OLA PROJECT

Pricing and Advertising

AY 2022/2023

STUDYING THE E-COMMERCE SALE OF A LEGO TECHNIC AT-AT SET

Marco Lucchini, Chiara Mocetti, Massimo Perna, Giulia Rebay. Giorgio Romano

OUR PROJECT IN STEPS

STEP 0
MOTIVATIONS AND ENVIRONMENT DESIGN

STEP 1
LEARNING FOR PRICING

STEP 2
LEARNING FOR ADVERTISING

STEP 3
LEARNING FOR JOINT PRICING AND ADVERTISING

STEP 4
CONTEXTS AND THEIR GENERATION

STEP 5
NON STATIONARY ENVIRONMENTS WITH TWO ABRUPT CHANGES

STEP 6
NON STATIONARY ENVIRONMENTS WITH MANY ABRUPT CHANGES

SETTING UP THE ENVIRONMENT

We consider an online e-commerce website selling a unique product, specifically a Lego Technic AT-AT (small version) set with moving components.

We assume that customers can be divided into three categories, each of which is defined by a specific combination of binary features, **F1** and **F2**.

FEATURE DESCRIPTION

- **F1**: Is the customer older than 30?
- **F2**: Is the customer a Lego membership owner?

CLASS	FEATURE 1	FEATURE 2
Collector	True	True
Parent	True	False
Young	False	False

CLIENT CLASS OVERVIEW

Customers will differ, based on their class, in terms of:

- How the conversion probability varies as the price varies;
- How the number of daily clicks varies as the bid varies;
- How the daily cost of the clicks varies as the bid varies.

Exploration is done over a time horizon of **365 days** and with the following assumptions:

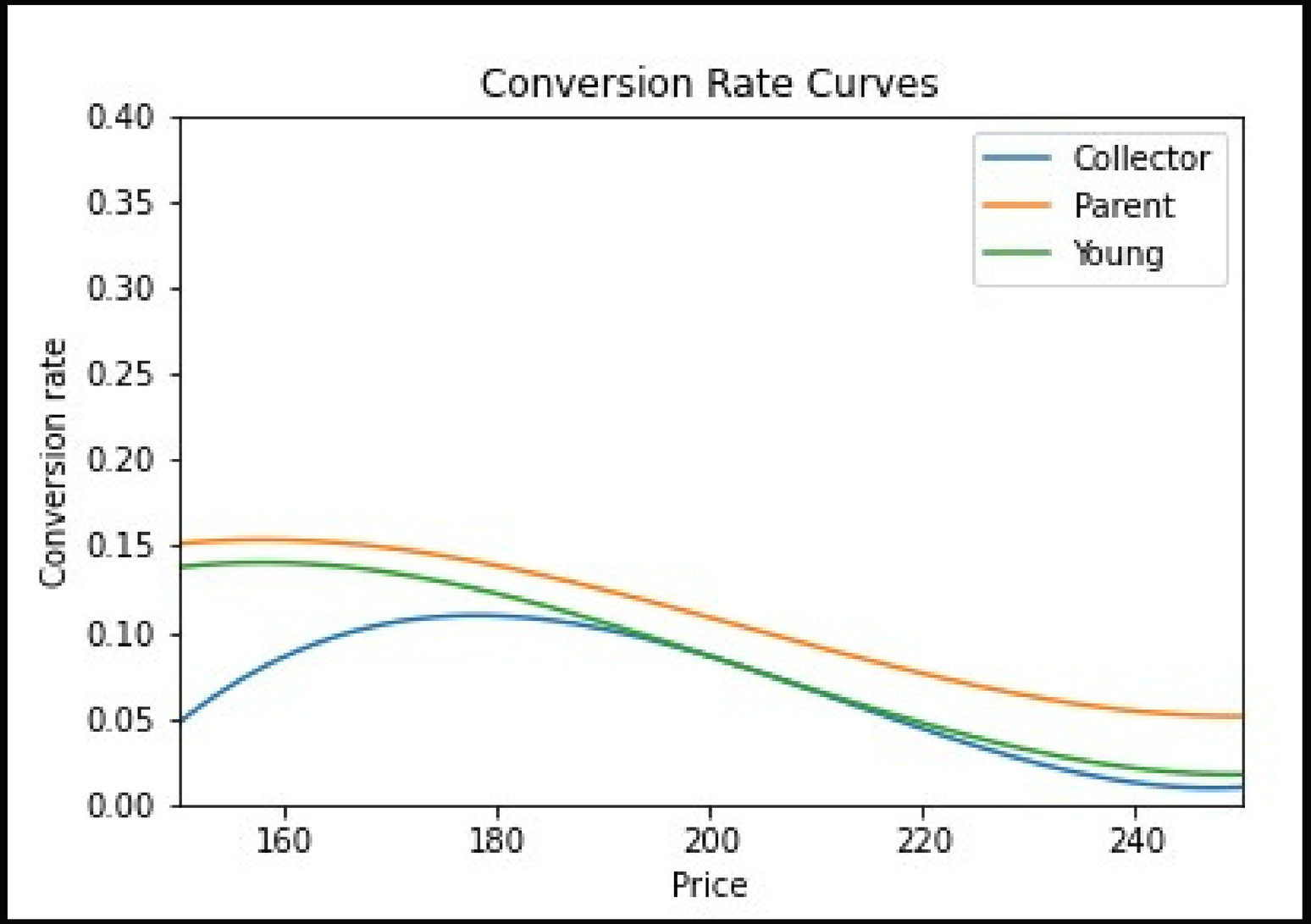
PRICING PROBLEM:

Our Lego Set is sold in a price range going from 150€ to 250€, equally split in five price intervals: **150€ - 175€ - 200€ - 225€- 250€**

ADVERTISING PROBLEM:

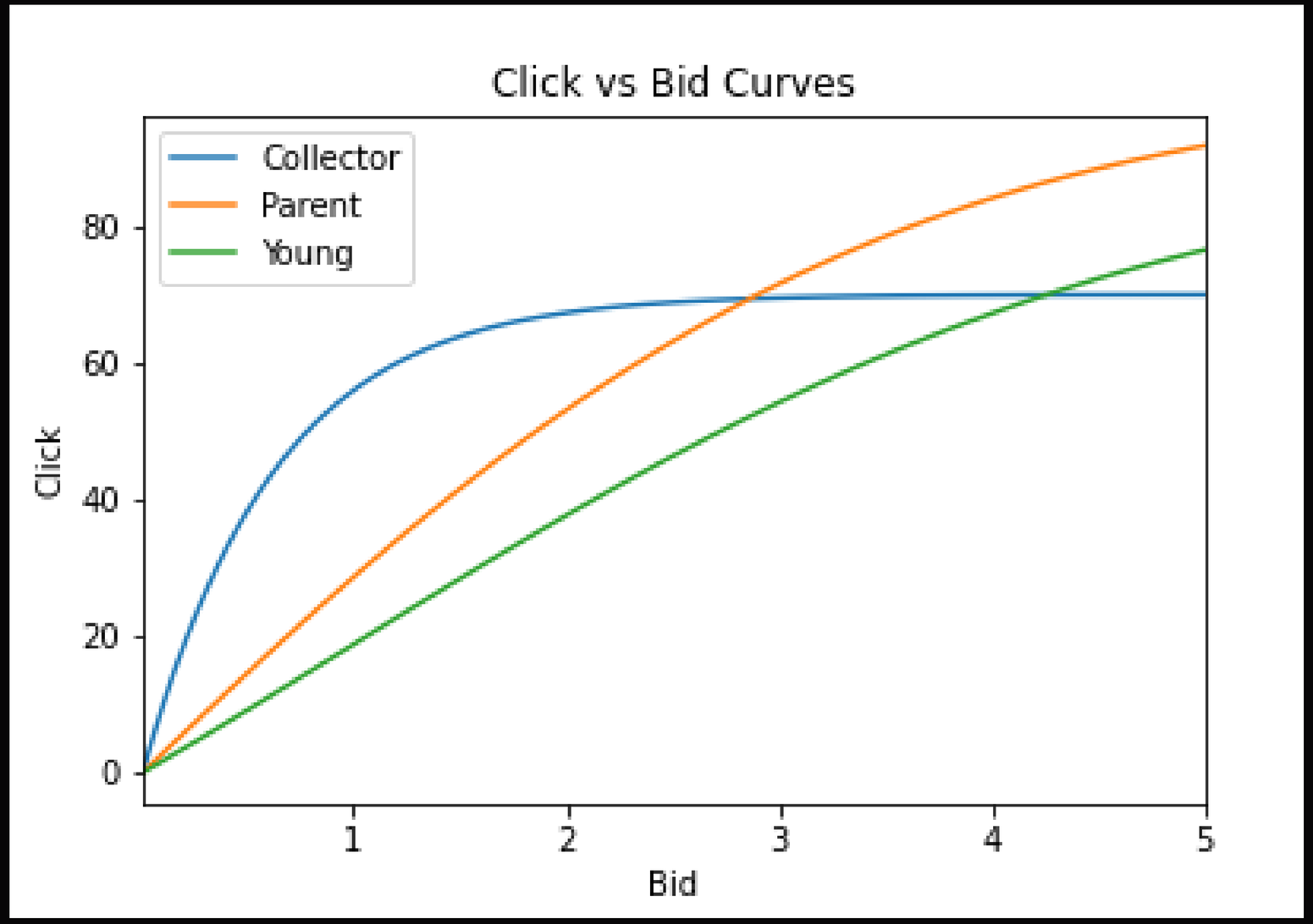
The bid can take any value in the interval 0.01 - 5€, and can take any of the 100 equally spaced values within that interval.

THE PRICING PROBLEM



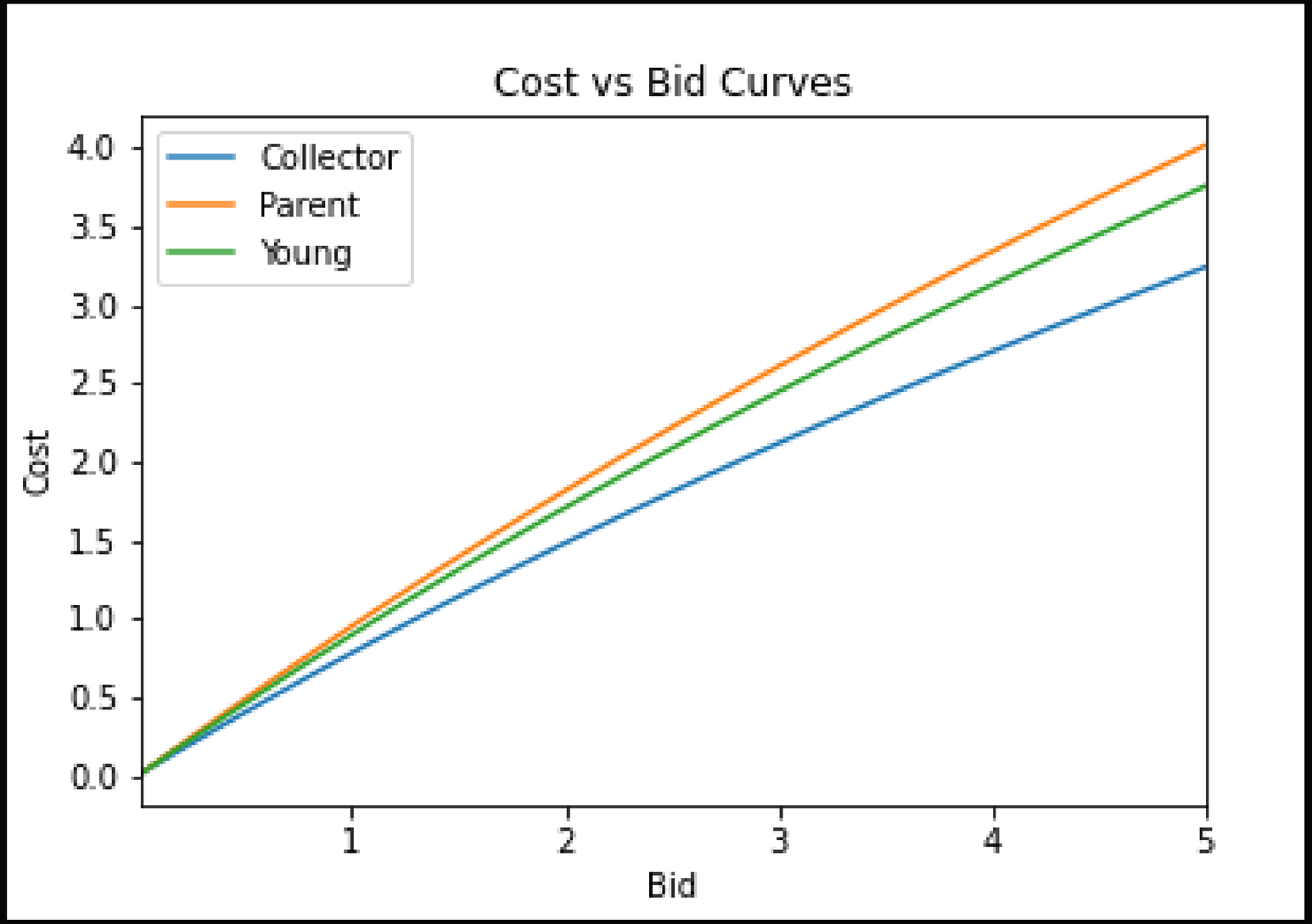
THE ADVERTISING PROBLEM

Number of daily clicks vs Bid



THE ADVERTISING PROBLEM

Cost of daily clicks vs Bid



STEP 0: IMPLEMENTATION

A class implementation approach is adopted: **Environment**, **Learner** and **User** classes are defined.

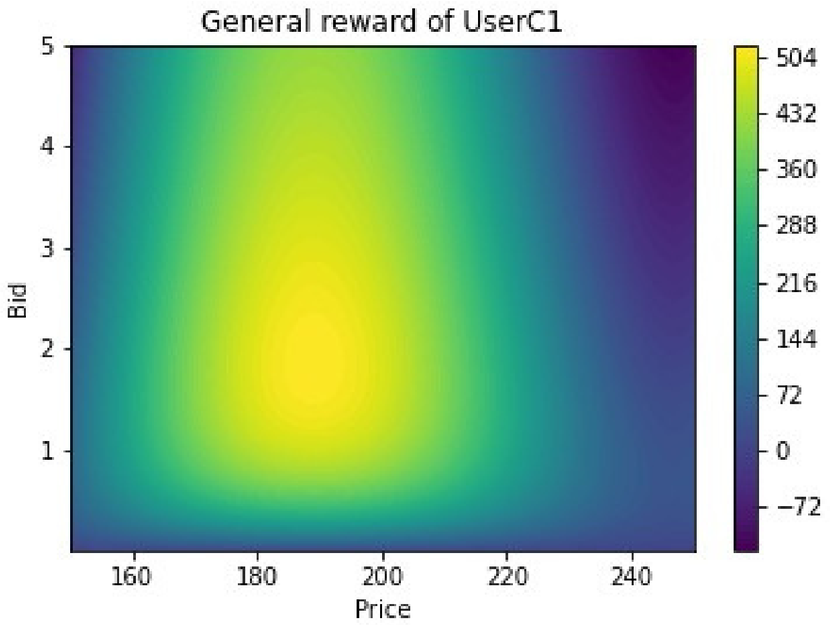
We assume the number of rounds to be equal to **365** days.

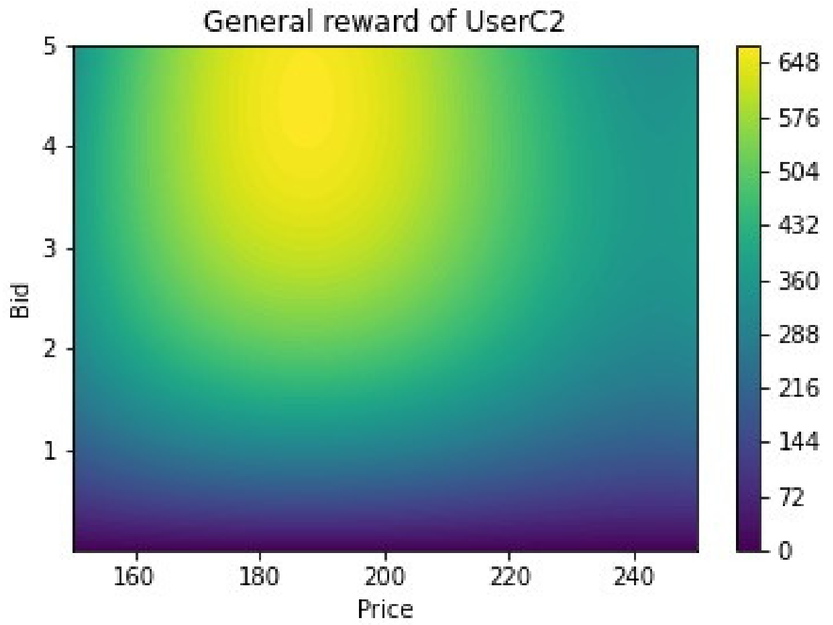
The user class is imported in the environment.
We define the round function exectuting one round of the simulation.

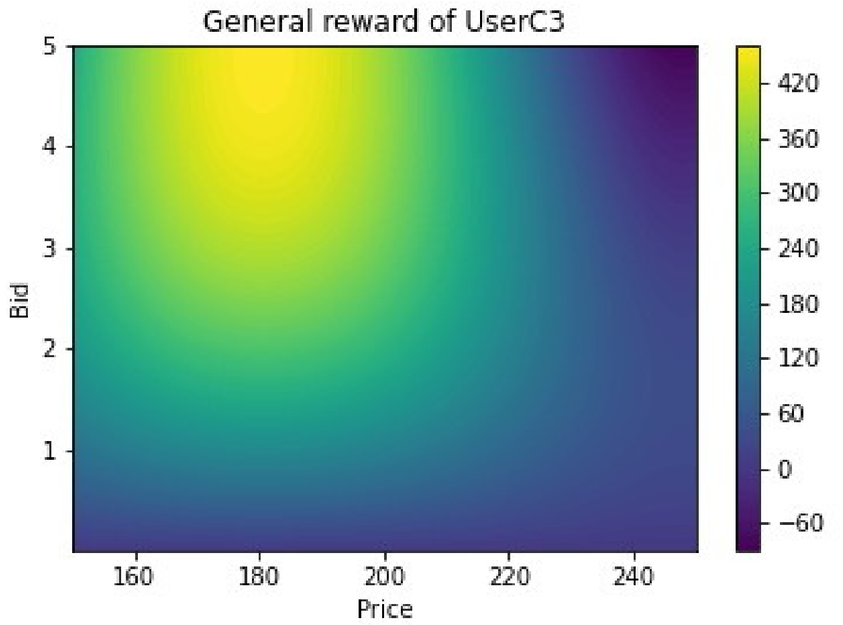
Five possible prices are considered for each user, and a Bernoulli distribution is considered for each price. A sample of the Bernoulli is drawn independently at each iteration.

The function round draws randomly from a binomial distribution. This distribution is updated with the probabilites associated to each pulled arm.

STEP 0: THE CLAIRVOYANT ALGORITHM







COLLECTOR

PARENT

YOUNG

$$R(b) = n_{ad}(b) \cdot (gain(\bar{p}) - c_{ad}(b))$$

$$gain(p) = (p - c) \cdot \alpha(p)$$

STEP I: LEARNING FOR PRICING

We consider a **single class setting**, where all customers belong to class C1, collector.

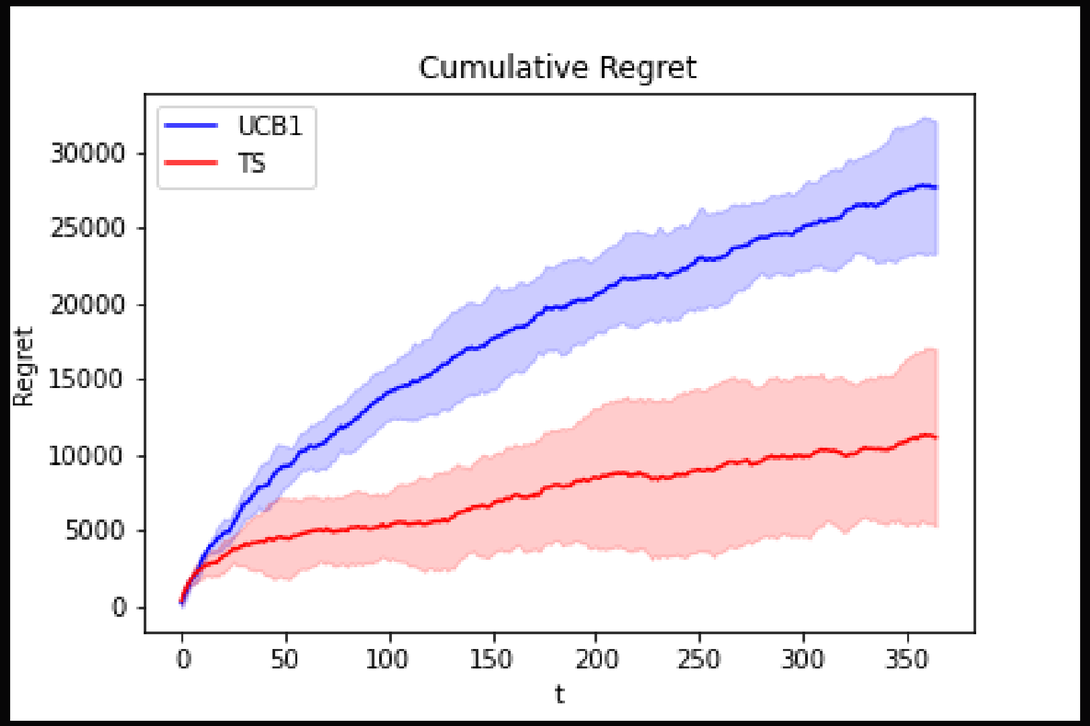
GOAL: Optimise the pricing part of the problem, given a **fixed bid** and using an online learning algorithm.

Two Multi Armed Bandits (MABs) were used to solve this problem in online fashion, minimising the regret: **UCB-1** (Upper Confidence Bound) and **Thompson Sampling**. Results were then compared in order to select the best-performing algorithm.

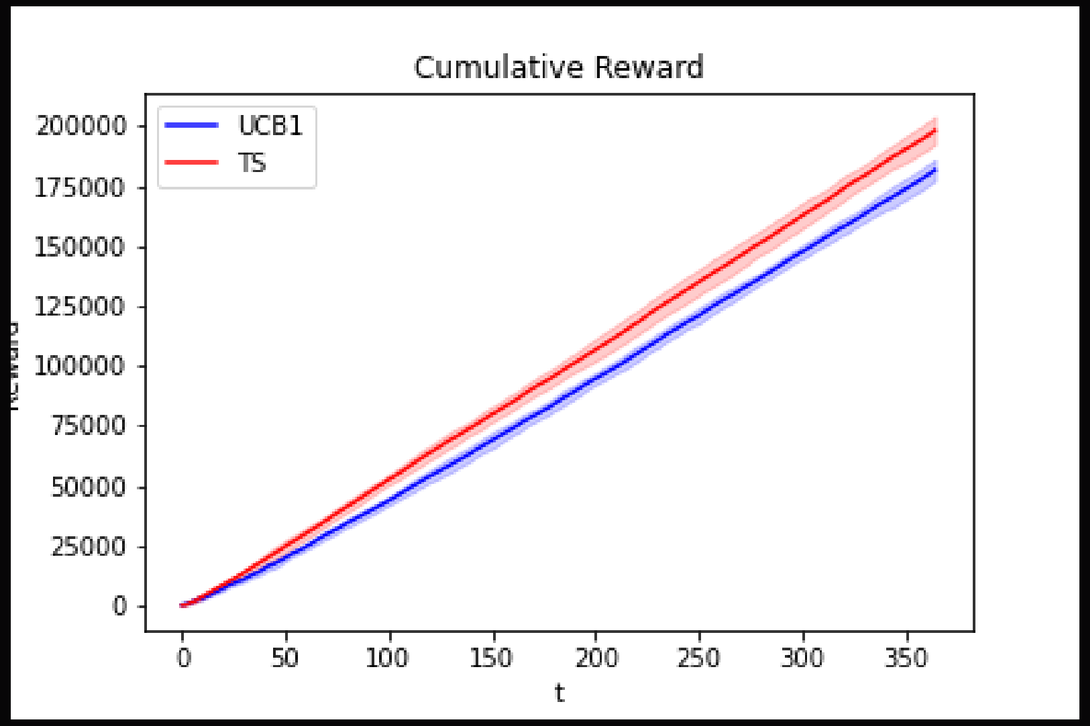
The REGRET: $R^* - R_{alg}$

The regret of a learning algorithm is simply given by the difference between the expected reward of the clairvoyant algorithm and the reward achieved by the algorithm.

RESULTS AND CONSIDERATIONS

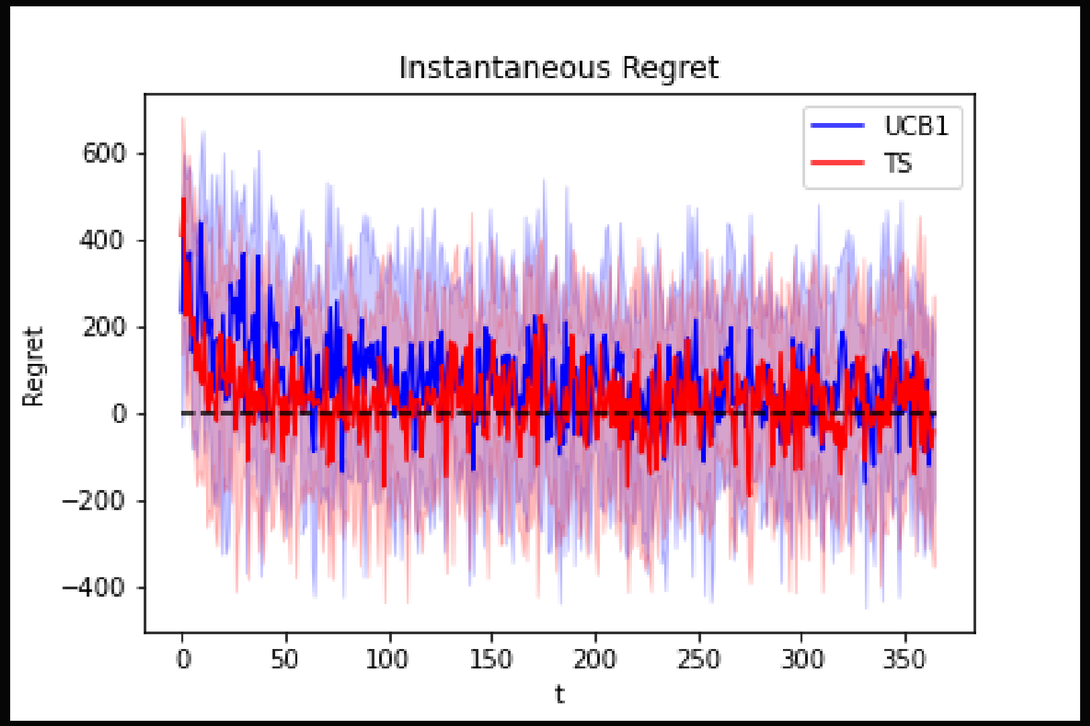


CUMULATIVE REGRET

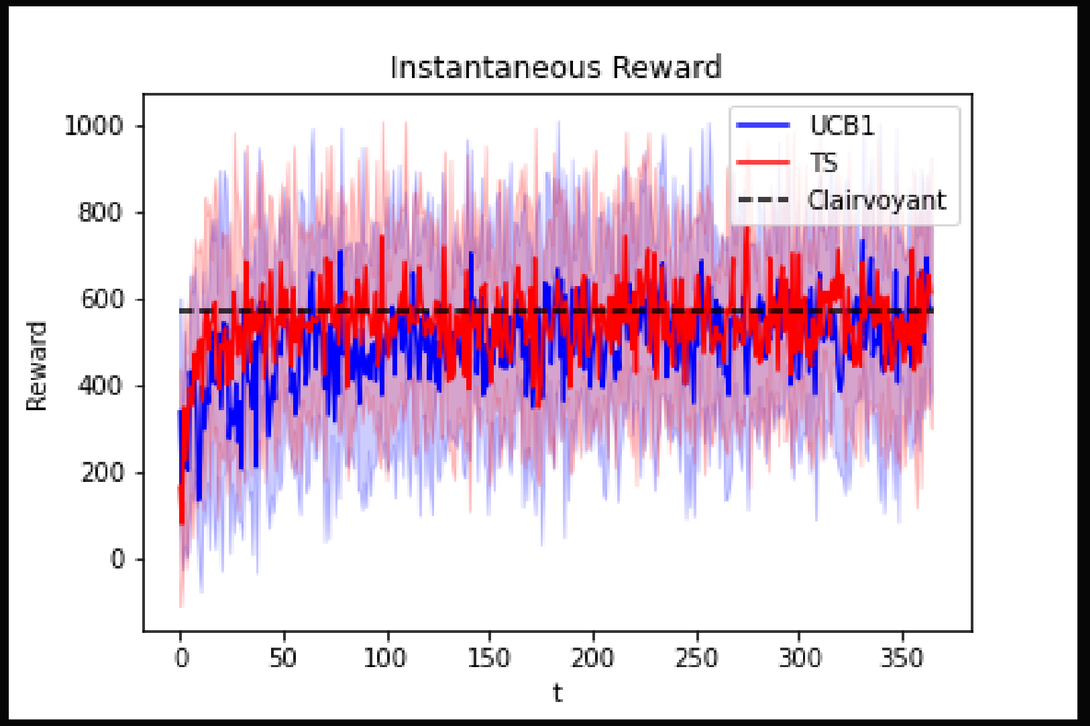


CUMULATIVE REWARD

RESULTS AND CONSIDERATIONS



INSTANTANEOUS REGRET



INSTANTANEOUS REWARD

STEP 2: LEARNING FOR ADVERTISING

We assume that the curves of the pricing problem are known and that the ones relative to the advertising problem are to be computed. We still work assuming that all customers belong to the first class: C1.

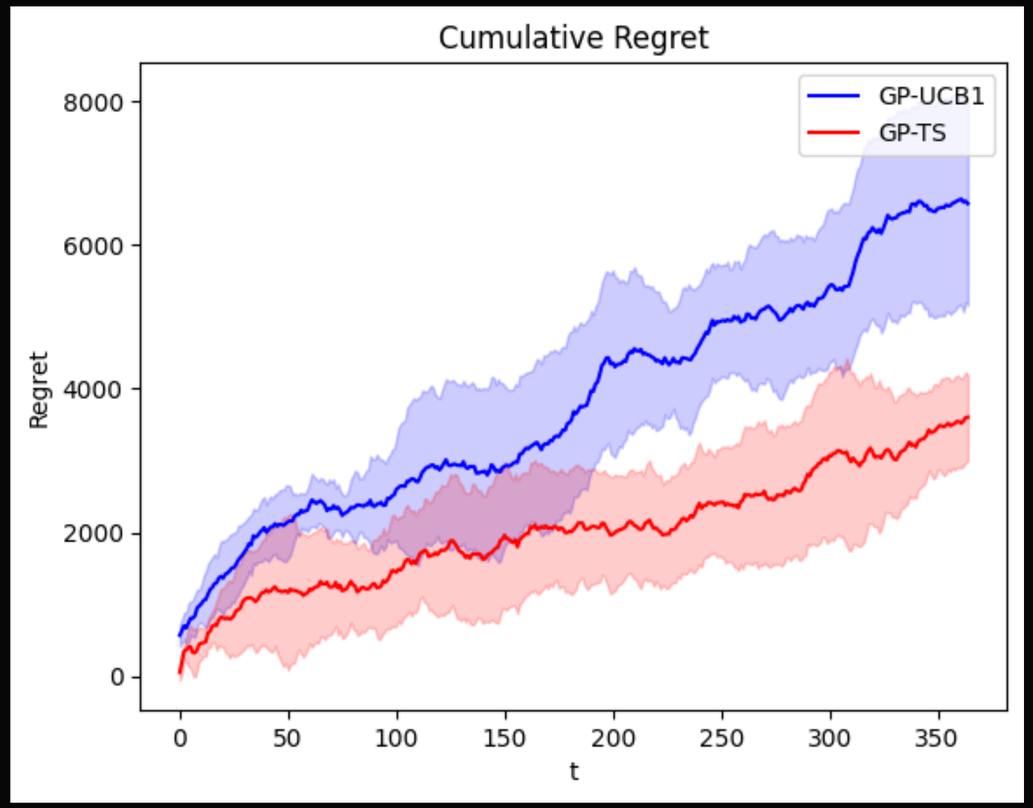
A remarkable threat when working in this setting is the considerable computational load.

SOLUTION: Resorting to **Gaussian Processes** allows us to significantly reduce the computational load, by assuming the existence of a more substantial structure explaining the distribution of the variables. We consider the collection of random variables to be such that any finite number of them is jointly distributed as a multivariate normal.

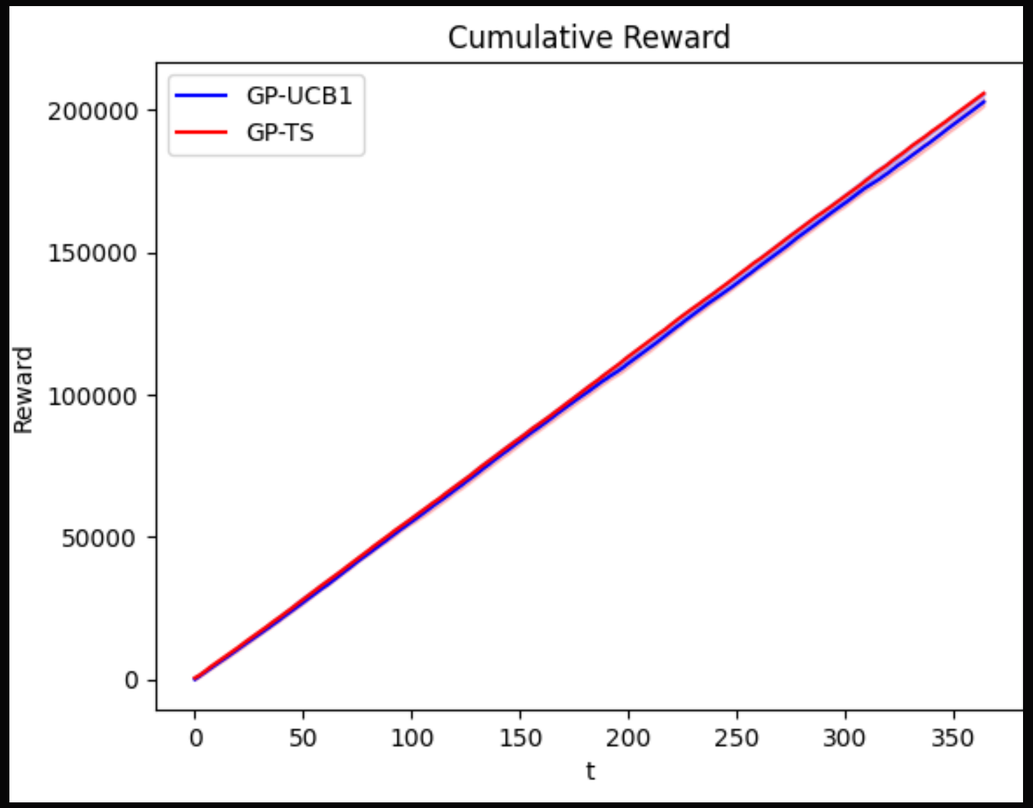
$$(X_1, \dots, X_N) \sim \mathcal{N}(\mu, \sigma^2)$$

By making quite mild assumptions on the function to approximate, Gaussian Processes (GPs) allow us to tackle the problem of regression easily and are able to return a probability distribution over the outcomes.

RESULTS AND CONSIDERATIONS

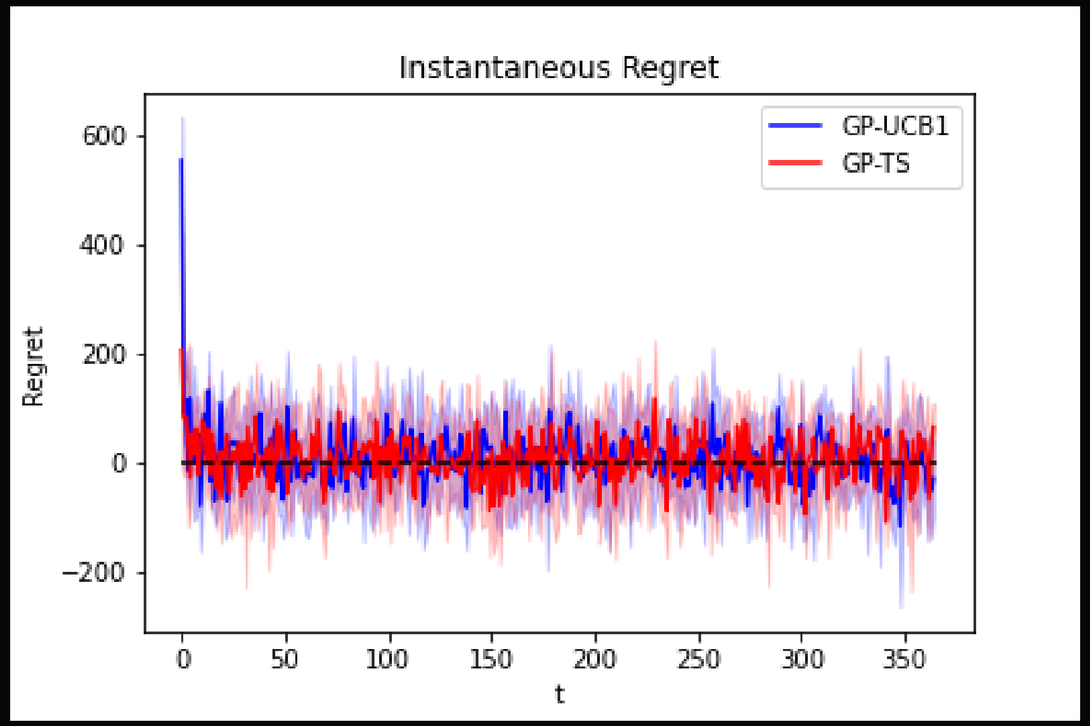


CUMULATIVE REGRET

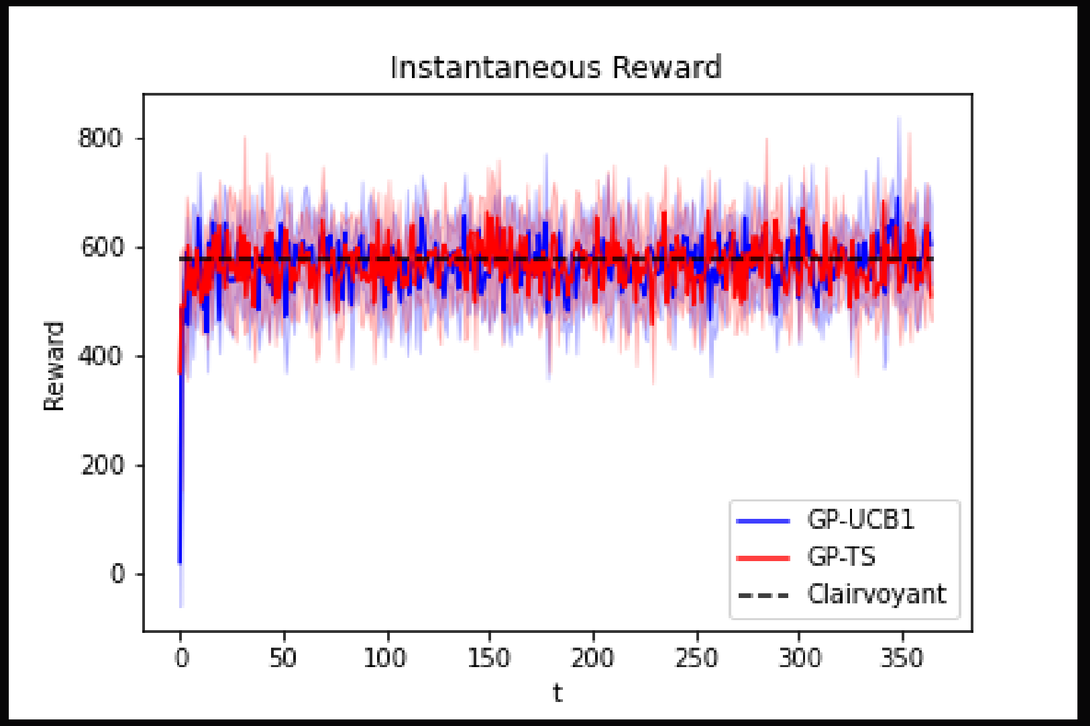


CUMULATIVE REWARD

RESULTS AND CONSIDERATIONS



INSTANTANEOUS REGRET



INSTANTANEOUS REWARD

STEP 3: LEARNING FOR JOINT PRICING AND ADVERTISING

HPs:

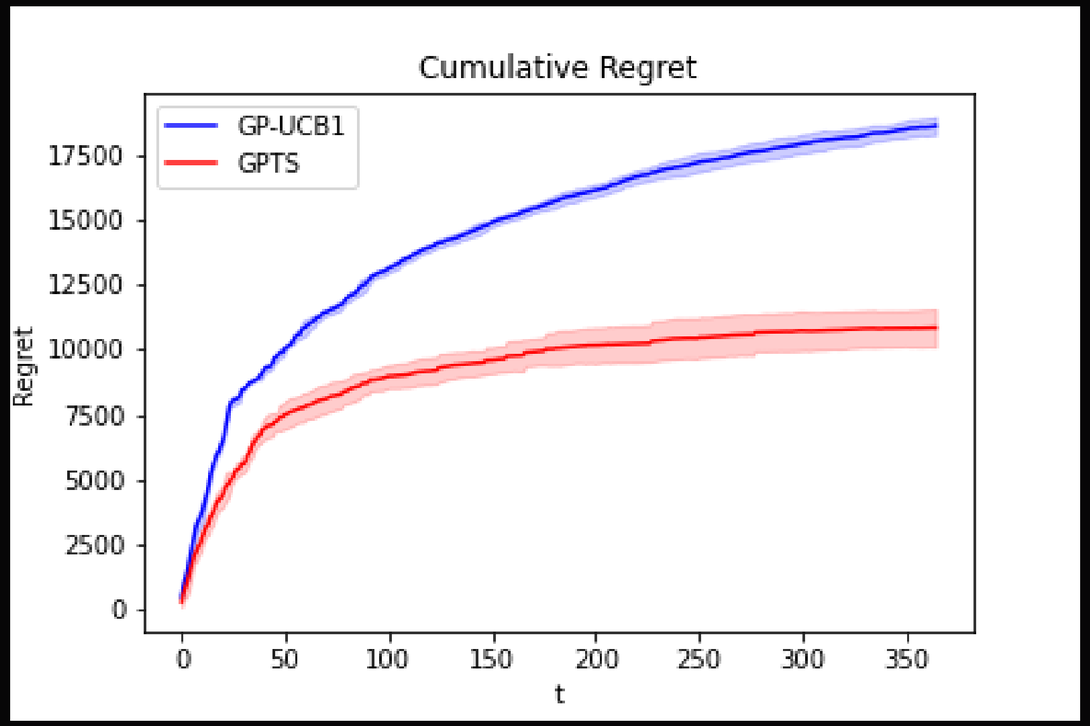
- All customers belong to **class C1**, collectors.
- Both the curves related to the pricing problem and the ones related to the advertising one are now **unknown**.

Step 3 implements a combination of the two preceding steps.
In a **common environment**, both problems are addressed simultaneously, as all curves are learnt at once.
The action space from which arms are pulled is given by **all possible combinations of bids and prices**.

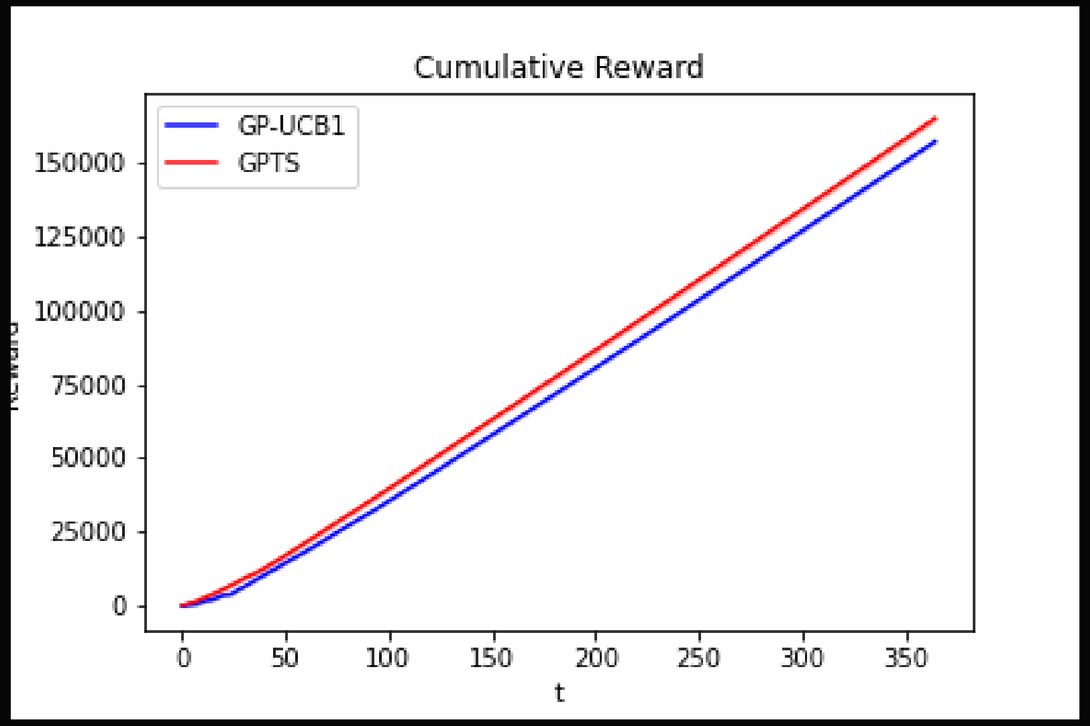
$$arm = [b, p] \qquad \begin{matrix} b \in \textbf{space of possible bids} \\ p \in \textbf{space of possible prices} \end{matrix}$$

We have an even bigger set of possible arms to pull from, making GP processes the only viable option for learning. MABs are not practical in this setting,

RESULTS AND CONSIDERATIONS

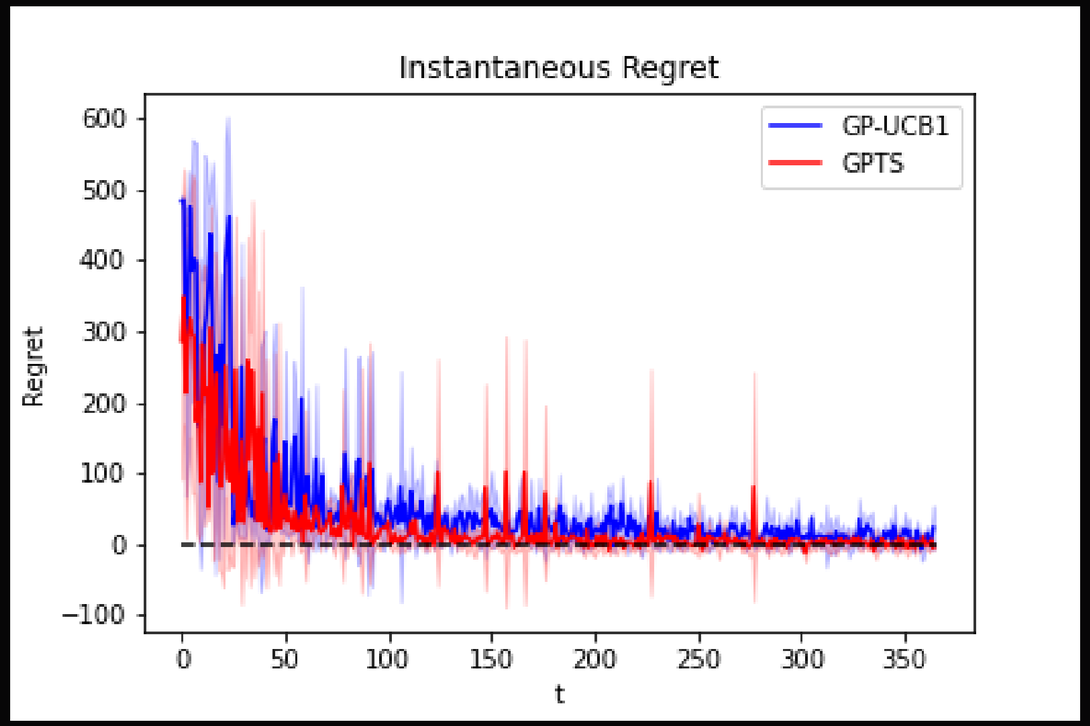


CUMULATIVE REGRET

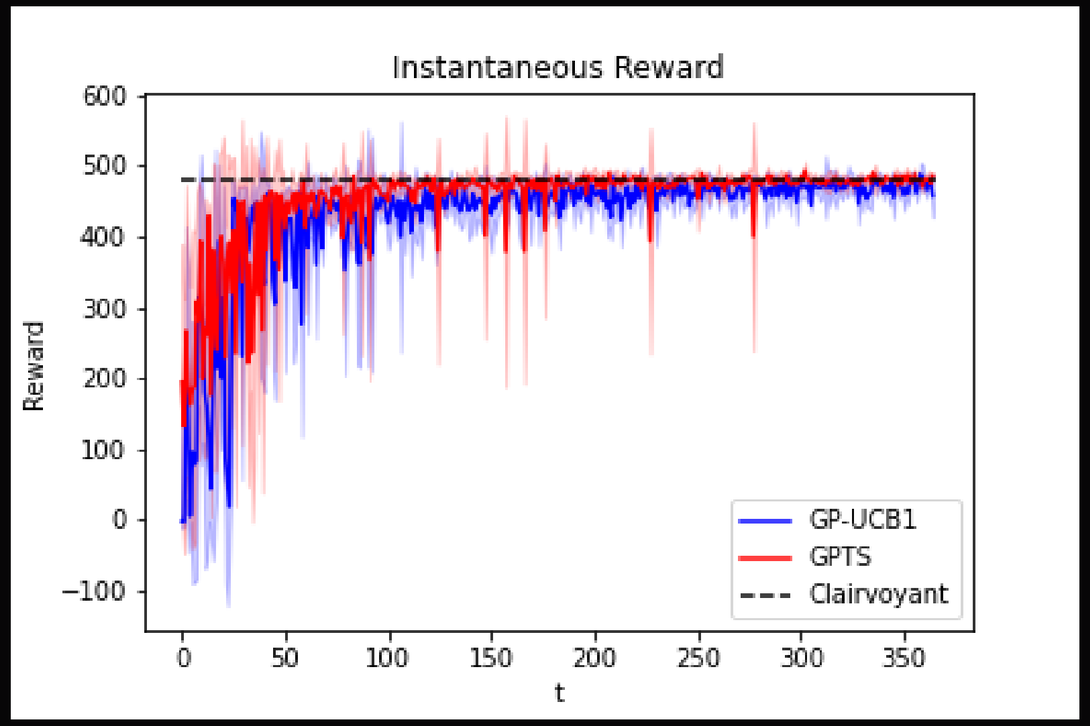


CUMULATIVE REWARD

RESULTS AND CONSIDERATIONS



INSTANTANEOUS REGRET



INSTANTANEOUS REWARD

STEP 4: CONTEXT GENERATION

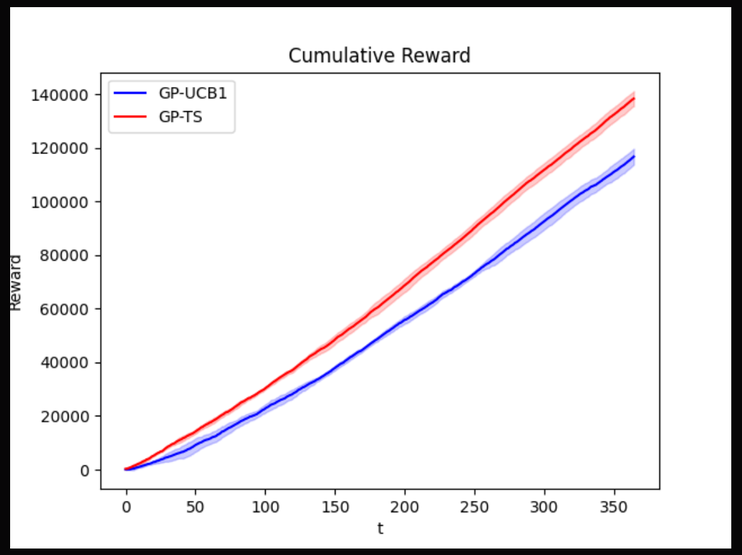
So far, we have always neglected the possibility of having different kinds of customers.
This is rarely the case: we introduce an adequate **context structure**.

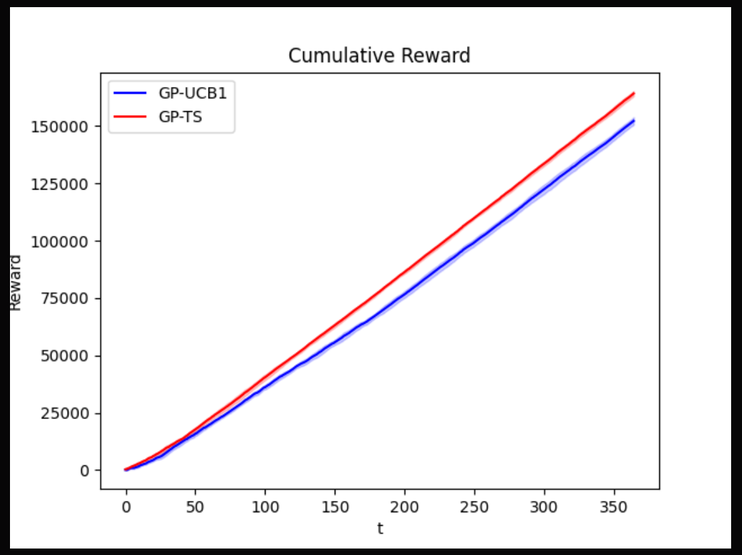
We can imagine to be dealing with **disaggregated curves**, one for each class of users. These can be combined in a unique, aggregated model by a simple weighted summation.
In our application, we assume the following weights:

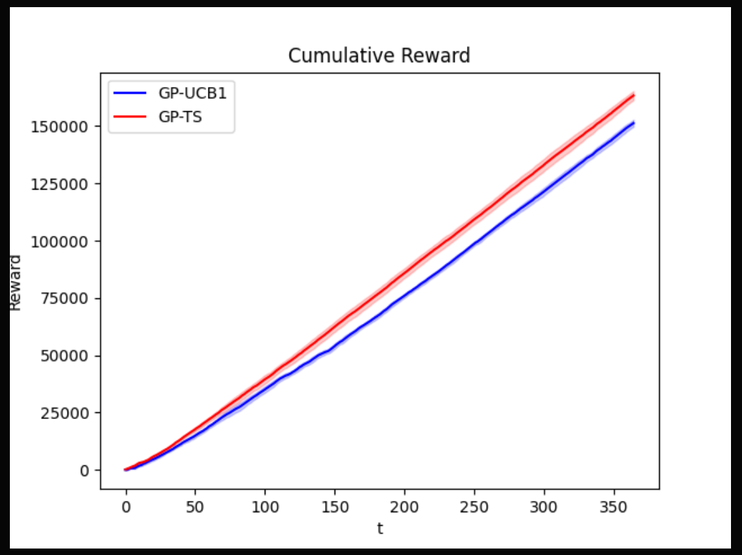
CUSTOMER	PROBABILITY
C1	0.33
C2	0.33
C3	0.33

A **context** is made up of a **collection of features**, such that each set of features univocally identifies a specific class of customers. The collections are such that pairwise intersection of sets is the null set and union of all sets is the total set of features.

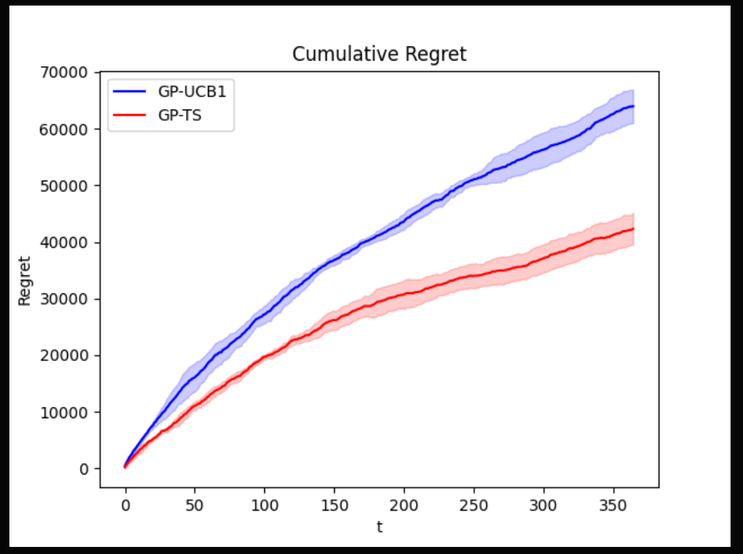
CUMULATIVE REWARD

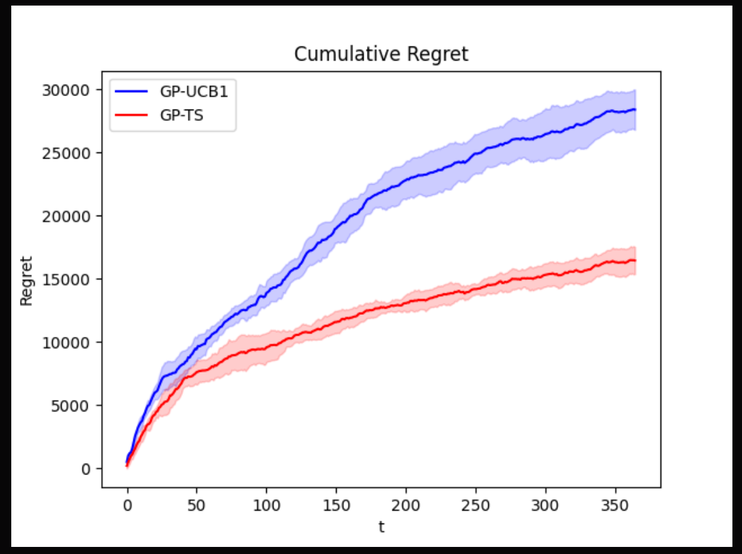


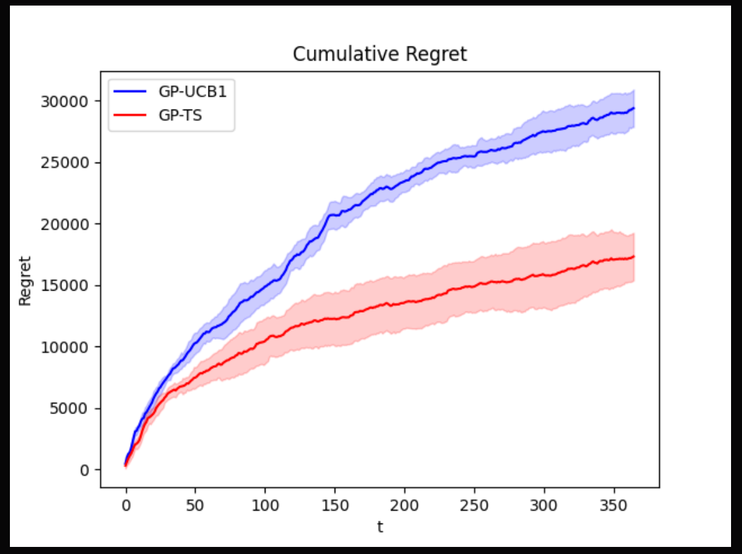




CUMULATIVE REGRET





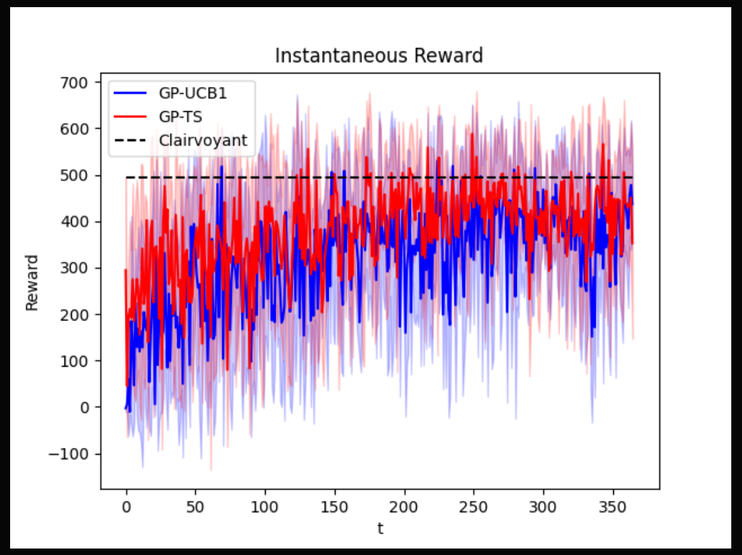


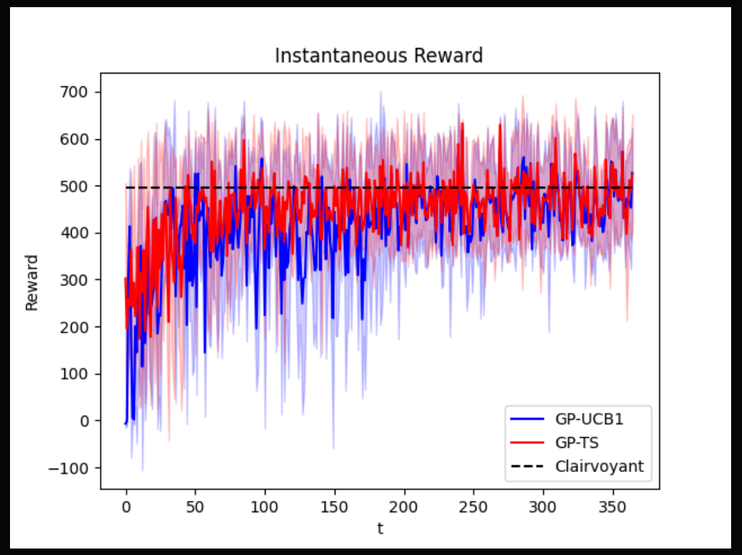
Known context

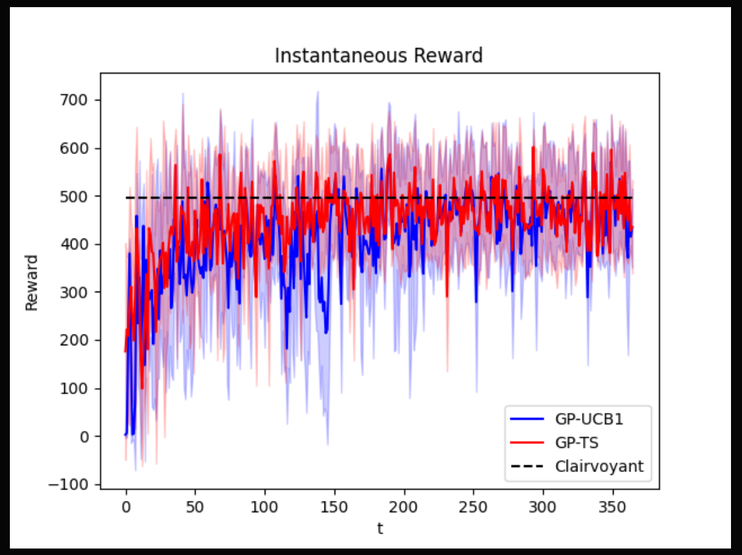
Without context

Unknown Context

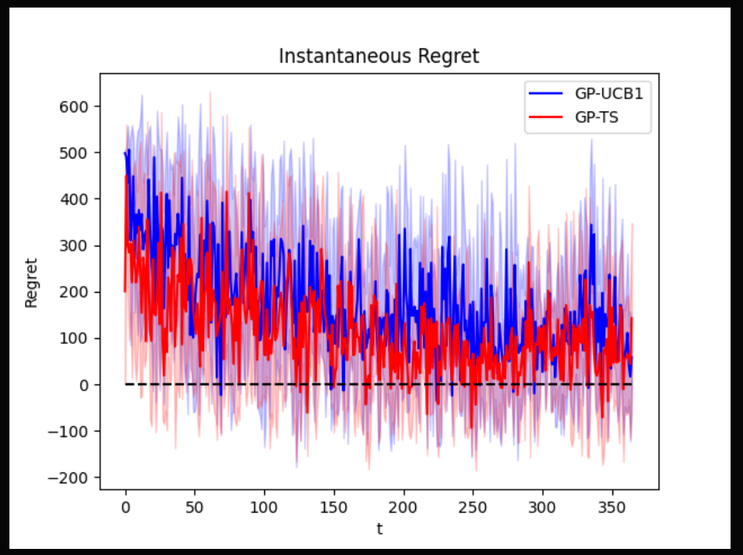
INSTANTANEOUS REWARD

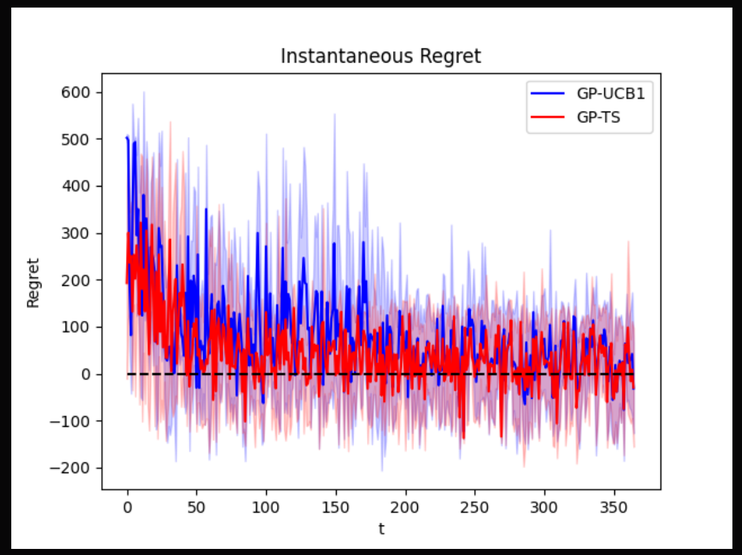


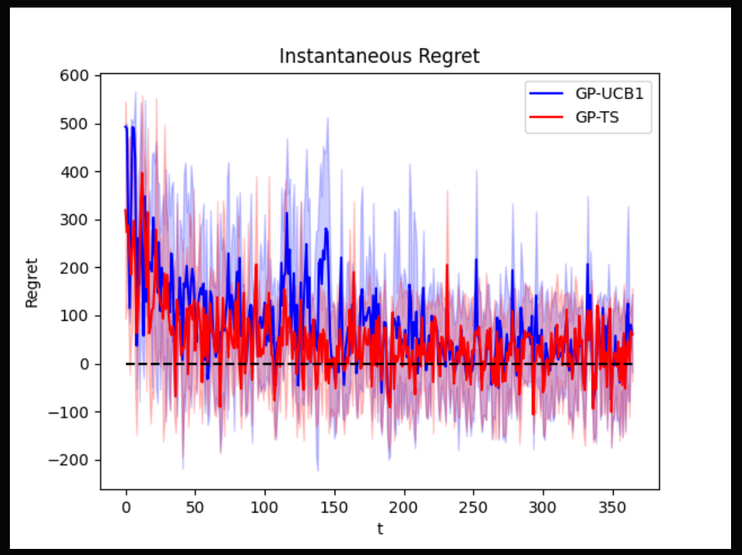




INSTANTANEOUS REGRET







Known context

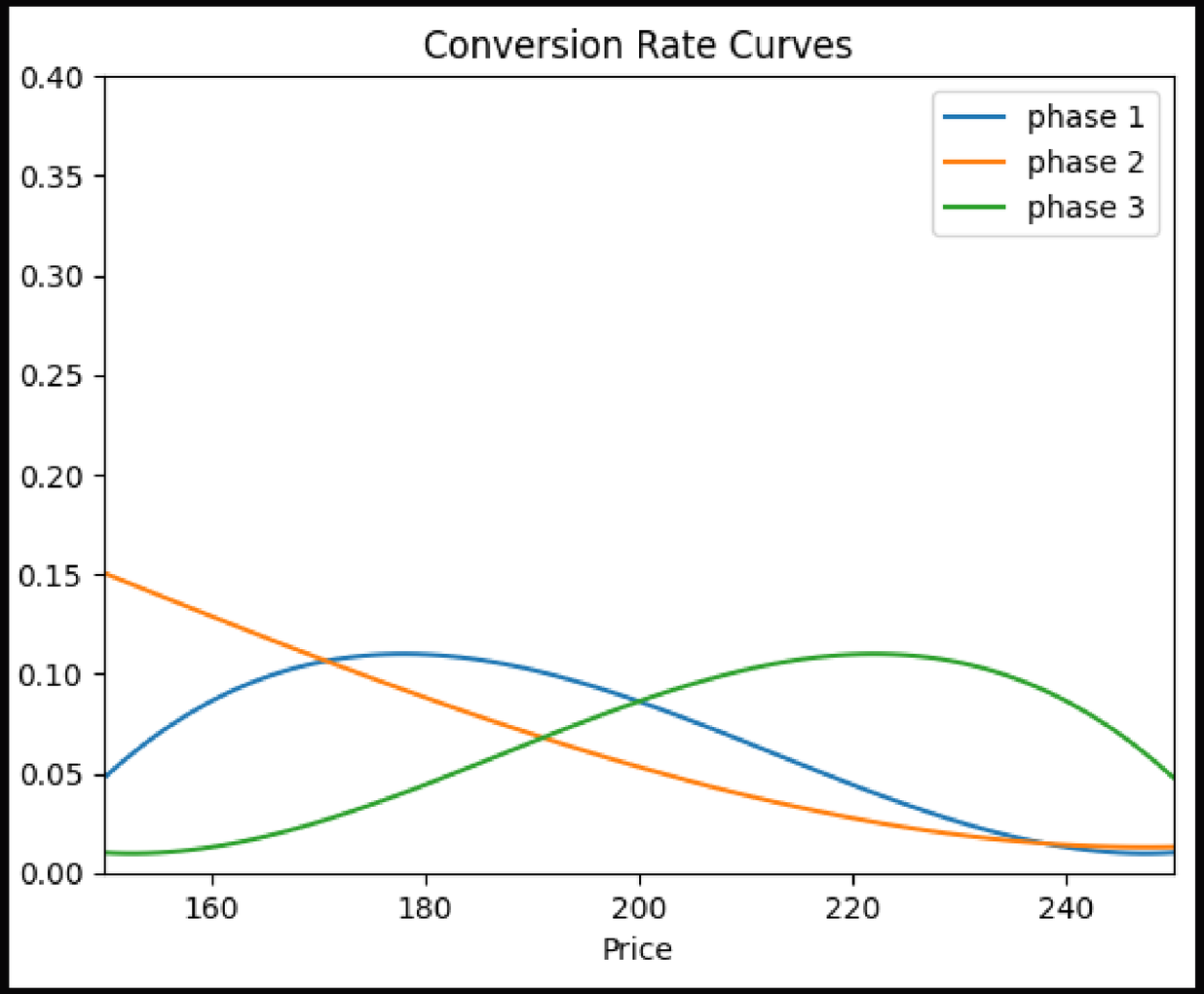
Without context

Unknown Context

STEP 5: DEALING WITH NON-STATIONARITY

SCENARIO 1: TWO ABRUPT CHANGES

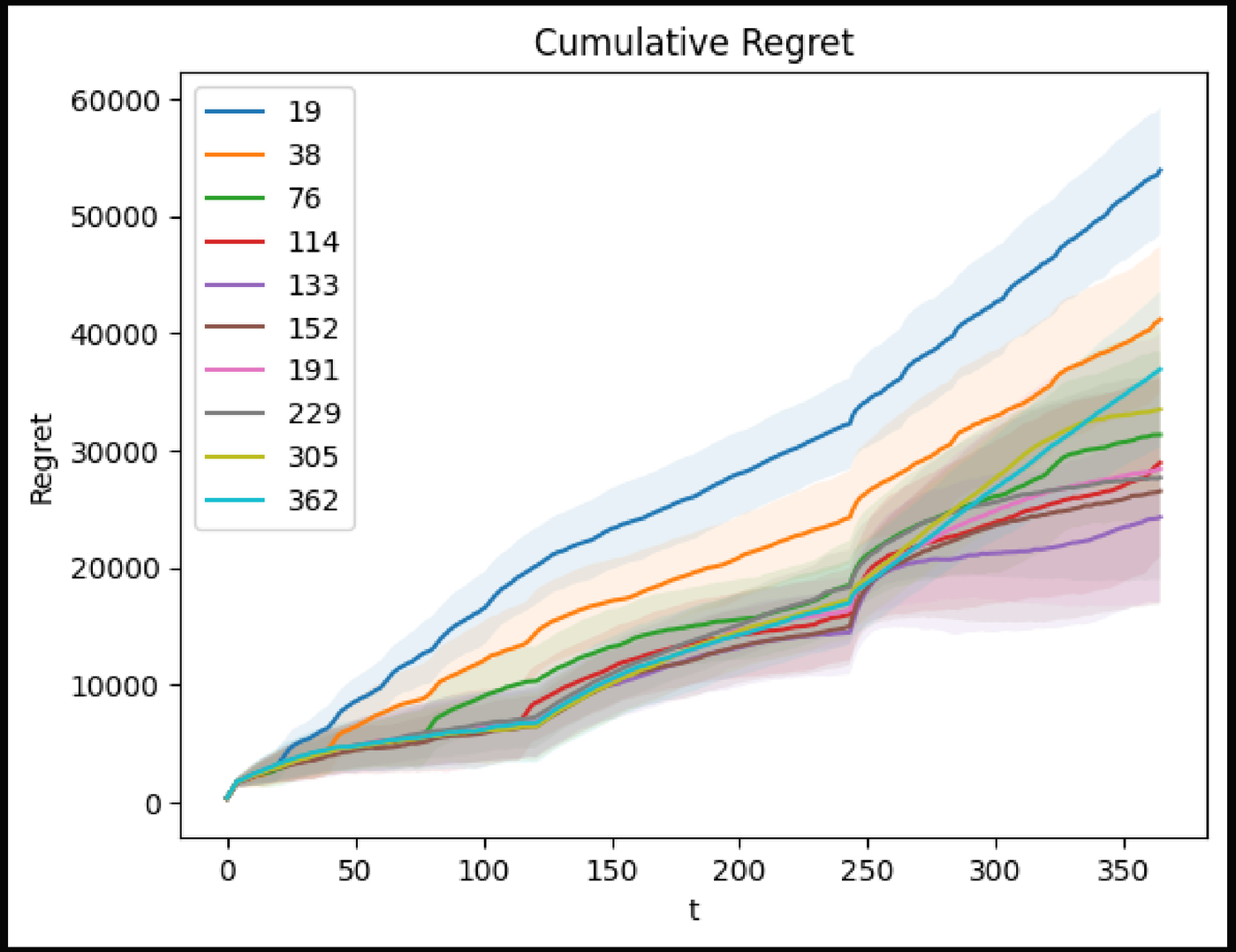
Simplified setting: **single-user class** (C1), curves related to the advertising problem are known.
We assume **three total phases** throughout the year.



- **Phase 1**: normal phase, the market is its regime;
- **Phase 2**: stagnation phase during summer months. Lower sales are experienced;
- **Phase 3**: rise in market activity due to limited edition releases and Christmas shopping.

SLIDING WINDOW UCB-I: SENSITIVITY ANALYSIS

Window size affects the **exploration-exploitation trade-off** and is the hyperparameter to be tuned and tested when performing sensitivity analysis.



- **Smaller window size** favours **exploration;**
- **Larger window size** encourages **exploitation** of the best arms.

Window size yielding lowest cumulative regret:

$$W = 7\sqrt{T}$$

CUSUM UCB-1: SENSITIVITY ANALYSIS

Four hyper-parameters:

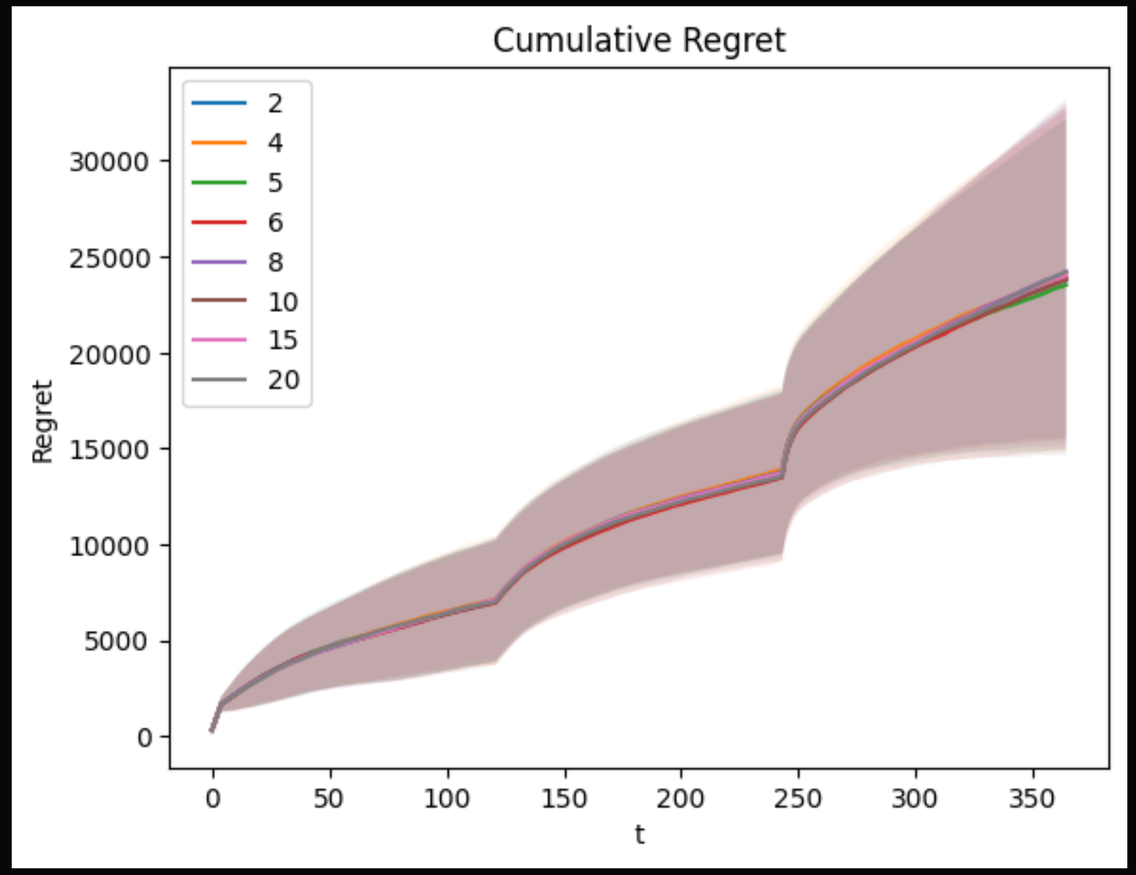
- **Number of samples** for the computation of the initial reference values. Once **M** samples have been observed, the algorithm starts looking for possible changes;
- **alpha** is the probability of randomly selecting a suboptimal arm at the current round. Bigger values of alpha increment exploration capabilities of the algorithm, to the loss of the exploitation of the best arms;
- **epsilon** is proportional to the **sensitivity to changes**. It is used to adjust the difference between the reference mean and a new sample. The bigger its value, the more changes need to happen for the detection mechanism to fire;
- **h** denotes the **change detection threshold**: higher values of h will lead to needing bigger fluctuations in the reward distribution for a change to be detected.

CUSUM UCB-I: SENSITIVITY ANALYSIS

TESTING ON M

Fixed parameters:

$\alpha = 0.001, \epsilon = 0.04, h = 0.1$

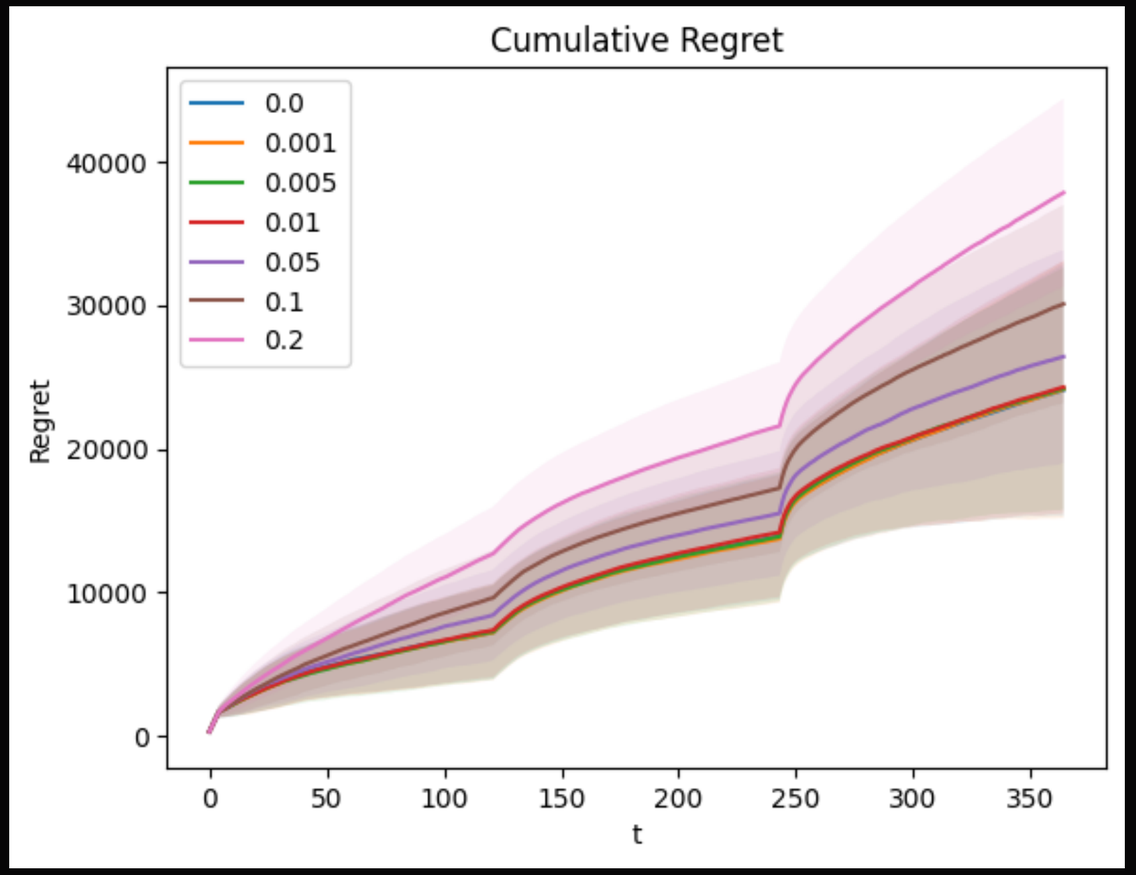


Empirically, CUSUM seems to be quite robust with respect to the choice of M. For this reason, a median choice, M = 5, was made.

TESTING ON ALFA

Fixed parameters:

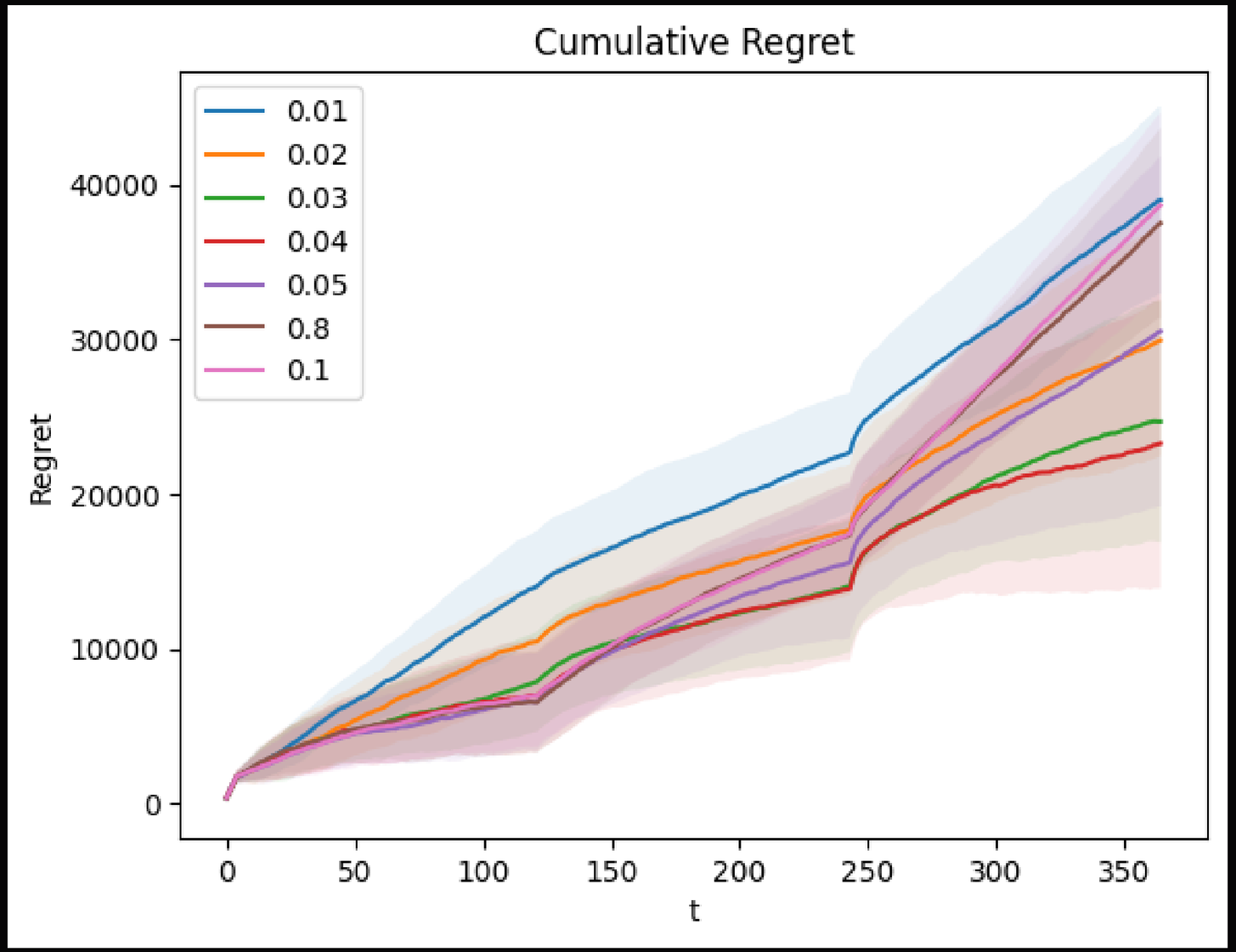
$M = 5, \epsilon = 0.04, h = 0.1$



The optimal value found was equal to 0.001, which is quite low. This suggests that exploitation is to be preferred over exploration despite the changes in distribution introduced.

CUSUM UCB-I: SENSITIVITY ANALYSIS

TESTING ON EPSILON



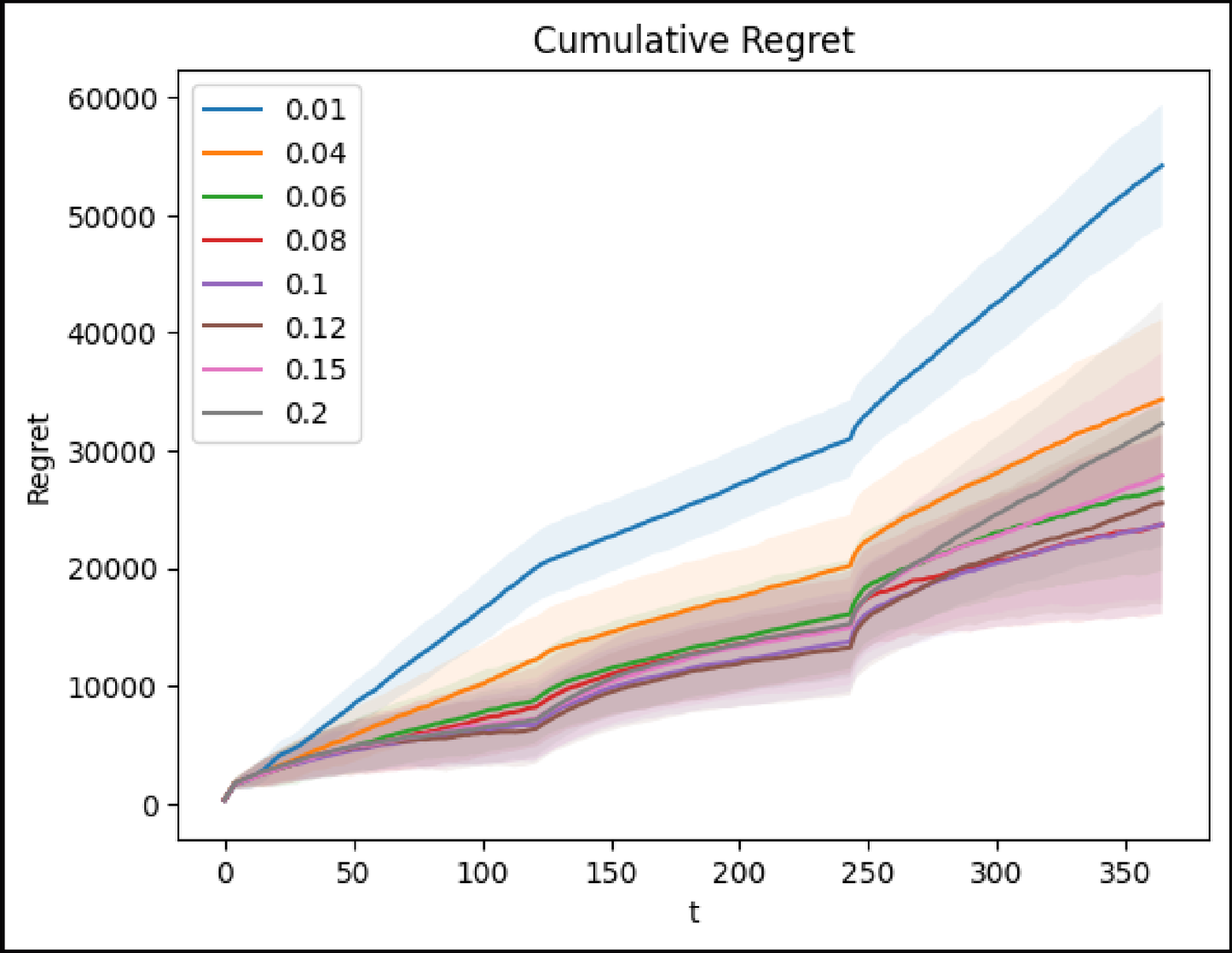
Fixed parameters:

$$M = 5, \alpha = 0.001, h = 0.1$$

Optimal value: between **0.03** and **0.04**

CUSUM UCB-I: SENSITIVITY ANALYSIS

TESTING ON THE THRESHOLD h



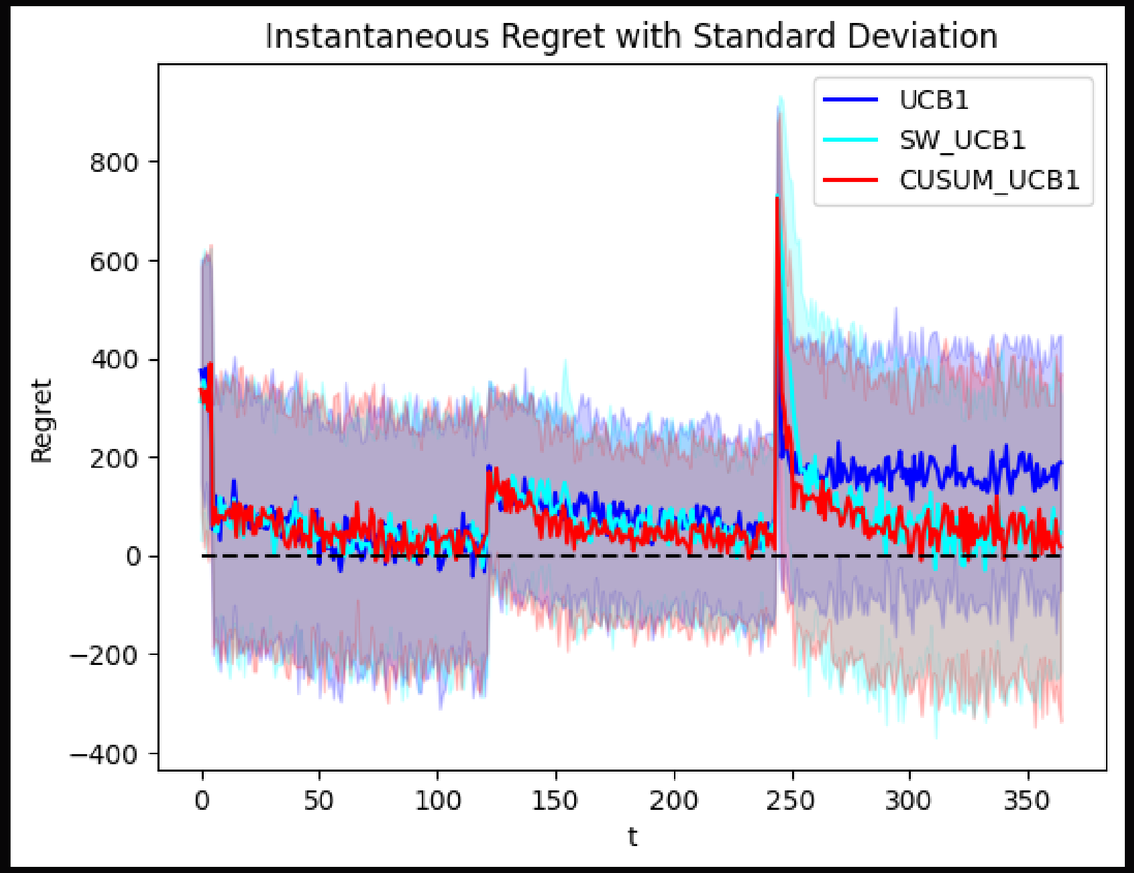
Fixed parameters:

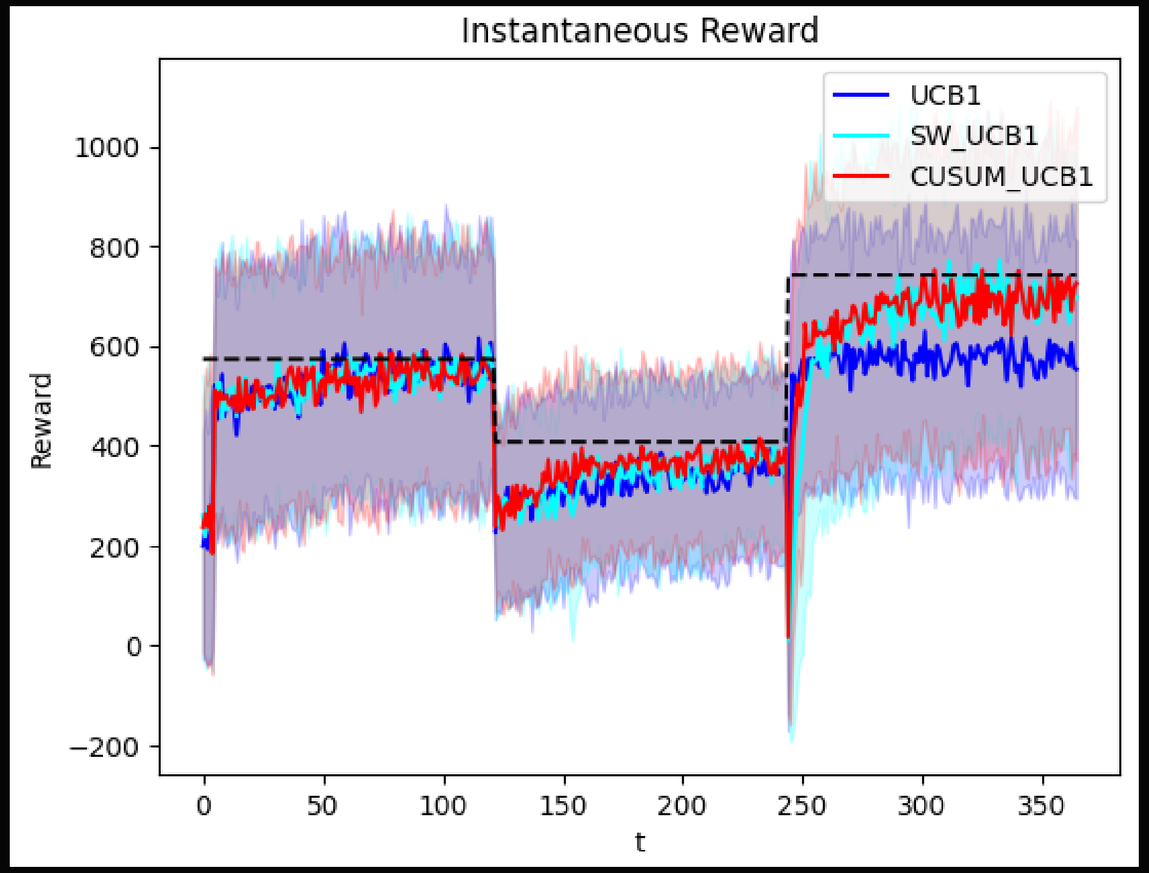
$$M = 5, \alpha = 0.001, \epsilon = 0.04$$

Optimal value: **0.1**

COMPARIMG THE ALGORITHMS

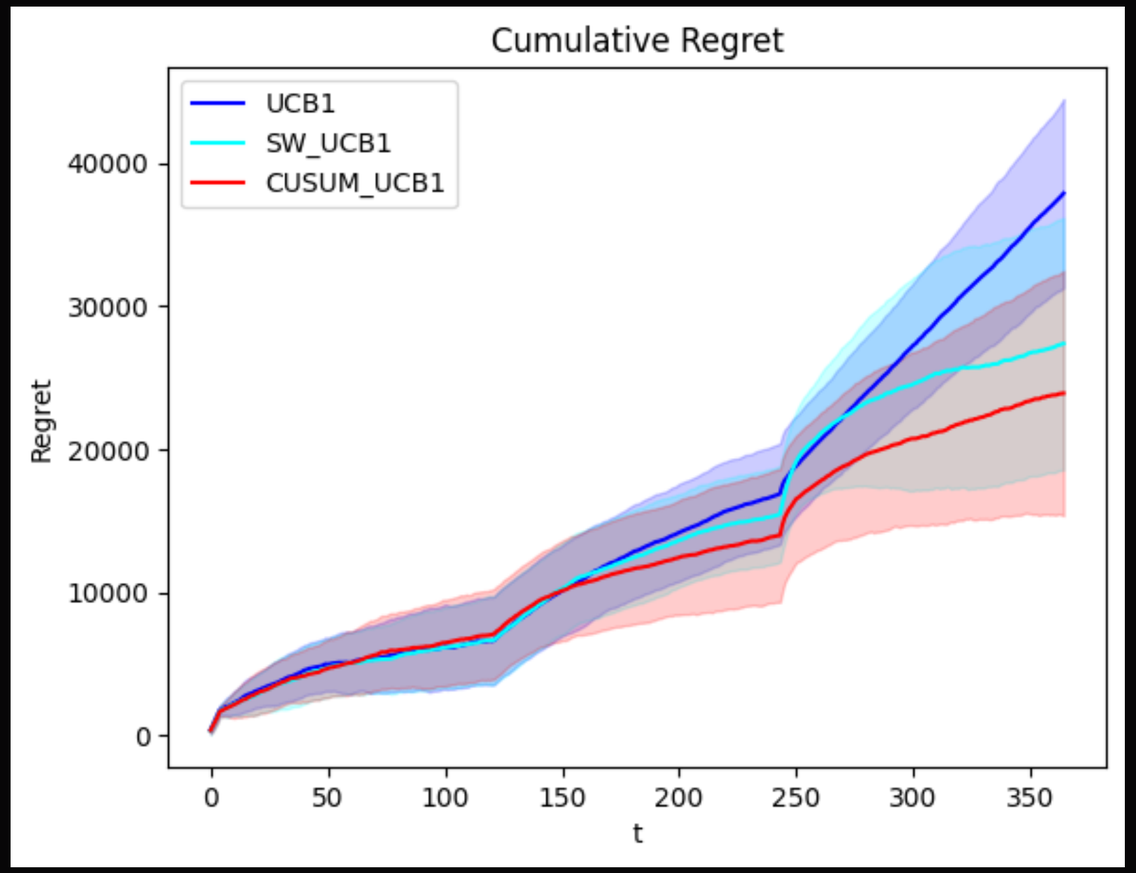
By reasoning only, we expect the UCB-1 algorithm, in its most standard form, to be the least performing of the three. This is due to the fact that it has **no way of accounting for possible variations in the reward distribution**, since it is implemented on the assumption of working in a static environment. On the other hand, we expect CUSUM to be the version of UCB1 yielding the best results, since it should be able to **actively adapt to the changes in the reward distribution**, instead of passively following the same approach like the sliding window variant does.

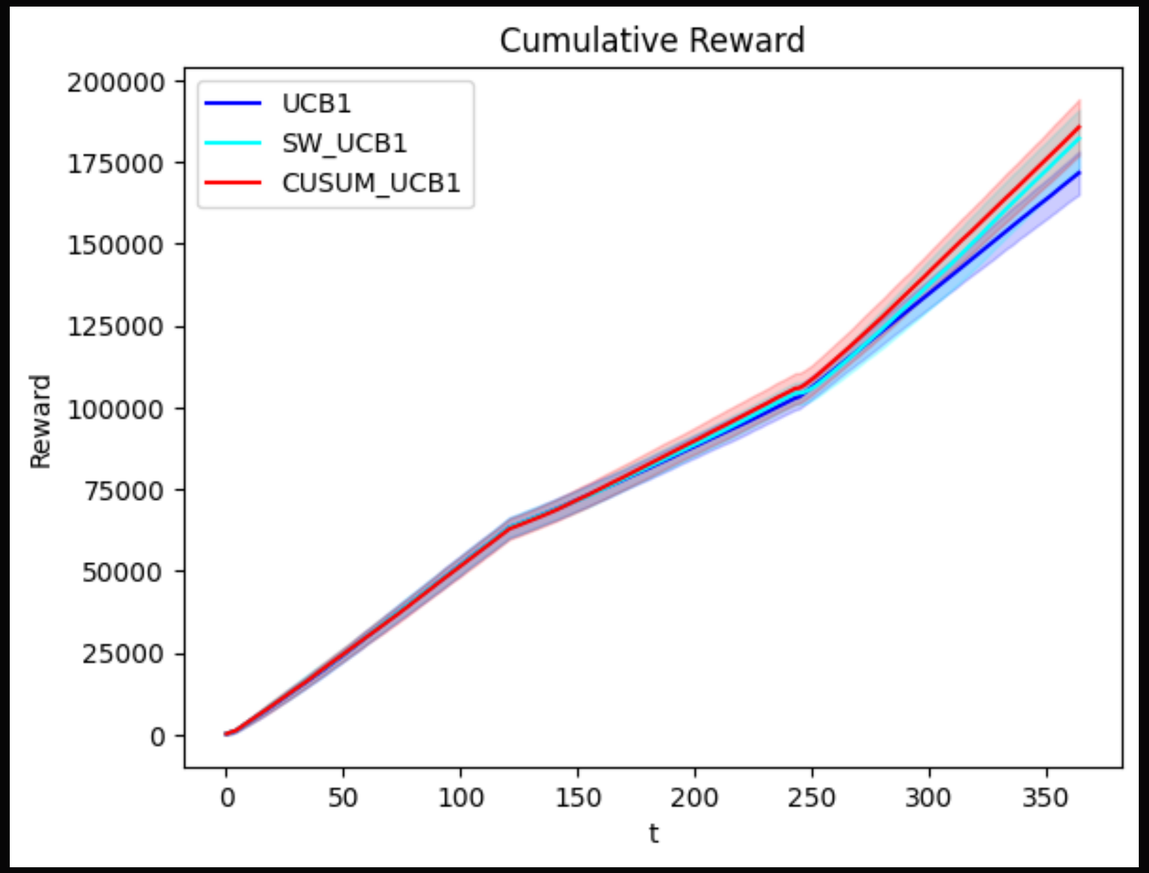




COMPARIMG THE ALGORITHMS

Our results are in line with theoretical expectations: CUSUM yields the lowest cumulative regret, proving to be the best at dealing with abruptly changing environments, followed by the sliding window (SW) UCB variant and finally by UCB1, whose regret explodes in the third phase of learning. CUSUM achieves also the greatest cumulative reward.





STEP 6: DEALING WITH NON-STATIONARITY

SCENARIO 2: MANY ABRUPT CHANGES

Our final goal consists in implementing the **EXP3 algorithm**, in a simplified setting where the bid is fixed and, once again, the only customer class considered is C1. EXP3 stands for **Exponential-weight algorithm for Exploration and Exploitation**. It works by maintaining a list of weights for each of the actions, using these weights to decide randomly which action to take next, and increasing (or decreasing) the relevant weights when a payoff is good (or bad).

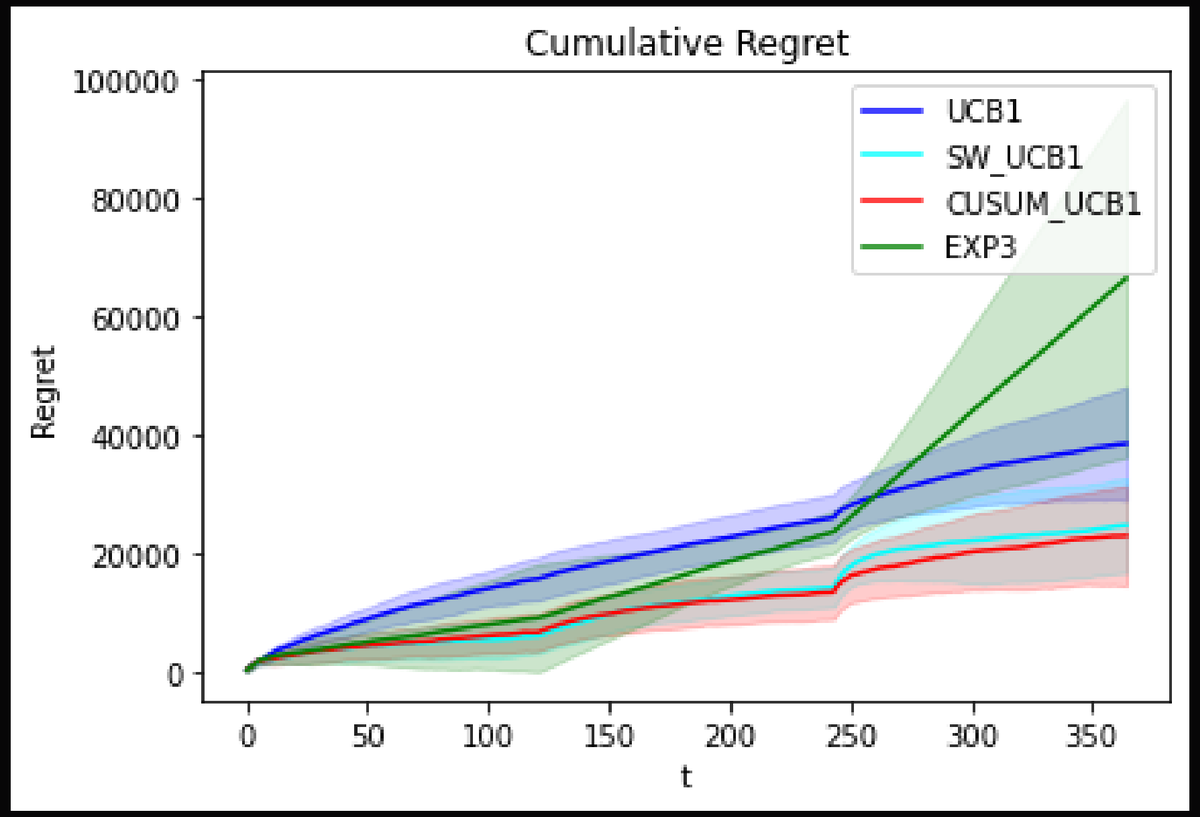
EXP3 introduces the egalitarianism factor, denoted by $\gamma \in [0, 1]$, which balances the importance given to the computed weights. In other words, it tunes the desire to pick an action uniformly at random.

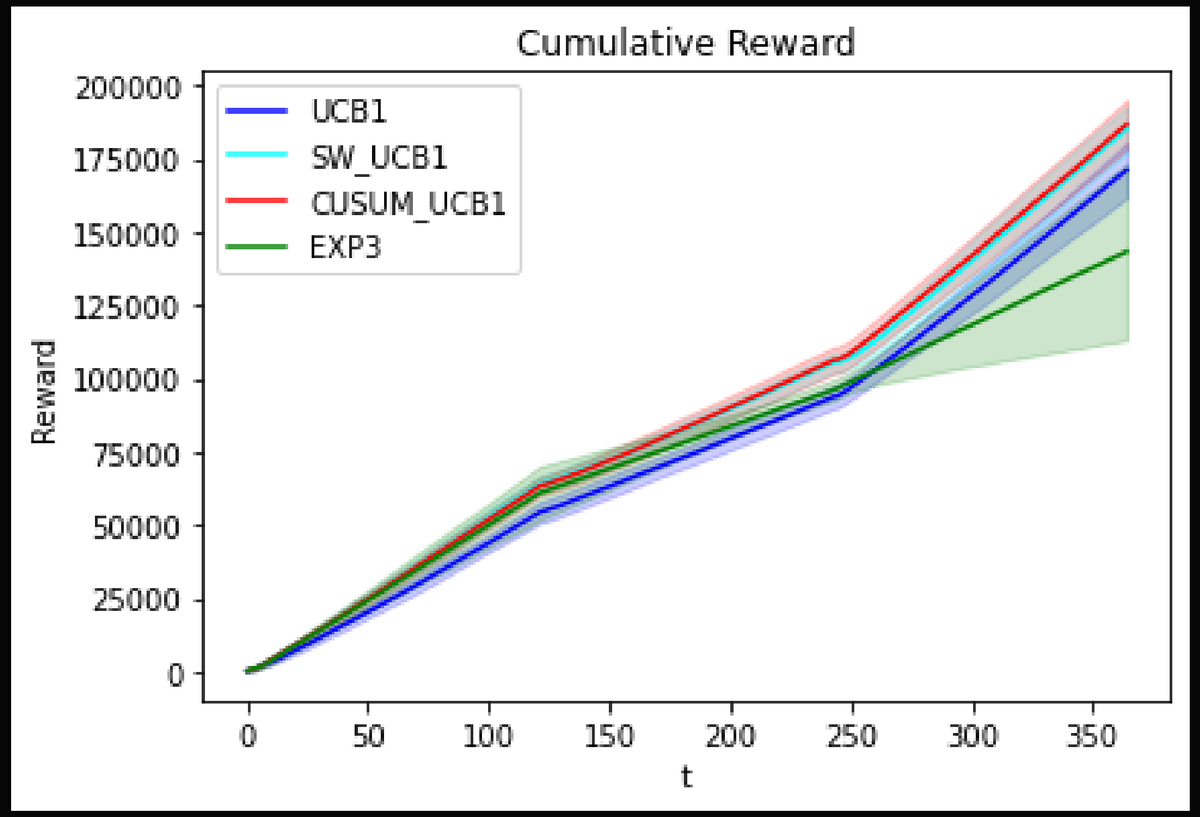
Taking the limit cases:

- if $\gamma = 0$, the algorithm will always select arms according to what dictated by the importance weights;
- if $\gamma = 1$, the weights have no effect on the choices at any step.

EXP3 works particularly well in **adversarial settings**, that is all contexts in which an opponent (the adversary) operates against the player by trying to select arm probabilities in such a way to minimise the utility. When variations in the reward distribution become unrestricted we can model the scenario as an adversarial setting, which is what we will be doing in this final step.

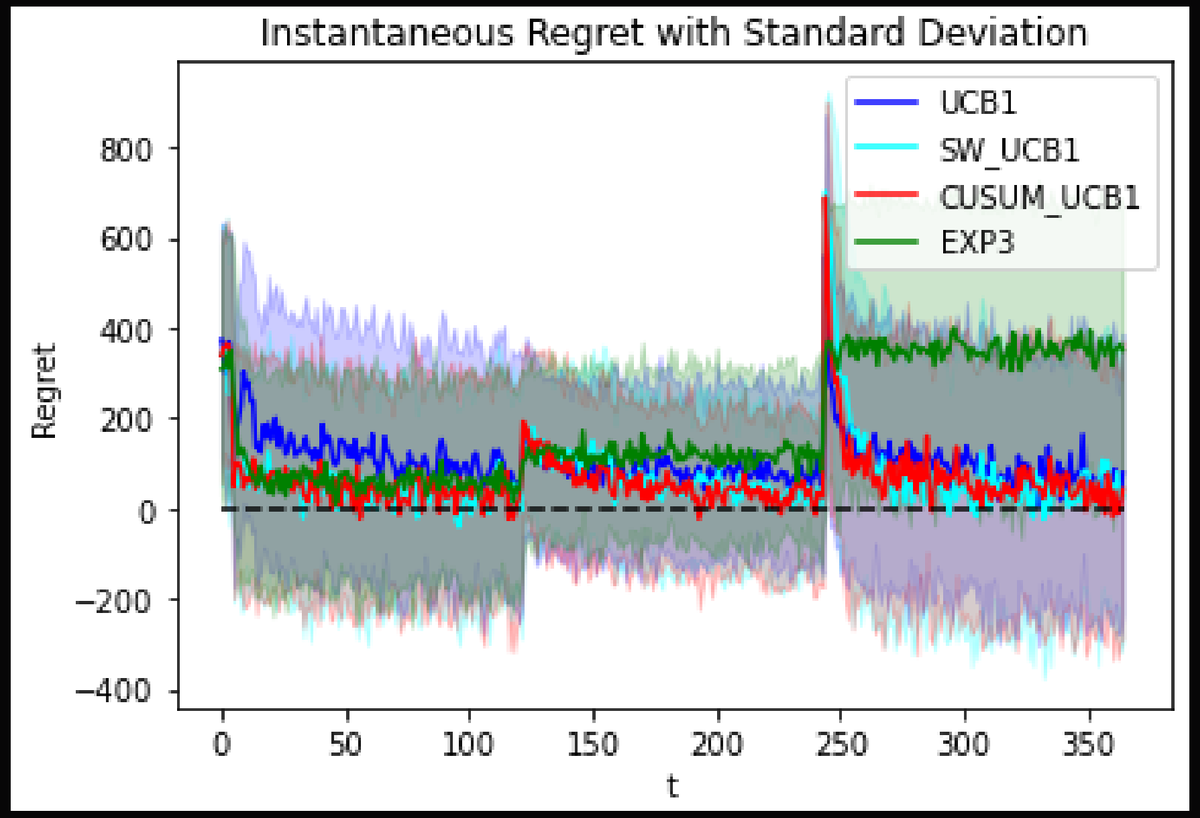
EXP3: 3 PHASES

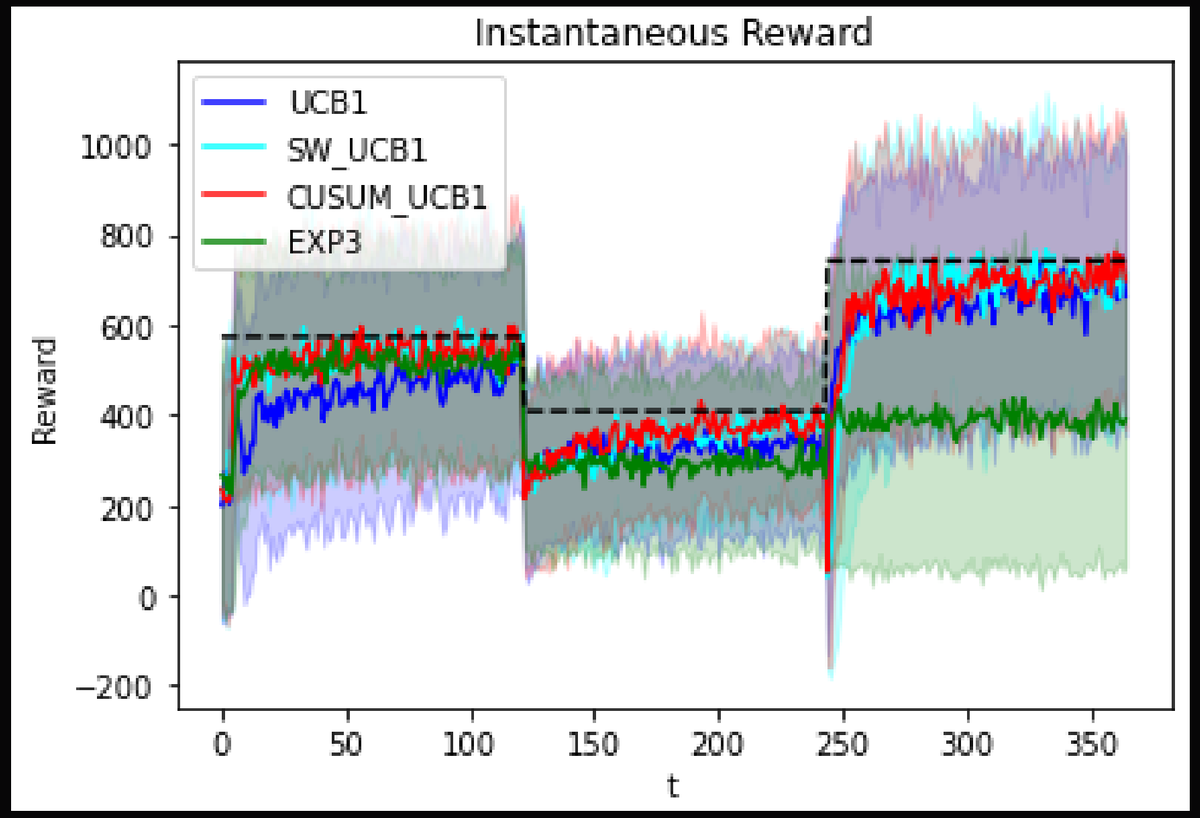




We decided to benchmark EXP3 against the optimal fits for UCB1 in its three variants (stationary, sliding window SW-UCB1 and CUSUM UCB1). This was done in a scenario of 3 total phases, caused by two abrupt changes. What we expected from theory is confirmed by our experimental results: indeed, EXP3 is outperformed both in terms of total cumulative regret (highest of the four alternatives) and in terms of cumulative reward (lowest).

EXP3: 3 PHASES

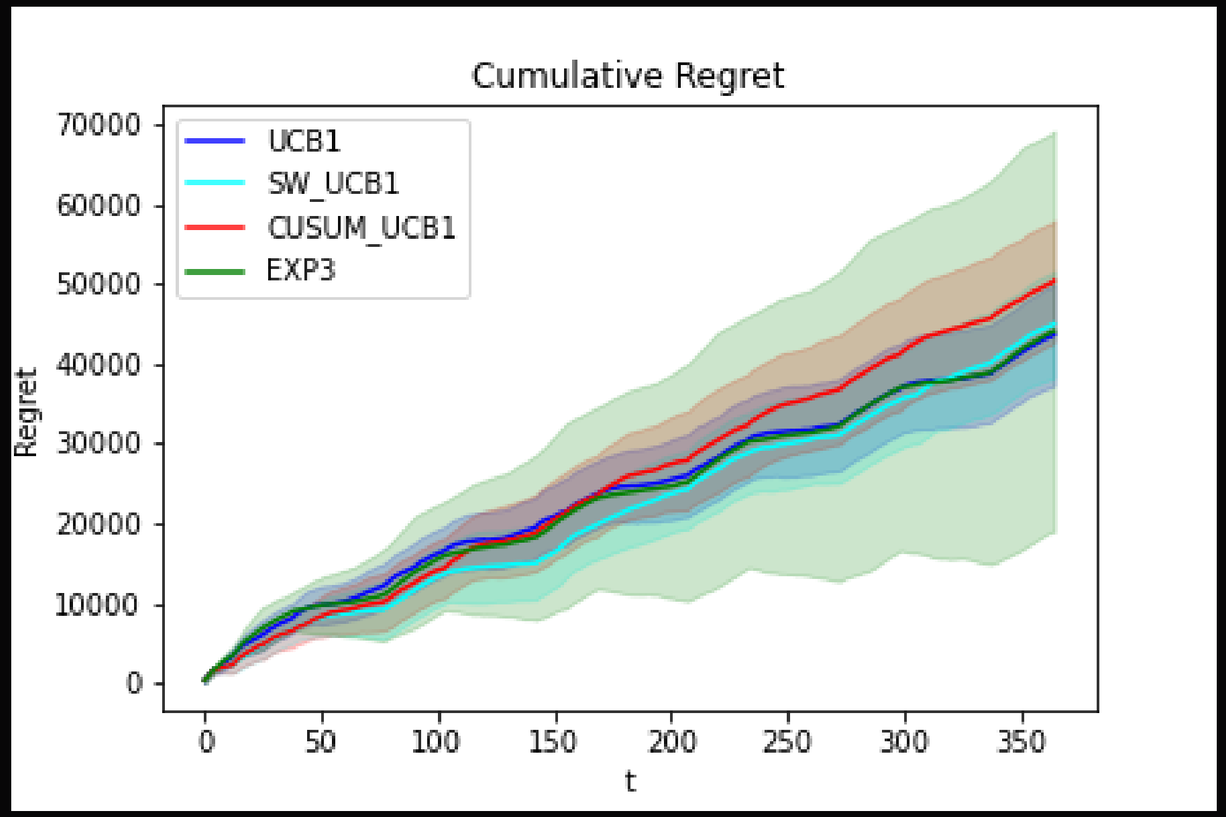


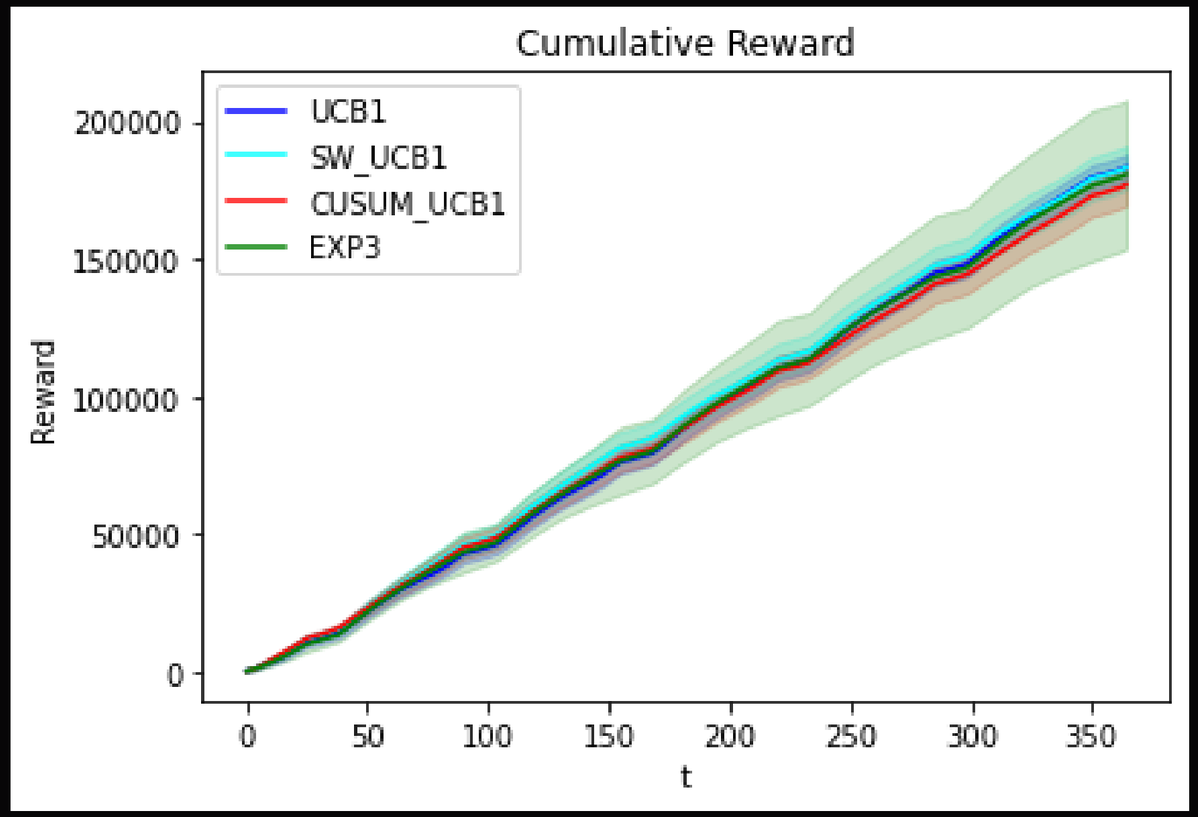


We also provide comparisons regarding instantaneous regret and instantaneous reward.

EXP3: 25 PHASES

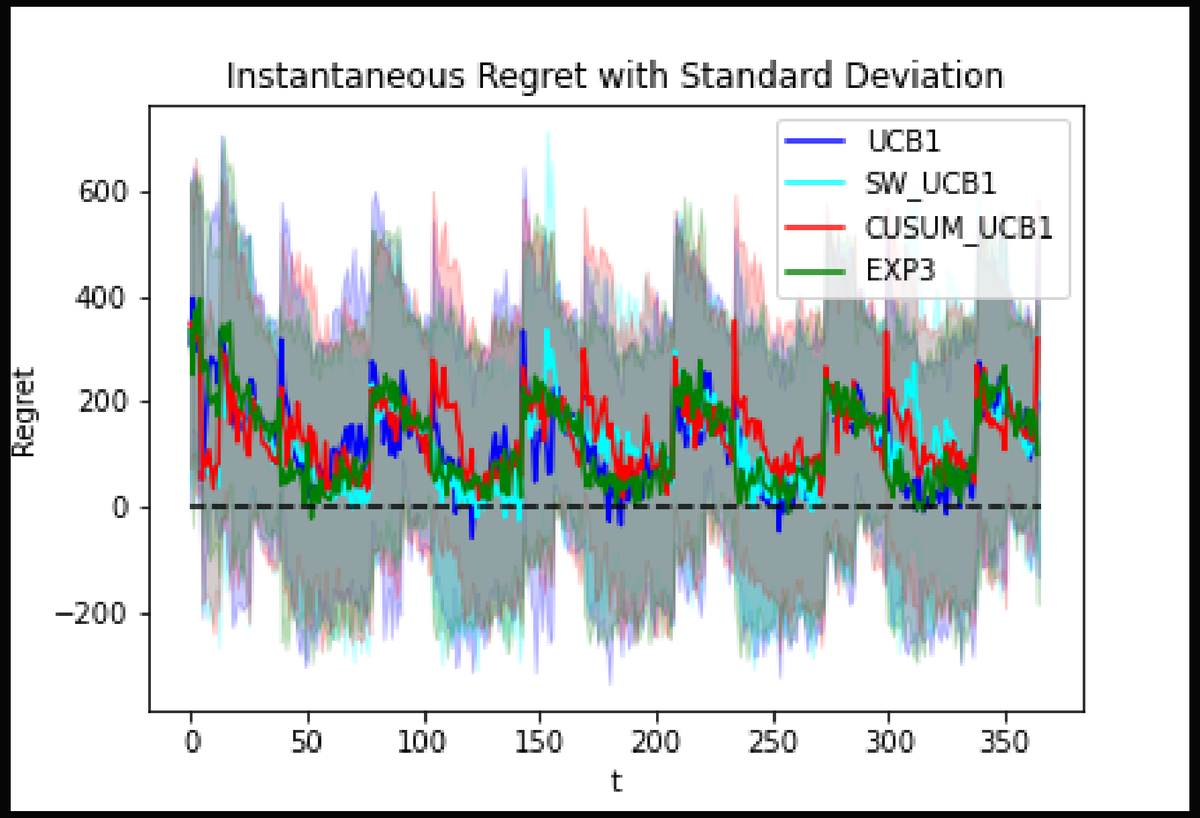
In case of highly frequent changes, the performance of EXP3 instead seems similar to the other algorithms, being on average slightly better than the others.

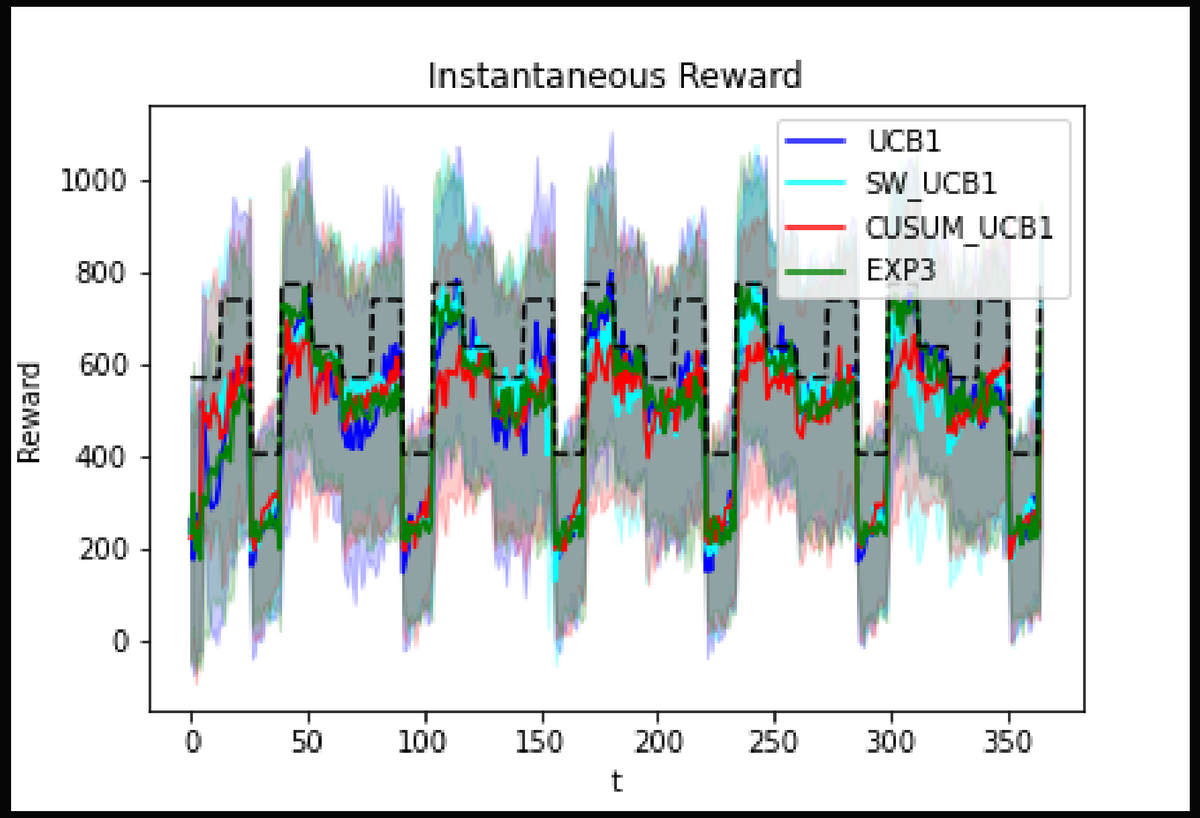




However, as the plots show, there is an issue connected to the variability of EXP3, which is extremely high and so makes EXP3 a non-reliable algorithm.

EXP3: 25 PHASES





As the plots can highlight, algorithms are all influenced by the great amount of changes introduced in the distributions. Any time a change occurs, the algorithms tend to all change their trends to adjust to the new distribution, but in this case none of them, including EXP3, is fast enough to stably reach the optimal configuration,