Data Intelligence Application

Liguoro Claudio, Mapelli Alessia, Tonazzi Sara

Abstract

This report presents an algorithm to study and search for the best joint bidding and pricing strategy in a particular environment scenario, online markets. Our result is theoretical and presents some simplifications but can be generalized to real case scenarios. We start by presenting the problem and it's mathematical modeling and then we construct our solution step by step from a smaller problem to the general solution. We will consider an experiment with 3 different classes of customers characterized by specific functions related to the probability of purchasing our item given a price, number of clicks gained and the cost per click for the advertisement campaign. Our simplification is mostly related to the advertisement; indeed, we imagine having an unlimited budget and we do not include in the problem the auction mechanism typical of the advertisement scenario. We intrude a complexity related to the pricing strategy: we suppose that the user will purchase again a stochastic number of times in the following month and that this information is available only at the end of the month.

Context description

We analyze the scenario in which advertisement is used to attract users to a streaming movie platform and we suppose that the user, after the purchase of its first movie, will come back in future to purchase more. The goal is to find the best joint bidding and pricing strategy considering future purchases.
We imagine the users to be characterized by two binary features:

Studying = {student, not student}
Age = {over 60, under 60}

Based on these features we consider three customers' class:

C1: not student - over 60;
C2: student - under 60;
C3: not student - under 60.

We don't consider the class student – over 60 because it is improbable.
We characterize each customers' class by:

- a stochastic number of daily clicks of new users as a function depending on the bid:
  this function is constructed as a deterministic function of the bid to which we add a gaussian noise

$$n(b) = N\left(1 - e^{-\alpha b}\right) + \mathcal{N}(0,2) \qquad (1)$$

  N: maximum number of clicks
  α: slope of the curve
  We define the parameters by customizing them for the specific classes.

  C1: N =50 and α= 1;
  C2: N =200 and α= 0.5;
  C3: N =150 and α= 0.7.

  We represent here a realization:

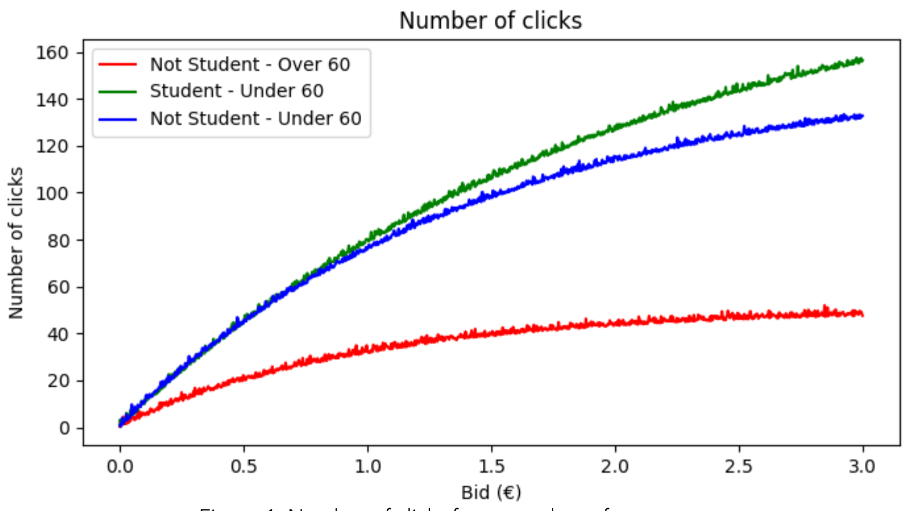


Figure 1: Number of clicks for every class of users

- a stochastic cost per click as a function of the bid: the function is given as an addition between a determinist function and a uniform noise

$$c(b) = b - \mathcal{U}[0, \frac{b}{k}] \qquad (2)$$

k: parameter related to the competition
As before we define the parameters for the specific classes.
C1: $k = 10$;
C2: $k = 6$;
C3: $k = 8$.
We represent here a realization:

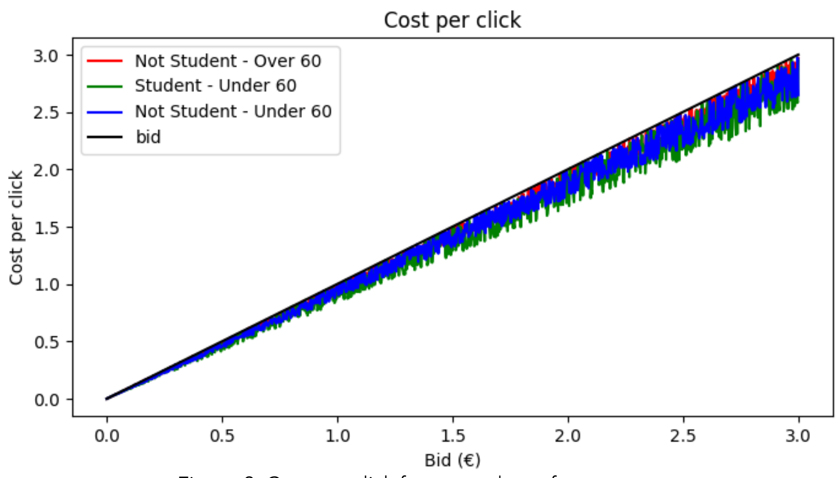


Figure 2: Cost per click for every class of users

- a conversion rate function providing the probability that a user will buy the item given a price: we construct the function as a quadratic spline interpolation of specific points. Given p = [3, 6, 9, 12, 15] then we relatively take
C1: [0.6 0.5 0.35 0.2 0.1];
C2: [0.9 0.75 0.4 0.25 0];
C3: [0.8 0.7 0.6 0.45 0.2].
We represent here a realization:

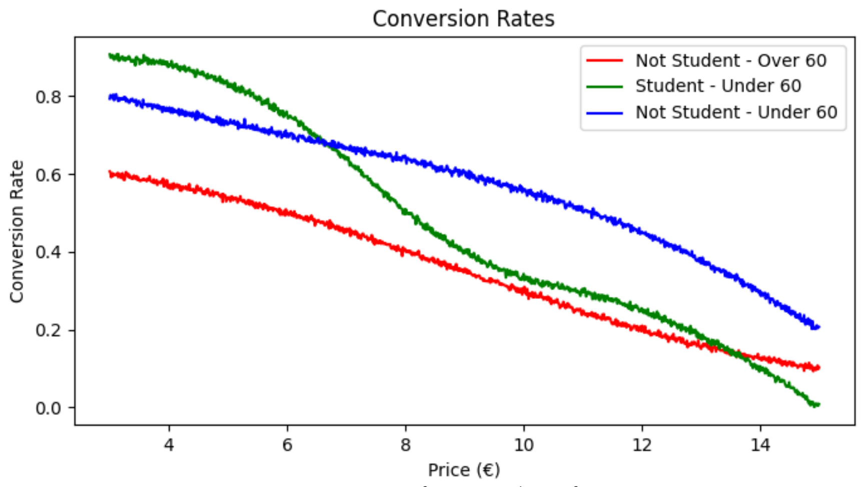


Figure 3: Conversion rate for every class of users

- a distribution probability over the number of times the user will come back to the ecommerce website to buy the same item within 30 days after the first purchase: we model this probability as a Poisson

$$\mathbb{P}_\lambda(n) = \frac{(\lambda)^n}{n!}e^{-\lambda} \qquad (3)$$

  λ is constant in each class and defined as:

  C1: $\lambda = 2$

  C2: $\lambda = 5$

  C3: $\lambda = 4$

  We represent here the probability distribution over the days in each class

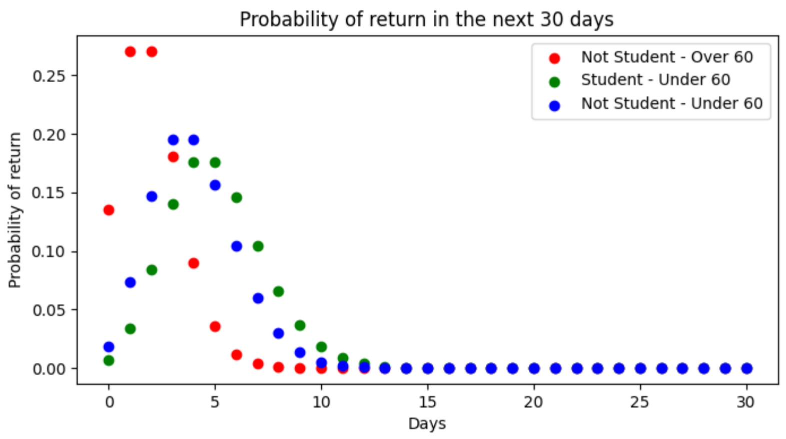


Figure 4: Probability distribution of the comebacks for every class of users

We assume that once a user makes a purchase with a price p, then the ecommerce will propose the same price p to future visits of the same user and this user will surely buy the item.

Problem set up

The first thing we concentrate on is the formulation of the optimization problem we need to solve. We suppose that every parameter is known in the beginning to find a general mathematical way to find the best pricing and advertising strategy. We imagine working on three different sub-campaign each one having as a target one of the class of customers described above. Some notation is useful: considering one round, in our case one day, we have:

$\boldsymbol{b_j}$	Bid related to the sub-campaign j

$\boldsymbol{p_j}$	Price related to the sub-campaign j
$\boldsymbol{m_j}$	Margin of sub-campaign j
$\boldsymbol{cpc_j}$	Cost-per-click of sub-campaign j
$\boldsymbol{n_j}$	Number of daily clicks of new users of sub-campaign j
$\boldsymbol{cr_j}$	Conversion rate of sub-campaign j
$\boldsymbol{\mathcal{P}_j}$	Poisson distribution of number of times the user will come back to the ecommerce website to buy the item by 30 days after the purchase for the sub-campaign j
$\boldsymbol{C}$	Fixed-cost referred to a single item, known beforehand

Table 1: Parameters related to the optimization problem

The optimization problem is evaluated on a time horizon of one year and the objective function we need to maximize is:

$$f(p_j, b_j) = \sum_{j=1}^{3} \left[m_j(p_j) - cpc_j(b_j)\right] * n_j(b_j) \qquad (4)$$

$$m_j(p_j) = cr_j(p_j) * (p_j - C) * (1 + \mathcal{P}_j) \qquad (5)$$

The aim of our optimization is maximizing the value of each day: to do so we refer to each revenue and cost in this unit. For each click we evaluate its cost and its value: the value is calculated based on the probability that a user will buy the item given the price and considering each repurchase after the first one. The objective function, $f(p_j, b_j)$, is stochastic: we need to maximize then its expected value $\mathbb{E}\left[f(p_j, b_j)\right]$. Each round the optimization problem becomes:

$$\mathbb{E}\left[\max_{p_j, b_j} \sum_{j=1}^{N} \left[m_j(p_j) - cpc_j(b_j)\right] * n_j(b_j)\right] = \max_{p_j, b_j} \sum_{j=1}^{N} \left[\mathbb{E}\left[m_j(p_j)\right] - \mathbb{E}\left[cpc_j(b_j)\right]\right] * \mathbb{E}\left[n_j(b_j)\right] \qquad (6)$$

Thinking about p and b to be positive real variables, solving this problem theoretically it's a challenge. The objective function is not convex, could present several local minima and it cannot be disaggregated in two different optimizations for the pricing and advertising. We could try and perform a gradient descent method in this case with no guarantee of success. However, in a real case scenario we usually discretize the prices and bids between which we can choose the best one. Usually not many values are considered since the budget spendable in each campaign is limited and the prices are considered in the range of the competitors. Note that in our optimization problem we will assume the budget to be unlimited and we will not reproduce the auction behavior typical of the advertising setting.

We can implement, then, an algorithm related to the discretized problem when we have a limited number of prices and bids: given the prices and bids considered we will simply evaluate the objective function on the grid and take the best solution. We consider the following prices and bids:

- Prices = [3, 4, 5, 7, 8, 9, 10 ,12, 13, 15]
- Bids = [0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 3]

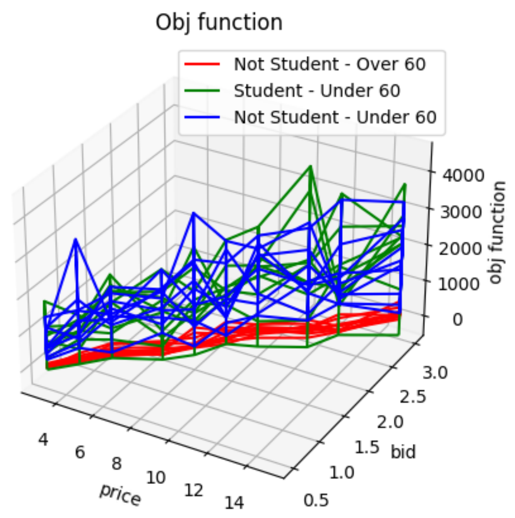


Figure 5: Objective Function evaluated on the grid of prices and bids

The algorithm has a complexity of $O(3 * N_p * N_b)$, where N_p and N_b are respectively the number of prices and the number of bids provided and 3 is the number of sub-campaigns considered.
Applying this "grid-search" algorithm on our specific problem we can find the best strategy for each of the class and the aggregated case:
For class C1 the optimal bid is 2 and the optimal price is 9 with a maximal objective function value of 262.64; for class C2 the optimal bid is 3 and the optimal price is 7 with a maximal objective function value of 2884.92; for class C3 the optimal bid is 3 and the optimal price is 10 with a maximal objective function value of 2775.75;
In the aggregated optimization problem, the one where we do not distinguish between the customers' classes, the optimal bid is 3, the optimal price is 8 and the objective function in this case is 5291.22.
Note that the total value of the disaggregated strategy is higher compared to the one of the aggregated strategy. In practice we start focusing on the aggregated case to understand afterwards if the disaggregation in more than one class is the best choice.

Modeling the unknown parameters

Once the problem has been modeled, we need to understand which parameters are unknown in a real case scenario and how to learn them in an online fashion. The goal of this section is to design an online learning algorithm for pricing and advertising parameters. The analyzed campaign consists of three sub-campaigns to advertise the product, each targeting a different class of users. Focusing on a single sub campaign we need to learn:

$\boldsymbol{cr(p)}$	Conversion rate related to price p
$\mathbb{E}[\boldsymbol{\mathcal{P}}]$	The expected value of the number of times the user will come back to the e-commerce website to rebuy the item by 30 days after the first purchase
$\boldsymbol{cpc(b)}$	Cost per click related to the bid b
$\boldsymbol{n(b)}$	Number of clicks related to the bid b

Table 2: Notation related the model of parameters

First, one needs to develop a model for each of the parameters the algorithm is going to learn. Considering each parameter as a random variable we can hypothesize their distribution. We suppose the conversion rate related to each price to be distributed as a Bernoulli with an unknown expected value $c_{rate}(p)$ that represent the probability of purchase of a custumer given the specific price for the item.

$$cr(p) \sim Be(\, c_{rate}(p)\,) \qquad (7)$$

Then we suppose that the probability distribution over the number of times the user will come back to the e-commerce website to buy that item by 30 days after the first purchase is a Poisson distribution with an unknown expected value.

$$\mathcal{P}_j(p) \sim Po(\,\lambda) \qquad (8)$$

Finally we suppose that the cost per click and the number of clicks are distributed normally with mean equal to the unknown true value of the function and variance that represent the error with which the data are measured.

$$cpc_b \sim \mathrm{N}(\mathrm{c(b)}, \sigma^2) \qquad (9)$$

$$n_b \sim \mathrm{N}(\mathrm{n(b)}, \sigma^2) \qquad (10)$$

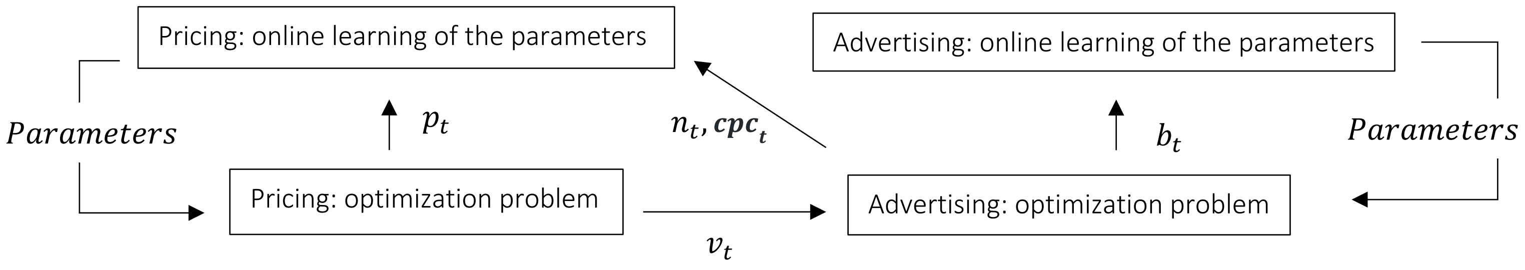
The goal now is to be able, during time, to learn these parameters based on data coming from responses of the users to possible strategies. The algorithm we propose assumes that, once chosen b and p, we receive feedback form the environment thanks to which we can update and improve our estimation. We need to consider, in our scenario, the delay feedback we receive due to the user's comeback. We imagine being able to evaluate this parameter only after 30 days from each user's first purchase, so that we receive incomplete information every day and we retrieve the missing part only after 30 days. Suppose to discretize the time in days and suppose to consider a set of ten bids and ten prices to work on in a time horizon of 365 days. We will briefly discuss the Environment, with which we can simulate real feedback, before presenting the algorithm.

Environment:

We design two Environments, one related to Pricing and one to Advertising. They are both black boxes from the point of view of the algorithm. The Pricing Environment iterates over the days for the entire duration of the experiment. Each day given the number of clicks, the cost per click and the specific sub-campaign, returns the feedback of each user and simulates the return of users for each purchase. The Environment related to advertising returns the number of clicks and the cost per click associated to each sub-campaign, by iterating over the days for the entire time horizon. In our setting, it computes the optimal number of clicks knowing the real functions, one for each sub-campaign, generating the number of clicks given a bid value as does with the cost per click.

Algorithm:

Our algorithm works separately on each sub-campaign and it is divided in 4 steps:



Pricing: online learning of the parameters
Advertising: online learning of the parameters
$Parameters$
p_t
n_t, cpc_t
b_t
$Parameters$
Pricing: optimization problem
Advertising: optimization problem
v_t

Figure 6: Schematic presentation of the algorithm

Pricing: online learning and optimization problem

We estimate $\mathbb{E}[\mathcal{P}]$ throw the average mean. To learn the conversion rate, we exploit a Thompson Sampling algorithm. For every price we have a priori on its expected value, in this case the reward is a Bernoulli distribution, and the natural conjugate prior is a Beta distribution. For every price we compute the value per click, and we choose the price that maximizes it. Once we get the observed realizations, we can update the Beta accordingly and do another iteration of the algorithm. Note that we consider only one price for each iteration of the algorithm, for each day, instead of differencing it to each client. This is far more accurate when we work on a real case scenario. We introduce the notation we will exploit in the algorithm pseudo-code.

$\boldsymbol{P}$	Set of prices
$\boldsymbol{p_t}$	The price played at time t
$\mathbb{P}(\boldsymbol{\mu_p} = \boldsymbol{\theta_p})$	Prior on the expected value of cr_p
$(\boldsymbol{\alpha_p}, \boldsymbol{\beta_p})$	Parameters of the Beta distribution $\mathbb{P}(\mu_p = \theta_p)$
$\boldsymbol{n_t}$	Number of clicks at time t
$\bar{\boldsymbol{x}}$	Sample average of the realization of $\mathcal{P}$
$\vec{\boldsymbol{x}}_{\boldsymbol{t}}$	Realization of random variable $\mathcal{P}$ at time t
$\overrightarrow{\boldsymbol{cr}}_{\boldsymbol{p,t}}$	Realizations of random variable cr_p at time t

Table 3. Notation related to Pricing Learner

$$\mathbb{P}(\mu_p = \theta_p) = \frac{\Gamma(\alpha_p,\beta_p)}{\Gamma(\alpha_p)+\Gamma(\beta_p)} (\theta_p)^{\alpha_p - 1}(1-\theta_p)^{\beta_p - 1} \qquad (11)$$

Beta probability distribution

0. Initialization: Since we do not have a priori information on the best price to choose we give each of them a uniform probability: $\forall p \in P$

$$(\alpha_p, \beta_p) = (1,1) \qquad (12)$$

$$\bar{x} = 0 \qquad (13)$$

1. At every time t play price p_t such that
   - $\forall p$ sample from the Beta distribution

$$\delta_p \leftarrow Sample(\mathbb{P}(\mu_p = \theta_p)) \qquad (14)$$

   - play price p_t such that

$$p_t \leftarrow argmax_{p\in P} \left(\delta_p * (p - C) * (1 + \bar{x})\right) \qquad (15)$$

   - define then v_t as:

$$v_t = \left(\delta_{p_t} * (p_t - C) * (1 + \bar{x})\right) \qquad (16)$$

2. Once the feedbacks are collected, update the Beta distribution of price p_t

$$(\alpha_{p_t}, \beta_{p_t}) \leftarrow (\alpha_{p_t}, \beta_{p_t}) + (n_{purchase}(t), n_{clicks}(t) - n_{purchase}(t)) \qquad (17)$$

Where $n_{purchase}(t)$ is the number of purchase at time t and $n_{clicks}(t)$ is the number of clicks at time t.

3. If t > 30 then update the estimation of $\bar{x}$

$$\bar{x} \leftarrow \frac{\bar{x} * z_{t-30} + \sum_{i=1}^{n_{purchase}(t-30)} \vec{x}_{t-30}^{\,i}}{z_{t-30} + n_{purchase}(t-30)} \qquad (18)$$

Where z_{t-30} represent the number of purchases made before t-30

Advertising: online learning and optimization problem

To learn the cost per click and the number of clicks functions we use a GTS. To every parameter is linked a Gaussian Distribution, for every bid then we have the probability distribution over the number of clicks and the cost per click. At every time t and for every bid we evaluate the objective function, and we choose the bid with the maximal value. Once we get the observed realizations, we can update the Gaussian distribution accordingly and do another iteration of the algorithm.

$\boldsymbol{B}$	Set of bids
$\boldsymbol{b_t}$	The bid played at time t
$\mathbb{P}(\boldsymbol{\mu_b}^{\boldsymbol{n}} = \boldsymbol{\theta_b})$	Prior on the expected value of N_b
$(\boldsymbol{\omega_b}^{\boldsymbol{n}}, \boldsymbol{\sigma_b}^{\boldsymbol{n}})$	Parameters of the Gaussian distribution $\mathbb{P}(\mu_b{}^n = \theta_b)$
$\mathbb{P}(\boldsymbol{\mu_b}^{\boldsymbol{cpc}} = \boldsymbol{\theta_b})$	Prior on the expected value of cpc_b
$(\boldsymbol{\omega_b}^{\boldsymbol{cpc}}, \boldsymbol{\sigma_b}^{\boldsymbol{cpc}})$	Parameters of the Gaussian distribution $\mathbb{P}(\mu_b{}^{cpc} = \theta_b)$
$\boldsymbol{v_t}$	Value related to the sub-campaign at time t
$\boldsymbol{n_{b,t}}$	Realization of random variable n_b at time t
$\boldsymbol{cpc_{b,t}}$	Realization of random variable cpc_b at time t

Table 4. Notation related to Advertising Learner

$$\mathbb{P}(\mu_b = \theta_b) = \frac{1}{\sqrt{2\pi(\sigma_b)^2}}\, e^{-\frac{(\theta_b - \omega_b)^2}{2(\sigma_b)^2}} \qquad (19)$$

Gaussian probability distribution

0. Initialization: Since we do not have a priori information about the best bid to choose $\forall b \in B$

$$(\omega_b{}^n, \sigma_b{}^n) = (0{,}50) \qquad (20)$$
$$(\omega_b{}^n, \sigma_b{}^n) = (0{,}50) \qquad (21)$$

1. At every time t play price b_t such that

$$\delta_b{}^n \leftarrow Sample(\mathbb{P}(\mu_b{}^n = \theta_b)) \qquad (22)$$
$$\delta_b{}^{cpc} \leftarrow Sample(\mathbb{P}(\mu_b{}^{cpc} = \theta_b)) \qquad (23)$$
$$b_t \leftarrow argmax_{b\in B}\, \delta_b{}^n\, (\boldsymbol{v_t} - \delta_b{}^{cpc}) \qquad (24)$$

4. Once the feedbacks are collected, update the Normal distribution of number of the clicks and the cost per click of bid b=b_t

$$\overrightarrow{cpc_b} = [\overrightarrow{cpc_b}\, cpc_{b,t}] \qquad (25)$$
$$\overrightarrow{n_b} = [\overrightarrow{n_b}\, n_{b,t}] \qquad (26)$$

$$\boldsymbol{\omega_b}^{\boldsymbol{cpc}} = \frac{\sum cpc_b{}^i}{|\overrightarrow{cpc_b}|} \qquad (27)$$
$$\boldsymbol{\sigma_b}^{\boldsymbol{cpc}} = \frac{\sum (cpc_b{}^i - \omega_b{}^{cpc})^2}{|\overrightarrow{cpc_b}|} \qquad (28)$$
$$\boldsymbol{\omega_b}^{\boldsymbol{n}} = \frac{\sum n_b{}^i}{|\overrightarrow{n_b}|} \qquad (29)$$
$$\boldsymbol{\sigma_b}^{\boldsymbol{n}} = \frac{\sum (n_b{}^i - \omega_b{}^n)^2}{|\overrightarrow{n_b}|} \qquad (30)$$

Learn the best pricing strategy

We start with the implementation of the algorithm related to the price optimization part. The idea is to learn in online fashion the best pricing strategy assuming known the advertising strategy. In this first step we focus on solving the aggregated optimization problem, so without any distinction between the classes. In our analysis we will assume the bid value to be fixed and consequently that the number of daily clicks and the daily cost per click will be known deterministically. We keep the discretization done in the previous point. We need to design an effective algorithm to learn the optimal price when the demand curve is not a priori known, and the estimation process takes a long time. To achieve this goal, we make use of general-purpose bandit algorithms. More specifically, we make use of the Upper Confidence Bound and Thomas-Sampling approaches. We need to implement the pricing optimization of the strategy seen in the previous section that will be specified for the two approaches.
The Environment in this section is implemented so that for every round given the number of clicks we cycle on every class in the optimization problem, so in our case all three of them, and it returns the total number of purchase, the total number of returns related to purchase of 30 days ago , the total number of clicks and the reward collected that day.

The Upper Confidence Bound approach to the problem is frequentist. Every price is associated with an upper confidence bound, an optimistic estimation of the conversion rate provided by that arm. At every round the arm with the highest value per click, calculated with the confidence bound, is chosen. This arm optimistically provides the largest expected the reward. Finally, after having observed the realization of the reward of the chosen arm the upper confidence bound is updated accordingly. We can show the pseudocode:

$\boldsymbol{P}$	Set of prices
$\boldsymbol{p_t}$	The price played at time t
$\boldsymbol{\mu_p}$	Mean related to each price p
$\boldsymbol{\Delta_p}$	Deviation from the mean consider doe each price p
$\boldsymbol{n_{clicks}(t)}$	Number of clicks at time t
$\boldsymbol{\bar{x}}$	Sample average of the realization of $\mathcal{P}$
$\boldsymbol{\vec{x}_t}$	Realization of random variable $\mathcal{P}$ at time t
$\boldsymbol{n_{purchase}(t)}$	Sum of the realizations of random variable cr_p at time t

Table 5. Notation related to Pricing Learner

1. Initialization: $\forall p \in P$

$$\mu_p = 0 \qquad (31)$$
$$\boldsymbol{\Delta_p} = +\infty \qquad (32)$$

2. Play once every $p \in P$

3. At every time t play price p_t such that
   - $\forall p$ evaluate

$$UB(p) = \mu_p + \boldsymbol{\Delta_p} \qquad (33)$$

   - play price p_t such that

$$p_t \leftarrow argmax_{p \in P} \left(\mathrm{UB(p)} * (p - C) * (1 + \bar{x})\right) \qquad (34)$$

4. Once the feedback are collected, update the parameters:

$$\mu_{p_t} = \frac{\mu_{p_t} * tot_{obs,p_t}(t-1) + n_{purchase}(t)}{tot_{obs,p_t}(t-1) + n_{clicks}(t)} \qquad (35)$$

   Where $tot_{obs,p_t}(t-1)$ is the number of observations had on p_t until t-1.
   Then $\forall p \in P$

$$\Delta_p = \sqrt{\frac{2 * \log(t+1)}{tot_{obs,p}(t-1)}} \qquad (36)$$

   If t > 30 then update the estimation of $\bar{x}$

$$\bar{x} \leftarrow \frac{\bar{x} * z_{t-30} + \sum_{i=1}^{n_{pucahse}(t-30)} x_{t-30}{}^{i}}{z_{t-30} + n_{pucahse}(t-30)} \qquad (37)$$

   Where z_{t-30} represent the number of purchase done before t-30

Thompson Sampling approach to the problem is instead Bayesian. The pseudo code is the one proposed in the previous section.

The results can be discussed based on the two plots below. These are the mean results of 50 experiments performed to get a general overview of the performance independent from the stochasticity of the parameters. We tested the performances of the models comparing them with the clairvoyant algorithm and evaluating their regret. We can see that both UCB and TS reward grow towards the optimal value of the objective function with UCB algorithm taking some rounds more to get to convergence then TS. Moreover, UCB suffers more in terms of reward for the incompleteness of the information about the returns. As Usual TS performs better than UCB, getting to convergence within the first 50 days, but UCB performs well in our case scenario too.

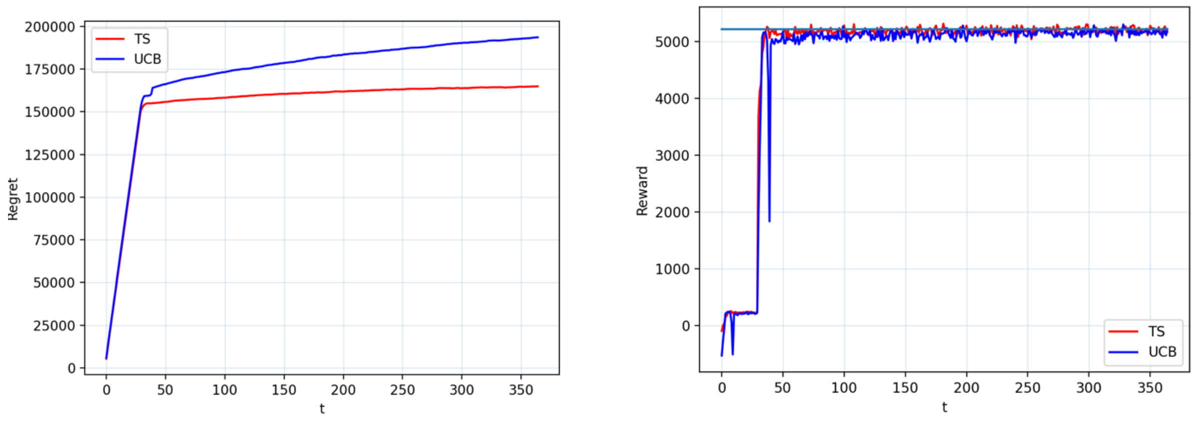


Figure 7: Regret and Reward representation of the implemented algorithm for the price optimization

Learn the best Context to perform the price optimization

At a later stage we adopted a context-generation approach to identify different classes of customers and different pricing strategies for each of them. The idea is to exploit the same algorithm described in the above section on different settings given by a context generation algorithm. In particular we will exploit TS algorithm that performs better in the aggregated case.
The context generalization discriminates the customers' classes at the pricing level and not at the advertising level, so that the price and context optimization is performed with fixed bid evaluated in the optimization stage of the problem for the aggregated case.
Starting from the aggregated class, we evaluate simultaneously the values of the current context and the values of the alternative ones, based on different repartitions of the class set. The value of a context structure is evaluated as:

$$\sum_{c \in \mathcal{R}} \pi_c \mu_{p_c^*,c} \qquad (38)$$

where π_c is the probability that context c occurs and $\mu_{p_c^*,c}$ is the best expected reward for context c and R is the partition considered within the customer set. To compare the value of different partitions R we consider the lower bounds of the expected reward. To compute it we need to evaluate a lower bound for the reward of the specific partition and a lower bound for the probability of the context occurrence. We choose for each of them the Hoeffding bound:

$$\bar{x} - \sqrt{-\frac{\log(\delta)}{2|Z|}} \qquad (39)$$

Where δ is the confidence, that we set equal to 0.05 and |Z| is the cardinality of the data set for each of the quantity considered.

The splitting condition for the first stage is:

$$\pi_{c_1}\mu_{p_{c_1}^*,c_1} + \pi_{c_2}\mu_{p_{c_2}^*,c_2} \geq \mu_{p_0^*,c_0} \qquad (40)$$

$c_1$and c_2 are the two classes of one of the possible alternative contexts:

- alternative 1: student in one class and not student in the other
- alternative 2: over 60 in one class and under 60 in the other
- alternative 3: student (under 60) and over 60 (not student) in one class and not student under 60 in the other

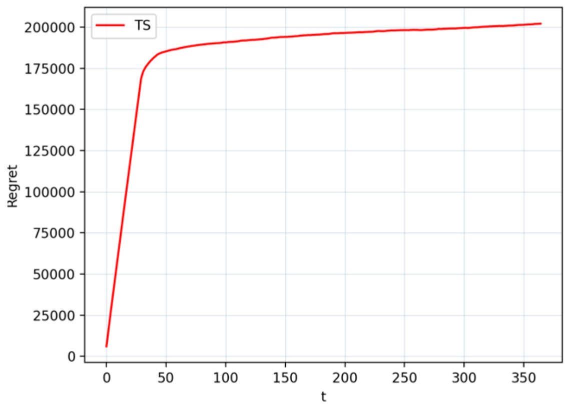
To every class of every considered context, we link a Learner to estimate the unknown parameters. For every context we create a different Environment in which we store the information about the past purchases, specifically for every type of user. This information can't be stored in the Learners because they don't know which type of users they refer to. Hence, the first stage consists of 7 different Learners: one for the current context and two for each of the three alternative contexts and 4 different Environment: one for the current context and one for the three alternative contexts.
The splitting condition is checked every two weeks and it is reached with the alternative 3 in a period between 20 and 30 days after the beginning of the experiment.
At this point the alternative 3 becomes the current context and the alternative context is the one composed of all the three types of users disaggregated.
Hence, in this stage we have five Learners: two of the current context, inherited from the past alternative 3 and the other three are the alternative ones: the student's one, the over 60's one and the under 60 not student's one.
Here the splitting condition is:

$$\pi_{c_3}\mu_{p_{c_3}^*,c_3} + \pi_{c_4}\mu_{p_{c_4}^*,c_4} + \pi_{c_5}\mu_{p_{c_5}^*,c_5} \geq \pi_{c_1}\mu_{p_{c_1}^*,c_1} + \pi_{c_2}\mu_{p_{c_2}^*,c_2} (41)$$

Where c1 and c2 are the two classes of the current context (student (under 60) and over 60 (not student) in one class and not student under 60 in the other). While c3,c4,c5 are the three different classes of customers considered in the alternative context.

This condition is reached at time t=70, so from now on the current context is the disaggregated one and obviously we don't have any alternative context.

The pricing optimization with context generalization performance can be explained by the graphs below. These are the mean results of 50 experiments performed to get a general overview of the performance independent from the stochasticity of the parameters. We tested the performances of the model comparing it with the clairvoyant algorithm and evaluating its regret. As we can see also in this case TS reaches the optimal disaggregated objective function quite rapidly: our algorithm is then able to manage and elaborate a class diversification and price differentiation. In more complex cases we usually evaluate the possible alternative contexts as splitting of only one customers' feature. In our case the possibilities are so restricted that we can think about using an extensive search.



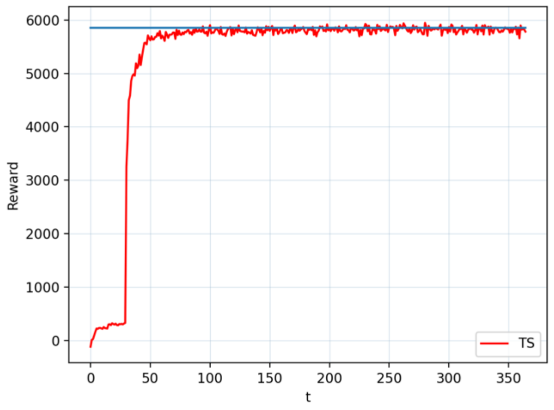


Figure 8: Regret and Reward representation of the implemented algorithm for the disaggregated price optimization

Learn the best bidding strategy

In this section we identify the best advertising strategy. Our aim is to select the best bid to maximize the profit. To simplify the task, we deal with unlimited budget (no budget constraints) and without the simulation of the auction mechanism. The bid optimization is implemented in the aggregated case, no distinctions between the different classes of users is performed and we will consider the price fixed to the optimal aggregated one, as well as all the price-dependent parameters.

The Environment 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 according to the selected bid, and then combines them through the relative revenue expression returning to the environment the daily value obtained and the objective function related to them.

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 the same discretization considered before. The Gaussian Thompson Sampling was exploited in particular: this 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. The pseudocode is showed below

$\boldsymbol{B}$	Set of bids
$\boldsymbol{b_t}$	The bid played at time t
$\mathbb{P}(\boldsymbol{\mu_b}^{\boldsymbol{obj}} = \boldsymbol{\theta_b})$	Prior on the expected value of the objective function
$(\boldsymbol{\omega_b}^{\boldsymbol{n}}, \boldsymbol{\sigma_b}^{\boldsymbol{n}})$	Parameters of the Gaussian distribution $\mathbb{P}(\mu_b{}^{obj} = \theta_b)$
$\boldsymbol{v_t}$	Value related to the sub-campaign at time t
$\boldsymbol{n_{b,t}}$	Realization of random variable n_b at time t
$\boldsymbol{cpc_{b,t}}$	Realization of random variable cpc_b at time t

Table 6. Notation related to Advertising Learner

$$\mathbb{P}(\mu_b = \theta_b) = \frac{1}{\sqrt{2\pi(\sigma_b)^2}}\, e^{-\frac{(\theta_b-\omega_b)^2}{2(\sigma_b)^2}} \tag{42}$$

Gaussian probability distribution

2. Initialization: Since we do not have a priori information about the best bid to choose $\forall b \in B$

$$\left(\omega_b{}^{obj}, \sigma_b{}^{obj}\right) = (0{,}50) \tag{43}$$

3. At every time t play price b_t such that

$$\delta_b \leftarrow Sample(\mathbb{P}(\mu_b = \theta_b)) \tag{44}$$

$$b_t \leftarrow argmax_{b \in B}\, \delta_b \tag{45}$$

5. Once the feedback are collected, update the Normal distribution of bid b=b_t

$$obj_{b,t} = n_{b,t} * \left(v_t - cpc_{b,t}\right) \tag{46}$$

$$\overrightarrow{obj_b} = [\overrightarrow{obj_b}\ obj_{b,t}] \tag{47}$$

$$\boldsymbol{\omega_b{}^{obj}} = \frac{\sum obj_b{}^i}{|\overrightarrow{obj_b}|} \tag{48}$$

$$\boldsymbol{\sigma_b{}^{obj}} = \frac{\sum(obj_b{}^i - \omega_b{}^{obj})^2}{|\overrightarrow{obj_b}|} \tag{49}$$

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

Results of the algorithm can be discussed in reference to the figures below. We tested the performances of the algorithm comparing the reward with the clairvoyant algorithm. These are the mean results of 50 experiments performed to get a general overview of the performance independent from the stochasticity of the parameters. This algorithm performs well, it captures the right bid in the first days of the year, performing as good as the clairvoyant algorithm after only a few weeks.

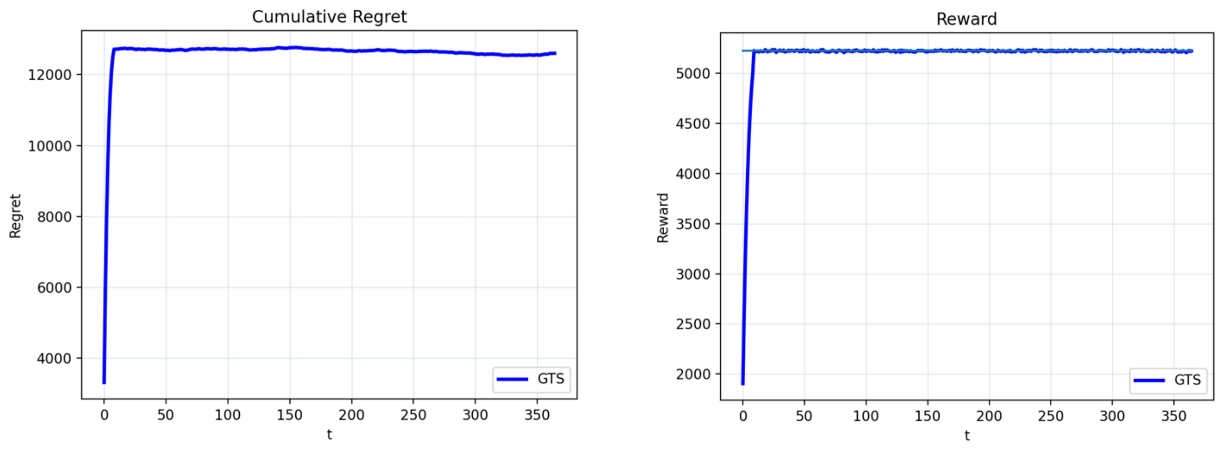


Figure 9: Regret and Reward representation of the implemented algorithm for the bid optimization

Learning the best aggregated pricing and bidding strategy

Once we have analyzed each single part of both the pricing and bidding strategy we now want to implement and analyze the algorithm to perform the simultaneous optimization of the two strategies. In this first phase we will still work on the aggregated scenario, without any distinction in both pricing and advertising between the customers' classes. Starting from what we have implemented so far, we will follow the algorithm developed in the theoretical resolution of the problem.

We construct an environment that gives us the possibility to both get feedback given the pricing strategy and the advertising strategy. The idea is to merge the environments from the price and bid optimization, getting one of them to evaluate the full objective function. The daily reward is obtained by drawing each of the quantities from the random variable associated with it. This environment, at each round, will return all the quantity needed for the update of both the pricing and the advertising parameters.
GTS for the To implement the algorithm, we use the logic explained in the theoretical resolution of the problem: we will exploit three MAPs, two advertising and a simple TS for the pricing part. Note that, in this case, we will work separately on the number of clicks and the cost per click estimation in the bid optimization, still pulling the arm that gives us the best objective function value, this allows the price and bid decision to be more independent. Moreover, in this setting the threshold of the advertising condition is increased to 30% to avoid empty eligible arms.

The resulting algorithm performance is presented in the figures below. We tested the performances of the algorithm comparing the reward with the clairvoyant algorithm. These are the mean results of 50 experiments performed to get a general overview of the performance independent from the stochasticity of the parameters. The algorithm converges to the desired result, we can see that the converse is also pretty fast implying that our algorithm is able to perform well on the problem presented in this report.

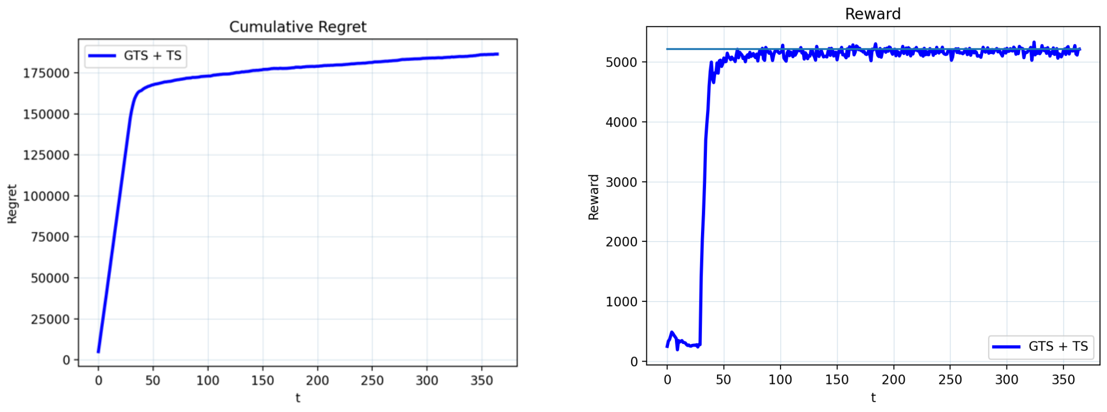
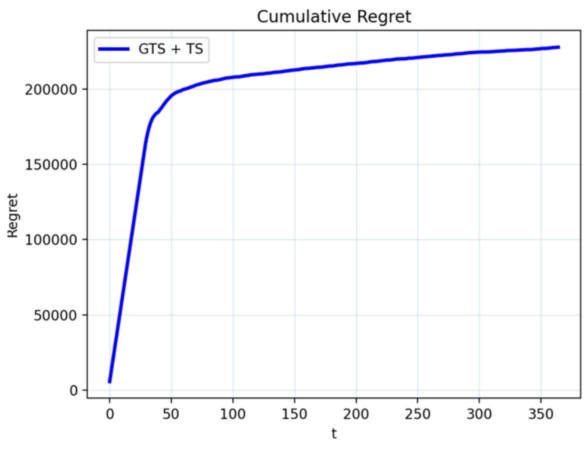


Figure 10: Regret and Reward representation of the implemented algorithm for the aggregated bid, price optimization

Learning the best disaggregated pricing and bidding strategy

The last step is then to exploit the algorithm on the context discovered in the disaggregated price optimization. The logic will be the same as in the previous section, we will just define the environment, the pricing and bidding learner for each class separately. We can then sum the daily objective function of each of them to evaluate the performance of the algorithm.
The performance is presented on the graph below and it is pretty similar to the one discussed in the section above. These are the mean results of 50 experiments performed to get a general overview of the performance independent from the stochasticity of the parameters. The convergence as we can see is slower and the algorithm assesses just below the optimal value. We are still able to learn the best price bid strategy in general.



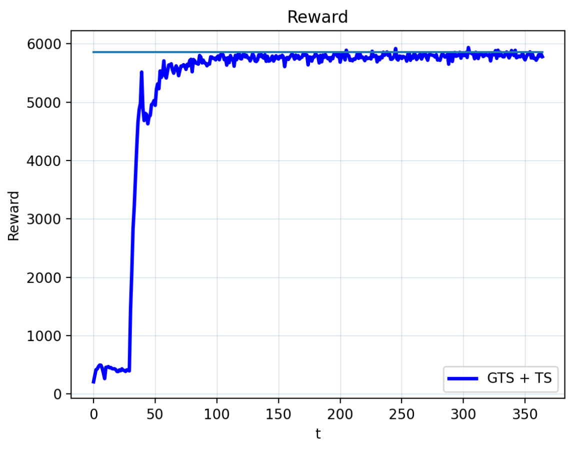


Figure 11: Regret and Reward representation of the implemented algorithm for the disaggregated bid, price optimization

Conclusions

We developed an algorithm that deals with scenarios in which we need to simultaneously learn the best bidding and pricing strategy to exploit in a marketing campaign. To get to the final implementation we studied the problem from a theoretical point of view, starting to describe it, evaluating its parameters and their modeling, and implementing a brute force algorithm for the resolution of the problem to the optimality in the case of fully known parameters. We started then to concentrate on the pricing part of the problem, fixing the bid and all the parameters connected to it to their optimal values and learning in an online fashion the price and the context to discriminate the prices. Then we studied the bid optimization in the case of a fixed price and related parameters to the optimal value. In the end we merge the two approaches to find the best joint bid pricing strategy in both an aggregated and disaggregated context.