POLITECNICO MILANO 1863

PROJECT REPORT DOCUMENT

DATA INTELLIGENCE APPLICATION - A.Y. 2019-2020

Pricing and Advertising

Authors	ID Numbers
Andrea FURLAN	944774
Cosimo RUSSO	945891
Giorgio UGHINI	944710

June 30, 2020 Version 1.0

Contents

1 Introduction

1.1 The Product

In this project we focus our attention on a mid-luxury scarf that is sold in a price range starting from €100 to €400. This item is a seasonal product and it is more searched by its buyers in the cold seasons rather than in the summer, when it has less demand. This periodic variation of the users' interest is a remarkable input to take into account during our analysis. We decided to use as a subject the classical Louis Vuitton brand logo scarf, well known and trending in our country.

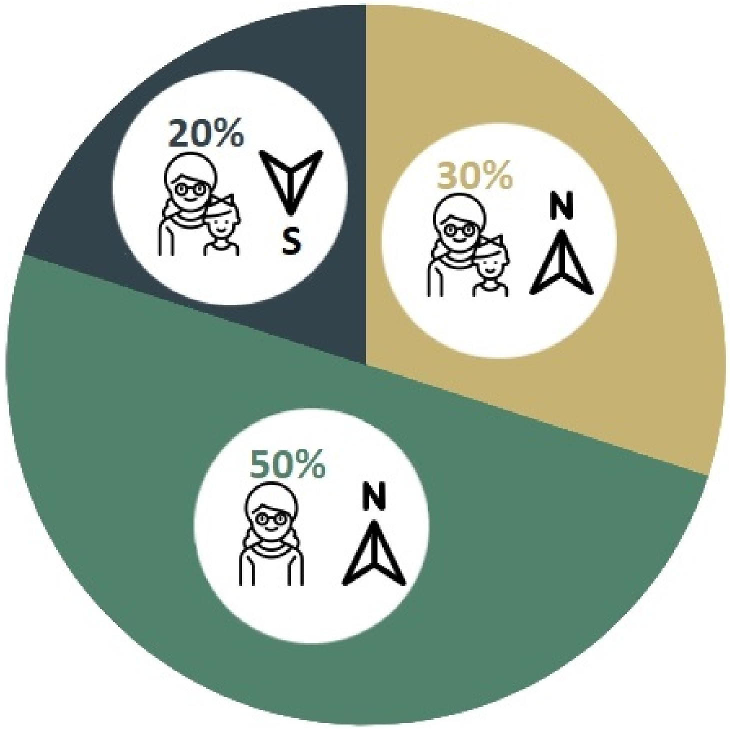
Figure 1: Louis Vuitton Brand Logo Scarf.

1.2 The Classes of Users

Given that the product we want to study generates more interest in women rether than in men, we decided to focus our studies on women only. We also came to the conclusion that since a luxury scarf is usually bought from the middle and the upper class person, a minimum family yearly income bound was necessary to set our focus on only the users that are able to generate more impact on the analysis. We set this bound at €80.000 gross. Finally, we thought that since trends can vary among countries, considering only the italian population was the most reasonable choice to avoid unwanted biases. The binary features that we decided to observe on our users are:

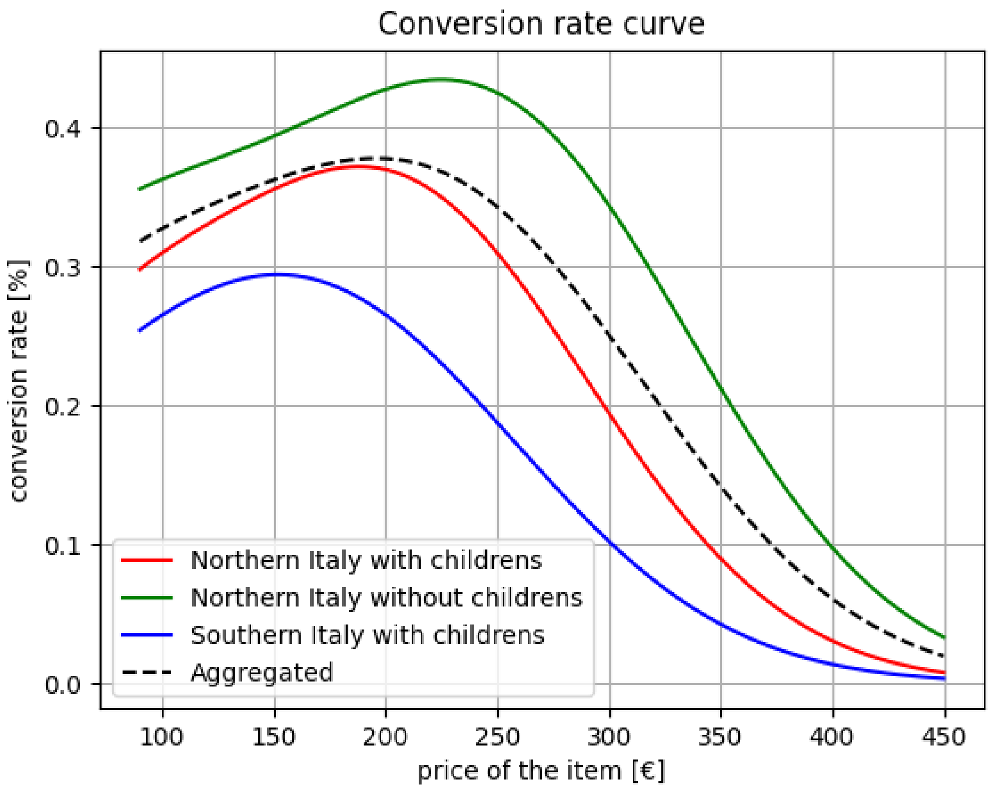
- **With/Without children**: To better observe the impact that a son can have in buying a luxury item
- **Living in the North/South of Italy**: To better observe the impact that the climate and the local traditions can have in buying an item that is useful only in certain seasons.

The distribution of our customer base is described in the following pie chart:



1.3 The Conversion Rate Curves

Since the Louis Vuitton brand logo scarf is classified as a luxury good, we hypothesised the three conversion rate curves to have a non-negative-slope on low prices and reaching their peak near the €200-250 range.
These curves are the curves that rule the environment of our experiment.
An illustration of our hypothesised conversion rate curve is presented below:

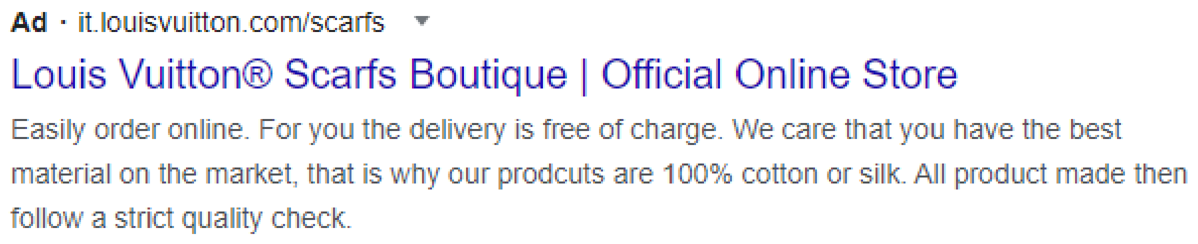


1.4 The Subcampaigns

In order to advertise our product we outlined three different sub-campaigns, each with a unique aspect and a particular type of user as a target. The three sub-campaigns are focused on:

- **The Quality Of The Product**:
  Targeting the women with children living in the north of Italy.
- **The Social Impact Generated By The Product**:
  Targeting the women without children living in the north of Italy.
- **The Peculiarity And The Uniqueness Of The Product**:
  Targeting the women with children living in the south of Italy.

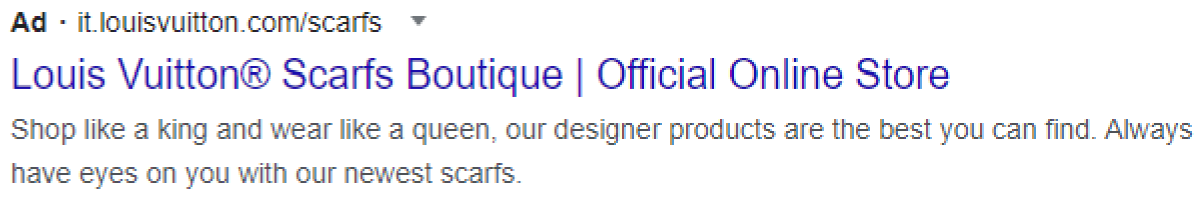
An idea of how the ads could be is the following:



Ad · it.louisvuitton.com/scarfs

Louis Vuitton® Scarfs Boutique | Official Online Store

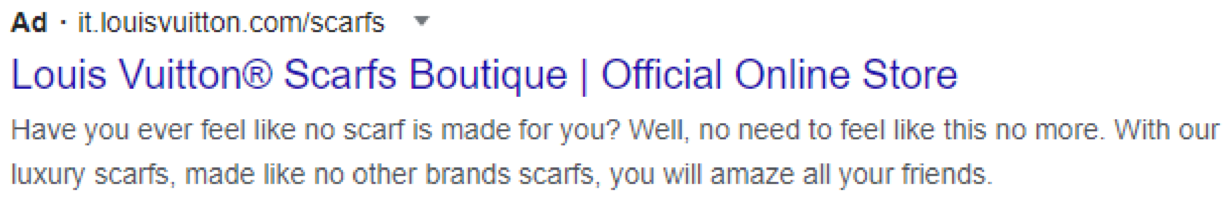
Easily order online. For you the delivery is free of charge. We care that you have the best material on the market, that is why our prodcuts are 100% cotton or silk. All product made then follow a strict quality check.



Ad · it.louisvuitton.com/scarfs

Louis Vuitton® Scarfs Boutique | Official Online Store

Shop like a king and wear like a queen, our designer products are the best you can find. Always have eyes on you with our newest scarfs.



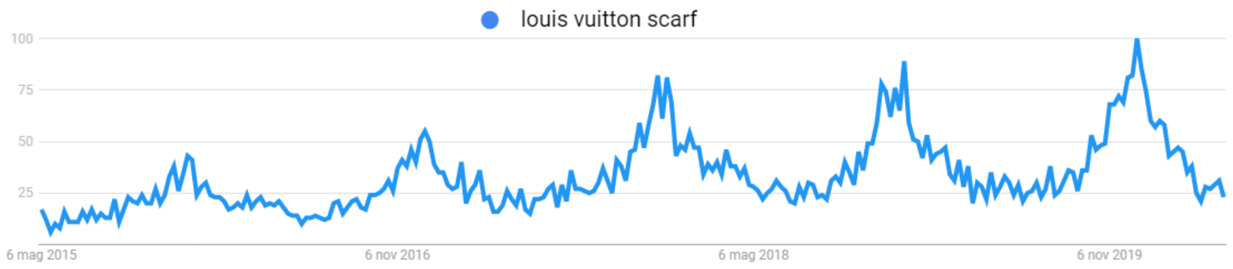
Ad · it.louisvuitton.com/scarfs

Louis Vuitton® Scarfs Boutique | Official Online Store

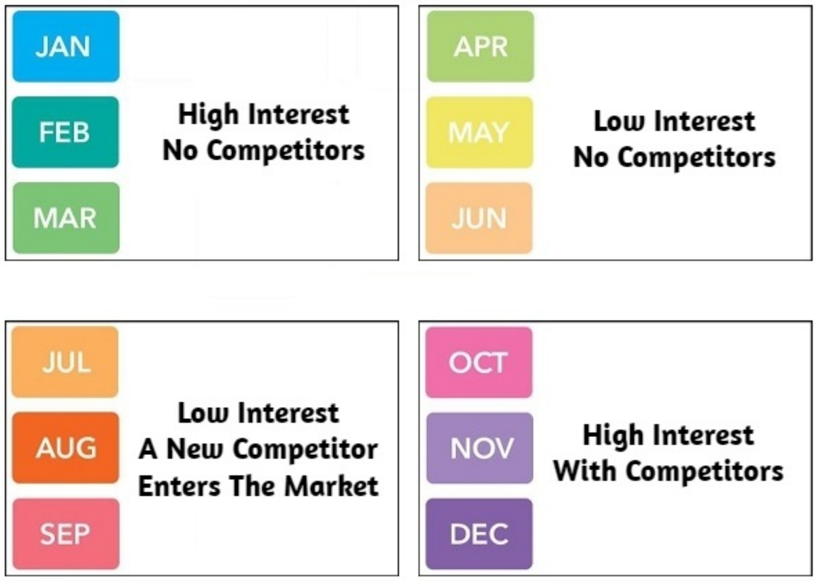
Have you ever feel like no scarf is made for you? Well, no need to feel like this no more. With our luxury scarfs, made like no other brands scarfs, you will amaze all your friends.

1.5 The Abrupt Phases

As we said in the first section, the product taken into analysis is a seasonal item. Using Google Trends, we were able to identify the region of maximum and minimum interest of the users regarding a Louis Vuitton scarf. In the graph below is clear that this product is searched primarily in the cold seasons and it is year after year more demanded by the customers. In the chart the x-axis represents a five year time period while the y-axis represents the popularity of the keywords ”Louis Vuitton Scarf” searched on Google (100 is the moment of maximum interest while 0 is the minimum).

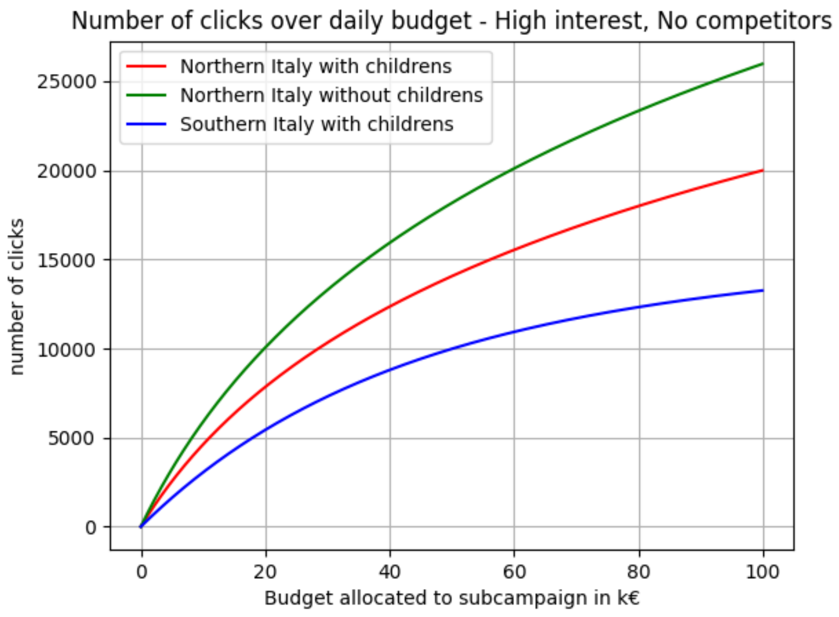


Based on the diagram we decided to divide the year in four periods of time, each one underlining a different phase.

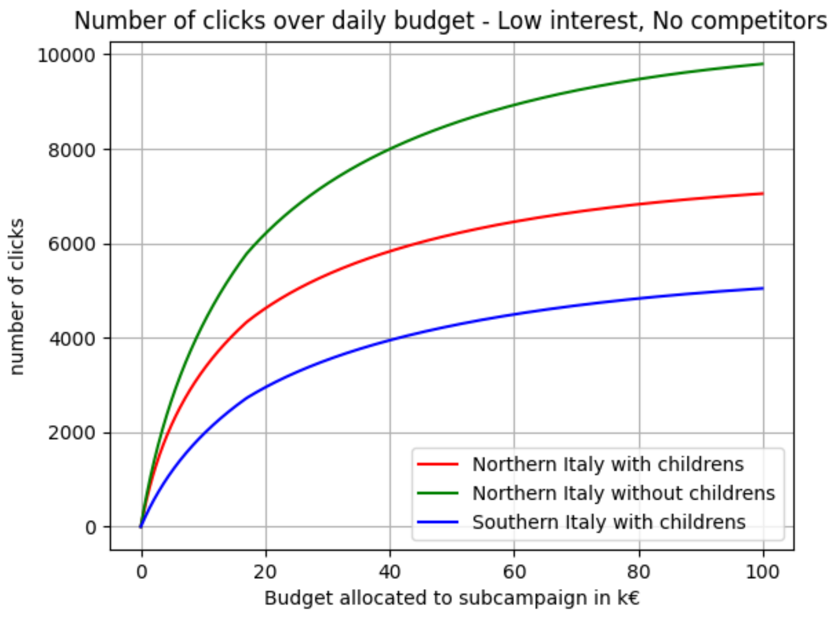


JAN FEB MAR	**High Interest** **No Competitors**
APR MAY JUN	**Low Interest** **No Competitors**
JUL AUG SEP	**Low Interest** **A New Competitor** **Enters The Market**
OCT NOV DEC	**High Interest** **With Competitors**

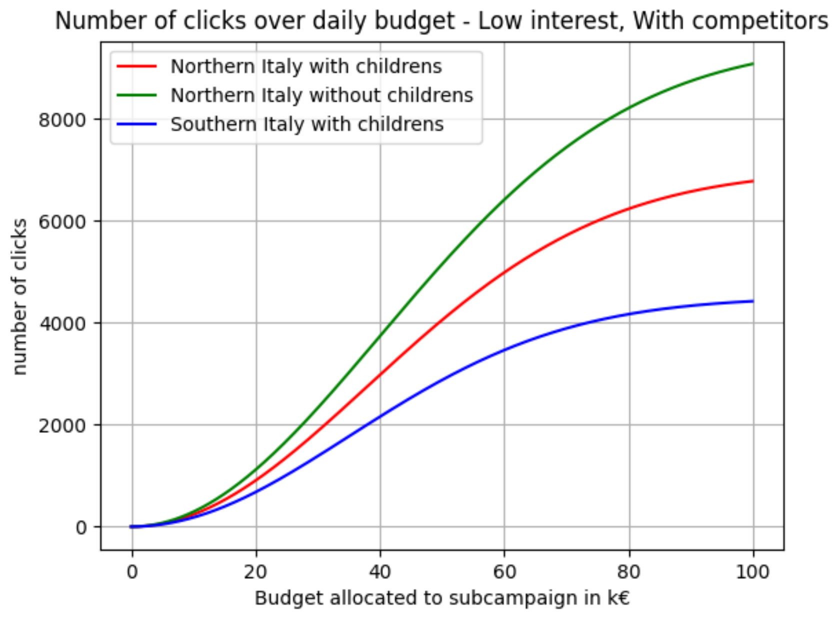
In order to better understand our reasoning behind the decisions that brought us to choose as graphs for the four different phases the ones showed below, is important to notice some remarks.
In the first phase (Jan-Feb-Mar) the growth of the curve is almost linear, since there are no competitors that can interfere with our advertising of the product. However, there is a large pool of users to influence, since it is a phase of high interest of the product.



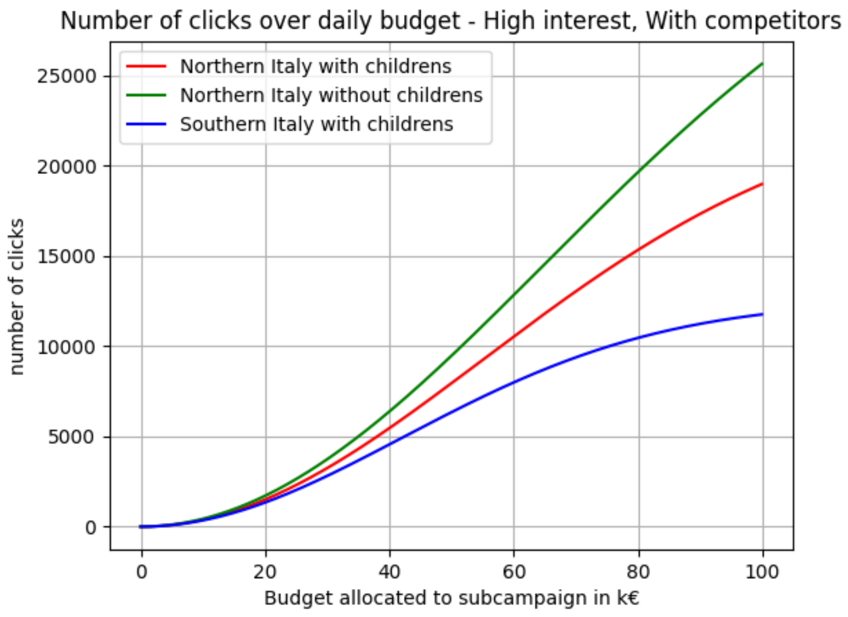
In the second phase (Apr-May-Jun) instead, since in these months the interest about the item is low, there will be less people interested in the scarf, therefore the impact that a higher budget allocated to the ads can make is less relevant than the impact that it can produce on a high interest phase. Based on this assumption we decided that the graph for the second phase should have a more remarkable growth with a small budget allocated to the ads, settling to much more slower growth as the budget significantly increases.



In the next phase (Jul-Aug-Sep), when the new competitor enters the market, the graph is different in the low budget allocated range with respect to the second graph, while maintaining a similar profile in the high budget allocated range. The cause is the following: since we need to beat the competition in order to sell our product, we would struggle to do so allocating a small amount of money to the advertisement, while instead allocating a sufficient amount of money to a certain class, we should be able to attract more user of this class.



Finally, in the last phase (Oct-Nov-Dec) we focused on the fact that since it is an high interest phase, this time with a competitor that challenges us, the budget that needs to be allocated to each sub-campaigns still needs to be a considerable amount if we want to attract a sufficient number of users as in the third phase. However, in this period the pool of users that can be reached is much wider, translating into a more linear curve in the high budget allocated range with respect to the third phase.



1.6 The code

In this document we will refer to algorithms implemented by us in Python which code can be found at https://github.com/GiorgioUghini/dia_project.

2 Case study and Algorithms

2.1 Budget allocation with only one phase

The first step of our project was focused on the budget allocation over the three subcampaigns and had as goal the maximization of the total number of clicks. In this first stage, for simplicity, only one phase is considered.

2.1.1 Experiment setting

The first step of the process is the definition of the size of the experiments, in particular:

- **The number of arms**: the higher it is, the more likely is the chance of exploring arms that are close to the optimal arm. On the other hand, having more arms implies that the optimization horizon will be required to be longer to allow a proper exploration of all the arms.
- **The optimization horizon**: this is the amount of time that will be spent running the price optimization algorithms. The longer this is, the more are the possibilities that can be explored. However, exploring more possibilities causes to have a bigger loss - or regret - in the initial phases of the exploration.

It appears clear that the the two quantities depend one on each other and represent the key of the optimization. A reasonable optimization horizon is three months. Indeed, the interest that people have for our product behaves in the same way every year in the same three months considered. The phases alternate cyclically and the period of the cycle is one year. Thus, as long as we are able to collect enough data during three months, it makes no sense to extend the optimization horizon to a time that is longer than this. The choice of the number of arms is an hyper-parameter that the corporate should input, by setting the minimum budget to allocate to each subcampaign ($min - budget$), the maximum one ($max - budgets$) and the granularity ($step$).
For this experiment, we set the same minimum budget for all subcampaign (equal to €1000) but different upper bound as we consider to have (very low) prior information about their classes of user. The upper bound for the budget was then set to €5400, €5800 and €5200 for the three categories of users, reflecting the fact that we can afford to spend more for his top spending users. As a luxury brand, we decided to have a high granularity in the budget

allocation, to exploit at best all the advantages of the CMAB approach in finding the best allocation.
Indeed, we could even have some prior information about users behaviour, but for sure we cannot a-priori know what is the best choice for the budget allocations as it depends upon too many factors.
In particular, *step* = 2 that means the allocation of budget has a granularity of €200. With the same granularity but different allocation range, each GP Learner resulted in a different number of arms (respectively 22 arms for North/WithChildren, 24 arms for North/WithoutChildren and 21 arms for South/WithChildren).

2.1.2 Gaussian Process Combinatorial Multi-armed Bandit

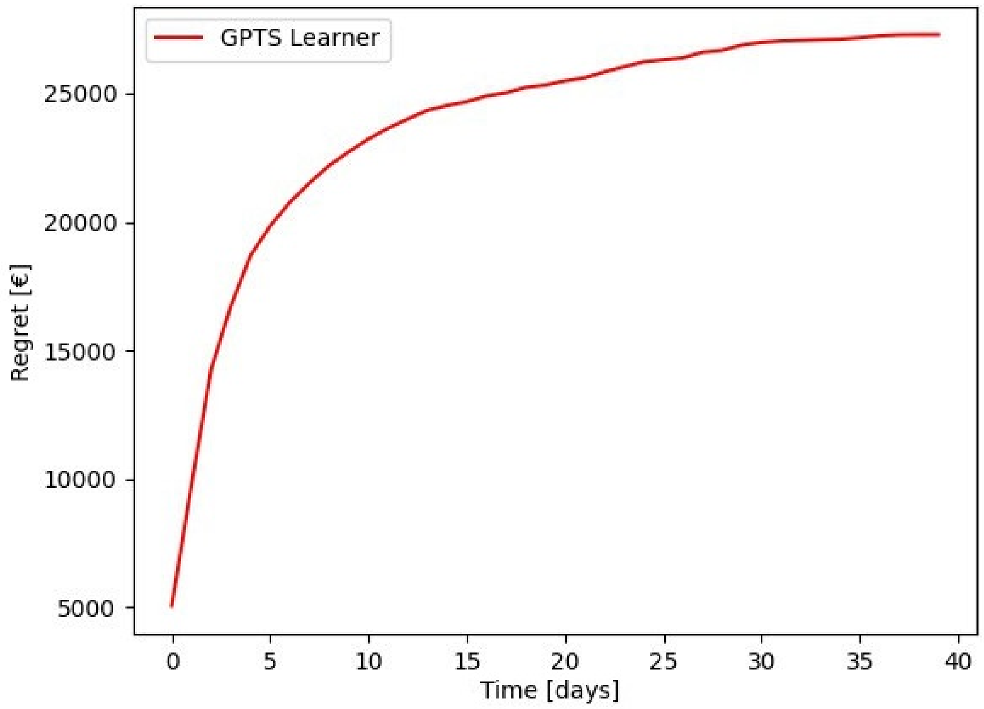
The pseudo code of the algoritm we implemented is as follow.

Algorithm 1 Gaussian Process CMAB

```
 1: $J \leftarrow$ all classes of users
 2: for $day \in T$ do
 3:     for $j \in \{1, ..., J\}$ do
 4:         $s \leftarrow$ Sample j-th GP-Learner
 5:         Add $s$ row to knapsack matrix
 6:     end for
 7:     Optimize knapsack matrix
 8:     Play selected superarm
 9:     Update GP-Learners model
10: end for
```

2.1.3 Results

The experiment shows that after only 7 days of exploration, the corporate could start to exploit the best arms, converging to the optimal one after about a month. The regret chart that resulted from this experiment is as follows:



2.2 Budget allocation with four phases

After having analized the budget allocation over the three subcampaigns considering only one phase, we are moving to a more general context.
In particular, we supposed the existance of four different phases in a year, that are:

- **JAN-FEB-MAR**: These months can still be considered ”cold months”, so the *interest* in a scarf *is still high.* In addition, we suppose that *no competitor* will launch a new product in these months because that competitor would miss the OCT-NOV-DEC high interest phase.
- **APR-MAY-JUN**: In these spring months *the interest* in a scarf *becomes low* as temperatures arises. Of course, *no competitors* are present for the same motivation as above.
- **JUL-AUG-SEP**: If a *new competitor wants to join the market*, the best time of the year to do so is the late-summer/beginning of fall. In this way, even if this phase can still be considered a *low interest phase*, the competitor could start to promote its product to be ready for late-fall and winter.
- **OCT-NOV-DEC**: A *new competitor has joined the market* in the previous months and now the customer base has to take a decision on which is their favourite, possibly based on the marketing strategy each party encompasses. These cold months will be, of course, a *high interest phase.*

2.2.1 Experiment setting

In case of non-stationary environment, it is important to define a proper value for the sliding window, that is, the length of the buffer where the latest collected samples are stored. We considered two different formulas to compute a reasonable sliding window length, which pros and cons are reported below:

- **Unknown environment**: This is the basic formula for the computation of the window length. It only considers the number of interactions T our algorithm is going to perform with the environment.

$$SW_1 = \sqrt{T}$$

It is a very simple formula, altough with large numbers of arms it leads to bad performances, as it does not take them into account.

- **Some environment knowelge**: Provided that all the phases of the demand curve have the same length so that they are uniformly distributed over the time horizon, we're setting the sliding window size equal to a third of this length, in order to be more ready in case of a abrupt change (like in correspondence of the beginning of a new phase) at the cost of re-exploring the same arms at least 2 times inside the same phase. With P as the number of phases:

  $$SW_2 = \frac{T}{3P}$$

  This is a very simple formula, tailored for a specific case, the phases should be uniformly distributed over the time horizon and their lengths have to be equal, which is not always the case.

We supposed that the compay could have a small prior information about the distribution of the phase that could come from previous campaign analisys or other sources.
As the phases are uniformly distributed and we know the number of it and the optimization horizon, we implemented the second formula for the Sliding Window lenght that resulted in:

$$SW = \frac{365}{4 * 3} = 1 \text{ month}$$

This is a reasonable amount of time as, provided the existance of 4 phases in a year, each phase would last 3 months and using this SW time we're sure that, after a month, a new phase has begun and no more dirty data are considered. In addition, the parameters that we set are as follows: $budgets_{min} = [10, 10, 10]$, $budgets_{max} = [70, 80, 60]$, $step = 2$ and $budget_{tot} = 110$.

2.2.2 Sliding-Window Gaussian Process Combinatorial Multi-armed Bandit

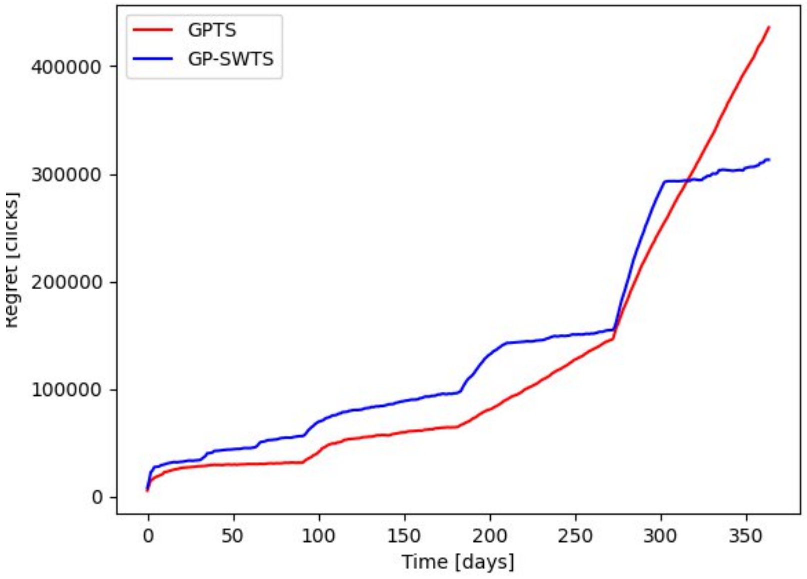
The pseudo code of the algorithm we implemented is as follows.
The update of the GP-Learner's model performed at time τ is achieved by adding the collected reward to the reward space and then discarding the least recent element if the current time exceeds τ.

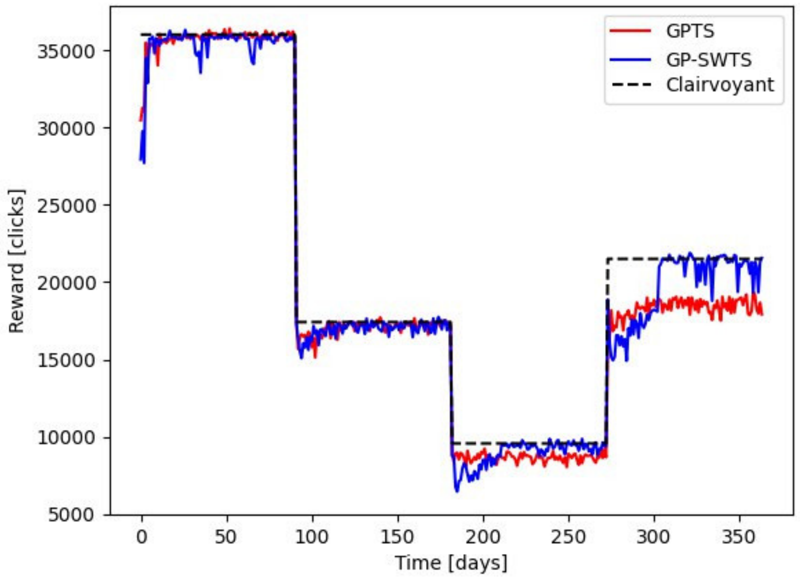
Algorithm 2 Gaussian Process CMAB

```
Input: $\tau$ sliding window size, $J$ classes of users
1: for $day \in \{1, ..., T\}$ do
2:     for $j \in \{1, ..., J\}$ do
3:         Sample j-th GP-Learner($\tau$)
4:         Add row to knapsack matrix
5:     end for
6:     Optimize knapsack matrix
7:     Play selected superarm
8:     Update GP-Learners model($\tau$)
9: end for
```

2.2.3 Results

We compare the behaviours of applying the same GPTS Learner of the previous experiment (this time in presence of all the phases) with the performance of the SW-GPTS Learner.
The experiment shows that the standard GPTS performs better in the first and the second phase of the year. The better performance in the first phase is due to the fact that, while the SW-GPTS always keeps at most 1 month of memory, the basic GPTS keeps collecting samples.
Regarding the second phase we can see that the performance of convergence of the two learners is quite the same. The standard GPTS does not suffer too much the fact that the first phase has ended because it is only a transition between a high interest phase and a low interest phase and therefore the budget allocation does not change much among the classes of users.
The presence of a competitor changes a lot into the budget allocation. It is not enough to allocate the budget with respect to the average client of the brand because now we need to invest more on a class to gain the same results on it, but the budget is limited. A choice is then needed in the third and the fourth phases and this explains the different behaviour.
After about one month of exploration the SW-GPTS is able to converge in this new phase while we see that the basic GPTS will not converge anymore from now on: the sample collected in the preceeding months weights too much on the choice of the budget allocation.
The regret chart that resulted from this experiment and the reward collected for each day are reported in the below figures.


GPTS
GP-SWTS
Regret [clicks]
Time [days]
0
100000
200000
300000
400000
0
50
100
150
200
250
300
350


GPTS
GP-SWTS
Clairvoyant
Reward [clicks]
Time [days]
5000
10000
15000
20000
25000
30000
35000
0
50
100
150
200
250
300
350

2.3 Learning Algorithm for Pricing

In this section, we describe the learning algorithm that we used to maximize the number of purchases made by users that have reached our website by clicking on ads. For simplicity, we will only consider 1 phase and that the allocation of the budget over the three subcampaigns is fixed

2.3.1 Setup of the experiment

Three Thompson Sampling learners are used, one for each class of users. Since the users arrive on the purchase page by clicking on the ads, we assume that we can differentiate them by their class, thus proposing a different price to each user.

Since we are only considering a single phase, the algorithm runs for a limited number of days, 50, enough to provide a significative result.

The average number of users that arrive per day are taken from the results of the previous point, which optimizes the budget in the fourth phase and maximizes the number of clicks. It returns the following average number of daily clicks: $C = [1800, 12000, 350]$. For this experiment, we will use these values as the daily average number of clicks, with a variance that is proportional to each value: $Var_i = C_i/4 \ \forall i \in \{1, 2, 3\}$.

Since the optimization of the budget is made once a day and depends on the price chosen, also the price (for each class of users) is chosen once per day.

The minimum and maximum prices (100, 400) are the same for all the classes of users. The same goes for the number of arms, for which we tested various configurations ranging from 3 to 9 arms to see which one provided the best result.

2.3.2 The algorithm

The high-level pseudo code of the algorithm is shown in Algorithm 3.

Each TS learner is responsible for a class of users. Each day, the 3 learners pull the best price from their beta distributions. The prices are then proposed to all the users that arrive to the website on that day after clicking on the ads. At the end of the day, the learners are updated with the number of successes (purchases) for each class.

Some remarks:

Algorithm 3 TS learners for pricing

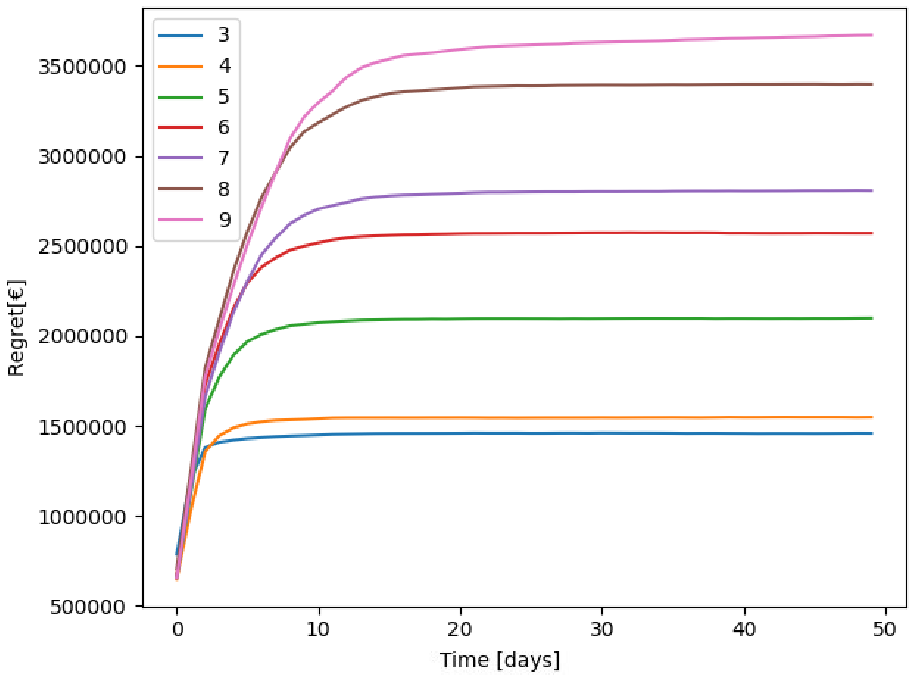
```
 1: $J \leftarrow$ all classes of users
 2: $T \leftarrow$ 45 days
 3: $regret \leftarrow 0$
 4: for $day \in T$ do
 5:     for $j \in \{1, ..., J\}$ do
 6:         $price \leftarrow$Draw a price from the j-th TS learner
 7:         $successes \leftarrow$play the pulled arm
 8:         $failures \leftarrow$clicks[j][day] - successes
 9:         $reward \leftarrow$successes * price
10:         regret += optimum - reward
11:         TS[j].update(price, successes, failures)
12:     end for
13: end for
```

- on line 6, to draw a price means that for each arm a we pull a sample θ_a from the beta distributions, and then return the price p_a of the arm that offers the best reward, that is $p_a = \underset{p_a}{argmax}\{p_a * \theta_a\}$
- on line 11 the update of the TS learner consists in updating the beta parameters of the arm pulled $\bar{a}$, in the following way: $(\alpha, \beta)_{\bar{a}}$ += $(successes, failures)$

2.3.3 Choice of the best arm

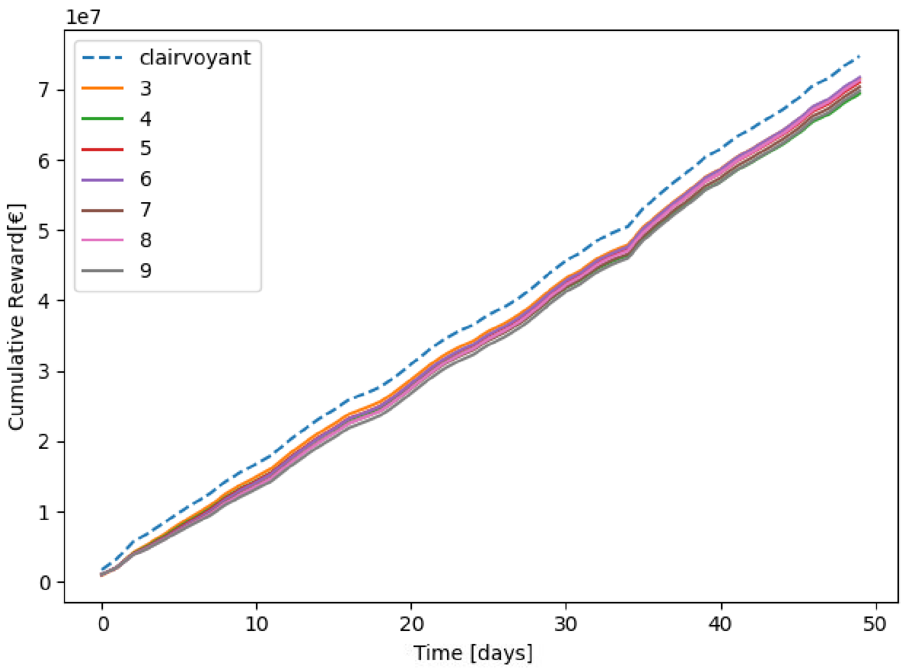
The following results come from averaging over 1000 experiments.

We tested the experiment with several numbers of arms. For each number n of arms, draw the n prices equally spaced in the interval and run the experiment, calculate the regret and the reward and use them to choose the best number of arms.



The first image shows the cumulative regret using different numbers of arms, ranging from 3 to 9. Given the limited time-span, the best result is provided by a low number of arms, which converges more rapidly to the best arm.

However, since in the calculus of the regret we use a discretized clairvoyant algorithm with the same level of discretization of every experiment, we know how fast the learner converges but not how much money we would be losing in the ideal case in which we could use an infinite number of arms. For this reason, we set up another experiment in which we calculate the reward and compare it with the full clairvoyant reward that, in this case, corresponds with the reward that we would get if we chose, for each class, its actual best price without any discretization.



As the image shows, all possible number of arms provide comparable results in terms of cumulative reward. In particular, they are ordered from best to worst cumulative reward in this way: $(6, 3, 8, 5, 7, 9, 4)$. The maximum difference in lost reward (4 arms vs 6 arms) is around 3%.

The fact that 6 is the best number of arms in our experiment does not imply that it is always the case, since we are only considering a single problem and it may be that its arms happened to be close to the best possible prices. However, increasing the number of arms also increases the possibility that an arm is close to the actual best price.

2.4 Context Generator Algorithm

The next few slides focus on a context generation algorithm for the pricing when the budget allocated to each single sub-campaign is fixed. This means that we focus on trying to understand the different behaviours that the three class of users studied may follow. Since the three types of users are different by means of purchasing power, price discrimination may be important to further increase the reward instead of considering the users as a single class. Therefore, our goal is to understand whether or not it can be useful to divide the users in the various classes. From the theory we know that it is worth splitting if the lower bound on the probability that context 1 occurs added to the probability that context 2 occurs is greater or equal to the lower bound on the best expected reward for the context not yet splitted; we test this condition and eventually perform the split every 7 days, that is a week.

2.4.1 Setup of the experiment

The algorithm requires to define some inputs from which the context generation will begin. In particular is important to define:

- **The bandit algorithm to implement:** Empirically the best algorithm to adopt is in almost all the scenarios the Thompson Sampling algorithm, so we opted to use it in order to perform our experiment.
- **The number of arms:** As stated in the chapter before, a number of arms that has a reasonably fast convergence to the best arm and enough arms to pull a price that is close enough to the clairvoyant is 6 arms. Therefore in this scenario we use 6 arms.
- **The number of clicks per day:** The more clicks per day on the ads, the more the algorithm will be able to exploit the disaggregation faster since there will be a huge number of data to work with. We tried to choose a number as close as possible to what it could be a real number of users that click daily on ads like ours. We opted to 17000 clicks per day, based on the based allocation algorithm presented in the chapter before.
- **The time span of the experiment:** It is considered a time span of 90 days because we are considering the experiment to the take place on the first phase (Jan-Feb-Mar).

Since the process involves some randomness, the splits can be done in different order and in different weeks, and the graphs shown as an output may differ. It is important to notice that since there are only 6 arms to explore, the splits always happen in the first weeks, while in a setting with more arms the splits can happen later in time.

2.4.2 The Algorithm

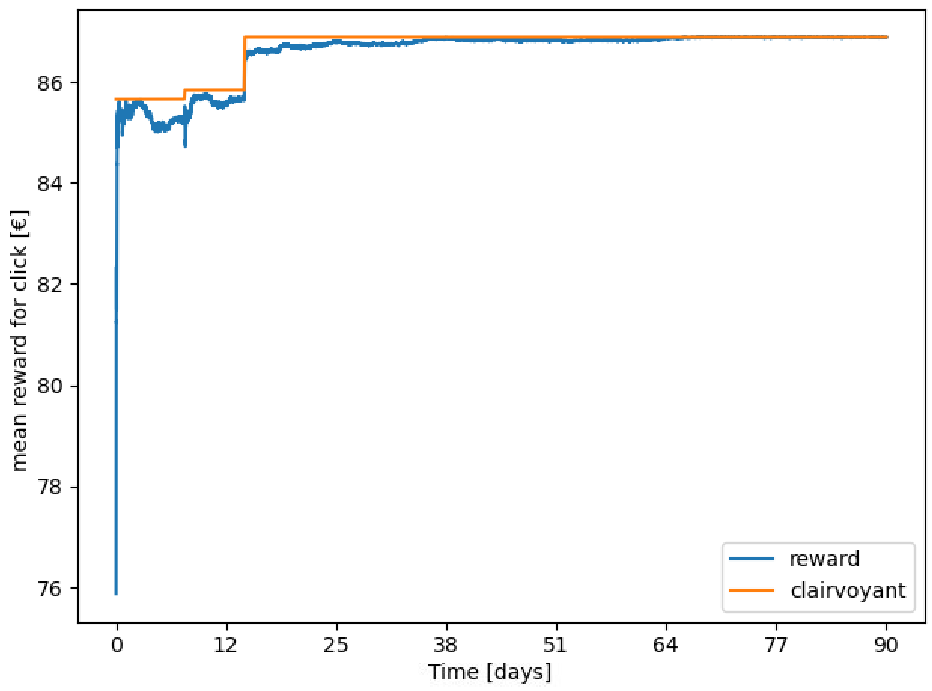
A pseudo code that describes how the algorithm works is this:

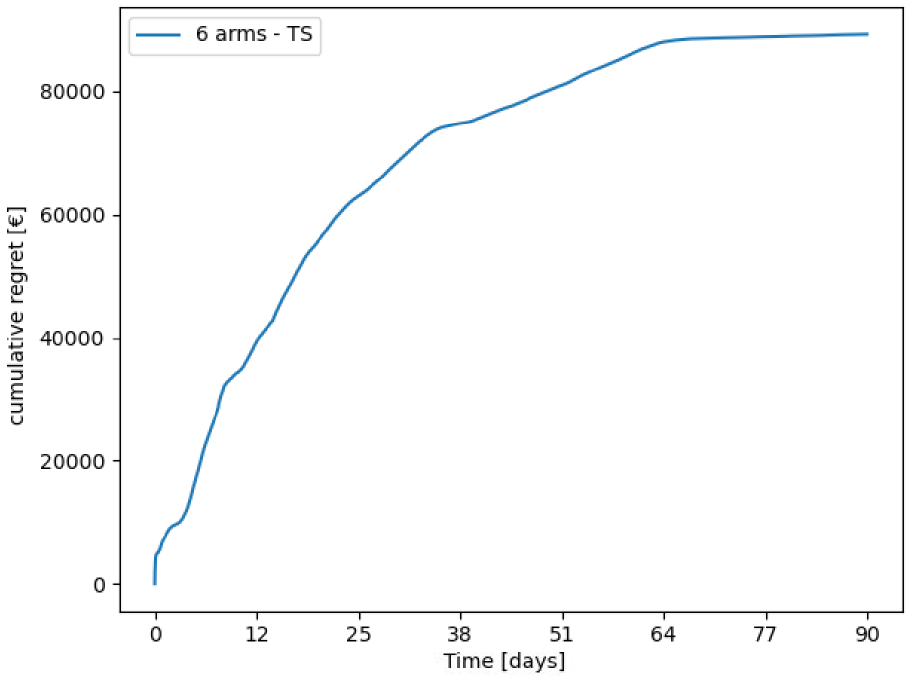
Algorithm 4 Context Generator Algorithm

```
Input: $T$ : time span of the experiment, $C$ : number of clicks per day
1: for $1 \leq t \leq T$ do
2:   for $1 \leq c \leq C$ do
3:     for $context \in Contexts$ do
4:       for $usertype \in context$ do
5:         $sort$ ← Draw the choice of the user from the binomial
6:         $successes$ ← Update the number of successes for that user and context
7:         $failures$ ← Update the number of failures for that user and context
8:       end for
9:       $reward$ ← Update the reward value for the day
10:    end for
11:    $rewards$ ← Append the reward value for the day
12:  end for
13:  if t mod 7 == 0 then
14:    for $context \in Contexts$ do
15:      for $usertype \in context$ do
16:        if split condition achieved then
17:          $c1, c2$ ← Perform the split
18:        end if
19:      end for
20:    end for
21:  end if
22: end for
```

2.4.3 Results

After the 90 days period the graphs of the reward and the regret results as follow:





By watching the clairvoyant plot, it is clear that the splits happened two times (expected since there are three class of users), the first one in the first week of the experiment, and the second one on the second week.
The reward deviates more from the clairvoyant in the initial weeks, when the splits still need to happen and influence the computations of the following weeks, and later converges to the clairvoyant, once the shift to the splitted scenario occurs.
Complementary, the cumulative regret graph shows an exponential and more substantial growth in the first weeks, and later settles after the splits have influenced the reward.

2.5 Optimization Algorithm

After studing the budget allocation and the pricing scenario separately, we imagined the union of the two, used in a real application of the selling of our scarf. Indeed, we designed an optimization algorithm combining the allocation of budget and the pricing when the seller a priori knows that every subcampaign is associated with a different context and *charges a different price for every context.*

The algorithm consists in a Combinatorial-MAB where the combinatorial part is very similar to the knapsack problem, where the value per click used in this problem depends on the pricing, that depends on the number of users of a specific class interested in buying the product.

The pricing and the advertising problem can, in this case, be decomposed since each subcampaign targets a single class of users. In this way, the computation of the value per click of a campaign depends only on the basis of the number of clicks generated by that specific subcampaign.

2.5.1 Experiment setting

As this experiment is substantially the union of the budget and the pricing problem, and given that the problem can be decomposed, no new Learner is implmented here. Instead, a new Environment is created, as this time it should keep memory of the real behaviour of the users based on the budget allocated and of the real behaviour of the users based on the pricing chosen. The core of the experiment here is the set up of the two classes of Learners (GPTS Learner of the budgeting and the TS Learner for the pricing) in a way that they can work together.

Aside from the common variables *TIME-SPAN*, *N-CLASSES* and *N-EXP*, in the main function we had to set up the parameter under our control, that are:

- **Budget bounds**: the variables min-budget and max-budget are arrays that contains the budget bount for the budget problem. In this case, we have chosen to allocate a minimum budget of €1000 to each subcampaign while it can afford to spend more on its top class buyers, respectively €5400 for class 1, €5800 for class 2 and €5200 for class 3.

- **Pricing bounds**: with the algorithm we designed, it will be possible for us to fix a different price for each category of users so it makes sense to give the possibility to the company to set different price bounds for each class of users. In this experiment, we set the same bounds for all the classes €100-€400 but the possibility to change is still possible by setting the parameters of the algorithm.

- **Maxmium budget**: this is the costraint needed for the knapsack-like problem that represent the maximum amount of money that we want to spend. For this experiment this variable is set to €9000.
- **Granularity**: as for the first experiment (the budgeting problem alone), the allocation of budget has a granularity of €200. With the same granularity but different allocation range, each GP Learner resulted in a different number of arms (respectively 22 arms for North/WithChildren, 24 arms for North/WithoutChildren and 21 arms for South/WithChildren).

2.5.2 Gaussian Process Combinatorial Multi-armed Bandit

Algorithm 5 Optimization Algorithm

```
1: $J \leftarrow$ all classes of users
2: for $day \in T$ do
3:     for $j \in \{1, ..., J\}$ do
4:         $p \leftarrow$ Sample(j-th TS)                     // Price
5:         $a \leftarrow$ Get predicted conversion-rate given the pulled arm
6:         $c \leftarrow$ Sample(j-th GPTS)                   // Number of clicks
7:         $b \leftarrow$ Budget(j-th GPTS)                   // Budget spent
8:         $v \leftarrow \frac{p \cdot a \cdot c - b}{c}$      // Value per click
9:         Add row $v \cdot c$ to knapsack matrix
10:    end for
11:    $a \leftarrow$ Optimize knapsack matrix
12:    $rew \leftarrow$ Play selected superarm $a$
13:    Update GPTS-Learners model
14:    Update TS-Learners model
15: end for
```

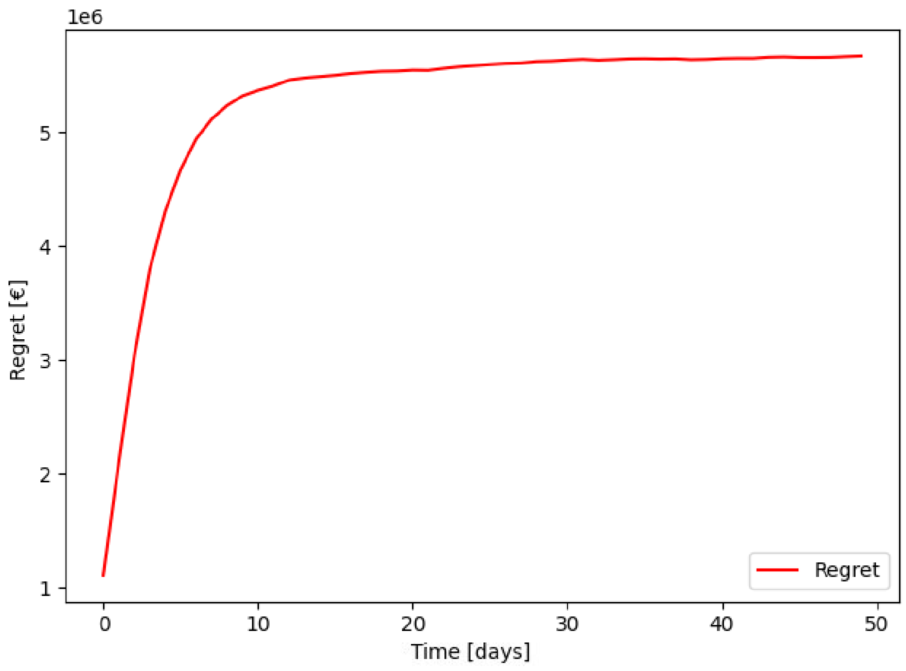
The pseudo code of the algoritm we implemented is as above, where the GPTS-Learners are those incharged of the budget problem and the TS-Learners are incharged of the pricing.

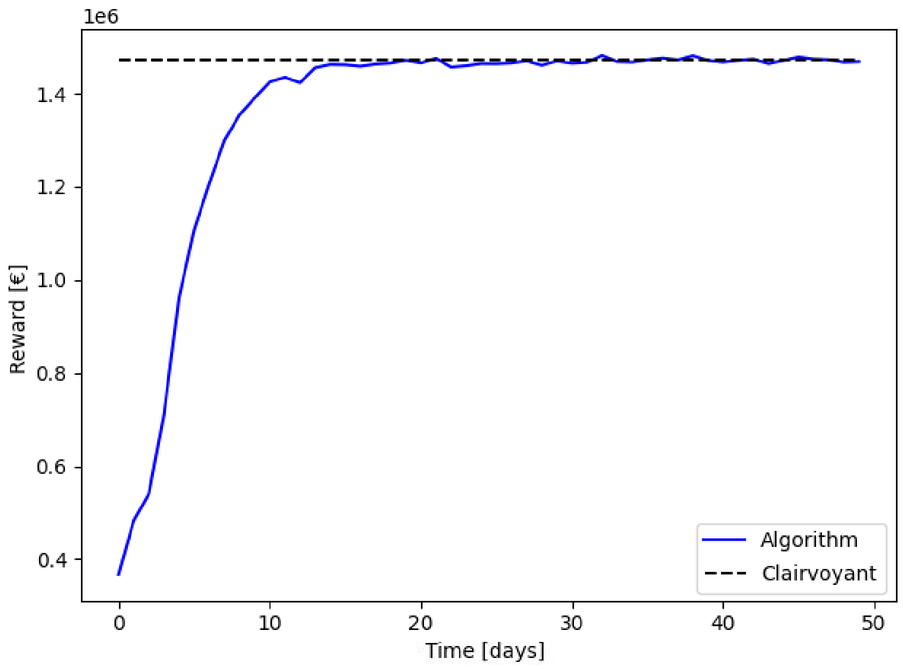
During the model update, an important consideration should also be taken into account: as problem can be decomposed, the optimal solution is the union of the three optimal rewards sub-solution (remember that the seller can set a different price for each class of user).

2.5.3 Results

The experiment shows that after only 10 days of exploration, the corporate could start to exploit good superarms, converging to the optimal one after about half a month.

This result is achieved because the combinatorial problem is solved in an exact way, thus increasing the execution time but lowering the additional regret. This is a reasonanble assumption as the corporate can run the algorithm over the night or in any time with low orders.
The approximation of the solution of the combinatorial problem would have saved just few minutes of computation but would have increased the regret of a non-trascurable amount of money, as a luxury brand cannot afford to sell products even with a sightly wrong price.
The cumulative regret when the algorithm learns both the conversion rate curves and the performance of the advertising subcampaigns is plotted as follows, along with its complementary reward chart that shows the reward at each time $t \in T$.


1e6
Regret [€]
Time [days]
Regret


1e6
Reward [€]
Time [days]
Algorithm
Clairvoyant

2.6 Optimization Algorithm Constrained

In this final step, we will analyze the design of an optimization algorithm combining the allocation of budget and the pricing when the seller a priori knows that every subcampaign is associated with a different context and charges a unique price to all the classes of users. For simplicity, we will only consider a single phase. Also, we assume that the optimal bid is automatically set

2.6.1 Setup of the experiment

For this experiment we will consider a time span $T = 50$ days. Other variables are:

- Total budget b_{tot} that can be used for advertising
- The array of possible budgets B, e.g. the arms of the GPTS learners. (We used the same array for all learners)
- The array of possible prices P, e.g. the arms of the TS learners. (We used the same array for all learners)
- $C = 3$ classes of users
- Maximum budget for each subcampaign: $b_{max_i}, 1 \leq i \leq C$ (the minimum is 0 for all subcampaigns)
- Minimum and maximum prices, that we set equal to 100 and 400.
- θ an array of 3 elements that contains the price pulled by the 3 TS learners on each day
- d_p the estimation of the demand curve at price p provided by the TS learner
- *clicks* is a $C \times |B|$ matrix, in which the element i, j corresponds to the estimated number of clicks for the i-th ad with budget B_j
- *vpc* is a $C \times |B|$ matrix, in which the element i, j is the estimation of the value of one click on an ad of class i if the budget is set to B_j.

- b_{best} is an matrix of size $C \times C$ in which the element i, j represent the best budget allocation for price i and class j, calculated from the CMAB optimizer
- r_{exp} is an array of size C, the i-th element contains the expected revenue of class i
- r_{max} the maximum among all the expected rewards

Also, we will have C TS Learners to learn the best price, C GPTS Learners to estimate the number of clicks as a function of the budget, and a CMAB optimizer to optimize the budget allocation among the subcampaigns.

2.6.2 The algorithm

The high-level pseudo code of the algorithm is shown in Algorithm 6.

Basically, every TS learner is responsible for finding the best price for the class of users it belongs to. However, since only one price can be selected, the algorithm solves the budget optimization problem for each price and chooses the one that has the greatest expected reward. In the budget optimization problem the value per click depends on the price chosen.

Algorithm 6 Budget and Pricing optimization with fixed price

```
1: for $1 \leq t \leq T$ do
2:     $\theta \leftarrow$draw a sample from all TS Learners
3:     for $p \in \theta$ do
4:         $d_p \leftarrow demand(p)$
5:         for $1 \leq c \leq C$ do
6:             for $1 \leq b \leq |B|$ do
7:                 $clicks_{c,b} \leftarrow$Estimate clicks from GPTS learners
8:                 $vpc_{c,b} \leftarrow$Estimate values per click
9:             end for
10:        end for
11:        $b_{best_p} \leftarrow$Use CMAB optimizer to get best budget allocation
12:        $r_{exp_p} \leftarrow$Use CMAB optimizer to get expected revenues
13:    end for
14:    $r_{max} \leftarrow \max r_{exp}$
15:    $(\bar{p}, \bar{b}_c) \leftarrow$Select best price and budgets associated with $r_{max}$
16:    for $1 \leq c \leq C$ do
17:        $(\bar{c}, \bar{b}, \bar{r}) \leftarrow$Test with env and get real clicks, buys and revenue
18:        Update TS and GPTS learners
19:    end for
20: end for
```

Some remarks:

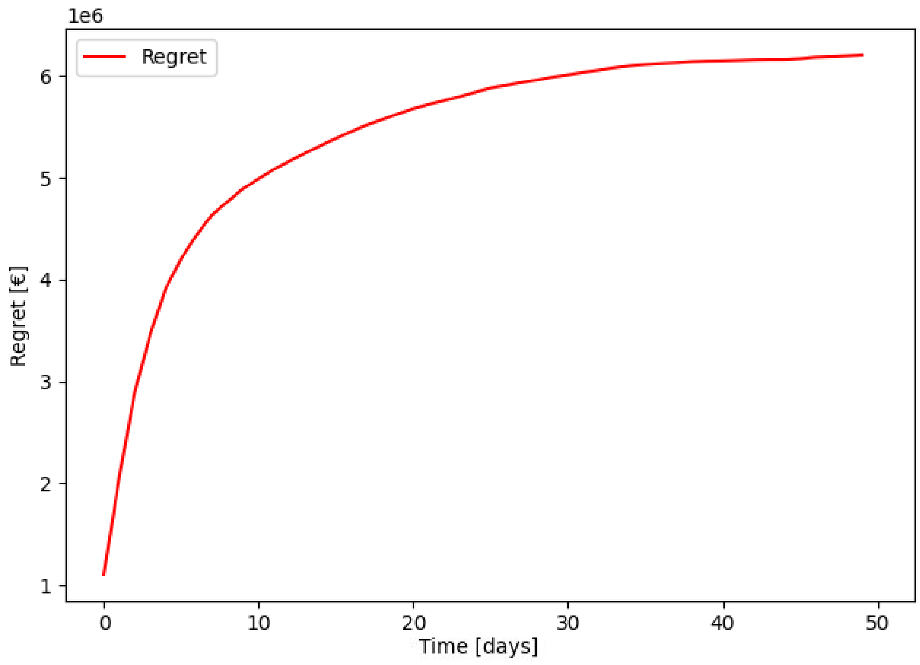
- All the TS learners have 6 arms. As discussed in 2.3.3, this guarantees a fair reward with the given timespan
- The value d_p is estimated from the TS learner, in particular it returns a sample from the beta distribution of price p, in the following way: $d_p = beta(\alpha_p, \beta_p)$
- To estimate the clicks given a class c and a budget b, the c-th GPTS learner returns a sample from the normal distribution of budget b, in which mean and variance are learned by the GP regressor
- the estimation of the values per click is obtained with the following formula:

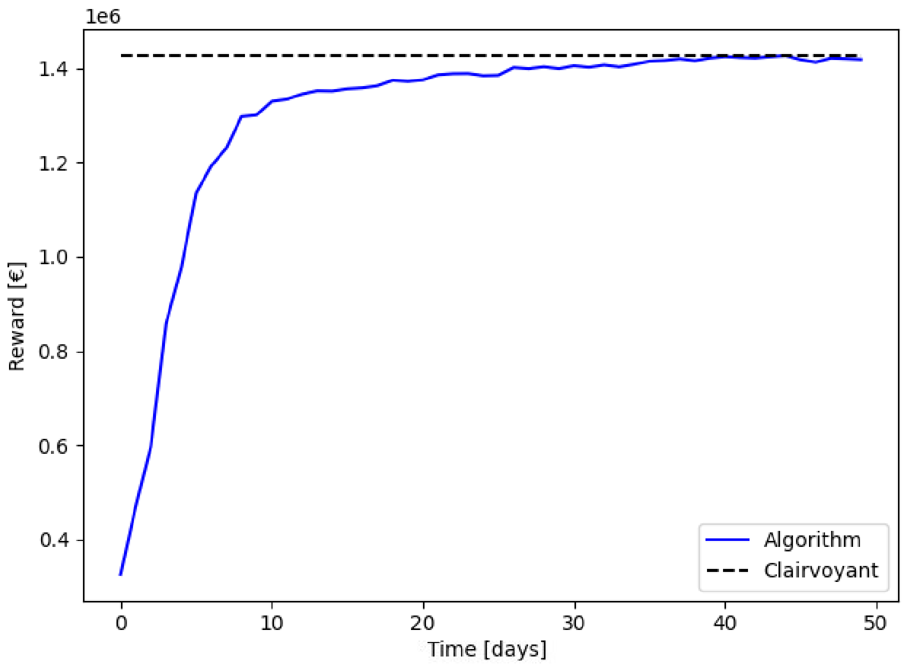
  $$vpc_{c,b} = \frac{p * clicks_{c,b} * d_p - B_b}{clicks_{c,b}} \quad \forall c \in \{1, .., C\} \; \forall b \in \{1, .., |B|\}$$

  in which the numerator represents the net expected revenue (tot revenue - budget spent), and is divided by the expected number of clicks to obtain the values per click, for each class and for each budget
- The CMAB optimizer takes as input the estimated values per click and number of clicks of each subcampaign and returns the best budget allocation by solving the knapsack problem
- In the final part the TS and GPTS learners are updated. In particular, the c-th GPTS takes the number of clicks that the c-th subcampaign received and uses them to update the model on its pulled arm Instead, each TS learner is updated on the same arm $\bar{p}$ but with different inputs: the c-th TS learner updates its arm $\bar{p}$ with the number of buys $\bar{b_c}$ that come from the c-th subcampaign as successes, and $\bar{c_c} - \bar{b_c}$ as the number of failures

2.6.3 Results

After running the algorithm for 50 days with 1000 experiments, we obtained the regret shown in figure 2.6.3. As in the previous points, this regret refers to the best arms, and it represents how fast the algorithm converges to the best solution.


1e6
Regret
Regret [€]
Time [days]
0
10
20
30
40
50
1
2
3
4
5
6


1e6
Reward [€]
Time [days]
Algorithm
Clairvoyant
0.4
0.6
0.8
1.0
1.2
1.4
0
10
20
30
40
50