Project: Pricing & Advertising

Data Intelligence Applications - 5 CFU

Team 20

Battistini Camilla

Di Luozzo Giovanni Antonio

Frigeri Michela

Scaramuzza Riccardo

A.Y. 2020/2021

POLITECNICO
MILANO 1863

Index

1 Assignment

Scenario. Consider the scenario in which advertisement is used to attract users on an e-commerce website and the users, after the purchase of the first unit of a consumable item, will buy additional units of the same item in future. The goal is to find the best joint bidding and pricing strategy taking into account future purchases.
Environment. Imagine a consumable item (for which we have an infinite number of units) and two binary features. Imagine three classes of customers C1, C2, C3, each corresponding to a subspace of the features' space. Each customers' class is characterized by:

- a stochastic number of daily clicks of new users (i.e., that have never clicked before these ads) as a function depending on the bid;
- a stochastic cost per click as a function of the bid;
- a conversion rate function providing the probability that a user will buy the item given a price;
- a distribution probability over the number of times the user will come back to the ecommerce website to buy that item by 30 days after the first purchase (and simulate such visits in future).

Every price available is associated with a margin obtained by the sale that is known beforehand. Do not need to simulate the functioning of the auctions and the other advertisers.

Steps. You need to complete the following steps:

1. Formulate the objective function when assuming that, once a user makes a purchase with a price p, then the e-commerce will propose the same price p to future visits of the same user and this user will surely buy the item. The revenue function must take into account the cost per click, while there is no budget constraint. Provide an algorithm to find the best joint bidding/pricing strategy and describe its complexity in the number of values of the bids and prices available (assume here that the values of the parameters are known). In the following Steps, assume that the number of bid values are 10 as well as the number of price values.
2. Consider the online learning version of the above optimization problem when the parameters are not known. Identify the random variables, potential delays in the feedback, and choose a model for each of them when a round corresponds to a single day. Consider a time horizon of one year.
3. Consider the case in which the bid is fixed and learn in online fashion the best pricing strategy when the algorithm does not discriminate among the customers'

classes (and therefore the algorithm works with aggregate data). Assume that the number of daily clicks and the daily cost per click are known. Adopt both an upper-confidence bound approach and a Thompson-sampling approach and compare their performance.

4. Do the same as Step 3 when instead a context-generation approach is adopted to identify the classes of customers and adopt a potentially different pricing strategy per class. In doing that, evaluate the performance of the pricing strategies in the different classes only at the optimal solution (e.g., if prices that are not optimal for two customers' classes provide different performance, you do not split the contexts). Let us remark that no discrimination of the customers' classes is performed at the advertising level. From this Step on, choose one approach between the upper-confidence bound one and the Thompson-sampling one.

5. Consider the case in which the prices are fixed and learn in online fashion the best bidding strategy when the algorithm does not discriminate among the customers' classes. Assume that the conversion probability is known. However, we need to guarantee some form of safety to avoid the play of arms that provide a negative revenue with a given probability. This can be done by estimating the probability distribution over the revenue for every arm and making an arm eligible only when the probability to have a negative revenue is not larger than a threshold (e.g., 20 %). Apply this safety constraint after 10 days to avoid that the feasible set of arms is empty, while in the first 10 days choose the arm to pull with uniform probability. Do not discriminate over the customers' classes.

6. Consider the general case in which one needs to learn the joint pricing and bidding strategy under the safety constraint introduced in Step 5. Do not discriminate over the customers' classes both for advertising and pricing.

7. Do the same as Step 6 when instead discriminating over the customers' classes for pricing. In doing that, adopt the context structure already discovered in Step 4.

Duties. You are required to:

- Produce the Python code.
- Produce a technical report describing the environment, the algorithms and the plot representing the regret and the reward of every algorithm. Provide also a practical application motivating the scenario.
- Produce a presentation as a summary of the technical report.

2 Introduction

This report refers to the project about Online Pricing & Advertising carried out during the course of Data Intelligence Applications. This introductory section is meant to specify the problem framework, the agents, the scenario in which they're acting, the notation followed in the paper and the hypotheses taken.

2.1 Scenario Description

We imagine to be hired by an online seller, VegStrenghty, which acts in the market of protein and vitamin shakes. It was born to provide an aid to vegan people who necessitate to integrate their diet, but has established himself as a good competitor in the field of Food Supplements, being appreciated also by non-veg people and athletes. In particular, they've come up with a new product, the AllMightyShake©, which is supposed to match the desires of all class of possible customers. Due to this, the company estimate a significant cut in the production costs thanks to scaling economic factors, which could lead to a selling price able to conquer the market (Prices considered will not exceed 25 €/Unit).

Our task is to propose a pricing strategy for this new product, in order to maximize the revenue due to users visiting a specific section on the company website. Thanks to the stakeholder prior knowledge of the market, we can imagine that the totality of new users visiting the page each day is retrieved thanks to Social Advertising (in particular, Instagram Stories Ads), this resulting in the necessity of proposing also a joint strategy for bidding the right Ads. [1]
Due to the product nature, a recurrent purchase mechanism need to be taken into account: once a user decides to buy the product, he will very likely come back to replenish its reserves periodically, within a customer-specific period.
Thanks to market investigation and surveys, the Stakeholder was also able to identify two binary features, a Physical Activity Profile and a Diet Profile, which are believed to determine possible differences in customers behaviour: as each class is likely to show specific dynamics and characteristics, this will require to take into account this partitioning at some point of the analysis.
Last assumptions to highlight are that we are neither required to deal with market competitors (Monopoly Assumption), nor with possible interactions with other products sold by the company (Single Product Assumption).

[1] Notice that the modeling of the ads auction is not required to be implemented and will be treated as a black box

2.2 Notation

In this section we go deeper into mathematics, specifying all the entities involved in the problem modeling and providing names with which we will refer to them all along the paper.

Optimization Variables

We will consider as variables of the optimization problem the following quantities:

- The **Price** p [€/unit] represents the price to the public of each unit of product and it is assumed to take values between 15€ and 25€
- The **Bid** b [€/click] specifies the company offer to the ADs Auctioner and will influence the number of new customers of the website. We considered the bid to take values between 0.5€ and 2€.

It is important to notice that both price and bid are supposed to be constant during each day (round). Actually this is a bit unrealistic assumption, since Ads Auctions take places repeatedly along the day (and usually with high frequency). However, this simplification is not restrictive, since in principle we can apply the implemented methods without any change, just considering a round as the time between one auction and the next one. [2]
Notice that price and bid values are assumed to take values in the reported ranges according to market dynamics and stakeholders' directives.

Functions of the Price

We define the following quantities that will vary with respect to the chosen price:

- The **Marginal Revenue** $MR(p)$ [€] is considered to be fixed and known, describes the profit of the company accounting for production costs.
- The **Conversion Rate** $CR(p) \in [0, 1]$ is the probability that a new user will finalize a purchase, given a certain price, after visiting the website.

Generally, in this kind of applications marginal revenue is influenced by variable costs, but in this case it is supposed to be known beforehand and will be modeled as a deterministic function of the price. The conversion rate curve instead is one of the key object that will be implicitly estimated during the learning phase.

[2] This assumption is taken because, working on simulated data, the computational cost would have been too high

Functions of the Bid

We define the following quantities that will depend upon the chosen bid:

- The **Cost per Click** $CPC(b)$ [€/click] is a stochastic function of the bid that specifies the cost per click obtain after the ADs auction.
- The **Number of New Users** $NDC(b)$ [users/day] is a stochastic function of the bid that defines the daily number of new users visiting the website after clicking the Ad.

Both the Cost per Click and the Number of New Users functions are required to be (implicitly) learned during the process.

Random Variables

We define a random variable R [times/month] expressing the number of times a user will come back within 30 days after the first purchase. Being a random variable, it will need to be modeled through a proper distribution, in order to account for its effect in the profit maximization problem.

Customers Classification

As already mentioned, the potential customers can be partitioned according to two binary features:

- Diet Profile: High for people which will likely requires a high amount of supplements due to their diet choices (e.g. vegan and vegetarian people), Low otherwise
- Physical Activity Profile: High for people practicing particular sports (e.g. bodybuilding, calisthenics, semi-professional athletes), which will likely requires supplements due to their particular needs, Low otherwise.

however, the true underlying segmentation is composed by only three classes, as shown below:

		DIET PROFILE	
PHYSICAL ACTIVITY PROFILE		*LOW*	*HIGH*
	LOW	C1	C2
	HIGH	C3	C3

Figure 1: Customers classification

This partition is important to be considered as it is likely to define different behaviours in term of Food Supplements needs. In particular, needs of each class will be different (low for C1, high for C2, very high for C3), and that will be taken in account in the design phase of the different curves used for Data Generation.
The proportions of each class with respect to the whole population (which are in principle unknown) are, respectively, 20%,50% and 30% for class C1, C2 and C3.

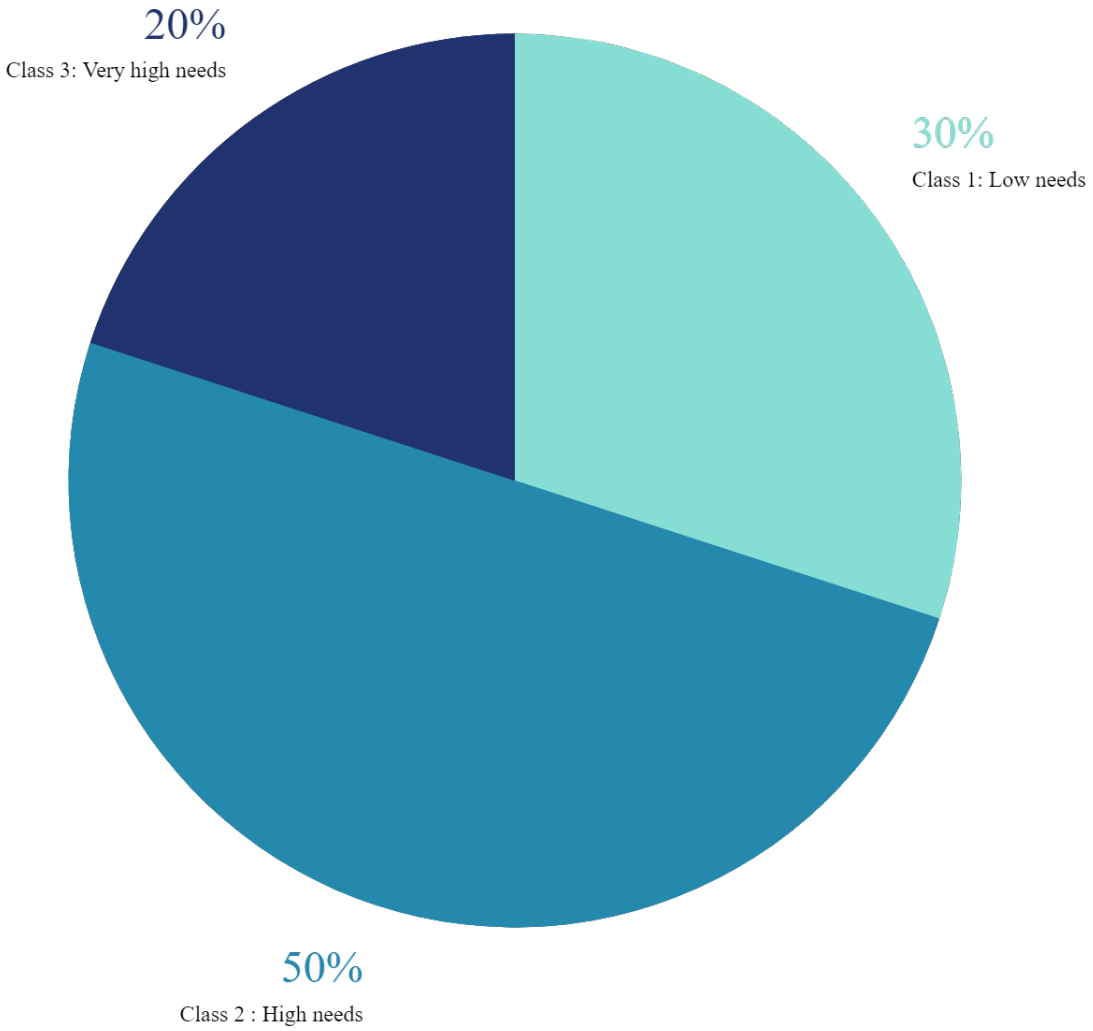

Figure 2: Customers class proportions

2.3 Simulated Data

In order to be able to perform experiments and evaluate performances of the developed algorithms, we will rely on simulated data. In this section we will design and describe:

- the Marginal Revenue Function
- the Conversion Rate Functions characterizing each class
- the probability distribution over the number of returns within 30 days after the first purchase
- the Cost per click stochastic function per each class
- the Number of daily clicks function per each class

Alongside the curves per each class, we also provide the relative functions for aggregated data. They are obtained through a mixture distribution approach, considering as weights the proportion of each class with respect to the whole population.

Marginal Revenue Function

We already have defined a range of possible prices to the public, namely between 15 and 25 €/unit. We consider different possible candidate shapes for the daily Marginal Revenue function. Firstly, we assume to have to take into account the presence of fixed costs deriving from manufacturing and materials. Hence, we defined two expressions for the Marginal Revenue considering, respectively, a fixed percentage and a fixed amount to be detracted from the daily income. Despite this first approach may seem reasonable, in real applications margin functions are not linear with respect to the price, but instead the profit decreases as the price of an item rises. This is due to the fact that highly priceable items requires, usually, additional cost to be developed, such as market research, product design, etc. Considering this phenomenon, we built a scaling Marginal Revenue function as:

$$MR(p) = p - 5 - \frac{1.5}{100} * p^2 \tag{1}$$

In order to be as realistic as possible, we choose this third alternative to describe the behaviour of the Marginal Revenue.
Below are reported the three functions proposed, considering the first two with a fixed percentage of costs of 30% and a fixed detracted amount of 5€, respectively.

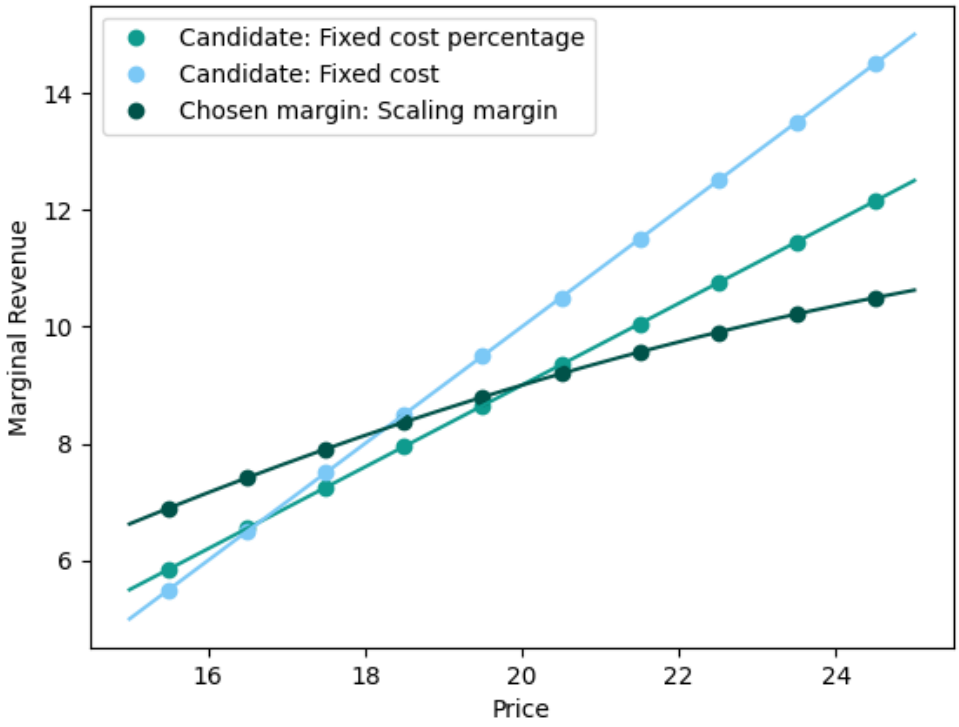

Figure 3: Marginal Revenue function design

Conversion Rate Function

The design of Conversion Rate curves starts from some realistic considerations about the behaviour of the three classes of customers; in particular, we discretize the price domain and assign to each point a Conversion Rate value according to some assumptions:

- customers with low needs (class C1) are more likely to finalize their purchase if the price is low, as they're not expert in the field.
- high needs customers (class C3) are likely to have already integrated some kind of supplement in their diet, so they may consider a cheap price indicative of a not so good product.
- subjects belonging to class C2 follow a behaviour in-between the two above

Finally, to obtain the curves we interpolate the chosen values. As a plain linear interpolation results in little too rough curves, we decided to adopt cubic splines interpolation (see Figure 4 and 5). Notice that the functions are purely deterministic of the price.

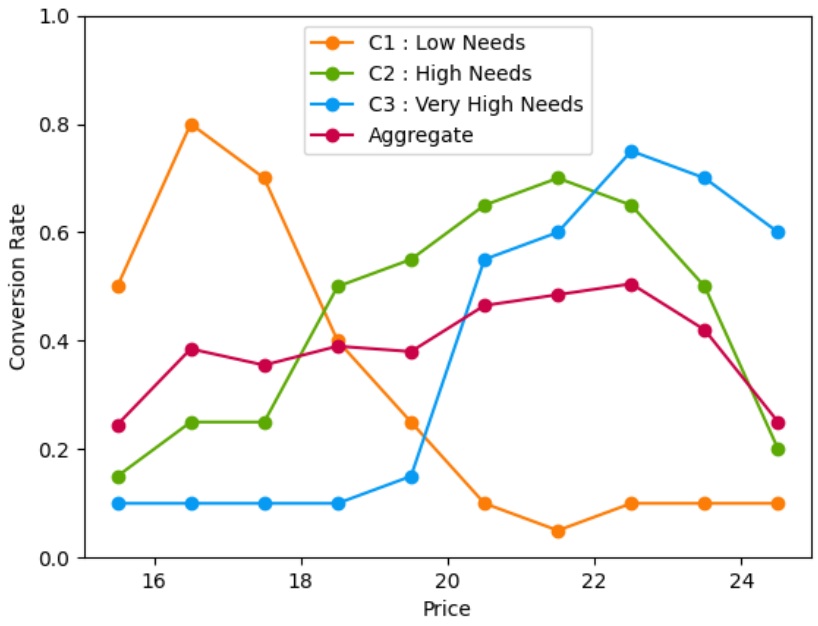

Figure 4: Conversion Rate Functions - Linear interpolation

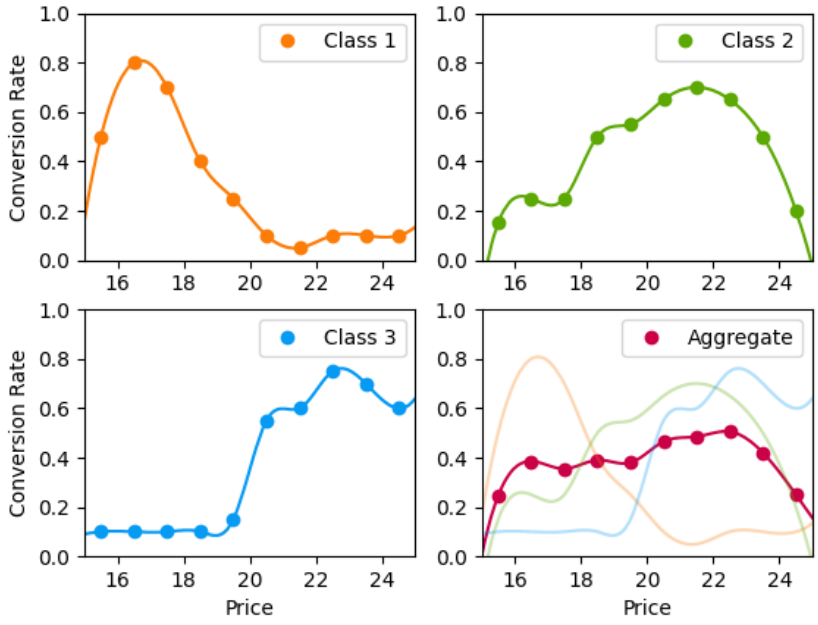

Figure 5: Conversion Rate Functions - Cubic splines interpolation

Customer Returns within 30 days

We model the random variable of Returns within 30 days from first purchase as discrete, following a distribution whose support is 0,1,2,3. The probabilities are assigned, also in this case, considering that any is likely to buy again the product proportionally to his/her needs (and so, to his class). The chosen distributions are shown in Figure 6.

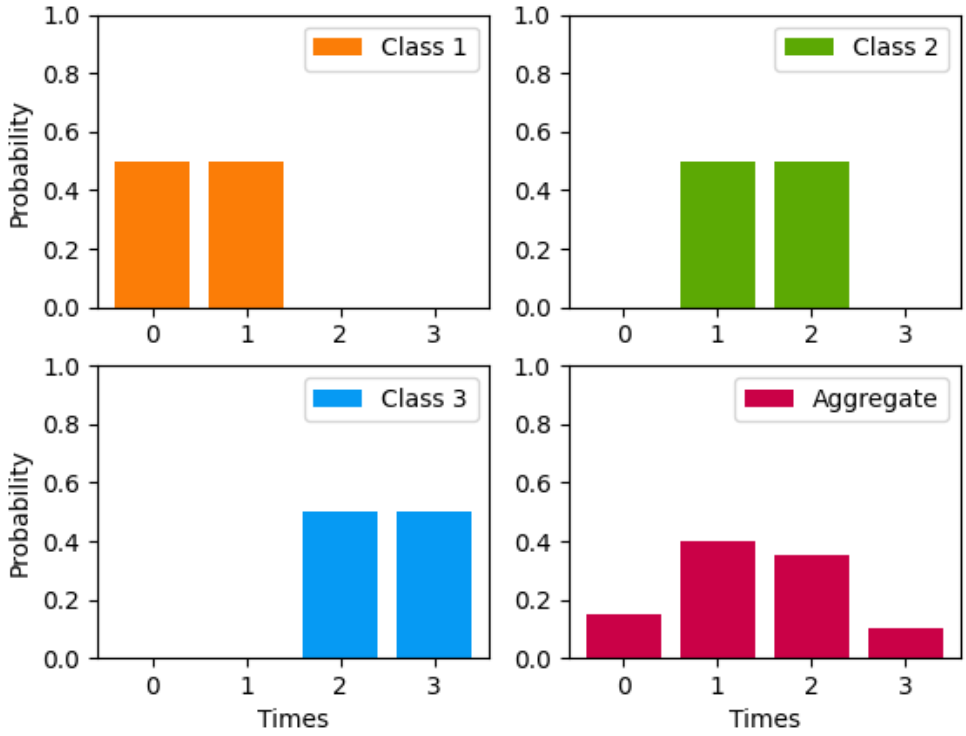

Figure 6: Number of Returns in 30 days distributions

Cost per Click Stochastic Function

We follow a similar approach as in the deterministic case. We discretize the bid space and choose corresponding values following some assumptions:

- Class specific mean levels are considered (non-decreasing, piece-wise linear)
- Saturation may happen both at small and high values of bid

then, we interpolate those values.At this point, we add a zero mean Gaussian noise to introduce stochasticity:

$$CPC_i(b) = CPC^*(b) + \epsilon_i \tag{2}$$

$$CPC^*(b) : [0.5, 2] \longrightarrow R \tag{3}$$

$$\epsilon_i \sim N(0, \sigma_i^2) \tag{4}$$

Notice that the error is designed to have a class-specific variance. The Cost per Click Function need to be stochastic to take into account the fact that the ads auction mechanism is considered as a black box.

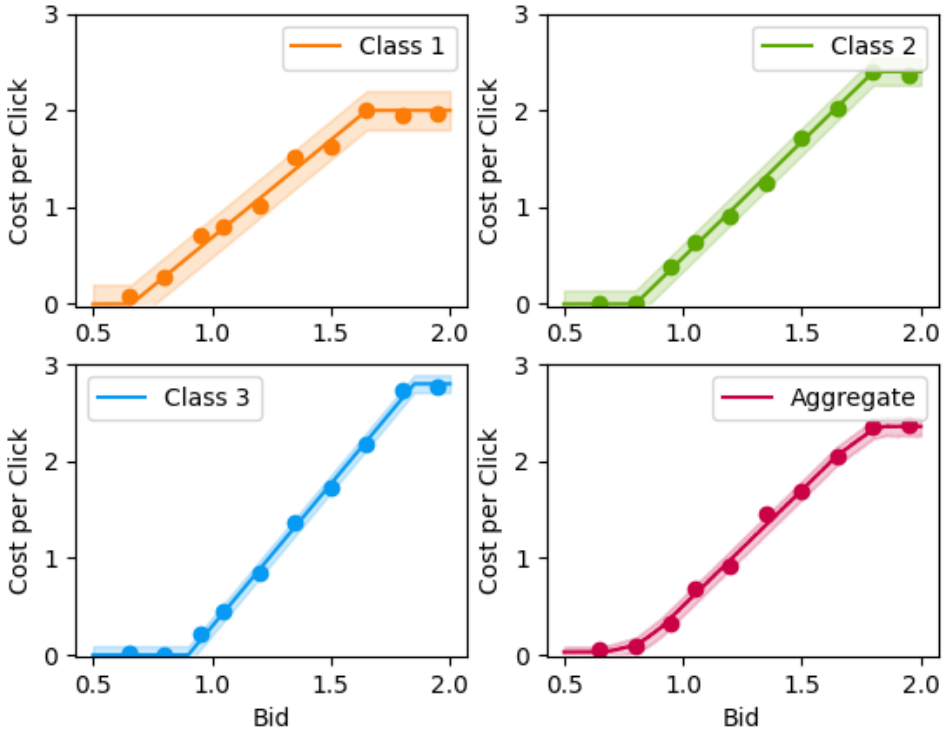

Figure 7: Cost per Click Functions

Bold points are realizations sampled from the obtained stochastic function.

Number of Daily Clicks Stochastic Functions

The approach followed in the design phase is the same adopted for the Cost per Click Functions. The only actual difference is that the function domain is the set of natural numbers. The assumptions taken for the deterministic baseline functions are:

- too low values of the bid may results in winning no auction, and thus to zero customers
- the function is increasing in the value of the bid, since better ads may be obtained. However, the effect loosen for high values of the bid.

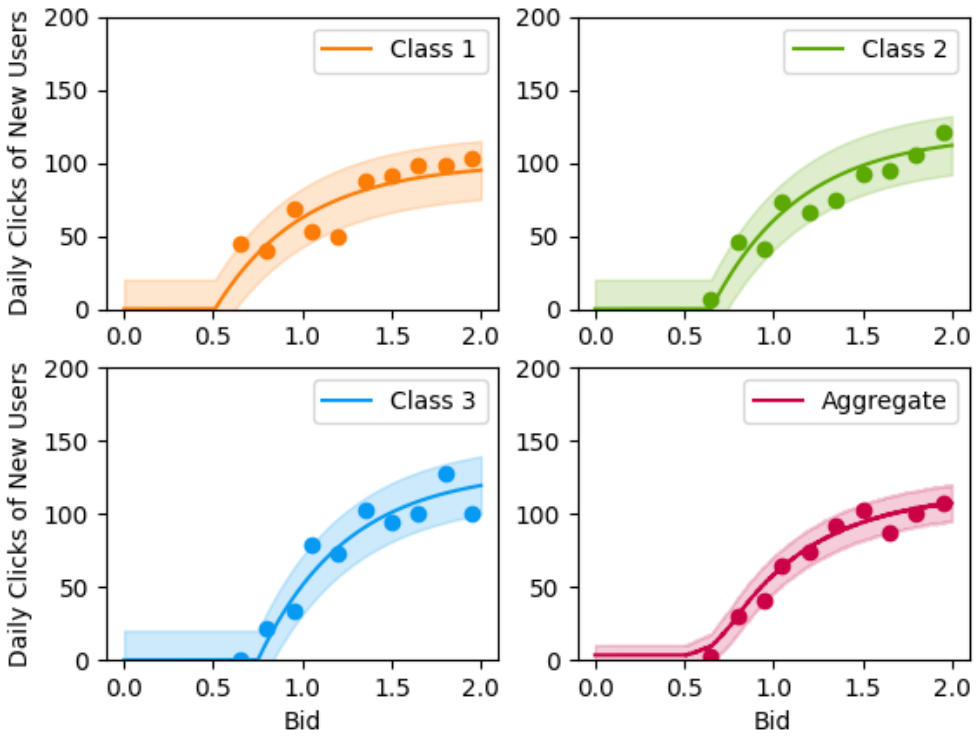

Figure 8: Number of Daily Clicks Functions

A note on Stochastic Functions in the Aggregated Case

Since the Aggregated functions are retrieved through a mixture approach, it may be difficult to properly characterize the variance. Moreover, in the NDC case, the capping of realizations to unity is performed a posteriori, thus the deterministic baseline function does not represent the Mean value anymore. For this reasons, we estimate the Mean Function and Variance Function of CPC and NDC in the aggregated case through a Monte Carlo simulation.

3 Optimization Problem

In this section we tackle the optimization problem from an offline perspective, assuming to know each parameter, function and distribution involved. This is done to provide a method to define the actual optimum values (those provided by a clairvoyant algorithm), in order to evaluate the performances of our algorithms.
In the following are thus provided a brief recap of the variables considered, the definition of the target function to be optimized, the algorithm used to solve the problem and the final results.

3.1 Problem formulation

The variables involved in the problem are:

Variable	**Description**
t	time instant
T	length of observation period
i	customer class
K	total number of customer classes
$b_{i,t}$	proposed bid for class i at time t
$p_{i,t}$	proposed price for class i at time t
NDC_i	stochastic number of clicks of new users of subcampaign i given $b_{i,t}$
R_i	stochastic number of purchases within next 30 days of users of class i given $b_{i,t}$
CR_i	conversion rate for class i given $p_{i,t}$
MR	marginal revenue for price $p_{i,t}$
CPC_i	stochastic cost per click of subcampaign i given $b_{i,t}$

Due to the fact that we must deal with stochastic functions (NDC and CPC) and random variables (R), the Revenue function to be optimized need to be defined in terms of their mean value.
Moreover, notice that, in principle, the proposed bid and price can be different in each day. However, since we assume all the distributions to be stationary along time, we can tackle the problem in a daily perspective, identifying the optimal bid and price to be played each day till T ($p_{i,t} = p_i$ and $b_{i,t} = b_i \ \forall t$).
Thus, our objective is:

$$\max_{p_i,b_i} \sum_{i=1}^{K} \mathbb{E}[NDC_i(b_i)] \cdot CR_i(p_i) MR(p_i)(1+1/30 \cdot \mathbb{E}[R_i]) - \mathbb{E}[NDC_i(b_i)] \cdot \mathbb{E}[CPC_i(b_i)] \quad (5)$$

where each expected value refers to the relative random variable distribution. [3]

[3] Recall that, for fixed values of the bid, $NDC(b)$ and $CPC(b)$ are actually random variables

The coefficient 1/30 multiplying the expected number of future purchases is considered in order to scale their contribution daily.
This formulation allows to solve the generic problem, in which both contextual advertising and pricing are considered for each underlying class of customers. However, we are actually required to consider slightly different frameworks: the first account only for aggregated data, neither applying a contextualization in pricing and bidding; the second, instead, consider a custom-pricing strategy for each class but a single advertising strategy based upon aggregated data. In both cases, the formulation is similar to the one above, in which the characteristics functions and variables have been replaced.

$$\max_{p,b} \mathbb{E}[NDC_{agg}(b)] \cdot CR_{agg}(p) MR(p)(1 + 1/30 \cdot \mathbb{E}[R_{agg}]) - \mathbb{E}[NDC_{agg}(b)] \cdot \mathbb{E}[CPC_{agg}(b)] \tag{6}$$

$$\max_{p_i,b} \sum_{i=1}^{K} \mathbb{E}[NDC_{agg}(b)] \cdot CR_i(p_i) MR(p_i)(1 + 1/30 \cdot \mathbb{E}[R_i]) - \mathbb{E}[NDC_{agg}(b)] \cdot \mathbb{E}[CPC_{agg}(b)] \tag{7}$$

3.2 Brute Force Approach

To solve the above problems, we resort to a Brute Force approach. In particular, candidates values for bid and price are provided a priori in the respective range of interest to build a grid, and then each couple is evaluated through the revenue function previously defined in order to find the best one. In this case the algorithm finds the optimal solution with time complexity directly proportional to the product of the number of candidates for bid and price(s). Following the assignment, we will consider 10 candidates both for the price(s) and the bid, but in principle an exhaustive grid search can be performed. In the following are reported the solutions of the aggregated case (and, for completeness, of each underlying class), as well as the respective revenue surfaces. [4]

[4]The solution of the contextual pricing problem is reported in section 6

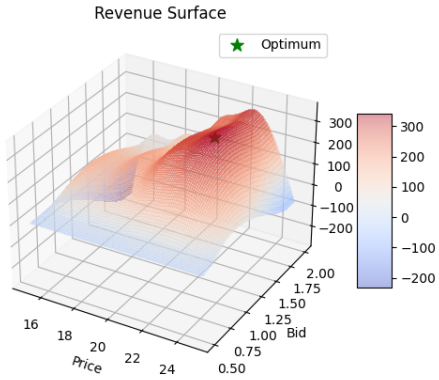

Figure 9: Aggregated

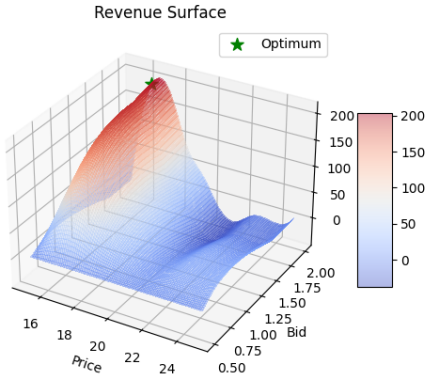

Figure 10: Class 1

Maximum expected revenue	337.91€
$p_{agg}*$	22.5
$b_{agg}*$	1.35

Maximum expected revenue	380.56€
p_1*	16.5
b_1*	1.95

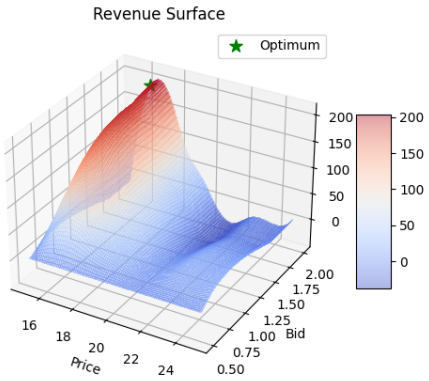

Figure 11: Class 2

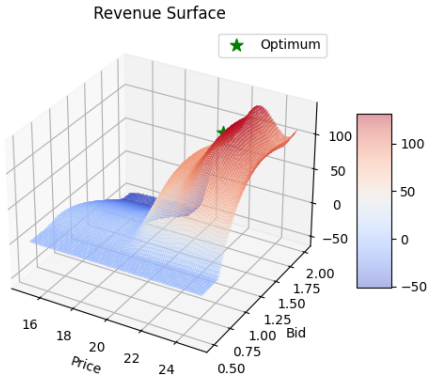

Figure 12: Class 3

Maximum expected revenue	524.83€
p_2*	21.5
b_2*	1.5

Maximum expected revenue	634.28€
p_3*	22.5
b_3*	1.5

4 Online-Learning Optimization Problem Formulation

So far we have described the general scenario of our research and how we designed the Data Generative functions used to simulate it. However, the optimization problem can't be tackled assuming to know them, as done in the previous section: in the real case framework, an online learning approach is required. This means we aim at estimating the best values for the price and the bid by observing values sampled from an environment, which simulates reality. It works upon our Data Generative functions, but only inputs and outputs are assumed to be observable.
The Random variables involved are the same as above, but instead of knowing their laws, we can only observe their realizations:

- **Random Variables:**

CPC	cost per click (fixed the bid)
ACC	draw from a Bernoulli of mean $CR(price)$
NDC	number of daily clicks (fixed the bi)
R	number of future purchases within 30 days

  ACC is a Bernoulli distributed variable, whose mean is given by the Conversion Rate function at the given price (which is hidden and need to be learned), and represent whether a customer finalize his purchase or not.

- **Potential delays in the feedback:** at run time, each variable realization is fully observed by the end of the day (round), with the exception of the Number of Returns within 30 days. To actually know the true value of this variable, we should wait 30 days after the first purchase by the specific customer. Instead of considering each datum available only after a month, we decided to estimate mean returns, at a given time t, with all available information at the considered time, also considering incomplete observations. This is done assigning each customer a counter which is updated daily, and incremented each time a returns happens. Notice that, after 30 days from the first purchase, the datum is fully observed and no more updated.

- **Time horizon:** The time horizon is fixed to be one year in which every round corresponds to a single day.

Our approach to solve the online optimization problem resort to Multi Armed Bandits algorithms.
Generally, MAB algorithms work minimizing the regret, defined as the cumulative difference between the reward of the clairvoyant algorithm, which always chooses the

optimal arm, and the reward given by the arm which we choose at a specific round. Each arm, in this case, represents a bid-value couple, which is associated to a (in principle stochastic) Expected Daily Revenue value, defined as in the offline optimization problem formulation. The mathematical formulation is the following:

$$\min T \cdot \mu * - \sum_{t=0}^{T} \mu_t \tag{8}$$

$$\mu* := \max_{p,b} DailyExpectedRevenue(CR(p), NDC(b), CPC(b)) \tag{9}$$

$$\mu_t := DailyExpectedRevenue(CR(p_t), NDC(b_t), CPC(b_t)) \tag{10}$$

where p is a scalar or a vector whether contextualization in pricing is considered or not.

5 Fixed Bid & Online Pricing

In this section we address a first simplification of the general online optimization problem. In particular, we consider the aggregated case, thus no contextual pricing is applied. At this stage we consider the value of mean returns to be known, even if it is required to be estimated in a real case scenario. This choice was taken due to the fact that, in an aggregated framework, it only affects the scale of the revenue function, but does not influence the choices taken by the proposed algorithms [5]. Moreover, the value of the bid is considered to be fixed at his optimum value, found in the previous step.

5.1 Environment

The environment in which the learners are acting is simply modeled by a Binomial sampler, used at each round (day) to simulate the acceptances of the customers visiting the website. It does that according to the price proposed, following the aggregate case Conversion Rate function. Notice that, actually, what the learners register is the Revenue at the end of the day: that is obtained passing the sum of the Bernoulli realizations to Equation 6 (where the bid-dependent stochastic quantities and R are set to their optimum value).

5.2 Learner

We consider, as said, a MAB setting with ten candidate prices in the range of interest. In particular, we implement two learners, relying on Thompson Sampling and UCB1 algorithms, and evaluate their performances. Notice that both algorithms are straightforwardly modified to account for a Binomial reward instead of a classic Bernoullli one.

TS - Thompson Sampling

The functioning of the algorithm is the same as in the Beta-Bernoulli case; unique difference is that the prior parameters are updated, at the end of each round, using realizations from a Binomial. Calling, respectively, r and n the number of positive realizations and of total customers:

$$\begin{cases} \alpha = \alpha + r \\ \beta = \beta + (n - r) \end{cases} \tag{11}$$

[5] The problem with aggregated data and mean returns estimation is nevertheless solved as a preliminary step of Section 6

UCB1 - Upper Confidence Bound 1

As before, the implemented algorithm works exactly as the classical one, but performs the update of parameters (number of positive samples and total samples per arm) necessary to compute the upper Hoeffding bound in a batch manner.

5.3 Perfomance Evaluation

In general, to evaluate the perfomances of algorithms in each task we resort to a standardized procedure. It consists in repeating 100 times a one year simulation, in order to obtain, through averaging, a reliable estimate of two indicative quantities:

- the **Cumulative Regret** along the year, to identify the better performing algorithm when comparing more than one;
- the **Mean Daily Expected Revenue** along the year, to understand if the algorithm actually converges to the optimum value.

Focusing on the task of Pricing in the aggregated case with known value of the optimal bid, the TS algorithm shows a significantly better perfomance than its deterministic counterpart, as testified in Figure 13:

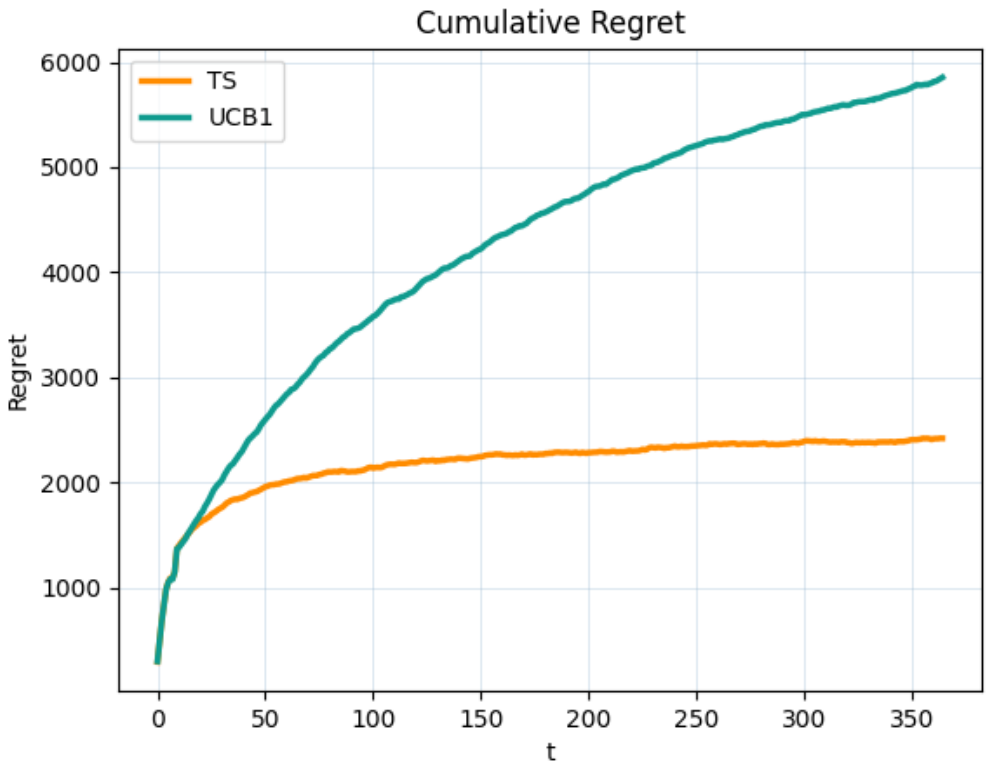

Figure 13: Cumulative Regret

Both the Learners, actually, converged pretty fastly, with the TS-based one achieving a perfect convergence to the optimum value:

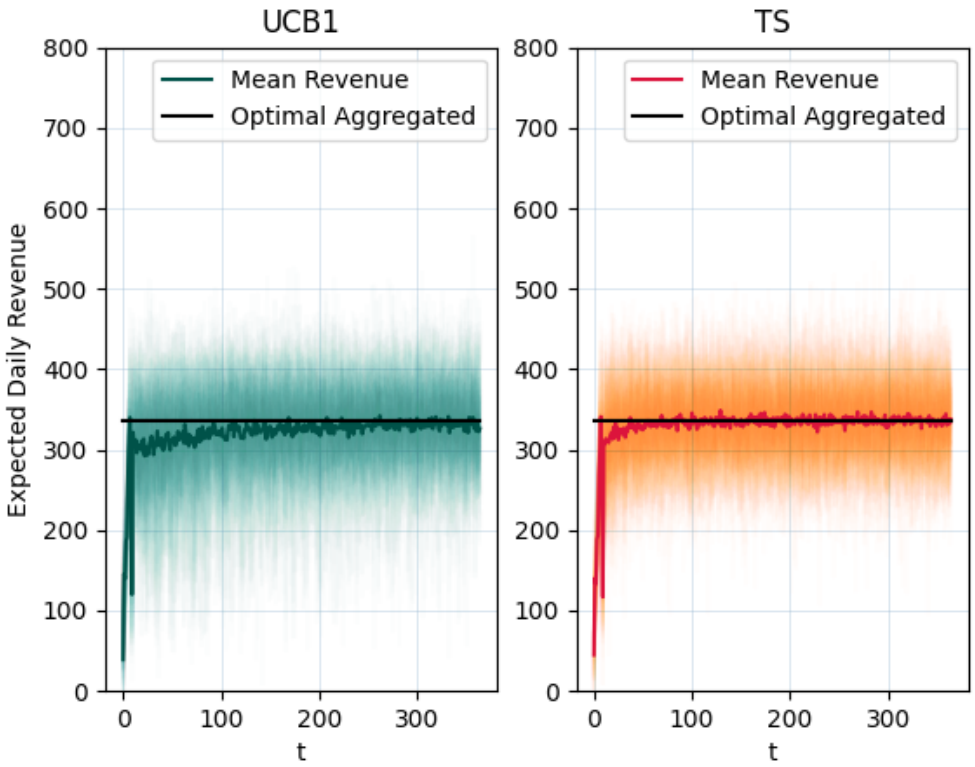

Figure 14: Estimated Expected Daily Revenue during time

The fact that the UCB1-based Learner seems to reach a slightly suboptimal value is probably due to the fact that it keep choosing (not often, but it happens) subotpimal arms (associated to values similar to the optimal one) along the year; instead, TS starts to play only the optimal arm very early. This was inferred looking at the choices history of the algorithms in some of the simulations; as an example, here is reported the one relative to Experiment 1:

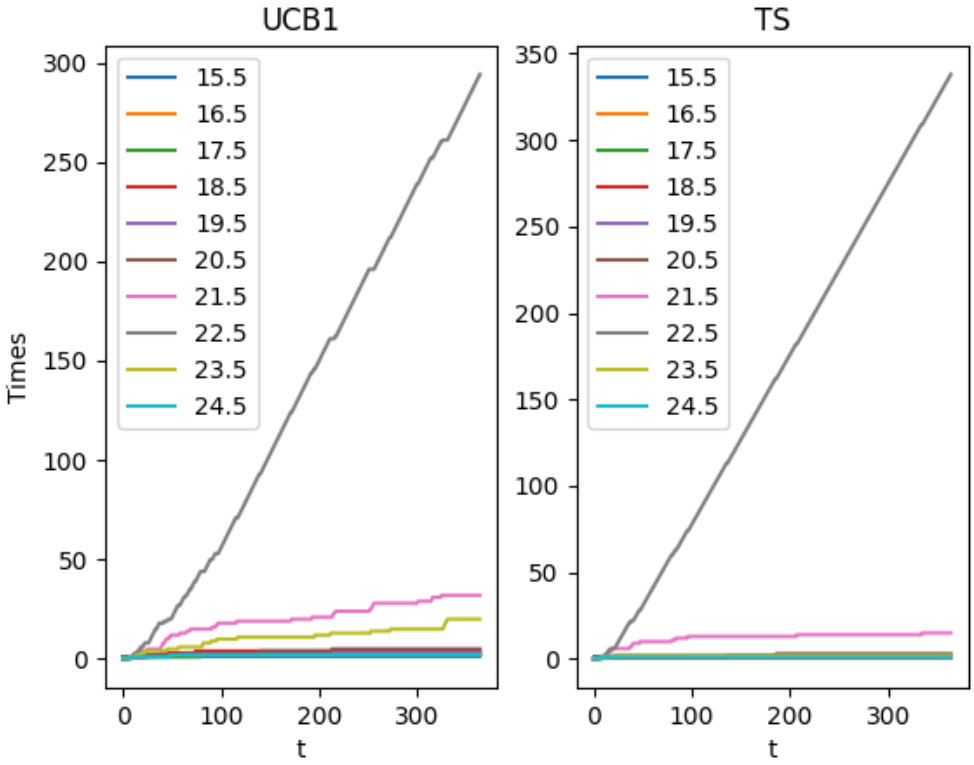

Figure 15: Arms chosen across time - Experiment 1

6 Fixed Bid & Contextual Online Pricing

In this section we show the implementation and performances of the algorithms considered to tackle the problem of online contextual pricing. As in the previous step, the bid is imagined to be known and fixed at his optimal value during the whole process, allowing us to focus on the learning of the contextual pricing strategy. The problem is non trivial, since we must provide an extension of a pricing algorithm able to learn the most promising context (the segmentation of the customers space according to provided binary features) and also to estimate the number of purchases following the first one per each identified segment. Our approach rely on the definition of a Customer Database able to collect and store daily recordings of customers visiting the website, which interacts with an Online Learner (at this stage, both UCB1 and TS are still considered) and with a Context Generator, which is run on a weekly basis.

6.1 Optimal solution

The optimal solution is recovered applying usual brute force algorithm to the revenue function defined in section 3 (Equation 7), resulting in the optimal context being the one in which we provide a different price to each true underlying class (coinciding with context C12, see below). The actual results are:

Maximum Expected Revenue	485.37€
p_1*	16.50€
p_2*	21.50€
p_3*	22.50€
$b*$	22.50€

6.2 Implementation

Below is reported a brief recap of the objects interacting during simulation and learning phases.

Feature Generation

Feature Generation is managed through a bivariate Bernoulli sampler. At simulation time, whenever a new customer arrival happens, the sampler randomly assign it to one of the three true underlying class (following the true segmentation of the space provided before), and then returns the values of the two features accordingly to the parameters of the assigned class.

Context Mapping

Contexts considered are, in principle, every possible partitioning of the feature space. We encoded them as reported below.

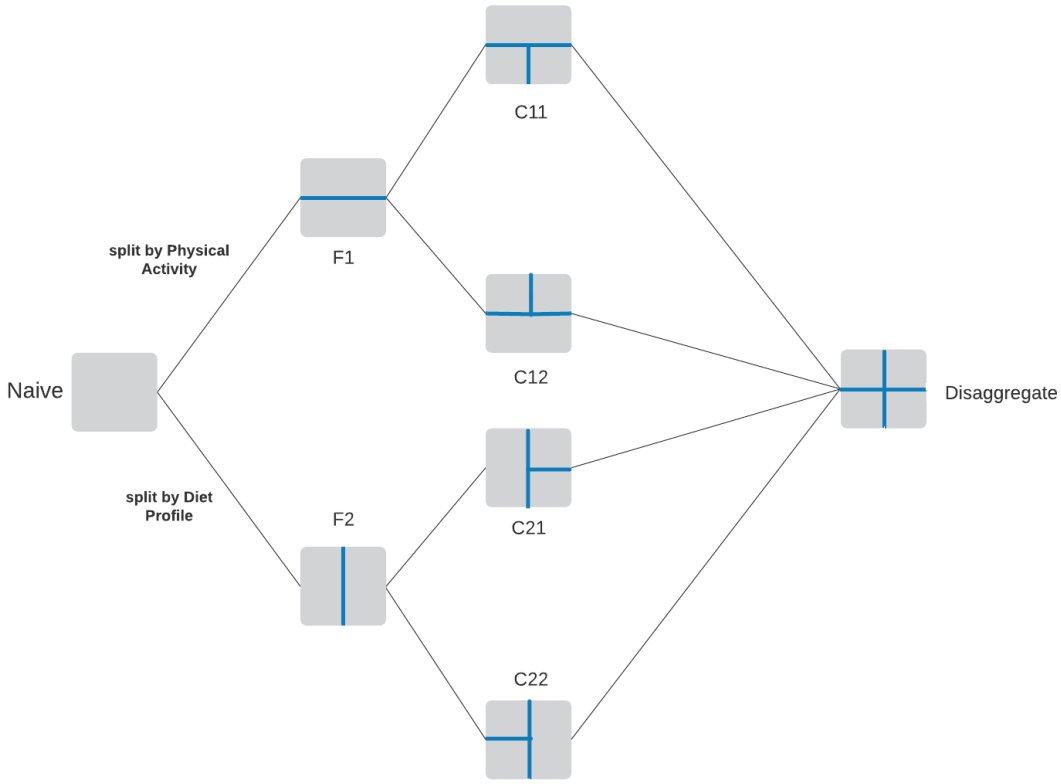

Figure 16: Search Tree for available Contexts

Context Generators

Context Generators are called at the beginning of each week in order to identify the most promising context with respect to data collected until that moment. They implement the search method upon the space of possible context, relying on a database method which, given a candidate context, returns a lower estimate for the best-strategy expected revenue value of that partition (see below for details). We actually consider three different context generators:

- **Naive**: choose always to keep data aggregated. It is used to check the consistency with the previous step results and to check functioning of Mean Returns estimation mechanism
- **Brute Force**: consider every partition and choose the most promising one

- **Greedy**: exploit the binary tree above to perform a greedy search. Each time it is called, built a new context upon the current one finding the most promising feature (if any), and stop when no more features or worthy splits are available. To ensure space exploration, with a given (set to 0.33% by default) probability it restarts the search from the root node.

Since we are only considering two features, and thus the last two methods do not differ much in term of computational complexity, we decided to choose the Brute Force generator in order to avoid any possible bias. However, if the number of features considered grows bigger, the Greedy generator could be a worthy (and sometimes necessary) tool.

Customer

Every time a customer arrival is simulated (i.e. click on the ad and visit the website), it is provided by an ID and a feature vector. These are recorded in the Customer Database, alongside the acceptance (encoded through a boolean) of the price proposed (according to the current context). Moreover, the number of future purchases is sampled from the relative class distribution. Notice that such number is in principle hidden to the Learners and discovered in the successive 30 days.

Customer Database

The database is implemented as follows. It stores:

- **Daily variables**: proposed prices and bid, obtained cost per click and number of daily customers
- **Collection of daily customers**
- **Context history**, recording the chosen context in each week

and provide two main context-dependent methods, working on the collected data:

- one to get the Mean Returns estimates in the provided context structure
- one to get a lower bound for the optimal revenue in the provided context structure

Mean returns estimates are obtained mapping the collected customers into the new context proposed. Notice that, at time t, we only have full observations of number of returns of customers who made their first purchase more than 30 days before, but we consider also partially observed data in the estimation process.
To estimate the Revenue of a proposed context, the database is scanned in order to map collected revenues of each customer in the new context as follows:

$$V_{context} = \sum p_{segment} * \mu^*_{segment}$$

where $\mu^*_{segment}$ refers to the optimal arm daily revenue, while p to the probability of occurrence of the specific segment in the considered context
Lower bounds are used to avoid considering more complex context with too similar expected revenue. We evaluated two different kind of lower bound: a Gaussian one and a modified Hoeffding one, with the former showing better performances.

Learners

The Learners considered are the ones presented in the previous section (TS and UCB1), modified to provide a candidate price per each class in the current context.Moreover, they are provided with an Offline Training method, which perform a batch update of the parameters using collected data, whenever the context is changed.

6.3 Simulation workflow

Each simulation is structured as follows.

Pseudocode for experiment workflow

```
Initialize:
Database
Context ← Naive
Learner
Environment
Forall day:
  if new week:
        new context ← ContextGenerator
        update Database.ContextHistory
        if Learner.Context != new context:
            update Learner.Context
            perform Learner.OfflineTraining
  Daily Routine:
        prices ← Learner.pull-arms
        Database.dailyVariables ←prices, bid, costperclick
        N ← sample from NDC_agg
        for each customer (1 to N):
            create Customer (ID, feature vector)
            acceptance ← Environment.round(feature, context)
            record Customer in Database
        update Learner
        update Database
```

6.4 Performances

To assess performances, as usual we perform one hundred experiments and then look at mean cumulative regret to identify best performing learner and at mean estimated daily reward to ensure convergence to the optimum value. Moreover, a plot of the context history per each experiment is provided.
The two algorithms achieve a cumulative regret shown in Figure 17: also in this case, the Thompson sampling clearly outperforms UCB1.

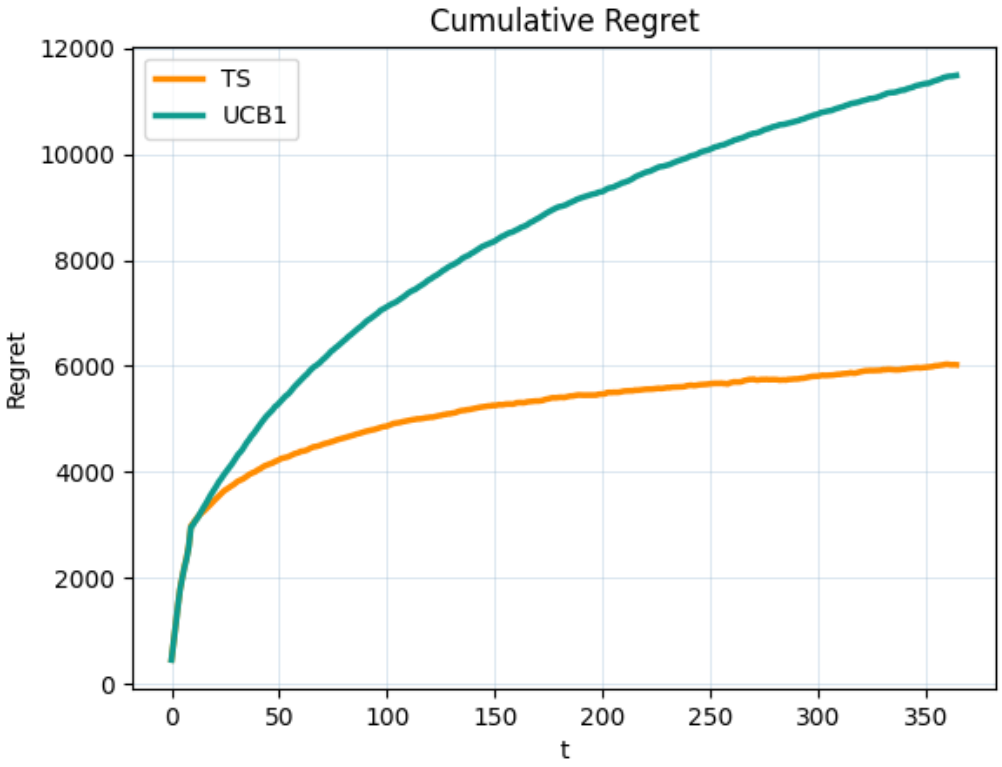

Figure 17: Cumulative Regret - Contextual Pricing

Both the Learners seems to achieve convergence to optimal value, with UCB1 perfomances little affected by the occasional choice of a suboptimal (but of very similar revenue value) arm, as seen in the previous step.

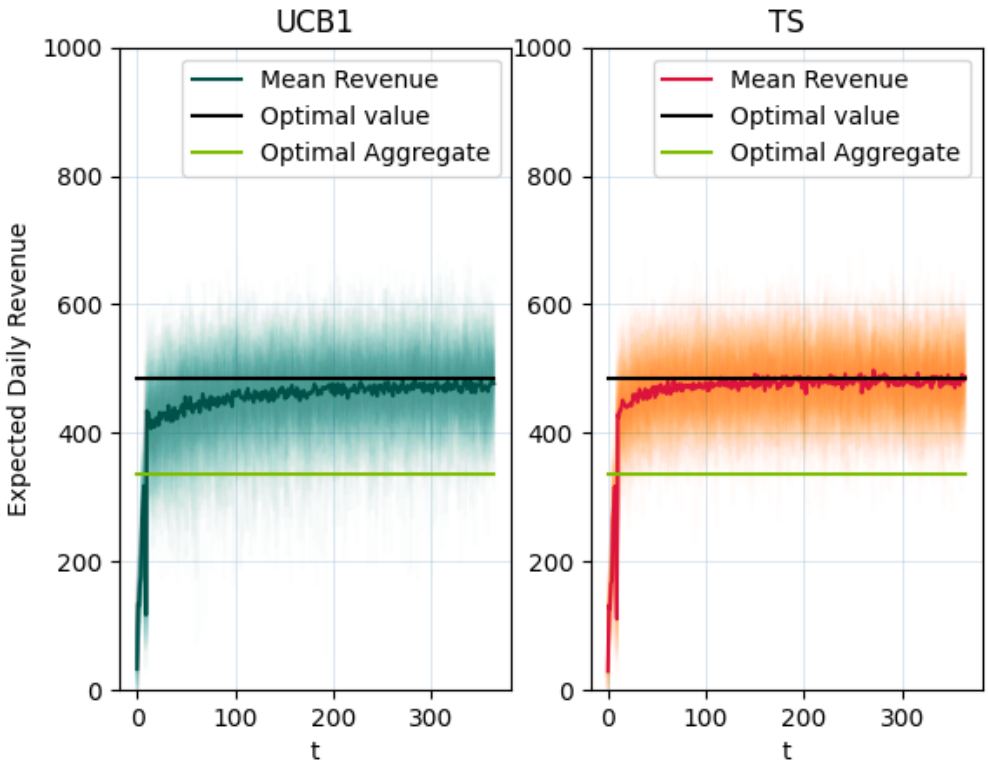

Figure 18: Estimated Expected Daily Revenue across time

Lastly, in Figure 19 isreported the COntext History per each of the 100 experiments.

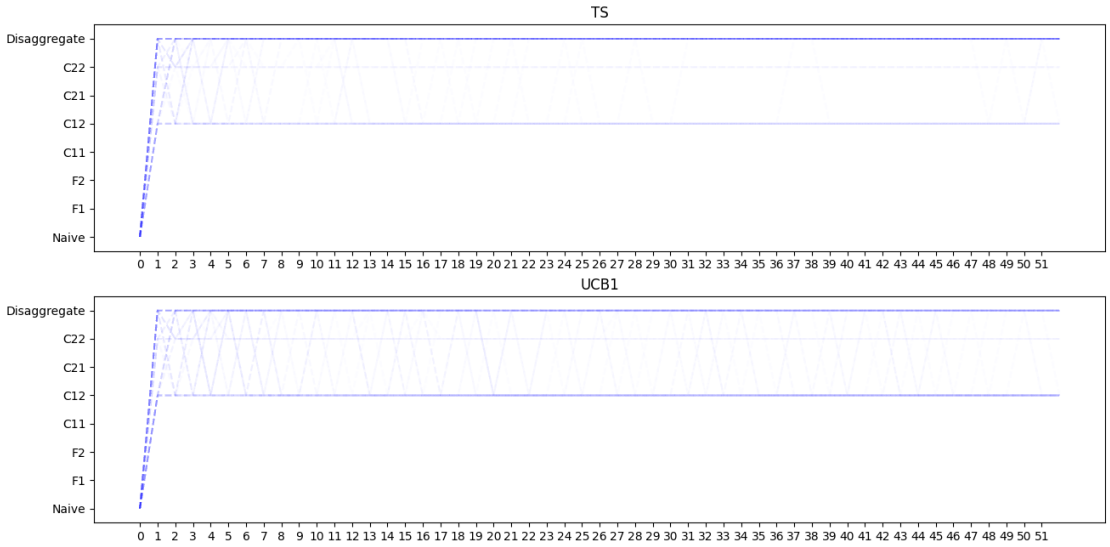

Figure 19: Context History

We can see that the Context Generator actually choose most of the times (and basically always after convergence) the correct contexts: C12 or Disaggregate. This is due to the fact that the Mean Revenue values associated are the same, since in the Disaggregated case the two subclasses composing C3 are provided the same price by the algorithm.

7 Fixed Price & Online Bidding

In this section we deal with the identification of the best advertising strategy, i.e. selecting the best bid in order to maximize the profit. We assume to have an unlimited budget (**no budget constraints**) and, as specified by the assignment, we won't deal with the auction mechanism.

For the advertising problem we will consider solely the **aggregate case**, therefore we will not distinguish between the different classes of users, searching for customized advertising strategies. At this stage we will consider the price fixed to the optimal one, as well as all the price-dependent parameters.

7.1 Environment

The chosen structure is similar to the pricing one, but in this case the environment produces only bidding related values. Specifically, at each round the environment samples values for NDC and CPC accordingly to the selected bid, and then combined them through the relative Revenue expression (with all price-related quantities fixed).

7.2 Learner

For the advertising scenario we consider a MAB setting in which each arm is assigned to a different bid value. Also in this case only 10 candidates are considered, and obtained applying a discretization of the range of interest.
We tested the performances of two different models, comparing their regrets and their convergence properties.

GTS - *Gaussian Thompson Sampling*

The GTS method works exactly as the Thompson Sampling Algorithm, the only difference resides in the conjugated distributions, which are assumed to be Gaussians (both prior and sampling distribution), and not Beta-Binomial as before.

GPTS - *Gaussian Process Thompson Sampling*

The GPTS method relies on the usage of *Gaussian Processes*. This kind of stochastic process takes as input a set of random variables and model each of them as a multivariate normal distribution, returning a Gaussian distribution over the outcome. In our specific case the GPTS learner make use of a GP regressor in order to model the Revenue stochastic distribution. The revenue parameters (mean and variance) are estimated accordingly to the daily collected revenues, dependent from the realizations of NDC and CPC sampled from the environment .
A Gaussian Process can be completely defined by its mean and covariance; in absence

of a-priori specific knowledge, we assumed a zero mean GP with covariance function modeled by a squared exponential kernel function *k(x,x')* plus a white noise:

$$k(x, x') \;=\; \theta^2 e^{-\frac{(x-x')^2}{2l^2}} \;+\; WN(\sigma^2) \tag{12}$$

where the hyperparameters (lengthscale l, scale factor θ and white noise variance σ^2) are estimated by REML methods during the fitting process.
The estimation process is performed each time a new observation of the revenue is obtained at the end of the day; proceeding in this way, the learner becomes more and more reliable reducing the uncertainty of its estimations along time.

Safety Constraint

For both the algorithms we imposed a mechanism that penalizes the arms that could possibly lead to a money loss. In order to avoid this situation, the learners will not play the arms (bid values) having a probability larger than 20% to return a negative revenue. Both algorithms will perform an initial exploration phase, lasting 10 days (rounds), in which this constraint will not be applied. This delay in the constraint application allow us to avoid remaining with an empty set of feasible arms. Notice that, at this point, the constraint will never be activated, due to the fact that the Revenue function with fixed optimal price is significantly positive; nevertheless, the constraint will be useful when dealing with the joint optimization problem.

7.3 Performance Evaluation

Also in the bidding scenario we evaluate the performances of the two methods performing one hundred experiments and then looking at mean cumulative regret to identify the best performing learner. As before we also look at estimated Expected Daily Revenue to ensure convergence to the optimum value.
The two considered algorithms cumulative regret is shown in Figure 20: the Gaussian Process TS seems to perform better than the Gaussian TS in the long run.

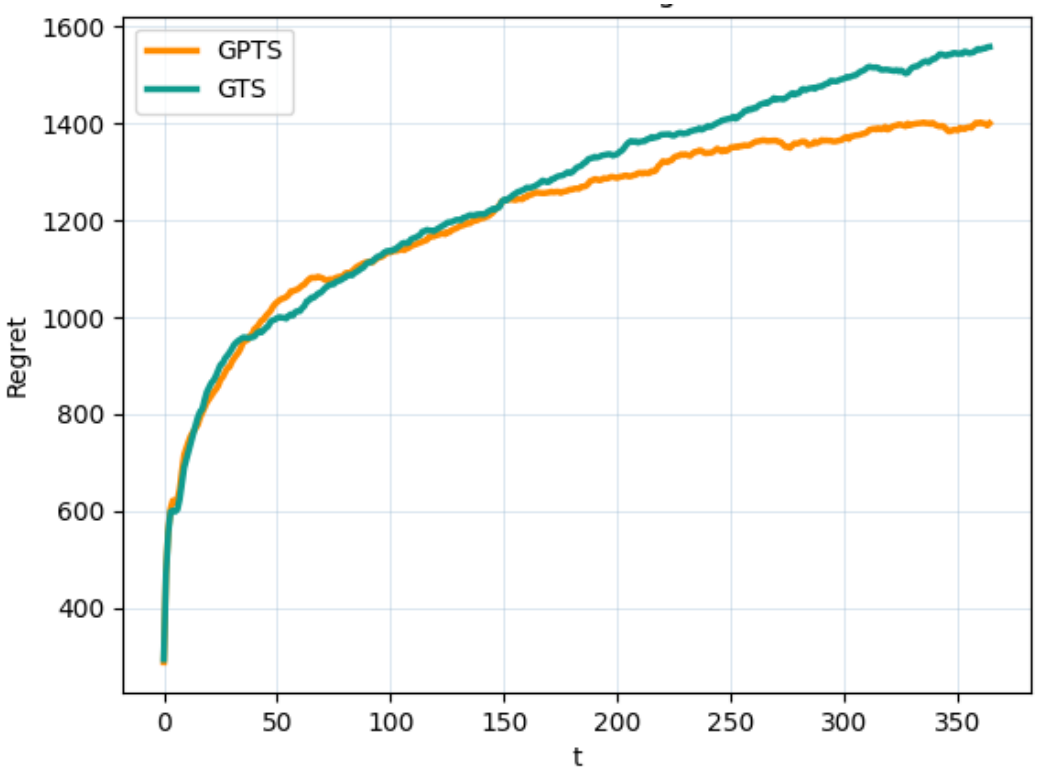

Figure 20: Cumulative Regret - Bidding

As shown in Figure 21, both algorithms appear to converge to the optimal revenue value, but the GPTS seems to perform slightly better, coherently with the regret behaviour. The better performance of GPTS algorithm in the long run may be due to a greater capacity of reducing estimates uncertainty as more data gets available.

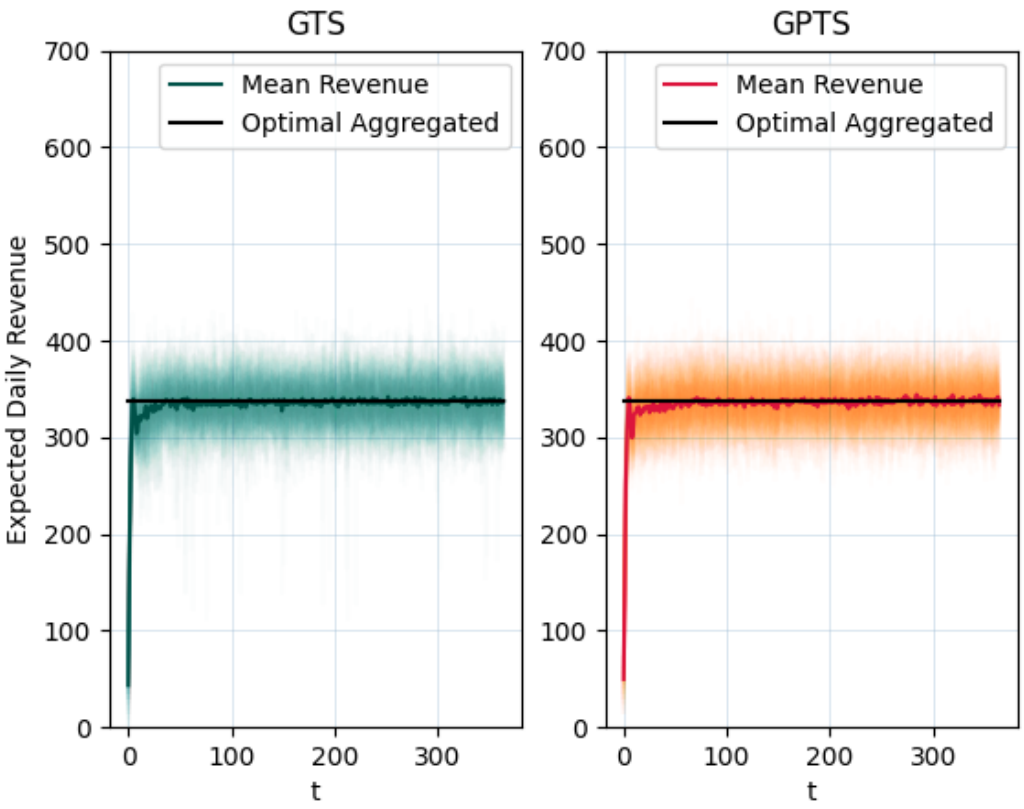

Figure 21: Estimated Expected Daily Revenue

8 Joint Bidding & Pricing Online Problem

Finally, we combine the previously obtained insights to tackle the original problem, consisting in learning the best pricing/bidding joint strategy. In particular we will first approach the aggregate case, in which no customer classification is performed. Then, we will perform an analysis considering contextualized pricing strategies. The idea is to combine sequentially the best performing algorithms considered in the previous sections. Thus, the GPTS algorithm is considered to take actions in the advertising topic and the TS to learn the correct pricing strategy. Of course, due to the increased complexity of the problem to handle, we expect an effect on perfomances. At this stage, the sequential algorithm performance is evaluate in the aggregate and contextual case with respect to the relative optimal value. The final choice about whether choosing to develop contextual pricing options will be carried out in Section 9.

8.1 Aggregate scenario

As said, the main idea behind the procedure is to approach the two problems of advertising and pricing sequentially, making the two algorithms interact continuously in time. At the beginning of each day (round), the GPTS learner pull an arm selecting a bid value and the environment returns the value of the related variables (NDC and CPC), accordingly to the drawn value. Then, the TS learner pull an arm selecting the price to propose to the public till the end of the day. The simulation of the day is carried out using the samples obtained from the environment. Once the joint bidding/pricing strategy is defined, the related daily revenue is returned to the environment and, at the end of each day, both learners update their parameters according to the obtained Revenue. This process is repeated during the whole time horizon allowing the algorithms to learn day-by-day the optimal joint strategy. More specifically:

- **Advertising:** The iteration starts with the GPTS selecting the bid to propose according to the actual collected data. It is important to note that, in this case, the price is not fixed, and thus the algorithm is trying to learn a much more complicated function. That may affect the convergence speed. After the bid choice, the environment proceed sampling a a number of clicks and a cost per click, necessary to simulate the behaviour of the users in the pricing campaign. At the end of the day, the algorithm updates its parameters according to the collected revenue as seen in Section 7,

- **Pricing:**The pricing algorithm will pull an arm, deciding which will be the price drawn for the day; the environment will decide for each user if they accept to buy or not. These values, together with the sampled NDC and CPC, permit to carry out a full simulation of the day and record the relative Expected Daily Revenue. At the end of the day, the algorithm parameters will be updated as seen in Section 5.

Performance evaluation

The sequential algorithm perfomances are evaluated following the usual procedure. In Figure 22 is reported the cumulative regret, computed with respect to the aggregated case optimal solution. The algorithm seems to perform well, although the transient phase, as expected, is longer than before.

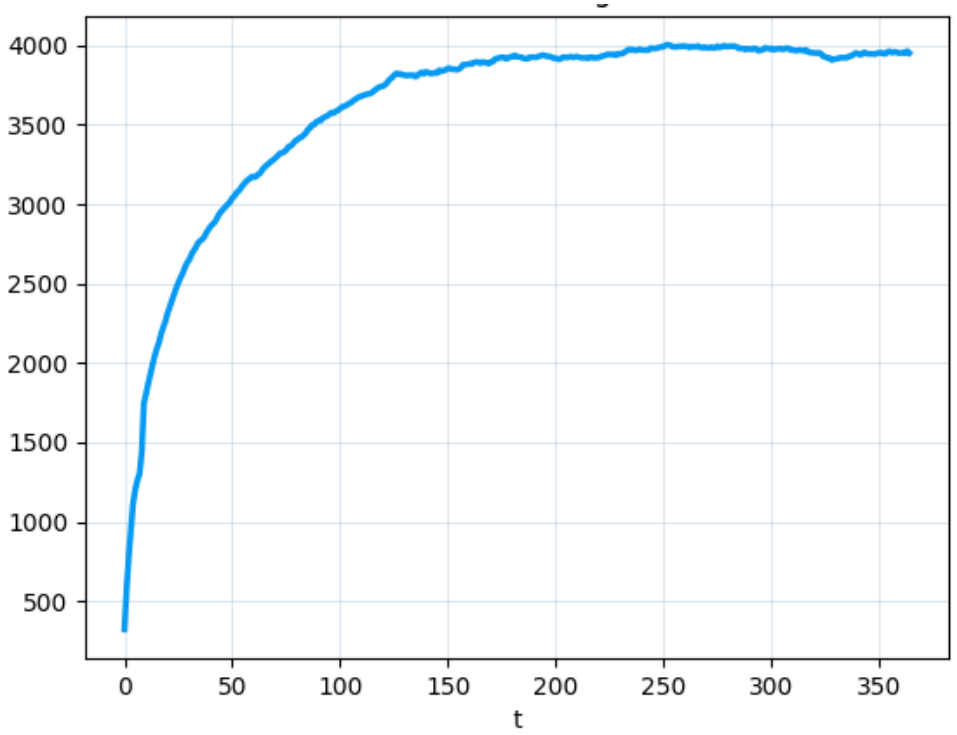

Figure 22: Cumulative Regret - Aggregated Case

Actually, in Figure 23 we can see that the algorithm succeeds in identifying the optimal joint strategy, converging to the optimal value. The very small value observed in the very early phase is due to an unfortunate joint strategy chosen in the purely exploratory phase.

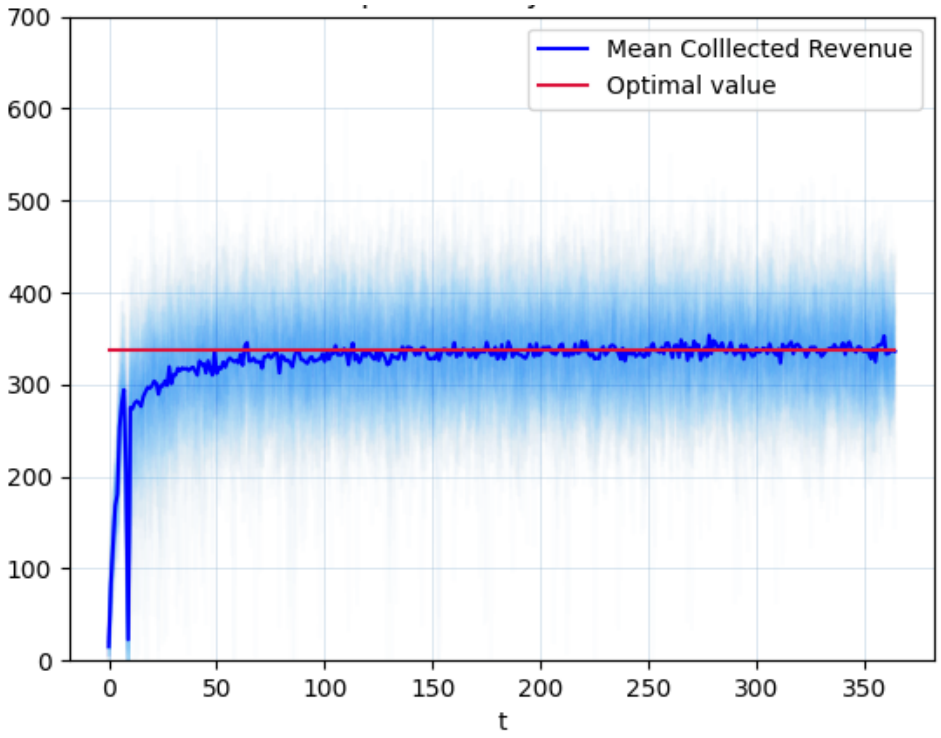

Figure 23: Estimated Expected Daily Revenue across time

8.2 Contextual Pricing scenario

In this section we consider the interaction mechanism described above, but considering custom pricing strategies for the optimal context discovered in Section 6 (C12). The simulation procedure works exactly the same, the only difference is that a price is proposed each day to each one of the three classes identified by Context C12. Notice that we are considering the optimal Context to be known, and not trying to discover it as in Section 6.

Performance evaluation

The cumulative regret is computed with respect to the Disaggregated optimal solution. Its behaviour (Figure 24) testifies that, in this case, the difficulty to converge are more relevant than before. However, from Figure 25 we can argue that the algorithm finally succeeds in identifying the optimal strategy also in this case.

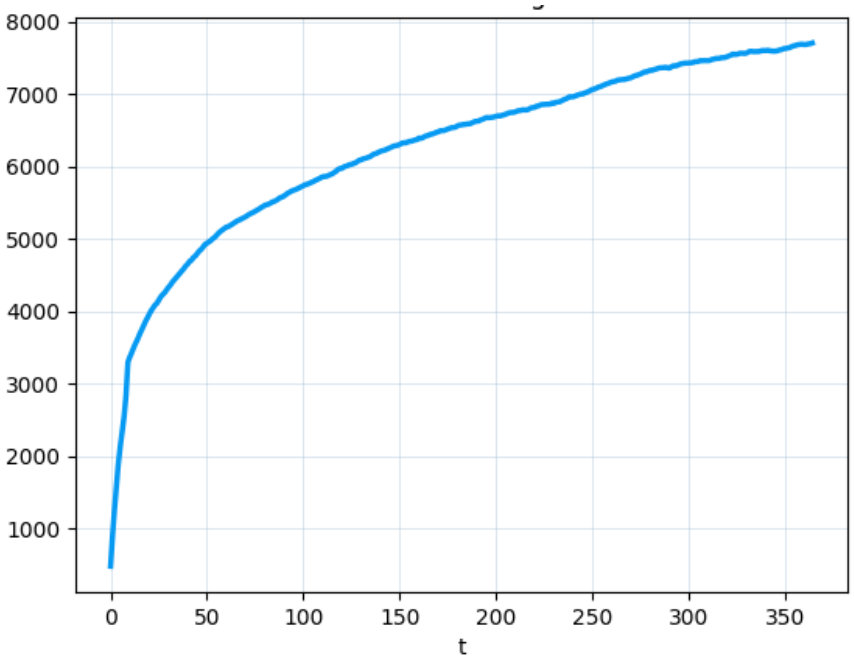

Figure 24: Cumulative Regret - Contextual Pricing Case

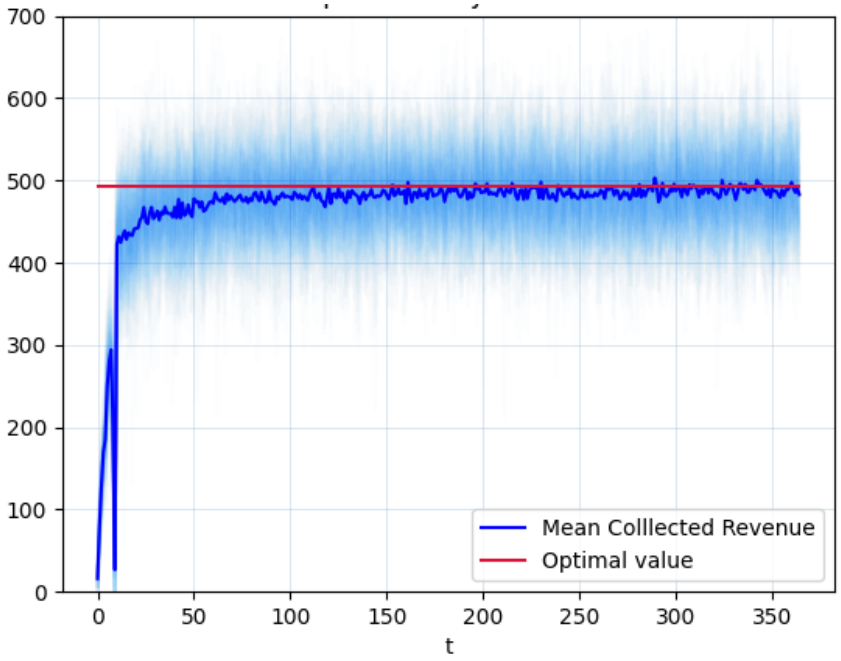

Figure 25: Estimated Expected Daily Revenue across time

9 Conclusion

In this final section we try to furnish a comprehensive solution to the presented problem. To recap, we have to design an online learning algorithm capable of identifying the best joint bidding and pricing strategy for an e-commerce website. We built a sequential learner, relying on a GPTS approach to tackle the advertising problem and on a TS approach to tackle the pricing one. We evaluate the Learner both in the case of aggregated data and in the case of contextual pricing, ensuring in both the convergence to the optimal strategy. We know from Section 3 that the contextual solution provide a higher Expected Daily Revenue, with a value of 492.50€, with respect to the aggregated case, with a value of 337.90€. However, what is left to do is to decide whether adopt a contextual approach or not during the learning phase. Actually, we must verify that the lower rate of convergence does not make the contextual solution less profitable with respect to the aggregated one. That may happen, but this is clearly not the case, with the contextual algorithm outperforming the aggregated one: in Figure 26 is reported the comparison between the two algorithms cumulative regret, computed with respect to the true optimal solution (the contextual one). Finally, we can conclude that the provided contextual sequential learner is the best option.

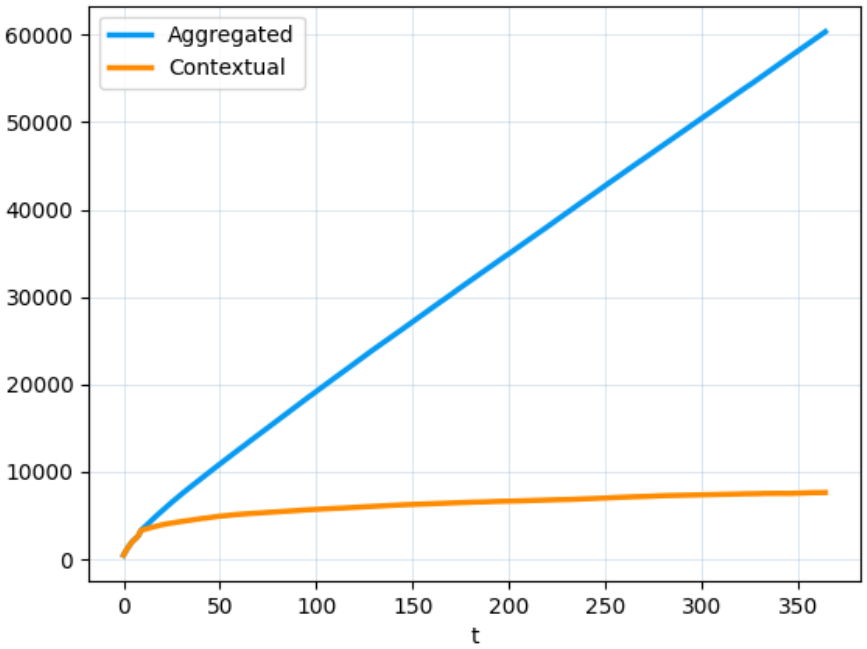

Figure 26: Comparison of Cumulative Regrets