Project Work Fater - SDA

Roberto Formicola, Daniele Petrillo

November 2024

Introduction

This project is inspired by the 2022/2023 Fater Challenge on developing a churn model for its Pampers' customers and finding the optimal strategy to detect and minimize churn, thus maximizing customer retention.

Our approach is mainly data-driven with a strong emphasis on the exploratory data analysis (given the sheer amount and heterogeneity of data at our disposal) by also integrating external information, such as demographic data, competition rules, and so forth.

The statistical data analysis will strictly follow the SLIM approach and, more broadly, the CRISP-DM one.

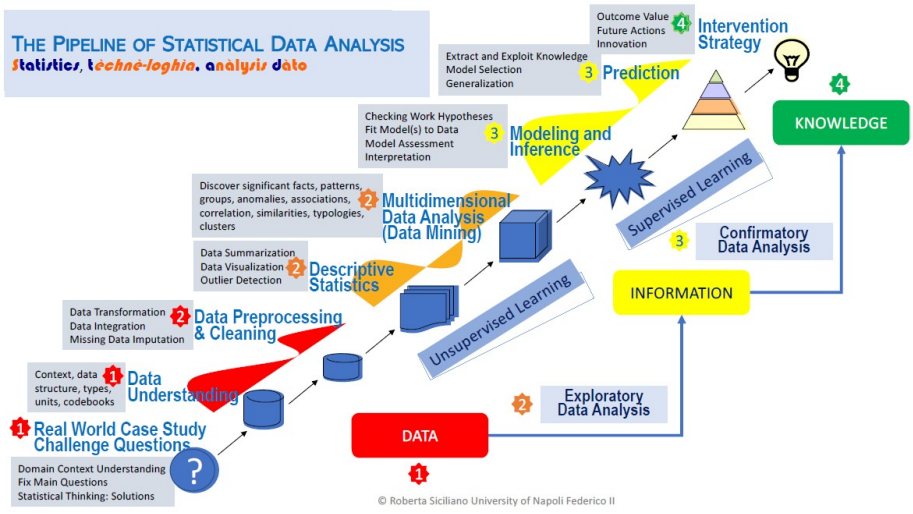


Figure 1: Our pipeline of reference

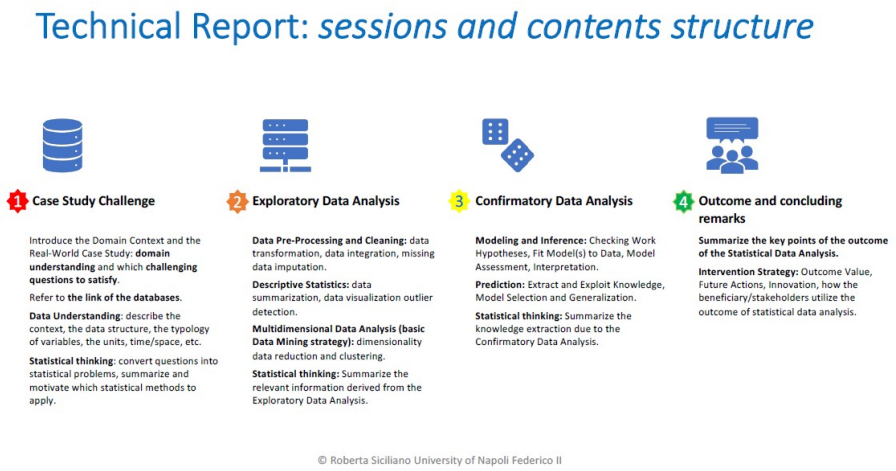


Figure 2: The same pipeline, in details

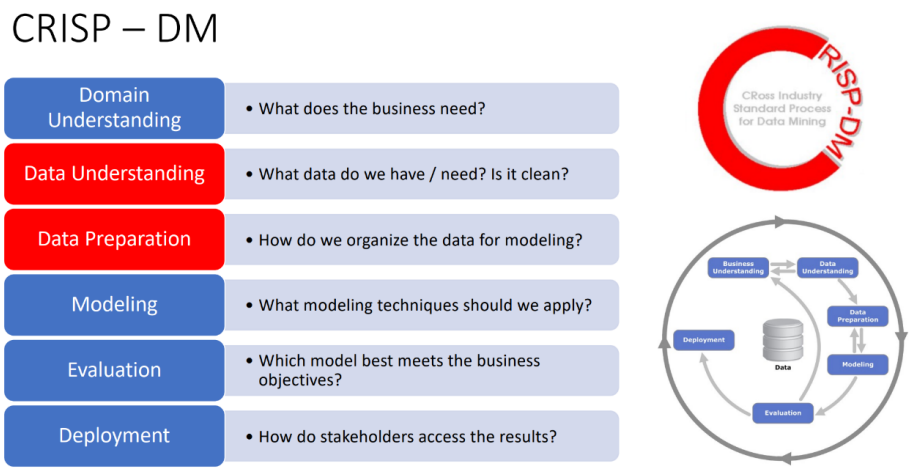


Figure 3: The CRoss Industry Standard Process for Data Mining: an overview

Contents

1 Case Study Challenge

1.1 Customer Churn Prediction: What Is It?

Churn prediction refers to a family of techniques to identify customers who are likely to stop using a product or service after a specified period. The primary objective is to develop predictive models that can accurately predict which customers are at risk of churning. The overall goal is to give stakeholders and management these insights to implement proactive and targeted marketing strategies. These strategies are designed to address the needs of different customer segments, showing different propensities to churn, with the goal of retaining them before they actually leave.

Churn prediction and retention of a firm strongly depend on the market structure, whether or not customers are contended between more firms (customer saturation). In this case, the prediction of churn may save the company a lot of resources [1].

1.2 Challenge Explanation

The *Pampers Collecting Anti Churn Model* project organized by FATER for the years 2022/2023 is an academic challenge in which teams of students competed with the aim of developing strategies to detect and/or prevent a case of customer churn.

In particular, students were given sampled data about user registry, registration and utilization of "Pampers Progressi" mobile and web application, along with prizes list, products description etc.

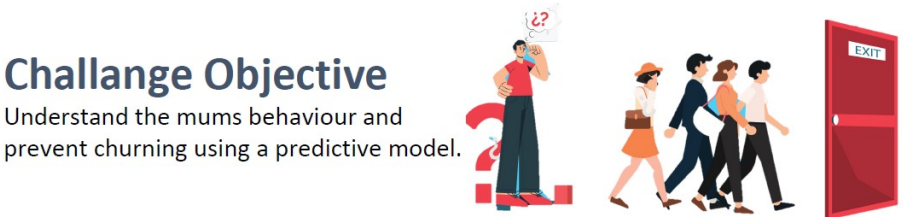


Figure 4: An image from the challenge presentation

The Fater Churn Challenge is more complex than normal subscription churn prediction for multiple reasons:

- No subscription is involved at all. The target variables on which we can train the churn models are not available.
- The data-set at our disposal only refers to a quite small subset of all the Pampers' customers. We do not actually know the true number of Pampers customers.

- We lack control over the customer population and their purchasing decisions, making it difficult to implement and measure the effectiveness of retention strategies or conduct controlled experiments, such as simple A/B testing.

- Eventually, all users will unavoidably churn because the products are not needed anymore (e.g., on average two-year-old children do not need diapers anymore). So, it may be important to discern which customer ends up purchasing diapers due to "children independence" versus those who churn prematurely due to dissatisfaction, competition, or other preventable factors.

1.3 Data Explanation

The provided data comes in *.csv* format in 6 different data-frames, as shown below:

ENTITY	ATTRIBUTES						
Premi Mamme	id_player	punti premio	nome premio	data richiesta premio	format premio	delivery Mode	tipo premio
Anagrafica	id_player	Provenienza (COMPOSTO)	Data Nascita Bambino	Età bambino oggi	Età bambino alla registrazione		
Accessi_app	id_player	source	updated_at (COMPOSTO)				
Prodotti Caricati	id_player	Mission Detail	EAN	Punti Guadagnati	created at		
Missioni Players	id_player	Mission Detail	missionSchedaID	Type (COMPOSTO)	points	created_at	
Conversione EAN Prodotto	EAN	Referenza_DES	Segmento_DES	Occuso_DES	Tier (3 tier diversi)		

Figure 5: Relational schema

The focus of our data is on the behavior of each single registered user, mainly by tracking every specific action performed during app-usage. For this reason, the primary and foreign key is *id_player* for the majority of data-frames.

- **Premi mamme**: it shows the prizes requested by the parents. As we know from legal documents, maximum 2 prizes can be requested for promotional periods.
  Click Here to access to the table R Notebook for more information.

- **Anagrafica**: it gives the residence and children (expected) date of birth. Date of birth can be true - if the registration happened after birth-date - or guessed, if the registration happened before the birth date. This information is not actually very robust, since the date of birth could be completely made up.
  Click Here to access to the table R Notebook for more information.

- **Accessi app**: the table with the least amount of variables/columns. For log-ins, its precision is up to seconds.
  Click Here to access the table R Notebook for more information.

- **Prodotti Caricati**: it tells the type of product loaded and how many points are gained from each upload. Essentially, it is one of our main tables of interest for the whole project.
  Click Here to access the table R Notebook for more information.

- **Missioni Players**: other than loading products, an user can obtain points from various missions.
  Click Here to access the table R Notebook for more information.

- **Conversione EAN Prodotto**: This table can be considered as an auxiliary one because it is the only one without *id_player* as primary key. From this it is possible to understand which products are loaded by merging on EAN (European Article Number), a standard for bar-codes that allows for the unique identification of products in retail and logistics.
  Click Here to access the table R Notebook for more information.

Other data are added from external sources, such as the median child weight associated with each diaper.

1.4 Potential Strategies

The standard approach taken from the churn prediction literature relies on the selection of meaningful variables and their exploitation through statistical and deep learning models to predict the target variable. This modeling is strongly interwoven with marketing and behavioral customer knowledge.

In our case, the target variable (hypothetically a binary one, churn/non-churn) needs to be created first, so an efficient method for distinguishing churn shall be developed; next, the prediction could be carried out with different strategies:

- Logistic Regression;
- Decision Tree;
- Random Forests;
- Neural Networks;
- Time series prediction models;
- Survival Analysis;
- etc.

In addition to the use of the previous supervised models, it is possible to make use of **clustering** methods, in order to distinguish between groups of customers based on their describing variables (*classical* clustering) as well as their behavior over time (*time series* clustering).

2 Exploratory Data Analysis

After understanding the case study and briefly the provided data, we perform the exploratory data analysis. In modern-day data-driven knowledge discovery, it is widely considered as the essential step where we try to extract as much information and hints as possible from the data, without actually deploying models or limiting us from using very basic ones. This extraction comes in the form of descriptive statistics, visualizations (i.e., plots and graphs), and basic statistical tests. Strictly related to EDA is *data cleaning* and *cleansing*, steps where the data is tidied/corrected/standardized for correct and robust model deployment. For example, conversion error, incorrect date setting, presence of NAs and redundant variables must be considered and dealt with in order to perform a correct analysis. EDA is a shallow and very broad process, which will eventually help to formulate the correct hypotheses and guide us to the choice of more advanced analytical techniques for Confirmatory Data Analysis.

Given the heterogeneity and vagueness of the data tables, an extensive data pre-processing is needed to clean and organize the (often fragmented) information shared from FATER. Additionally, some useful information has been extrapolated from the different tables and saved as new data-frames; this transformed data significantly helped with gaining useful insights hidden before and were the foundation of the subsequent exploratory data analysis.

The most important steps and findings from the exploratory data analysis are highlighted in this section. Detailed information about the exploratory data analysis with access to interactive plots are available, see Section 1.3 for script usage and replication.

2.1 Accesses

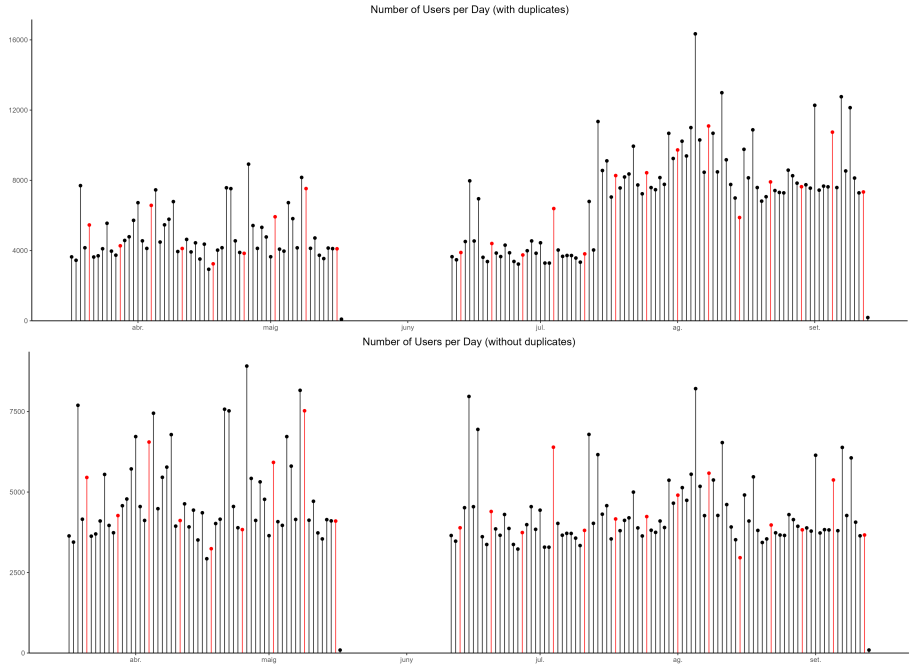
The table `Accesses` presents 975,948 rows and three variables, being `id_player`, `source`, and `updated_at`.

The number of users who accessed at least once from April to September is 31995, almost 3/4 of the total number of users from the table `anagraphics`.

It is the table with the least number of variables. The table does not have missing values (NAs), but it has many duplicate rows, over 1/4 of all observations.

Before performing a further inspection on the nature of these row duplicates and the table time series realizations for each user, we give a look at the variable `source`: this variable is dichotomous and simply tells if the access happened through the app or the website, the second one being at a mere 0.5% of the time. The variable is thus removed.

The variable `updated_at` is in format `yyyy-mm-dd hh:mm:ss`, even though as character data-type. After being converted into proper date format (thus losing the intra-day access information), the aggregate accesses per day have been plotted, either with and without duplicates:



The red observations are Mondays. This has been done simply to assess eventual weekly cycles.

From these plots, it is possible to notice three main phenomena.

- The time series without duplicates appears almost stationary, and it does not show a trend. Some days exhibit a notably higher number of accesses, possibly indicating specific promotions or deadlines.
- A significant gap is observed between May and June. The cause of this gap is currently unknown and may warrant further investigation.
- All duplicates in the data occur from mid-July until the end of the observation period. This could be attributed to either:
  - The implementation of a new technical check for users
  - A technical bug in the system

Finer granularity can be achieved by plotting the accesses by hours or even minutes, especially to assess potential cyclical patterns in access times throughout the day. This has not been done due to computational constraints and doubtful usefulness of such information.

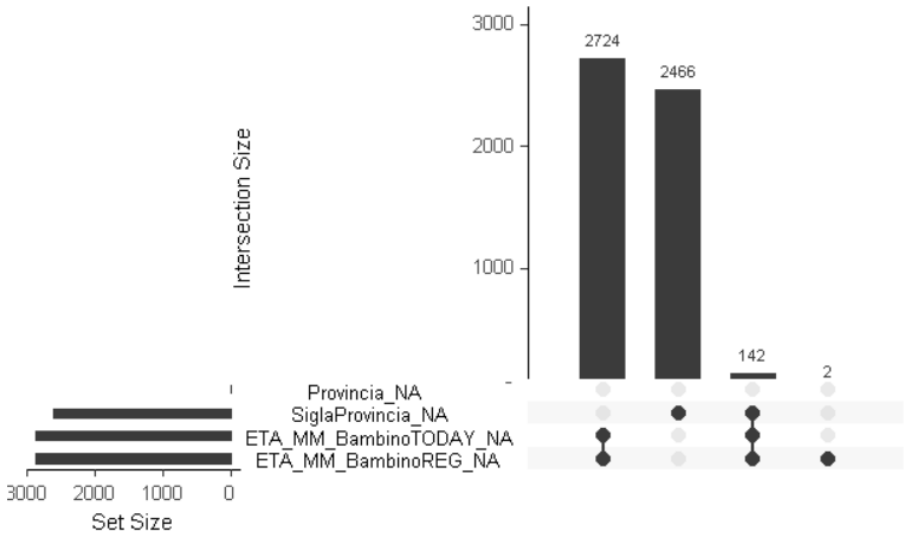
To sum up, `Accesses` does not seem very useful, also after cross-checking with the `products` data-frame which showed problematic time discrepancies

between the two. The short time-span, the presence of duplicates and the lack of data around June could cause issues when using this dataset; hence it is not used further for our analysis.

2.2 Anagraphics

The `anagraphics` data-frame presents `45971` observations and `9` columns. The `id_player` values present in this data-frame represents the entire population of users that are also present in the other data-frames, such as `products_loaded` and `missions`.

There are no duplicates. This is expected given the "legal" importance of the data. The presence of missing values is checked:



The plot above shows the number of missing values and especially their dependence/intersection. The vast majority of missing values come together, especially between the ones about the age, i.e. `ETA_MM_BambinoTODAY_NA` and `ETA_MM_BambinoREG_NA`. This is logical, given that today's age is automatically calculated from the registered age.

For now, the missing values are not imputed, mainly due to their low incidence.

Around 0.5% of the users do not seem to have inserted the child's age. A similar number of NA occurs in the `SiglaProvincia_NA` variable.

Moving to the variable without missing values, `DtaRegUserData` represents the date when the user has registered on the Fater's application. It is a variable obtained automatically.

`DtaPresuntoParto`, along with `ETA_MM_Bambino_TODAY` and `ETA_MM_BambinoREG`, gives us information about the newborn's age. The first one is an educated guess from the users in case the registration occurs before birth, while the second one

tells us the newborn age at data extrapolation moment from the database, happened around September 2022.

`Provincia`, `SiglaProvincia`, `Comune`, `Regione` are geographical variables. The first two are obviously perfectly correlated with each other, so `SiglaProvincia` is dropped.

`Regione` variable has 21 unique elements. The last value is a void character, so we give a further look to it by retrieving the subset:

```
voidChar <- anagraphics[anagraphics$Regione == "",]
voidChar
```

	id_player <int>	DtaRegUserData <chr>	DtaPresuntoParto <chr>	ETA_MM_BambinoTODAY <dbl>	ETA_MM_BambinoREG <dbl>
1049	1012816	2020-10-25	2021-06-28	15	-9
1111	82576	2019-06-24		NA	NA
1246	170108	2020-02-10	2022-09-09	0	-31
1264	453502	2020-05-15	2020-08-19	25	-4
1370	1222364	2020-11-20	2021-07-22	14	-9
1374	118492	2019-05-24		NA	NA
1413	1884096	2022-03-31	2022-11-01	-2	-8
1454	1567180	2021-11-06	2020-07-08	26	15
1601	1003254	2020-10-11	2021-06-19	15	-9
1765	493354	2018-10-19	2021-03-23	18	-30

1-10 of 294 rows | 1-6 of 8 columns Previous 1 2 3 4 5 6 ... 30 Next

Hide

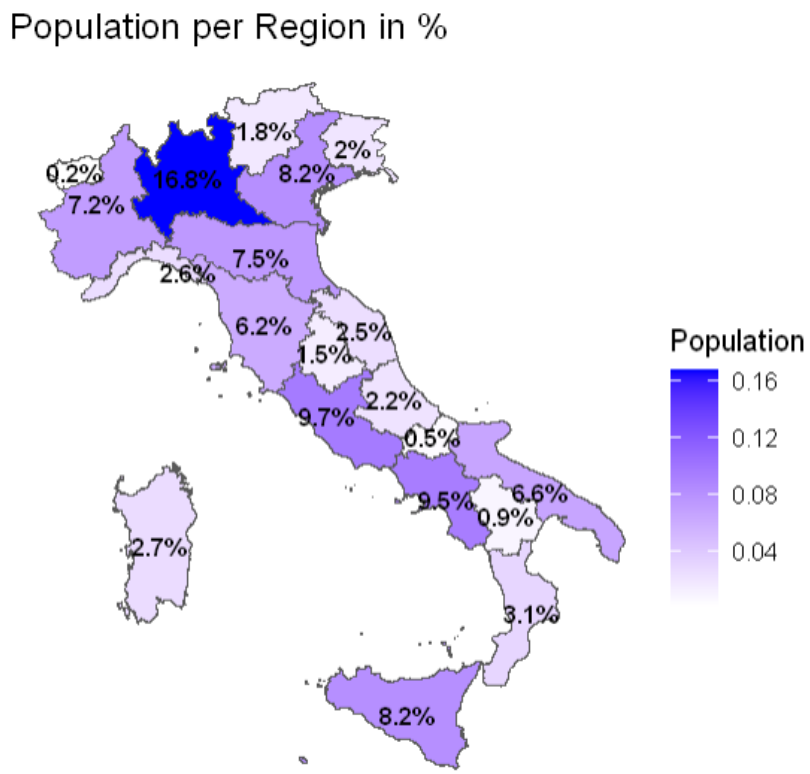
```
nrow(voidChar)
```

```
[1] 294
```

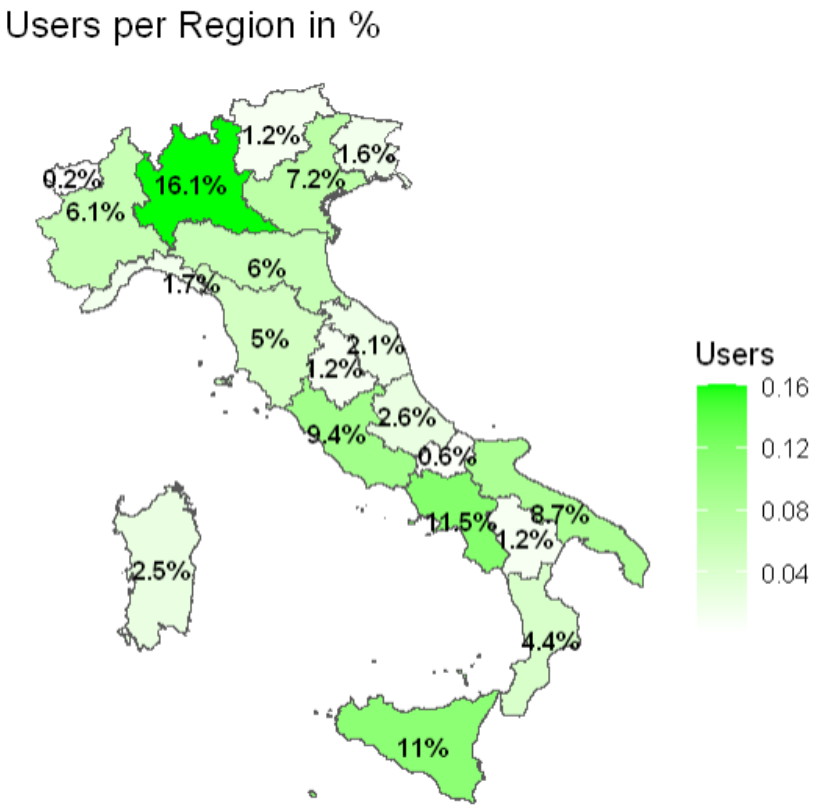
Almost 300 entries do not report a correct location format. This might be due to the user living outside of Italy or maybe just being disinterested in providing correct details, thereby signaling small interest.

The data-frame is corrected: the unusual entries are removed from the main data-frame.

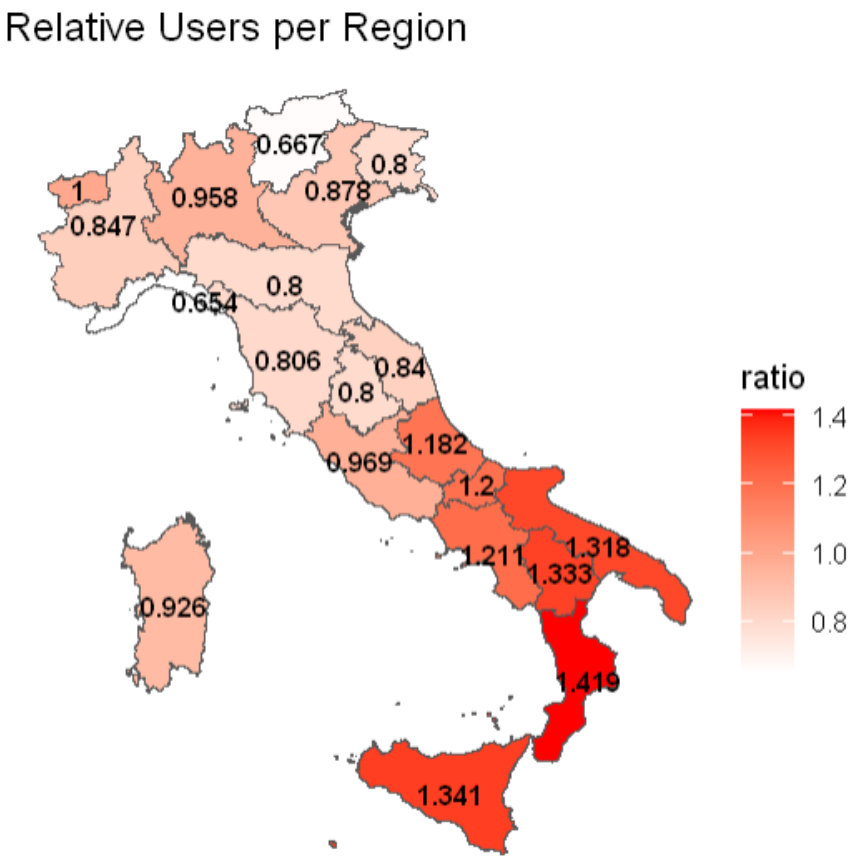
After these adjustments, geographical visualizations have been created:



This first plot is simply the percentage of Italian population per region. Data are retrieved from the 2020 ISTAT census.



The second plot instead shows the percentage of users from `anagraphics` per region.

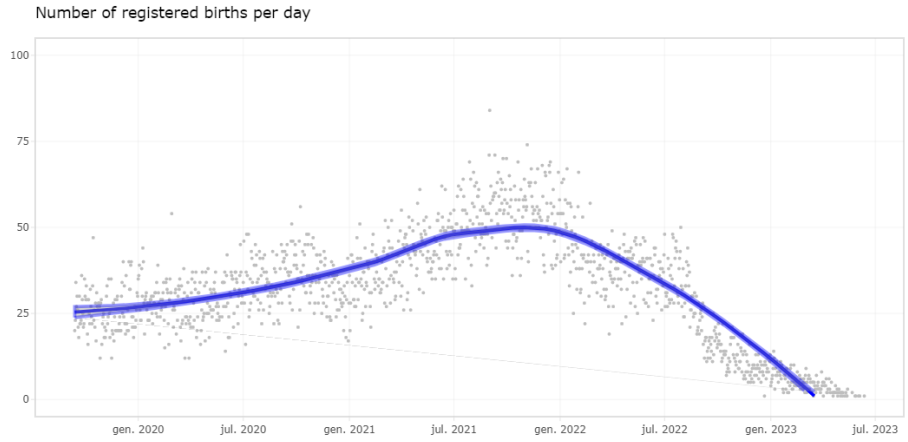


This plot is simply the ratio of the two above. It shows the relative incidence of FATER's users per region. It is interesting to note that the lower GDP per capita regions have a higher incidence of users, signaling that there is more interest in saving through promotional events.

Even though the last plot seems very interesting, it does not look useful for our churn prediction analysis.

It is possible to create a similar visualization for provinces or even "comuni".

Another interesting visualization is the temporal distribution of the `date of birth` variable:

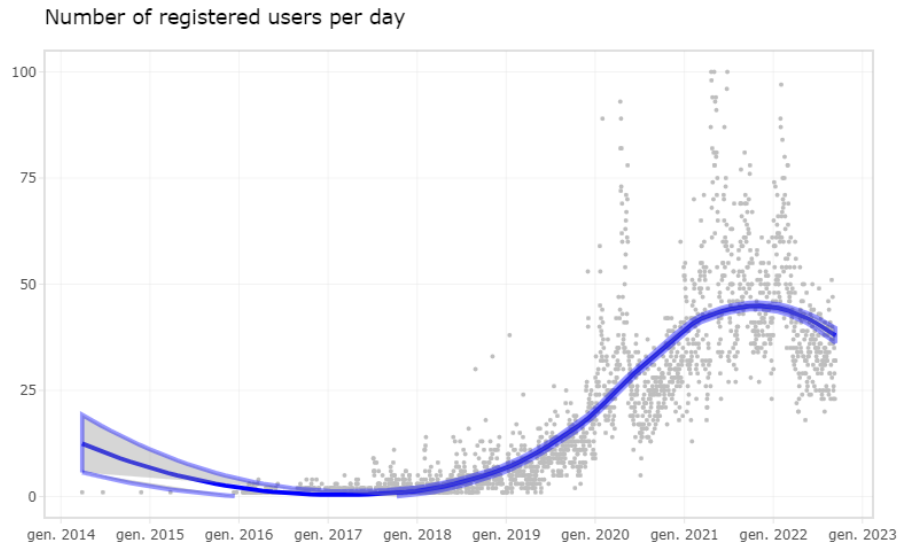


The greatest number of births per day is around the second half of 2021. This looks slightly correlated with the actual population birth patterns in Italy.

The convergence to zero of births could mean that most users register the birth after the actual birth of the child. This is hinted by the fact that the last observation available is in September 2022.

The fitting line is the LOESS (LOcally Estimated Scatterplot Smoothing), which is a non-parametric non-linear method to estimate the trend of the data based on weighted least squares. In this case, it is simply used to visually assess the time series pattern; hence no further analysis of its inner workings is needed.

Another interesting time series is the frequency of registrations per day, again plotted along with a LOESS function:



The last plot has peaks concentrated in few days. This could mean that specific events or marketing campaigns have been launched during those days. Prize redemption deadlines could be another explanation.

2.3 EAN Products

This data-frame can be considered as an auxiliary table: its main importance derives from the fact that it is possible to understand the commodity-related categorization of certain products. It is the only data-frame that does not contain the `id_player` variable.

It has 239 observations and 5 variables. All the values are present and there are no duplicates.

The first variable is `EAN`, which stands for *European Article Number*. It is a standardized way to define generic product types when commercialized for logistical purposes. The EAN code is usually placed under the bar-code of a product.

An EAN code looks like this: "4015400327776".

The first 2 digits of this code tell us the company's country, while the next 5 digits define the company that commercializes the product. Together they form the **GS1 Company Prefix**. The next digits, instead, identify different types of product.

The digits **8001480** refers to Fater SpA.

Only 4 EAN codes out of 239 are from a different country (they start with 40 and 84 instead of 80, which is Italy). The first 3 products are from **P&G** GmbH, based in Germany: this makes sense, because it is the Fater's parent company. The last one is from **Hero Espana**, which commercializes food products. Fater and Hero collaborate to produce and commercialize specific food products.

The meaning of other variables is retrieved from the provided metadata and description from the EAN website, linked above.

The following variables are described from the most generic to the most specific.

The variable `REFERENZA_DES` includes information on the product line (defined in `SEGMENTO_DES`) and also information about the weight and bundle type of the product. For example, as shown below, the *Sluna* product is differentiated in *Maxi*, *Mini* etc. In this case, each "referenza" refers to a specific dimension tailored to children with different weights (roughly from 3 to 22 kilograms).

`REFERENZA_DES` should provide the most detailed characteristics of a product. With further inspection, this does not look enough, given that there are 239 unique EAN codes versus 179 specified products in this categorization, as shown below.

A tibble: 179 × 2

REFERENZA_DES <chr>	n <int>
BD Esapack Junior	3
Sluna Junior VP	3
Sluna Maxi VP	3
Sluna Midi VP	3
Sluna XL VP	3

The variable `SEGMENTO_DES` refers to the product line, for example *Progressi*.

A tibble: 16 × 2

SEGMENTO_DES <chr>	n <int>
Baby dry	72
Progressi	56
Baby Dry Mutandino	35
Soleluna	25
Progressi Mutandino	17

The variable `OCCUSO_DES` refers to the category (the generic product) type, for example *pannolini* (diapers). The majority of products are diapers, and given the second most frequent product are 5% of the total diapers, we might eventually remove them and the other less frequent products ("vario" and "biscotti solubili").

OCCUSO_DES <chr>	n <int>
Pannolini	223
Wipes	13
Vario	2
Biscotti solubili	1

The variable `TIER` identifies the premium products (TIER1), mid range products (TIER2) and low range products (TIER3). There are also products with NO TIER.

TIER <chr>	n <int>
TIER2	111
TIER1	90
TIER3	31
NO TIER	7

There is one EAN for each single row. This means that, even if a certain value of `REFERENZA_DES` appears twice or more, there is still a greater level of detail we cannot get.

The main table connected with this one is the `products_loaded` table. This can be compared with the number of unique EAN codes from the EAN table.

The number of EAN in the *products* table is equal to 282, while the number of EAN in the *EAN_table* is equal to 239.

There are more EAN than the rows from *EAN_table*: this means the nature of certain products loaded cannot be understood in the same fashion.

An outer-join is then performed to see the EAN not present in the *EAN_table*:

```
 [1] "fitmktwipes"           "Fit1punti400"          "WIPESSL"               ""
 [5] "WIPESBF1"              "fitmktimm"             "WIPESSNS1"             "Fit1punti200"
 [9] "fitcopuponsl139"       "Fit2punti300"          "fitmktpoe"             "Fit2punti200"
[13] "FitBirthdayCodes"      "WIPESAP2"              "WIPESBF2"              "Fit1punti300"
[17] "WIPESSNS2"             "fitwipescollecting"    "fitmktstore50"         "fitmktstore100"
[21] "WIPESAP1"              "fitmktgiftbox"         "lovelymind"            "mktValuepackExtra1"
[25] "fittraccpntgen201603"  "fitmkttool100"         "fitmktnvt2"            "fitstickerteli02"
[29] "fitstickerteli04"      "mktValuepackExtra2"    "fitmktfmm"             "BUONNATALE"
[33] "FITMARKSUB02"          "fitstickerteli01"      "fitmkttool150"         "fitbfcodecoupon"
[37] "fitcouponpharmax63"    "fitmktnvt"             "FITMKTGREENG"          "fitstickerteli03"
[41] "FITMARKSUB01"          "fitmktstore200"        "CodiceSalvietta2Punti"
```

As shown above, all the entries not present in the *EAN_table* are not actual EAN codes.

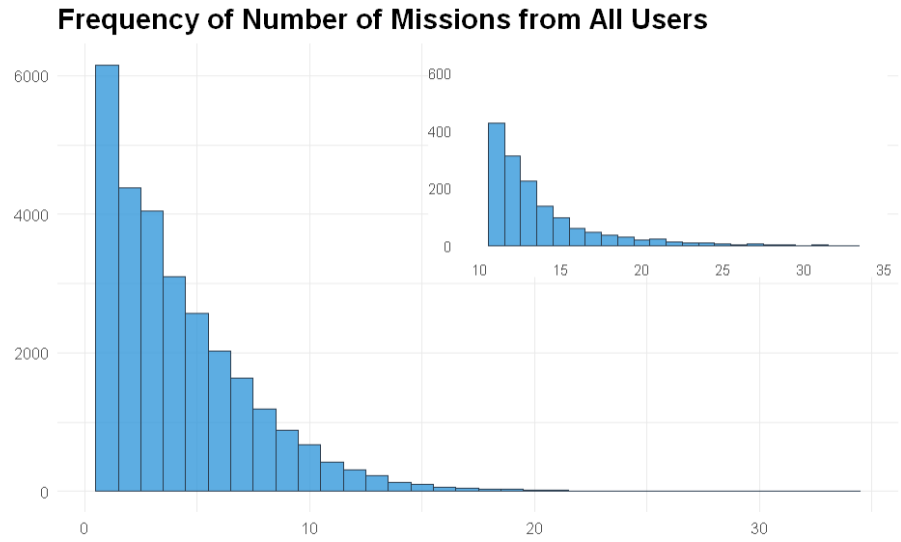
2.4 Missions

In this data-frame it is possible to track the mission performed by each single user from March 2021 to September 2022. The data-frame has 119677 observations and 7 variables. The presence of null values and row duplicates is checked; none was found.

The `type` variable presents only one unique value ("special"), so it can be removed.

Also in this data-frame there is the `id_player` variable. The number of users who have done at least one mission is 28104, a little more than half of the population in `anagraphics`.

A plot is created to show the frequency of missions per player:



On the x-axis is shown the number of missions, while on the y-axis the count of players who did x_i missions. For example, the majority of players have completed a few missions, while around 200 have completed 13 missions, as shown in the inset plot.

The variable `missionDetailId` is made of 10 digits and it uniquely identifies single missions. Its value is sorted chronologically, as shown below.

Description: df [119,677 × 3]

id_player <int>	missionDetailId <dbl>	created_at <chr>
206996	100088014	2021-05-02 14:40:27
921088	100088087	2021-05-02 14:40:27
994006	100088220	2021-05-02 14:40:27
514212	100088535	2021-05-02 14:40:27
27126	100088600	2021-05-02 14:40:28
536060	100088755	2021-05-02 14:40:28
713406	100088765	2021-05-02 14:40:28

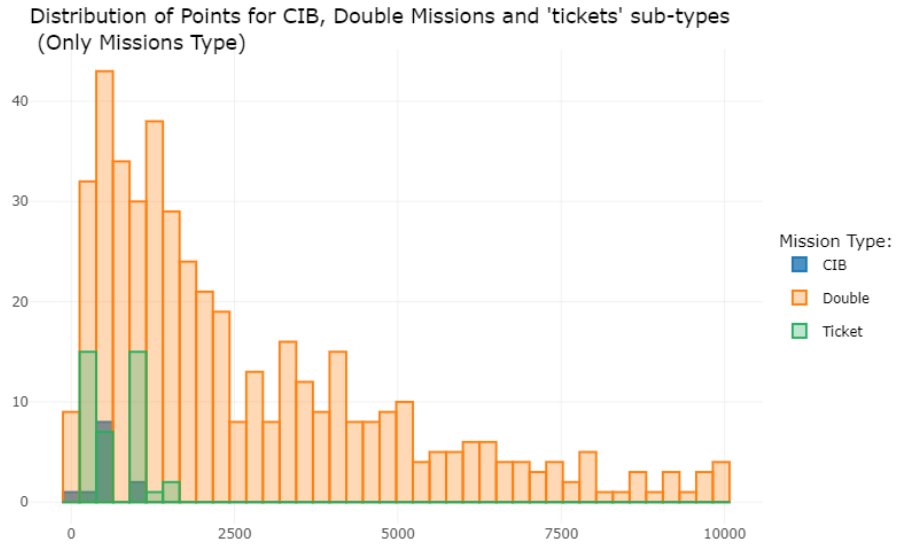
The variable `missionSchemaId`, instead, identifies the type of missions. With the same type of mission, the user can achieve different quantities of points, as shown below:

missionSchemaId <dbl>	subType <chr>	points <dbl>
158	double	150
158	double	200
158	double	300
158	double	350
158	double	400
158	double	450

This occurs especially with the "double" category of `subType`.

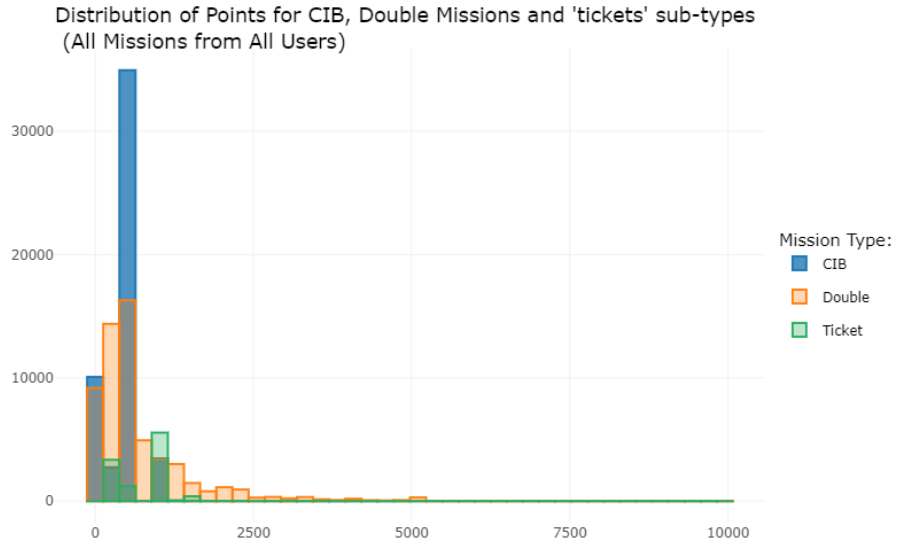
The variable `sub-type` is categorical and defines the type of mission. It has 3 categories: *cib*, *double* and *ticket-punti*. The exact meaning of these categories is unknown, but it probably refers to the quantity of points obtainable (especially for *cib* and *double*).

The next plot with histograms describes the points obtainable from each type of mission:



All distributions are skewed towards lower values. The *double* category has a wider distribution.

The next plot with histograms describes the points obtained from all users:

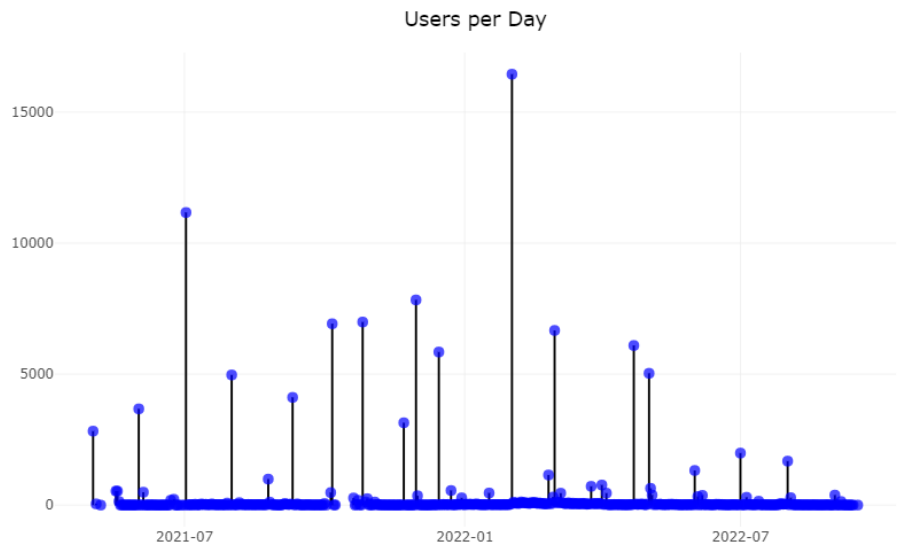


This shows a different phenomenon. The majority of users engage in missions

with lower points, and the most preferred type is *CIB*, concentrated around 500 points.

The variable `created_at` is in the format `yyyy-mm-dd hh:mm:ss`. It tells the moment when the mission is completed and thus when the points are gained.

It is possible to plot the distribution of missions completed over time:



The peaks in mission dates could be attributed to the following factors:

- Presence of specific events decided by the company.
- Deadlines for prize redemption.
- Mission points are imputed to users during pre-specified moments.

On an intra-day inspection, the missions are concentrated in certain seconds, leaving wide gaps. This could be due to database management choices or limitations.

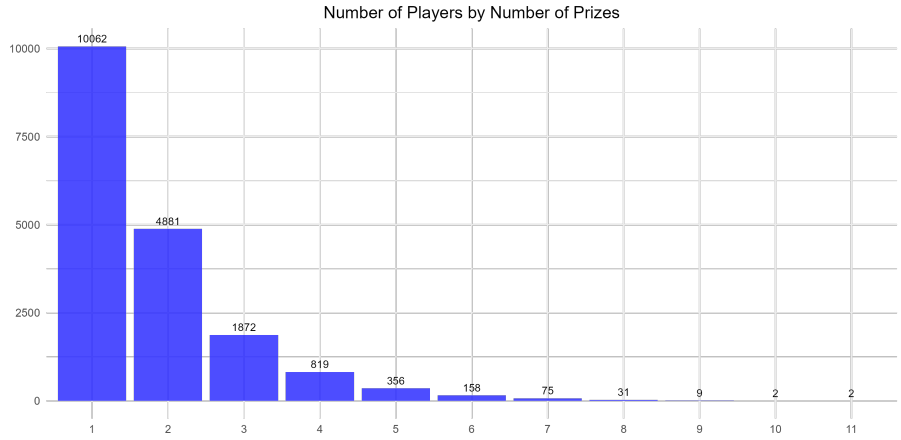
2.5 Prizes

In this data-frame, it is possible to track the prizes requested by each single user from March 2021 to September 2022. The data-frame has 32340 observations and 7 variables. The presence of null values and row duplicates is checked; no null values are found and only 166 row duplicates are found. These are not removed because it could happen that the same user requests the same prize twice or more during the same day, given that `datarichiestapremio` is in the format `yyyy-mm-dd`. A quite rare occurrence but still possible.

Analyzing `id_player`, it is possible to notice that 18267 people have requested at least one prize. So this means around 2/3s of registered users have not.

It is then possible to get a comprehensive list of the non-requesting users.

An interesting visualization is the count of the number of users who redeemed a specific amount of prizes:



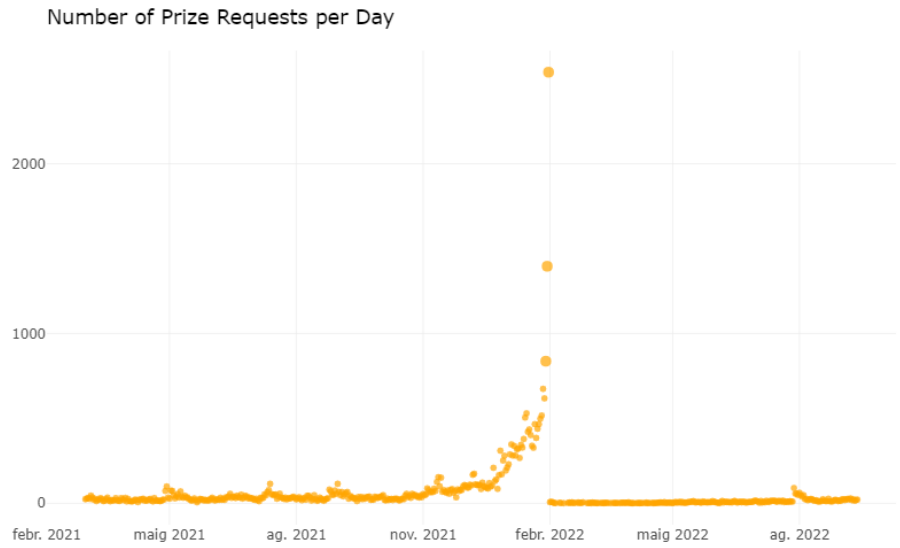
For each user, it is possible to calculate the total number of points she/he spent overall:

id_player <dbl>	TotalPoints <int>
101666	112000
359492	95000
587878	94000
513828	90000
717504	89000
215434	86000
389440	86000
95238	85500
68672	85000
243362	84500

1-10 of 18,267 rows Previous 1 2 3 4 5 6 ... 100 Next

For example, the user with ID `101666` has accumulated the most points, reaching 112000 points (!).

Also for prizes, it is possible to visualize an aggregate time series of prizes requested per day:



It is clearly noticeable a structural break in January 31st. This is due to the prize request deadline, as clearly specified in the legal documentation:

DENOMINAZIONE DELL'OPERAZIONE	**Raccolta Punti "Coccole Pampers-2"**
TIPOLOGIA DELLA MANIFESTAZIONE	**OPERAZIONE A PREMIO**
DURATA	Valida per gli acquisti effettuati dal 1° febbraio 2021 al 31 gennaio 2022. I premi e gli omaggi potranno essere richiesti dal 1 Marzo 2021 al 31 Gennaio 2022.

The majority of prizes requested are physical ones.

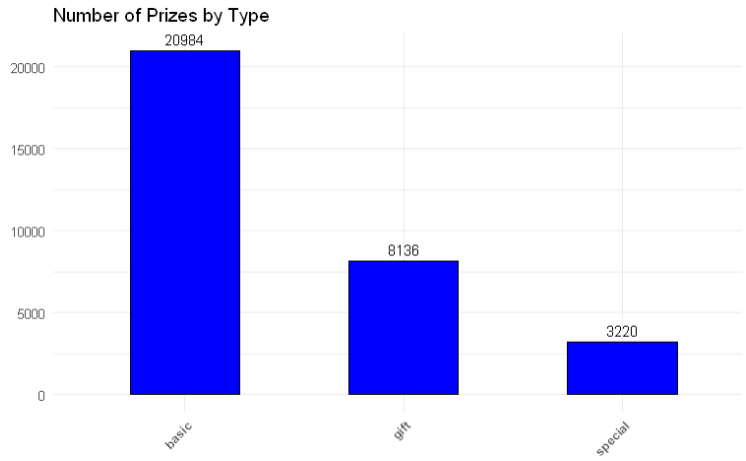
Starting from February 1st, all the points are lost and the new promotional period starts.

It is also important to point out the presence of a temporal dependency: before requesting prizes, it is necessary to have accumulated points through missions or loading products.

Moving to the other variables, such as `nomepremio`, `tipopremio`, `formatPremio` and `deliveryMode`, they are all categorical.

`formatPremio` (binary variable with *physical* and *digital*) and `deliveryMode` (binary variable with *email* and *home*) are perfectly correlated; hence, the first one is removed.

Plot for number of prizes requested per `tipopremio` is also made:



The summary statistics for each category are retrieved:

Description: df [3 × 10]

	mean <dbl>	sd <dbl>	median <dbl>	mad <dbl>	min <dbl>	max <dbl>	range <dbl>	skew <dbl>	kurtosis <dbl>
Basic	13183.95	6153.53	12500	6671.7	5500	25000	19500	0.51	-0.88
Special	0.00	0.00	0	0.0	0	0	0	NaN	NaN
Gift	2980.46	2273.82	3000	2965.2	1000	9000	8000	1.37	1.20

The *special* category, the least requested one, does not require point in order to be redeemed.

2.6 Products Loaded

This table is the simplest in terms of variables but the largest one in terms of rows. It describes the products loaded to gain points from each single user. It has `1596529` observations and `5` variables.

The dataset was evaluated for missing values, revealing that the `missionDetail` column contains over 1 million missing entries, accounting for 68% of all observations. Due to the large proportion of missing data and the limited utility of this column, it has been removed.

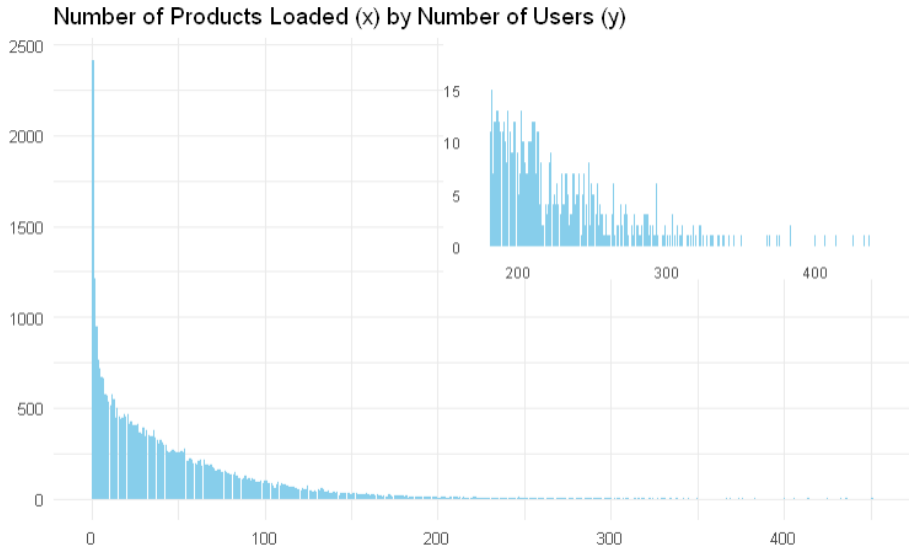
The number of unique values of `id_player` is 35949, representing around 80% of the overall population in `anagraphics`.

A table of the number of products loaded per user is created:

A tibble: 35,949 × 2

id_player <int>	n <int>
1272930	451
1077568	436
1390754	433
286240	425
490898	414
961522	406
58786	400
381086	383
1040118	383
506894	376

From this, it is then possible to visualize the number of products loaded by the number of users:



`EAN` is the linking column to the `EANTable`. With this, it is possible to get more details about the specific product loaded.

Some EAN codes presented in this data-frame do not belong to `EANTable`. The ones missing are already shown in `EANTable` subsection.

The count of products with non-EAN codes is shown:

Description: df [43 × 2]

EAN <chr>	n <int>
fitmktwipes	221551
	10086
fitmktpoe	8888
Fit1punti400	8807
Fit1punti200	7422
Fit1punti300	5662
WIPESBF1	3201
fitcopuponsl139	3064
WIPESSNS1	2935
fitwipescollecting	2815

`fitmkwipes` has over 220000 occurrences. These likely are cleaning wipes with low points associated, and hence not expensive to purchase.

The current table is merged with the `EANTable`. In this way, it is possible to count the specific products (`REFERENZA_DES`) along its generic categorization (`OCCUSO_DES`) loaded by every user.

The number of generic products loaded by all users is shown, along with their category mean price and loaded products' conditional mean price.

A tibble: 4 × 4

Product <chr>	Count <int>	Category_Mean_Points <dbl>	Users_Mean_Points <dbl>
Pannolini	1268843	664.2857	298.0967
Wipes	38122	237.5000	353.1137
Vario	469	100.0000	189.7655
Biscotti solubili	238	150.0000	150.0000

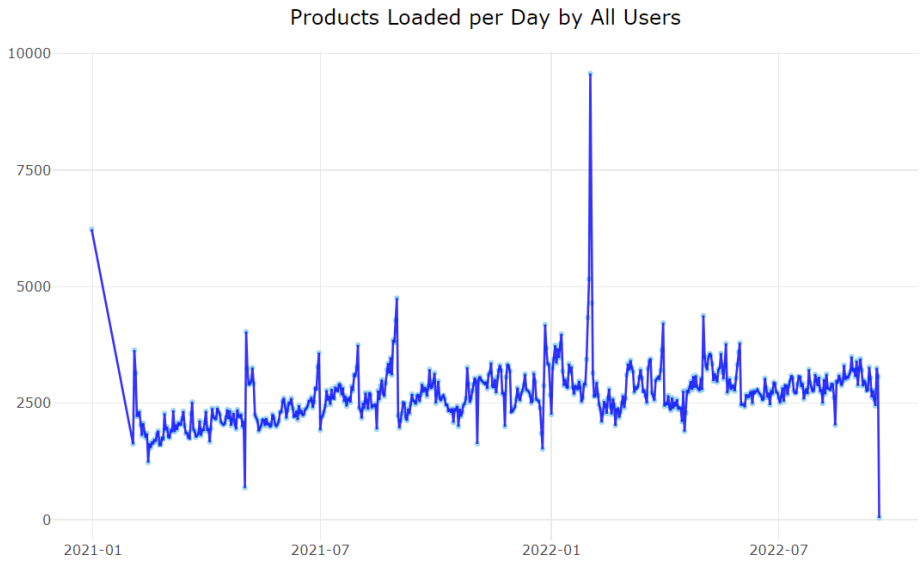
As expected, the vast majority of the products are diapers, that also show a category average higher price than the other ones, even though users load more cheaper diapers.

Now the same metrics are shown, but for `REFERENZA_DES`, i.e. the most specific categorization:

A tibble: 179 × 4

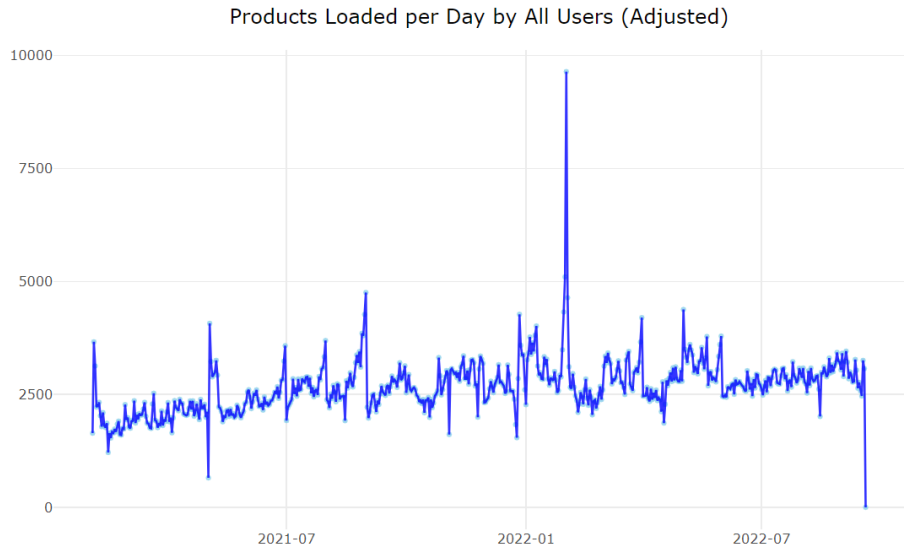
Product <chr>	Count <int>	Category_Mean_Points <dbl>	Users_Mean_Points <dbl>
Sluna XL CP	58775	75	145.4419
Sluna Junior CP	54257	75	146.0936
BD Junior Downcount	52455	75	145.9651
BD Extralarge Downcount	52422	75	145.6192
Sluna Maxi CP	43224	75	146.1549
BD Mutandino SP Junior	41475	100	194.1603
BD Maxi Downcount	41405	75	145.8628
Progressi Newborn CP	41182	100	194.5073
Progressi Mini CP	39429	100	194.0247
Progressi Maxi CP	37055	100	194.5486

Finally, as already similarly done for the other tables, a time series of overall uploads is shown:



This time series has a more expected behavior: products are loaded evenly during the period with few outliers. So the overall `products` table seems more reliable than, for example, the `missions`.

A gap is noticeable between the first observation `(2020-12-31)` and the second one `(2021-02-02)`. The first date is the only one that belongs to the first promotional period. All the products of this date are removed.



Two pre-processed data-frames have been saved: one with only diapers (the main product overall) and one with every type of product.

2.7 Customer Activity table

Since the `accesses` data-frame contains limited and distorted data about user activity, a new one is built based on **products** loaded, **missions** completed and **prizes** requested. In this way, it is possible to determine the activity of a customer as the number of *meaningful* accesses to the website/app in order to claim products, complete missions, or redeem prizes. Meaningful because, in order to do one of these actions (load a product code, complete a mission or redeem a prize) it is necessary to access. Also, the data window is larger, of almost two years.

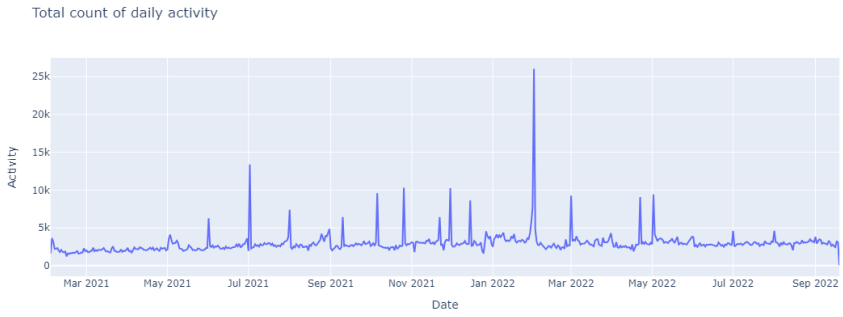
To build the activity dataset, the three data-frames (`products_loaded`, `missions` and `prizes`) are concatenated. This new dataset has been built using `Polars` in Python, a high-performance package built on Rust data manipulation library that offers significant speed improvements (up to 50 times) over `Pandas`.

The final dataset tells about each event, in particular when it happened, which player created it, and its nature (new defined variable `source`):

id_player	created_at	identifier	source
534214	2021-02-02 11:29:52	8001480092235	products
472666	2021-02-02 11:35:02	8001480092211	products
260826	2021-02-02 11:35:32	8001480059504	products
472666	2021-02-02 11:35:33	8001480093591	products
160610	2021-02-02 11:37:06	8001480301856	products
...	...	...	...
1939226	2022-09-19 01:33:06	8001090900227	products
1720458	2022-09-19 01:33:40	4015400327776	products
1896844	2022-09-19 01:46:42	fitmktwipes	products
1967866	2022-09-19 01:50:54	fitmktwipes	products
1966980	2022-09-19 03:28:56	4015400327776	products

1742315 rows × 3 columns

A visualization of all activities of all users is shown:



As we can see, we removed all the products loaded events that were collapsed on 31/12/2021, so they do not appear in this table too.

2.8 Points Trackers

The data provided by FATER did not include any single customer summary or behavior dataset; these kinds of data-frames (usually time series) are extremely important for user profiling, clustering, and, eventually, behavior prediction.

For this reason, a significant effort was made to create time series for each user. The main intuition behind this is that the behavior of a customer could be extrapolated from its personal points balance over time:

- when loading a product, their points balance would go up by an amount specified in the `points` value of the products table;

- when completing a mission, the balance would go up by the relative points won;
- when a prize is requested, the balance would go down by the amount needed for the prize.

Also, by looking at external sources (i.e., legal documents available on the web), we understood that, when a promotional period ends, the overall user balance is either halved (31st Jan - 1st Feb February 2021, end of promotional period 2.0) or completely reset to 0 (31st Jan - 1st of February 2022, end of promotional period 3.0). The code includes both, even though the first one isn't really needed, since those values were filtered from EDA.
A formula is defined mathematically:

$$\mathrm{TP}_{i,t} = \sum_{q=0}^{t} \mathrm{PRO}_{i,t-q} + \sum_{q=0}^{t} \mathrm{M}_{i,t-q} - \sum_{q=0}^{t} \mathrm{PRI}_{i,t-q} \tag{1}$$

where:

$$\begin{aligned} \mathrm{TP}_{i,t} &= \text{Total Points for user i at time t} \\ \mathrm{PRO}_{i,t} &= \text{Product Points for user i at time t} \\ \mathrm{M}_{i,t} &= \text{Mission Points for user i at time t} \\ \mathrm{PRI}_{i,t} &= \text{Prize Points for user i at time t} \end{aligned} \tag{2}$$

$$\text{and if } t = 365 \Rightarrow \mathrm{TP}_{i,t} = 0 \quad \forall i \ \bullet \tag{3}$$

Different users have different starting days (i.e., the first relevant relevant activity date). To avoid this, the time series are uniformed, adding trailing zeros if the first activity occurs after 1st of February 2022.

With this temporal information correctly coded, the points balance over time can become a fundamental piece of information to do customer clustering and prototyping, in the sense that customers with similar behavior would possess similar patterns.

Another time series was created based on the weight associated with each type of diaper, getting the values from *EAN_table* data-frame and then searching on the web the associated median weight. This approach allows us to track not only the evolution of the point balance, but also the changes in the weight classes of the loaded diapers.

This is very useful to pin down the "expected" behavior of a typical FATER customer: the child grows in weight as time passes, hence it is logic that the size of the diapers purchased and loaded increases over time; this is also useful in distinguishing the customers that are buying products for their babies from those that are buying size-less products (e.g. wipes) that may not be related to newborn growth.

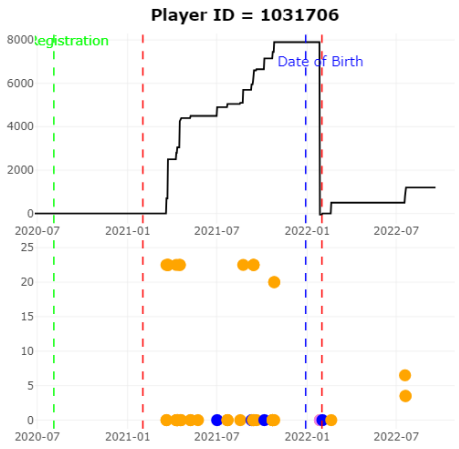
Focusing on product size evolution, it serves for a crucial purpose: distinguishing genuine churners from users who naturally stop purchasing due to their child outgrowing diapers. This distinction can be easily traced by examining the weight class of the last diapers purchased before app usage ceases.

This is a peek at the products data-frame augmented with this new information:

	Source <S3: AsIs>	Factor <S3: AsIs>	EAN <chr>	avgWeight <dbl>
41	progressi mutandino midi cp	4	8001090900180	8.5
42	bd mutandino cp midi	4	8001090956880	8.5
43	bd mutandino sp midi	4	8001090957054	8.5
44	progressi midi quadri	4	8001480035249	6.5
45	progressi midi penta	4	8001480035270	6.5
46	progressi midi cp	4	8001480041325	6.5
47	bd esapack midi	4	8001480058477	6.5
48	bd megapack midi bp	4	8001480058507	6.5
49	bd mutandino vp midi	4	8001480058828	8.5
50	progressi midi vp	4	8001480086890	6.5

41-50 of 239 rows Previous 1 … 3 4 **5** 6 7 … 24 Next

Let us take a look at the `Points Tracker` of a random customer:

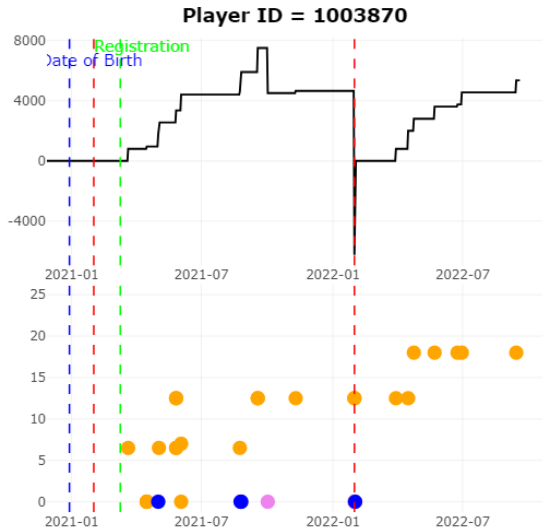


By looking at these trackers, many insights on both the users' behavior and dataset's nature assessed be found.

In this specific example, it is possible to notice that the points are actually correctly reset (or spent by the user) at the date relative to the end of the second promotional period. Also, it looks like the date of birth provided by the user corresponds to the end of their activity, which is a strange behavior; these kinds of illogical timings happen throughout all the users, and can be attributed to different causes (e.g. users are not truthful at registration or dataset is not

consistent). However, this newly generated time series is extremely useful for understanding the actual nature of the dataset.

A normal and expected behavior curve should show a surge in points and then a sudden drop, which is related to redeemed prizes. Also, coherent plot should not display negative values. We have found many players that break this rule, for example, player num. **1003870**:



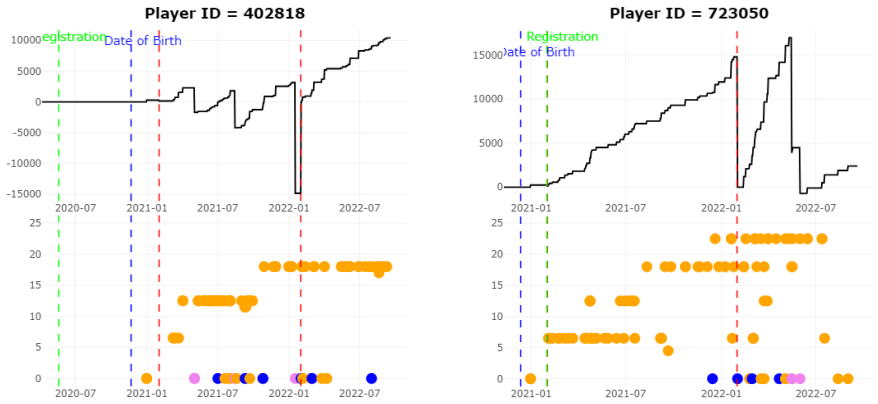
This could mean two things:

- there are other ways to gain points we cannot understand from the provided data, or
- the points in the data-sets are not coherent with the true ones gained throughout the promotional periods.

Let's take a look at some other interesting cases:

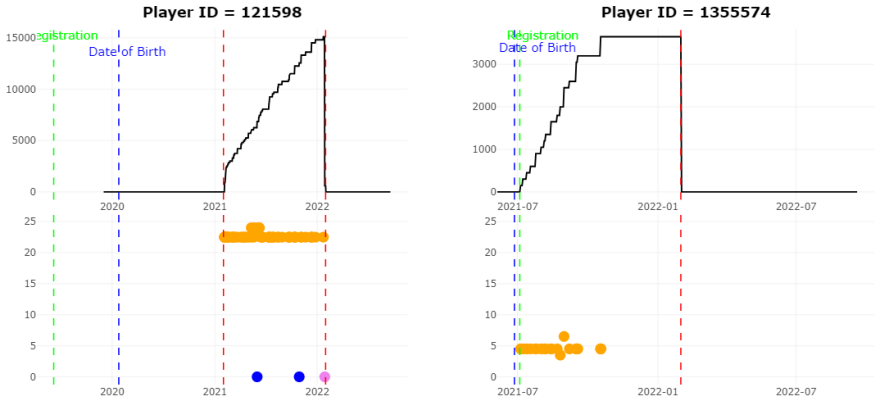
player 402818: 3 redeemed prizes in MP2. This goes against the MP2.0 rule which explicitly says maximum two prizes can be required.

player 723050: gets 250 points before registering.



player 121598: has an expected behavior from the typical FATER user; points are spent just before the reset at the end of MP2.0.

player 1355574 clearly shows a churn we are interested in detecting, especially by analyzing the weight progression:



We recall that, as stated by the app rules: "*Le Coccole eventualmente eccedenti la richiesta del 2 ° premio o non utilizzate per richiedere il premio al momento della scadenza della raccolta punti, saranno eliminate al termine della raccolta punti.*"

This selection of time series clearly shows the different and complex data at our disposal, yet it is still very informative.

2.9 What about our target variable?

The primary challenge is to define when churn actually occurs in order to deploy proactive customer retention strategies. The purchase of diapers and other products is not subscription-based. Without a clear manifestation of churn, making churn prediction becomes more difficult.

The most logical strategy is to define as a churner someone who has stopped buying diapers for more than a certain number of days. This is a basic strategy, straightforward to understand, but it does not take into account the fact that a customer could still be buying FATER products but not loading the products anymore.

Another important problem with this simple approach is that it is not clear how many days need to pass without activity to decide whether a user has churned or not; for this reason, an approach based on time series clustering could exploit the similarities in shape between the point balance of users, identifying customers *types* based on their behavior in earning/spending points.

The time series clustering algorithm (in our case, a crisp K-Means) is implemented to generate a predefined number of clusters. Each time series centroid can then be analyzed from a human-eye point of view to check if it matches with the intuitive idea of customer profiles.

For this reason, we have to define who the average customer actually is, ideally the FATER's *customer persona*.

Our prior belief is that the main customer persona is a mother/father that starts using the service when waiting for a child. This is coherent with the types of products sold, especially diapers.

There might be other personas defined by FATER, but they are not as important as the one mentioned above due to the centrality of diapers in our dataset and chosen approaches.

2.10 Clustering Methods

The approach that was taken in this project was to focus on the individual `Points Trackers` time series and use some clustering algorithms to identify customer prototypes or profiles, assuming that one of those would correspond to churners.

However, given the time-dependent nature of the trackers, an implementation of simple clustering techniques would not be suitable. Instead, a manipulation of them or a time series-specific metric is needed.

For this reason, we followed two different approaches to the clustering problem, **Window-based fragmentation** and **Dynamic Time Warping** methods.

2.10.1 Window-Based Clustering

The first method's goal is to transform each user tracker from time series to single rows, in order to create a dataset of customers compatible with the standard version of the K-Means algorithm.

The idea behind window-based techniques is to transform a time series into a "classical" tabular data-set: this is done by extracting a number of descriptive statistics (e.g. mean, std, median, etc) from a so-called *window* of **n** rows (time-steps).

In our case, each customer tracker (whole time series) is transformed into a single row, which columns correspond to the value of a particular variable (e.g. mean) over a particular period (e.g. first month of the promotional period, or first arbitrary subdivision of different length).

The points accumulated over time are scaled: this permits having more robust data to use for cluster analysis and minimizes the occurrence of distortions, especially for distance-based models.

Two methods are typically implemented for this: the first method normalizes the values but keeps the sign of negative ones; it basically consists of dividing each value by the maximum absolute value of the variable (or time series, in this case):

$$\text{Points}_t = \frac{\text{Points}_t}{\max\left(|\text{Points}|\right)} \tag{4}$$

Another method, often referred to as **MinMaxScaler**, normalizes the values between 0 and 1 included. All the negative values are shifted to $\mathbb{R}^+$:

$$\text{Points}_i = \frac{\text{Points}_i - \min(\text{Points})}{\max(\text{Points}) - \min(\text{Points})} \tag{5}$$

This has the advantage of turning all values to be positive, but can also create an upward "shift" that can seem unnatural.

Another step in the pre-processing involves trimming the time series: all trailing zeros are removed because they do not carry relevant information.

Because any customer only starts accumulating points after a certain day (its relative first day), each time series always starts with a sequence of zeroes (from the first available day of promotion to the first day of specific user activity); this usually doesn't carry any additional information, instead it may even bias any clustering algorithm, that could start to focus more on the initial zeroes trail (common to all time series) rather than the similarities of points accumulation/expenditure during the active period of the user.

To avoid this, all the time series are cut at the beginning and "start" at the first actual day of activity (points $\neq 0$) of the user.

Coming back to window-based clustering, the first step consists in dividing each time series in a sequence of temporally ordered and non-overlapping windows. Next, a series of summary/descriptive statistics (variables) of interest are calculated for each window, after which we will have created a series of features (columns) per period (per user).

In essence, consider two time series $\bar{X}$ and $\bar{Y}$, and let $\hat{X}_1 \dots \hat{X}_1$ and $\hat{Y}_1 \dots \hat{Y}_1$ be temporally ordered and non-overlapping windows.

The goal is to find the overall similarity between X and Y; it can be computed as follows:

$$Sim(\hat{X}, \hat{Y}) = \sum_{i=1}^{r} \text{Match}(\hat{X}_i, \hat{Y}_i) \tag{6}$$

When dividing the time series into windows, it is important for the number of periods of each time series to be equal; otherwise, the two time series would be transformed into different-sized rows, thus making it impossible for any clustering algorithm to work.

For this reason, each time series has been divided in an arbitrary (20) number of periods; on the other hand, following this procedure means that the longer the time series, the more time steps each sub-window will contain.

After the subdivision, each aggregate variable (e.g. mean, std, median) is calculated for each window.

The chosen statistics/variables that are associated to each period are:

- **Maximum** value of total points held in a day.
- **Minimum** value of total points held in a day.
- **Mean** of points.
- **Median** of points.
- **Harmonic Mean** of points.
- **Variance** of points.
- **Average Gradient** of change of points possessed (between consecutive days).
- **Frequency of update**, number (or frequency) of updates to the point balance.

It is important to point out that these are arbitrarily chosen; either because the easier to retrieve and/or understand or because being, at our knowledge, informative.

After having calculated all the features for all the windows, the data-frame looks like this:

	id	dob	reg	Min 1	Max 1	Mean 1	Median 1	Harmonic Mean 1	Variance 1	Average Gradient 1	...	Average Gradient 19	Fou 19	Min 20	Max 20	Mean 20	Median 20	Harmonic Mean 20	Variance 20	Average Gradient 20	Fou 20
0	1142	NaN	2018-01-04	0.176471	0.882353	0.511765	0.264706	0.289296	0.121142	0.018256	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000
1	1190	NaN	2015-12-04	0.021429	0.185714	0.067619	0.050000	0.044076	0.002074	0.004926	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000
2	1348	2021-06-02	2017-09-11	0.251678	0.422819	0.326816	0.342282	0.321168	0.001718	0.007779	...	0.001438	0.142857	0.996644	1.000000	0.999085	1.0	0.999083	0.000002	0.000160	0.047619
3	1380	2020-04-05	2017-04-12	0.243151	0.657534	0.263064	0.243151	0.253641	0.006010	-0.015543	...	0.000000	0.000000	0.958904	1.000000	0.993678	1.0	0.993450	0.000220	0.001644	0.040000
4	1562	2019-11-23	2019-10-02	0.019802	0.039604	0.022937	0.019802	0.021715	0.000040	-0.000171	...	0.000000	0.000000	0.000000	0.029703	0.003073	0.0	0.029703	0.000082	0.001061	0.035714

5 rows × 163 columns

A sample of 70% of customers from this data frame is used to train a *crisp K-means algorithm* to perform clustering. Here, *crisp* refers to the fact that each data point is assigned *deterministically* to a particular cluster without overlap. Since all our variables are quantitative, K-means is suitable for this application.

K-means is one of the most popular iterative descent clustering algorithms. It iteratively refines clusters to minimize within-cluster variance, assigning each point to the nearest cluster center based on a distance measure, in our case euclidean, also being the most common one. The main objective function, which K-means seeks to minimize, is given by:

$$\min_{\mathcal{C}} \sum_{j=1}^{k} \sum_{x_i \in C_j} \|x_i - \mu_j\|_2^2$$

This objective function drives each iteration in K-means to converge by assigning data points to the nearest centroid and updating each centroid to the mean of points in its assigned cluster.

Upon inspection of the `Point Trackers`, it was determined that the most meaningful number of clusters (hyper-parameter needed for K-Means) is 3, under the assumption that each customer can be identified to belong to three different profiles:

- "Good user": those who accumulate points, spend them for a prize (usually a few days before the reset dates) and then keep accumulating again.
- "New user": those who started to accumulate points recently (with respect to the day in which the data was collected) who have not arrived to a point balance big enough for a prize yet;
- "Churner": those who accumulate points, but at some point stop, after having or not redeemed a prize.

Here we can visualize the results of training the clustering algorithm on a subset of the features (min, mean and avgGradient):

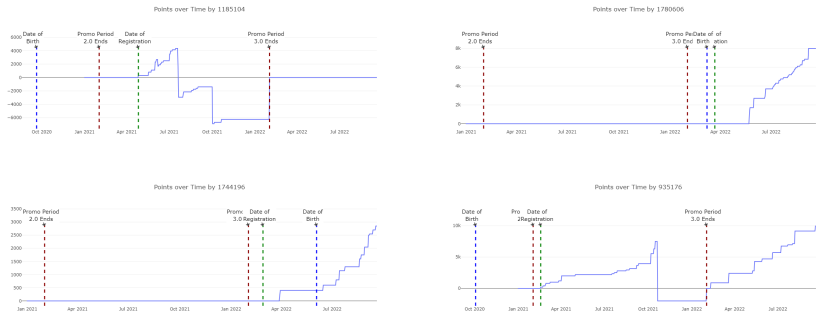


Figure 8: Sample of user trackers assigned to the first cluster

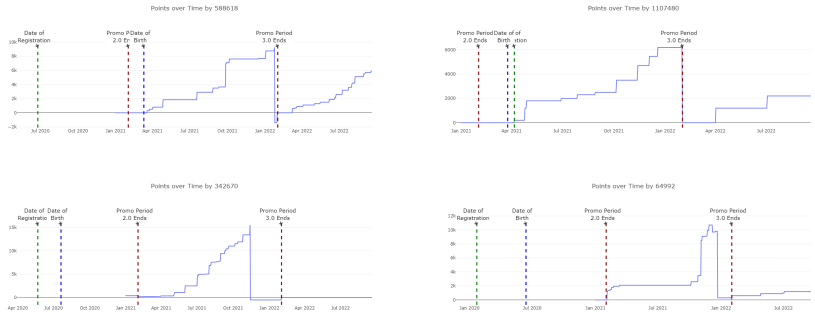


Figure 9: Sample of user trackers assigned to the second cluster

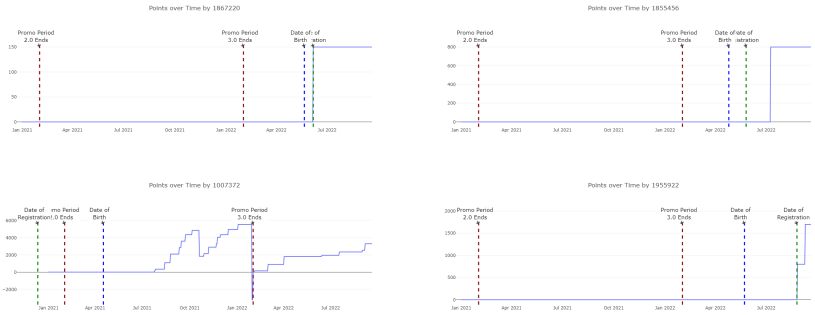


Figure 10: Sample of user trackers assigned to the third cluster

The algorithm seems to have learned a bit to differentiate between the three profiles, although it clearly makes mistakes. One of the reasons could be that the time series themselves cannot be associated to a specific profile (not even by human inspection).

Another reason could lie behind the multiple possible choices of aggregate features, given that there is nothing to suggest which subset of features will describe the differences between profiles the best.

Last but not least, the number of total columns (features) of the dataset is always equal to $\#periods \times \#features$; this means that it can get really big easily; this constitutes a real problem for the K-Means algorithm, since it suffers from the curse of dimensionality.

All these drawbacks led us to the application of time series-specific clustering algorithms, based on adequate metrics for time series like **DTW** or **SoftDTW**, which we describe in the next section.

2.10.2 DTW-based K-Means

The Time Series K-means algorithm with Dynamic Time Warping (DTW) or SoftDTW as the similarity measure is an adaptation of the classic K-means clustering algorithm specifically designed for time series data. Unlike traditional K-means, which relies on the Euclidean distance for measuring similarity between data points, this kind of time series K-means employs DTW or SoftDTW. These quasi-metrics are particularly well-suited for time series data due to their ability to handle temporal distortions, such as shifts or variations in speed and alignment. Quasi-metric because the triangular inequality property is not guaranteed.

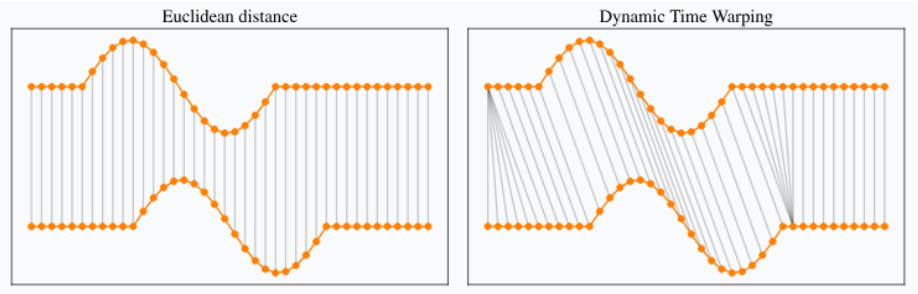


Figure 11: A graphical illustration of measure differences

Dynamic Time Warping is equivalent to minimizing Euclidean distance between aligned time series under all admissible temporal alignments

The steps followed by this kind of algorithms are common to the classical K-Means:

1. **Initialization**: randomly (or heuristically) select initial cluster centroids, which are time series sequences.
2. **Assignment Step**: for each time series in the dataset, calculate the DTW (or SoftDTW) distance to each cluster centroid and assign the time series to the nearest cluster.
3. **Update Step**: for each cluster, update the centroid by finding a new time series that minimizes the total DTW distance to other members of the cluster.
4. **Repeat**: iterate between assignment and update steps until convergence (i.e., centroids no longer change significantly or assignments stabilize) or termination critera are satisfied.

The key points in this time series specific context consist in:

- **DTW** aligns two time series that can vary over time, capturing similar shapes or trends even if they occur at different rates (click here to see illustration).
- **SoftDTW** is a differentiable, smoothed version of DTW, making it easier to compute and is generally more robust to local variations.

This method allows time series that are stretched or compressed in time to still be grouped meaningfully, which standard K-Means (using Euclidean distance) cannot handle well.

This approach is widely used in clustering problems where the time dimension is the most important one, hence being useful for our customer segmentation goal.

2.10.3 Dynamic Time Warping

DTW is a similarity measure (quasi-metric) between two time series, allowing for temporal distortions (e.g., stretching or compressing in time) to optimally align the series. Unlike Euclidean distance, which requires time points to align exactly (vertically), DTW accommodates time shifts and irregularities.

Consider two time series:

$$\begin{aligned} A &= (a_1, a_2, \ldots, a_n) \\ B &= (b_1, b_2, \ldots, b_m) \end{aligned} \tag{7}$$

where A has n points and B has m points. DTW finds the best non-linear alignment between these two sequences.

A *warping path* P is a sequence of index pairs that aligns the elements of A and B:

$$P = \{(p_1, q_1), (p_2, q_2), \ldots, (p_k, q_k)\} \tag{8}$$

where p_i and q_i are indices corresponding to points in A and B, respectively. The warping path must satisfy:

- **Boundary condition**: P starts and ends at the first and last points of both time series:
$$P_1 = (1, 1) \quad \text{and} \quad P_k = (n, m) \tag{9}$$
- **Monotonicity**: The indices should not decrease:
$$p_{i+1} \geq p_i \quad \text{and} \quad q_{i+1} \geq q_i \tag{10}$$
- **Continuity**: Only adjacent indices are considered in the path:
$$(p_{i+1} - p_i, q_{i+1} - q_i) \in \{(1, 0), (0, 1), (1, 1)\} \tag{11}$$

At each step along the warping path, we compute the *local distance* between aligned points a_i and b_j. This is usually the squared Euclidean distance:

$$d(a_i, b_j) = (a_i - b_j)^2 \tag{12}$$

The *cumulative distance* $D(i, j)$ between points a_i and b_j is the minimum distance from the beginning of the series to that point, considering all possible alignments. It is computed recursively as follows:

$$D(i, j) = d(a_i, b_j) + \min\{D(i-1, j), D(i, j-1), D(i-1, j-1)\} \tag{13}$$

where $D(i, j)$ is the cumulative distance at position (i, j), and $d(a_i, b_j)$ is the local distance at that point.

The total DTW distance between the time series A and B is given by the cumulative distance at the last points of both sequences:

$$DTW(A, B) = D(n, m) \tag{14}$$

This value represents the minimal total cost (or distance) of aligning the two series while allowing for shifts in time.

The time and space complexity of computing DTW is $O(n \times m)$, which can be large for long time series.

The key features of Dynamic Time Warp are:

- **Flexibility**: allows non-linear alignment by stretching and compressing parts of the series.
- **Optimal Alignment**: finds the best path that minimizes the cumulative distance.
- **Invariance to Time Shifts**: handles local time shifts effectively, unlike Euclidean distance.

2.10.4 Soft Dynamic Time Warping (SoftDTW)

SoftDTW is a differentiable modification of the classic Dynamic Time Warping (DTW) algorithm. The traditional DTW is non-differentiable, which makes it unsuitable for gradient-based optimization methods. SoftDTW overcomes this limitation by introducing a "soft-minimum" operation that smooths the DTW cost function.

Consider two time series:

$$\begin{aligned} A =& (a_1, a_2, \ldots, a_n) \\ B =& (b_1, b_2, \ldots, b_m) \end{aligned} \tag{15}$$

where A has n points and B has m points. Like DTW, SoftDTW computes the optimal alignment between these two sequences. However, it modifies the cost function to be differentiable.

At each step, we compute the *local distance* between aligned points a_i and b_j. As in DTW, this is usually the squared Euclidean distance:

$$d(a_i, b_j) = (a_i - b_j)^2 \tag{16}$$

In standard DTW, the cumulative cost is calculated using the min operation. In SoftDTW, this operation is replaced with a *soft minimum*, which provides a smoothed approximation of the minimum function. The soft-minimum for a set of values $\{x_1, x_2, \ldots, x_k\}$ is defined as:

$$\text{softmin}_\gamma(x_1, x_2, \ldots, x_k) = -\gamma \log \left(\sum_{i=1}^{k} \exp\left(-\frac{x_i}{\gamma} \right) \right) \tag{17}$$

where $\gamma > 0$ is a smoothing parameter. As $\gamma \to 0$, the soft-minimum converges to the exact min operation, and for larger values of γ, the function becomes smoother.

In SoftDTW, the cumulative cost $D_\gamma(i, j)$ at each point (i, j) is computed similarly to DTW but using the soft-minimum operation. The recursion formula is:

$$D_\gamma(i, j) = d(a_i, b_j) + \text{softmin}_\gamma\{D_\gamma(i-1, j), D_\gamma(i, j-1), D_\gamma(i-1, j-1)\} \tag{18}$$

where $D_\gamma(i, j)$ is the cumulative cost at position (i, j), and $d(a_i, b_j)$ is the local distance.

The total SoftDTW distance between the time series A and B is given by the cumulative cost at the last points of both sequences:

$$\text{SoftDTW}_\gamma(A, B) = D_\gamma(n, m) \tag{19}$$

This represents the smoothed cost of aligning the two series while allowing for shifts in time. The smoothness of the function depends on the choice of γ. A small γ will approximate the original DTW, while a larger γ provides more smoothing.

The key features of SoftDTW are:

- **Differentiability**: unlike traditional DTW, SoftDTW is differentiable, making it suitable for use in gradient-based optimization and machine learning models.
- **Flexibility**: likewise DTW, it allows non-linear alignments by stretching and compressing parts of the series.
- **Control over Smoothing**: the parameter γ controls the smoothness of the cost function. Smaller values of γ closely approximate DTW, while larger values smooth the alignment.

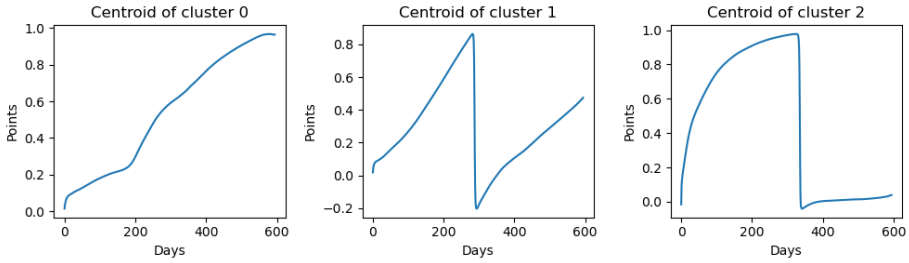
The time and space complexity of SoftDTW is the same as that of DTW, i.e., $O(n \times m)$. However, SoftDTW has the additional benefit of being differentiable.

Given that the time-series version of K-Means is a lot more computationally intensive than its classical counterpart, it is unfeasible to train it using a high percentage of the total trackers.

One fundamental advantage of this method is that it produces human-readable **cluster centers**, in the form of time series. In this way we can directly look at them and interpret them as *prototypes* of customer behaviors.

In our case, 30% of the total users have been included in the set; in case of slow convergence, the maximum number of iterations has been set to 4, the starting cluster centers are determined by the *K-Means++* procedure, and all other parameters have been left to default as provided by the `tslearn` Python package.

The resulting cluster centers at the end of the training phase were the following:



We can clearly see that the algorithm has correctly identified the three customer profiles we mentioned earlier.

Again we can visualize the results of training the clustering algorithm; by looking at to which cluster each Tracker is assigned, we can definitely tell that this new method is way more precise in identifying profiles:

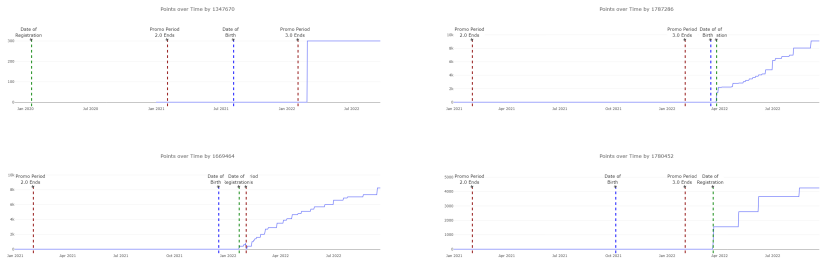


Figure 12: Sample of user trackers assigned to the first cluster

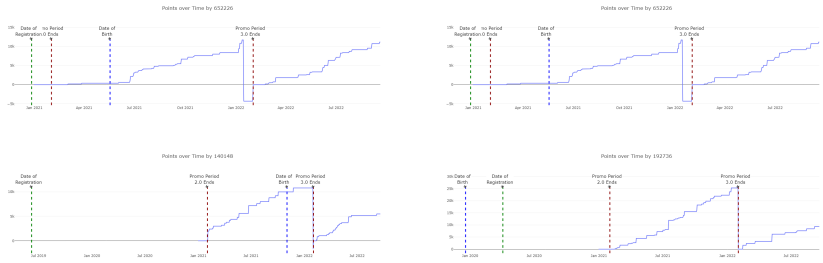


Figure 13: Sample of user trackers assigned to the second cluster

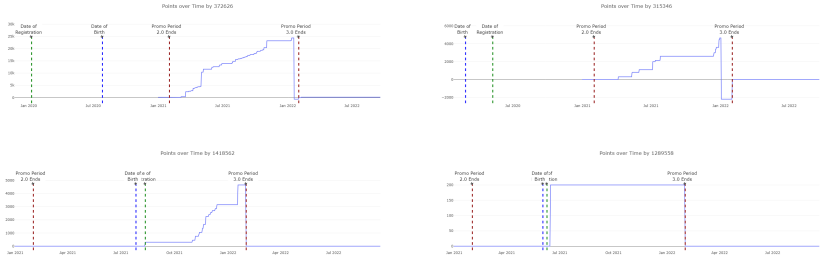


Figure 14: Sample of user trackers assigned to the third cluster

After inferencing the cluster information to all the point trackers (the remaining 70% not used for training data) the algorithm assigned:

- 37.04% of users to cluster 0
- 44.00% to cluster 1 and
- 18,96% to cluster 2 (churners).

3 Confirmatory Data Analysis: Modeling and Inference

3.1 Survival Models

Survival analysis is a statistical method that is primarily used to analyze the time until an event occurs, such as failure, death, or, in our case, churn. It originated in medical research, but its use has expanded greatly to many different fields. Survival analysis models are employed in different contexts:

- Banks, lenders and other financial institutions use it to compute the speed of repayment of loans or when a borrower will default.
- Businesses adopt it to calculate their customers **LTV** (lifetime value) or when a client will churn.
- Companies use it to predict when employees will decide to leave.
- Engineers/manufacturers apply it to predict when a machine will break (predictive maintenance).

The real strength of Survival Analysis is its ability to handle situations when the event of interest has not happened yet; this is also referred to as **censoring**.

To illustrate this, let us take the example of two hypothetical FATER customers and follow their active/churn status between January 2022 and April 2022:

- **customer A** started being active prior to the time window, and as of September 2022, is still a client of the company;
- **customer B** also started doing business before January 2022, but churned in March 2022.

We have an explicit depiction of the event for customer B. However, we have no information about customer A, except that he/she has not churned yet at the end of the January 2022 to April 2022 time window.

Survival models can take censoring into account and incorporate this uncertainty, so instead of predicting the time of the event, ***we predict the probability that an event happens at a particular time***.

The **survival function** is the building block of survival analyses.

A survival function $S(t)$ represents the probability that an event has not yet occurred in time t. It is defined as:

$$S(t) = P(T > t) \tag{20}$$

where T is the random variable for the time of the event. Typically, $S(t)$ starts at 1 (indicating 100% survival at $t = 0$) and decreases over time, as the probability of the event increases. Statistically, it can be considered as a cumulative distribution function.

Another function related to the survival function and also important in this context is the so-called **hazard function**, which represents the instantaneous event rate at time t, since the event has not yet occurred. It is defined as:

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t \mid T \geq t)}{\Delta t} \tag{21}$$

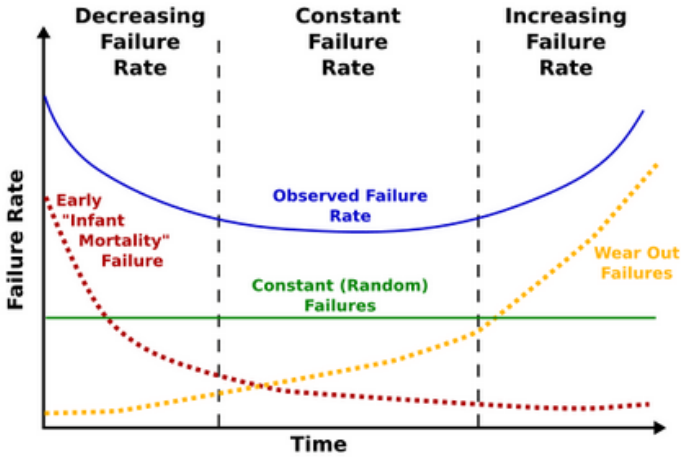


Figure 15: Examples of hazard function. The blue function is the so-called "bathtub curve", which describes the average person hazard function

The relationship between the two functions can be expressed as:

$$S(t) = e^{-\int_0^t h(u)\,du} \tag{22}$$

Thus, the survival function can be derived from the cumulative hazard, indicating how the survival probability changes over time.

Of course, these functions are theoretical and, in general, unknown for any particular study case, hence they need to be estimated.

Two prominent methods for estimating survival functions are the following:

- Kaplan-Meier Estimator: the most basic estimator, it a non-parametric statistic used to estimate the survival function from lifetime data.
- Cox Proportional-Hazards Model: A semi-parametric model that assesses the effect of several variables on survival time.

Both methods are essential for effectively analyzing survival data, particularly when dealing with censored observations.

The **Kaplan-Meier estimator** is a non-parametric method used to estimate the survival function $S(t)$ from time-to-event data, even with censored observations. The estimator calculates the probability of survival at each observed event time and is defined as:

$$\hat{S}(t) = \prod_{t_i \leq t} \left(1 - \frac{d_i}{n_i}\right) \tag{23}$$

where t_i is each unique event time, d_i is the number of events at t_i, and n_i is the number of individuals at risk just before t_i.

The Kaplan-Meier curve visualizes these survival probabilities over time, effectively capturing the decreasing survival trend.

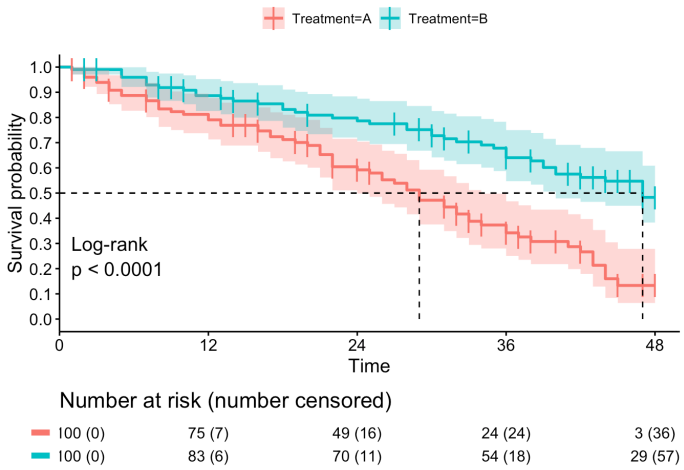


Figure 16: An example of two KM curve estimates, along with a log-rank test result rejecting the H_0 of no difference in the survival distributions

The Kaplan-Meier estimator has limitations, such as its inability to incorporate covariates, which means it provides a univariate survival estimate, without accounting for factors that might influence survival times. Additionally, it does not provide a way to compare different groups in a meaningful way when controlling for other variables.

In contrast, the **Cox Proportional-Hazards model** addresses these limitations by allowing for the inclusion of multiple covariates. This model estimates the effect of these variables on the hazard function, enabling researchers to analyze and interpret the influence of different factors on the survival while also providing hazard ratios for comparison.

Going more in-depth through the mathematical details, we can say that the Cox Proportional-Hazards model is a semi-parametric regression method used to analyze the effect of several covariates on the hazard function $h(t)$. It is defined by the equation:

$$h(t \mid X) = h_0(t) \cdot \exp(\beta^T X) \tag{24}$$

where:

- $h(t \mid X)$ is the hazard function for an individual with covariate vector X,
- $h_0(t)$ is the unspecified baseline hazard function,
- β is the vector of coefficients corresponding to the covariates.
- The quantity $\exp(\beta^T X)$ is called the *relative risk* for the specific individual feature vector that adds up to the baseline hazard function.

The model estimates hazard ratios, facilitating the interpretation of the relative risk associated with each covariate while assuming that the hazard ratios are constant over time. It is particularly useful in scenarios where the proportional hazards assumption holds, allowing to understand how different factors influence survival analysis.

Another assumption to be held is about the *non-informativeness* of censored data, meaning that censored individuals have the same probability of experiencing the event as those who are not censored.

Given these rather restrictive assumptions, a more flexible model is often employed: **Random Survival Trees**, a model developed by Hemant Ishwaran et al.
As a non-parametric approach, it extends traditional survival analysis techniques by allowing for non-proportional hazards and interactions between covariates. This flexibility makes them particularly useful in scenarios where the assumptions of the previously-cited Cox model may not hold. The low generalization error and flexibility that made RF a state-of-the-art model are properly retained in this extension.

As its popular counterparts for classification and regression tasks, a Random Survival Forest is an ensemble of tree-based weak learners grown on bootstrapped data. A Random Survival Forest ensures that individual trees are de-correlated by 1) building each tree on a different bootstrap sample of the original training data, and 2) at each node, only to evaluate the split criterion for a randomly selected subset of features and thresholds, in this case using as splitting criterion the log-rank test discrepancy. Predictions are formed by aggregating the predictions of individual trees in the ensemble.

3.2 Data preparation

We will incorporate **anagraphical** as well as **behavior-specific** features in order to exploit the most important characteristics that could be useful to discern churners and predict the total lifespan of any customer.

3.2.1 Anagraphical information

The anagraphical information (e.g. date of registration/birth, region etc.) has been extracted from the `anagraphics` table; during this process, the `Comune`, `SiglaProvincia` and `Provincia` and features have been dropped, since they

only hurt the analysis due to their high cardinality and do not carry relevant information.

The two dates features are modified to represent their *distance* (in days) from the data extraction day; this procedure transforms the variables from **dates** (difficult to interpret for models) to **integers** (straightforward interpretation).

To finish the preparation of the anagraphical information, the **class** (cluster) information is added to the dataset, this will act as our **target feature**.

	id_player	ETA_MM_BambinoTODAY	ETA_MM_BambinoREG	Regione	registration_diff	birth_diff	cluster
0	1878238	10	6	PUGLIA	-95	-301	0
1	1289508	18	3	TOSCANA	-467	-565	2
2	1170534	20	-27	LOMBARDIA	-1431	-613	2
3	964802	19	0	EMILIA-ROMAGNA	-566	-583	1
4	549416	22	-8	PIEMONTE	-909	-680	1
...	...	...	...	...	...	...	...

Figure 17: Our current dataframe. `cluster` is our target variable.

3.2.2 Target Variables

In every survival analysis, there are two main features each dataset should have in order to be compatible with the estimator methods:

1. A (binary or boolean) feature that tells us if the event (in our cas churn) occurred and

2. a feature that tells us the **tenure**, or time being alive (active), of each subject.

In our case the *churner* variable is easily extracted from the respective cluster assigned, while the *tenure* requires a bit more finesse.

This is because, technically, every user that has not churned is part of the *censored* part of the data, so it is actually unknown what their total tenure would be. To calculate tenures, the `custom_activity` table we have handcrafted comes in handy, under the assumption that the start of an user "lifespan" has to be determined by his/her first actual activity, and not by the date of registration, since that feature is highly inconsistent, as shown in the exploratory data analysis section; the end of their "lifespan" will also be determined from the `custom_activity` table, but this time there is a distinction between the two classes:

- For churners, the tenure will be the difference (in days) between their first activity and their last.

- For non-churners, it will be from their first activity and the last day of data collection (i.e., the latest date possible).

From the `activity` table, a number of users (2,096) have only a single recorded activity. These are unusable for our purpose and, in fact, pose a challenge to generating the tenure variable, as they would lead to errors. Therefore, we remove them from the dataset.

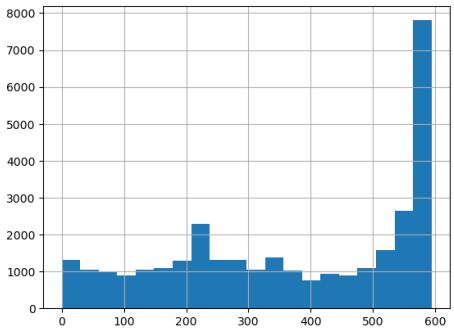
It could be insightful to understand the **average** time between activities for each player: in this way, it is possible to distinguish those with frequent activity from those with low activity; this feature is added to the dataset.

To calculate **tenure**, the first and last activity day for each user are needed, these are again extracted from the `activity` table.

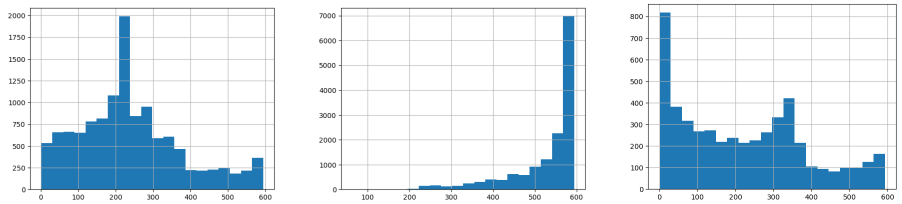
The dataset so far looks like this:

	id_player	ETA_MM_BambinoTODAY	ETA_MM_BambinoREG	Regione	registration_diff	birth_diff	cluster	first_activity	second_to_last_activity	last_activity	avg_days_between_activities	tenure
0	1878238	10	6	PUGLIA	-95	-301	0	2022-06-16	2022-09-06	2022-09-15	5.312500	95
1	1289508	18	3	TOSCANA	-467	-565	2	2021-06-09	2021-07-23	2021-09-26	17.833333	109
2	1170534	20	-27	LOMBARDIA	-1431	-613	2	2021-04-22	2022-02-01	2022-02-11	29.200000	295
3	964802	19	0	EMILIA-ROMAGNA	-566	-583	1	2021-03-02	2022-09-10	2022-09-10	8.123077	566
4	549416	22	-8	PIEMONTE	-909	-680	1	2021-02-23	2022-08-17	2022-08-31	12.511628	573
...	...	...	...	...	...	...	...	...	...	...	...	...

Let's take a look at the total distribution of the tenure:



This is the total distribution, a focus on the distribution across the three clusters can help gaining additional insights:



From left to right: tenure distribution across *newcomers*, *good users* and *churners*

By looking at the trackers, most of the *newcomers* with high tenure are just misplaced by the clustering algorithm (something inevitable) but a portion of

them are users that started to load points a long time ago, stopped for some reason and then started again recently.

Nevertheless, their presence should not pose any problem, given that they will be unified with the cluster 1 (since both classes are considered non-churners by our criteria).

Further investigation is also carried on about the three peaks of the tenure distribution of churners; it looks like the churners with a low tenure are just "once in a lifetime" users, they only make few points and then leave, we can drop them off without hurting too much the analysis since they are unpredictable and only hurt survival analysis as outliers.

To be consistent, all the rows are dropped with tenure less than 16 days (around two weeks), under the (somehow legitimate) assumption that this kind of "shift" in the zero day does not really affect the models, given that day act on relative differences and that the tenure usually arrives to at least 10 times the amount.

For our application, we only need to distinguish between churners and non-churners, so we merge cluster 0 and 1 together. This also helps turning the problem from **multi-class** to **binary** classification/regression, making it simpler.

3.2.3 Behavioral information

Some features may be extracted from the tables that track user products, missions and prizes such as the **most frequent** products redeemed, missions completed and prizes requested for each user.

While merging the "favorites" datasets together, if a customer happens to have never completed a mission or redeemed a prize, we will put "nothing" in the respective feature; this is useful in understanding mission completion and prize redemption influence on churning.

Another variable added to the dataset is related to the size of the diaper purchased; this will make use of the `item_tiers` table created during data preprocessing.

The fundamental idea here is that, by knowing the weight associated to the first item loaded, the weight associated to the last item loaded, and the median weight of all products loaded by a single user, a model could identify patterns associated with churning. For example, it may be that a user who consistently loads products of the same weight is more likely to churn, as the "natural" and expected behavior from an ideal customer would involve buying progressively larger products as their child grows.

Another possible correlation could be drawn between the weight of the last product loaded and the likelihood of churning. Intuitively, once a child reaches a certain size, they approach the point of no longer needing diapers, which would likely lead to a natural churn.

We will only keep the activity instances (and by consequence the players) that contain a loaded product with a weight of more than 0; the excluded products are usually **size independent** (e.g. wet wipes), and do not carry

important information to predict customer behavior in this fashion. Also, from the exploratory data analysis, we established that they constitute a minority of the total products loaded anyway, so we can discard them without losing too much information.

The final dataset consists in a total of 29365 rows (unique players) and 24 feature columns, though we already anticipate that some will not be used at all.

3.3 Survival Analysis

After having properly crafted a suitable dataset, it is possible to perform a survival analysis to generate an useful insight about into customer churn, hopefully developing a way to predict the churn probability of given user.

An important decision made in selecting customers for the statistical models was to limit the selection to those with a total tenure of less than 575 days.

This decision came after the realization that most of the churners with higher tenure had been placed in the cluster "2" because they were customers that churned before and after a long time returned and had a recent `last_activity` with respect to the date of data extraction (19/09/2022).

These customers are problematic to the survival analysis models, because they imply that a non-negligible portion of churners have a long tenure, even though they are not really churners; for this reason, we decided to only keep customers (regardless of cluster assignment) with total tenure of less than 575 days, which seemed a reasonable threshold to distinguish natural churners from long-term misclassified churners, without sacrificing too many correctly identified long-term non churners.

Another relevant choice is linked to whether to use the "nothing" information about preferred missions and prizes generated during dataset creation.

Keeping it (instead of dropping every row that contains at least one occurrence) results in a dataset that is twice as large. It also leads to higher scores and greater distinctions between churners and non-churners, which is expected, as players who fail to complete missions or claim prizes are likely to be churners, particularly when it comes to claiming prizes.

In addition, players that do not complete missions and do not claim prizes are likely to be short-term users which is confirmed by inspecting the distribution of tenure across these users. Any company would be less interested in preserving these kind of customers; on the other hand, it's more profitable to find a way to preserve the mid and long-term users, which are likely to spend more overall.

On the other hand, by dropping it, half of the total players is lost, but we get an in depth look at those customers that are more consistent and more important. We addressed both the situations for the sake of completeness.

The retained features are:

- `ETA_MM_BambinoREG`: age of the child at user registration.
- `Regione`: region of origin of the user.

- `avg_days_between_activities`: average time (in days) passed between activities.
- `SEGMENTO_DES`: most frequent segment of loaded products.
- `TIER`: most frequent product tier.
- `subType`: most frequent mission subtype.
- `tipopremio`: most frequent prize type.
- `deliveryMode`: most frequent prize delivery mode.
- `first_weight`: weight class of the first product loaded.
- `last_weight`: weight class of the last product loaded.
- `median_weight`: median of the weights of all products loaded.

The final dataset used for the analysis looks like this:

	ETA_MM_BambinoREG	Regione	avg_days_between_activities	SEGMENTO_DES	TIER	subType	tipopremio	deliveryMode	first_weight	last_weight	median_weight
0	-8	PIEMONTE	12.511628	Baby dry	2	double	basic	email	6.5	11.5	11.5
1	-33	TOSCANA	6.389610	Soleluna	3	double	basic	home	18.0	15.0	15.0
2	-22	SICILIA	15.680000	Progressi	1	cib	gift	email	3.5	11.5	3.5
3	8	PIEMONTE	17.677419	Baby dry	2	double	basic	home	12.5	18.0	12.5
4	-1	LOMBARDIA	9.212766	Baby dry	2	cib	gift	email	18.0	12.5	12.5
...	...	...	...	...	...	...	...	...	...	...	...

A correlation investigation (Pearson for numerical features, Cramér's V for categorical) hasn't shown particular insights:

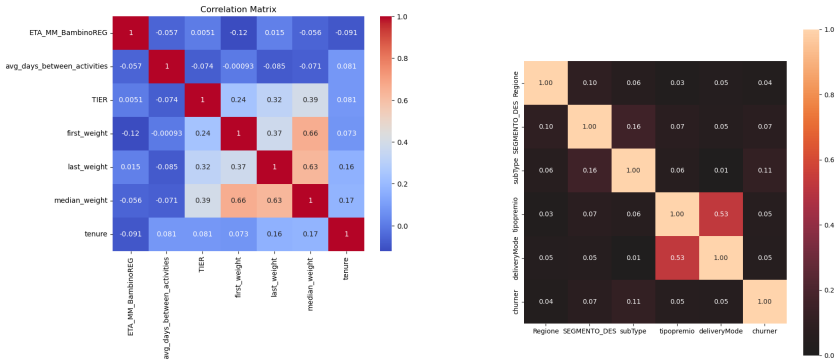
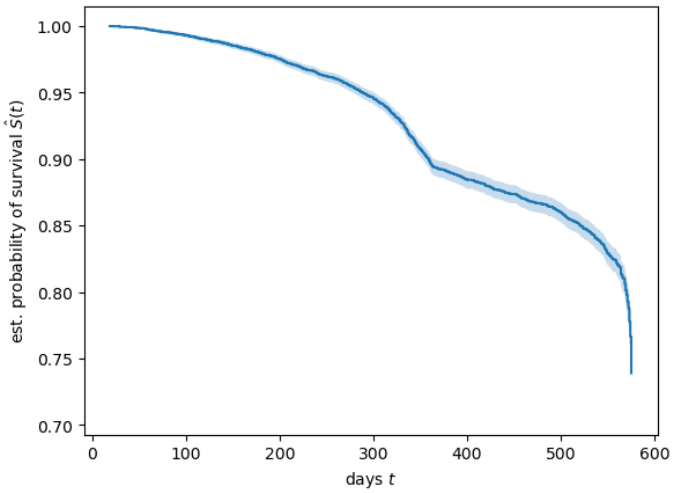


Figure 18: Correlation of numerical features (on the left) and categorical ones (on the right).

3.3.1 Kaplan-Meier estimator

The **Kaplan-Meier estimator** trained on just the prediction data (tenure + churner in boolean format) returned the following survival curve:

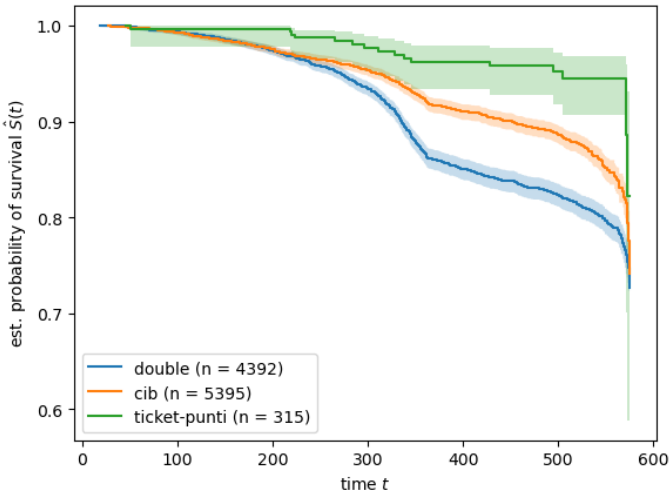


Something we can say about this curve:

- At $t = 0$ days, $\hat{S}(t) \approx 1.0$, meaning all subjects start event-free.
- $\hat{S}(t)$ decreases gradually, indicating an increasing rate of events as time progresses.
- Around 400–500 days, $\hat{S}(t)$ declines more sharply, suggesting a period of elevated risk.
- The shaded area around $\hat{S}(t)$ represents the confidence interval, which widens over time, reflecting higher uncertainty in survival estimates at later time points.
- By 600 days, $\hat{S}(t) \approx 0.75$, meaning about 75% of subjects are expected to survive up to this point.

Overall, the curve shows a general survival decline with increased risk from around 350 days onward and a higher uncertainty in survival estimates for longer durations.

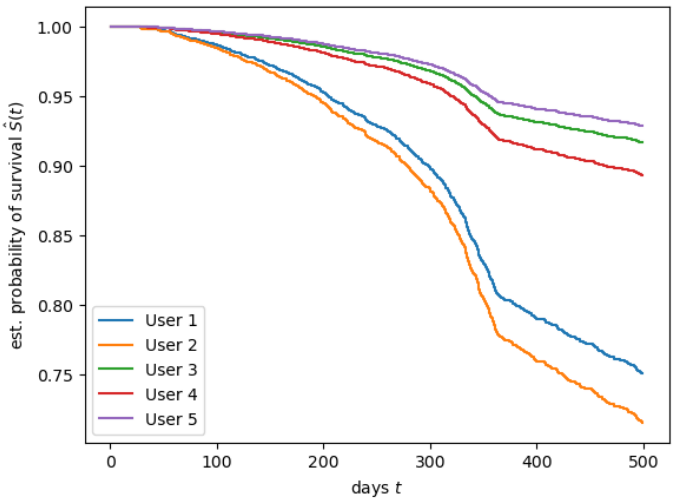
It's also possible (and useful) to plot the survival functions estimated for subsets of the users differentiated by a unique value of some feature. For example, we can take a look at the difference in survival probabilities depending on favorite mission type:



In order to continue with the analysis, the categorical variables have been encoded in a One-Hot fashion, while the numerical features have been standardized.

3.3.2 Cox Proportional Hazard estimator

The next model implemented is the **Cox Proportional Hazard** estimator. It is trained with the whole dataset (not just the information about churns); after training, it can be used to predict the survival function of any given customer, as shown in the next picture:



We can see that the model can definitely help us determine which customers are most likely to churn and at what t.

In general, it seems like any difference in survival can be noticed not before around 250 days; this is in accordance with the manual inspection of points trackers.

The main metric used to check the performance of these type of models is the **concordance index** (also called C-score).

It evaluates how well the model can discriminate between pairs of subjects in terms of their risk of experiencing an event (e.g. churn) over time. The C-index is essentially a generalization of the **AUC ROC** (Area Under the Curve Receiver Operating Characteristic) approach used in binary classification, adapted for censored data and time-to-event predictions in survival analysis.

The model has returned a C-score of 0.65. This is to considerate a moderate result, something that indicates that the model's predictions are fair to some extent. A value like this is kind of expected given that the problem is very difficult and there is a really low amount of churners with respect to censored users.

It is also possible to rank each feature by its resulting score when used as a single predictor (only column). In this way we can have a grasp of what are the most important ones:

first_weight	0.613287
subType=double	0.560501
median_weight	0.552290
avg_days_between_activities	0.530215
deliveryMode=home	0.523179
last_weight	0.521748
SEGMENTO_DES=Baby Dry Mutandino	0.517368
SEGMENTO_DES=Progressi	0.516850
TIER	0.511705
subType=ticket-punti	0.510537
ETA_MM_BambinoREG	0.506896
Regione=PIEMONTE	0.505445
Regione=TOSCANA	0.505232
tipopremio=gift	0.505201

The most valuable features are those we handcrafted, suggesting that our intuition about their importance was well-founded.

3.3.3 Random Survival Forest

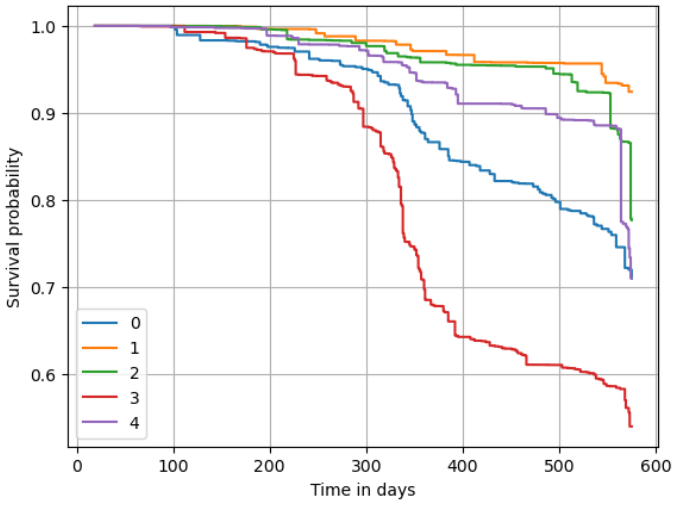
As previously mentioned, our last predictive model is a Random Survival Forest, an extension of its classical counterpart for survival analysis.

This time, having to deal with a statistical learning model, we must divide the dataset into training (70%) and test-set (30%). This has been carried on with a **stratification** method, so that the distribution of the two classes (churner and not) is kept as equal as possible between train and test splits.

After having split the dataset, the random survival forest is fit with default parameters and evaluated on the test dataset.

The trained RSF returned a concordance index of 0.7, which is an improvement from the Cox Proportional Hazard model.

Similarly to the previous one, this model can be used as a tool to predict risk factors and survival functions of given users:



The implementation used in scikit-survival is based on scikit-learn's Random Forest and inherits features like parallel tree building. However, it lacks the `feature_importance` attribute, as scikit-learn calculates feature importance by measuring impurity reduction, a method that doesn't apply to survival analysis due to censoring.

Instead of traditional feature importance, we can use **permutation importance**, which is preferred and implemented in `permutation_importance` in scikit-learn, fully compatible with scikit-survival.

Again, by looking at the most important features, we confirm that the newly generated variables have helped the model:

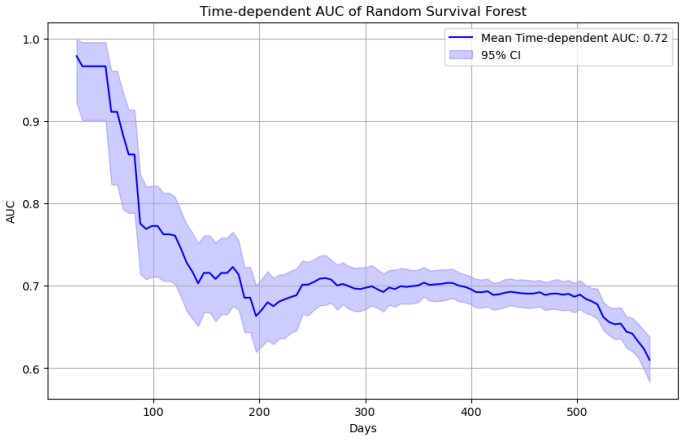
	importances_mean	importances_std
first_weight	0.078877	0.013833
avg_days_between_activities	0.035848	0.007711
last_weight	0.029799	0.006865
subType=double	0.028329	0.006760
ETA_MM_BambinoREG	0.025945	0.004893
median_weight	0.015804	0.004493
tipopremio=gift	0.002176	0.001742
Regione=TOSCANA	0.002086	0.001084
subType=ticket-punti	0.001740	0.001253

Finally, because of the same censoring problem as before, the AUC-ROC is not a suitable metric for random survival forests.

When extending the ROC curve to continuous outcomes like survival time, a patient's disease status changes over time. For example, a subject may be healthy at enrollment but develop the disease later, making sensitivity and specificity **time-dependent** measures.

To address this, we use **cumulative cases** and **dynamic controls** at each time point (t), resulting in a **time-dependent cumulative ROC**. Cumulative cases are individuals who experience the event at or before t ($t_i \leq t$), and dynamic controls are those with events after t ($t_i > t$). By calculating the area under this cumulative ROC, we assess the model's ability to distinguish between subjects who fail by time t and those who do not.

The `cumulative_dynamic_auc` function from `scikit-survival` implements this **time-dependent AUC** at specific time points:



In this plot we also included confidence intervals calculated through the **bootstrapping** sampling technique with a focus on the 95th percentile, they are represented as shaded areas around the curve.

We can see that the average **time dependent AUC** is around 70%, which is a fair value overall. It is evident that the model somehow performs really well in the starting [15, 100] days range; this is difficult to interpret, it could be caused by imbalances in the data, or, perhaps the model could be easily predicting early churning rather than long-term, which is plausible in a churning prediction context.

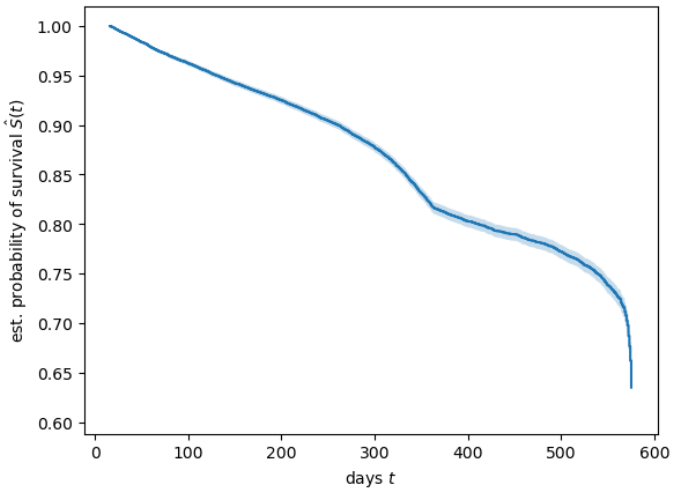
We can also notice how the model gets again more confident in the [350, 400] days range, while it falls short at the right extremity. This is both expected and favorable, because a lot of the churns happen in this range (end of promotional period), so predicting them correctly means having a higher possibility of intercepting important churning activities.

Survival Analysis with "nothing" values included

In this last subsection we illustrate the results of the same models trained on the totality of the dataset, thus also including all rows with possible 'nothing" values mentioned at the start of the subsection.

It is evident, as anticipated, that these are better results than those just discussed, but also must be taken with additional caution, as they may be coming from *biased* models, nevertheless, we think that these models could still be useful in predicting customer churn.

The total **survival function** generated by the Kaplan-Meier estimator looks similar to that previously shown in the "normal" case:



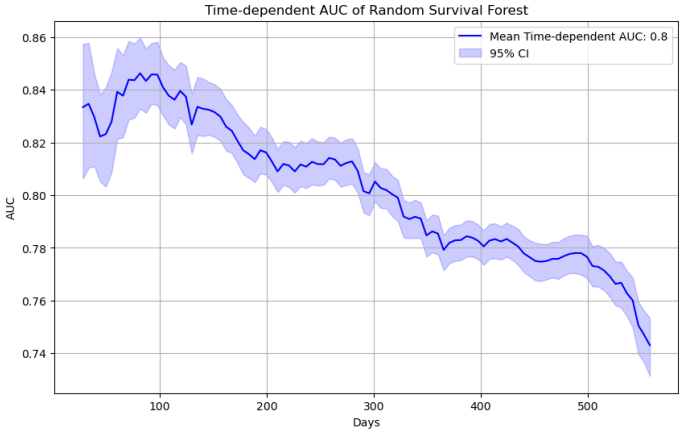
The only notable difference is in the fact that it reaches a lower probability at end range (before 75% , now 65%); this is expected since we already knew that a lot of low-life churners have been added to the dataset by including all the "nothing" values.

The **Cox Proportional-Hazard** model returned a higher overall C-index, around 0.76, a 10% increase with respect to the previous setting.

The **Random Survival Forest** also saw an increase in score to about the same value, 0.77; as we foresaw, the added information has become the most important feature for churning prediction, this is immediately understood by looking at the most important features returned after **permutation importance**:

	importances_mean	importances_std
subType=nothing	1.136030e-01	0.004940
subType=double	3.272763e-02	0.004013
first_weight	2.326523e-02	0.003787
avg_days_between_activities	1.628198e-02	0.003827
ETA_MM_BambinoREG	1.461555e-02	0.003002
last_weight	1.139252e-02	0.002933
deliveryMode=nothing	1.115061e-02	0.001673
tipopremio=nothing	8.834529e-03	0.001262
median_weight	3.964720e-03	0.002683

Finally, a time-dependent AUC ROC with confidence intervals is also shown to try to understand model performance:



It's obvious that the model performs better than before, it has a lower score at the very start from the previous, but it decreases more steadily and is always better at mid to long term ranges.

Given the appropriate precautions, the model can have powerful implications in customer churn and shows potential.

4 Intervention Strategies

The information generated by the proposed models can be helpful in retaining customers by acting preemptively through personalized intervention strategies.

The survival functions, hazard functions, and risk factors extracted from these statistical models would be the foundation behind early alerting and risk judgment that can put forward cost-effective interventions against churns.

Some intervention strategies could be focused on:

- **Early warning system** and **real-time** monitoring: an early warning system could be developed to flag customers whose survival probability dips below a threshold within a certain time frame. This real-time monitoring can help to act quickly whenever a customer's survival probability goes below a certain threshold.
- **Risk segmentation** and **customer tiering**: by identifying risk factors associated with higher churn, customers may be classified into distinct risk groups (tiers).
- **Timing** optimization of **retention offers**: by analyzing the survival function over time, it can be determined when customers are usually most likely to churn; with this information, periodical retention offers or promotional campaigns may be employed right before the peak churning probability.
- **Cost-benefit** analysis for retention action: survival analysis can also be exploited to determine the expected lifetime value of a customer, then a cost-benefit analysis can be carried on to understand if the eventual retention costs would be justified by a future expected profit.

These intervention strategies can help the company target customers more accurately, reduce unnecessary spending on low-risk customers, and maximize the impact of retention efforts on those most likely to churn. Regularly revisiting the model predictions and refining interventions based on new data (especially longer data with less censoring) will further improve the effectiveness of these retention strategies.

5 What Can be Done Better and Future Improvements

This work has been performed under several important problems, such as data inconsistency, incomplete information, missing values, illogical features and so on. This required us to think critically and outside of the box in order to extract as much knowledge as possible from what was presented to us. Although the analysis returned interesting results, it may still require significant improvements to be effectively deployed and presented to stakeholders. Some of such future improvements could be:

- Implementation of *soft* clustering techniques (e.g. Fuzzy C-means) instead of *hard* ones.
- Implementation of *hierarchical* clustering techniques.
- Acquisition of longer data to lower censoring occurrence.
- More in-depth parameter tuning.

- Exploration of more than 3 customer classes (requires more information).
- Cluster validation: Internal (Silhouette, Dunn scores), External (Adjusted Rand Index), Stability-Based validations (sensibility to data perturbance). About this point: it was tried to compute a silhouette score, but the high computational complexity of DTW made it unfeasible for any big enough fraction of the customers, further optimizations are needed to actually perform these type of scores.
- Distinction between *real* and *inevitable* churners.

The last point leads us to our final discussion about a **promising** approach that takes advantage of WHO curves.

5.1 WHO Growth Curves

From the exploratory data analysis, we retrieved the median weight associated with each diaper purchased. This can be considered as a proxy for the real weight of the children.

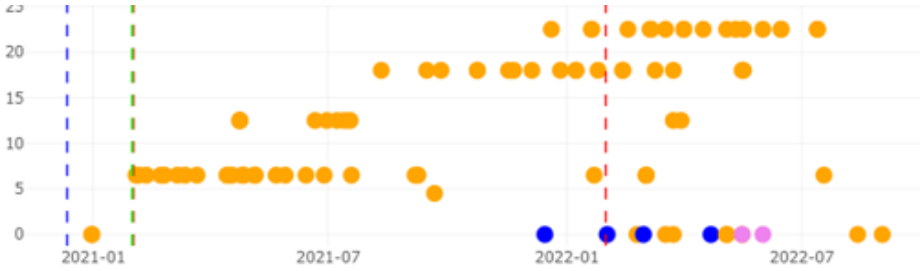


Figure 19: Example of weight progression over time

As shown in Figure 19, all the products with a weight (y) different from 0 are diapers. In this case, the weight progression of the diapers resembles the average weight progression of a healthy child. The upward progression also starts close to the green vertical line, which tells the date of birth.

This empirical distribution curve can be compared with the **WHO Standard Growth Curves**. The curves - also called 'charts' - are percentile curves showing the distribution of selected body measurements in children. Growth charts are used by pediatricians, nurses, and parents to track the growth of infants, children, and adolescents. From a descriptive statistics point of view, this is evident analyzing the conditional-time heteroskedasticity of weight, which being generally low during the first 2 years. The estimation procedure of the growth curves was very complex with thousands of observations, and the fact that it is followed trying to converge the weight growth - especially in OECD countries - of each newborn makes it especially suitable as the theoretical distribution curve.

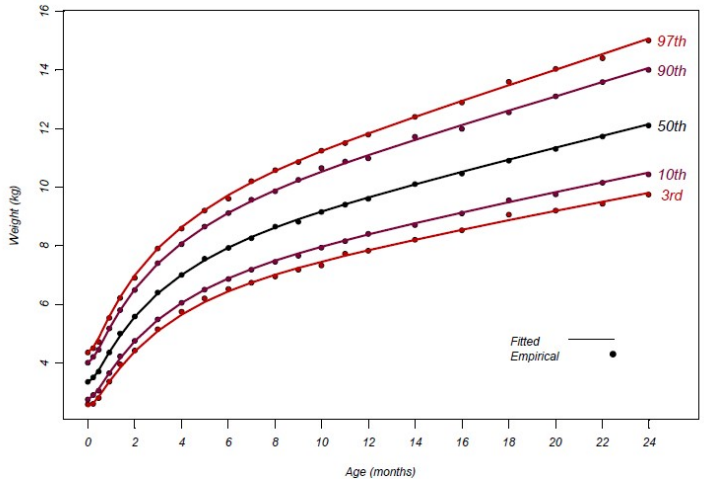


Figure 20: Standard WHO Smoothed centile curves and empirical values, weight-for-age for boys from birth to 24 months

The idea is to compare the two curves, the empirical and the theoretical, with a statistical goodness-of-fit test, such as the one-sample Kolmogorov-Smirnov test or more robust variants, such as the Kramer-Von Mises or Anderson-Darling tests.

For the KS test, the test statistic is the supremum (infinite norm) of the difference between the theoretical and the empirical distributions. The greater the supremum, the more unlikely the empirical distribution comes from the theoretical under consideration.

The statistic test finds its rationale in the Glivenko-Cantelli theorem and is calculated as follows:

$$D_n = \sup_x |F_n(x) - F(x)|$$

Obviously, as shown before, not all users are parents. This method or related ones can help us to identify real parents. As these are identified, churn monitoring and prediction could be considerably easier, especially if the purchasing pattern stops before 15 kilograms or less. In that case and if other products are not bought, the churn occurred because the consumer switched to another brand.

The complexity of this approach, even if straightforward to understand, lies in the data quality and the incidence of two children born very close over time. Moreover, the statistical complexity the theoretical growth curve (a Box-Cox Power Exponential estimated with GAMLSS) and the lack of this specific curve on Python limited our efforts. Also, the WHO standard curves are defined differently for boys and girls, an information we are not able to retrieve from the current data at disposal.

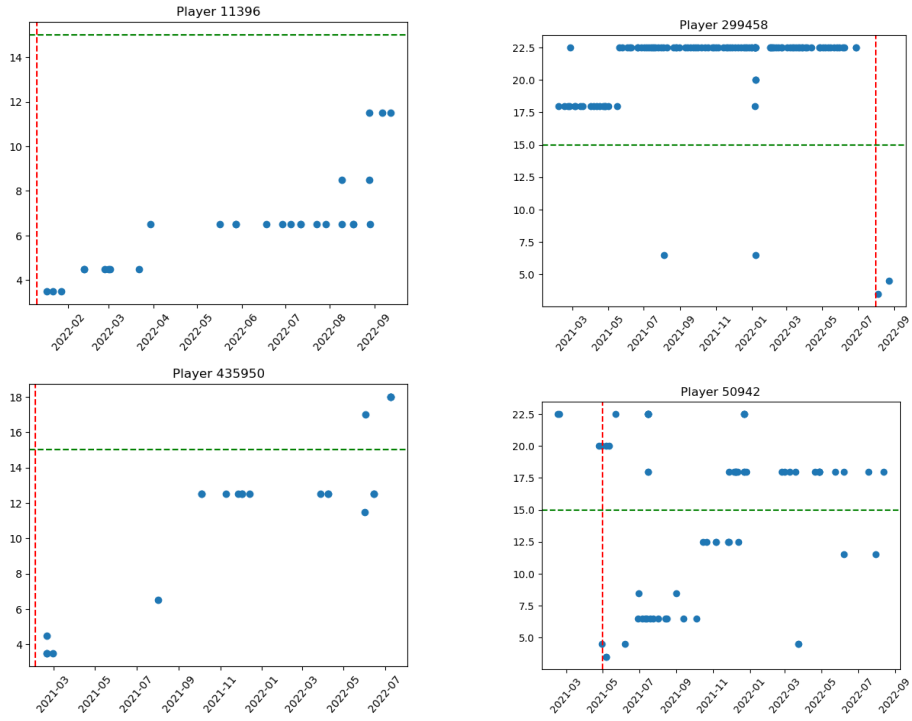


Figure 21: Four random examples of users

Sometimes, the variable `date_of_birth` is not consistent with the actual birth, a fact that add more uncertainty and make this approach, technically, more useful.

Bibliography

Pfeifer, P. The optimal ratio of acquisition and retention costs. J Target Meas Anal Mark 13, 179–188 (2005). https://doi.org/10.105/palgrave.jt.5740142

https://demo.istat.it/

Cleveland, W. S. (1979). Robust Locally Weighted Regression and Smoothing Scatterplots. *Journal of the American Statistical Association*, *74*(368), 829–836. https://doi.org/10.1080/01621459.1979.10481038

Han, J., Kamber, M., & Pei, J. (2012). *Data mining: Concepts and techniques* (3rd ed.). Morgan Kaufmann.

Cuturi, M., & Blondel, M. (2017). *Soft-DTW: A differentiable loss function for time-series.* In *Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 894–903). PMLR.

Kaplan, E. L., & Meier, P. (1958). *Nonparametric estimation from incomplete observations. Journal of the American Statistical Association, 53*(282), 457–481.

Cox, D. R. (1972). *Regression models and life-tables. Journal of the Royal Statistical Society: Series B (Methodological), 34*(2), 187–202.

Hemant Ishwaran. Udaya B. Kogalur. Eugene H. Blackstone. Michael S. Lauer. *Random survival forests. Ann. Appl. Stat. 2 (3) 841 - 860, September 2008.* https://doi.org/10.1214/08-AOAS169

WHO Child Growth Standards Length/height-for-age, weight-for-age, weight-for-length, weight-for-height and body mass index-for-age Methods and development Department

https://www.ospedalebambinogesu.it/estate-tempo-di-togliere-il-pannolino–80520/