Master Thesis : Continuous learning churn prediction in the context of insurance subscriptions

Auteur : Poizat, Adrien
Promoteur(s) : Geurts, Pierre; 19551
Faculté : Faculté des Sciences appliquées
Diplôme : Master : ingénieur civil en science des données, à finalité spécialisée
Année académique : 2022-2023
URI/URL : http://hdl.handle.net/2268.2/18336

Université de Liège - Faculté des Sciences Appliquées

Année académique 2022 - 2023

Continuous learning churn prediction in the context of insurance subscriptions

Poizat Adrien - s162655
Promoteur académique : Pr. P. GEURTS

Travail de fin d'études réalisé en vue de l'obtention du grade de master "Ingénieur Civil en Sciences des Données" par POIZAT Adrien.

Contents

Chapter 1

Context

Customer retention, also known as customer loyalty, refers to the ability of a business to keep its customers during a specific amount of time, without them leaving for the profit of market competitors. It is a key factor in any business, as several studies([1]) have proven that the costs induced by the set of actions taken to acquire new customers are much higher than the costs to retain existing ones.

"A significant factor in the difference in cost is that consumers tend to buy from brands they trust. This is why it takes a lot more effort to convert a new customer than to hold a loyal one" ([2])

Besides brand loyalty, several factors can explain why companies are that focused on customer retention. One of the main expenses in the acquisition of new customers revolves around advertising and the need to have successful promotion campaigns on multiple dimensions. These campaigns sometimes start without much knowledge, targeting a large audience without much focus. In terms of resources, finding new customers requires a marketing team that will identify factors and leads. Then, a sales team should be able to reach out to the targets and convince potential future buyers.

Churn is a phenomenon where customers who have agreed to terms and conditions of a contract in the past make the choice of canceling or terminating that contract, resulting in the loss of a customer for the company that initiated the contract. In the context of insurance subscriptions, most of the subscribers who churn are customers who decide not to renew their contract after a given period. This decision is very challenging to predict, as the cause can come from any source: financial issues, competitors' offers, change in lifestyle... These factors are not numerically conceivable as they are specific to each situation. Even if they were, most of them are considered as personal private information and cannot be gathered legally by companies. Because of that, churn prediction is one of the most challenging tasks in the field of artificial intelligence.

Throughout this work, the three insurance types that are tackled are Car Insurance, Fire Insurance (for home tenants) and Fire Insurance (for home owners). Car insurances are mandatory for anyone and fire insurance for owners. Depending on the conditions of a location, tenants can also be obliged to subscribe to an insurance too. Besides these insurances, Ethias provides insurance contracts such as Healthcare, Finance Products, General Assistance or Common Right. The three types for which we will conduct the study are the three contract types with the highest customer base and, thus, are of the highest interest for Ethias. It is known that the proportion of customers who churn is very low in comparison to faithful customers. Such an imbalance can cause severe issues in the classification process.

Binary classification problems exist in multiple fields of work: They are used to diagnose diseases, for quality control in industries or even spam email detection.The applications of binary classification are numerous and diverse. In the financial sector, they are applied to trends and market evolution.

[1] *Amy Gallo: The Value of Keeping the Right Customers*[6]
[2] *Forbes: Customer Retention Versus Customer Acquisition* [9]

Related to finance, in recent days, numerous businesses tend to focus their effort towards churn prediction, as it costs less to keep faithful customers than find leads to acquire new ones. Such a growth in interest made the companies very dependent on new predicting technologies. In this context, turning to artificial intelligence has to be considered, as it is a key concept. Machine learning techniques, especially, whose role is to predict outputs based on a statistical model that combines multiple inputs, play a major role in these applications. It was in this context that I got to work on the churn prediction problem.

1.1 My Mission within the NRB Group

D-AIM is a marketing data analysis platform that is oriented towards artificial intelligence. In February 2022, they unilaterally ended a contract with Ethias Assurance about the product called "d-predict". This product is a pipeline module whose goal is to predict and detect customers who tend to churn in the context of insurance subscriptions using machine learning models. Since the contract was ended unilaterally, Ethias asked the Data Science team of the NRB Group to investigate and recreate this product from scratch and within a limited amount of time. Although a quite efficient module was built by C.Schleich and her colleagues, there are some parts of the pipeline that are not well-designed. The parts that require additional care revolve around three subjects:

- The current module is built with sketchy data cleaning/preprocessing techniques. Obsolete and irrelevant variables are used for modelling, and no analysis of the data has been conducted. As a data scientist, my mission is to gain an insightful grasp of the available data in order to be more efficient in the task of selecting and transforming the information.

- Performance of the model : The models are evaluated using accurate techniques. However, no retrospective analysis of the results has been conducted with respect to the choices that are made with the data. My mission is to use my statistical expertise on the data to build models that would match (or even surpass) the performance of already existing models.

- Continuous learning : The current pipeline is a module that is instantiated each month. Each month, new models are trained from scratch based on the most recent data and no feedback is provided about the performance of the module from earlier months. My mission is to propose a pipeline architecture that would take into account data and models from previous months without having to rebuild models from scratch. This is a hard challenge as it comes with its set of constraints such as memory, data and model storage.

These three subjects are treated separately in three chapters, with the first chapter about Data analysis being the chapter that got most of my attention for multiple reasons. During my Master's in Data Science Engineering, I've had the chance to learn multiple techniques in the domain of Exploratory Data Analysis and my wish was to apply these methods to a real-life situation with datasets that presented a structural challenge. Churn prediction is a good opportunity, as it is a very challenging task that requires a deep knowledge of its influential factors.Moreover, deepening Data Analysis is a duty for a Data Science Engineer. While I consider that a Data Science Engineer should be an expert in various aspects of Data applications (such as Machine learning, Data Analysis, Data Visualization or Data Engineering), I consider that discovering wishful insights and relations between customer features and using them to give interpretations to behaviors is the most important part of such work (and it is also personally the most exciting). In such a field, there are models that are already known to be performing well and finding models that would perform better than those that already exist is not within everyone's reach. I don't have the audacity to state that I will find better working models during this work, but I will try, at the very least.

Another interesting aspect of focusing efforts on data insights is from the side of the company. The potential market value of such insights is non-negligible, as it allows insurance companies to alter their way of thinking when it comes to customers retention. If a subset of factors is considered heavily significant in the churn outcome, the marketing team should apply their expertise on this specific subset

and change their retention strategies. As a student who did a lot of theoretical work and projects, I wanted to look for an internship for which my data science expertise could be of any use in a real life application as it gives me a strong feeling of accomplishment.

1.2 Internship follow-up

During the internship, I had the chance to work under the wing of C.Schleich who worked on churn prediction for a few months. She was responsible for the D-Predict module after the contract with D-AIM had ended. She was very available and since the module was built recently, my discussions with her about the different matters were rich and resourceful. Three years ago, Ms. Schleich pursued a master's in Computer Science Engineering at the University of Liege, with the "Intelligent Systems" specialization. Since the programs of this master and mine had a lot of common lessons, we could easily discuss most of the ideas, especially when it came to artificial intelligence, machine learning, and pipeline structures. Every Friday, I summarized my activities of the week, and at each time she gave me valuable feedback and ideas for improvement and future work.

I also had the chance to have a rigorous follow-up from my academic promoter Prof. P. Geurts during the semester, with regular meetings every 2 or 3 weeks. During these remote meetups where Ms. Schleich was present too, I demonstrated various results from the applications of my different ideas and discussed uncertainties about innovations. Since Pr. Geurts is a machine learning and algorithmic expert, his expertise was on point and he oriented my work towards interesting solutions. I appreciated this follow-up, as it led me to continuously find new ideas for improvement.

Most of my internship was conducted remotely. I was lent a laptop with remote access to the company's intranet to work from home. NRB didn't give me any obligation in terms of office presence, as they have quite a lax policy in terms of remote working. The team I joined had to be in office for minimum two days a week, and most of them were comfortable with these two days. I still had the opportunity to discover the office and I had a dedicated desk to do so. From the beginning of February to the end of April, I went to the office 1 or 2 times a week, coordinating with Ms. Schleich so I could directly ask her if I had any questions. I spent most of the remaining time working from home, occasionally going to the office whenever I had significant uncertainties. I still had the opportunity to get to know the AI team and have lunch with them. They are not a big team (4 persons) and it was easier to get to know each one of them individually. The office presence was not very important, as everything I needed was available on the laptop that I was lent. Whenever I needed data or scripts, Ms. Schleich would send them to me, as long as she was able to.

In data science studies, there exists the apprehension that not enough data is available to conduct an efficient study and obtain results that we can trust. There is also the fear that the data is too old and therefore not representative of the present situation. Fortunately, this is not the case here, as I had access to half a year of data for three insurance types (car, fire insurance (for tenants) and fire insurance (for owners), from April 2022 to August 2022. Each dataset has sizes ranging from 60 000 to 300 000 customers with more than 300 features. While the duration might be too short for a long-term continuous learning study, it remains sufficient for a research work based on historical data.

1.3 About NRB

([3]) With a consolidated turnover of € 501,6 million and over 3,300 employees, the NRB Group is one of the main Belgian players in the ICT sector with a European vocation. NRB's mission is to provide optimal end-to-end IT solutions and services through a close and long-term partnership with its public and private sector clients to simplify technological, economic, and societal transformation.

Within this context, NRB provides a complete range of ICT services based on four key areas: consultancy, software, infrastructure & cloud services, and managed staffing.

- **Consultancy**: The consultants assist clients throughout the process of their digital transformation. In addition, a team of cybersecurity specialists helps them develop and implement the appropriate policies and technologies, ensuring the integrity of their organisation, systems, and data.
- **Software**: The Software Factory consists of development teams, based in Belgium (Afelio) and in the nearshore centre in Athens, mastering a wide range of technologies for the creation of mobile and web applications as well as for customised solutions for distributed and mainframe environments. Moreover, they implement packages from major software vendors such as SAP, Microsoft, IBM, Cisco, Software AG, and others. NRB disposes of more than 100 SAP experts implementing ERP, IS-U, FI-CA, financial, logistic and HR projects in public and corporate organisations and utilities. Their specialists ensure the customisation, the integration and the maintenance of the solutions supplied.
- **Infrastructure services**: NRB's infrastructure and managed operations services encompass housing and hosting of mainframe, AS400 and distributed systems. To provide these services, NRB can rely on its own data centres located on two georesilient sites in Belgium, an infrastructure that comes up to the Tier 3+ requirements of the Uptime Institute.
- **Hybrid cloud Services**: Thanks to this infrastructure and its strategic partnership with IBM, NRB launched a unique offering for intelligent hybrid cloud services, branded NECS, short for NRB Enterprise Cloud Services. This enables customers to access – through one single interface – and make optimum use of the NRB private cloud and public cloud services from leading global providers, such as IBM, Microsoft Azure, Amazon or Google.
- **Managed Staffing**: Finally, NRB provides managed staffing services aimed at offering the best-fitted profiles at the best possible price taking into account the customer's requirements. In February 2020, this service was reinforced by the acquisition of People & Technology, a company specialising in the provision of IT profiles.

The NRB Group takes a lead in the field of Smart Cities, as we continue to deploy our efforts in the field of merging technologies, such as Artificial Intelligence (AI), Internet of things (IoT), Robotic Process Automation (RPA), and security.

NRB focuses on specific sectors such as the public and social sector, the sector of energy & utilities, the financial services sector and the business & industrial sector. Trasys International, business unit of NRB S.A., aims at European and international public organisations and corporations. The subsidiaries of the NRB Group offer sector-specific solutions supported by the Group's scale, ICT infrastructure and other support.

[3]More about the NRB Group: https://www.nrb.be/en/about

1.4 Literature Review

There are multiple arguments that lead to the need for deeply exploring churn analysis in multiple fields of work. Zhang Rong et al.[17] highlight the main arguments and propose a way to approach the problem in terms of modelling. However, before actually trying to predict churn, it is mandatory to have a deep knowledge of the influential factors and attempt to find interpretability with the results [13].

The different techniques for treating customer data in the context of insurance subscriptions with the goal to predict churn is a subject that is already frequently studied. A.Groll et al.[1] propose a wide range of machine learning techniques that can be used for the purpose of detecting relevant factors to the churn problem. The main challenge behind churn prediction is that there is a severe class imbalance between positive and negative churn cases. V.Effendy et al.[5] propose to handle class imbalance with the use of combined sampling and weighted random forest, which is an idea that we will tackle in Section 4.3.1. Customers data can also come with a high proportion of missing data which needs to be handled. Most state-of-the-art techniques rely on deep learning methods to synthesize missing data, such as Iterative imputation[16].
Customers databases are highly multivariate and treating them efficiently is a challenge in itself. With a high number of dimensions and a large customers base, it is mandatory to consider dimensions and data reduction techniques[11].

Predicting churn is a problem that has already been confronted by multiple machine learning models, such as tree-based models, Bayesian models, Neural networks, Support Vector Machines and so on. Currently, the models that have proven to be working at best for churn prediction are boosting algorithm based on stochastic gradient descent [14], as boosting algorithms have been surpassing averaging algorithms for more than 15 years[10]. As for other techniques, there is also the possibility of focusing on non-static features and building recurrent neural networks that take into account customer time-varying data[12]. In the present work, we consider that since clients overlap and the objective is to perform continuous training, the time-varying features are implicitly taken into account. In the case of more simple models such as Logistic Regression, there is the question of adding supplementary terms to the learning algorithm, such as Lasso or Ridge regularization[18].
Continuous churn prediction is a challenge in many ways. Iterative learning requires the use of On-Line variations of Ensemble methods implementing voting methods[2]. These On-Line Ensemble methods would use feedback of their training on previous data sets to improve their future inference performances[7].

Chapter 2

Data

2.1 Terminology

In the context of this work, some expressions will be used multiple times and sometimes using a different terminology. Here is a non-exhaustive list of these terms and their signification and synonyms used in the text:

- Churned customer/positive churned customer: Customer who is positive to churn. Customer whose output label is Positive (equal to 1 when mapped to an integer, in opposition with negative churned customer who is labeled as negative, value 0).
- Feature/attribute/candidate variable/input variable: individual measurable property of the data. In a dataset presented as a table, a feature is represented by a column. Although the term might vary depending on the discussion that is made (theoretical or practical), the sense remains the same.
- Output : The term output is defined as the label or target value associated to each data point in a supervised classification problem.
- Output value/Target output/Target value/Output label: This set of terms refers to the output of a data point under its different forms. The label is either "Churn" or "No churn" and is mapped to the output values 1 and 0. The remaining terms can be used when talking about both representations.
- observation/observed data point: The set of input feature values for one customer. An observed data point also contains its output, in opposition to unobserved data that contains only the inputs. The term "observed" refers to the availability of the output value.

About the Data

This section contains all the accessible information about the data that will be manipulated in the present work. This includes how the data is gathered (multiple sources) and how it is presented (nomenclature of the different variables, variable types, format). Working with data that belongs to a client involved not being able to have access to the whole process of client information gathering due to confidentiality issues.

The nomenclature and specification of the data is displayed in Appendix .1. The file summarizes all the disclosed information about the definition and nature of the features. The name, description and values of the features have been kept in their language and form of origin.

Single Views (SV)

The presented datasets concern the identification of risk factors and the prediction of indicators of churn in the context of insurance subscription. The aim of these datasets is to identify, for each churn

type, the factors that tend to influence the choice of clients about an insurance contract renewal. The conditions in which this data was collected remain unknown, since data gathering is on the client's side and there is a confidentiality agreement between them and NRB that allows a non-disclosure of the gathering process.

There are three types of insurance that will be covered in this study (also known as *tasks*) : car insurance, fire insurance (homeowner) and fire insurance (tenant). In the process as is, *Ethias* provides NRB with customers information that belongs to one of the 5 following categories :

- `Client`: Personal information such as age, profession, seniority, family situation...
- `Products`: Information about products and services for which clients have signed up, number of products, product expiration...
- `Sinistre`: Information about claims (months since last accident or loss, number of claims...).
- `Intervention`: Number and type of interventions, dates of intervention...
- `Contracts`: Specific information about the car and fire insurance contracts : presence of warranties, types of insurance contracts,...

The remaining variables are used for identification. Following the European GDPR security law, the data requires to be anonymized. The anonymization process is done here by "pseudonymization" of the customers, which is a mapping of each customer to a series of numbers.

Once all features from each subset are received on NRB's side, they merge them together to create what is called a Single View. At each month, a Single View consists of all the information about the clients that are in the client base. This means that the same client can be present in multiple consecutive Single Views (while still having some changes in the features because some of them include non-static information). This observation is interesting, as having multiple occurrences of the same client until he churns will allow one to monitor the changes in (only) non-static values and possibly interpret these changes.

The exposed process takes place every month. There is however an important factor that needs to be taken into account. Every month, the goal of *D-Aim* is to provide *Ethias* with the most accurate predictions of the current month's datamart of clients. The issue is that the initial learning process which is a pipeline that is depicted in Section 3 requires that each month a new model should be trained with labeled data. Since the data of the present month is not supposed to be labeled (because otherwise it would mean that the job of predicting would be pointless), there is a need to make use of old data for which ground truth (the knowledge that a client churned or not) is available. The old data should not be too old though, because if the gap in time between the present month and the month of reference is too important, the client base changes too much and the models that will be trained based on the month of reference will not be as representative as needed for new data (model generalization will be laborious). The month that is taken as reference is thus the most recent month for which the target output is available. On the side of *Ethias*, it takes 3 months to collect the decision of the clients, since this is the period of time after which *Ethias* assumes that if one of their client did not give any intention of renewing the contract, then they indeed are considered as churned.

With this in mind, the data that will be used each month consists of two Single Views : one of the current month that is the data for which we want to perform a prediction and the other that is the Single View of the most recent month for which we have acquired the targets, labeled with these targets. For further understanding, let us consider an example : We want to perform a prediction for the month of August 2022. It takes 3 months to collect client targets. At the end of July, we should have thus received the targets of the client base of April 2022. The dataset that will be used for training the supervised learning algorithm will thus consist of the Single view of April 2022 merged with the targets obtained in July 2022 while the dataset that will be used for inference (predictions) will consist of the Single View of August 2022.

Historical Data

In the last paragraph, it was explained that it took some time on the side of Ethias to acknowledge that customers had churned. Most of the customers who churn will not give any news during a certain period, period after which Ethias will consider that they churned. They chose a period of 3 months. Fortunately, we will work on Historical Data, meaning that we don't have to wait for 3 months to acquire targets. Instead, we can directly work on each Single View with the available output labels. This aspect of the work is important to mention, because when it comes to continuous learning in the present, this delay is non-negligible.

Household View

There are a lot of features (labeled with "`_HH`" at the end of the name of the feature) that are said to be "On Household view". This signifies that there exists a feature with the same name (without the "`_HH`" at the end) that is used to determine its value. The value is determined based on the following schema: If a customer is the only customer in his household, the value of the Household-specific variable will be the same as the single customer. If there are multiple customers in the same household, the value might differ (every given precision is shown in the Appendix .1). It is important to specify this aspect of the Single Views, because when dealing with hypothesis tests and other data analysis tools, there will be obvious strong relations between these kind of features.

High-dimensional data

With a set of more than 300 features and monthly datamarts whose client base range from 60 000 to 300 000 members depending on the insurance type, it is needless to say that a thorough analysis of the data will be required to efficiently grasp the full potential of the available information without sacrificing too much resources. This involves selecting the best subsets of features to build representative models via methods that are regrouped under the term "Feature engineering". There is also a possibility to select a subset of observations such as to drastically reduce computation time without compromising too much on quality.

Whenever the information of a problem is presented as tabular data, most real-life machine learning processes tend to prioritize an efficient use of computational resources over the faithfulness of models to this data as it was presented initially. Feature and row selection, if done correctly, reduces the need of high-cost computational resources without sacrificing too much information. There is however a catch: Making assumptions that ultimately result in the loss or selection of a fraction of all the available information induces bias, i.e the outcome of a model built on unfounded (or even founded) assumptions is prejudiced by these assumptions and there is a possibility that this outcome is not in accordance with reality. There can be bias anywhere in a machine learning application. Most of it can be justified and used skillfully with the help of expertise in the domain or a good knowledge of machine learning processes.

2.2 Exploratory Data Analysis

In this section, we do not consider the changes between the different datamarts of distincts months and we assume that choosing one Single View per insurance type will be representative enough to accomplish exploratory data analysis. In this case, the Single Views of April 2022 will be taken as a reference.

2.2.1 Overview

Table 2.1 is a summary of the dimensions of the available data for April 2022. It can be seen that depending on the insurance type, the number of clients is quite different. The number of features used for car insurance is higher because there are multiple warranties that are to be taken into account that

are not proposed in the other insurance types. Appendix .1 gives full details about the definition and nature of the features contained in the Single Views.

Insurance type	Rows	Features
Car insurance	280591	345
Fire insurance (owner)	176459	306
Fire insurance (tenant)	83512	306

Table 2.1: Dimensions of the Single Views for each insurance type, April 2022.

Among the whole set of features, no recent cleaning has been done, therefore there is a non-negligible presence of obsolete variables that need to be disposed of. In some cases, the obsolete variables are already replaced by other features in the datamarts. It is thus mandatory to remove them, otherwise we could be dealing with multicollinearity (variables that are highly correlated) , which we want to avoid. In some other cases, the features are not collected for new clients or are soon to be replaced. Moreover, there are variables that are either unique (identification from the pseudonymization) or missing most of the time (see Section 2.2.3). Table 2.2 gives a display of the dimensions of the problem after the removal of obsolete and irrelevant variables. It can be seen that there was a lot of rubble in the datamarts, since we got rid of more than 30% (100+) of the features for each insurance type. After having analyzed the missingness of the different features, we will choose either to impute the missing variables or get rid of the feature if the missing ratio is too high. We will see that this first filtering will heavily reduce the number of features in the datamarts (see Section 2.2.6).

Insurance type	Rows	Features	Numerical	Categorical	Binary
Car insurance	280591	220	148	46	26
Fire insurance (owner)	176459	196	144	44	8
Fire insurance (tenant)	83512	193	140	44	9

Table 2.2: Dimensions of the Single Views after filtering, April 2022.

Table 2.2 also shows the count of feature types in the datamarts. Having knowledge about the nature of the features that are dealt with is very important for multiple reasons. First, the statistical measurement that can be made on the features are not the same depending on the type of feature that is dealt with. For instance, it is impossible to compute the mean or the percentiles of a categorical variable so the tools used should be different. In the Appendix 5, some statistical features of the different variables are computed. For numerical variables, there exists features such as mean, standard deviation, minimum and maximum value and quarter percentiles. For categorical variables, there exists the count of unique occurrences (number of category for categorical variables), the most common value (top) and its frequency. Each of these metrics can be used to perform statistical tests that will be covered later. The second reason why it is mandatory to know about the data types is about the learning phase. Some models only work when dealing with numerical data. To make efficient use of these models, the non-numerical data should be pre-processed and converted into numerical data without loosing too much information. As a side note, binary indicators are treated as categorical variables, but it was interesting to observe the quantity of such indicators in the Single Views, hence the need to separate features that are considered to be categorical in 2 columns : purely categorical features such as features that contain characters or symbols ("Categorical") and binary indicators ("Binary"). 2.2.

2.2.2 Target variable

The target variable is a boolean variable that is equal to 1 if a client has churned and 0 if not. Let us first look at the different churn distributions on Figure 2.1. It is first observed that we will be facing a binary classification problem with a severely unbalanced target class distribution. This observation implies a lot of things as detecting positive churn cases in such an environment will become quite challenging. As it is detailed in Section 3.4.1, there are multiple ways of measuring the performance of a

binary classification task. They depend on the motivation behind the study and the client's wishes. A highly unbalanced target distribution means a very high difficulty of achieving some of the objectives desired by the client or the researchers. For starters, *Ethias*'s wish is to detect customers that are positive to churn, because customer attrition is one of their main concern. If a model is trained using the whole Single View as reference, it means that almost 99% of the observations in the training phase are labeled as negative. This would result in a model being very efficient in the detection of negative cases, but not efficient at all in detecting positive cases. In terms of pure accuracy, such a model would have almost a perfect score (because it would recognize negative cases that are present almost 99% of the time in every datamart) but it would be useless for the client.

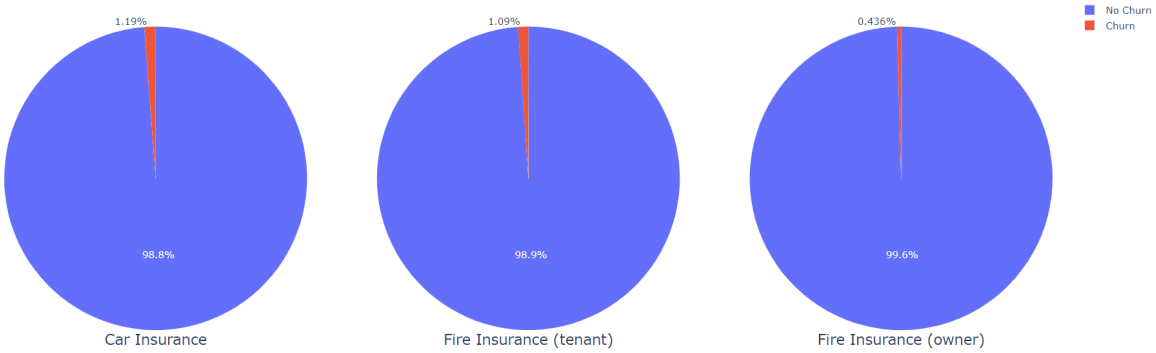


Figure 2.1: Churn distribution of the 3 insurance types, April 2022

With such high imbalance, there is a need to make assumptions on the prior knowledge that we have by either under- or oversampling the datamarts such as to balance the training and have a model that perform well both for negative and positive cases. We will see later that balanced sampling gives more interesting classification performances (see Section 3).

2.2.3 Missing data

The analysis of missing values in the Single Views requires a clear definition of what is considered missing and what is not. The Single Views are composed of features from different sources and the missingness is not always expressed as a *NULL* value. However, the documentation does not mention whether or not specific symbols express missingness or some sort of information. These symbols include *"?"*, *"."* or *"-1"*. An interrogation point *"?"* could represent an information that should be delivered in a future Single View, but it could also mean that the information is and won't be available at any time. For a static analysis of an individual Single View, it does not change much, but in terms of interpretability, it can be discussed. This reflection leads to the need of choosing whether or not these symbols should be considered as missing values (and thus be dealt with using techniques such as Imputation, see Section 2.2.8). We first take a look at missingness with the assumption that the mentioned symbols are missing values and then we take a look at the missingness with the assumption that they are not.

Figure 2.2 is a data-dense display that allows to visualize patterns in data missingness (called nullity matrix). Each row is an observation (100 were randomly picked) and each column is a feature. The white spaces represent missing values. This matrix shows two things : First, the proportion of missing data is very high. Some variables are missing more than 90% of the time and dealing with such a lack of information is complex. Secondly, there exists multiple patterns of jointly missing values that allow us to introduce the first assumption about the Single Views: Missing Not At Random (MNAR). The MNAR assumption refers to datasets for which the missingness of variables is related to unobserved values (other missing variables), as opposed to "Missing at Random" or "Missing Completely at Random", which assume that there is no relation between the missingness of variables and unobserved values. This is one important assumption to make, as dealing with MNAR datasets is more challenging

than dealing with MAR or MCAR. For random missingness, statistical methods can be applied without much thought, but when there exist patterns, their use can lead to biased or misleading results if they are not properly addressed.

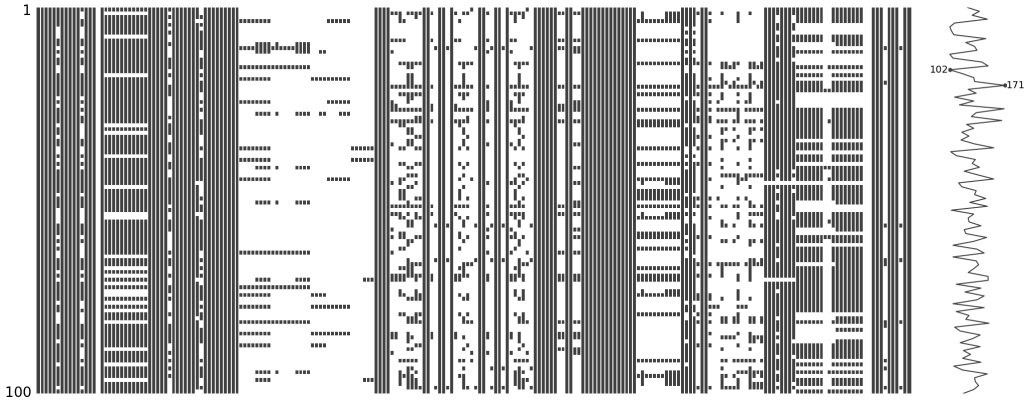


Figure 2.2: Nullity matrix of 100 random samples of a Single View, with symbols as NaN (Car Insurance, April 2022)

Concerning the Single Views, there is a choice that needs to be made. The goal is to minimize the loss of information while minimizing the bias that is introduced. Moreover, we want to be able to interpret results. Bias can be introduced while (for example) synthesizing data using observed data. However, the more value a column is missing, the more values will be synthesized and the more bias will be introduced. The best compromise that was chosen is to keep every feature (whose missing rate did not exceed 99% and had at least two unique values) and consider only values that are "NULL" as missing (as they are not a majority). The symbols *"?", "."* will be treated as particular categories and will later undergo the process of Categorization (see Section 2.2.7). As for -1, they will be treated as any integer value, as -1 values only appear in numerical features. The new nullity matrix is displayed in Figure 2.3. It can be seen that there is a lot less to deal with, and no information is lost for now. Here again, there are missingness patterns consolidating the assumption of a MNAR Single View.

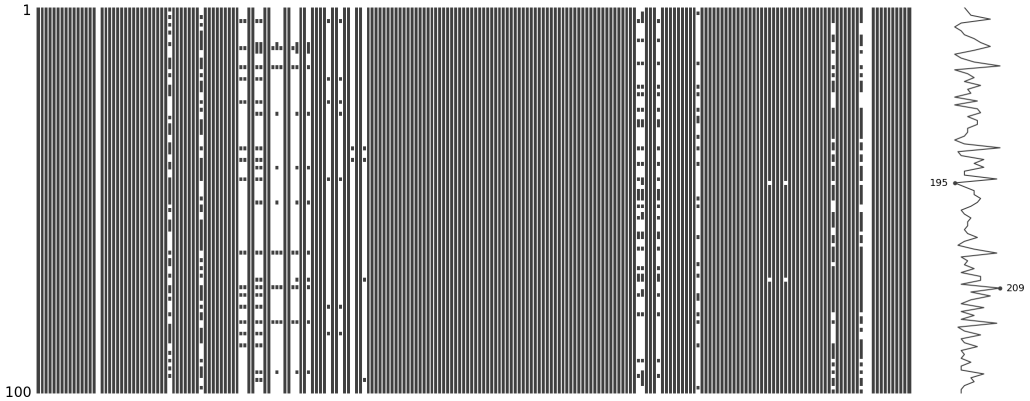


Figure 2.3: Nullity matrix of 100 random samples of a Single View, without symbols as NaN (Car Insurance, April 2022)

Let us have a closer look at the patterns. We avoid dealing with every feature that has missing values by applying a filter of minimum 30% of missing values (the threshold is chosen arbitrarily). Figure 2.4 is a heatmap that shows the correlation of the missingness of each feature two-by-two. Values that are close to -1 signify that if one of the two variable is present, then the other is likely to be missing. A value close to 0 doesn't imply any relation between the missingness of two variables and a value close to 1 implies a strong relation. The majority of joint missingness comes from the "Products" subset of features, where we observe correlations ranging from 0.3 to 1. The reason behind

the patterns lies in the definition of these features: Some of them are dependent and if one is missing, then another one or multiple others are automatically missing too, as they are conditional of each other. This is a phenomenon that occurs a lot with binary indicators. Most of the time, there is an indicator that conditions the value of another variable. If this indicator is not available, then the other variable isn't either. During the analysis, we also discuss how to manage multiple variables that carry redundant information. Dealing with multicollinearity is a different problem from exposing missing patterns and will be discussed in Section 2.2.9.

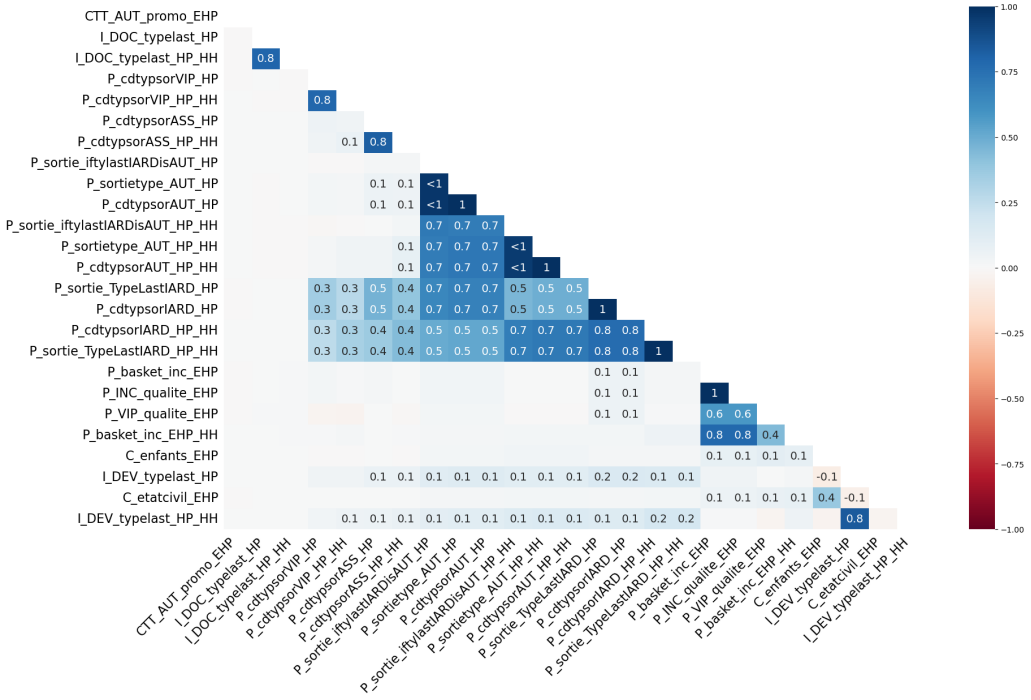


Figure 2.4: Correlation Heatmap of Missingness for all features that are missing at more than 30% (Car Insurance, April 2022)

2.2.4 Categorical variables

The following discussion is a first attempt towards finding the potential risk factors of churn (First for categorical features and then for numerical features). This first forecasting attempt is done by individually looking at each feature and observing the percentage of churned customers for each value of a feature. We know for sure that the analysis of the relation between individual variables and the output is not a task that should be taken as a reference for determining risk factors, as relations between features and the output are far more complex than simple one-to-one relations. We could however get insight about some features and then verify our speculations with more elaborate ideas. One of these ideas involve the use of Random Forest to determine feature importance in terms of classification and is tackled in section 2.2.10.

Note: For loyalty purposes, the following analysis has been made on datasets with only complete-case scenarios for the studied features. The analysis considers that there are no missing values, as the case of these missing values will be covered in Section2.2.8.

Let us first define what is considered non-numerical and numerical variable in the context of Single Views. Anything that is sorted as a non-numerical variable includes variables that have a string format (categories, flags, contract types, demographic and personal information, profiles,...) and boolean indicators. Anything that is sorted as a numerical variable includes temporality variables (seniority in the company, months since last contract exit,...) and quantities (number of contracts, monetary values, customer values,...).

We get some insight about the categorical features by building the histograms of occurrences of the target variable w.r.t to their categorical value. If the difference in proportion of churned customers is high between different categories, then it can mean that there is a link between the current feature and the target variable. An aspect that will be focused on is the difference between churn types. Some factors can affect only one specific type of insurance while some others are more global and affect multiple types. The difference in proportion should be relevant, as the number of churn cases is low. The intuitive meaning is that small differences cannot lead to reliable hypotheses. For example, Figure 2.5 is a histogram of the count of churned customer w.r.t their marital status in the context of car insurance. Between single and divorced customers, the difference in proportion is low ($578/45660 = 1,2\%$ for divorced customers and $91/11120 = 0,8\%$ for single customers), so there is not much to say about it. However, when we compare single customers ($0,8\%$) with widowers ($62/2225 = 2,8\%$), this difference starts to be higher. But one should be careful, because a high difference in the number of samples can induce this analysis to be skewed. To get efficient insight, we look for high difference in churn proportion between category values that have a similar number of samples.

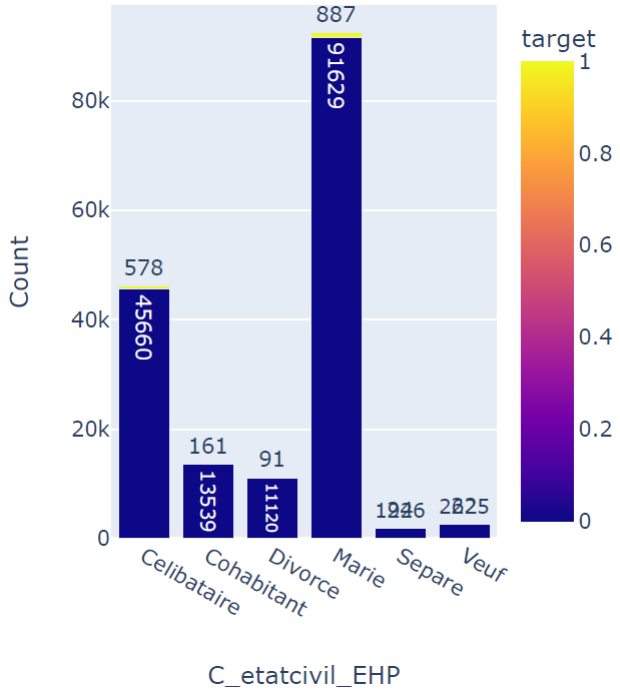


Figure 2.5: Count of churn occurrences based on the marital status (car insurance, April 2022)

While doing some exploration of the data, some interesting features that could lead to relevant hypotheses were observed. Since there is a lot of variables, the exploration is done in accordance with the 5 feature subsets of the Single Views that are described in Section 2.1 (`Client, Products, Sinistre, Intervention, Contracts`) even if sometimes not all subsets are represented.

In the `Products` variables, the variable `P_sortie_TypeLastIARD_HP` is defined as the type of the last contract that was terminated during the historical period of the present insurance subscription (also called "contract exit"). A customer has a value for this variable if he subscribed for any other product or service aside from the present insurance subscription. These kind of values are interesting in the case of cross-selling inquiries (selling products to clients that are already customers). In the case of Fire insurance (tenant), Figure 2.6 shows us that the different exit types have varying churn rates (DCO:$2,3\%$, ASS:$1,6\%$, AUT:2%, INC:$1,1\%$) with a (roughly) similar number of samples. Customers whose last contract exit type is DCO (Common right) are more than two times prone to churn than customer whose last contract exit type is INC (Fire) and in general, the INC type has the lowest churn

rate. Without the requirement of expertise in the domain, it is intuitive to say that customers who already exited a fire insurance contact will most likely not sign up for another fire insurance contract in the same company and those who still do will therefore be less likely to churn. There is also another notable observation: the mean churn rate of customers with a contract exit type is higher than the global churn rate of Fire insurance (tenant) shown on Figure 2.1. This can lead to the hypothesis that customers who already exited a contract of any type are more prone to churn than those who didn't.

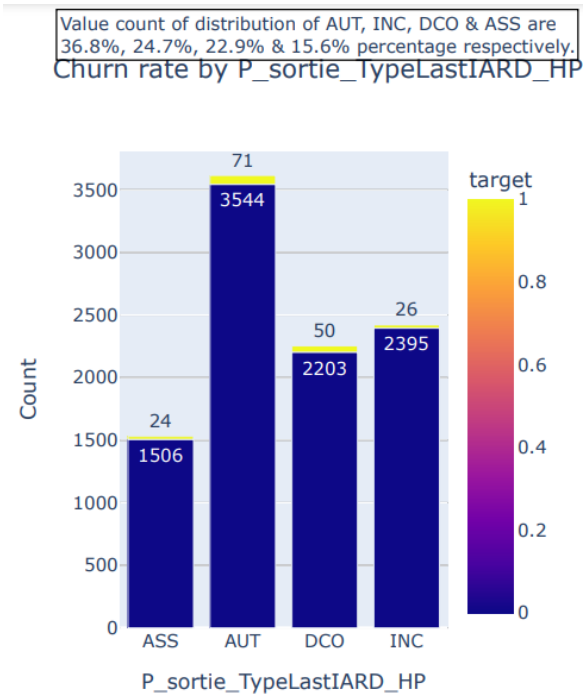
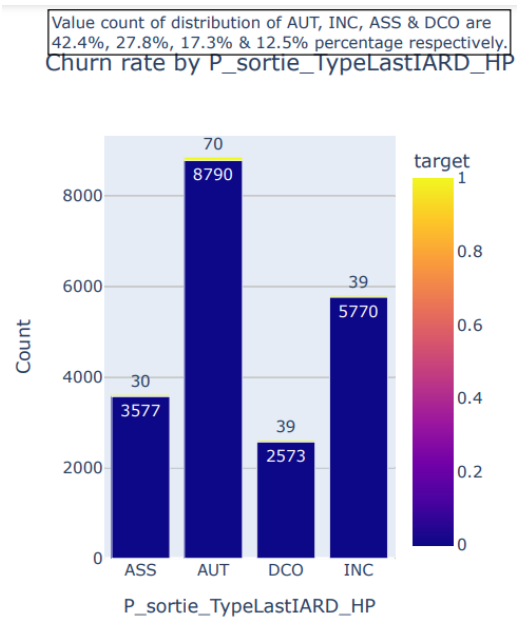
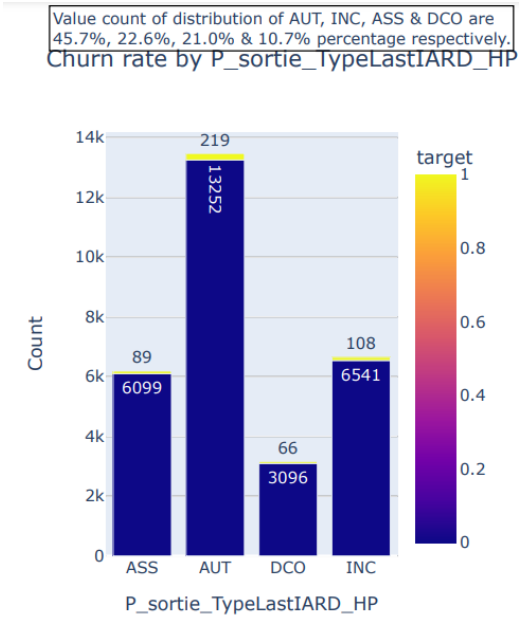


Figure 2.6: Count of churn occurrences based on `P_sortie_TypeLastIARD_HP` (Fire insurance - tenant, April 2022)

We now confirm whether or not these intuitions are plausible by looking at the same histogram for the two other insurance types. Figure 2.7 shows that, in the case of Fire insurance (owner), the INC type contract exit has still the lowest churn rate and DCO has still the highest (DCO:1, 5%, ASS:0, 8%, AUT:0, 8%, INC:0, 65%). The intuition that was exposed earlier can lead to a good hypothesis: In the case of fire insurance (both types), customers that have already exited a fire insurance contract are less likely to churn from the current contract. Taking a look at Car insurance now, the churn rates vary in a slightly different pattern (DCO:2, 1%, ASS:1, 45%, AUT:1, 65%, INC:1, 65%). The DCO type has the highest churn rate, but the INC type has not the lowest rate anymore. Here, the churn rates are quite close and the difference between the number of cases per category is higher than in the case of fire insurance. Any hypothesis that stems from these observations would be quite weak. Still, we can confirm the hypothesis that for all insurance types, the mean churn rate is higher if the customers have already exited any type of contract. Table 2.3 shows that the churn rate is multiplied by 1,5 to more than 2 depending on the insurance type.



(a) Count of churn occurrences based on `P_sortie_TypeLastIARD_HP` (Fire insurance - owner, April 2022)



(b) Count of churn occurrences based on `P_sortie_TypeLastIARD_HP` (Car insurance, April 2022)

Figure 2.7: Comparison of churn occurrences based on `P_sortie_TypeLastIARD_HP` for Fire and car insurance types.

Insurance type	Global churn rate	Mean churn rate for any `P_sortie_TypeLastIARD_HP` value
Car insurance	1,19%	1,7125%
Fire insurance (owner)	0,436%	0,9375%
Fire insurance (tenant)	1,09%	1,75%

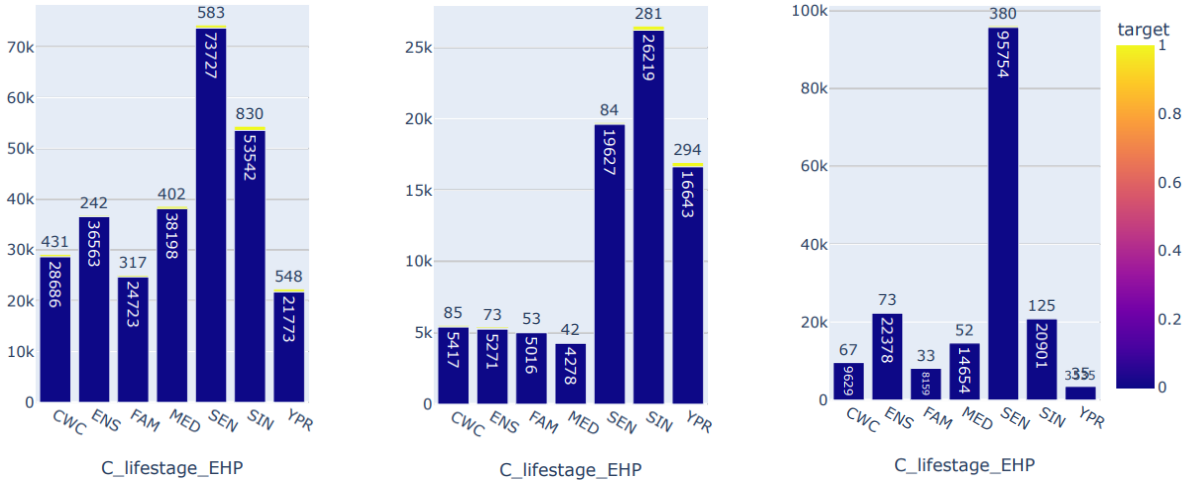
Table 2.3: Comparison of global churn rate with churn rate specific to the presence of `P_sortie_TypeLastIARD_HP`.

The observations made by looking at the distribution of `P_sortie_TypeLastIARD_HP` are reinforced by doing the same analysis on variables that belong to the same subset of features and that also concern contract exits. In the Appendix .3.1, there is a compilation of histograms of `Product` variables that confirm the hypothesis that if a customer already got out of a contract, he is more prone to churn. We also take a look at temporal factors that could indicate higher churn rates. In certain situations, customers tend to exit a contract after a given period of time (counted in number of months) and it is hard to guess for which reason he would take this decision.

Unfortunately, the variable `P_sortie_TypeLastIARD_HP`, just like all the other variables tackled in Appendix .3.1, has a very high missing value ratio (89, 49% in the case of Car insurance). This means that approximately 10% of the customer base has exited another contract during the period of the insurance contract. This observation showcases how important features can be even if most of their values are missing in a dataset. The decision about the conservation of features with missing values will be influenced greatly by such analyses.

Let us now try to look at variables that are more represented in the dataset in order to obtain hypotheses that are compliant with a large portion of the data. In the `Clients` subset of features, the variable `C_Lifestage_EHP` is defined as the customer's life stage at the end of the historical period of the present insurance contract. The different categories are: CWC (couple without children), ENS

(teacher), FAM (family), MED (medior), SEN (senior), SIN (single), YPR (young person). We can first have a look at the distribution of the dataset among the different lifestage categories. It is important to note that the low missing value rate of `C_Lifestage_EHP` (less than $0,01\%$) allows us to consider that the distribution shown on Figure 2.8 is quite representative of the whole dataset. We can first see that there is a dominant presence of SEN (senior) profiles among the different contracts, this dominance being only surpassed by SIN (single) profiles in the case of Fire insurance for tenants. YPR (young) profiles are the least present when it comes to car insurance or Fire insurance for owners (the latter might be coming from the fact that there are not much young home owners). However, they are on the heels of SEN and SIN profiles when it comes to Fire insurance for tenants. This can be explained by the fact that in the world of real estate, there are much more young home tenants than young home owners.



(a) Car insurance, April 2022

(b) Fire insurance (tenant), April 2022

(c) Fire insurance (owner), April 2022

Figure 2.8: Comparison of churn occurrences based on `C_lifestage_EHP` for all insurance types, April 2022.

Looking at the churn rate now, YPR profiles seem to have a very high ratio in comparison with older profiles such as MED or SEN. When it comes to car insurance, the churn rate of customers belonging to the YPR category is equal to $2,5\%$ which is more than twice the global churn rate of car insurance customers ($1,19\%$). For young owners too, this rate is equal to 1% against $0,436\%$, which means that the behavior of YPR profiles doesn't change much among the different contract types. Table 2.4 gives a ranking of the different lifestage categories based on a weighted sum of their churn rate among the 3 insurance types, from highest to lowest value. The weights are based on the global churn rate of each insurance type. As detailed previously by Figure 2.8, YPR profiles have the highest churn rate across all contract types and SEN profiles have the lowest. It also appears that teachers (ENS profiles) seem to be less prone to churn than families (FAM) or couples without children (CWC).

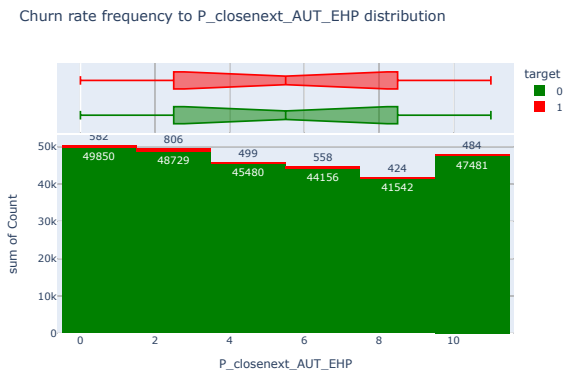
Lifestage Category	Car insurance	Fire insurance (tenant)	Fire insurance (owner)	Weighted Sum
YPR	2.52%	1.77%	1%	**1.9768%**
CWC	1.5%	1.57%	0.7%	1,4%
SIN	1.55%	1.07%	0.6%	1.206%
FAM	1.28%	1.06%	0.4%	1.0512%
MED	1.05%	0.98%	0.35%	0.91%
ENS	0.66%	1.38%	0.33%	0.8952%
SEN	0.79%	0.43%	0.4%	0.5836%

Table 2.4: Ranking of the lifestage categories in terms of overall churn rate, April 2022.

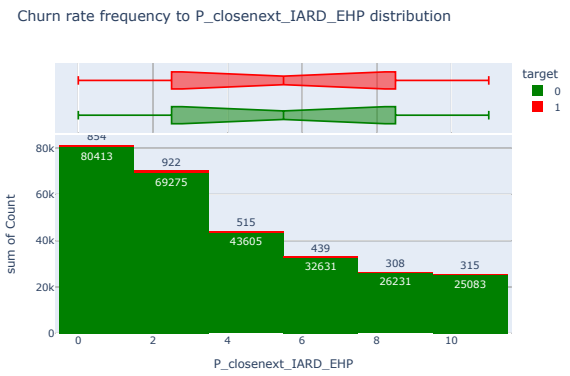
2.2.5 Numerical variables

In the case of numerical variables, the analysis is quite similar to categorical variables with the difference that instead of monitoring individual categories, we monitor the evolution of churn rate among a continuous variable and we introduce the use of boxplots. They are a great tool for comparing distributions and, combined with the evolution of churn rate, they can lead to a good interpretation of a distribution's skewness and overall behaviour.

Let us illustrate how boxplots can give interesting insights about numerical variables. We first take a look at `P_closenext_AUT_EHP` and `P_closenext_IARD_EHP` distributions, in the case of car insurance (on Figure 2.9). These variables represent the number of months before the end of the next contract (car insurance contract for `AUT` and any insurance for `IARD`). The first thing that is interesting to notice is that the distribution of `P_closenext_IARD_EHP` is right-skewed for the car insurance Single View. Without a good knowledge in the domain, it is not possible to determine clearly the reason of this skewness, but there are two reasons that come to mind: Either most customers decide to sign up for `IARD` contracts that don't last too long, or the fact that `IARD` contracts include most of the contract types means that there are more chances for a customer to have a side-contract that is ending soon. Since `IARD` contracts include Car and Fire insurance contracts too, the second reason is quite plausible as it is not uncommon for a customer to have multiple contracts.



(a) Count of churn occurrences based on `P_closenext_AUT_EHP` (Car insurance, April 2022)



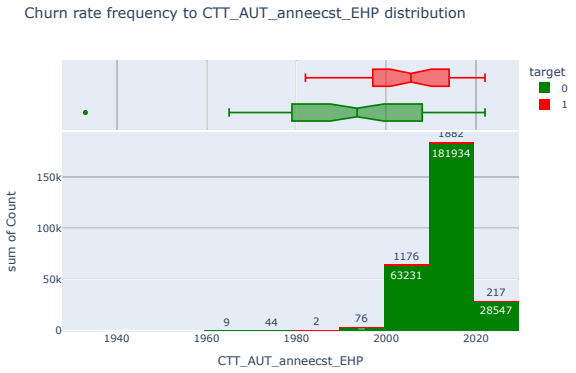
(b) Count of churn occurrences based on `P_closenext_IARD_EHP` (Car insurance, April 2022)

Figure 2.9: Comparison of churn distribution for Car insurance (`P_closenext_IARD_EHP` and `P_closenext_AUT_EHP`).

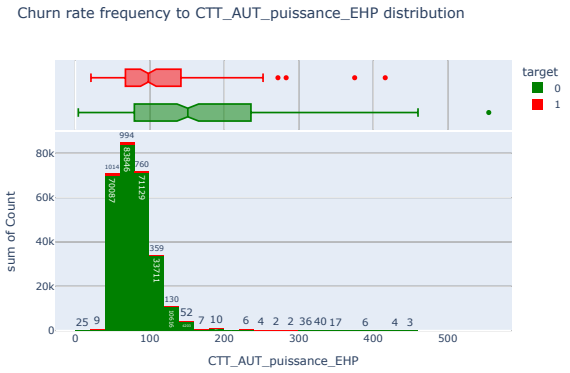
Let us now have a look at the boxplots of the distributions of churn for positive (red boxplot) and negative (green boxplot) based on the distribution of the exposed variables. As done in the previous section (Section 2.2.4), by looking individually at each numerical value on the graph, it is possible to observe some changes in churn rate. For example, in the case of `P_closenext_AUT_EHP`, we observe that customers that have from 2 to 3 months left until the end of their car insurance contracts are more prone to churn than customers who only have 0 to 1 month left ($806/48729 = 1,65\%$ vs. $582/49850 = 1,17\%$). However, if we take a look at the boxplots of the distribution of churn with respect to the variables distribution, it can be seen that the boxplots are similar. This signifies that the positive and negative churn cases have similar distributions among the variables that are being exposed. Now this is not much problematic since as said earlier, the individual changes between values can lead to interpretation. But if we wish to have a more global approach and be able to interpret the changes in a variable as a whole, we should look for distributions of positive and negative churn that are not similar.

To illustrate the previous statement, let us look at two variables that are specific to the Car insurance contracts (from the `Contracts` subset of features): `CTT_AUT_anneecst_EHP`, which is defined

as the year of construction of a vehicle and `CTT_AUT_puissance_EHP` which is defined as the power of the most powerful insured vehicle. The distribution are displayed on Figure 2.10. In the case of `CTT_AUT_anneecst_EHP`, we can see that both distribution are slightly left-skewed with more insured vehicles that were constructed in the recent years. Additionaly, we see that the distribution of positive churn rates (red boxplot) is more left-skewed by simply comparing the positions of the medians in the positive case interquartile range and the negative case interquartile range. A relation between the distribution of `CTT_AUT_anneecst_EHP` and the distribution of churn can thus be enunciated: Customer with vehicles that were constructed recently tend to churn more than customer with vehicles that were constructed a long time ago. In the case of `CTT_AUT_puissance_EHP`, there is an observable positive (right) skew that is even more marked in the positive churn cases. The underlying assumption of this observation is that vehicles that are insured are not too powerful most of the time and among these vehicles, the occurrences of churn are more important among the less powerful vehicles.



(a) Count of churn occurrences based on `CTT_AUT_anneecst_EHP` (Car insurance, April 2022)



(b) Count of churn occurrences based on `CTT_AUT_puissance_EHP` (Car insurance, April 2022)

Figure 2.10: Comparison of churn distribution for Car insurance (`CTT_AUT_anneecst_EHP` and `CTT_AUT_puissance_EHP`).

The analyses that were done in the previous sections could be done for multiple features, but the interesting bits of analysis will come with more efficient techniques. There exists tools that are much more efficient to perform global analyses. One of them involves the use of random forests to rank the different features and avoid having to go through all of them one by one (see Section 2.2.10).

2.2.6 Correlation structure

Section 2.2.4 and 2.2.5 gave us an incentive about the potential direct relations between Single View features and churn. Let us now explore the relation between the features. For this purpose, three statistical tests are performed: Correlation test (between numerical variables), Chi-squared test (between categorical variables) and Analysis of variance (ANOVA) test (between numerical and categorical variables). Each test is given a minor mathematical introduction and then it is performed on the group of variables that are concerned. Eventually, compelling relations between variables are detected and a strategy to deal with multicollinearity is established.

Note: The following discussions will take as a reference the 3 Single Views of April 2022. The relations might differ depending on the insurance type. Some relations might appear in some cases and they might not be verified in other cases. In order to cover most of the features in most situations, the 3 statistical tests are each performed on a different insurance type. The correlation test takes as reference the Car Insurance Single View, The Chi-Square test takes as reference the Fire Insurance (tenant) Single View and the ANOVA test takes as reference the Fire Insurance (owner) Single View.

Correlation test

Before performing each of the statistical tests, one should be comfortable with the notion of hypothesis test and p-value. In any hypothesis test, there is a null-hypothesis and an alternate hypothesis. Most of the time, the null-hypothesis states that two measured experiences involving some variables are not related at all while the alternate hypothesis states that they are in some way. The p-value is defined as the probability of observing data at least as favorable to the alternative hypothesis as our current data set if the null hypothesis is true. In other words, it is a way of quantifying the strength of the evidence against the null-hypothesis and in favor of the alternate hypothesis.

Let us now introduce one of the most used correlation test. This test will tell us how closely the distribution of two features fit on a line. The Pearson's Correlation Coefficient of two random variables X and Y is defined as

$$r_{XY} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}, \tag{2.1}$$

where μ is the mean and σ is the standard deviation of a random variable. Because of this definition, the coefficients take their value in the interval $[-1, 1]$. A positive coefficient indicates that the value of the slope of the regression line that approximates at best the distribution of both random variables is positive and the same goes for a negative coefficient with a negative slope. Intuitively, a positive coefficient means that both variables increase and decrease proportionally while a negative coefficient means that X decreases as Y increases and vice versa. From this coefficient, one can calculate the t-statistic that is defined as

$$t = r * \sqrt{\frac{n-2}{1-r^2}}, \tag{2.2}$$

where n is the sample size. The p-value can thus be computed as such (assuming a two-sided distribution):

$$p = 2 * \min\left(P(T \geq t|H_0), P(T \leq t|H_0)\right), \tag{2.3}$$

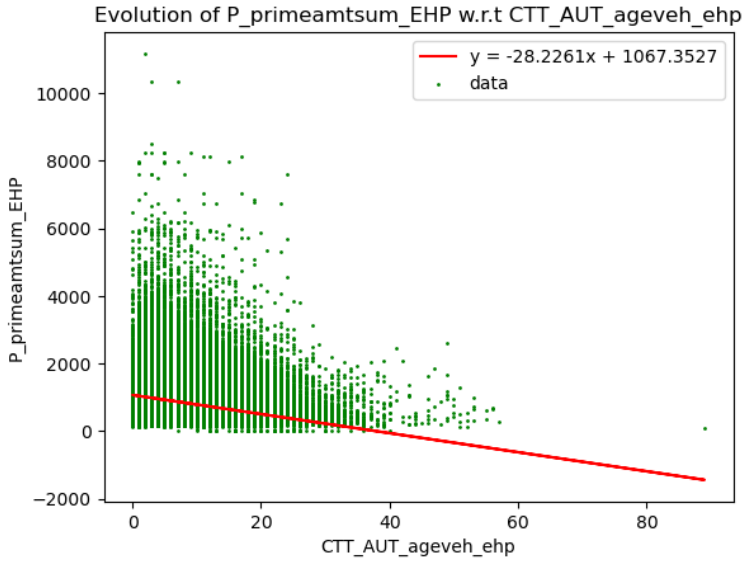
where H_0 is the null-hypothesis. If p is lower than a critical value (often chosen as 0.05), then the null-hypothesis can be rejected and there is statistical significance. Here, t is obtained by using the t-statistic, but it can theoretically be obtained using any statistical test.

Figure 6 of the Appendix .3.2 shows a heatmap of the pair-wise correlations coefficients (w.r.t Pearson's definition) of all numerical features. It can be seen that there are definitely features that are pair-wise related and exploration could lead to interesting insights. Here there is a need to apply a filter to avoid having to go through all the coefficients. This is why a sample of interesting correlations is extracted from the whole set of correlations disposed on Figure 6. To detect interesting correlations, one should take absolute values that are important enough to be considered as interesting for a deep observation, but not too close to 1, as we would be facing multicollinearity. The latter problem is tackled in Section 2.2.9. As an arbitrary choice, we will take a look at correlations whose absolute value range from 0.3 to 0.7. A correlation whose absolute value is less than 0.3 is considered not interesting and a correlation whose absolute value exceeds 0.7 is considered as too strongly correlated.

Table 7 lists all the pairs of features whose absolute correlation coefficient is between 0.3 and 0.7. Among these values, we will select 3 pairs (one with a moderate correlation, one with an average correlation and one with a strong correlation) and take a look at how each variable behaves w.r.t the other variable of the pair. It is mandatory to keep in mind that Table 7 contains all the correlations that were defined as "interesting" w.r.t the arbitrary range that was defined earlier, but some of them remain not much interesting for analysis, so the three examples will be chosen accordingly. The 3 examples are pairs of features that don't belong to the same subset of features as described in Section 2.1. This choice is motivated by the fact that some subset of features such as `Product` or `Contract` contain variables whose relations with other variables are already given by their definition. For instance, the variable `CTT_AUT_Value_EHP` (Defined as "Vehicle Value") is calculated based on most of the features of the `Contract` subset (and some features of the `Product` subset). Still, if the analysis

was centered around CTT_AUT_Value_EHP, it could be interesting to observe the value of each the correlation coefficients of its pairs to determine which are the most taken into account when computing CTT_AUT_Value_EHP. For features that are dependent of multiple (at least 5) features in Table 7, the Table is a good indicator of how they depend on each other.

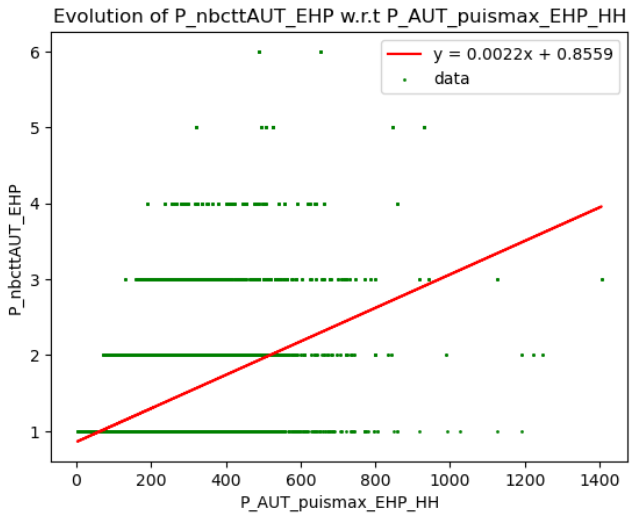
Let us first consider the pair of features (P_primeamtsum_EHP,CTT_AUT_ageveh_ehp)that has a correlation of -0.311041. The first feature is defined as the total sum of bonuses of IARD type contracts and the second feature is defined as the age of the vehicle. A negative correlation coefficient indicates that the more one of the feature increases in value, the more the other feature tends to decrease and vice-versa. Figure 2.11 is a visualization of the evolution of both features, along with a linear regression of the available data. The slope of the linear regression is indeed negative but quite gentle, and this is a good occasion of observing some anomalies or outliers. For example, there is an occurrence of a vehicle that has an age of more than 85 years while most of the ages do not exceed 60 years. There are also 3 occurrences of bonuses that exceed 10000€ while most bonuses do not exceed 8000€. Such outliers can influence greatly the issue of hypothesis tests and are investigated more deeply in Section 2.2.12. In the time being, the focus is on the distribution of features and not single values.



Feature	Mean	Standard deviation
P_primeamtsum_EHP	815.55	493.59
CTT_AUT_ageveh_ehp	8.92	5.44

Figure 2.11: Relation between P_primeamtsum_EHP and CTT_AUT_ageveh_ehp (Car insurance, April 2022).

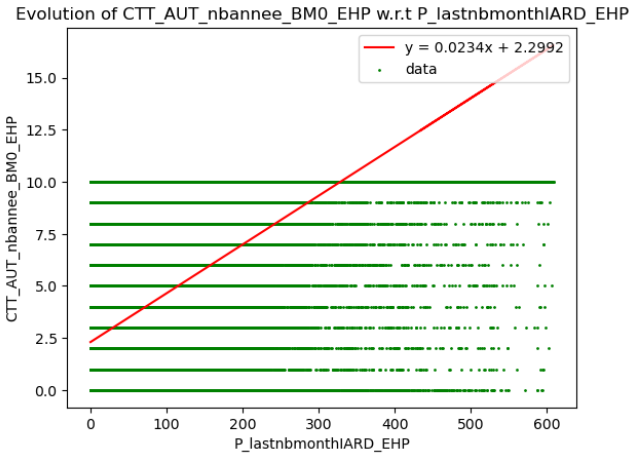
The next pair is (P_nbcttAUT_EHP,P_AUT_puismax_EHP_HH) with a correlation coefficient of 0.453619. P_nbcttAUT_EHP is defined as the number of car contracts and P_AUT_puismax_EHP_HH is the power of the most powerful insured vehicle **on household view**. The last notion is important to highlight: We know for a fact that the pairs of features that include one feature and the corresponding feature on household view induce strong correlations (see Section 2.1). However, the slight differences between the two features of the pair can have different impacts when it comes to a comparison with other features. In fact, the correlation that is considered here is equal to **0.453619**, but if the feature on household view is swapped with the feature on single customer view (P_AUT_puismax_EHP), the correlation coefficient skyrockets to **0.68**. Such a leap in value shows that it is sometimes interesting to separate the single customer view and the household view, as they sometimes lead to distinct relations. Figure 2.12 shows that the slope of the linear regression for the forementioned pair is positive and we observe a high number of pairs of low power vehicles with a low number of car contracts. Unfortunately, there are not much possibilities of values of P_nbcttAUT_EHP so the data points tend to stack, hence the need to fit a simple linear regression line on the distribution.



Feature	Mean	Standard deviation
P_nbcttAUT_EHP	1.1	3.3
P_AUT_puismax_EHP_HH	8.92	5.44

Figure 2.12: Relation between `P_nbcttAUT_EHP` and `P_AUT_puismax_EHP_HH` (Car insurance, April 2022).

One of the strongest interesting relations found during the analysis is between `CTT_AUT_nbannee_BM0_EHP`, the number of years in the `BM0`[1] insured drivers category, and `P_lastnbmonthIARD_EHP`, the number of months since the most recent signed `IARD` type of contract (all contracts that are concerned by this study). The coefficient of **0.550872** can be explained by the fact that both features are time-dependent and reflect some sort of seniority in the customer database. Figure 2.13 is again quite messy as the number of different values taken by `CTT_AUT_nbannee_BM0_EHP` is low. there again, fitting a linear regression model on this data provides an increasing line that approximates the distribution.



Feature	Mean	Standard deviation
CTT_AUT_nbannee_BM0_EHP	4.31	4.24
P_lastnbmonthIARD_EHP	86.97	99.6

Figure 2.13: Relation between `CTT_AUT_nbannee_BM0_EHP` and `P_lastnbmonthIARD_EHP` (Car insurance, April 2022).

All of the 3 examples are approximated with a linear regression model that is without a doubt very coarse in most situations. This Section does not go deep into machine learning models, so this is sufficient for our analysis (see Section 3.2 for further analysis of machine learning models). The

[1]Unfortunately, no further information has been disclosed about Car insurance categories

correlation is a good measure of how likely an unobserved data point is going to fill in the distribution of the observed data. However, one must note that the residuals of a prediction obtained via a machine learning model are not a function of the correlation. The correlation is a good indicator of multicollinearity and it can give good insight about relations between variables. But when it comes to performing predictions on unseen data, high correlations can lead in larger residuals than with low correlations. This is a direct consequence of multicollinearity and the issue is tackled in Section 2.2.9.

Chi-squared test

The following hypothesis test is now performed with as reference the Fire Insurance (tenant) Single View. In the previous section, the pairwise correlation coefficients between numerical variables were observed. The definition of the correlation coefficient implied that the mean and standard deviation of the variables must be known. Categorical features, in their form of origin, cannot provide such statistical features, as they are not numeric. For such variables, it is mandatory to use another type of hypothesis test that does not require the presence of means and standard deviations. Pearson's chi-squared test is an evaluation method that is useful in 3 situations: A test of goodness of fit (checking if an observed frequency matches a well-known distribution), a test of homogeneity (comparison of two or more groups using the same categorical variable) and the test of independence. The chi-squared test of independence is a hypothesis test that is based on the contingency table of two random variables X and Y. The contingency table of two random variables X and Y is defined as a mxn matrix, where m is the number of categories taken by X and n is the number of categories taken by Y such that

$$CT_{XY} = \begin{pmatrix} o_{1,1} & o_{1,2} & ... & o_{1,n} \\ o_{2,1} & ... & & ... \\ ... & & ... & ... \\ o_{m,1} & ... & ... & o_{m,n} \end{pmatrix}. \tag{2.4}$$

In the matrix 2.4, $o_{i,j}$ is the number of occurrence of the pair $\{i, j\}$ and is defined as the observed frequency of the pair $\{i, j\}$. The expected frequency of the same pair is noted $e_{i,j}$ and is computed as the frequency of the pair under the assumption that the null-hypothesis is true. Chi-Square is thus given by the squared differences between the observed and expected frequency such that

$$\chi_c^2 = \sum_j^N \frac{(O_j - E_j)^2}{E_j} \tag{2.5}$$

Where c is the number of degrees of freedom which is computed as $(nbCategories_X - 1) * (nbCategories_Y - 1)$ and O_j is the sum of all observed frequencies of a column of the contingency table

$$O_j = \sum_i^M o_{i,j} \tag{2.6}$$

The same definition goes for E_j and $e_{i,j}$. As with the correlation test, the p-value can be deduced from χ^2.

Let us now perform the chi-square test of independence on the categorical values of the Fire Insurance (tenant) Single View of April 2022. We choose 0.05 (arbitrarily) as critical value for the p-value: Every pair of feature whose p-value exceeds 0.05 cannot reject the null-hypothesis to a 95% level of confidence, hence they are declared as independent. Table 7 gives the Chi-Square Distribution Table w.r.t the chosen degree of confidence and the degrees of freedom. Along with the results of the chi-square tests of pairs of categorical variables in Table 8 (the table is a compilation, not all results are displayed), one can check whether or not two categorical features of the set are totally independent or not. Unfortunately, the chi-square test of independence is not as intuitive as the correlation test: In a correlation test, a high correlation coefficient could be interpreted as a strong dependence between numerical variables. In the case of a chi-square test, a high value of χ^2 can only be interpreted as a strong rejection of the null-hypothesis, i.e a strong rejection of independence. What is meant is that

the value of χ^2 can not be interpreted as intuitively as the value of a correlation coefficient. A high value of chi-square does not necessarily mean a high dependence of both variables, as its value is largely influenced by the dimensions of the problem (sample size, categories) and the difference between observed and expected frequencies.

Table 8 of the results of the chi-square tests applied to the Fire Insurance (tenant) contracts is sorted by p-value in descending order. The first pairs that appear in the table are the ones that are the most likely to not reject the null-hypothesis, i.e the hypothesis saying that the two features of the pair are independent. There are two ways of verifying whether or not two variables are independent. One can directly look at the p-value and if it is higher than 0.05, not reject the null-hypothesis. One can also look at the value of chi-square and degrees of freedom (dof) and, using the chi-square distribution table, verify if, for a given significance level (0.05 here), the value of χ^2 is higher than the value in the table. For example, `P_cdtypsorASS_HP` (Type of last `IARD` contract exit) and `P_lasttypeIARD_EHP` (Type of most recent `IARD` contract started) have a chi-square value of 1.605067 for 4 degrees of freedom. Since we chose a significance level of 0.05, Table 7 tells us that to reject the null-hypothesis with a 95% confidence level, the minimum value that is required for χ^2 is 9.488, which is not the case here. At the bottom of the Table 8, the pairs of variables have p-values that are quite close to 0 and even sometimes equal to 0. The features in these pairs are most likely dependent even though we can't assure it. For instance, the last pair of the table is `C_enfants_EHP` (if the customer has children or not) and `C_etatcivil_EHP` (marital status) with a χ^2 of 5485 with only 5 degrees of freedom and a p-value equal to 0. The relation between these two variables is quite founded but we see some other pairs whose relation can sometimes be more ambiguous such as `C_sexe_EHP` (customer's gender) and `S_ASS_nbtot_HP` (Number of declared accidents during historical period) with a p-value of $7.335791e-04$.

Another interesting aspect of the chi-square test of independence is that it allows to detect the absence of direct relation between the categorical features and the target variable. The target variable is boolean, hence it is categorical. For example, we see that `target` and `P_basket_inc_EHP` have a p-value of 0.965, which is equivalent to a 96,5% chance of not rejecting the null-hypothesis. Despite that, one should note that the previous test is only a test of direct independence: Categorical features that have a p-value of more than 0.05 with the target variable are not necessarily useless when it comes to building a model that has the goal to predict a target output. Machine learning models do not result from simple direct relations between variables and the output. They have the capability to capture complex relations between groups of features and the target variable. What chi-square tests with the target variable give us is a simple intuition of variables that could not be of significance when a predictive model is built. We could consider that building a model that puts more importance on `P_cdtypsorVIP_HP` rather than on `P_basket_inc_EHP` might lead to better performance, based on the p-value of their chi-square test with the target variable, but again this is just an assumption.

Analysis of variance (ANOVA)

The Correlation test allowed us to determine some levels of dependency between numerical features. The chi-square test of independence allowed us to check if pairs of categorical features could reject total independence. We now want to be able to observe if there is statistical significance between numerical and categorical features. For that purpose, one can monitor the changes in means of a numeric variable among the different groups of a categorical variable (or multiple categorical variables). This test is known as the Analysis of variance (ANOVA) and is said to be "one-way" if there is only one categorical feature and "two-way" if there are two. A two-way ANOVA test is quite interesting in the situation where we have two categorical variables that are known to have a minimum of interaction. Instead of performing two separate one-way ANOVA, we test them both at the same time to assess both their individual effects and their interaction effect on the dependent (numerical) variable. In the first time, we will take a look at some interesting results of one-way ANOVA tests with the Fire insurance (owner) Single View and in a second time, we will use the previous insights to proceed to specific two-way tests that could be of relevance for analysis.

ANOVA tests come with a set of assumptions that are crucial. If these assumptions are violated, the results of the tests may not be accurate. If the data that is used for the ANOVA test does not meet the requirements, it should be transformed. The 4 assumptions of an ANOVA test are the following:

- Independence: The observations within each group are independent of each other.
- Normality: The dependent (numerical) variable should follow a normal distribution within each group.
- Homoscedasticity (homogeneity of variances): The variances of the dependent variable should be equal across all groups.
- Interval or ratio scale: ANOVA assumes that the dependent variable is measured on an interval or ratio scale.

The ANOVA test is performed as follows. For a categorical variable c with k different groups and a dependent numerical variable n, once the null-hypothesis (H_0: all group means are equal) is stated:

1. Calculate the group means for each value of the categorical feature μ_1,μ_2,...,μ_k.
2. Calculate the overall mean μ_T
3. Calculate the variability between group means vbg. This is done by computing the sum of squares between groups:
$$VBG = \sum_{i}^{k} n_i * (\mu_i - \mu_T)^2, \tag{2.7}$$
where n_i is the number of observations in group i.
4. Calculate the variability within the groups :
$$vWG = \sum_{i}^{k} \sum_{j}^{N_i} (x_{i,j} - \mu_i)^2, \tag{2.8}$$
where $x_{i,j}$ are the observed values in group i (number of observations in the group: N_i)
5. Determine the degrees of freedom between (dfB) and within the groups (dfW). It is equal to k-1 between the groups and N-k within each group where N is the total number of observations.
6. The F-value can be obtained the mean square between and within groups:
$$F = \frac{MSB}{MSW} = \frac{VBG}{dfB} \frac{dfW}{VWG} \tag{2.9}$$

Once obtained, just like chi-square, the value can be either used to determine the p-value or it can be compared with a critical value using the F-Distribution with the appropriate degrees of freedom.

In the context of Fire Insurance (owner), the Single Views contain 32 numerical features and 167 categorical features (as provided by Table 2.2). Performing a one-way ANOVA test with each pair of variable would result in a total of 5344 tests and again going through most of them would be laborious, so we won't explore too much. Instead, Table 2.5 compiles a small fragment of these tests that contain interesting results. It is simpler to directly look at the p-values in this case again to keep the table as light as possible (there is no need to display dfB and dfW). The f-value is solely provided for guidance.

Pair of features	f-value	p-value
C_surveillance_EHP_HH and P_INC_flagvol_EHP	0.231814	7.930938e-01
C_notel_EHP_HH and S_ASS_nbtot_HP	1.123279	3.434655e-01
C_nomail_EHP_HH and S_ASS_nbtot_HP	1.518584	1.447019e-01
P_NBCTTSDSPAR_HH_EHP and P_nbcttAUT_EHP	1.865263	8.263958e-02
P_NBCTTSDSPAR_HH_EHP and C_langue_EHP	3.021792	4.871634e-02
C_nomail_EHP_HH and C_epub_EHP	44.353512	1.213923e-28
C_taillefam_EHP and C_langue_EHP	128.689149	1.418185e-56
C_robinson_EHP and P_nbcttAUT_EHP	83.508960	7.070359e-105
C_anciennete_EHP_max_HH and P_nbcttDCO_EHP	367.840531	0.000000e+00
C_anciennete_EHP_max_HH and P_nbcttPFI_EHP	273.548225	0.000000e+00
C_anciennete_EHP_max_HH and P_nbcttAUT_EHP_HH	1011.018067	0.000000e+00

Table 2.5: Interesting results of ANOVA tests (Fire Insurance (owner), April 2022)

Among all ANOVA tests, we detect a lot of relations between features that belong to `Products` and `Features`. The relations between features that involve seniority and the number of contracts seem to be quite strong. We could notice it by simply looking at the bottom of the table and observing that most pairs are related to these 2 groups of features.Looking at the top of the table, `S_ASS_nbtot_HP` (number of declared accidents) that was previously exposed seems to have no relation with `C_nomail_EHP_HH` and `C_notel_EHP_HH` (flags that indicate the presence of email address and phone number on household view). In the middle of the table, we observe that `C_langue_EHP` (language spoken by the customer) could be somehow related to the size of his family C_taillefam_EHP and his number of contracts of healthcare insurance on household view `P_NBCTTSDSPAR_HH_EHP`. Stating both relations separately does not make much sense, but knowing that the size of the family of the customer could be related to the number of health insurance contract for his household, we could find sense in the relations that would be indirect. Basically, most of the time, the fact that a p-value is less than 0.05 can not be interpreted directly. Instead, one should take groups of pairs with one or more common features and create clusters of features that could lead to interpretation. This is how relations can be found within and between groups of variables.

2.2.7 Data Transformation

For the remaining part of the Single View analysis, the data has to undergo a series of transformations depending on the situation. This section gives a few details about these transformations and in which context they are used.

Categorizing

In Python language, Categorizing refers to the process of assigning an integer value to each distinct instance of a feature (supposedly non-numerical). The integer values are incremental, with the first category starting at 0 (categories are sorted in alphabetical order). Missing values are given the value -1. Each non-numerical feature is categorized using this mapping. For instance, the marital status variable `C_Etatcivil_EHP`, which has 6 different values, is mapped as follows:

$$\begin{aligned} Celibataire &\to 0 \\ Cohabitant &\to 1 \\ Divorce &\to 2 \\ Marie &\to 3 \\ Separe &\to 4 \\ Veuf &\to 5 \\ NULL &\to -1 \end{aligned}$$

This transformation is very relevant to use, because even though strings are mapped to integers, the features are still treated as particular discrete features by Python, meaning that this mapping doesn't induce any order of importance between the different values. This transformation is necessary in many cases, as it allows to have a Single View that consists of 100% of numerical data. When it comes to computing spatial distances with algorithms such as K-nearest neighbors (used in Imputation and Clustering), the transformation is mandatory.

Standard Scaling

Some of the tools used for data analysis involve using and fitting models to some extent. Data that is not preprocessed can sometimes cause problems in the fitting process, as the impact of the different features on the model is quite different because of the scale of the numerical values. In these cases, data standardization/normalization is required in order to give each feature a chance to compete one against another. The goal of Standard Scaling is to have the mean of each feature equal to 0 with a standard deviation of 1. For this purpose, the mean μ and standard deviation σ of each column is computed. Each scaled value z is such that (for an initial value x) :

$$z = \frac{x - \mu}{\sigma} \tag{2.10}$$

It is important to note that both transformations are reversible and, in some cases, for the sake of interpretability, they should absolutely be reverted. It is the case for variable imputation. If a variable is categorized and missing values are imputed using this categorization, considering that the imputation process does not synthesize undefined values w.r.t the mapping (for example, non-discrete values), the mapping of the categorization should be kept in memory in order to revert the transformation and observe the potential changes in the distribution. The same goes for standard scaling, for which the parameters of the normalization should be kept in order to perform denormalization.

2.2.8 Variable Imputation

The different hypothesis tests were performed on data in its most natural form in order to get a first faithful insight about the features. Before moving on with the rest of the analysis, it is important to address the extreme levels of missingness that are exposed in Section 2.2.3. Addressing imputation before dealing with problems such as multicollinearity or outliers is a good way of enabling a pre-imputation of the Single Views. This pre-imputation is mandatory to be able to observe potential outliers, as the fraction of observations that are complete is extremely low and thus, keeping only the complete observations is not representative of the data at all.

In statistics, imputation is the process of replacing missing data with substituted (or synthesized) values. Adding synthetic information to data that is considered truthful to reality adds bias and reduces overall analysis efficiency. Hence, a baseline for a good imputation strategy requires good insights about the nature of the data. Imputation also requires validation, i.e observing how the changes operate on the statistical features of the variables (mean, median, percentiles, variance and so on) and if the overall initial representation of the data is not too much denatured by the different methods.

In this section, the different methods that were used to impute data are discussed with as reference the Car Insurance Single View of April 2022 (to use the summary statistics detailed in Appendix 5). These methods range from very simple and naive techniques (such as median or mean imputation) to more complex techniques (regression or machine learning techniques such as iterative imputation). Each of these techniques are tested on the features and the changes in statistic values are observed.

Statistical features

Naive imputation methods consist in replacing the missing values based on statistical features. Mean/median imputation can be performed on numerical features by replacing missing values by the mean/median of

the observed values. When it comes to categorical features, one can use the top value (most recurrent value) as a reference. In the context of Single Views, the missing features are either categorical or discrete. Discrete variables can be imputed using the median and categorical variables will be imputed using the most frequent value. `P_VIP_qualite_EHP` is a discrete value that takes either the value 0,1 or 2 and whose median is 1. Figure 2.14 show that the high missing ratio of `P_VIP_qualite_EHP` has a huge impact on the distribution of its value, as the 71,29% of missing values are imputed as 1. This simple example shows that using single values to impute highly missing features can distort the data a lot.

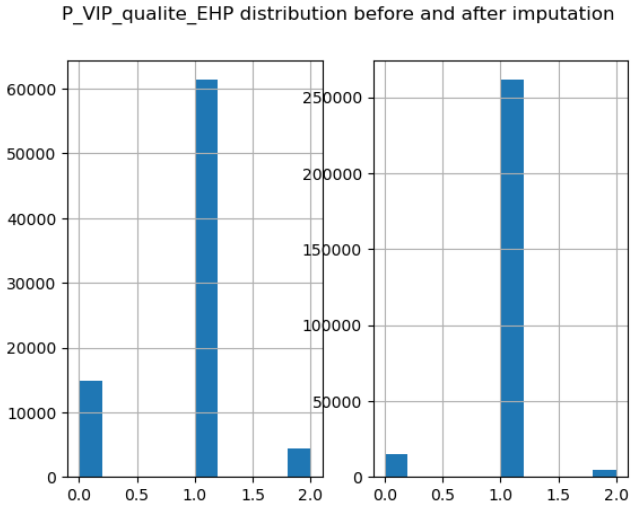


Figure 2.14: Distribution of `P_VIP_qualite_EHP` before and after imputation (Car insurance, April 2022).

We now test the "most frequent" method on two categorical features : `C_epub_EHP` which has a missing ratio of $9,67\%$ and `P_cdtypsorASS_HP_HH` which has a missing ratio of $96,31\%$. The histograms on Figure 2.15 show that depending on the missing ratio, the impact can be important, as it can completely change the distribution of the variables.

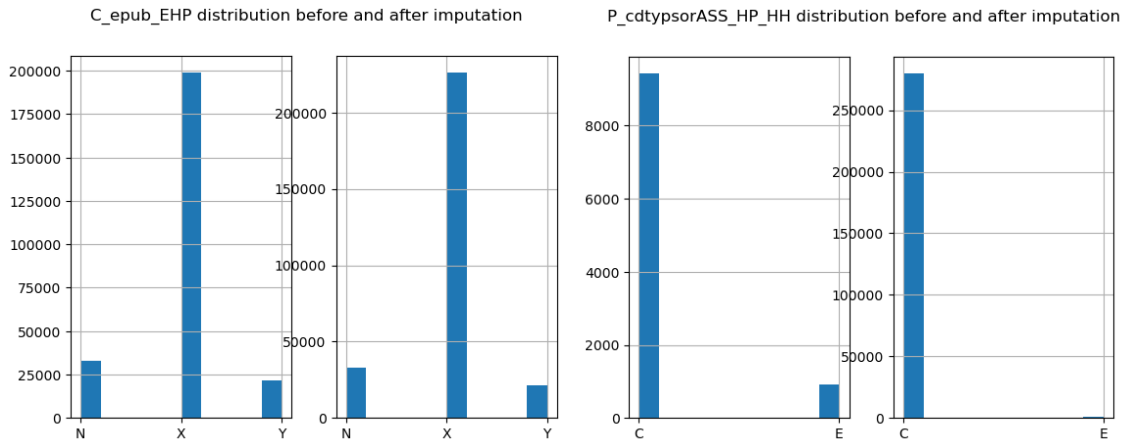


Figure 2.15: Distribution of `C_epub_EHP` and `P_cdtypsorASS_HP_HH` before and after imputation.

The conclusion is clear, simple methods based on statistical features cannot be used in the context of largely missing data. The distribution of the dominant variable values increase a lot while the minority values don't, which creates an ever more unbalanced distribution. It is thus required to address the matter with more complex ideas. Still, such coarse imputation techniques could work with a dataset with a low percentage of missing values.

K-nearest neighbors imputation

Before going on with the imputation techniques, one should note that applying machine learning techniques to imputation comes with an assumption about the pattern of the missingness. The data can either be missing partially or completely at random (MAR/MCAR) or not (MNAR). KNN imputation assumes that the data is MAR/MCAR, which, as we have seen in Section 2.2.3, is not the case.

The K-nearest neighbors method is a machine learning algorithm that labels an observation point based on the labels of its k closest points in the input space. In classification problems, it is mostly used to classify unobserved data points based on the observed points. The KNN algorithm requires to choose a distance metric that fits the problem. Here, as long as the data is normalized, we stick to the euclidean distance. With missing data, the euclidean distance between two data points X and Y can be computed as follows:

$$d_{XY} = \sqrt{weight * squared\_distances}, \tag{2.11}$$

where weight is equal to the total number of coordinates (i.e. the size of the input space) divided by the number of non-missing coordinates. For example, in a input space of size 3, the euclidean distance between $X = (3, NA, 5)$ and $Y = (1, 0, 0)$ is

$$\sqrt{\frac{3}{2}\{(3-1)^2 + (5-0)^2\}} = 6.59 \tag{2.12}$$

Once a distance metric has been chosen, each data point with missing values chooses its k nearest neighbors that have non-missing values for the specific variables that are missing and use the distance as weights to determine the value that should be imputed. This method is not expected to work efficiently, as it assumes random missing patterns (MCAR/MAR), which is not the case here. It is however interesting to test it, as bad results could reinforce the observations about missing patterns that were made in Section 2.2.3.

Iterative Imputation

Iterative imputation is a multivariate imputation technique that uses complex machine learning concepts for synthesizing missing data. In term, this is the method that will be kept for imputing all of the variables in the dataset as we will see that it is quite more efficient than the other techniques.

Iterative imputation uses the principle of chained equations to obtain imputed values (more details are given in this paper[8]). At each iteration, features with missing data are selected as the dependent variable one after another and the other features are used as predictors to build an imputation model. This imputation model uses the observations that contain observed data for the current dependent variable to be trained. This process is done for each variable with missing data and it is then repeated usually from 10 to 20 times to account for uncertainty and reach a certain level of convergence. To build the imputation model at the start, missing values in the predictors are initialized either randomly or based on some strategy. The final imputation is a combination of the different imputed datasets that uses a set of rules to take into account every imputation (known as Rubin's rules).

Mathematically, chained equations imputation works as follows: For every variable with missing values, the algorithm requires an imputation model (which is typically a simple regression model) and a prior distribution for the model's parameter. Let X_{-j} be the vector of random variables that excludes the variable X_j and $x_{-j} = (x_{-j}^{obs}, x_{-j}^{mis})$ the matrix of all observations x corresponding to X_{-j} (missing observations are included). We also note $p(x_j|x_{-j}, \psi_j)$ as the probability distribution function of the imputation model for X_j and $p(\psi_j)$ for the prior distribution of the unknown parameter ψ_j. Before starting the iterations, the missing values x_{-j}^{mis} in x_{-j} are replaced by either the mean, the median, the most frequent occurrence or a random observed value. In our case, since most missing variables are categorical or discrete, the strategy used is "most-frequent". Imputations are drawn from the posterior predictive distribution of the missing data given the observed data, noted $p(x^{mis}|x^{obs})$. Under the ignorability assumption (once observed variables are taken into account, the treatment of the missing

values is independent of potential outcomes, i.e it ensures that the treatment is randomized), this distribution is equal to

$$p(x^{mis}|x^{obs}) = \int p(x^{mis}|x^{obs}, \psi)p(\psi|x^{obs})d\psi. \tag{2.13}$$

At each iteration, there are thus two steps for each missing variable: First draw ψ_j^* from the posterior distribution that is proportional to $p(\psi_j)p(x_j^{obs}|x_{-j}^*, \psi_j)$ and secondly, draw missing values from the predictive posterior $p(x_j^{mis}|x_{-j}^*, \psi_j^*)$ (which is obtained when the previous iteration is completed). If we have a total of K variables with R variables that are incomplete ($R \leq K$), given the imputations from last draw, the drawing of the next iteration will thus take the form (for each variable X_j such that $1 \leq j \leq R$)

$$\psi_j^{(t)} \sim p(\psi_j)p(x_j^{obs}|x_1^{(t)}, x_2^{(t)}, ..., x_{j-1}^{(t)}, x_{j+1}^{(t-1)}, ..., x_R^{(t-1)}, x_{R+1}, ..., x_K, \psi_j) \tag{2.14}$$

$$x_j^{mis(t)} \sim p(x_j^{mis}|x_1^{(t)}, x_2^{(t)}, ..., x_{j-1}^{(t)}, x_{j+1}^{(t-1)}, ..., x_R^{(t-1)}, x_{R+1}, ..., x_K, \psi_j^{(t)}) \tag{2.15}$$

Chained equations use the results of the previous iteration and the previous results of the current iteration to draw values. The final iteration gives the imputations that will form the imputed dataset. To account for uncertainty, the process is repeated multiple times with different initializations and the resulting datasets are merged.

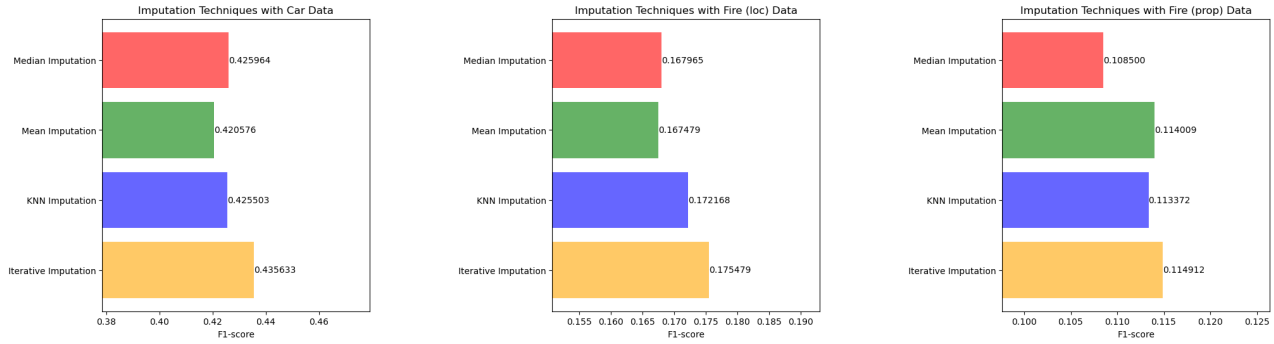
Before applying Iterative Imputation on the whole set of features of the Single Views, it is mandatory to note that, even if the initial strategy was to replace missing values with the most frequent observed value, the nature of some features might be lost within the process if the type of model used for imputation is not appropriate. Using regression models for categorical and discrete variables is irrelevant, as it outputs continuous values that cannot be mixed with discrete values without distorting the nature of the variables. There are thus two possibilities here: Either we use models that can handle categorical data such as models based on decision trees, or we transform categorical data by encoding it to a format that can be handled by regression models. The possibility of using decision tree-based models was chosen first during the analysis, but because of the high memory usage of these kind of models, the resources available to perform the tests were insufficient. The tests are performed on a local machine with a local environment that has no access to any sort of cloud technology, so the memory usage was limited. The second option is thus chosen. The data is encoded using the same dummy encoding as presented in Section 2.2.9. With the encoded data, one can use a simple regression model as imputation model.

Comparing the different methods

The comparison of the different imputation methods is made using the three available Single Views of April 2022 (for the three insurance types). For the sake of efficiency and memory usage, a fraction of each Single View is selected (at random) instead of the whole Single View for each insurance type. The selected fraction of the Single View contains a balanced number of positive and negative churned customers. This choice is motivated by the fact that the comparison of the imputation techniques relies on cross-validation using a simple classification model that would not be much efficient if the selected fraction was too unbalanced. As a consequence, the size of the subsets used for the different insurance types are different based on the number of available positive churn cases.

The evaluation of each method relies on a cross-validation score that uses a Random Forest Classifier with the data once it has been imputed. The score is obtained by computing the F1-score of the classifier with the imputed data. Basically, such an evaluation method would show how the different imputation techniques impact the way a classification model could efficiently fit the Single Views. The F1-score is defined as a measure of classification accuracy and more details are given in Section 3.4. Looking at Figure 2.16, It is observed that the three insurance types are better performing when it comes to iterative imputation. It can also be seen that on average, KNN imputation does not outperform the naïve methods, and it might be explained by the false assumption made for performing this machine learning based imputation technique (KNN imputation assumes MCAR or MAR while we

have proven that we have MNAR). The difference in magnitude of the F1-score between the insurance types is a direct consequence of the size of the selected fraction of the Single View. The classifier that is used for each subset has the same architecture and no tuning has been performed, because the goal here was simply to show the difference in performance for each imputation method with a simple classification model. Moreover, the discussion about the different models used for classification does not take place here, but in Section 3. This comparison allows us to use iterative imputation for our future analyses without restraint, but let us not forget that it may cost more resources than a classic imputation method such as median or mean imputation.



(a) Car insurance, April 2022

(b) Fire insurance (tenant), April 2022

(c) Fire insurance (owner), April 2022

Figure 2.16: Comparison of the different imputation methods on the Single Views of April 2022.

2.2.9 Multicollinearity

In High-dimensional statistics, there is a possibility that, among the set of features that are used to model an output, some of them give information that is common and building a model using these features is equivalent to capturing information redundantly. This could lead to inflated prediction errors. Multicollinearity occurs when there is such a strong relation between two variables. It is mandatory to tackle this subject in the context of the Single Views, as we have seen that some features are directly dependent of others (see Section 2.1). In order to detect multicollinearity between numerical variables, a high correlation coefficient is a good indicator. Unfortunately, other hypothesis tests like the chi-square test or the ANOVA test are quite limiting when it comes to evaluating it, so it is best practice to transform categorical data in order to evaluate its potential multicollinearity.

Dealing with the multicollinearity issue is trivial when it comes to numerical variables. For each pair of variable, run a correlation test and, if the resulting coefficient exceeds a certain threshold (in absolute value), there is a risk of multicollinearity. Generally, an absolute value of 0.7 is chosen and this is also the threshold that was selected when running the correlation tests in Section 2.2.6. When all correlation tests are done,one should get rid of one of the two features of a pair that present a risk of multicollinearity. The choice of the feature that should be removed is based on potential overlapping. If there are features that overlap between pairs, the features that should be removed are the ones that are present in the highest amount of pairs. If there is no overlapping and all the problematic pairs have distinct features, the choice can either be arbitrary or based on the intention of keeping specific variables in the model (based on expertise).

In the case of categorical features, one should perform transformation on the data. If one could resort to the use of category transformation as depicted in section 2.2.7, it is sometimes more convenient to use methods that allow a direct observation of the relations between different values of a categorical variable and another variable. The problem of category transformation is that the mapping between categories and their numerical values is a black-box process and the only way to revert this mapping is to keep in memory the mapping for each feature. To avoid having to deal with this intermediate step, we introduce the use of dummy variables. They are binary indicators that indicate the presence

of a category: For a categorical feature with k categories, $k-1$ dummy variables are created (as the reference category is represented implicitly if all dummy variables are equal to 0). The value 1 is assigned to the corresponding dummy variable and 0 is assigned to the others. A regression model is then fit using the dummy variables as predictors. Multicollinearity can be then assessed using the Variance Inflation Factor (VIF), which is defined as (for one variable i)

$$VIF_i = \frac{1}{1 - R_i^2}, \tag{2.16}$$

where R_i^2 is the coefficient of determination for regressing the ith independent variable on the remaining ones. It is function of the target output's variance and the residuals of the fitted model. A VIF value of 1 signifies that there is absolutely no relation between the variable and the others, a VIF value between 1 and 5 means that there is slight correlation between the variables and a value of more than 5 indicates a high correlation. If the VIF exceeds 10, there is significant multicollinearity that cannot be ignored.

To illustrate this technique, let us try and fit a model using only categorical variables by using dummy variables. The selected variables are `P_sortie_TypeLastIARD_HP_HH`, `P_sortie_TypeLastIARD_HP`, `P_cdtypsorIARD_HP_HH` and `P_cdtypsorIARD_HP`. The following model is fit using the Fire Insurance (owner) Single View (April 2022). The first two variables have $k_1 = 4$ categories and the two last have $k_2 = 2$ categories, which makes a total of $2 * (k_1 - 1) + 2 * (k_2 - 1) = 8$ dummy variables. A model is fit for each dummy variable and the values of R-squared and VIFs are reported in Table 2.6.

Dummy Variable	R squared	VIF
P_sortie_TypeLastIARD_HP_HH_AUT	0.974273	38.870263
P_sortie_TypeLastIARD_HP_HH_DCO	0.966288	29.662591
P_sortie_TypeLastIARD_HP_HH_INC	0.975079	40.126883
P_sortie_TypeLastIARD_HP_AUT	0.973954	38.393758
P_sortie_TypeLastIARD_HP_DCO	0.966129	29.524202
P_sortie_TypeLastIARD_HP_INC	0.974811	39.699083
P_cdtypsorIARD_HP_HH_E	0.906013	10.639735
P_cdtypsorIARD_HP_E	0.906002	10.638525

Table 2.6: VIF values of the complete subset of features

As one can see, every VIF is above 10, meaning that there is significant multicollinearity. To get rid of it, we remove the dummy variables iteratively (w.r.t the VIF value) and observe the changes in value. `P_sortie_TypeLastIARD_HP_HH_INC` has a VIF of 40.13 so we get rid of it first. As we can see on Table 2.7, getting rid of it significantly reduced the VIF of `P_sortie_TypeLastIARD_HP_INC`(from 39.7 to 1.9) which makes sense since both variables represent the same category with the sole difference that one concerns the customer himself and one is on the household view. We have seen (see Section 2.1) that the relation between these features is strong. We also observe that the other categories of the same features have their VIF also decreased by a lot, but the VIFs of `P_cdtypsorIARD_HP` haven't much decreased.

Dummy variable	R squared	VIF
P_sortie_TypeLastIARD_HP_HH_AUT	0.952598	21.096184
P_sortie_TypeLastIARD_HP_HH_DCO	0.954953	22.199045
P_sortie_TypeLastIARD_HP_AUT	0.954164	21.816894
P_sortie_TypeLastIARD_HP_DCO	0.955568	22.506382
P_sortie_TypeLastIARD_HP_INC	0.472622	1.896173
P_cdtypsorIARD_HP_HH_E	0.905346	10.564806
P_cdtypsorIARD_HP_E	0.905435	10.574684

Table 2.7: VIF values once `P_sortie_TypeLastIARD_HP_HH_INC` is removed.

Getting rid of `P_sortie_TypeLastIARD_HP_HH_AUT`, `P_sortie_TypeLastIARD_HP_HH_DCO` and `P_cdtypsorIARD_HP_HH_E` gives the VIFs values presented in Table 2.8. The successive removal of variables gives values that are indeed between 1 and 5, which indicates that there is no more risk of multicollinearity between them. Basically, what was done here is getting rid of two out of the 4 initial categorical variables: the two "Household view" variables that correspond to the two other variables.

Dummy variable	R squared	VIF
P_sortie_TypeLastIARD_HP_AUT	0.499600	1.998403
P_sortie_TypeLastIARD_HP_DCO	0.339932	1.514994
P_sortie_TypeLastIARD_HP_INC	0.472535	1.895860
P_cdtypsorIARD_HP_E	0.024409	1.025020

Table 2.8: VIF values of the final subset of features.

The previous manipulation can be performed repeatedly on the whole set of features in the Single Views to determine a clean set of features to use for a model. In the context of Single Views, the multicollinearity between numerical features is dealt with using a threshold in correlation value. When it comes to categorical features, the use of dummy variables is not advised, as some categorical features have a high number of values. Instead, as detailed in section 2.2.7 category transformation is performed such that we don't have to deal with a high amount of new variables.

2.2.10 Feature importance using Random Forest algorithm

In Section 2.2.4 and 2.2.5, we tried to get some insight about the different variables and how they could impact positively or negatively the presence of churned customers. These analyses were conducted without much thought, as we blindly wandered across the set of available features, looking for any potentially interesting results. It is now time to put machine learning to the service of the research in order to narrow it down to the most interesting features. In this section, we use the transparency of Random Forest models to establish a ranking of the most important features, as far as the binary classification problem of churned customers is concerned. They offer a possibility to evaluate the relevancy of each feature to a problem by looking at the accumulated decrease of impurity in each decision tree. Although more details are given about impurity measurement and decision trees in Section 3.2, we will here limit ourselves to using the Random forest archetype without questioning its architecture or the methods used for measuring impurity.

The process for determining the importance is quite simple: For each Single View, a Random Forest model is fitted on a fraction of the Single View, called training set. Once the model has completed the training, the standard deviation and the mean of the accumulation of impurity decrease within each tree is computed (SDDI and MDI). The features with the highest MDI are defined as the most "informative" for the problem. However, there is an issue that needs to be addressed before processing: Features that have a high cardinality (numerous unique values), tend to have a low MDI because of their nature. The sparsity of the feature cause a low number of instances for each unique value and the decision trees have more trouble for detecting patterns that have meaning for the classification. Additionally, with a large number of unique values, the values become more evenly distributed across different subsets of data. This lessens the ability of a single value to reduce impurity by itself. To overcome the bias of the mean impurity decrease, we adopt a more destructive approach called feature permutation.

The idea behind feature permutation is to train a model just as previously, but then, instead of observing each tree to compute the decrease in impurity, a test set (fraction of the Single View that was not used for training) is used. Individually (feature by feature), the values of the feature are shuffled and we monitor the drop in performance. Basically, the features for which we observe the largest performance drop are the most important ones. For the sake of model efficiency, since the problem is quite unbalanced, the whole process (fitting and computation of the mean accuracy decrease) is

repeated on subsets of each Single View from April to August 2022, where each subset is a balanced subset of positive and negative churned customers.

The advantage of having discussed multicollinearity in Section 2.2.9 is that when we establish a ranking, there is no redundancy since any variable that carries information that is similar to another variable would have been removed. Figure 2.17, 2.18 and 2.20 show the 30 features with the highest mean accuracy decrease (on average, from April to August 2022) for the three insurance types. With this display, one can expose which variable types are the most responsible of the model's performance. Before looking at each insurance type individually, let us detect which features are redundant between the types. The presence of `C_age_EHP` (Customer's age) really ties in with the observations made in Figure 2.8 and Table 2.4 of Section 2.2.4, as a customer's age is quite related to his/her lifestage. We have seen in Table 2.4 that depending on the customer's lifestage, the overall churn rate could range between 1.97% for young profiles to 0.58% for senior profiles. When performing feature permutation, shuffling the values of `C_age_EHP` led to drops in accuracy of more than 1.5% in worst case scenarios, which is significant for a single feature among hundreds. We also observe that the variable `P_sortie_TypeLastIARD_HP` is present in every top 30. The intuition exposed thank to Figure 2.6 and 2.7 of Section 2.2.4 was indeed correct. Some other variables slipped under the radar such as `I_MAI_nbmonthlast_HP` (number of months since the last intervention of type 3) which causes large drops in accuracy when its values are shuffled. It appears that features that belong to the `Intervention` feature subset are important, as they have a strong presence when it comes to great drops in accuracy.

In the context of Car Insurance, the feature that causes the largest accuracy drops is `CTT_AUT_anneecst_EHP` (Year of construction of the insured vehicle). Previously, it was observed that this feature's distribution changes whether we considered positive or negative churn cases (see Figure 2.10). The relation was defined as follows: Customers with older vehicles tend to churn less than customers with recent vehicles. The drops in accuracy when performing permutation prove that the relationship is grounded. Additionally, `CTT_AUT_puissance_EHP` is also present, but has less impact on the accuracy. There is another feature that is interesting to be discussed. The relationship between `C_etatcivil_EHP` and churn rate was exposed at the beginning of Section 2.2.4. Figure 2.17 proves us that there is indeed a drop in accuracy on average. However, let us note the very high standard error of the feature in comparison with the others. The process of feature permutation, in some cases for this feature in particular, led to large accuracy growths instead of drops, resulting in a widely spread population of mean accuracy changes. The resulting accuracy drop is thus not much representative of the overall changes and any relation that stems from the feature `C_etatcivil_EHP` should be taken with caution.

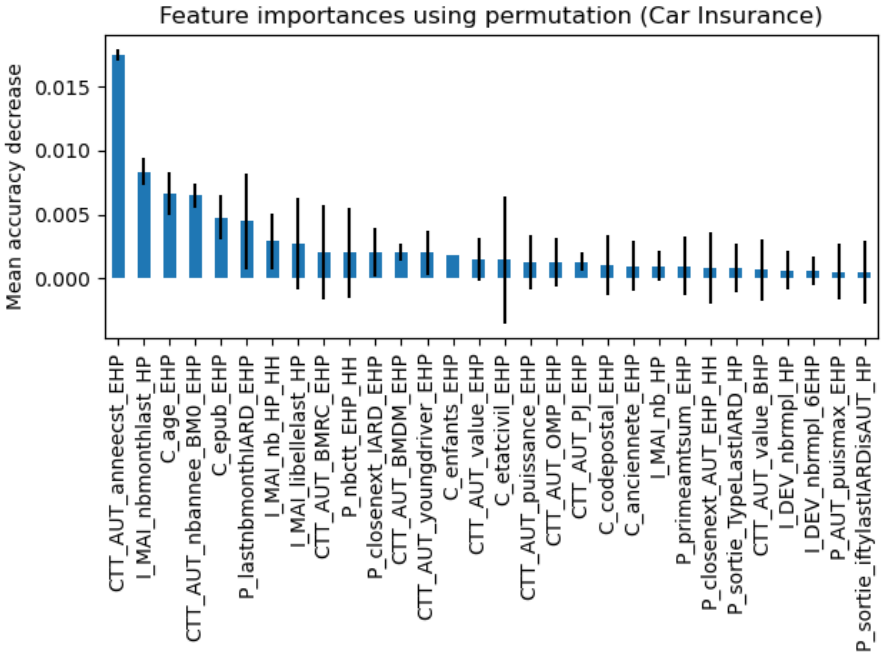


Figure 2.17: Feature importance ranked by mean accuracy decrease (on average from April to August 2022) of Car insurance.

In the context of Fire Insurance (for tenants) (Figure 2.18), the feature named `CTT_INC_NBPIECES_EHP` (number of rooms in the insured housing) has a dominating impact on the performance of the classification algorithm, with drops in accuracy of more than 4% in worst case scenarios! To understand the reason behind this tremendous drop in accuracy, one should look at the distribution of churn cases among the different values of `CTT_INC_NBPIECES_EHP`. Figure 2.19 shows first the distribution of positive and negative churn cases w.r.t the distribution of the feature and then it shows the sum of counts for positive and negative values of `CTT_INC_NBPIECES_EHP`. The available documentation about the Single Views does not mention how a negative value should be treated in this case (since the feature should by definition have only positive integer values), but it can be guessed that the value -1 expresses a value that is missing/not available. Out of the 913 positive churn cases, only 2 of them have a positive value of `CTT_INC_NBPIECES_EHP`! While we have seen that the average churn rate in the context of Fire insurance (tenant) is 1.09%, the churn rate of a tenant for which the number of rooms of his/her housing is not available exceeds 1.5%. Now it is trivial to understand why feature permutation has such an impact: Any input that has the value -1 for `CTT_INC_NBPIECES_EHP` will be permuted with a positive value and most of the time, the classification algorithm will classify it as negative churn (because of the absence of positive churn cases among the positive values of `CTT_INC_NBPIECES_EHP`). This means that the great majority of the test set will be classified as negative. Since the training and testing sets are balanced in terms of target output, an algorithm that classifies most of the unobserved data as negative is less accurate.

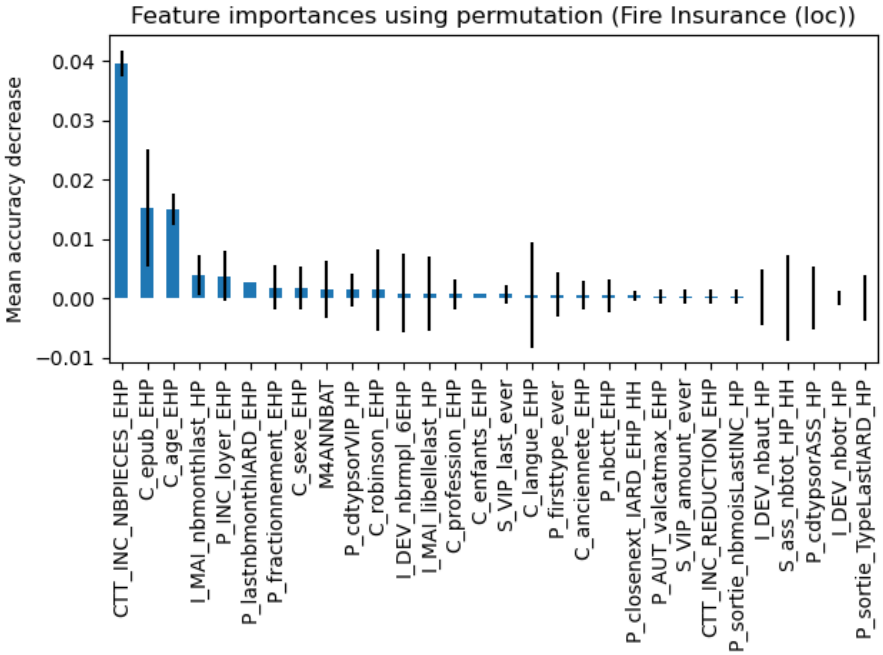
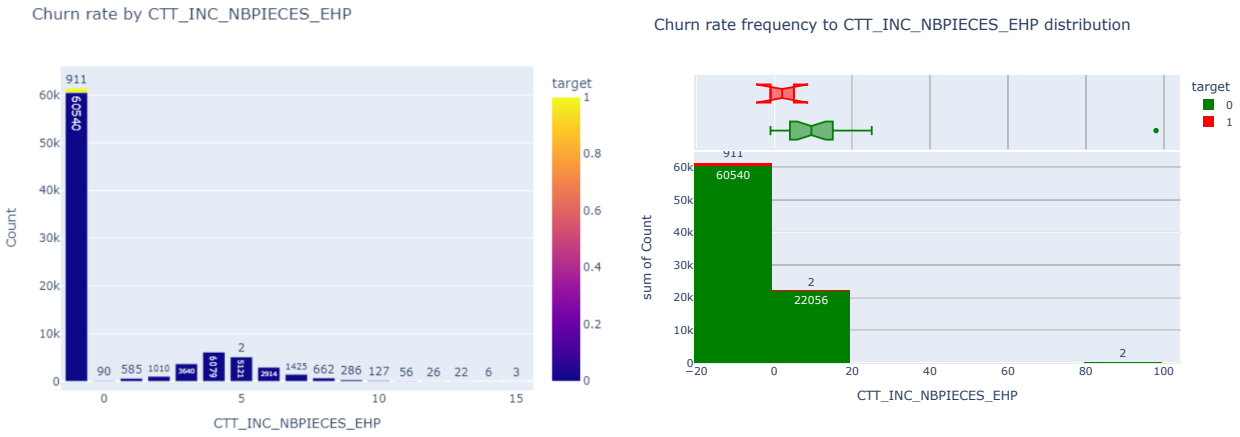


Figure 2.18: Feature importance ranked by mean accuracy decrease (on average from April to August 2022) of Fire insurance (tenant).



(a) Count of churn occurrences based on `CTT_INC_NBPIECES_EHP`

(b) Count of churn occurrences based on `CTT_INC_NBPIECES_EHP`, grouped by value sign

Figure 2.19: Comparison of the distribution of `CTT_INC_NBPIECES_EHP` and churn (Fire insurance (tenant), April 2022).

On the side of the owners ((Figure 2.20), the number of rooms seem to have less impact on the performance of the model. It can be seen that customer-related features such as age, postal code or profession can influence greatly the decision made by the model. In comparison with the other insurances, the standard errors of the most impactful features are high, which means that the establishment of a ranking would be more laborious.

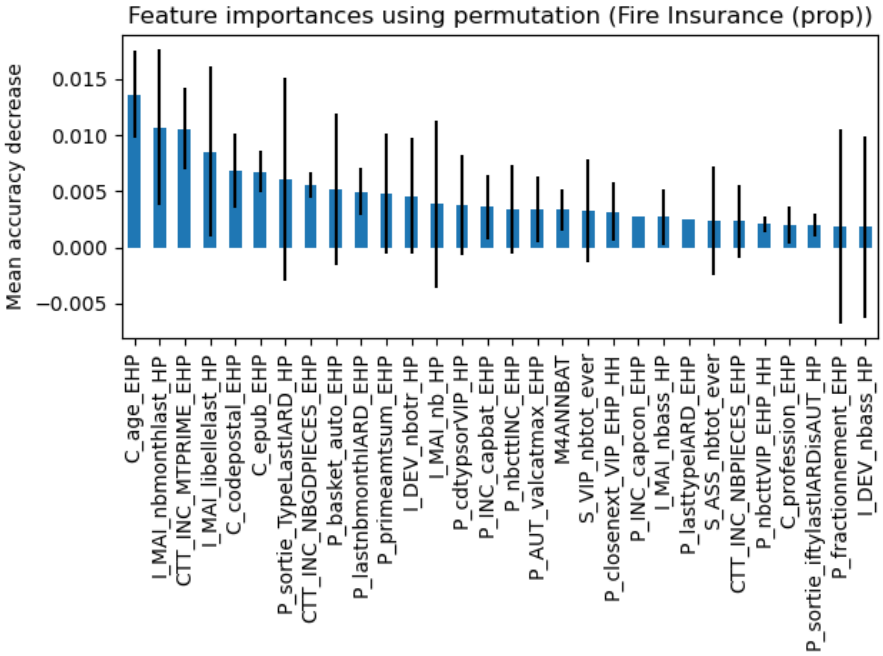


Figure 2.20: Feature importance ranked by mean accuracy decrease (on average from April to August 2022) of Fire insurance (owner).

2.2.11 Dimension reduction

Now that we have a good idea of the nature of the input space, we might want to deal with the large quantity of data. Even though the process of avoiding multicollinearity allowed to get rid of a fraction of the Single Views, we are still left with hundreds of features whose organization cannot be observed. Dimension reduction techniques are defined as methods whose goal is to reduce the number of dimensions of the input space of dataset, i.e. its number of features. They are also useful for detecting outliers and clusters of data (details covered in Section 2.2.12), as we will see that the dimension reduction techniques that are covered will serve as basis for detecting clusters and dealing with outliers. Even though we see in this Section that the methods that were applied did not lead to much interesting results, it was still considered interesting to conduct the analysis for a potential optimization in terms of memory and time usage.

Principal Component Analysis

Before going deep into the analysis of the different methods, it should be mentioned that most of the methods are supposed to be used in an unsupervised setting, meaning that we explore the data without the knowledge that customers are classified as positive or negative to churn. However, in this context, the objective of dimension reduction and clustering is to have a clear visual representation of the global structure of the data. The main focus is about being able to separate positive and negative churn cases. This is why the following methods will be processed by considering a supervised setting, i.e. we will consider the output class labels as input values. This choice is motivated by the fact that in such a dense setting, separating the cases without having prior knowledge of their distribution (and when the distribution is that much unbalanced) is too laborious. One example of this statement is shown in Figure 9 of Appendix .3.4. One of the projection was done using the labels as input, the other was done without them. The global structure is similar, but the separation is extremely different from one projection to another (more about T-SNE in Section 2.2.11. Thus, the methods loose some interest, because unobserved data cannot use the findings of these procedures. The priority is given to a projection of the data that separates well both cases while having the knowledge of their distribution and all that while having a faithful global structure of the Single Views.

Principal Component Analysis is a projection method that can represent a largely multivariate dataset

as a smaller set of features. It is an iterative method that is well-suited for observing projected data in 2D or 3D that works by finding principal components, i.e elements that account for as much variance as possible. Finding a principal component is equivalent to solving the following optimization problem: Let us assume that we have a matrix of centered observations $\mathbf{X}$ such that

$$\mathbf{X} = [x_1 - \mu, ..., x_N - \mu] \tag{2.17}$$

where μ is the mean vector (used to center the data) and $\mathbf{X}$ has the dimensions F x N where F is the number of features and N the number of observations. The covariance matrix is given by

$$\mathbf{S_t} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)(x_i - \mu)^T = \frac{1}{N}\mathbf{X}\mathbf{X}^T \tag{2.18}$$

The goal is to maximize the explained variance of each dimension by maximizing

$$\mathbf{W_0} = \underset{\mathbf{W}}{argmax}\ \ tr(\mathbf{W}^T\mathbf{S_t}\mathbf{W}) \tag{2.19}$$

$$s.t. \qquad \mathbf{W^T W} = \mathbf{I} \tag{2.20}$$

The solution of this problem can be found by performing an eigenanalysis of $\mathbf{S_t}$ such that

$$\mathbf{S_t W} = \mathbf{W\Lambda} \tag{2.21}$$

where Λ are its eigenvalues([2]). The pseudocode for the PCA procedure is displayed on Figure 2.21. Depending on the percentage of variance explained by the different components, we desire to keep d of them. To account for the majority of the information, a minimum of 70% to 80% of the total explained variance should be contained in the d components. There are cases where only a few of them suffice but we will observe that this is not the case in our context.

Algorithm 1 Principal Component Analysis

1: **procedure** PCA
2: Compute dot product matrix: $\mathbf{X}^T\mathbf{X} = \sum_{i=1}^{N}(\mathbf{x}_i - \boldsymbol{\mu})^T(\mathbf{x}_i - \boldsymbol{\mu})$
3: Eigenanalysis: $\mathbf{X}^T\mathbf{X} = \mathbf{V\Lambda V}^T$
4: Compute eigenvectors: $\mathbf{U} = \mathbf{XV\Lambda}^{-\frac{1}{2}}$
5: Keep specific number of first components: $\mathbf{U_d} = [\mathbf{u}_1, \ldots, \mathbf{u}_d]$
6: Compute d features: $\mathbf{Y} = \mathbf{U_d}^T\mathbf{X}$

Figure 2.21: Pseudocode of the PCA procedure

In the following, we apply PCA to the Car Insurance Single View. For PCA to be applied, the data has been normalized using standard scaling (after each categorical variable has been discretized). Ideally, we would want the number of principal components d to be at most equal to 3 to be able to perform a visual exploration. The first PCA is performed on the whole Single View. Figure 2.22 represents the quantity of explained variance for each of the 15 first principal components. It can be seen that we are quite far from the desired 70% threshold. This simple observation makes it clear that a 2D or 3D projection is not conceivable. Still, we want to be able to observe if there exist differences between groups of individuals labeled as negative to churn and groups of individuals that are detected as churned customers. For this purpose, a subset is built with all of the observations that are positive churn cases and a sample of negative churn observations. A second PCA is performed on this subset with the goal to find differences in projection between positive and negative cases. In this particular setup, Figure 2.24 shows that the first principal component explains 77% of the total variance, which

[2]More details about the implementation of PCA here

allows to observe some differences in projection. We can see on both projections that positive churn cases tend to have smaller values of PC1. The explained variance is specific to the selected subset and thus it is not representative of the whole Single View at all. But the idea of sub-sampling datasets for highlighting differences between specific features or outputs is an idea that has already been explored in multiple fields of work and is known for bringing interesting results. Still, in the context of Single Views, PCA in itself is not much efficient and linear dimension reduction seems to be a task that does not need to be deepened.

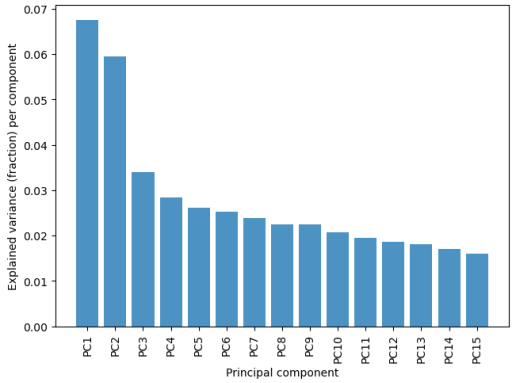
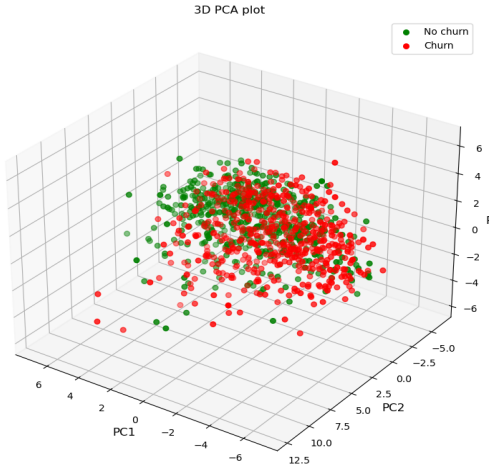


Figure 2.22: Explained variance of the 15 first principal components of a complete Single View (Car Insurance, April 2022).



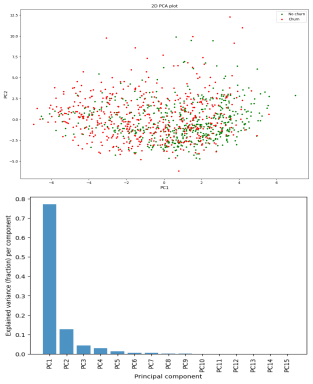


Figure 2.23: 2D projection and explained variance.

Figure 2.24: 2D and 3D PCA on a balanced sample of a Single View (Car Insurance, April 2022). The explained variance is sample-specific.

T-SNE

T-distributed Stochastic Neighbor Embedding (or T-SNE) is another great method to visualize high-dimensional data. It belongs to the Manifold learning algorithms, which is a family of techniques to approach non-linear dimensionality reduction. Principal Component Analysis and other techniques such as LDA or IDA are powerful enough to capture linear structural information, but fail to capture non-linear information. Manifold learning techniques attempt to generalize linear frameworks to be sensitive to non-linear structures.

T-SNE, specifically, is a method that converts the affinity between data points to probabilities. It uses the principle of similarity between points to produce a probability distribution and then embed the data in a lower-dimension space. Mathematically, let x_i and x_j be two different points of a dataset. For each data point j we model the probability of x_i belonging to the same class (belonging to the

same class means being a neighbor in this sense) with a Gaussian distribution:

$$p_{i|j} := \frac{exp(|x_i - x_j|^2/2\sigma_j^2)}{\sum_{k \neq j} exp(-|x_k - x_j|^2/2\sigma_j^2)} \tag{2.22}$$

where σ_j is the variance. Instead of directly setting it as a parameter, the method suggests setting the expected number of neighbors, also known as perplexity. We will see that the value of this parameter is quite important. Intuitively, the perplexity represents the balance between conserving the local structure and the global structure. A higher perplexity prioritizes the global structure, which we will be looking for, as our Single Views are very dense and heavy. We form the distribution $\mathbf{P}$ that represents the data by symmetrizing the probabilities:

$$P = (p_{ij})_{i,j=1}^n, \quad p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \tag{2.23}$$

We also define the distribution $\mathbf{Q}$ that will be the distribution in the lower-dimensional space:

$$Q = (q_{ij})_{i,j=1}^n, \quad q_{ij} = \frac{(1 + |y_i - y_j|^2)^{-1}}{\sum_{k \neq j}(1 + |y_k - y_l|^2)^{-1}}. \tag{2.24}$$

In order to find the y_i, the goal is to have $\mathbf{P}$ and $\mathbf{Q}$ being as close as possible, which is computed using the Kullback-Leibler divergence:

$$KL(P|Q) = \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} log \frac{p_{ij}}{q_{ij}}. \tag{2.25}$$

The gradient of the KL divergence w.r.t y_i can thus be found using gradient descent and the optimization problem is solved. Figure 2.25 is a simplified summary of the whole algorithm. What is left to discuss are the parameters. The main parameter that needs to be discussed is the perplexity. Unfortunately, there is no clear mathematical definition of the perplexity and determining a good value for it requires some experimentation. Appendix .3.4 contains a battery of tests with multiple perplexity values on 2 and 3 T-SNE components. Out of these tests, some gave interesting structural forms which can be observed on Figure 2.26. As the different tests can attest, the value of the expected number of neighbors has a great influence on the projection in the embedded space. It can be seen that greater values allow to better distinguish the negative and positive churn cases. Starting from $p = 100$, one can already observe a clear separation between the two output classes. However, to stay faithful to the global structure of the Single View, the perplexity is chosen to be greater than 100. A perplexity of 750 is great, because not only is the global structure of the data conserved (very dense data), the choice of this value marks a clear separation between negative and positive cases, with only a few positive cases belonging to the very dense mass of data in the center of the embedded space.

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity *Perp* (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=1 **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

Figure 2.25: Pseudocode of the T-SNE algorithm

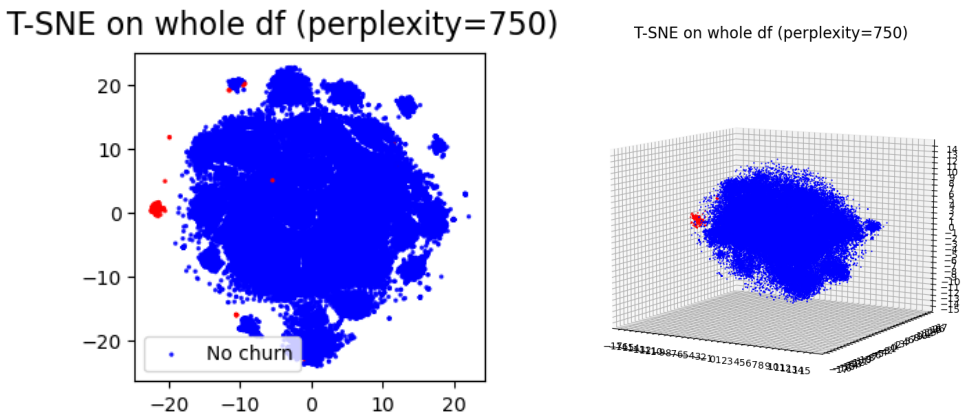


Figure 2.26: Application of the T-SNE embedding with a perplexity of 750 (Car Insurance, April 2022).

The results observed in Figure 2.26 will be crucial for the next section about clustering and anomaly detection, because of two positive elements: The data is very dense and it should be represented as such when embedded or projected in a lower-dimension space. Secondly, even though the discussed methods are supposed to be used in an unsupervised setting, the fact that we dispose of the true output labels as inputs allow us to observe if clustering techniques could reinforce this separation of cases. Moreover, we could even compare positive churn cases with anomalies and assess the nature of these cases based on the anomaly detection algorithm (outlier or inlier) and the choice of values for the tuning parameters.

2.2.12 Clustering and Outliers

The previous analyses are the perfect transition to tackle clustering and anomaly detection techniques. In fact, Component Analysis and Manifold learning can serve as basis for more complex clustering techniques, as they already attempt to use some characteristics of the data to create groups of points. We have seen that T-SNE is indeed a nice way of separating the negative and positive churn cases, hence we will use it when we have to observe whether or not clusters are well formed and outliers are found.

As explained in the beginning of Section 2.2.8, the matter of the presence of outliers in the data is dealt with once the Single Views have already undergone variable imputation. What is meant is that the potential outliers found within the Single Views are not issued only from natural information, but can originate from the imputation methods. Even though it is very unlikely for an imputation method to create outliers (by its nature, imputation uses the trends in the data to synthesize data that follows these trends), it is important to state that from this point, the Single Views contain synthetic data.

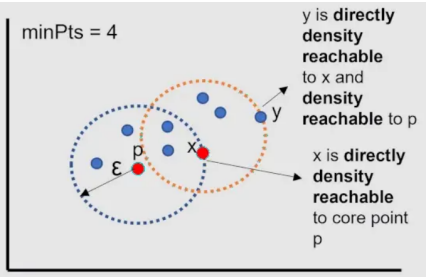
The detection of observations that are noticeably different from the rest of the data is a task whose solutions are constantly evolving. During the analysis, three state-of-the-art methods were used on the Single Views: The first method consists in building density-based spatial clusters and detect the anomalies within and out of the clusters (DBSCAN). The second method is also density-based and depends on a score that is based on the local density deviation of an observation w.r.t its neighbors (Local Outlier Factor LOF). The third method is based on the Random Forest algorithm and consists in creating Isolation Trees that isolates observations by randomly selecting features and partition the data based on a random split (Isolation Forest). Each of these methods is tested on the same Insurance type to compare the performances.

DBSCAN

Density Based Spatial Clustering of Applications with Noise, or DBSCAN, is an unsupervised algorithm that uses the k-nearest neighbors paradigm to create arbitrary-shaped clusters. There also exists a clustering method called k-means clustering, but DBSCAN has multiple advantages that this method does not have. First, the clusters of DBSCAN are more flexible in terms of shape, where k-means clustering is limited to spherical shaped clusters. Secondly, unlike k-means, DBSCAN does not require to specify the number of clusters initially. However, it requires two parameters: the radius of neighborhoods for a given data point (called ϵ) and the minimum number of data points in a given neighborhood (called minPts). The DBSCAN algorithm can be summarized as follows:

1) Take any point p randomly. This point is called a core point if it has a minimum of $minPts$ points within its ϵ-neighborhood

2) Identify all density reachable points from p with ϵ and $minPts$ parameter. A point y is density reachable from a point p if and only if a point y is directly density reachable to core point x, which is also density reachable to core point p.



3) If p is a core point, create a cluster (with ϵ and $minPts$).

4) If p is a border point, visit a next point in the dataset.

5) Continue the algorithm until all points are visited.

What is left to do now is to set $minPts$ and ϵ. $minPts$ can be either set as double the number of features or be fine-tuned depending on the desired number of clusters. However, it is not as simple for the radius ϵ. Since it is also a question of distance, one of the solutions to find the optimal ϵ is to compute the average distance of every data point to its k-nearest neighbors, known as kNN distances, and find the elbow point that indicates a sharp change in the value of the distance for the sorted k^{th} column. Let us apply this method to a sample Single View. Figure 2.27 displays the kNN distance

plot of the 6^{th} nearest neighbor for each observation in the Single View. The value 6 corresponds to the number of nearest neighbors used during the computation of the distances. The elbow is located at $\epsilon = 0.93$.

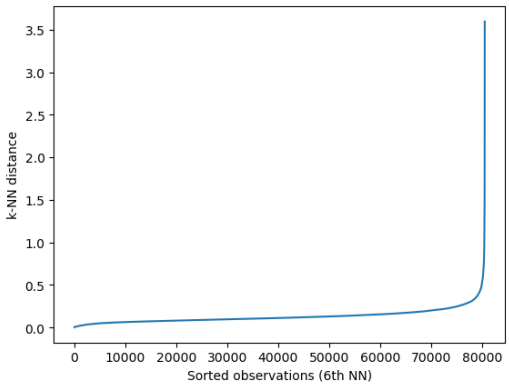


Figure 2.27: KNN Distance plot of the 6^{th} nearest neighbor (T-SNE embedded Car Insurance, April 2022)

Now that we have set our parameters values, let us proceed with the DBSCAN algorithm. It is mandatory to note that, since we want to assess the quality of the method, the data that is used for the algorithm is not the raw Single View, but the T-SNE embedding of the Single View with the best parameters that were discussed in Section 2.2.11, such that we can visualize the efficiency of the clustering algorithm on Figure 2.28 ([3]). While the visual exploration of the previous section was centered around verifying if the embedding of the data had differences for negative and positive churn cases, the visual analysis that is done here consists in checking how the different clusters are formed. Merging both analysis can thus tell us if DBSCAN is a good solution for separating positive and negative churn cases when the data is embedded using supervised T-SNE with accurate parameters.

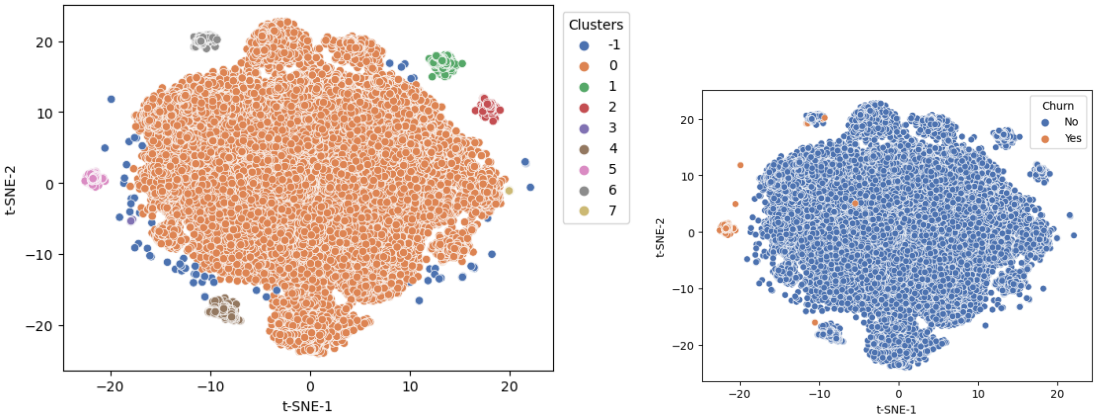


Figure 2.28: Left: DBSCAN clustering ($minPts = 12$ and $\epsilon = 0.93$) of the T-SNE embedded Car Insurance Single View. Right: Color map of the output labels for the same embedding.

The DBSCAN algorithm creates arbitrarily shaped clusters that are numbered from 0 to n where n is the number of clusters. The number given to a cluster is not representative of its size and the clusters are not sorted. The cluster associated to the value -1 corresponds to the set of values that are not included in any cluster. These values are considered as outliers by the DBSCAN algorithm. We will see later with more anomaly detection techniques that the outliers found using DBSCAN are quite accurate (see Section 2.2.12) in this context. Figure 2.28 displays the different clusters of the embedding discussed in the previous section. There is a total of 8 clusters and since the data is very dense, the size of each cluster is provided:

$$ClusterSizes = \{0 : 76436, 4 : 962, 1 : 858, 2 : 734, 6 : 705, 5 : 639, -1 : 104, 7 : 80, 3 : 35\}.$$

[3]Clarification: DBSCAN is an unsupervised algorithm, but it will use a supervised T-SNE projection. This means that the T-SNE components include the information about class labels (and thus is considered supervised)

With such a global structure, it was expected that one cluster would totally dominate the rest of them. However, this is where our T-SNE embedding comes in handy: The dominating cluster doesn't contain much positive cases. The majority of the positive cases are contained in one of the smaller clusters (Cluster 5, pink in Figure 2.28), which shows that DBSCAN did well when separating positive and negative churn cases. Out of curiosity, it would be relevant to see which proportion of positive churn cases is contained in the 5th cluster. Table 2.9 gives the count of positive and negative churn cases for each cluster. Out of the 673 positive cases, 639 (95%) of them are in the 5th cluster. Moreover, this cluster contains only positive churn cases. The combination of supervised T-SNE embedding and DBSCAN with well-tuned hyperparameters leads to a almost perfect separation of the two classes. Besides that, we observe other small clusters that are formed around the cluster 0. These small clusters mainly contain negative cases, but not more than 1000 samples each. What is left to discuss are the isolated values around the main cluster, which are considered as outliers. However, HDBSCAN is not the best tool to highlight outliers, which is why we will now resort to more adequate methods in terms of anomaly detection.

Cluster	-1	0	1	2	3	4	5	6	7	Total
Size	104	76436	858	734	35	962	639	705	80	80553
Positive cases	10	13	0	0	0	0	**639**	9	2	673
Negative cases	94	76436	858	734	35	962	**0**	696	78	79880
P/N	0.1064	0.0002	0	0	0	0	**+inf**	0.0129	0.0256	0.0084

Table 2.9: Count and proportion of churn cases per cluster.

Local Outlier Factor and Isolation Forests

In this section, we talk about two anomaly detection techniques that were used. The first technique is called Local Outlier Factor. It consists in giving each observation a score that is based on the difference in density between this point and its n closest neighbors. The density of a point is proportional to its quantity of close neighbors (within a certain radius). The number n of neighbors is a hyperparameter that should be at least greater than the desired minimum number of samples in a cluster and not too high, to account for local and non-local outliers. We have seen that a value of $minPts = 12$ was great to have good clusters, so we should have $n \geq 12$. Figure 2.29, as well as Figure 10 in the Appendix .3.5 are displays of the T-SNE embedding with each point being circled (with different values of n). The size of the circle represents the score, with a high score representing a high probability of being an outlier. In some of the smaller clusters, we observe a few local outliers. When comparing this Figure with results found with the DBSCAN algorithm, we observe that the outliers with the highest scores are located in similar zones of the 2D space. They are mainly located around the central dense area and the smaller areas on the edge don't seem to have a particularly high amount of local outliers. This reinforces the idea that all the observations in the small clusters around the main one have very similar characteristics that is specific to their corresponding cluster, just as the cluster that contains only positive churn cases.

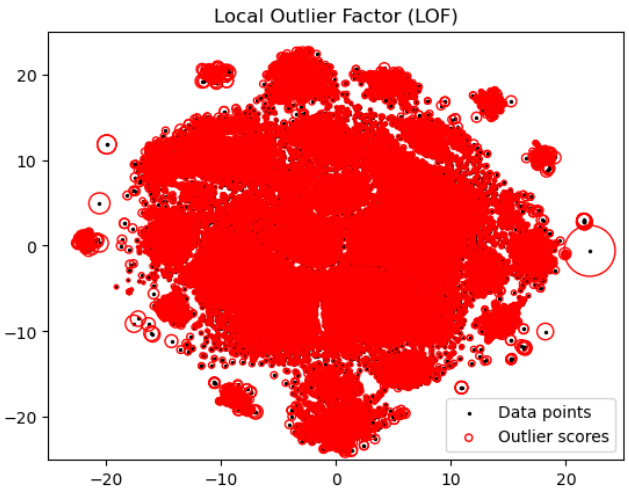


Figure 2.29: Local Outlier Factor scores with $n\_neighbors = 50$ (T-SNE embedding, Car Insurance April 2022)

The second anomaly detection method relies on boundary decision making. An Isolation Forest is an ensemble of Isolation Trees that isolate observations by randomly partitioning in a recursive fashion. Each tree uses n samples for training, where n is a hyperparameter. The algorithm also uses contamination c, which refers to the proportion of the data that is expected to be outlying. A contamination of 0.2 signifies that 20% of the dataset contains outliers. Once a tree has chosen its training set, it randomly chooses one of the features. It then randomly chooses a split value between the minimum and the maximum values and starts partitioning. An outlier can be separated from the rest of the samples in fewer steps than inliers. Isolation Trees thus find outliers by finding the observations that are the fastest at being separated from the others.

Just as for Local Outlier Factor, multiple tests were performed with different values of the hyperparameters n (number of samples) and contamination c. All the tests are available in the Appendix 11. The yellow zone indicates the area where points are considered as inliers and the grey zone is for outliers. A high contamination tends to segregate a lot data points that were not detected as outliers by DBSCAN and LOF. As Figure 2.30 shows, if the expected proportion of outliers is 10%, the small clusters are partly considered as outliers and the cluster that contains the majority of positive churn cases is fully outlying. To have these small clusters being part of the inliers, one should use a very low contamination value with a sufficient number of observations for training.

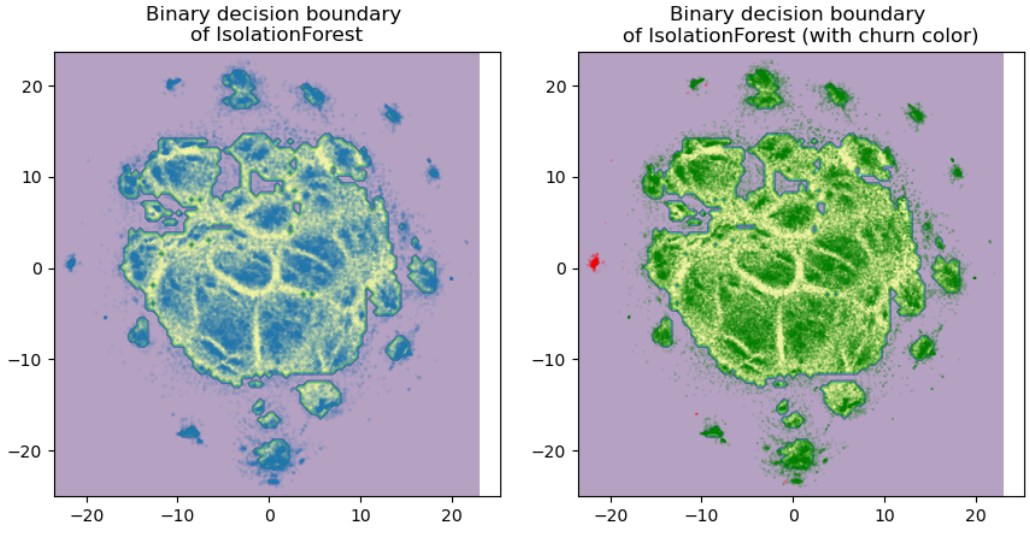


Figure 2.30: Left: Isolation Forest decision boundaries with $n = 80000, c = 0.1$. Right: Same decision boundaries with color mapping of the output labels.

2.3 Data Preprocessing Protocol

This section is a summary of the set of methods applied to prepare the data for the purpose of modelling, in the sense of predicting the target output. As a lot of data analysis tools were tested, a clarification of what is really used for preprocessing is necessary.

- Variables filtering : Obsolete variables were removed, as well as data that were missing or had the same value at more than 99% of the time.→2.2.6
- Categorization : Each non-numerical variable is mapped into a numerical variable.→2.2.7.
- Normalization : The data was normalized using Standard scaling.→2.2.7.
- Iterative imputation : Every column that contains missing data is filled using iterative imputation.→2.2.8.
- Multicollinearity : If the correlation of a pair of variables exceeds 0.7 in absolute value, get rid of one of them.→2.2.9.

On an indicative basis, Table 2.10 shows how the different steps change the size of the input space.

Single View	Initial	After Filtering	After multicollinearity (final)
Car	345	229	104
Fire insurance (loc)	306	195	74
Fire insurance (prop)	306	199	82

Table 2.10: Evolution of the number of features during data transformation

As the last part of the Chapter was mainly focused on the observation of the structure of the data, the methods applied will not be used for preprocessing the data, as they are used in a supervised setting and they imply a non-negligible loss of information. The main purpose of Section 2.2.11 and 2.2.12 is observation and in this context, the discussed techniques are not intended for output modelling.

Chapter 3

Churn prediction

While the Chapter about Data included multiple different settings depending on the methods with the objective to explore the data and find relations, the Chapters about Churn prediction and Continuous learning both have the same objective: Predicting whether or not customers will churn in the nearest future. It involves building machine learning models using supervised learning. The following sections will exclusively discuss supervised learning techniques.

3.1 Supervised learning

Although some supervised learning techniques were already tackled in the previous chapter, this section provides a clear definition of its nature and objectives. The chapter about modelling is exclusively about supervised techniques, hence it is mandatory to formally describe some fundamentals.

Supervised learning refers to the task of finding a function that uses the inputs of a learning set to approximate at best the output. This task has two main objectives, which are predictive and informative. New objects, also called unobserved data in opposition with observed data, are given an output prediction based on their attributes (inputs). Supervised learning is also informative, because it helps learn the relations between an output and the inputs. The informative goal of supervised learning has already been achieved in the first chapter of the present work. The second chapter is focused on the predictive objective of such methods.

Formally, a learning algorithm learns by finding a function f that minimizes some loss function l over a learning sample. From a learning sample $LS = \{(x_i, y_i)|i = 1, ..., N\}$ with $x_i \in X$ and $y_i \in Y$ (X and Y are the distributions of x_i and y_i), find a function $f : X \to Y$ that minimizes the expectation of some loss function $l : Y \times Y \to R$ over the joint distribution of input/output pairs:

$$E_{x,y}\{l(f(x), y)\}. \tag{3.1}$$

For supervised classification, the outputs are symbolic. In this context, there are two output categories, called Negative and Positive. An observation called Negative corresponds to a customer who did not churn while a Positive observation refers to a customer who churned. Negative and positive outputs are mapped to integers in the form $Negative \to 0$ and $Positive \to 1$ just as the `False` and `True` labels in binary applications.

3.1.1 Model selection

Later in this chapter, we will talk about the different models that are known to be performing well for such classification tasks. However, one should know how to be able to evaluate which model would be the the best in each situation. The best learning algorithm can be chosen based on three criteria: accuracy, measured by the generalization error, efficiency (computing power and scalability) and interpretability. For now, let us focus on the first criterion. To understand how the generalization error is defined, one should be aware of the bias/variance decomposition.In a regression problem, let

LS be a learning sample drawn from a distribution. A good learning algorithm should perform well generally for every learning sample. What is sought to be minimized is thus

$$E = E_{LS}\{E_y\{(y - \hat{y})^2\}\}, \tag{3.2}$$

where $\hat{y}$ is the output predicted by the model and y is the true output. This error can be decomposed into two elements with the following:

$$\begin{aligned}
&E_{LS}\{E_y\{(y - \hat{y})^2\}\} \\
&= E_{LS}\{E_y\{y - E_y\{y\} + E_y\{y\} - \hat{y})^2\}\} \\
&= E_{LS}\{E_y\{y - E_y\{y\})^2\}\} + E_{LS}\{E_y\{(E_y\{y\} - \hat{y})^2\}\} \\
&+ E_{LS}\{E_y\{2(y - E_y\{y\})(E_y\{y\} - \hat{y})\}\} \\
&= E_y\{(y - E_y\{y\})^2\} + E_{LS}\{(E_y\{y\} - \hat{y})^2\} \\
&+ E_{LS}\{2(E_y\{y\} - E_y\{y\})((E_y\{y\} - \hat{y})\} \\
&= E_y\{(y - E_y\{y\})^2\} + E_{LS}\{(E_y\{y\} - \hat{y})^2\}
\end{aligned}$$

The first term is called the residual error. It is the minimal attainable error. The second term can also be decomposed into two terms:

$$\begin{aligned}
&E_{LS}\{(E_y\{y\} - \hat{y})^2\} \\
&= ... \\
&= (E_y\{y\} - E_{LS}\{\hat{y}\})^2 + E_{LS}\{(\hat{y} - E_{LS}\{\hat{y}\})^2\}
\end{aligned}$$

where the first term is the squared difference between the Bayesian model (model based on Bayesian probability theory and is the best possible model))and the average model (a learning algorithm trained on a learning sample). This term is called the squared bias. The last term is the expected squared difference between the predicted output and the expected predicted output from the learning sample and is called the estimation variance. This term is a consequence of overfitting the learning sample. The expected generalization error can thus be rewritten as:

$$E = var_y\{y\} + bias^2 + var_{LS}\{\hat{y}\}. \tag{3.3}$$

If we apply the generalization error to classification problems, the mean misclassification error corresponds to:

$$E = E_{LS}\{E_{x,y}\{1(y \neq \hat{y}(x))\}\} \tag{3.4}$$

where x corresponds to the set of inputs. The Bayes model is obtained by

$$h_B(x) = \underset{c}{argmax} P(y = c|x) \tag{3.5}$$

and the average model is obtained by finding

$$\underset{c}{argmax} P(\hat{y}(x = c|x)). \tag{3.6}$$

The mean misclassification error cannot be decomposed into a bias and variance term, but the same phenomena can be observed.

The mean misclassification error can be computed by splitting the learning sample into two distinct sets: A training set (observed data) that would be used to train the model and a test set (unobserved data) that would be used to perform predictions and monitor the model's performance. Usually, the test set is smaller than the training set. Most of the time, it is more efficient to make this separation multiple times in a process called cross-validation. The k-fold cross-validation, for instance, divides the learning sample into k subsets. The learning algorithm is trained with every subset except one that will serve as test set. The learning process is repeated until all subsets served as test set. The

error is computed as the mean error over the predictions made for each subset. Cross-validation is a tool that will be used multiple times during evaluations as it one of the best ways to exploit the data contained in a learning sample. In the context of this work, the best models will be chosen as the ones that minimize the cross-validation mean misclassification error. This is equivalent to maximizing the classification accuracy, which can take different forms. We will see in Section 3.4 that depending on the context, the performance measurement tools should be carefully chosen as the distribution of the observed outputs has a great influence on the choice of relevant metrics.

3.2 Decision trees

When it comes to classification algorithms, decision trees appear to be one of the most state-of-the-art methods to use. Its simplicity and easily understandable features allow to create multiple complex variations using trees with different structures. In the context of this work, it is mandatory to introduce the concept as early as possible, as it is both a strong tool for performing predictions (modelling) and for data analysis (feature importance and more).

A decision tree is defined as such: Each interior node tests an attribute, each branch corresponds to an attribute value and each leaf is labelled with a class. These attributes can be of any type and tree splitting will be made according to the values taken by the observations in the learning sample. Choosing accurate attributes to split the tree the most efficiently requires to define a heuristic which is based on the fact that a tree should remain as small (a small depth and a few splits per level) as possible. To achieve this, most algorithms choose a top-down approach which consists in selecting the best splitting attribute at each level of the tree. The best splitting attribute is the attribute that will maximize class separation at each level, i.e. reduce impurity in the successors as much as possible. Such a strategy will favor short paths in trees and thus will produce smaller trees. It is however important to note that impurity measurement is not the only way to define a splitting criterion, but it is still the most used method.

3.2.1 Gini index

Maximizing class separation at each iteration of tree building involves making the successors of a node as pure as possible. Intuitively, for a binary classification problem, the objective would be to have successors with the highest percentage of one of the two classes. For instance, a successor with 80% observations labelled with the negative class is more pure than a successor with 60% observations belonging to the positive class.

There are several tools to measure impurity in the successors of a decision tree. One of them is called the *Gini index* and is defined as such. For an instance of a learning sample LS , let us call p_j the proportion of observations belonging to the output class j. It is important to note that we are at a certain step of the tree building and, in this context, LS corresponds to the fraction of the initial learning sample that is assigned to the current node. The *Gini* impurity I_{Gi} is such that

$$I_{Gi}(LS) = \sum_{j=1}^{J} p_j(1 - p_j). \tag{3.7}$$

At each step, the goal is to reduce as much as possible this impurity, i.e maximize the expected reduction of impurity, which is defined as

$$\Delta I(LS, A) = I(LS) - \sum_{a \in A(LS)} \frac{|LS_a|}{|LS|} I(LS_a), \tag{3.8}$$

where LS_a is the subset of objects o from LS such that an object o takes as value for the attribute A $A(o) = a$. A(LS) is the set of values of A observed in LS. Equation 3.8 states that we are trying to find the attribute A that maximizes the difference between the impurity of LS at the current node and the

sum of all impurities obtained by splitting all objects of the current node into n successors (multiplied by a factor defined by the ratio between the quantity of objects in the successor and the total number of objects in LS). The value of n depends on how the node will be split (which intrinsically depends on the nature of the attribute and how its values are distributed among all objects of LS)
Once the best attribute A is found, the algorithm of tree building can split objects within the successors according to their value w.r.t A and go through next iterations.

3.2.2 Overfitting

One of the reason why trees should remain as small as possible is the risk of overfitting. A decision tree T overfits a learning sample if (simultaneously) there exists a tree T' such that the predictive error of T on LS is smaller than the predictive error of T' on the same LS while the error of T on unseen data (data that was not used for building the tree) is bigger than the error of T' on the same unseen data. The over/under-fitting situation if summarized on Figure 3.1. Since complexity mainly translates by tree depth in decision trees, the deeper a tree goes, the higher the chances are of overfitting.

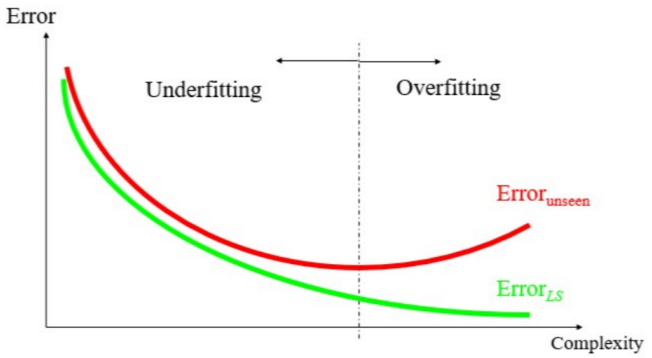


Figure 3.1: Evolution of model error w.r.t model complexity (*source*)

The previous statement can be intuitively explained. In the context of large and dense datasets such as those studied in this work, there are groups that contain a high fraction of the learning sample and because of that, groups containing only a few objects will influence the decision trees into making bad decisions (because of the lack of representation). More importantly, if some groups are too dense, any object that doesn't belong to these groups can be considered as outliers (noisy object) and the more a tree is fitted to noisy data, the more it is prone to overfitting.

There are techniques to avoid overfitting and they involve manipulating the tree building algorithm with either a pre-emptive (pre-pruning) or a corrective (post-pruning) approach. Pre-pruning consists in interrupting the tree building before it reaches perfect classification (perfect classification is reached when each leaf's impurity is equal to 0, i.e. at maximal tree depth). It is done by defining parameters that are specific to the problem such as a local sample size N_{min} or local sample impurity I_{th} and use them as benchmark for stopping the tree splitting. On the other side, the idea of post-pruning is to split the learning sample into two samples : one for building the tree and one for testing the built tree. The first sample allows to build multiple trees each with different sizes (one is complete and the others are copies of the complete trees from which we removed test nodes) and the second sample allows to calculate the generalization error of each tree to check whether or not they are overfitting. The tree that minimizes the error on the second sample is then chosen.

Performing pruning on a decision tree is a good step towards an optimal decision tree building. However, decision trees are very unstable due to their high variance and they are not always competitive with other algorithms in terms of accuracy. Still, they offer the possibility to observe and monitor how the attribute values of the different inputs can be efficiently used and classified. They are a great tool in terms of data analysis thank to their transparency and the absence of a black box effect. It would be a waste not to make use of such a convenient tool. This is why methods emerged to combine multiple decision trees together or more generally some model architectures with other model architectures.

Such strategies are called Ensemble methods and are state-of-the-art in most applications of today's society.

3.3 Ensemble methods

In the context of a given method, a learning algorithm is adapted to find the best trade-off between bias and variance. There is no possible algorithm that minimize both of them at the same time, since they are complements of each other. There is a need to find the best method with the best learning parameters that will adjust this trade-off to find the right balance. The purpose of Ensemble methods is to combine the predictions of several built models to improve predictive power with respect to a single model. Each small estimator can also be called a "weak learner" as the predictions made by every weak learner will be combined. There are two main families of ensemble: One of them consists in building models independently and have as final prediction an aggregation of the predictions of all models (averaging techniques). The other consists in building models sequentially on modified versions of the data (Boosting techniques). A model M_2 uses as input a modified version of the learning sample combined with the output of a model M_1 that has been built prior to M_2 and based on another modified version of the learning sample. Both families of methods implement some sort of voting system to take into account the predictions of all models belonging to the ensemble. Most known averaging techniques include Bagging, Random Forests and famous boosting algorithms include Gradient Boosting or AdaBoost.

3.3.1 Voting mechanisms

Grouping models together requires that the output of each model should be taken into account. This require the use of a voting system that needs to be adapted to the application. This voting system can either be totally fair or not. If the trained models of an ensemble are not supposed to perform equally relative to each other, voting can be weighted such as to give more importance to models that are supposed to perform better. For instance, boosting algorithms allow to build models sequentially by using results of the previously trained models. In most cases, the subsequent models should be weighted more importantly.

Hard vs. soft voting

Depending on the nature of the output, the mathematics behind the voting mechanism is different. In the context of classification, Hard voting consists in voting based on an output that is a label, i.e an output whose nature is one of the classes that need to be predicted. The class label with the most votes among models is the final prediction. For churn prediction, the outputs would either take the value 0 (not churned) or 1 (churned). Hard voting is used when the learning algorithm is extreme and decisive. The K-nearest neighbors method and Support Vector Machines are some examples of such algorithms. The rigidity of hard voting can sometimes limit the predictive power of an ensemble. The solution to this rigidity is to add flexibility to the voting by taking into account outputs that are class membership probabilities (when available). Every classifier provides a probability value of membership for each class. These probabilities are weighted and summed up. The target label with the highest sum of (weighted) probabilities is the predicted class label.

Mathematically, Let o be an object of a testing sample TS (unseen data) and $M_i(p_j(o))$ the probability that o belongs to class j ($j \in 0, 1, ..., J$) given by the model M_i of an ensemble of models $E = \{M_0, M_1, ..., M_I\}$. Assuming that each model is weighted equally, the final prediction $\hat{y}(o)$ is the target label that maximizes the sum of probabilities such that

$$\hat{y}(o) = \arg\max_j \sum_{i=0,1,...}^{I} \frac{1}{I} M_i(p_j(o)), \tag{3.9}$$

where I is the number of models (equal weighting). It is important to notice that the predicted class is indeed the highest sum of probabilities if and only if the probability thresholds of class membership

are all equal ($\frac{1}{J}$ for a classification problem with J target classes). For instance, if the soft voting mechanism of an ensemble gives as final prediction a vector $p(o) = (p(j = 0), p(j = 1)) = (0.30, 0.69)$ but that the probability threshold for class $j = 1$ is 0.75, the predicted class is 0 and not 1, even though the weighted sum of probabilities is lower for $j = 0$. Attention is drawn here because dealing with highly unbalanced datasets such as those of this project require to tune the probability threshold to something that is not equal among the negative and positive classes.

3.3.2 Random Forests

Introduced by *L.Breiman* and *A.Cutler*, Random Forests is an algorithm that is designed around multiple decision trees. Its principles combine bagging and random attribute subset selection. Figure 3.2 gives a schematic view of the Random Forest algorithm. It is inspired by decision trees in a sense, but instead of selecting the best splitting attribute among the whole set of attributes of the input data, it selects the best splitting attribute among a random subset of attributes. For each subset of attributes, a tree is built by following the tree building algorithm introduced in Section 3.2. Once each tree is built, predictions are performed by each tree on unseen data. For one particular observation, the final prediction takes into account each prediction for this object by using a voting mechanism.

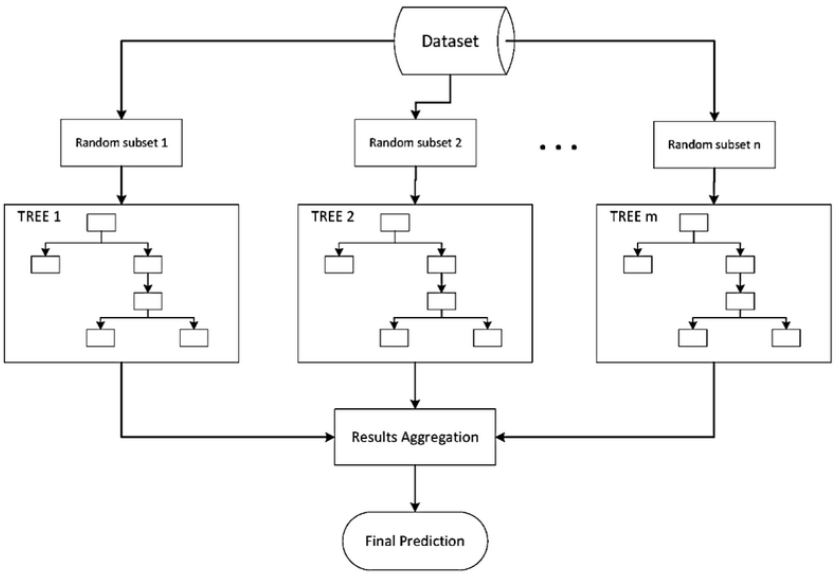


Figure 3.2: Random Forest algorithm

Random Forest apply the principle of Decision Trees on multiple training samples built from the training set. One of the advantages of Deriving the use of Decision Trees to Random Forest is that we get rid of the overfitting problem: A single Decision Tree for a whole dataset can be very deep until we reach a desired accuracy threshold. Since Random Forests use bagging (it creates subsets of data with subsets of features), the trees are far more smaller than if we had to use one for the whole dataset. Each tree is small enough for us not to worry about overfitting. Another very interesting thing about Random Forests is that they don't use any mathematical formulas to select training samples and subsets of feature. It is still interesting to note that there exists a variation of Random Forest called Extremely Randomized Trees, or Extra Trees, where the randomness is pushed one step further. Instead of choosing the best splitting attributes among features (the most discriminative thresholds), thresholds are drawn randomly for each feature. This reduces the variance of the model but in counterpart, it increases the bias. Extra Trees models are not discussed in the present work.

3.3.3 Gradient Boosting

Gradient Tree Boosting or Gradient Boosted Decision Trees (GBDT), is an ensemble method that is also based on decision trees. In opposition to Random Forest, GBDT is a boosting techniques,

meaning that each tree is built sequentially to correct the inaccuracies of previous trees. This boosting technique places more emphasis on the data points that have been previously misclassified.While it is an efficient learning technique, GBDT has its disadvantages: First, a complex GBDT model is more prone to overfitting, as it is a boosting technique. As explained in Section 3.2.2, any Decision Tree algorithm can overfit the data if the tree is too deep. Since Random Forest uses bagging, it cannot overfit, as the final prediction is obtained by majority voting of all the trees built from subsets of the training set. However, GBDT should be used with a good knowledge and early-stopping of the trees, as it does not work with bagging. GBDT also has more hyperparameters to tune than Random forest: Besides the number of trees and the number of features that are common to both approaches, GBDT requires a learning rate and, just as it was discussed, a maximum tree depth. It is harder to fine-tune a GBDT model than a Random Forest model.

To understand why Gradient Boosting is called Gradient Boosting, let us formally describe the situation. Let F_0 be a first ensemble that is already built based on a learning sample. For any observation x of the learning sample, we can model the output $F_0(x)$ as

$$F_0(x) = \underset{\gamma}{argmin} L(y, \gamma), \tag{3.10}$$

where L is the loss function, y the true output value associated to x and γ the predicted value. For each evaluated observation, one can compute the residual error $r = y - \gamma$ and build a second model $h_1(x)$. Based on this new model, the output can be modeled as:

$$f_1(x) = f_0(x) + h_1(x) \tag{3.11}$$

As we proceed sequentially, the above operation is repeated and the expression can be generalized to

$$f_m(x) = f_{m-1}(x) + \alpha h_m(x), \tag{3.12}$$

where α is a parameter that will be discussed. This equation is the general equation for any boosting algorithm. The value of α depends on the learning method that is used. In the case of Gradient Boosting, we incorporate the principle of gradient descent into boosting. At each iteration of the boosting algorithm, we can compute the pseudo residuals $r_{i,m}$ as

$$r_{i,m} = -\left[\frac{\delta L(y_i, F(x_i))}{\delta F(x_i)}\right]_{F(x)=F_{m-1}(x)} for\ i = 1, ..., n, \tag{3.13}$$

where $F(x_i)$ is the previous model, m is the number of Decision Trees and n is the size of the learning sample. The loss function of the classification problem is defined as:

$$-\sum_{i=1}^{n} y_i log(p) + (1-p) log(1-p) \tag{3.14}$$

where p is the output probability (probability that an observation belongs to a certain class). we compute the next model with the goal to minimize the loss function based on the pseudo residuals. Instead of using the original outputs as targets, we use the residuals. Let us say that $h_m(x)$ is the Decision Tree built based on these residuals. The output value of a leaf is the value of γ that minimizes the Loss function such that

$$\gamma_m = \underset{\gamma}{argmin} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)). \tag{3.15}$$

The second argument of the loss function is similar to Equation 3.12. Before looping again for each sequence (each tree building) of the boosting algorithm, one should update the model:

$$F_m(x) = F_{m-1}(x) + \nu_m h_m(x) \tag{3.16}$$

where F_{m-1} is the previous model ν_m is the learning rate and h_m is the tree model made on residuals. The learning rate is a value between 0 and 1 and should be carefully chosen. A high lr can lead to faster convergence since the model adapts faster to the training set, but this convergence can be towards a local minimum which is suboptimal. A lower learning rate requires more iterations but is generally more stable. Moreover, it acts as a form of regularization, as it lessens the contribution of each tree, leading to less overfitting.

There exist multiple boosting techniques that differ by the way the model is updated. Where Gradient Boosting uses the negative gradient of the loss function from previous predictions to build new weak learners, the AdaBoost algorithm assigns weights to data points. Higher weights are given to the points that were misclassified at the last iteration. The loss function used for both variations is different too: GBDT uses the deviance and Adaboost minimizes the exponential loss function. Because of the exponentiality, the misclassified examples are penalized more heavily. Adaboost does not need to specify the maximum tree depth, as it uses only one-node trees. Moreover, it does not use any learning rate as the final predictions are simple weighted sums. There are multiple differences between every boosting technique and in the context of this work, GBDT and Adaboost will be the two whose performance will be compared. We will however see in the next chapter that there exist a more recent boosting technique that fills in most of the disadvantages of GBDT. This algorithm is called XGBoost (Extreme Gradient Boosting) and will be tackled in Section 4.4.

3.4 Metrics

This section consists of a discussion about the use of accurate measurements in the evaluation protocol of the classifiers. There is a need to determine the best metrics, as the case of unbalanced binary classification requires to search deeper than simply considering classification accuracy. It is also important to note that with the diversity of the exploited strategies, some models directly gave as outputs class labels while other gave probabilities of class membership. The defined metrics can only be used in the case of probabilities once a threshold value has been set. A discussion about the relevant metrics for the problem (Section 3.4 to 3.4) is followed by the different strategies to use for choosing this threshold (Section 3.4 to 3.4.1).

Confusion matrix

Binary classification is a problem whose methods of evaluation depend on the context of the problem. Some problems require to maximize the detection of positive (P) (resp. negative (N)) cases without paying attention to the fact that false positive (resp. negative) cases can be detected, while some problems require a cutoff between correctly and wrongly predicted outcomes. One of the most widely used tools in binary classification is the confusion matrix. Most of the metrics used in classification problems are based on this artifact.

The confusion matrix is a two by two table formed by counting the number of the four outcomes of a binary classifier. We denote as true positive/negative (TP/TN), the cases that were correctly predicted as positive/negative by the classifier, while the false positive/negative (FP/FN) are the cases that were wrongly predicted as positive/negative by the classifier with respect to the groundtruth values. It is needless to say that in the context of churn prediction, we refer to positive cases as the cases of clients who effectively churned. Figure 3.1 represents the definition of the confusion matrix in the context of churn prediction. Based on this matrix, some basic measurements are derived, each of them being useful depending on the situation:

- **Error rate (ERR)**: The number of all incorrect predictions divided by the total number of samples.

$$ERR = \frac{FP + FN}{TP + TN + FP + FN} \tag{3.17}$$

Confusion matrix		Predicted	
		Positive	Negative
Observed	Positive	Churned client detected as churned client (TP)	Churned client detected as non-churned client (FN)
	Negative	Non-churned client detected as churned client (FP)	Non-churned client detected as non-churned client (TN)

Table 3.1: Definition of the confusion matrix in the context of churn prediction

- **Accuracy (ACC)**: The number of all correct predictions divided by the total number of samples.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} = 1 - ERR \tag{3.18}$$

- **Sensitivity (SN)**: Also called recall or true positive rate (TPR), it is the number of correct positive predictions divided by the total number of positive samples.

$$TPR = \frac{TP}{TP + FN} \tag{3.19}$$

- **Specificity (SP)**: Also called true negative rate (TNR), it is the number of correct negative predictions divided by the total number of negative samples.

$$TNR = \frac{TN}{TN + FP} \tag{3.20}$$

- **Precision (PREC)**: It is the number of correct positive predictions divided by the total number of positive predictions.

$$PREC = \frac{TP}{TP + FP} \tag{3.21}$$

- **False positive rate (FPR)**: It is the number of incorrect positive predictions divided by the total number of negatives.

$$FPR = \frac{FP}{FP + TN} \tag{3.22}$$

In the context of churn prediction, the choice of the relevant concepts among this list is discussed around two aspects. We first want to assess a model's general performance by observing how it classifies positives as well as negatives samples. Since the problem is very imbalanced, there is a need to scale the classification accuracy of both categories (see Section 3.4). In a second time, the case of churn prediction has as center of attention the efficient detection of churned clients (i.e. positive cases). With this assumption, it would only be natural to choose measurements such as sensitivity and precision (see Section 3.4).

Balanced accuracy

To understand why raw accuracy is not a good metric for this problem, let us have the following example: In a binary classification problem where the class of interest is the least dominant, we have 1000 samples with only 100 samples that belong to the class of interest. A classifier that would classify every sample as a member of the most dominant class would have an accuracy of 0.9, even though it didn't detect any sample belonging to the class of interest. This is a problem, since the priority in this case is to classify these samples accurately. For such a matter, a metric called **balanced accuracy (B-ACC)** is introduced. It is defined as the arithmetic mean of the sensitivity and specificity, i.e. we have from 3.19 and 3.20

$$\text{B-ACC} = \frac{TPR + TNR}{2} = \frac{1}{2}\frac{TP * TN + TP * FP + TN * TP + TN * FN}{TP * TN + TP * FP + FN * TN + FN * FP} \tag{3.23}$$

Taking our example, the balanced accuracy would be equal to 0.5 ($TPR = 1$ and $TNR = 0$) which is a better way of observing that the classifier is not a good fit (if the primary goal is to detect the least dominant class). Most classifiers tend to jointly maximize sensitivity and specificity, and balanced accuracy is a good trade-off between both.

F1 score

Focusing on the least dominant class (churned cases), it is now important to see how well detected cases indeed belong to the positive class. The two questions to ask are: *"Which proportion of true positive cases are detected as positive ?"* and *"Which proportion of positive detected cases are indeed true positive cases ?"*. Sensitivity and Precision are the metrics of interest. The **F-score** is defined as a harmonic mean of these two measurements.

$$F = \frac{(1 + \beta^2)(PREC * TPR)}{\beta^2 * PREC + TPR}, \tag{3.24}$$

where the value of β is discussed in the following paragraph.

The value of β will depend on the priority we want to give to the two measurements that are used in the definition of the F-score (3.24). A small β value will give more weight to precision while a high β value will give more credit to sensitivity. Usually, β takes values ranging between 0.5 and 2. The **F1-score** is a particular case of the F-score function where $\beta = 1$. In this case, we have

$$F1 = \frac{2 * PREC * TPR}{PREC + TPR}. \tag{3.25}$$

A higher value such as $\beta = 2$ could also have been chosen, because in this case, it is assumed that since the imbalance is quite important, there will be a lot of cases that are wrongly detected as positive whatever the classifier, hence the precision will be low. This is why sensitivity should be prioritized (the main objective is the minimization of false negatives (FN)). Choosing $\beta = 1$ is a good trade-off between the minimization of false negatives and false positives.

Thresholding

"Thresholding" or "Decision Threshold tuning" is defined as the task of finding the optimal threshold value that will serve to determine how a given output probability will be converted into a class label. There are multiple ways to obtain this value: Some of them are directly computable, while some other require domain knowledge or data analysis.

In a binary classification problem, the default threshold value is equal to 0.5.An output probability inferior to 0.5 results in a class label of 0 while a probability superior to 0.5 results in a class label of 1. However, several problems can arise from this biased choice:

- Uncalibrated probabilities: Some models output probabilities that are not calibrated such as decision trees or SVMs (the output probabilities of such models do not correspond to the true likelihood of the event).
- Use of metrics: The metrics used to train the model can be different than the ones used to evaluate it (which is not our case).
- Skewness: The class distribution is severely skewed/imbalanced.
- Classification cost: Some problems put priority on the accurate classification of one particular class by associating costs to misclassification. Different costs result in a sub-optimal interpretation of the output probabilities.

A naive method of obtaining the optimal threshold value consists in choosing an evaluation metric, computing class labels for each possible threshold value and keep the value for which the metric scores

the highest. The problem with this method is that it requires to choose a specific metric (choosing multiple metrics can lead to the fact that not all of them are maximized for a particular threshold value). Nevertheless, if a metric is proven to be the most useful, then this approach still remains efficient in most cases. Besides that, one can choose to combine the use of concepts based on the confusion matrix with visual representations that are built with respect to the evolution of a threshold value. Two of the most common visual tools are the precision-recall curve and the ROC curve. Since the reasoning behind both representations is quite similar, only one of them is tackled in this paper (see Section 3.4).

ROC and AUC

The Receiver Operating Characteristics (ROC) plot is a display of the evolution of sensitivity/1-specificity pairs of a classifier with respect to the threshold value that a probability needs to reach to be considered as part of a class of interest. This curve is one of the most common tools used in binary classification for decision threshold tuning. With a similar reasoning as when choosing metrics, the choice of this cut-off depends on the main objective of the classification problem. Maximizing the prediction capacity of positive cases involves prioritizing a high sensitivity while minimizing the number of true negatives predicted as positives. If specificity is defined as the proportion of correct negative predictions among all negative samples (3.20), the 1-specificity refers to the probability that a true negative case will be classified as positive. The ROC cut-off problem thus involves finding a threshold value that will maximize the sensitivity while minimizing the 1-specificity.

Figure 3.3 shows examples of different ROC curves for the same problem. Intuitively, the optimization problem that is tackled requires the curve to be as close as possible to the top-left corner (maximization of ordinate and minimization of abscissa). The best classifier will then be defined as the model with the highest **Area under the ROC curve (AUC)**. A perfect classification corresponds to an AUC of 1 while a classification that predicts none of the samples correctly will have an AUC of 0. It also means that any trained classifier should have at least an AUC of 0.5, as this value is the result of a classifier that randomly classifies samples as positive or negative (assuming that the chances of the classifier predicting a case as positive or negative are equivalent to the proportion of positive/negative outputs in the test set). It is important to note that the ROC curve and the AUC are defined as model-wide metrics, meaning that they don't particularly focus on an aspect of a classification problem: they allow one to assess the overall performance of a classifier.

With a good analysis of the ROC curve and the rigorous understanding of the problem, one can define a threshold value that will serve as frontier between the positive and the negative class. There

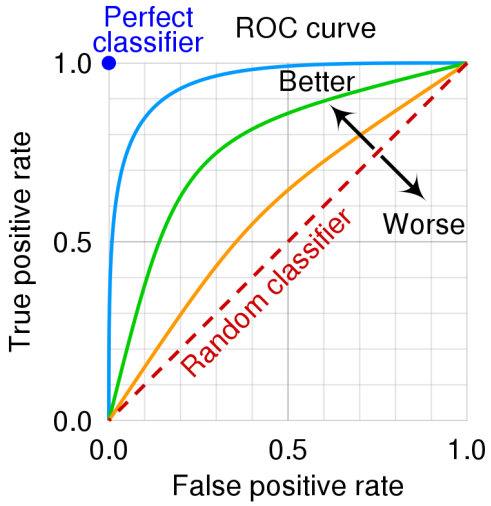


Figure 3.3: ROC curves of different classifiers. The blue curve is the best classifier while the orange curve is barely better than a random classifier. *(source)*

exists multiple selection criteria based on the ROC curve. The Youden J Index, introduced by Youden in the medical literature, is defined as

$$J(c) = SN(c) + SP(c) - 1, \tag{3.26}$$

where c is the cutoff value. The value of c that maximizes the J index is the optimal cut point. Another approach which is more intuitive relies on finding the threshold that will result in the closest point to $(0, 1)$ in the ROC plot. The ER criteria is defined as

$$ER(c) = \sqrt{(1 - SN(c))^2 + (1 - SP(c))^2} \tag{3.27}$$

In opposition to Youden's J Index, the best cutoff is the value of c that minimizes ER. There exists other recently proposed methods[15].

3.4.1 Evaluation

In the context of churn prediction, it was decided that the best way to assess a model's performance would be to look either at the balanced accuracy, the F1-score or the AUC for the last part of the project. It was observed that it was interesting to look at multiple metrics, as they all give different insights about classification performance. No priority is given to any of these three scoring methods.

In this section, the tools that were described earlier are applied in the context of churn prediction.We use the described classification algorithms to predict churn and we verify that the performance of models that are supposed to perform well are up to the task in this specific setting (see Section 2.3 for more details about data preprocessing). According to NRB Data specialist and machine learning experts, the models that are supposed to perform at best are Gradient Boosting models, Decision Tree classifiers and bagging techniques, with Gradient Boosting supposedly being the most efficient model for churn prediction.

Before going on with the evaluation of the most known models, one should remind that the Single Views are heavily unbalanced in terms of output. This difference has an impact on the training of the models, as they will take the habit of being confronted to negative churn cases, and not enough positive cases. Thus, they will specialise in recognizing negative cases among all cases, while the recognition of positive cases might remain too poor. There exist (at least) two solutions to this problem. We start with the assumption that output classes are balanced in the dataset using balanced sampling to train the models. One can also synthesize positive cases based on the nature of the existing positive cases. The problem with this method is that the Single Views contains very few positive cases and it has been shown that separating positive and negative churn cases is too laborious of a task ([1]). The synthesized data could very well not be representative of the original positive cases and introduce a heavy bias during the training, leading to important performance losses. Consequently, the solution that was chosen is to train and evaluate every model in two settings: a first setting where the training set contains 70% positive churn case and an equally sized sample of negative churn cases. The test set contains the rest of the positive cases and the totality of the negative cases. We will refer to this setting as **Balanced sample**. The second setting **Full Single View** corresponds to the evaluation with the complete data (Note: The data is preprocessed before being sampled, so both settings undergo the same set of transformations). The training set of the second setting contains 70% of the whole Single View, sampled randomly and the test set contains the remaining 30%.

Figure 3.4 represents the evolution of the (cross-validated) balanced accuracy and F1 scores of different known models for the Car insurance type. Appendix .4 contains the same measurements for Fire insurance types. As it was forecast by the experts, Gradient Boosting is the most efficient on average with results peaking to more than 70% balanced accuracy. Some algorithms such as Naive-Bayes offer remarkable performances at some points, but these performances are too irregular for us to trust

[1]Note: The only time where separation between the cases was possible was during supervised dimension reduction (Section 2.2.11). However, these transformations are not revertible

the algorithm. It can also be seen that for almost every case, the performance of the balanced setting is greater than the unbalanced setting. It would appear that it is a good thing to set the prior assumption that the positive and negative churn cases are balanced, as it improves the models' performances. In the set of models, Naive-Bayes seems to be the only one that is able to make the difference when it comes to unbalanced data. Logistic regression and AdaBoost seem to perform quite well as they win over Gradient Boosting in some cases during one or two months. Random Forest is the black sheep as it is the least performing model on average.

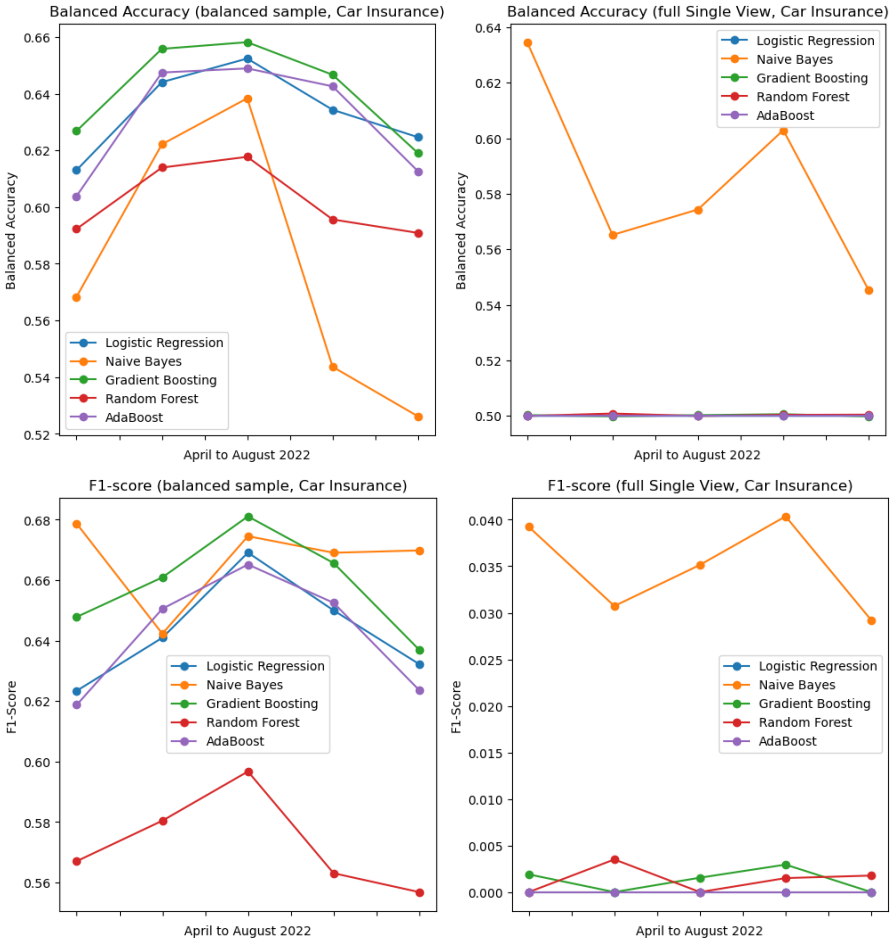


Figure 3.4: Evolution of the cross-validated balanced accuracy and F1 scores (Car Insurance, April 2022 to August 2022).

While the results that are found are acceptable, they require further fine-tuning. Instead of wasting a lot of time on optimization of the models, it was decided to move on to the next objective. If we remember well, the third and last objective of the work was to elaborate a model architecture that would remain persistent throughout time. Instead of retraining a model from scratch each month using only the most recent data, the idea would be to use a model built from previous months and continue to improve its performance by taking into account more recent data. The discussion about this challenge takes place in the last chapter of the present work.

Chapter 4

Continuous learning

Pipeline architectures have always been a widely-used tool for companies. They offer automated movement and transformation of the data by defining dedicated data-driven workflows. Famous platforms such as Amazon or Google developed infrastructures that allow companies to benefit of these Web services facilities. In the case of D-predict, a new pipeline is instantiated each month such that everything is automated, from the raw Single Views to the associated predictions for these Single Views. Because each instance is a new instance, it only takes into account the present data, i.e the two Single Views for training and inference (spaced 3 months from each other). Each pipeline instance trains a model only based on data from 3 months ago (most recently observed data) without taking anything else into account. Besides the fact that we cannot change this 3 months duration, as it was defined by Ethias, this training method is improvable. One could for example take into account multiple months before the most recently observed data to account for the evolution of the customer base. This could be done by training a model with a training set composed of multiple months of data. However, with the actual pipeline architecture, this would require loading multiple Single Views (more than the 2 initial) each month, leading to a high memory usage. There is another indirect way to take into account past data that can be implemented by ensuring continuity in the models. This alternative is the focus of the discussion in this chapter. Since it is not advised (and practically hard) to load multiple Single Views each months and it adds redundancy, we look to use the most efficient models that were previously identified to build variations of these models that would be able to persist in time while being frequently updated. Instead of training new models from scratch each month and select the best for each month, we look to reuse past models that were trained on past data and update these models each month. This process can be assimilated as Continuous learning.

The current pipeline module consists of 4 blocks : Data Loading, Preprocessing, Training and Inference. The Data Loading block is responsible for loading two Single Views each month: the unlabeled Single View of the current month (for inference) and the labeled Single View of the most recently observed data (4 months old data used for training). The preprocessing block consists in all the data transformation that is performed, as depicted in Section 2.3. A pipeline instance that accounts for continuity should add Model Loading/Saving , Predictions Loading/Saving and Predictions Scoring. The model should be loaded and saved, as the main objective of the chapter is to retrieve the same model and improve its training each month. The predictions made each month should be saved, as they are necessary for obtaining feedback. In order to keep all predictions until the labels are obtained, there is a need to use a queue to buffer the predictions that are waiting for feedback. Figure 4.1 gives a schematic view of the queuing process. The size of the queue is equivalent to the number of months that are needed to obtain the true output values of a datamart. The predictions are retrieved from the queue once the queue is full. Figure 4.2 gives a schematic overview of the pipeline module in its static form and its transformed version intended for continuous learning. Ideally, the feedback obtained with Predicstions Scoring could be used for monitoring and eventually bring changes to the model. For instance, repeated drops of accuracy could lead to a complete or partial reset of the model, if reverting such a model to one of its previous states is possible.

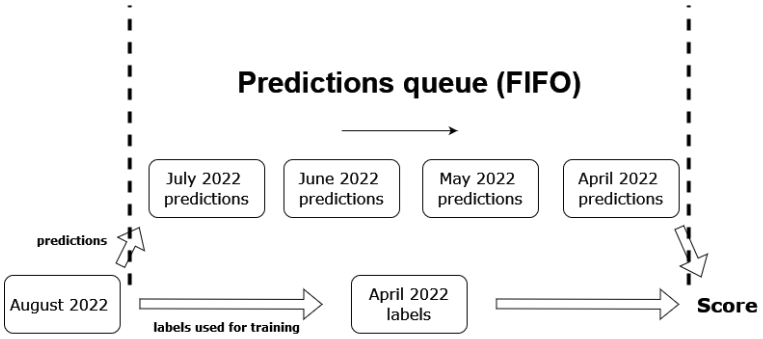


Figure 4.1: Functioning of the predictions queuing process in the pipeline module.

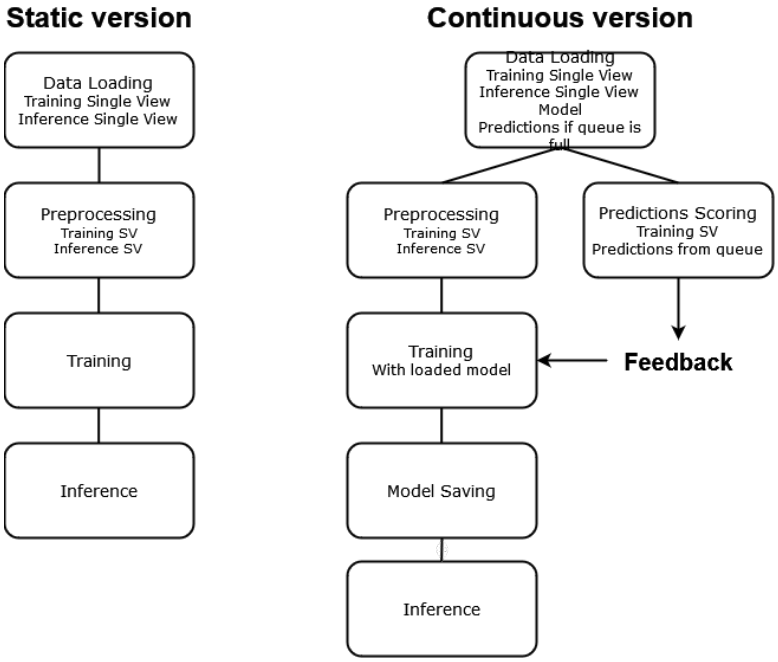


Figure 4.2: Schematic comparison between the static pipeline module and the continuous module.

4.1 Historical vs. Real-time data

Up until now, churn prediction was performed with the assumption that the output labels were directly available, as we built the models on historical data. The pipeline process that takes place in real-time does not have access to the labels of the current month's Single View. What is meant is that each month, the pipeline computes predictions, and the true output values are obtained 3 months later. In the current pipeline, there is absolutely no feedback implemented when it comes to the evaluation of the inference 3 months ago. What we need to add to this architecture is a feedback system that returns useful statistics. This system would serve two purposes as it would allow to monitor the evolution of the performance of the inferences: First, the analyst could evaluate if the predictions were accurate and take measures. Secondly, since we use a model that persists over time, this monitoring could serve as a controller of the continuous learning phase of the model. Depending on the possibilities related to each continuous model, one could revert the model to one of its previous states if the inference performances start to drop or even reinitialize it if we consider that the training has been going on for too long. If the performance becomes stagnant, it would be advised to fine-tune the learning algorithm.

4.2 Batch vs. On-line learning

We define algorithm continuity as the ability of a learning algorithm to continuously learn without any limit while training data is available. A continuous algorithm is able to start its training based on a first subset of data d_1, interrupt its training, perform inferences and then resume its training with others subsets of data $d_2, d_3, d_4, ...$ while being able to compute predictions at any time in the process.

Algorithms that are able to learn while performing inference at any time are called on-line algorithms while algorithms that are not able to are called batch learning algorithms. In the batch learning category, we have seen the KNN algorithm that stores the training data and classifies new instances based on their proximity with existing data points. If new data is nested to this training set, the algorithm cannot adapt without retraining. Decision trees, in their canonical form, are built on a static dataset. Just like KNN, if some new data is loaded, they have to be rebuilt from scratch. It is also the case for Support Vector Machines and the Naives Bayes algorithm. Dimension reduction techniques such as PCA that was introduced in Section 2.2.11 are also performed on static data, as any new data would change the whole structure of the projection. To build efficient on-line variations of learning algorithms, we could consider Ensemble methods. This family of methods uses weak learners, i.e a set of similar simple models, to perform predictions that originate from a consensus between the weak learners. One of the idea behind On-line learning is to alter the structure of the Ensemble such as to have room to account for the new data. It can be done by adding new weak learners, removing them, or retraining some of them.

4.3 On-Line Random Forest

Taking Random Forest as an example, an On-line version of the algorithm would consist in initializing a model with a number of trees n_trees that is inferior to a parameter that indicates the maximal number of trees called max_n_trees. The first set of n_trees is trained on the first set of training data. If inference is already required at this point, the model uses these n_trees trees to compute predictions. Then, when new training data is available, we add a certain number of new trees to n_trees while ensuring that the sum of trees is still inferior to max_n_trees. The new training data is used for these new trees. When n_trees is equal to max_n_trees, we can decide either to remove the n oldest trees to add n new ones, or remove more trees if the performance of the model was better at some point in time. This illustrates well why feedback is important, as we will be able to benchmark the moment where the algorithm performed best and eventually revert it to a state prior to its peak performance.

4.3.1 Iterative Ensemble of On-Line Random Forests

The Iterative Ensemble of Random Forest is an adaptation of the idea of weighted Random Forest and combined sampling proposed by V. Effendy et al. (2014). The main idea is to combine multiple samples of a training data by using multiple Random forests and obtaining the final prediction as a weighing of all the predictions made by the different models. In the context of this work, we develop an On-Line version of the concept with the use of an object that is called Iterative Ensemble RF. It is defined as an object that has several Random Forest that are updated each month. Each Random Forest is trained based on a different subset of data. These subsets of data are all composed of the same majority of positive churn cases, the sole difference in the subsets being the negative churn cases. For an IERF object that contains n Random Forest, the model is updated by creating n training subsets that contain the positive churn cases along with different random samples of negative churn cases such that each subset is balanced with different negative churn data. In terms of implementation, an IERF is an object for which 4 procedures are implemented (besides the constructor of the object): A training procedure, An inference procedure, A model update procedure and a model saving/loading procedure. This model works as follows: Just as On-Line Random Forest, the IERF object initializes n Random Forest models. When new data is available, each RF is trained on a separated balanced subset. The model is then updated and saved. Whenever inference is required, the IERF object is loaded. The final

prediction is obtained by soft-voting of the n Random Forest models (more details about soft voting in Section 3.3.1). This final prediction is a "vote of votes", as Random Forests are averaging models. The difference with classical Random Forest is that additional trees are added to each Random Forest such that the Ensemble can keep on training. This requires setting a maximum number of trees and an increment to define the number of trees added for each new training session.

4.3.2 Evaluation protocol

During experiments, it was determined by cross-validation that random forest models that were trained based on one month of data (one small balanced sample of data of one month) had the highest results when the number of estimators was between 10 and 30. It was then observed that when the ensemble exceeded 50 estimators, drops of performance were starting to be observed. These experiments led to the choice of two important parameters: As the Iterative Ensemble is composed of r Random Forest models (the default value of r is arbitrarily set to 5), one should initialize these r models with a certain number $n\_estimators$ of estimators. Moreover, we cannot add trees to each Random Forest indefinitely, as it would lead to overfitting and drops of performance. The number 50 was chosen as value for $max\_n\_estimators$. Finally, we need an increment for the model, i.e. a number of trees to add to each forest at each new month. Since it is more practical to choose a multiple, the increment value is chosen as 10. Now, if we proceed with continuous training, the first month consists of 5 Random Forest of 10 estimators each that are trained on 5 balanced subset of this month's most recently observed data. The second month would then consist of the same 5 Random Forest to which we fit 10 more estimators, so a total of 20 estimators: 10 estimators fitted on month $m-1$ and 10 estimators fitted on month m. Once the 5th month is reached, The 10 oldest estimators are discarded and replaced by 10 new estimators that are fitted on the new most recently observed data.

With the described procedure, we can now observe the evolution of the model's performance over the months. Figure 4.3 represent the evolution of the model's performance over a 5 months period. These performances are computed based on the test sets of the observed Single Views. The performance on the test sets are quite constant but it appears that adding estimators didn't lead to better results. There is a possibility that Random Forests are capped in terms of capability for this specific context, as they cannot exceed certain value even if the training phase is long and steady. Let us take a closer look at how the algorithm classifies the data. Appendix .5.1 gives the Confusion matrices of all 5 test sets from April to August 2022. It is by looking at these matrices that one can get aware of the difficulty of the task. Because of the very low churn rate, for one correctly predicted positive churn case, between 25 and 50 negative cases are also predicted as positive. Among all cases predicted as positive, there is only a probability of 2% to 4% that the case is indeed a true positive. It is clear that there is room for improvement that might be found with other machine learning techniques.

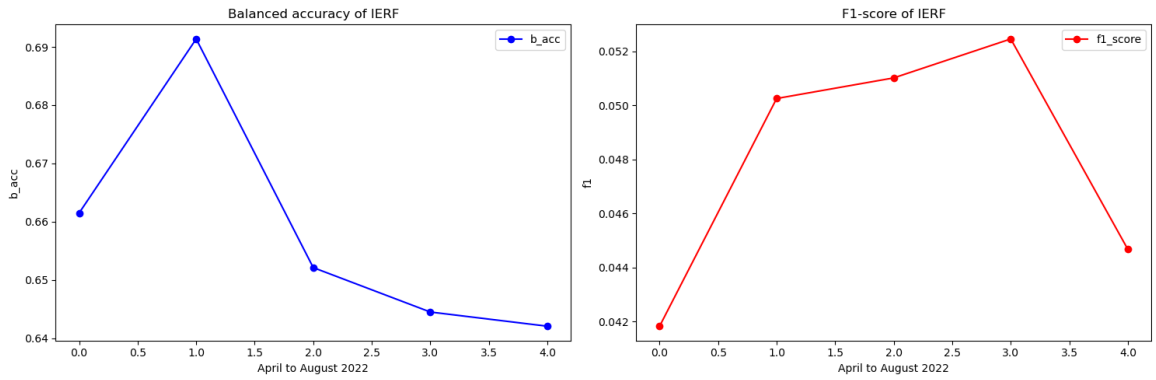


Figure 4.3: Evolution of b_acc and f1_scores of the test sets from April 2022 to August 2022 (Car Insurance)

4.4 On-Line Gradient Boosting and Extreme Gradient Boosting

The previous chapter led to the conclusion that Gradient Tree Boosting techniques are the most efficient for our binary classification problem, so it is a logical follow-up to try and push this optimization to the extreme by looking at what could be improved with Gradient Tree Boosting. Extreme Gradient Boosting is a variation of GBDT that addresses the majority of its issues. This Scalable Tree Boosting System proposed in 2016[3] is a revolutionary tool that has already been adopted by many fields of activities, as it is one of the most state-of-the-art machine learning models of nowadays. XGBoost brings minor improvements into the implementation of GBDT, but without going too much into its details, let us list some of these improvements.

- Task distribution: In the literature dedicated to the description of the XGBoost System, a module called Column Block is introduced. The purpose of this module is to perform parallel learning, which consists in separating the different operations performed during the training process to reduce computation complexity. Additionally, the XGBoost technology works with a Cache-Aware Access, meaning that an internal buffer is allocated for each thread such that the gradient's statistics can be fetched in it. This is especially useful when these statistics do not fit in the CPU. These resources create more comfort in big data settings and can efficiently accelerate the learning phase.

- Regularization: XGBoost adds a penalty term to the loss function (either Lasso or Ridge regularization) to increase its punitive power. As a consequence, it is less prone to overfitting as it lowers the variance, but it adds slightly more bias.

- Missing values: Historically, the first public version of XGBoost was not able to manage missing data. However, recent experiments have led to a version of XGBoost capable of imputing Missing Data [4]. What is interesting is that the imputation technique that is used is based on MICE, just as the method described in Section 2.2.8.

- Imbalanced data: The variation of GBDT also offers the possibility to initialize weights for classes to account for class imbalance. This will be very helpful in the context of churn prediction, as this imbalance is severe.

XGBoost is an algorithm that requires a lot of fine-tuning, as it can be used for various applications and its set of hyperparameters is wide. This section compiles the experiments that were performed to get a fine-tuned optimized version of the algorithm. First, it is important to provide a baseline of a default performance, as the algorithm was not tested in Section 3. For this purpose, the algorithm is instantiated without any modification of the parameters. It is trained using a full Single View preprocessed as explained in Section 2.3 and evaluated with 3 series of 10-fold cross-validation using 3 scoring metrics. The values obtained are reported in Table 4.1. These values will serve as a baseline for the rest of our experiments. The first observation to be made is the very low mean balanced accuracy and f1-score that probably comes from the class imbalance. The AUC remain high, as negative churn cases are dominant and most of them are classified as negative, which gives a high quantity of True Negative in comparison with the total size of the Single View.

Single View	balanced accuracy	f1-score	AUC
Car	0.50018	0.00079	0.6850
Fire insurance (loc)	0.50122	0.00081	0.6859
Fire insurance (prop)	0.50093	0.00077	0.6855

Table 4.1: Cross-validated mean scores of a default XGBoost Classifier.

The next thing there is to know about XGBoost is that we can choose to consider a prior class imbalance. The class weighting is a parameter that can also be tuned. We know how the class are weighted (close to 99% for each insurance type) but it does not necessarily mean that choosing the true weighting will lead to the best results. The weight tuning is done by performing a variation of

cross-validation that will compute scores with models that differ in weight initialization. As it is our main metric, we use two series of 5-fold cross-validation with the balanced accuracy as the deciding score. The result are reported in Table 4.2. We can see that the scores have improved since the default testing, with a peak value reached at $w = 1000$ (w is the expected ratio of negative per positive cases), which is more than 10 times greater than the true ratio (+- 99). Choosing good class weights can increase the balanced accuracy by almost 10% in best case scenarios.

Car Insurance	w=1	w=10	w=50	w=100	w=1000	w=10000	w=100000
balanced accuracy	0.500137	0.503340	0.550412	0.588775	0.596142	0.584093	0.580536

Table 4.2: Cross-validated mean balanced accuracies per weight value w.

The difference between class weights and other hyperparameters such as learning rate or number of weak learners is that the class weight can be tuned based on only one Single View, as we consider that the proportion of churned customers is similar for each Single View of a same Insurance Type. However, the learning rate and the number of weak learners cannot be tuned based on only one Single View, as the goal is to have one individual persistent model for each Single View. What is meant is that if the training was limited to one month (one Single View), the values of learning rate and weak learners that would have been found by fine-tuning are not the same as if fine-tuning was performed on a long term basis. The learning rate of a 1-month training should be higher than a continuous learning rate, as the model needs to adapt to the training data much faster. Moreover, the number of estimators needs to be large enough in order for the new training phase not to have a too large impact in comparison with the previous training phases. There is a possibility of performing cross-validation over multiple months of data, but it would present some issues in terms of memory usage and computation power. Still, we have the reasons to state that the learning rate should be smaller than for one month and the number of estimators should be higher. Let us proceed with some more cross-validation. This time, we use the previously found class weighting with different combinations of learning rate and number of estimators. Each combination is evaluated using a 5-fold cross-validation with balanced accuracy as scoring metric. The results are reported on Table 4.3. The peak accuracy is obtained with the highest learning rate and the highest number of estimators, meaning that we should turn to a even higher number of estimators when performing continuous learning.

Learning rate Number of estimators	0.01	0.001	0.0001
100	0.547642	0.549800	0.551018
200	0.556126	0.548614	0.550038
300	**0.559002**	0.548317	0.549909

Table 4.3: Cross validated mean balanced accuracies w.r.t learning rate and number of estimators (Car Insurance, April 2022).

Now that we have a good idea of the hyperparameter values, we can proceed with the continuous learning process. Just as with On-Line Random Forest, we instantiate a XGBoost Model with a sufficient number of weak learners. At the first month, the model is trained on the most recently observed data and performs inference for the month's current data. On an indicative basis, the tests are performed for the Car Insurance type. We look at Figure 4.4 for the evolution of scores and Appendix .5.2 for the confusion matrices. Here, in opposition with Random Forest, the model seems to be greatly influenced by the initialized class weights, as they detect more positive churn cases that are indeed true positive cases. In counterpart, the proportion of false positive is tremendous. The class weight is a strategy that can influence a model towards higher classification of the least dominant class, but it has a serious impact on the classification of the dominating class. One hypothesis is that this strategy induces a lot of bias and is not robust for any kind of problem.

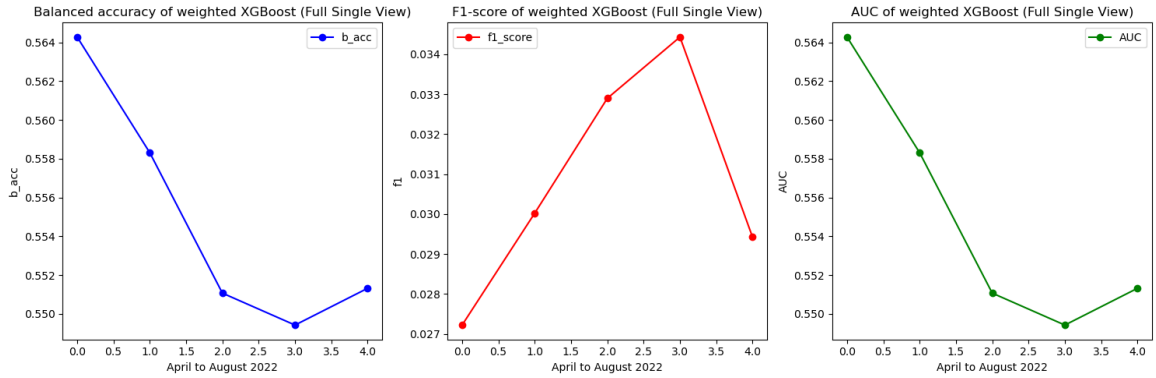


Figure 4.4: Evolution of the scoring metrics on the different test sets (Car Insurance)

Another solution is to apply the principle of combined sampling to XGBoost just as it was used with On-Line Random Forest. This way, we get rid of the weights initialization, as the classes are already balanced by undersampling the negative class. Appendix .5.3 displays the evolution of the metrics of an XGBoost Classifier that uses combined sampling. This version is quite better than the weight initialization version.

4.4.1 Algorithms comparison

Let us now compare the different algorithms described in this chapter. We first described an On-Line variation of the Random Forest algorithm that uses the principle of combined sampling and soft voting. We then introduced the possibility of using Extreme Gradient Boosting in an On-Line fashion. We experimented with a version that is trained on full Single Views by initializing class weights and finally, we experimented with a XGBoost classifier with the combined sampling idea of On-Line Random Forest. We also compare these continuous models with the performances of the best non-continuous model to see if there is a significant difference of performance between the static and the continuous approach.

As it was explained earlier, the pipeline performs inference on new data each month and the outputted predictions are kept until the associated true outputs are received. In our case, since the experiments were performed from April 2022 to August 2022, the comparison will be done for the inference made in April 2022, as the true outputs associated to these inferences are obtained in August 2022. Table 4.4 gives a summary of the performance of the different models.

Car Insurance, April 2022	b_acc	f1-score	AUC
Non-Continuous Gradient Boosting	0.6274	0.0204	0.5632
On-Line Random Forest with combined sampling	0.6354	0.0447	0.6445
Weighted XGBoost	0.5314	0.0293	0.5494
XGBoost with combined samples	0.6391	0.0399	0.6446

Table 4.4: Comparison of the different learning algorithms based on inference For Car Insurance (feedback).

In most cases, the classification is similar. At first sight, one would argue that it is not necessary to keep the same model each month, as the difference between non-continuous and continuous methods is small. But the problem seems to be somewhere else. Let us take the current situation: Each dataset that requires inference of the model can only have access to up-to 3 months old data and not any data in this 3 months interval, as it is the duration for receiving true outputs. The questions that come to mind are: "For how much months do one need to train a model before it actually starts being successful?", as 5 months don't seem to be enough in this case. Another question to be asked is "What if the duration of the waiting time was reduced to one month instead of 3?". Since we

have access to historical data, the last question was probed by training an On-Line Random Forest model with combined sampling on the Single View of July 2022 and perform inference for the month of August of the same year. Appendix .5.4 show the difference in score if the duration was reduced. It is needless to say that the differences observed are a direct cause of the highly multi-varying datasets.

The exposed continuous learning models did not lead to significant results, as the differences in performance between static models and updated models are weak. The main cause for this indifference could be that the duration of the training is too short in comparison with the gap between a month where data is received for inference and the month where its outputs are obtained. For further improvement, a long-term experiment should be conducted, with a larger training duration, as 5 months worth of training don't seem to have much impact on the inferences. The m month gap between inference and receiving of the outputs is also an issue, as it has been seen that training a model with a shorter gap leads to higher performances (Appendix .5.4).

Bibliography

[1] Carsten Wasserfuhr Andreas Groll and Leonid Zeldin. Churn modeling of life insurance policies via statistical and machine learning methods - analysis of important features. 2022.

[2] Leo Breiman. Pasting small votes for classification in large databases and on-line. *Machine Learning*, 1999.

[3] Tianqi Chen and Carlos Guestrin. XGBoost. aug 2016.

[4] Yongshi Deng and Thomas Lumley. Multiple imputation through xgboost. 2023.

[5] Veronikha Effendy, Kang Adiwijaya, and Abdurahman Baizal. Handling imbalanced data in customer churn prediction using combined sampling and weighted random forest. *2014 2nd International Conference on Information and Communication Technology, ICoICT 2014*, 05 2014.

[6] Amy Gallo. The value of keeping the right customers. 2014.

[7] William Heavlin. On ensembles, i-optimality, and active learning. *Journal of Statistical Theory and Practice*, 15, 09 2021.

[8] White Ian R. Seaman Shaun R. Carpenter James R. Tilling Kate Sterne Jonathan AC Hughes, Rachael A. Joint modelling rationale for chained equations. *BMC Medical Research Methodology*, 2014.

[9] Saravana Kumar. Customer retention versus customer acquisition. *Forbes*, 2022.

[10] Aurélie Lemmens and Christophe Croux. Bagging and boosting classification trees to predict churn. *Journal of Marketing Research*, 43, 05 2006.

[11] Wei-Chao Lin, Chih-Fong Tsai, and Shih-Wen Ke. Dimensionality and data reduction in telecom churn prediction. *Kybernetes: The International Journal of Systems Cybernetics*, 43, 04 2014.

[12] C. Gary Mena, Arno De Caigny, Kristof Coussement, Koen W. De Bock, and Stefan Lessmann. Churn prediction with sequential data and deep neural networks. a comparative analysis. 2019.

[13] David Hason Rudd, Huan Huo, and Guandong Xu. Causal analysis of customer churn using deep learning. dec 2021.

[14] S.M.Jai Sakthi, N Gayathri, Kanumuri Uma, and Vijayan Vijayarajan. Customer churn prediction using stochastic gradient boosting technique. *Journal of Computational and Theoretical Nanoscience*, 15:2410–2414, 06 2018.

[15] Ilker Unal. Defining an optimal cut-point value in roc analysis: An alternative approach. *Computational and Mathematical Methods in Medicine*, 2017:1–14, 05 2017.

[16] Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3):1–67, 2011.

[17] Rong Zhang, Weiping Li, Wei Tan, and Tong Mo. Deep and shallow model for insurance churn prediction service. pages 346–353, 06 2017.

[18] Petra Posedel Šimović, Davor Horvatic, and Edward W. Sun. Classifying variety of customer's online engagement for churn prediction with mixed-penalty logistic regression. 2021.

Appendix

.1 Dataset - Description and Nomenclature

Note: The following file contains all the available descriptions about the variables in the Single Views. Some variables discussed in the analysis could possibly be missing a description.

Attached file : double click on the following icon 
(*Some PDF readers do not support file attachment, try using Adobe Acrobat Reader*, *Foxit Reader* or *Wondershare PDF elements if you cannot open the file).*

.2 Dataset - Statistical features

Table 5: Numerical features (car insurance, April 2022)

	count	mean	std	min	25%	50%	75%	max
CTT_AUT_DMA_EHP	280591.0	6.613897e-02	2.485253e-01	0.0	0.0	0.0	0.0	1.0
CTT_AUT_MDM_EHP	280591.0	2.044720e-01	4.033160e-01	0.0	0.0	0.0	0.0	1.0
CTT_AUT_PJ_EHP	280591.0	2.713237e-01	4.446435e-01	0.0	0.0	0.0	1.0	1.0
CTT_AUT_ASC_EHP	280591.0	8.622087e-01	3.446814e-01	0.0	1.0	1.0	1.0	1.0
CTT_AUT_youngdriver_EHP	280591.0	1.463197e-01	3.534271e-01	0.0	0.0	0.0	0.0	1.0
CTT_AUT_value_EHP	280591.0	1.334063e+04	1.260442e+04	-1.0	-1.0	13576.0	21418.0	185099.0
CTT_AUT_puissance_EHP	280591.0	7.820430e+01	2.653638e+01	4.0	59.0	74.0	90.0	555.0
CTT_AUT_anneecst_EHP	280591.0	2.013079e+03	5.439221e+00	1933.0	2010.0	2014.0	2017.0	2022.0
CTT_AUT_BMRC_EHP	280591.0	1.460311e+00	2.735103e+00	0.0	0.0	0.0	2.0	10.0
CTT_AUT_nbannee_BM0_EHP	280591.0	4.314244e+00	4.235839e+00	0.0	0.0	3.0	10.0	10.0
CTT_AUT_MOP_EHP	280591.0	8.421867e-02	2.777160e-01	0.0	0.0	0.0	0.0	1.0
CTT_AUT_OMP_EHP	280591.0	1.725572e-01	3.778647e-01	0.0	0.0	0.0	0.0	1.0
CTT_AUT_PJP_EHP	280591.0	6.808807e-01	4.661362e-01	0.0	0.0	1.0	1.0	1.0
CTT_AUT_joker_EHP	280591.0	3.841499e-01	4.863945e-01	0.0	0.0	0.0	1.0	1.0
CTT_AUT_fractionnement_HP	280591.0	4.178431e-01	4.932049e-01	0.0	0.0	0.0	1.0	1.0
CTT_AUT_modulo_EHP	280591.0	6.817396e-02	2.520446e-01	0.0	0.0	0.0	0.0	1.0
C_codepostal_EHP	280591.0	4.524190e+03	2.637204e+03	1000.0	2150.0	4100.0	6780.0	9992.0
C_profession_EHP	280576.0	7.237629e+00	1.361005e+01	1.0	4.0	5.0	7.0	99.0
C_modulo_EHP	280591.0	9.211628e-02	2.891905e-01	0.0	0.0	0.0	0.0	1.0
C_robinson_EHP	280591.0	1.890296e-01	3.915328e-01	0.0	0.0	0.0	0.0	1.0
C_surveillance_EHP_HH	280591.0	2.104487e-02	1.457517e-01	0.0	0.0	0.0	0.0	3.0
C_taillefam_EHP	280591.0	1.464683e+00	6.395380e-01	1.0	1.0	1.0	2.0	7.0
C_robinson_EHP_HH	280591.0	2.959539e-01	6.087539e-01	0.0	0.0	0.0	0.0	5.0
C_notel_EHP_HH	280591.0	1.335396e-02	1.157746e-01	0.0	0.0	0.0	0.0	2.0
C_nomail_EHP_HH	280591.0	1.806900e-02	1.343999e-01	0.0	0.0	0.0	0.0	3.0
P_closenext_IARD_EHP	280591.0	3.960305e+00	3.312343e+00	0.0	1.0	3.0	6.0	11.0
P_closenext_AUT_EHP	280591.0	5.364481e+00	3.517517e+00	0.0	2.0	5.0	9.0	11.0
P_primeamtsum_EHP	280591.0	8.155513e+02	4.935934e+02	0.0	483.0	701.0	1016.0	11177.0
P_primeamtAUT_EHP	280591.0	6.503164e+02	3.918817e+02	0.0	372.0	552.0	816.0	6034.0
P_nbctt_EHP	280591.0	2.394375e+00	1.524378e+00	1.0	1.0	2.0	3.0	31.0
P_nbcttIARD_EHP	280591.0	2.340717e+00	1.473332e+00	1.0	1.0	2.0	3.0	31.0
P_nbcttAUT_EHP	280591.0	1.102391e+00	3.297734e-01	1.0	1.0	1.0	1.0	6.0
P_lastnbmonthIARD_EHP	280591.0	8.599742e+01	9.958395e+01	0.0	18.0	49.0	113.0	609.0
P_AUT_puismax_EHP	280591.0	8.905026e+01	4.833483e+01	3.0	61.0	77.0	100.0	1405.0

P_AUT_valcatmax_EHP	280591.0	1.530725e+04	1.612467e+04	-28.0	-1.0	14298.0	22944.5	362457.0
P_AUT_puismax_EHP_HH	280591.0	1.117042e+02	6.779497e+01	3.0	66.0	88.0	141.0	1405.0
P_AUT_valcatmax_EHP_HH	280591.0	1.952103e+04	2.039403e+04	-29.0	-1.0	16486.0	28223.0	382903.0
P_VIP_qualite_EHP	80554.0	8.696899e-01	4.704881e-01	0.0	1.0	1.0	1.0	2.0
P_fractionnement_EHP	280591.0	4.312326e-01	4.952494e-01	0.0	0.0	0.0	1.0	1.0
P_fractionnement_EHP_HH	280591.0	4.569320e-01	4.981426e-01	0.0	0.0	0.0	1.0	1.0
P_NBCTTSDS_HH_EHP	280591.0	1.323742e-01	3.777944e-01	0.0	0.0	0.0	0.0	6.0
P_NBCTTSDSPAR_HH_EHP	280591.0	1.980106e-02	1.549863e-01	0.0	0.0	0.0	0.0	5.0
P_NBCTTSDSCOL_HH_EHP	280591.0	1.125731e-01	3.468849e-01	0.0	0.0	0.0	0.0	4.0
P_NBCTTSDSCOL_EHP	280591.0	6.346604e-02	2.500921e-01	0.0	0.0	0.0	0.0	4.0
P_NBCTTSDS_EHP	280591.0	7.565816e-02	2.722075e-01	0.0	0.0	0.0	0.0	5.0
P_NBCTTFFI_EHP	280591.0	2.913137e-02	1.792934e-01	0.0	0.0	0.0	0.0	2.0
CTT_AUT_ageveh_ehp	280591.0	8.920874e+00	5.439221e+00	0.0	5.0	8.0	12.0	89.0

Table 6: Categorical and mixed types features (car insurance, April 2022)

	count	unique	top	freq
CTT_AUT_BMDM_EHP	147980	5	0	136958
CTT_AUT_promo_EHP	29	3	P98L	23
CTT_AUT_DMA_BHP	220986	2	0	206739
CTT_AUT_MDM_BHP	220986	2	0	173574
CTT_AUT_PJ_BHP	220986	2	0	148404
CTT_AUT_ASC_BHP	220986	2	1	188817
CTT_AUT_value_BHP	220986	27155	-1	75104
CTT_AUT_puissance_BHP	220986	277	66	20061
CTT_AUT_BMRC_BHP	220986	18	0	157774
CTT_AUT_MOP_BHP	220986	2	0	202291
CTT_AUT_OMP_BHP	220986	2	0	179109
CTT_AUT_PJP_BHP	220986	2	1	139011
CTT_AUT_BMRC_inc_HP	220986	32	0	154020
C_sexe_EHP	280591	2	M	148018
C_langue_EHP	280591	3	F	171221
C_enfants_EHP	132800	2	Y	105558
C_age_EHP	280567	55	63	7393
C_anciennete_EHP	280585	64	1	18061
C_lifestage_EHP	280565	7	SEN	74310
C_epub_EHP	253444	3	X	198828
C_etatcivil_EHP	168262	6	Marie	92516
C_age_EHP_min_HH	280570	64	30	6675

C_anciennete_EHP_min_HH	280590	62	1	23853
C_age_EHP_max_HH	280570	87	63	8134
C_anciennete_EHP_max_HH	280590	67	1	13433
P_sortie_TypeLastIARD_HP	29470	4	AUT	13471
P_cdtypsorIARD_HP	29470	2	C	28591
P_sortie_AmtLastIARD_HP	29470	1358	88	2610
P_sortie_nbmoisLastIARD_HP	29470	24	0	1782
P_sortie_TypeLastIARD_HP_HH	46798	4	AUT	24263
P_cdtypsorIARD_HP_HH	46798	2	C	45099
P_sortie_AmtLastIARD_HP_HH	46798	1671	88	3200
P_sortie_nbmoisLastIARD_HP_HH	46798	24	0	2964
P_sortie_iftylastIARDisAUT_HP	13471	2	voiture	7483
P_sortie_iftylastIARDisAUT_HP_HH	24263	2	voiture	15160
P_cdtypsorAUT_HP	14266	2	C	14114
P_sortie_AmtLastAUT_HP	14266	1377	109	275
P_sortie_nbmoisLastAUT_HP	14266	24	2	800
P_sortietype_AUT_HP	14266	2	voiture	7985
P_cdtypsorAUT_HP_HH	26263	2	C	25717
P_sortie_AmtLastAUT_HP_HH	26263	1700	109	481
P_sortie_nbmoisLastAUT_HP_HH	26263	24	2	1644
P_sortietype_AUT_HP_HH	26263	2	voiture	16566
P_sortie_AmtLastINC_HP	8180	423	82	859
P_sortie_nbmoisLastINC_HP	8180	24	1	499
P_sortie_AmtLastINC_HP_HH	12532	460	82	1231
P_sortie_nbmoisLastINC_HP_HH	12532	24	1	754
P_cdtypsorVIP_HP	3853	2	C	3630
P_sortie_AmtLastVIP_HP	3853	21	67	2290
P_sortie_nbmoisLastVIP_HP	3853	24	0	277
P_cdtypsorVIP_HP_HH	6084	2	C	5623
P_sortie_AmtLastVIP_HP_HH	6084	24	67	3661
P_sortie_nbmoisLastVIP_HP_HH	6084	24	0	407
P_cdtypsorASS_HP	7137	2	C	6580
P_sortie_AmtLastASS_HP	7137	65	88	2969
P_sortie_nbmoisLastASS_HP	7137	24	18	446
P_cdtypsorASS_HP_HH	10342	2	C	9416
P_sortie_AmtLastASS_HP_HH	10342	81	88	3930
P_sortie_nbmoisLastASS_HP_HH	10342	24	18	654
P_closenext_IARD_EHP_HH	280568	12	0	58670
P_closenext_AUT_EHP_HH	280565	12	0	32430
P_closenext_AUT_o_EHP_HH	42719	12	0	5686

P_closenext_AUT_o_EHP	28048	12	0	3574
P_closenext_inc_EHP_HH	111246	12	3	10524
P_closenext_inc_EHP	80544	12	3	7513
P_closenext_ASS_EHP_HH	134933	12	3	27879
P_closenext_ASS_EHP	103994	12	3	20496
P_closenext_VIP_EHP_HH	112619	12	5	10509
P_closenext_VIP_EHP	80700	12	5	7539
P_primeamtAUT_o_EHP	28048	1640	104	550
P_resmatPFI_EHP	10707	8317	5000	59
P_nbcttVIE_EHP	12189	6	1	9857
P_nbcttAUT_o_EHP	28048	16	1	22724
P_nbcttINC_EHP	80544	13	1	73995
P_nbcttASS_EHP	103994	4	1	103127
P_nbcttVIP_EHP	80700	2	1	80689
P_nbcttDCO_EHP	29877	8	1	26657
P_nbcttPFI_EHP	10707	6	1	8688
P_primeamtsum_EHP_HH	280566	4412	556	304
P_primeamtAUT_EHP_HH	280563	3551	343	446
P_primeamtAUT_o_EHP_HH	42719	2000	104	726
P_resmatPFI_EHP_HH	14929	10126	5000	93
P_nbctt_EHP_HH	280569	26	1	59718
P_nbcttIARD_EHP_HH	280566	26	1	61281
P_nbcttVIE_EHP_HH	16924	12	1	9787
P_nbcttAUT_EHP_HH	280563	7	1	193998
P_nbcttAUT_o_EHP_HH	42719	16	1	32126
P_nbcttINC_EHP_HH	111246	15	1	98214
P_nbcttASS_EHP_HH	134932	4	1	130323
P_nbcttVIP_EHP_HH	112619	4	1	111003
P_nbcttDCO_EHP_HH	42531	8	1	34229
P_nbcttPFI_EHP_HH	14929	12	1	8708
P_firsttype_ever	280585	39	AUT	193392
P_firstnbyear_ever	280585	64	1	18061
P_firsttypeniv3_ever	280585	8	AUT	193392
P_firsttype_ever_HH	280587	39	AUT	183988
P_firstnbyear_ever_HH	280587	67	1	13405
P_firsttypeniv3_ever_HH	280587	8	AUT	183988
P_firstnbmonthAUT_o_pEver	34651	527	22	722
P_lastnbmonthAUT_o_pEver	34651	498	22	869
P_firstnbmonthAUT_pEver	278929	610	10	2848
P_lastnbmonthAUT_pEver	278929	610	10	3509

P_firstnbmonthAUT_o_pEver_HH	52512	561	22	1000
P_lastnbmonthAUT_o_pEver_HH	52512	527	22	1299
P_firstnbmonthAUT_pEver_HH	279983	610	14	2372
P_lastnbmonthAUT_pEver_HH	279983	610	10	3931
P_lasttypeIARD_EHP	280591	5	AUT	175251
P_lasttypeIARD_EHP_HH	280568	5	AUT	167985
P_lastnbmonthIARD_EHP_HH	280568	606	1	6127
P_basketIARDVIE_EHP	280591	2	IAR_	268402
P_basketIARDVIE_EHP_HH	280569	3	IAR_	263645
P_basket_niv3_EHP	280591	32	_ AUT_ _ _ _ _	116773
P_basket_niv3_EHP_HH	280591	36	_ AUT_ _ _ _ _	79963
P_basket_auto_EHP	280591	12	AUT_ _ _ _ _	256399
P_basket_garauto_EHP	280591	16	RCI_PJ _ASC_ _ _	100946
P_basket_auto_EHP_HH	280565	12	AUT_ _ _ _ _	243259
P_basket_garauto_EHP_HH	280565	16	RCI_PJ _ASC_ _ _	94307
P_basket_inc_EHP	80537	3	BAT_CON	60646
P_basket_inc_EHP_HH	111241	3	BAT_CON	86606
P_INC_capbat_EHP	80544	8529	68085	10826
P_INC_capcon_EHP	80544	4858	0	9800
P_INC_loyer_EHP	80544	927	0	61648
P_INC_qualite_EHP	80544	3	PROPRIETAIRE	48388
P_INC_flagvol_EHP	80544	2	1	41955
P_INC_capbat_EHP_HH	111246	11536	68085	13041
P_INC_capcon_EHP_HH	111246	6337	0	12228
P_INC_loyer_HH	111246	1069	0	87036
P_INC_flagvol_EHP_HH	111246	2	1	61080
S_VIP_nbtot_ever	85551	16	0	67215
S_VIP_amount_ever	12089	3657	0	1149
S_VIP_last_ever	18336	373	47	203
S_ASS_nbtot_ever	112645	45	0	66409
S_ASS_amount_ever	45227	4741	115	253
S_ASS_nbtot_HP	14454	12	1	9940
S_ASS_last_ever	46236	358	3	849
S_rcvp_nbtot_ever_HH	119296	18	0	91574
S_rcvp_amount_ever_HH	18398	4341	0	1697
S_rcvp_last_ever_HH	27722	376	47	310
S_ass_nbtot_ever_HH	146078	45	0	80551
S_ass_amount_ever_HH	64129	5277	91	324
S_ass_nbtot_HP_HH	20995	12	1	14293
S_ass_last_ever_HH	65527	363	3	1217

I_MAI_nb_HP	269490	51	39	15790
I_MAI_libellelast_HP	269490	113	Emailing XSell Bike&More	109866
I_MAI_nbmonthlast_HP	269490	24	0	208105
I_MAI_nbass_HP	200472	27	1	39606
I_MAI_nb_HP_HH	274264	156	39	15066
I_MAI_libellelast_HP_HH	274264	109	Emailing XSell Bike&More	133767
I_MAI_nbmontlast_HP_HH	273691	24	0	230703
I_MAI_nbass_HP_HH	231504	49	2	42920
I_DEV_nbtot_HP	160070	55	1	48843
I_DEV_nbaut_HP	160070	52	1	56941
I_DEV_nbrmpl_HP	160070	37	0	103041
I_DEV_nbinc_HP	160070	16	0	139027
I_DEV_nbass_HP	160070	12	0	103103
I_DEV_nbotr_HP	160070	23	0	91728
I_DEV_nbmonthlast_HP	160070	25	1	11058
I_DEV_nbrmpl_6EHP	15838	18	1	11270
I_DEV_nbrmpl_12EHP	30225	24	1	20552
I_DEV_typelast_HP	160063	6	AUT	101443
I_DEV_nbtot_HP_HH	182846	60	1	44038
I_DEV_nbaut_HP_HH	182846	58	1	54572
I_DEV_nbrmpl_HP_HH	182846	40	0	111729
I_DEV_nbinc_HP_HH	182846	20	0	153098
I_DEV_nbass_HP_HH	182846	14	0	113333
I_DEV_nbmonthlast_HP_HH	182846	25	1	14393
I_DEV_typelast_HP_HH	182838	6	AUT	115191
I_DOC_typelast_HP	144	7	PFI	59
I_DOC_typelast_HP_HH	226	7	DCO	90
P_max_amountPFI_HP	19183	10752	0	4438
P_PIncAmtPFI_HP	11167	20	0	9244

.3 Exploratory Data Analysis

.3.1 Variable distributions

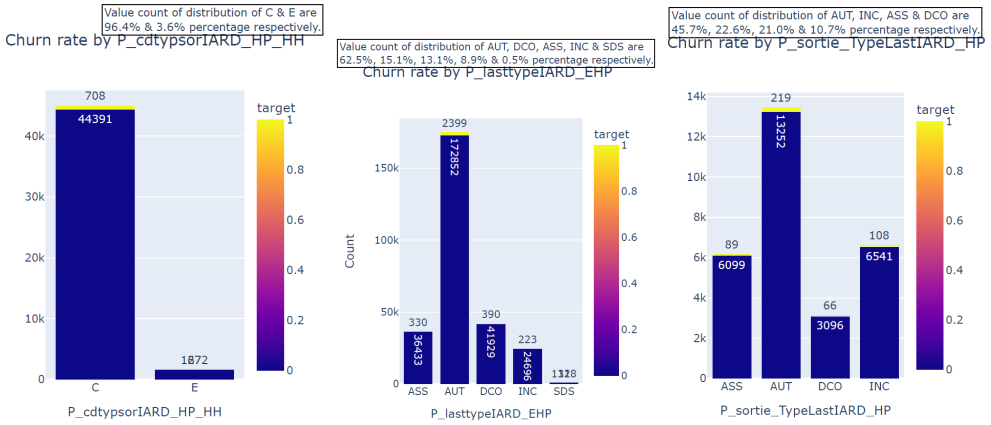


Figure 5: Compilation of distribution of "Product" variables (Car Insurance, April 2022)

.3.2 Correlation structure

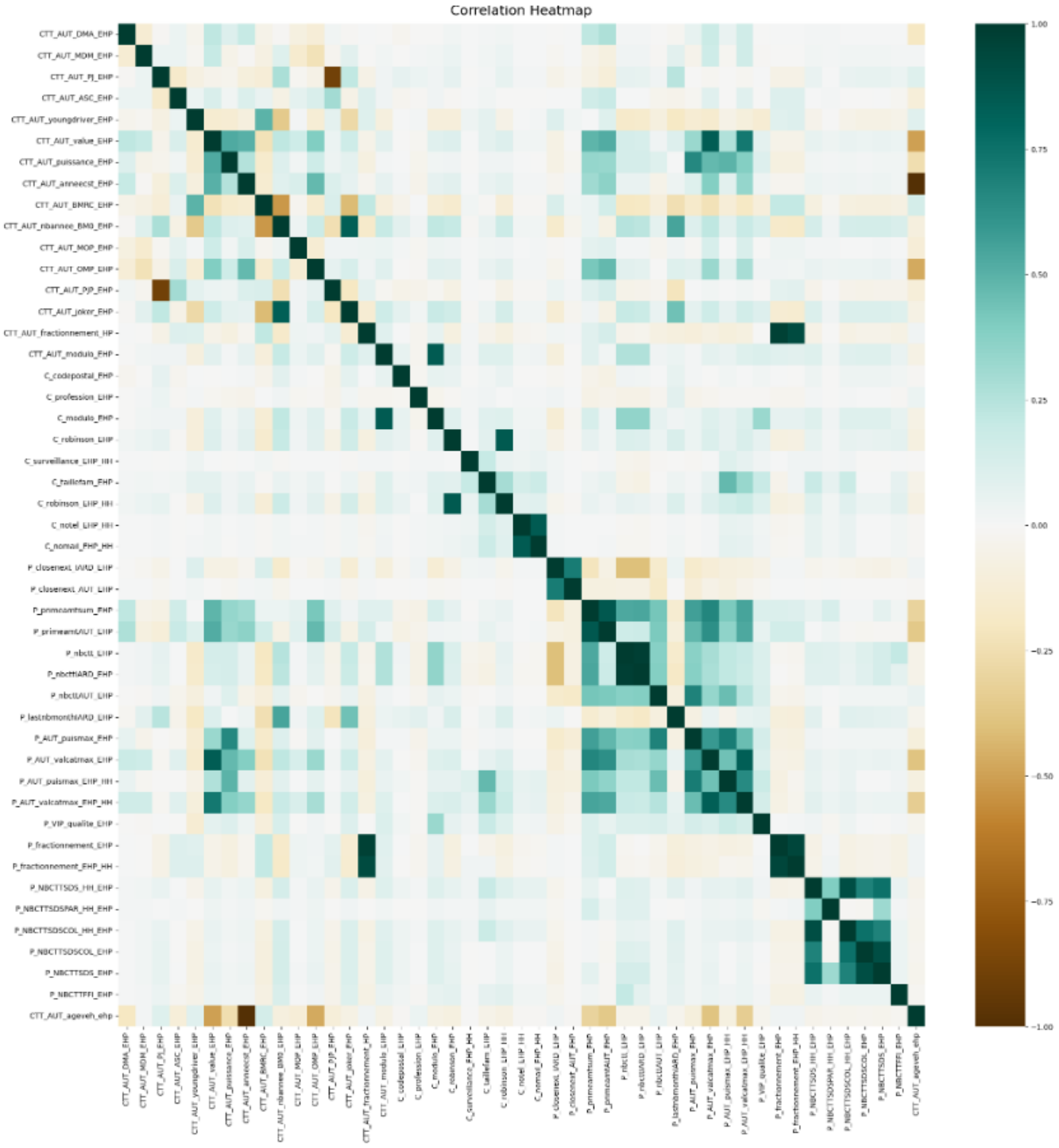


Figure 6: Heatmap of the correlation coefficients of all numerical variables (Car Insurance, April 2022)

Table 7: Pairwise correlation coefficients that range from 0.3 to 0.7 in absolute value

Pair of features	Correlation	P-Value
(P_primeamtAUT_EHP,P_AUT_valcatmax_EHP)	0.668158	0.0
(P_AUT_puismax_EHP_HH,P_AUT_valcatmax_EHP_HH)	0.661465	0.0
(P_primeamtsum_EHP,P_AUT_valcatmax_EHP_HH)	0.640584	0.0
(P_AUT_puismax_EHP,P_AUT_valcatmax_EHP)	0.620621	0.0
(P_primeamtsum_EHP,P_AUT_puismax_EHP)	0.611700	0.0
(CTT_AUT_puissance_EHP,P_AUT_puismax_EHP)	0.597720	0.0

(P_primeamtAUT_EHP,P_AUT_valcatmax_EHP_HH)	0.594949	0.0
(P_primeamtsum_EHP,P_nbcttIARD_EHP)	0.584476	0.0
(P_primeamtsum_EHP,P_nbctt_EHP)	0.564786	0.0
(CTT_AUT_value_EHP,CTT_AUT_puissance_EHP)	0.546523	0.0
(P_nbcttAUT_EHP,P_AUT_puismax_EHP_HH)	0.543969	0.0
(P_AUT_puismax_EHP,P_AUT_valcatmax_EHP_HH)	0.538267	0.0
(P_AUT_valcatmax_EHP,P_AUT_puismax_EHP_HH)	0.528542	0.0
(CTT_AUT_puissance_EHP,P_AUT_puismax_EHP_HH)	0.527328	0.0
(P_primeamtAUT_EHP,P_AUT_puismax_EHP)	0.526888	0.0
(P_primeamtsum_EHP,P_AUT_puismax_EHP_HH)	0.521193	0.0
(CTT_AUT_value_EHP,P_primeamtAUT_EHP)	0.519307	0.0
(CTT_AUT_anneecst_EHP,CTT_AUT_OMP_EHP)	0.492546	0.0
(CTT_AUT_nbannee_BM0_EHP,P_lastnbmonthIARD_EHP)	0.480379	0.0
(P_closenext_IARD_EHP,P_closenext_AUT_EHP)	0.479475	0.0
(CTT_AUT_value_EHP,P_primeamtsum_EHP)	0.479183	0.0
(P_primeamtAUT_EHP,P_nbcttAUT_EHP)	0.473806	0.0
(CTT_AUT_value_EHP,CTT_AUT_anneecst_EHP)	0.469096	0.0
(CTT_AUT_OMP_EHP,P_primeamtAUT_EHP)	0.465997	0.0
(CTT_AUT_puissance_EHP,P_AUT_valcatmax_EHP)	0.458427	0.0
(P_primeamtsum_EHP,P_nbcttAUT_EHP)	0.458249	0.0
(P_nbcttIARD_EHP,P_AUT_puismax_EHP)	0.456508	0.0
(P_nbcttIARD_EHP,P_nbcttAUT_EHP)	0.443074	0.0
(P_nbctt_EHP,P_AUT_puismax_EHP)	0.440667	0.0
(C_taillefam_EHP,P_AUT_puismax_EHP_HH)	0.439747	0.0
(P_primeamtAUT_EHP,P_AUT_puismax_EHP_HH)	0.431440	0.0
(P_nbctt_EHP,P_nbcttAUT_EHP)	0.428126	0.0
(CTT_AUT_puissance_EHP,P_AUT_valcatmax_EHP_HH)	0.423285	0.0
(CTT_AUT_youngdriver_EHP,CTT_AUT_BMRC_EHP)	0.414241	0.0
(P_NBCTTSDS_HH_EHP,P_NBCTTSDSPAR_HH_EHP)	0.398227	0.0
(P_nbcttAUT_EHP,P_AUT_valcatmax_EHP)	0.386603	0.0
(CTT_AUT_OMP_EHP,P_primeamtsum_EHP)	0.383887	0.0
(P_nbcttIARD_EHP,P_AUT_puismax_EHP_HH)	0.376581	0.0
(CTT_AUT_value_EHP,CTT_AUT_OMP_EHP)	0.370401	0.0
(P_nbctt_EHP,P_AUT_puismax_EHP_HH)	0.365406	0.0
(CTT_AUT_joker_EHP,P_lastnbmonthIARD_EHP)	0.364688	0.0
(CTT_AUT_puissance_EHP,P_primeamtsum_EHP)	0.350028	0.0
(P_NBCTTSDSPAR_HH_EHP,P_NBCTTSDS_EHP)	0.348344	0.0
(CTT_AUT_anneecst_EHP,P_primeamtAUT_EHP)	0.336935	0.0
(CTT_AUT_puissance_EHP,P_primeamtAUT_EHP)	0.335235	0.0
(P_nbcttIARD_EHP,P_AUT_valcatmax_EHP)	0.328766	0.0
(C_modulo_EHP,P_VIP_qualite_EHP)	0.325072	0.0
(P_nbctt_EHP,P_AUT_valcatmax_EHP)	0.322925	0.0
(P_nbcttAUT_EHP,P_AUT_valcatmax_EHP_HH)	0.315562	0.0
(C_taillefam_EHP,P_AUT_valcatmax_EHP_HH)	0.314132	0.0
(CTT_AUT_anneecst_EHP,P_AUT_valcatmax_EHP)	0.312316	0.0
(CTT_AUT_youngdriver_EHP,CTT_AUT_nbannee_BM0_EHP)	-0.310318	0.0
(P_AUT_valcatmax_EHP,CTT_AUT_ageveh_ehp)	-0.312316	0.0
(P_primeamtAUT_EHP,CTT_AUT_ageveh_ehp)	-0.336935	0.0
(CTT_AUT_BMRC_EHP,CTT_AUT_joker_EHP)	-0.405170	0.0
(CTT_AUT_value_EHP,CTT_AUT_ageveh_ehp)	-0.469096	0.0
(CTT_AUT_OMP_EHP,CTT_AUT_ageveh_ehp)	-0.492546	0.0
(CTT_AUT_BMRC_EHP,CTT_AUT_nbannee_BM0_EHP)	-0.499151	0.0

.3.3 Chi-Square test of independence

Degrees of freedom (*df*)	Significance level (α) .99	.975	.95	.9	.1	.05	.025	.01
1	-------	0.001	0.004	0.016	2.706	3.841	5.024	6.635
2	0.020	0.051	0.103	0.211	4.605	5.991	7.378	9.210
3	0.115	0.216	0.352	0.584	6.251	7.815	9.348	11.345
4	0.297	0.484	0.711	1.064	7.779	9.488	11.143	13.277
5	0.554	0.831	1.145	1.610	9.236	11.070	12.833	15.086
6	0.872	1.237	1.635	2.204	10.645	12.592	14.449	16.812
7	1.239	1.690	2.167	2.833	12.017	14.067	16.013	18.475
8	1.646	2.180	2.733	3.490	13.362	15.507	17.535	20.090
9	2.088	2.700	3.325	4.168	14.684	16.919	19.023	21.666
10	2.558	3.247	3.940	4.865	15.987	18.307	20.483	23.209
11	3.053	3.816	4.575	5.578	17.275	19.675	21.920	24.725
12	3.571	4.404	5.226	6.304	18.549	21.026	23.337	26.217
13	4.107	5.009	5.892	7.042	19.812	22.362	24.736	27.688
14	4.660	5.629	6.571	7.790	21.064	23.685	26.119	29.141
15	5.229	6.262	7.261	8.547	22.307	24.996	27.488	30.578
16	5.812	6.908	7.962	9.312	23.542	26.296	28.845	32.000
17	6.408	7.564	8.672	10.085	24.769	27.587	30.191	33.409
18	7.015	8.231	9.390	10.865	25.989	28.869	31.526	34.805
19	7.633	8.907	10.117	11.651	27.204	30.144	32.852	36.191
20	8.260	9.591	10.851	12.443	28.412	31.410	34.170	37.566
21	8.897	10.283	11.591	13.240	29.615	32.671	35.479	38.932
22	9.542	10.982	12.338	14.041	30.813	33.924	36.781	40.289
23	10.196	11.689	13.091	14.848	32.007	35.172	38.076	41.638
24	10.856	12.401	13.848	15.659	33.196	36.415	39.364	42.980
25	11.524	13.120	14.611	16.473	34.382	37.652	40.646	44.314
26	12.198	13.844	15.379	17.292	35.563	38.885	41.923	45.642
27	12.879	14.573	16.151	18.114	36.741	40.113	43.195	46.963
28	13.565	15.308	16.928	18.939	37.916	41.337	44.461	48.278
29	14.256	16.047	17.708	19.768	39.087	42.557	45.722	49.588
30	14.953	16.791	18.493	20.599	40.256	43.773	46.979	50.892
40	22.164	24.433	26.509	29.051	51.805	55.758	59.342	63.691
50	29.707	32.357	34.764	37.689	63.167	67.505	71.420	76.154
60	37.485	40.482	43.188	46.459	74.397	79.082	83.298	88.379
70	45.442	48.758	51.739	55.329	85.527	90.531	95.023	100.425
80	53.540	57.153	60.391	64.278	96.578	101.879	106.629	112.329
100	61.754	65.647	69.126	73.291	107.565	113.145	118.136	124.116
1000	70.065	74.222	77.929	82.358	118.498	124.342	129.561	135.807

Figure 7: Chi-Square distribution table (green column represent a 95% level of confidence).

Table 8: Results of the chi-square tests (Fire Insurance (tenant), April 2022).
Note: Compilation of results.

Pair of features	chi-square	p-value	dof
P_cdtypsorVIP_HP and I_DOC_typelast_HP_HH	0.000000	1.000000e+00	1.0
P_sortie_iftylastIARDisAUT_HP_HH and P_cdtypsor...	0.000000	1.000000e+00	1.0
P_cdtypsorAUT_HP_HH and P_basketIARDVIE_EHP_HH	0.000000	1.000000e+00	1.0
P_sortie_iftylastIARDisAUT_HP_HH and P_INC_flag...	0.000000	1.000000e+00	1.0
P_firsttypeniv3_ever and I_DOC_typelast_HP	13.179424	9.966527e-01	30.0
P_firsttypeniv3_ever and I_DOC_typelast_HP_HH	14.356203	9.929020e-01	30.0
P_nbcttDCO_EHP_HH and I_DOC_typelast_HP_HH	6.679510	9.925621e-01	18.0
CTT_INC_REDUCTION_EHP and P_nbcttAUT_o_EHP_HH	18.083136	9.919268e-01	35.0
P_nbcttASS_EHP_HH and target	0.540667	9.905585e-01	5.0
P_nbcttDCO_EHP_HH and I_DOC_typelast_HP	7.013172	9.900148e-01	18.0
P_firsttypeniv3_ever_HH and I_DOC_typelast_HP	15.855633	9.839319e-01	30.0
CTT_INC_REDUCTION_EHP and P_cdtypsorAUT_HP_HH	0.803012	9.768408e-01	5.0
P_sortie_iftylastIARDisAUT_HP and P_basketIARDV...	0.001102	9.735166e-01	1.0
P_basket_inc_EHP and target	0.070017	9.655970e-01	2.0

P_sortie_iftylastIARDisAUT_HP_HH and P_INC_flag...	0.002745	9.582174e-01	1.0
P_nbcttAUT_EHP_HH and I_DOC_typelast_HP	7.608138	8.149548e-01	12.0
P_cdtypsorASS_HP and P_lasttypeIARD_EHP	1.605067	8.078811e-01	4.0
P_sortie_iftylastIARDisAUT_HP_HH and P_nbcttPFI...	0.981201	8.058007e-01	3.0
C_langue_EHP and I_DOC_typelast_HP	3.033102	8.046808e-01	6.0
C_sexe_EHP and P_basketIARDVIE_EHP_HH	0.066107	7.970916e-01	1.0
CTT_INC_REDUCTION_EHP and P_nbcttAUT_o_EHP	23.443280	7.966068e-01	30.0
P_cdtypsorAUT_HP_HH and P_cdtypsorVIP_HP	0.067651	7.947884e-01	1.0
C_lifestage_EHP and I_DOC_typelast_HP	28.942210	7.919793e-01	36.0
P_firsttypeniv3_ever_HH and I_DOC_typelast_HP_HH	23.577026	7.908148e-01	30.0
P_sortie_iftylastIARDisAUT_HP and P_basketIARDV...	0.078946	7.787311e-01	1.0
C_sexe_EHP and I_DOC_typelast_HP_HH	4.328861	6.322660e-01	6.0
P_cdtypsorASS_HP and P_nbcttINC_EHP	0.942692	6.241617e-01	2.0
P_sortie_iftylastIARDisAUT_HP and target	0.242916	6.221081e-01	1.0
P_sortietype_AUT_HP_HH and P_nbcttPFI_EHP_HH	3.548420	6.160741e-01	5.0
P_nbcttAUT_o_EHP and I_DOC_typelast_HP	10.010546	6.150354e-01	12.0
C_lifestage_EHP and I_DOC_typelast_HP_HH	32.941479	6.148408e-01	36.0
C_enfants_EHP and P_cdtypsorASS_HP	0.263965	6.074089e-01	1.0
P_sortie_iftylastIARDisAUT_HP_HH and P_nbcttDCO...	2.721352	6.054835e-01	4.0
P_sortietype_AUT_HP_HH and P_cdtypsorVIP_HP	0.267425	6.050642e-01	1.0
P_cdtypsorVIP_HP_HH and P_nbcttINC_EHP	1.847223	6.047131e-01	3.0
P_cdtypsorAUT_HP_HH and S_ASS_nbtot_HP	3.627057	6.042554e-01	5.0
P_sortie_iftylastIARDisAUT_HP_HH and P_nbcttPFI...	3.642708	6.019125e-01	5.0
P_nbcttAUT_EHP and I_DOC_typelast_HP_HH	4.599233	5.961405e-01	6.0
C_enfants_EHP and P_nbcttAUT_o_EHP_HH	5.534181	5.950643e-01	7.0
C_sexe_EHP and P_nbcttVIE_EHP	2.813069	5.895789e-01	4.0
C_etatcivil_EHP and P_cdtypsorASS_HP_HH	3.730309	5.888601e-01	5.0
P_sortie_TypeLastIARD_HP_HH and P_nbcttVIE_EHP_HH	19.971533	3.344246e-01	18.0
C_langue_EHP and P_INC_qualite_EHP	2.210389	3.311465e-01	2.0
P_cdtypsorASS_HP and P_nbcttVIE_EHP_HH	3.451884	3.270564e-01	3.0
C_etatcivil_EHP and P_sortietype_AUT_HP_HH	5.812496	3.248936e-01	5.0
P_sortietype_AUT_HP_HH and S_ass_nbtot_HP_HH	6.964294	3.241619e-01	6.0
CTT_INC_REDUCTION_EHP and P_sortie_iftylastIARD...	5.886477	3.174224e-01	5.0
C_enfants_EHP and P_cdtypsorVIP_HP	1.014830	3.137485e-01	1.0
P_cdtypsorASS_HP_HH and P_nbcttPFI_EHP	2.326315	3.124980e-01	2.0
P_cdtypsorASS_HP_HH and P_nbcttVIE_EHP	2.361164	3.071000e-01	2.0
P_cdtypsorAUT_HP_HH and S_ass_nbtot_HP_HH	7.157583	3.065196e-01	6.0
P_sortie_TypeLastIARD_HP and P_nbcttDCO_EHP	18.829137	9.273614e-02	12.0
P_cdtypsorASS_HP_HH and P_nbcttINC_EHP	4.841073	8.887391e-02	2.0
P_cdtypsorASS_HP_HH and S_ass_nbtot_HP_HH	9.556351	8.882722e-02	5.0
P_sortie_TypeLastIARD_HP_HH and P_nbcttVIE_EHP	18.989305	8.878687e-02	12.0
P_cdtypsorASS_HP_HH and P_nbcttAUT_o_EHP	8.094399	8.818081e-02	4.0
C_sexe_EHP and P_nbcttDCO_EHP_HH	11.386766	7.713362e-02	6.0
P_cdtypsorIARD_HP and P_basketIARDVIE_EHP	5.478232	1.925473e-02	1.0
P_cdtypsorVIP_HP and P_INC_flagvol_EHP_HH	5.528754	1.870642e-02	1.0
CTT_INC_REDUCTION_EHP and P_nbcttDCO_EHP	28.860075	1.676501e-02	15.0
C_lifestage_EHP and P_sortietype_AUT_HP_HH	15.498758	1.671277e-02	6.0
P_cdtypsorVIP_HP_HH and P_nbcttDCO_EHP	8.260714	1.607714e-02	2.0
P_cdtypsorVIP_HP and target	5.819530	1.584918e-02	1.0
P_sortie_iftylastIARDisAUT_HP and P_nbcttDCO_EHP	12.264644	1.548807e-02	4.0
C_sexe_EHP and P_cdtypsorVIP_HP	5.885833	1.526317e-02	1.0
P_cdtypsorIARD_HP_HH and P_nbcttDCO_EHP	12.318686	1.513229e-02	4.0
P_sortie_TypeLastIARD_HP and P_nbcttDCO_EHP_HH	29.214439	1.509161e-02	15.0
C_epub_EHP and P_sortie_iftylastIARDisAUT_HP	10.503249	1.473887e-02	3.0
C_lifestage_EHP and P_sortietype_AUT_HP	17.497911	7.617433e-03	6.0
P_sortie_iftylastIARDisAUT_HP and P_firsttypeni...	19.406705	7.004368e-03	7.0
C_langue_EHP and P_nbcttDCO_EHP_HH	27.328164	6.928684e-03	12.0
C_sexe_EHP and P_cdtypsorAUT_HP_HH	7.482999	6.228426e-03	1.0
P_nbcttAUT_o_EHP_HH and P_INC_flagvol_EHP	23.108523	5.957135e-03	9.0
C_langue_EHP and I_DEV_typelast_HP	24.722664	5.896724e-03	10.0
C_epub_EHP and P_cdtypsorVIP_HP_HH	16.207737	1.028023e-03	3.0

P_nbcttVIE_EHP_HH and I_DEV_typelast_HP_HH	73.409134	9.981834e-04	40.0
P_sortie_TypeLastIARD_HP and P_INC_qualite_EHP	16.441311	9.205879e-04	3.0
P_sortie_iftylastIARDisAUT_HP_HH and P_firsttyp...	24.807237	8.208657e-04	7.0
P_cdtypsorAUT_HP_HH and P_nbcttAUT_EHP_HH	24.886383	7.948028e-04	7.0
P_cdtypsorAUT_HP_HH and P_nbcttASS_EHP	14.322045	7.762605e-04	2.0
P_cdtypsorIARD_HP and P_nbcttAUT_o_EHP	21.204120	7.411873e-04	5.0
C_sexe_EHP and S_ASS_nbtot_HP	25.082785	7.335791e-04	7.0
C_sexe_EHP and P_INC_flagvol_EHP_HH	108.627379	2.581454e-24	2.0
P_nbcttPFI_EHP_HH and P_basket_inc_EHP_HH	151.986182	2.878293e-25	14.0
P_sortie_iftylastIARDisAUT_HP_HH and P_lasttype...	123.836174	8.092672e-26	4.0
C_sexe_EHP and P_INC_flagvol_EHP	110.564221	7.371812e-26	1.0
C_sexe_EHP and P_sortie_TypeLastIARD_HP_HH	120.593791	5.748456e-26	3.0
P_sortietype_AUT_HP_HH and P_lasttypeIARD_EHP	124.636827	5.457415e-26	4.0
C_sexe_EHP and C_lifestage_EHP	643.671253	8.820646e-136	6.0
P_nbcttASS_EHP_HH and P_INC_flagvol_EHP	649.903168	3.322118e-138	5.0
C_epub_EHP and P_INC_flagvol_EHP	642.384969	6.522118e-139	3.0
C_lifestage_EHP and P_INC_qualite_EHP	661.228557	1.433124e-139	6.0
P_nbcttVIP_EHP and P_firsttypeniv3_ever_HH	3672.677292	0.000000e+00	14.0
P_nbcttDCO_EHP_HH and S_ass_nbtot_HP_HH	1698.138301	0.000000e+00	42.0
P_cdtypsorIARD_HP_HH and P_cdtypsorASS_HP_HH	2186.278003	0.000000e+00	1.0
P_nbcttVIP_EHP and P_firsttypeniv3_ever	4732.516878	0.000000e+00	14.0
CTT_INC_REDUCTION_EHP and P_basket_inc_EHP_HH	2176.868649	0.000000e+00	5.0
CTT_INC_REDUCTION_EHP and P_basket_inc_EHP	2259.043112	0.000000e+00	5.0
C_enfants_EHP and C_etatcivil_EHP	5485.059811	0.000000e+00	5.0

.3.4 T-SNE

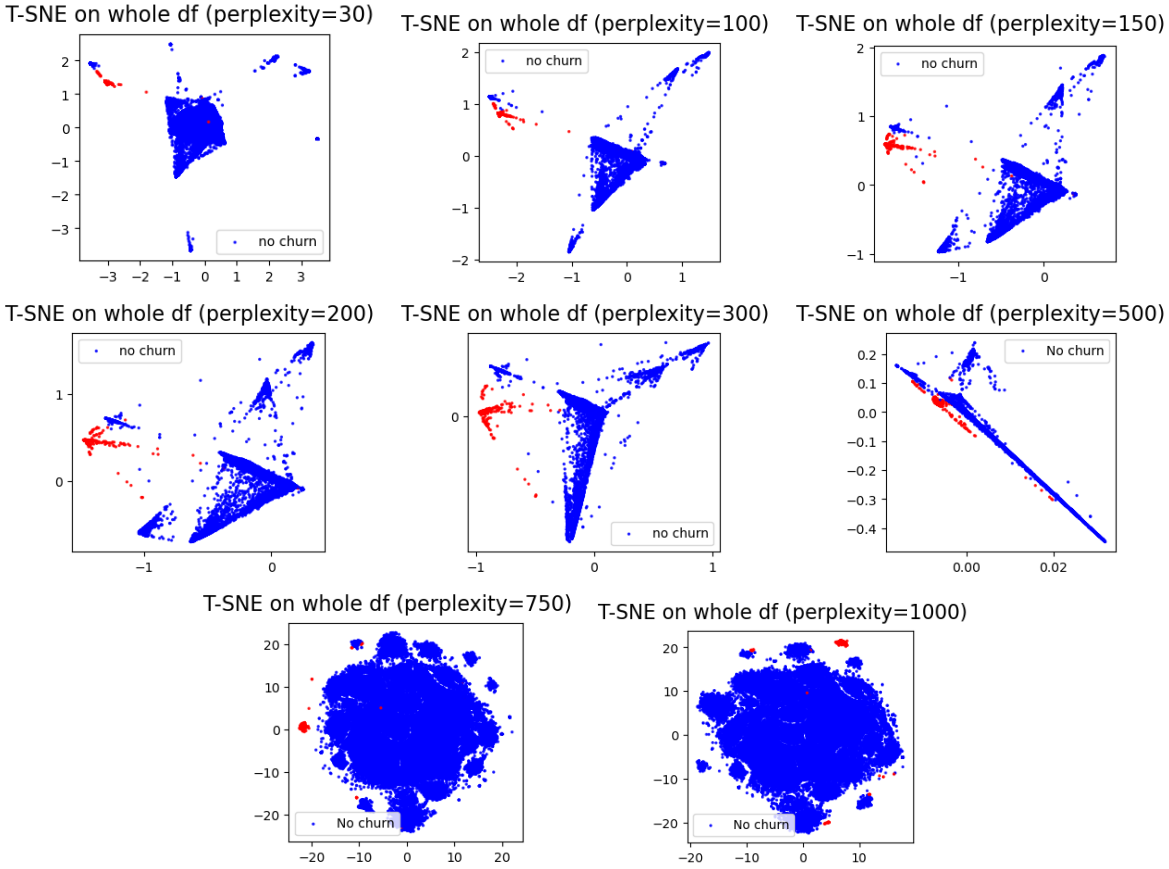


Figure 8: T-SNE on the Car Insurance Single View with multiple perplexity values.

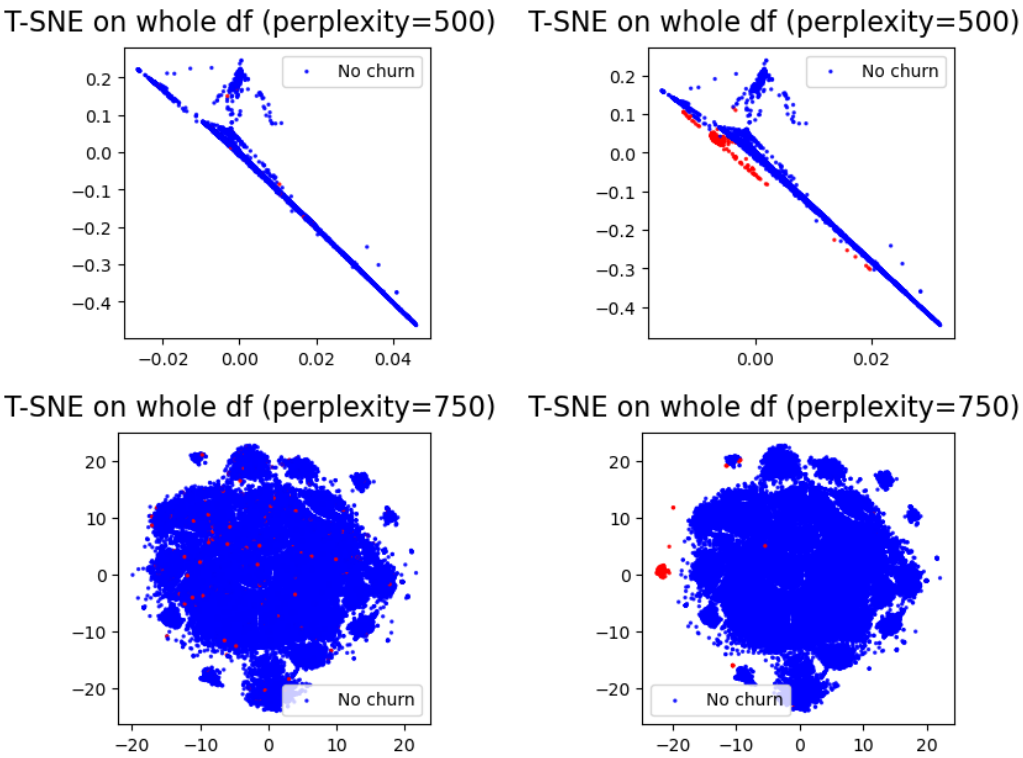
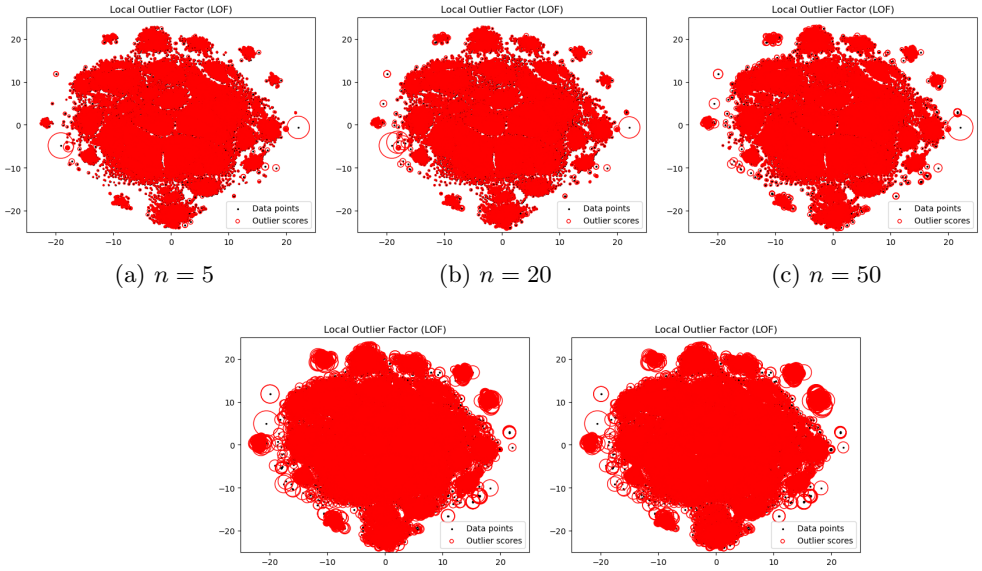


Figure 9: Left: Examples of T-SNE embedding (unsupervised). Right: Same examples of T-SNE embedding (but supervised).

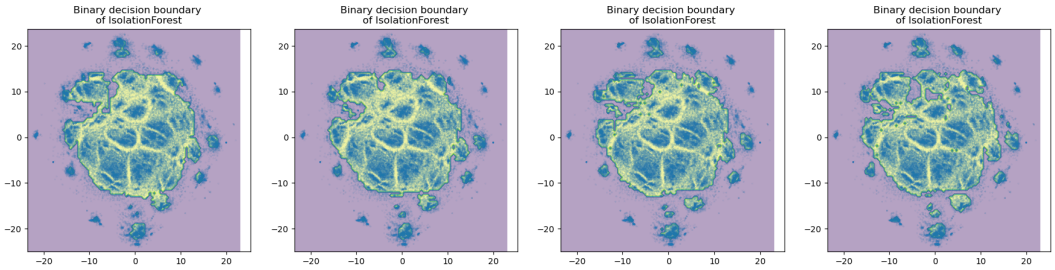
.3.5 Local Outlier Factor and Isolation Forests



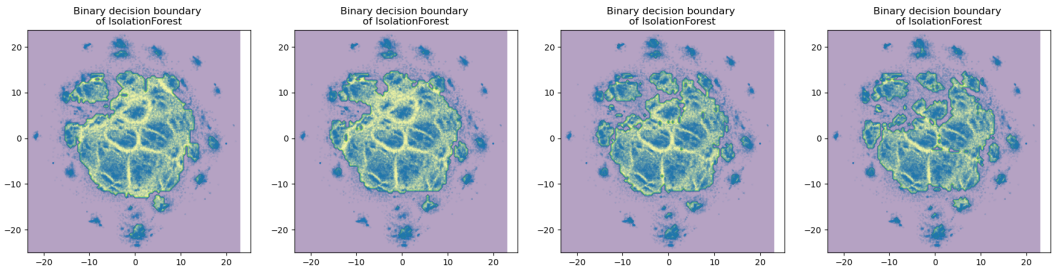
(a) $n = 5$ (b) $n = 20$ (c) $n = 50$

(d) $n = 100$ (e) $n = 200$

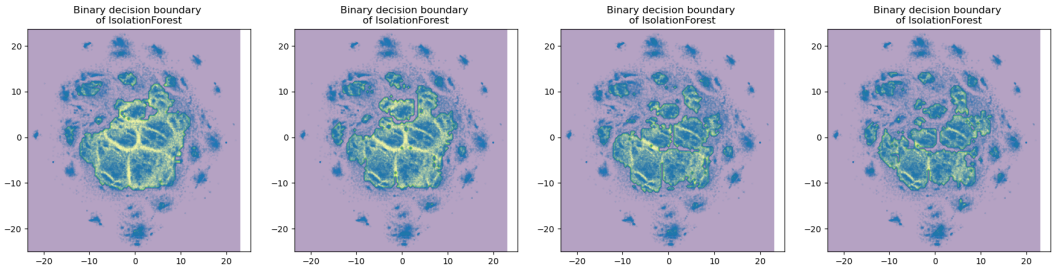
Figure 10: LOF scores with different values of n (number of neighbors) (T-SNE embedding of Car Insurance April 2022).



(a) $n = 10000, c = 0.1$ (b) $n = 25000, c = 0.1$ (c) $n = 50000, c = 0.1$ (d) $n = 80000, c = 0.1$



(e) $n = 10000, c = 0.2$ (f) $n = 25000, c = 0.2$ (g) $n = 50000, c = 0.2$ (h) $n = 80000, c = 0.2$



(i) $n = 10000, c = 0.4$ (j) $n = 25000, c = 0.4$ (k) $n = 50000, c = 0.4$ (l) $n = 80000, c = 0.4$

Figure 11: main caption (T-SNE embedding of Car Insurance April 2022).

.4 Models

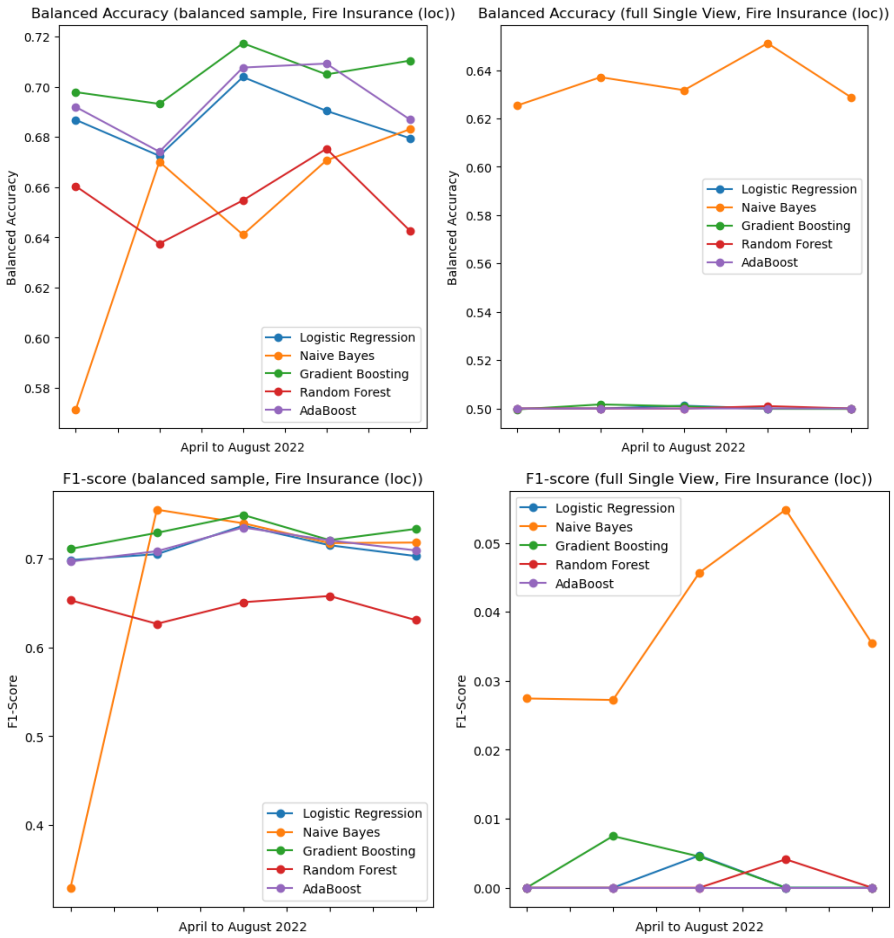


Figure 12: Evolution of the cross-validated balanced accuracy and F1 scores (Fire Insurance (loc), April 2022 to August 2022).

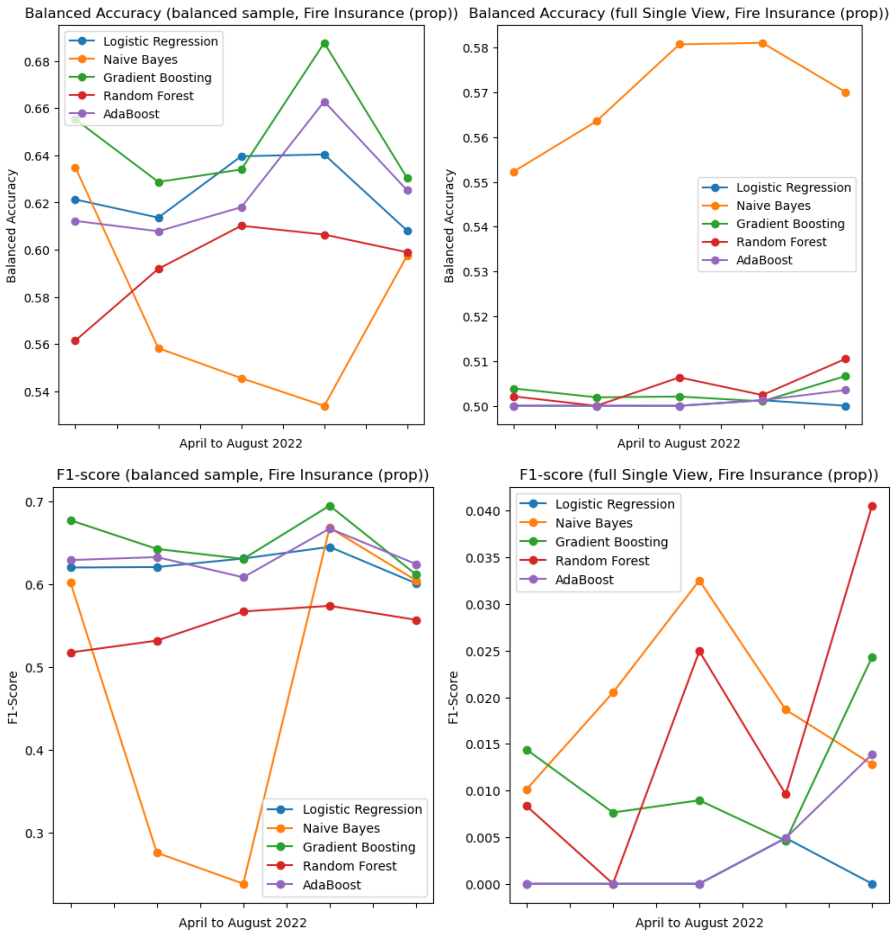


Figure 13: Evolution of the cross-validated balanced accuracy and F1 scores (Fire Insurance (prop), April 2022 to August 2022).

.5 Continuous learning

.5.1 IERF - Confusion matrices

April 2022		Predicted	
		Positive	Negative
Observed	Positive	731	281
	Negative	33210	49956

Table 9: Confusion matrix on the test set (Car Insurance, April 2022)

May 2022		Predicted	
		Positive	Negative
Observed	Positive	889	245
	Negative	33355	49779

Table 10: Confusion matrix on the test set (Car Insurance, May 2022)

June 2022		Predicted	
		Positive	Negative
Observed	Positive	879	390
	Negative	32309	50876

Table 11: Confusion matrix on the test set (Car Insurance, June 2022)

July 2022		Predicted	
		Positive	Negative
Observed	Positive	897	432
	Negative	31970	50872

Table 12: Confusion matrix on the test set (Car Insurance, July 2022)

August 2022		Predicted	
		Positive	Negative
Observed	Positive	752	374
	Negative	31787	51058

Table 13: Confusion matrix on the test set (Car Insurance, August 2022)

.5.2 Weighted XGBoost - Confusion matrices

April 2022		Predicted	
		Positive	Negative
Observed	Positive	984	28
	Negative	69640	13526

Table 14: Confusion matrix on the test set (Car Insurance, April 2022)

May 2022		Predicted	
		Positive	Negative
Observed	Positive	1110	24
	Negative	72285	10849

Table 15: Confusion matrix on the test set (Car Insurance, May 2022)

June 2022		Predicted	
		Positive	Negative
Observed	Positive	1240	29
	Negative	71671	11514

Table 16: Confusion matrix on the test set (Car Insurance, June 2022)

July 2022		Predicted	
		Positive	Negative
Observed	Positive	1309	20
	Negative	72722	10120

Table 17: Confusion matrix on the test set (Car Insurance, July 2022)

August 2022		Predicted	
		Positive	Negative
Observed	Positive	1107	19
	Negative	71685	11160

Table 18: Confusion matrix on the test set (Car Insurance, August 2022)

.5.3 XGBoost with combined sampling

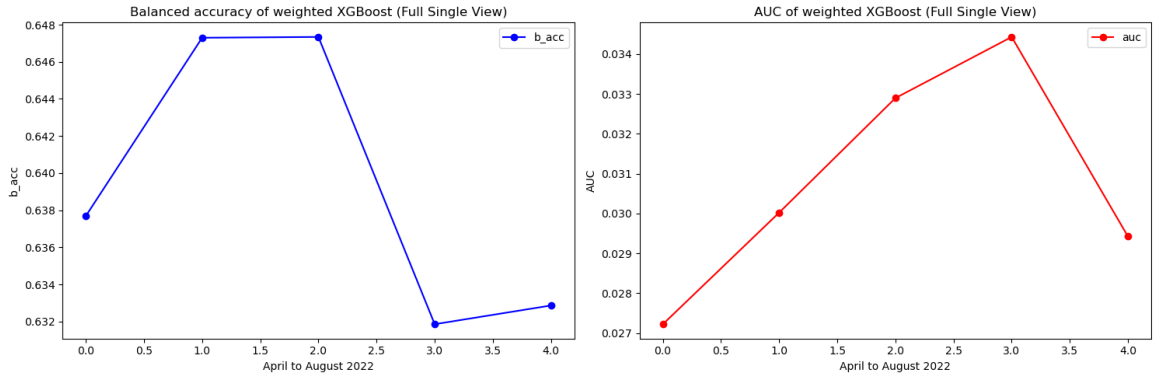


Figure 14: Evolution of the scoring metrics of the test sets evaluated by a XGBoost model with combined sampling (Car Insurance)

.5.4 Comparison of performances

Car Insurance, August 2022	b_acc	f1-score
IERF (trained on July 2022)	0.7304	0.0608
IERF (trained on April 2022)	0.6421	0.0447

July-August 2022 (1 month)		Predicted	
		Positive	Negative
Observed	Positive	3157	732
	Negative	96876	179137

Table 19: Confusion matrix on the test set (Car Insurance, August 2022, 1 month interval between train and test set)

April-August 2022 (3 months)		Predicted	
		Positive	Negative
Observed	Positive	752	374
	Negative	31787	51058

Table 20: Confusion matrix on the test set (Car Insurance, August 2022, 3 months interval between train and test set)