and uses one extra minute to upload the information to the CRM. Hence, the salesmen lose 4 minutes in total to fill the new company information. Recall that this was not the case for all the companies, only 31.83% of the leads had this information before the web scraping was performed on this project.

Hence, to calculate the cost reduction we need the cost per minute of the salesperson. This is estimated to be 0.273€/min with the information given by HP's human resources. Thus, the implementation of the automatic web scraping is saving 4 minutes for a mean 257 new lead each week, with a total cost of 280.87€/week or a saving 17 hours research of work each week. This number might not seem much, but in the long run, this information can save up to 1,123.50€ each month (72 hours of work) or a very respectable 13,481.97€ per year (891 hours of work).

Moving to improvements not as easy to quantify as the last one, we have the score for every lead and the explainability of each decision. With the feedback given by the ISR's of the team, we can say that the implementation has been successful and has added important information to the CRM information.

These benefits can take the form of a priority list build with the ordered scores of the machine learning predictions that the salesman uses to contact the best scoring leads first. Using this new feature, the team prioritizes the calling for the best leads, and for the bad leads, they could only send an email due to its low probability to become a deal. This saves the salesman time of going through all the leads and finding the best ones (difficult to measure) and ensures that the leads with potential are contacted. Think it this way, now the new arriving mail is classified to junk or business mail automatically without having to go through all the reading.

The high accuracy of the developed score will lead to better decision making and a reduction of human decision bias. Because of this high accuracy and AUC seen in previous, less good leads are expected to be neglected. In addition, the ISR's can use the score for the new leads to see what is the quality of these leads for each country or marketing campaign, having now an indication of the business potential.

Regarding the explainability features, the salesmen are now able to use this attribute to gain trust in the algorithm, and for the new hires, they can be helped in the learning process.

And lastly, all this information is shared online within all the organization and all the new information from the scraping and machine learning scores are incorporated in the EMEA Sales PowerBi.

Chapter 6

Discusion

6.1 Future work

There are some lines of work that could be developed to further enhance the developed pipeline. These proposed future improvements are:

- Move the pipeline to Airflow to have better flow management of the whole pipeline to be capable of sending mails with the scores or retry failed processes. Furthermore, LinkedIn and Glassdoor scraping could be executed in parallel using Airflow DAGs. Regardless, this approach was tried but was unsuccessfully due to server permissions, even though that the DAG code is already developed. After the delivery of this thesis, this line of work will be retried.

- Use data from paid specialized web site to obtain more details from the companies in the CRM and improve the quality of extra information obtained from the web. Especially valuable would be the income of the companies but this is not available most of the time using the web scraping techniques employed in this project. Hence, if we want to obtain more accurate data from companies, the next step would be to purchase this information from web sites like *Hoovers.com*. At the beginning of the project, Hoovers was tried to scrape but this web site detected the web browser to be a scraper in a few companies and didn't allow to refresh the page. Thus, this option was dismissed.

- Connect Salesforce files to the server. This is a line of work that is currently being developed, from a local folder, upload automatically the Salesforce files needed for the pipeline. Further discussions with the Salesforce team will be needed to connect the pipeline directly to the corporate CRM.

- Connect the PowerBi directly to the Z8 Server to have every day the newest leads scraped and with predictions and their explainability automatically every day.

- Try to use different machine learning algorithms to further enhance the model and try to solve the problem using Neural Networks with LIME.

It is clear that more Data Engineer work need to be done to enhance and fully automate this pipeline. Work that after the delivery of the thesis I will continue doing inside HP.

6.2 Conclusion

The purpose of this thesis was to predict a score for each lead to help the salesman know if the lead will become good or bad for HP's MJF printers, moving from a decision based on intuition to a more data-driven decision-making organization. This involved the development of a data science pipeline spanning multiple processes such as joining of the data, obtaining extra information of the leads, process the data and train the algorithm to do the predictions along with the explainability for every lead. And most importantly, due to the dynamic nature of leads, this process needed to be automated and executed daily.

Based on the results of previous chapters, we can say that the objectives were successfully accomplished. The scraping information improved HP's vertical segmentation from 31.83% to 84.26%, and so did the machine learning performance. This process proved to reduce the time that the ISR's spend looking for the information of the lead, estimated to be 891 hours of salesman work and a subsequent cost reduction of 13,481.97€ each year.

In this project, we have proved that the Extreme Gradient Boosting is the best algorithm for predicting the quality of the leads and that this model outperforms the two other tree-based algorithms in all the tested validation metrics. The final model achieved an Accuracy of 0.94282 and an AUC of 0.973318, which is a 13.45% improvement on the accuracy with respect to the model baseline. Furthermore, the use of this score enhances greatly the ISR's decision process since now they have access to the quality of each lead. Therefore, their decisions based on intuition and experience have become data-driven decisions with higher accuracy and with reduced human bias. Furthermore, we achieved no information loss with the change of ISR's, because all the information is stored and used by the algorithm, and hence, this knowledge will always reside inside the company.

Appendix A

Master's thesis source code

The source code for this master thesis can be found in: https://github.com/jordisc97/MSc_Data_Science-Master_Thesis

Bibliography

Bohanec, Marko, Mirjana Borstnar, and Marko Robnik-Sikonja (Apr. 2017). "Explaining machine learning models in sales predictions". In: *Expert Systems with Applications* 71, 416–428. DOI: 10.1016/j.eswa.2016.11.010.

Chen, Tianqi and Carlos Guestrin (2016). "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. DOI: 10.1145/2939672.2939785. URL: http://dx.doi.org/10.1145/2939672.2939785.

Cheriyan, Sunitha et al. (Aug. 2018). "Intelligent Sales Prediction Using Machine Learning Techniques". In: pp. 53–58. DOI: 10.1109/iCCECOME.2018.8659115.

G. Lemaitre F. Nogueira, D. Oliveira C. Aridas (2017). *Balanced Random Forest Classifier*. URL: https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.ensemble.BalancedRandomForestClassifier.html (visited on 09/13/2020).

Korolev, Maksim and Kurt Ruegg (2015). "Gradient Boosted Trees to Predict Store Sales". In: URL: http://cs229.stanford.edu/proj2015/193_report.pdf.

Provost, Foster and Tom Fawcett (2013). "Data Science and its Relationship to Big Data and Data-Driven Decision Making". In: *Big Data* 1.1. PMID: 27447038, pp. 51–59. DOI: 10.1089/big.2013.1508. eprint: https://doi.org/10.1089/big.2013.1508. URL: https://doi.org/10.1089/big.2013.1508.

Quinlan, J. R. (1986). "Induction of Decision Trees". In: *MACH. LEARN* 1, pp. 81–106. DOI: https://doi.org/10.1007/BF00116251.

Ribeiro, Marco Túlio, Sameer Singh, and Carlos Guestrin (2016). *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938.

Wikipedia contributors (2020). "List of mobile telephone prefixes by country". In: [Online; accessed 03-August-2020]. URL: https://en.wikipedia.org/wiki/List_of_mobile_telephone_prefixes_by_country.