DATA INTELLIGENCE APPLICATION
PRICING & ADVERTISING

PROJECT REPORT

POLITECNICO DI MILANO

Diego Riva – 10467099

Matteo Rubiu – 10505109

Paolo Saccani – 10496490

PART 1

1.1 Environment Introduction

We chose to model a scenario in which an ISP provider must sell a tariff plan to new clients. The company wants to exploit advertising tools to maximize its income by investing every day a given budget in advertising campaign, in order to increase the audience of eligible users that will eventually buy the product (i.e. our tariff plan). Moreover, we want also to exploit the information of the buyers to optimize our product selling price, in some way. For advertising and pricing the company wants to try different approaches, both as independent problems and as complementary ones.

We identified two binary features that seemed relevant to plan a better advertising campaign; thus, we exploited this information to categorize 3 possible classes of visitors. In general, these features come from aggregate data, like cookies and social media analytics. From the combination of these features, we build this class matrix:

FEATURES	*Student*	*Not Student*
Consumer	Young	Adult
Not Consumer	---	Business

The 3 classes are:

- YOUNG: combination of *Student* and *Consumer*
- ADULT: combination of *Consumer* and *Not student*
- BUSINESS: combination of *Not student* and *Not consumer*

1.2 Conversion Rate Curves

We defined a demand curve for each class of user: these curves describe the assumed ad's click rate at each selling price for our product, in a real application these environments must be modelled tracking real users' behaviour.
In our curve more specifically we have:

- On the x axis, the possible prices for our product
- On the y axis, the click rate $r_c \in [0, 1]$, that is, the probability with which a user belonging to the class clicks on the ad, which shows the x price

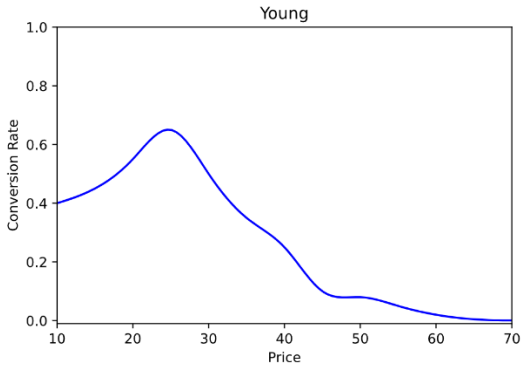


Figure 1.1: class Young Demand Curve

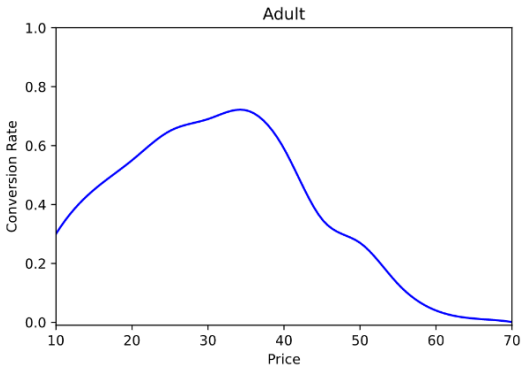


Figure 1.2: class Adult Demand Curve

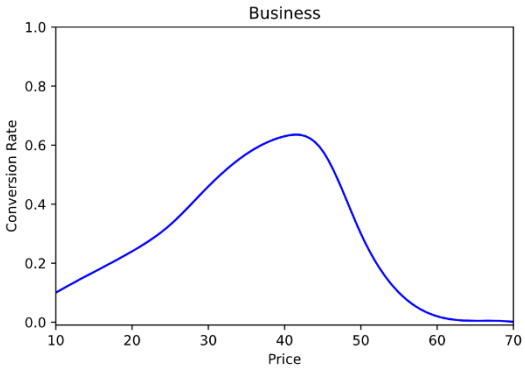


Figure 1.3: class Business Demand Curve

Modelling these curves, we tried to represent a “close to real-life” scenario:

- In figure 1.1, the younger clients are the less loyal and more value-focused customers, so the market is very aggressive with peak’s offers around 25$
- In figure 1.2, the Adult class comprehend a large slice of the population, indeed including a lot of family members, thus they are willing to pay more till around 35$
- Last in figure 1.3, the Business class refers to professionals that most of the time need advanced personal customer services, thus the peak’s price around 45$

Then, when necessary for our assignments, we aggregated these curves depending on the assumed percentage of population belonging to each class *c*, that we fix before a run:

- This occurrence probability $p_c \in [0, 1]$ represent the number of users of a class over the total amount of users.
- It must hold the property $\Sigma_C\, p_c = 1$

(NB: notice that we also refer to these percentages as “*weights”,* later in the projects)

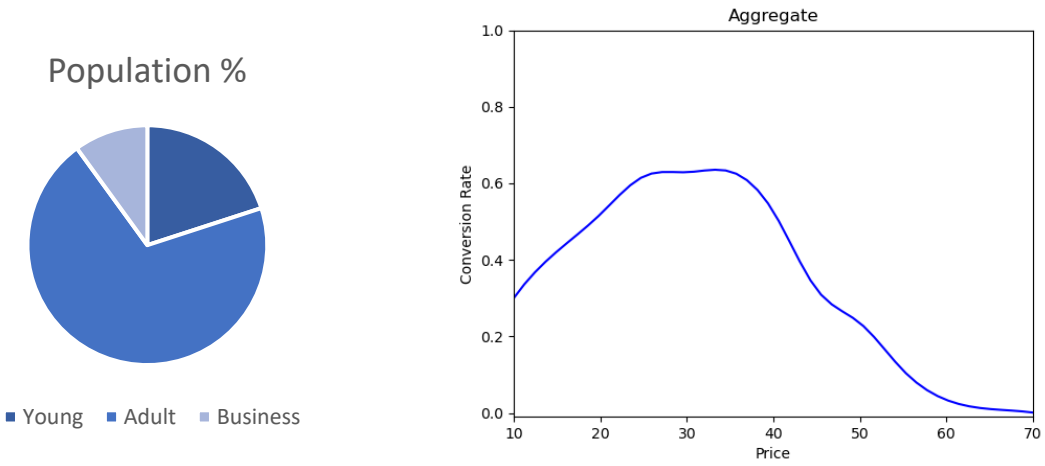


Figure 1.4: Aggregate Demand Curve and weights

In figure 1.4, we report an example of aggregated curve in which the occurrence probabilities are 20% for Young class, 70% for Adult and 10% for Business.

1.3 Sub-campaigns and Phases

We defined the experimental sub-campaign curves, starting from the assumption that each sub-campaign is associated to a class of users. These models plot the clicks an ad receive for each value of daily budget; the parameters of this plots are the slope and the saturation click value, after which the curve assume a constant behaviour.

We also defined 3 variants for each model representing 3 abrupt phases, that is, a shift in the curves caused by deep and quick changes in the market in a determined temporal slot. This adaptation of the model for the different phases avoids the loss caused by a simple average on data from all the time slots.

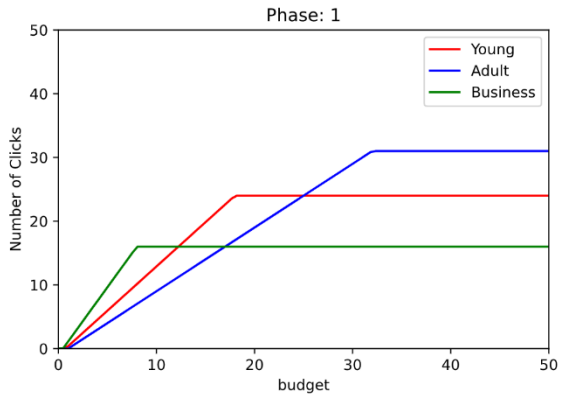


Figure 1.5: clicks/daily budget for each class, phase 1

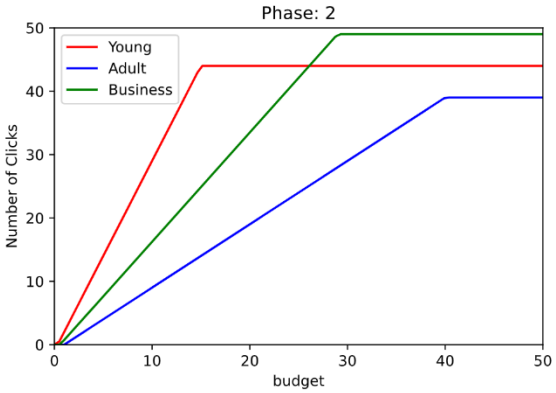


Figure 1.6: clicks/daily budget for each class, phase 2

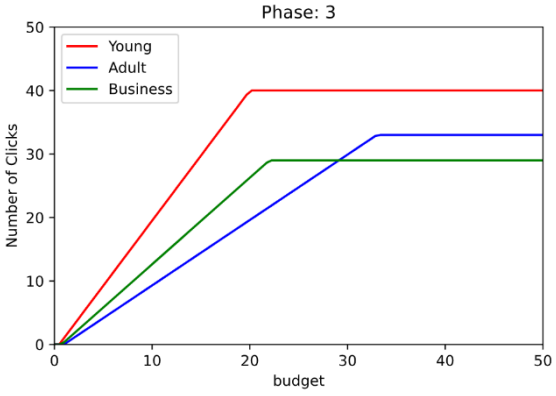


Figure 1.7: clicks/daily budget for each class, phase 3

In the first phase (Figure 1.5) we modelled our assumed curves model for the average market, with advertisement more effective on Young and Adult.

During the second phase (Figure 1.6), we tried to represent a possible impact of the pandemic on an ISP, with the increased general need of stable internet connection, especially for the Business professionals.

Lastly, in the third phase (Figure 1.7), a competitor enters the space subtracting market share.

PART 2

2.1 Model Description

Our aim, in this assignment, is to design a combinatorial bandit algorithm to optimize the budget allocation over the 3 sub campaigns to maximize the total number of clicks under the constraint that we have a maximum cumulative daily budget.

In this part we will run the experiments only on the first phase, without considering any abrupt changes that modifies the environment's behaviour.

First, we make an estimate of the true curves representing the normalized number of clicks over the budget by means of a *Gaussian Process Regressor*. This approach is able to produce an evaluation that fit the shape of our ideal curves correctly and with a negligible noise.

Then we use the results of the previous step to build the *Budget's Environments* for each sub campaign (corresponding to a specific user class).

We define a *Gaussian Process Thompson Sampling Learner* that, in each time instant and for each arm, draws a sample from the corresponding Gaussian Process and updates the one corresponding to the arm that generated the best sample. In this way the Learner can produce an estimate of the expected number of clicks associated with each budget's arm that will be used as input of our knapsack optimization problem: we obtain then the best allocation of budget over the three sub-campaigns and the corresponding expected value.

We repeat this process in different experiments, and at the end we average all the results, obtaining a final plot for the reward during time.

As the last step we use the reward data to compare them with our optimal value, in order to also plot the cumulative regret during time.

2.2 Model Set-Up and Analysis of the Results

For this part, we set our parameters in the following way:

- Time interval T: 122 days (corresponding to our first phase)
- Min budget allocation to sub-campaign: 0.0
- Max budget allocation to sub-campaign: 50.0
- Cumulative daily budget constraint: 50.0
- Number of experiments: 10

Now we provide some examples of *Gaussian Process Regressor* estimates of the true starting curves of the budget w.r.t. the number of clicks, for each sub campaign:

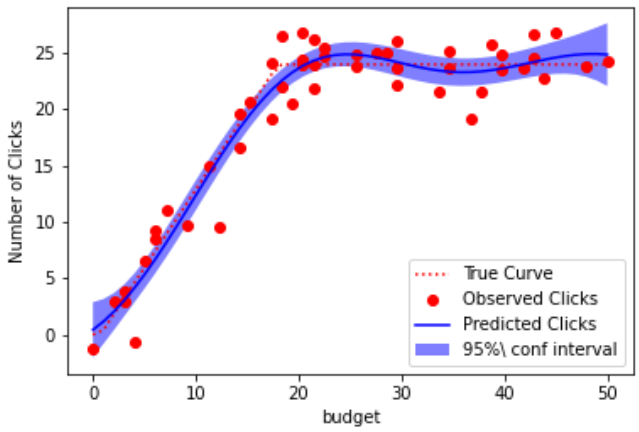


Figure 2.1 - YOUNG

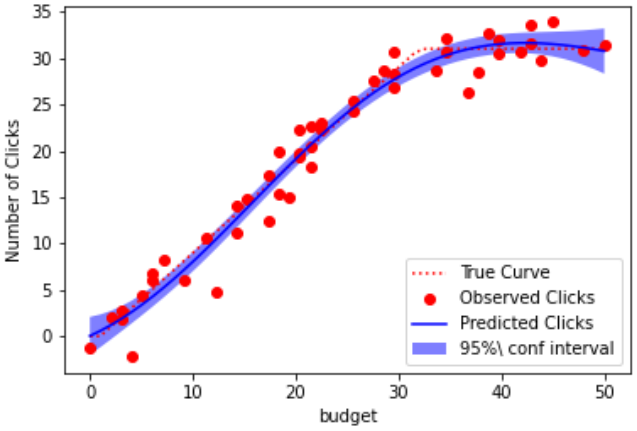


Figure 1.2 - ADULT

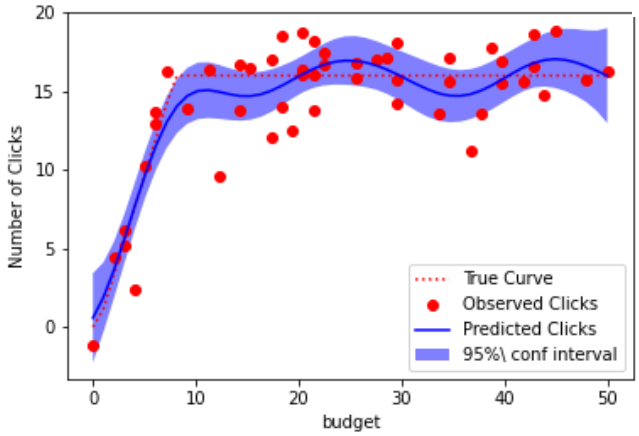


Figure 2.3 - BUSINESS

We compare two different setups in the number of arms:

1. Number of arms: 11

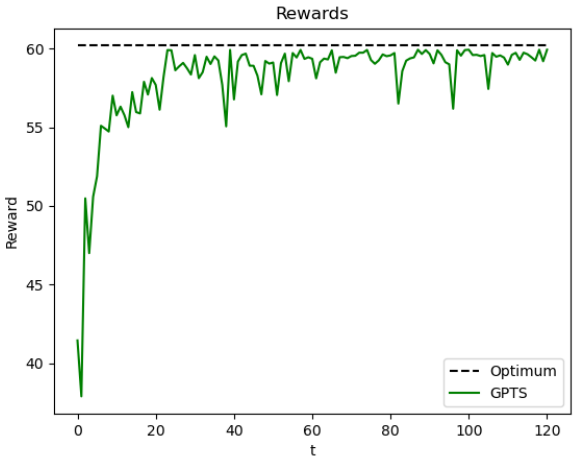


Figure 2.4

Figure 2.5

2. Number of arms: 22

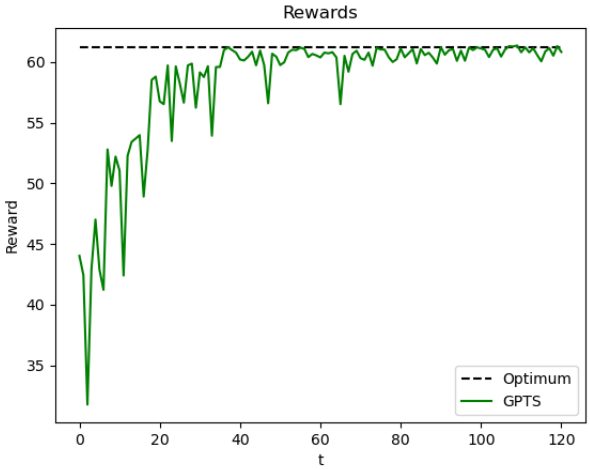


Figure 2.6

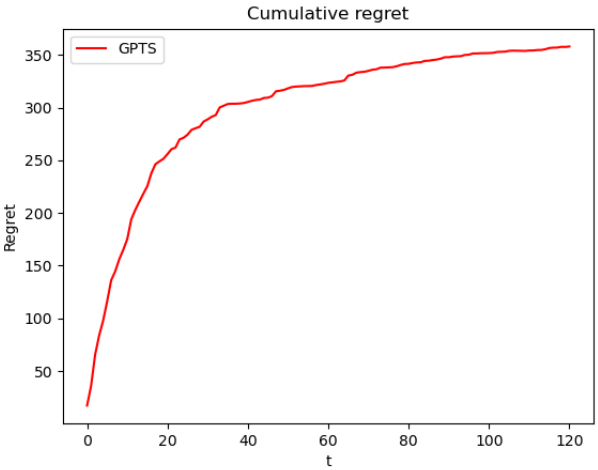


Figure 2.7

In the plots are represented the rewards (Fig. 2.4, 2.6) and the cumulative regret (Fig. 2.5, 2.7) of the two corresponding configurations of arms.

As we can see, the two results are coherent with the theory: in both cases we can observe that there is a first phase of exploration, in which the algorithm tunes the probabilities during time, followed by the exploitation one, in which it converges to the optimal value.

As we expected, the time needed to the case with 22 arms to reach the optimum is larger than the case with 11 arms, but it brings to optimal values that are a bit higher.

In both experiments the algorithm converges, and we can find the best allocation of our budget in order to maximize the total number of clicks given by each sub campaign under our budget constraint.

PART 3

3.1 Model Description

In this paragraph the focus moves on the non-stationary environment case in which we consider the three abrupt phases. We want to compare the performance of the classical approach seen in Part 2 to a variant, that is, a sliding-window behaviour on the collected observation during time.
The implemented sliding-window combinatorial bandit algorithm should be less impacted by the change in the environment, due to its capability of "forget" samples older than a specific *tau* (time instants of the window).
To face the problem, it is convenient to change the logics of the Learner described in the previous assignment and modelling each arm by means of a Beta distribution instead of a Gaussian Process.
This change also allows the algorithm to perform faster.
For each instant of time the knapsack table is populated with the value sampled by the beta distribution of each arm to find the best allocation and the best value for that iteration. Then we update the beta distribution of the arm chosen by the knapsack optimization for each sub-campaign.
When a phase finishes, all the environments swap to the next phase, and the algorithm should adapt to the modifications, converging to the new optimal value.

3.2 Model Set-Up and Analysis of the Results

Parameters settings and assumptions:

- Time interval T: 366 days (122 days for each phase)
- Min budget allocation to sub-campaign: 0.0
- Max budget allocation to sub-campaign: 50.0
- Cumulative daily budget constraint per phase: [55.0, 60.0, 50.0]
- Number of experiments: 50
- Number of arms: 6

Comparing two different sliding window size:

1. Sliding window size: sqtr(T) = 19

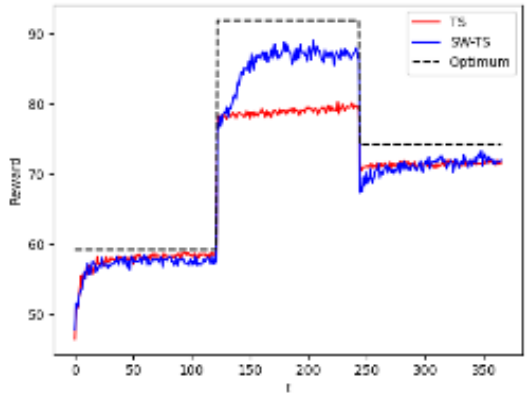


Figure 3.1

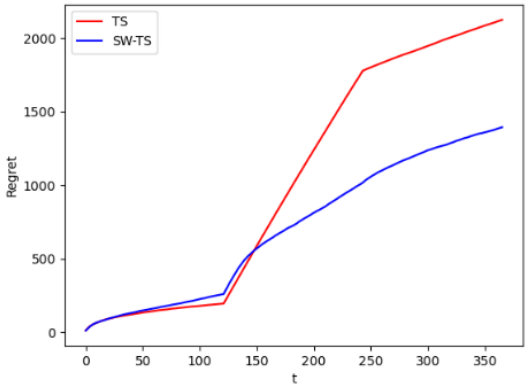


Figure 3.2

2. Sliding window size: sqtr(T)*2.5 = 47

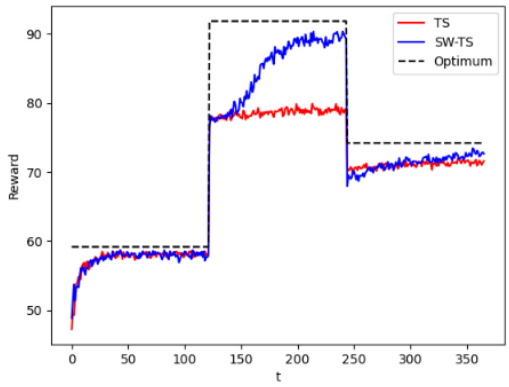


Figure 3.3

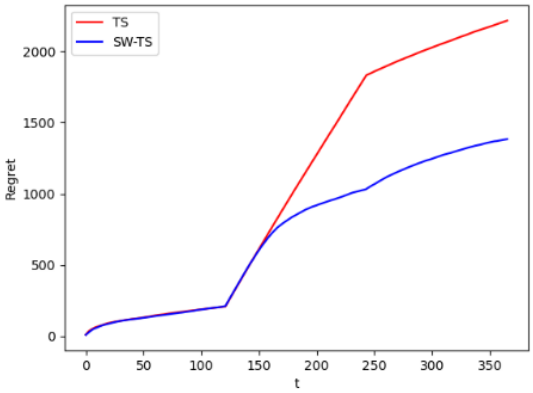


Figure 3.4

It is easy to see that the algorithm shows results that are not completely satisfying, but it is still possible to make some considerations: in both cases we obtained the evidence that the sliding window allow the adaptation to the new environment while the case without this implementation seems not able to explore properly the new settings. This is visible also in the regret plot: the regret of the TS Learner is lower or equal to the SW-TS Learner before the first abrupt change, but it is exceeded in performance as the environments change.

The only remarkable comparison between the two tests is that a longer sliding window leads to better rewards but the number of samples needed to adapt are larger than the case with a smaller sliding window size where the algorithm seems to converge faster.

PART 4

4.1 Model Description

Our aim in this assignment is to design a learning algorithm for pricing when the users that will buy the product are those that have clicked on the ads, assuming the budget over the three sub campaigns as fixed.

From now on we will also focus only on the first phase without considering any abrupt changes that modifies the environment's behaviour.

For the pricing part we used a different approach for building the conversion's rate curves: we performed an interpolation on a set of points defined as input of our algorithm.

Each curve is associated with a different probability, we call *weight.*

Then we build an aggregate *Price Environment,* containing a weighted sum over the three conversion's rate curves.

We define a *Thompson Sampling Learner* that, in each time instant and for each arm, draws a sample according to the corresponding *Beta distribution* and updates the one of the arm that generated the best sample, collecting the corresponding reward.

We repeat this process in different experiments, and at the end we average all the results, obtaining a final plot for the reward during time.

As the last step we use the reward data to compare them with our optimal value, in order to also plot the cumulative regret during time.

4.2 Model Set-Up and Analysis of the Results

As regards this assignment we decided to present different examples of results, with different choices of Environment's weights and different numbers of arms. We set our main parameters in the following way:

- Time interval T: 365 days
- Number of experiments: 30

Now we present the results with different *weights*:

- Weights: [Young = 20%, Adult = 40%, Business = 40%]

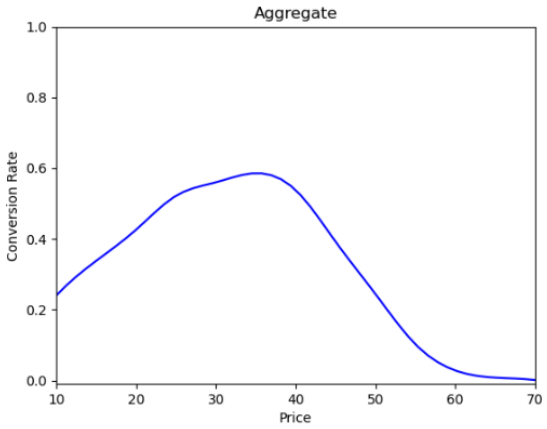


Figure 4.1

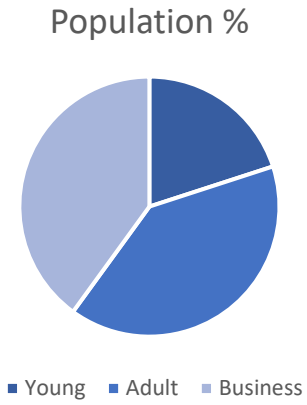


Figure 4.2

- Number of arms = 4

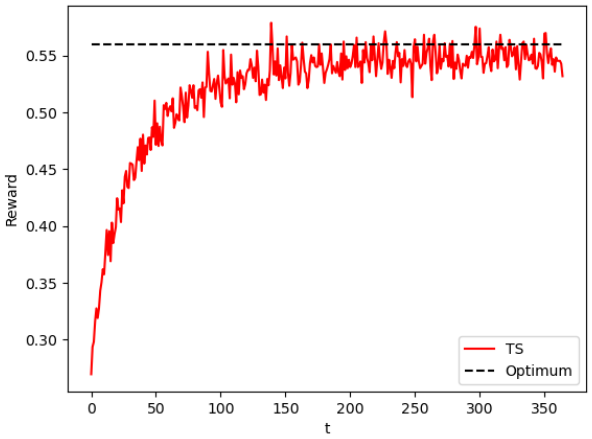


Figure 4.3

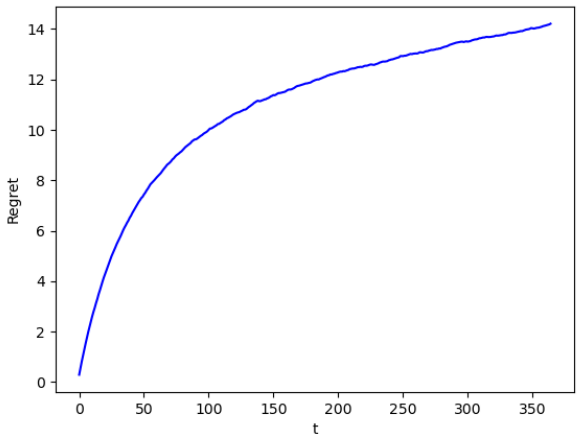


Figure 4.4

- Number of arms = 6

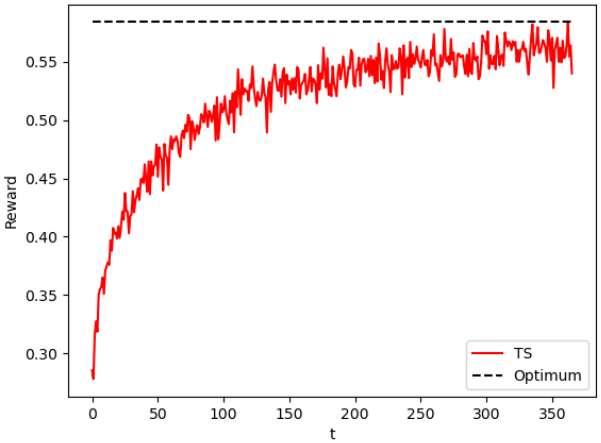


Figure 4.5

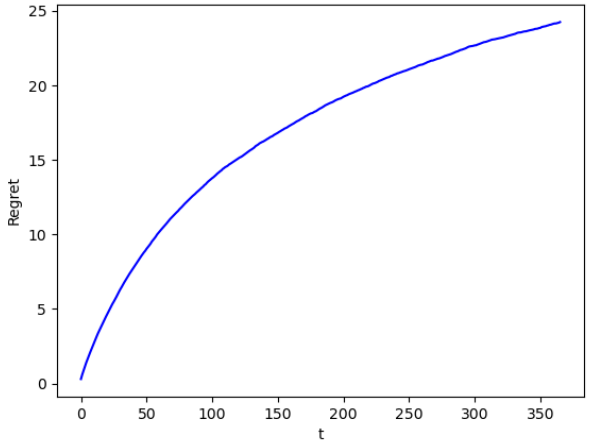


Figure 4.6

- Weights: [Young = 20%, Adult = 70%, Business = 10%]

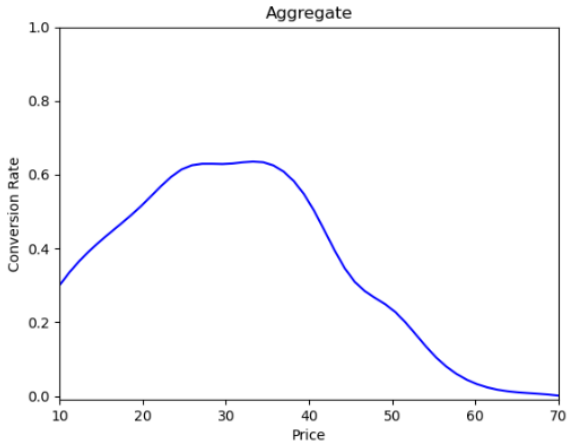


Figure 4.7

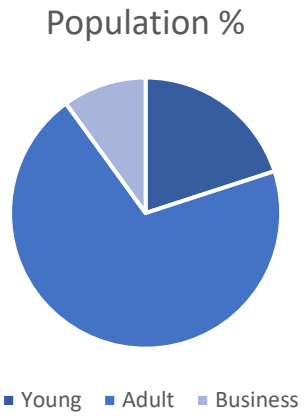


Figure 4.8

- Number of arms = 4

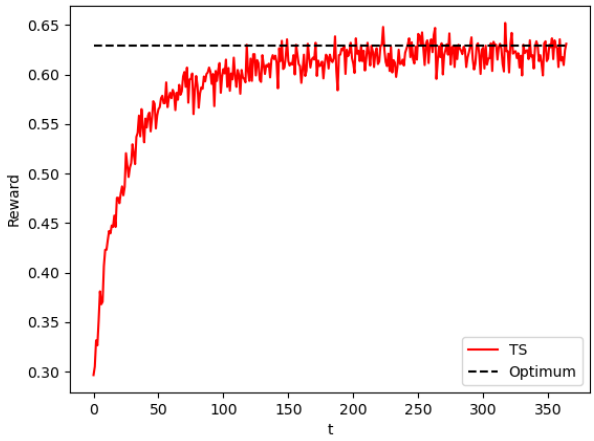


Figure 4.9

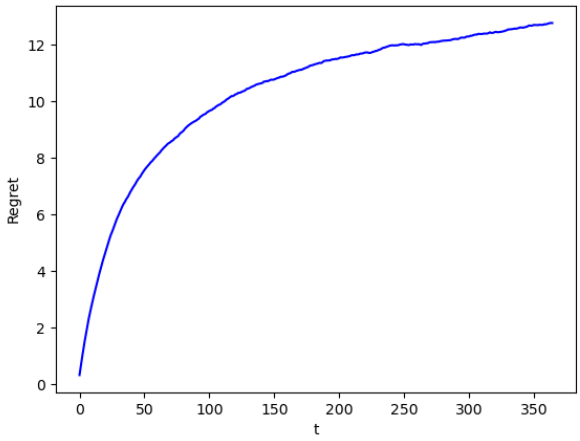


Figure 4.10

- Number of arms = 6

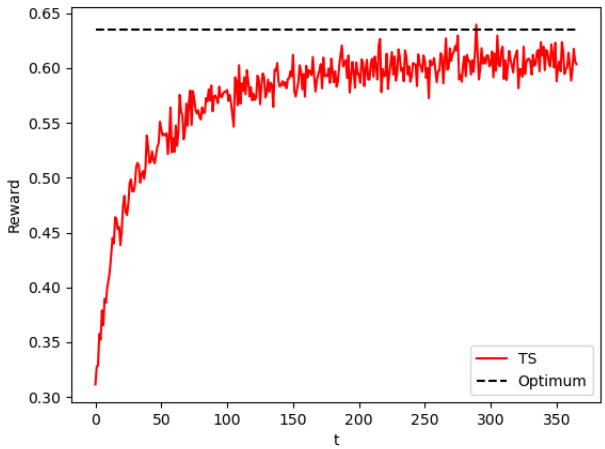


Figure 4.11

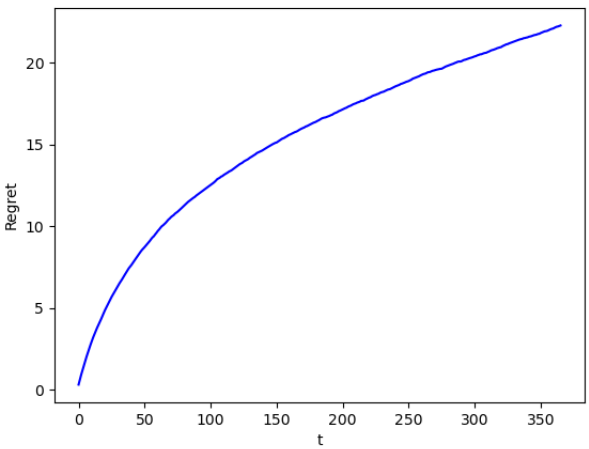


Figure 4.12

The above figures represent the rewards and the cumulative regrets of the two corresponding configurations of arms, for each choice of weights.

Figures 4.1 and 4.7 show that a change in the weights lead to a different aggregate conversion rate curve. For each configuration we can see how the optimum value is influenced by both the weight configuration and the number of arms: the analysis shows that a loss in weight on the Business class in favour of the Adult one makes the algorithm to converge to higher rewards and to a lower regret.

The number of arms also affects the speed to which the algorithm converges to the optimum and then the regret curve: lower is the number of arms, and faster will be the convergence of the reward and the regret curves.
This fact is evident comparing configurations with 4 and 6 arms, respectively.

PART 5

5.1 Model Description

For this assignment, we implemented a context generator algorithm, starting from the general implementation of the Thompson Sampling of part 4.

Our aim is to maximize the total expected reward in a time horizon, making a decision on the context at the end of every week: the goal is to select the best and most reliable configuration of curves from which to choose the arms, deciding for each following week the "predicted to be best" combination of classes, also referred as a context.

To obtain this, we initialized all the environments generated from all the possible combination of aggregated curves, that depend on the class's aggregation.

We start our experiments with the completely aggregated curve and at the end of each time slot:

- calculate the lower bound of the reward for the best arm of each context
- compare the result of the previous step and select the maximum, therefore the corresponding context as the most promising one for the next week
- choose this context for the subsequent 7 days, and at the end of the week we repeat the procedure until the conclusion of the time horizon

At the end we plot the cumulative reward that is the result of all context's swap performed at the end of every week. This should lead to a faster convergence of the regret to the optimum value.

The process result is averaged on the number of experiments to better approximate the obtained results.

5.2 Model Set-Up and Analysis of the Results

Parameters setting and assumptions:

- Time interval T: 365 days (52 weeks)
- Number of experiments: 2500
- Number of arms: 6
- Weights: [young: 20%, adult: 30%, business: 50%]
- Lower bound confidence: 0.9

The plots capture the behaviour of each single context. The algorithm compares them every week to choose which is the most promising context for the next week.

In the figures the classes Young, Adult and Business are annotated respectively as 1, 2 and 3.

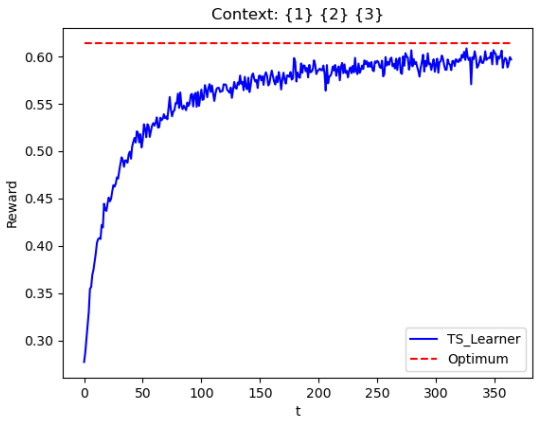


Figure 5.1

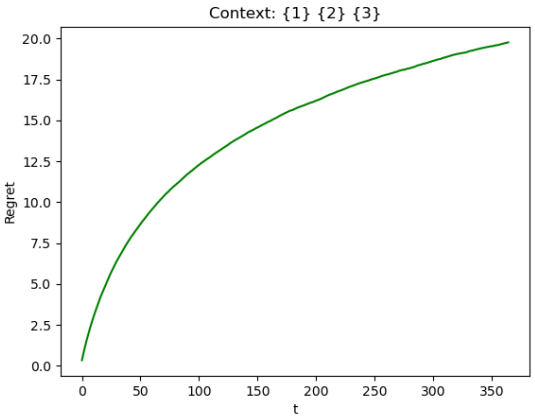


Figure 5.2

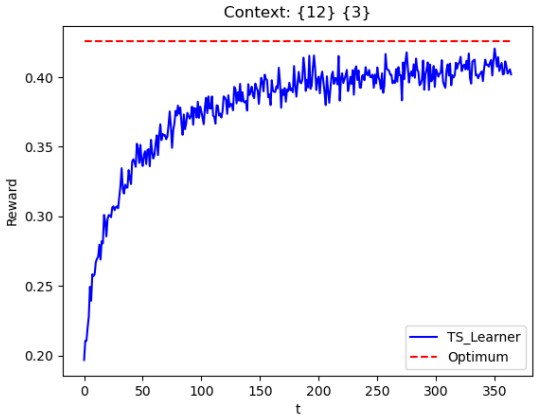


Figure 5.3

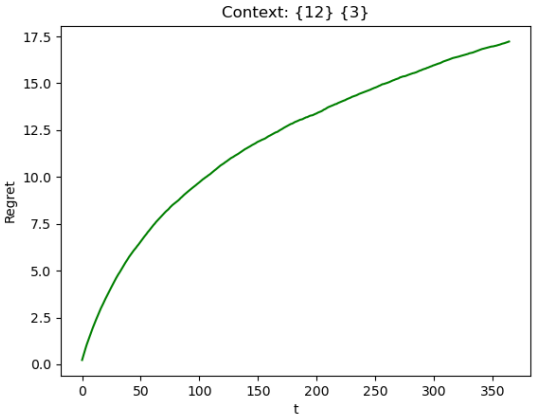


Figure 5.4

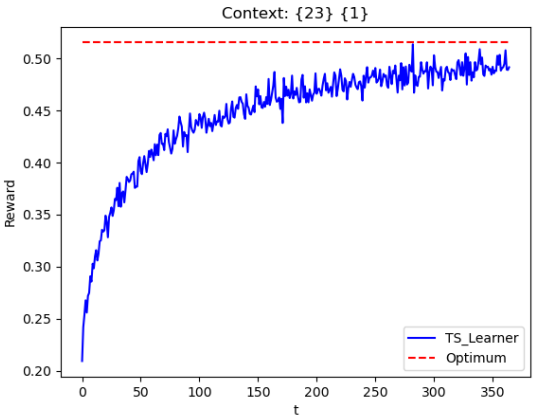


Figure 5.5

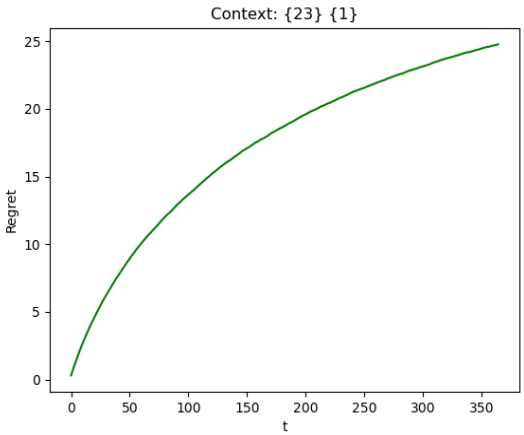


Figure 5.6

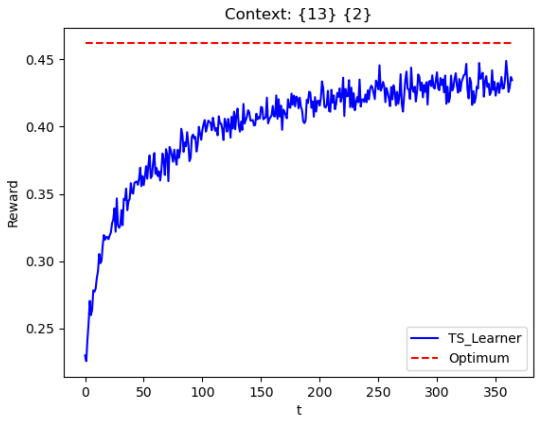


Figure 5.7

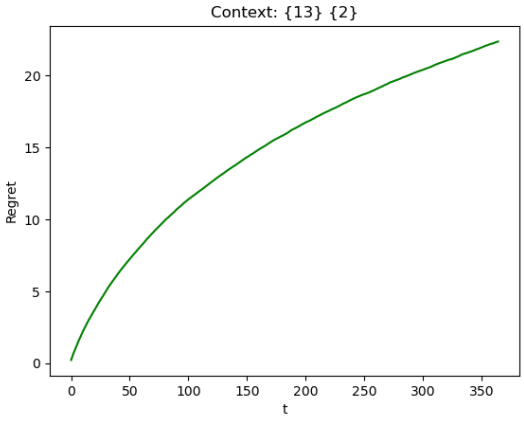


Figure 5.8

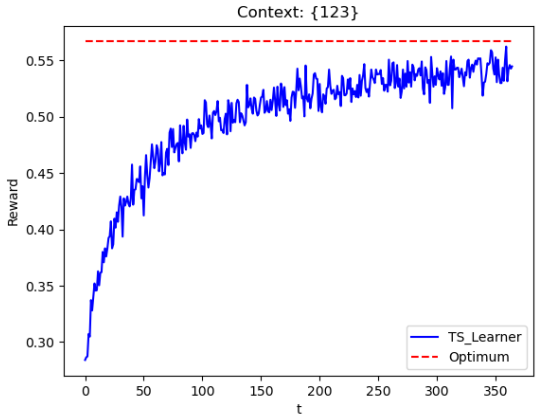


Figure 5.9

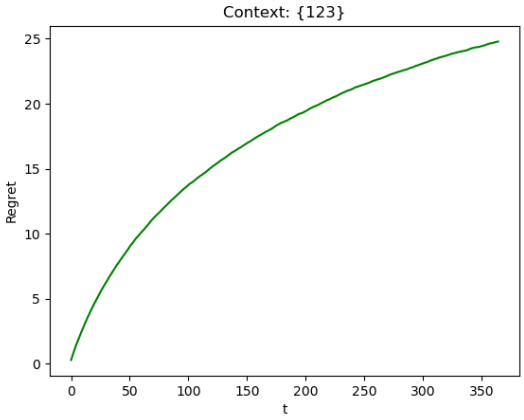


Figure 5.10

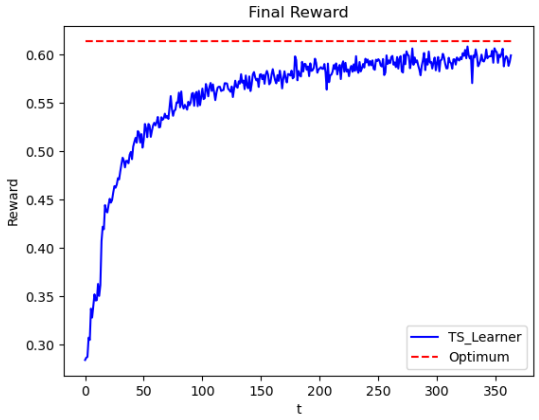


Figure 5.11

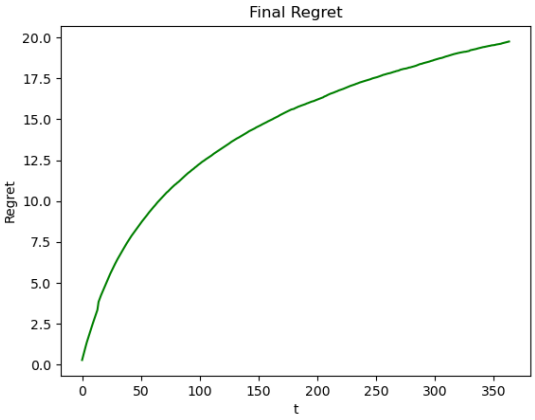


Figure 5.12

Even if the convergence speed seems similar in the various contexts, they lead to different expected rewards, based on the adopted weights and the optimal reachable value.

In the early phases we can observe how the lower reward bound of each context is in continuous mutation, in such a way that the algorithm adapt itself very frequently while moving forward in time. The disaggregate case in the end will be always the most promising, reaching higher optimal values, but at the start has a slower growth.

In our analysis, this predominance of the disaggregate case is visible after some weeks from the start.

In these figures below, we report the lower bounds of each contexts, from some “sampled” experiment, to show their behaviour. Obviously after thousands of iterations these small variations will converge into an initial preference of more aggregated contexts followed by the steady choice of the disaggregate one.

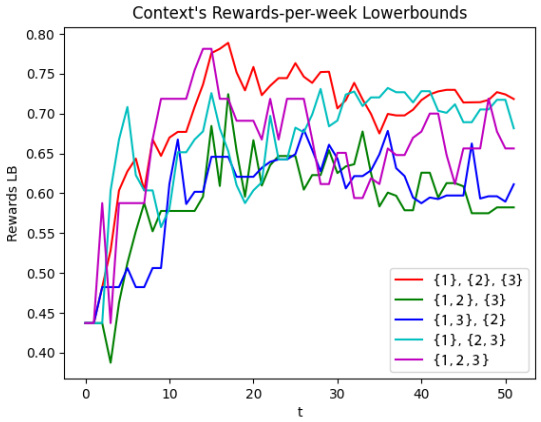


Figure 5.13

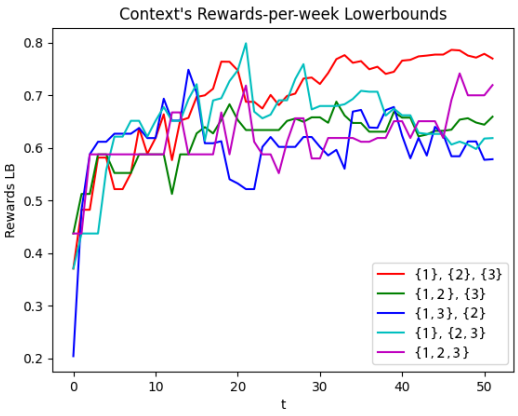


Figure 5.14

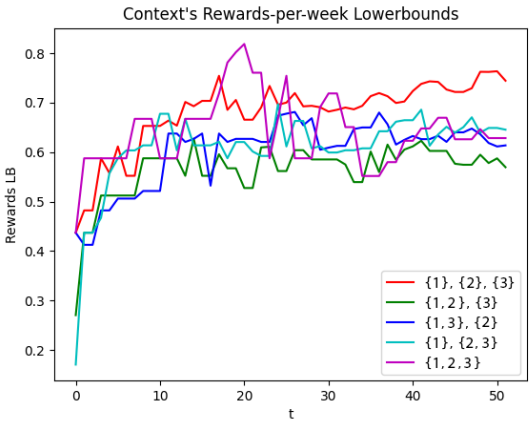


Figure 5.15

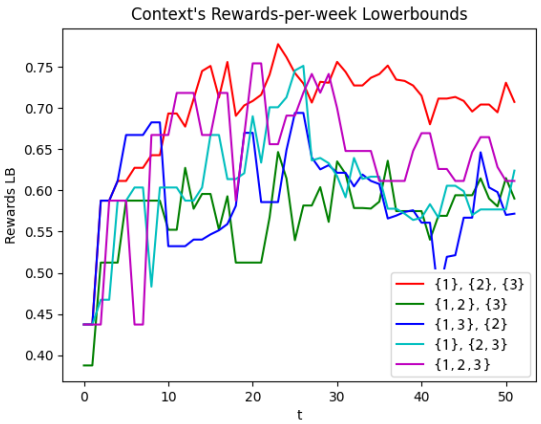


Figure 5.16

PART 6-7

6.1 Model Description

Previously the pricing and advertising problems have been considered singularly. Now we merge them, fitting the result of the pricing TS model multiplied by the number of users that will click on the ads inside the Knapsack Optimization Algorithm. This will return the results of the corresponding advertising problem.

In this way we build a model that allow us to allocate optimally our budget, basing our choices on the pricing data collected from the users.

We implemented a TS algorithm and a GP-TS to approximate respectively the conversion rate curves and the number of clicks per budget curves, under the assumption that each sub campaign corresponds to a specific class of user.

We analyse the disaggregate case in which each class has its own conversion rate (Part 6) and the aggregate one in which we look for a unique fixed price for all the users (Part 7).

We will highlight some differences between the disaggregate and aggregate cases, in order to understand what the best strategy is to adopt in our case.

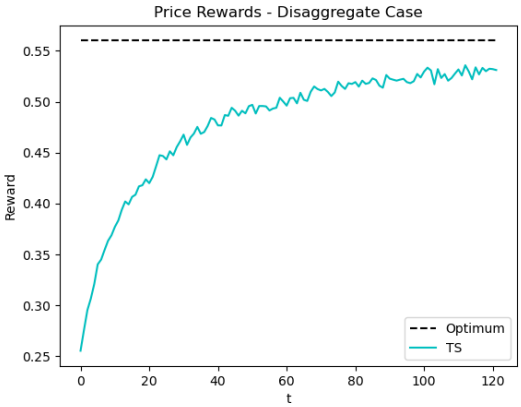
6.2 Model Set-Up and Analysis of the Results

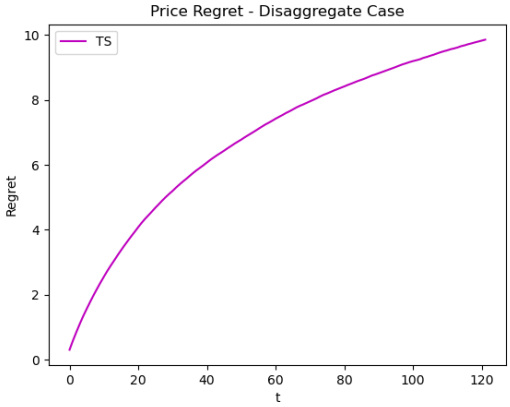
Parameters setting and assumptions:

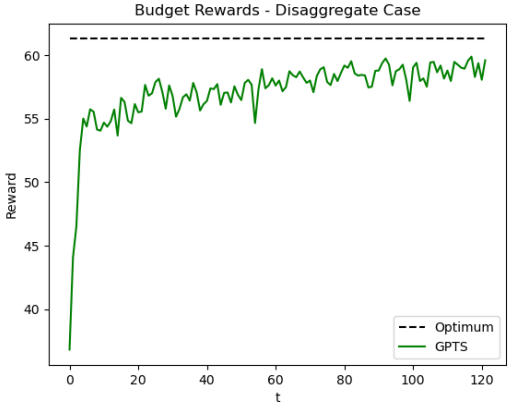
- Time interval T: 122 days
- Min budget allocation to sub-campaign: 0.0
- Max budget allocation to sub-campaign: 50.0
- Cumulative daily budget constraint: 50.0
- Number of experiments: 50
- Number of arms for pricing: 4
- Number of arms for advertising: 7
- Weights: [young: 20%, adult: 40%, business: 40%]

The following figures represent the two casuistic that we are going to compare:

Disaggregate case:







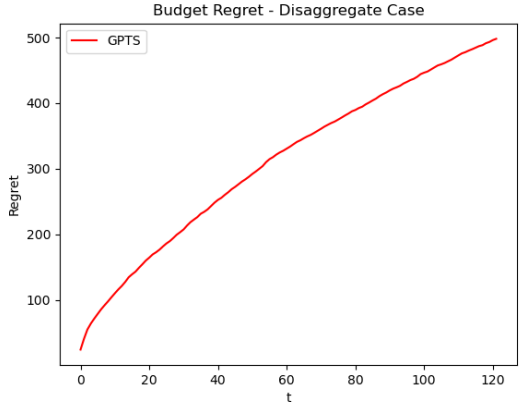
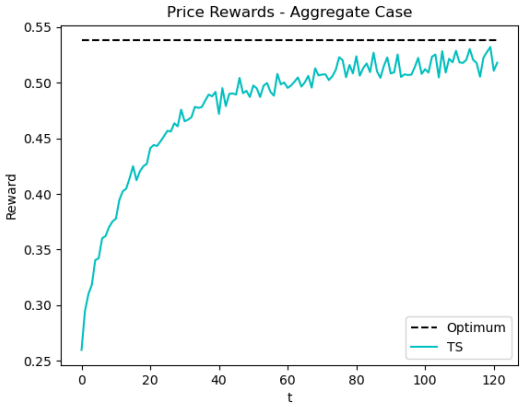
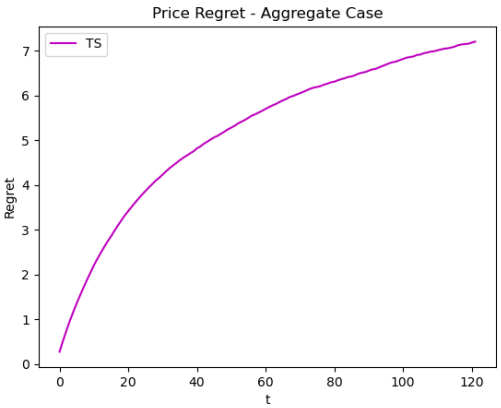
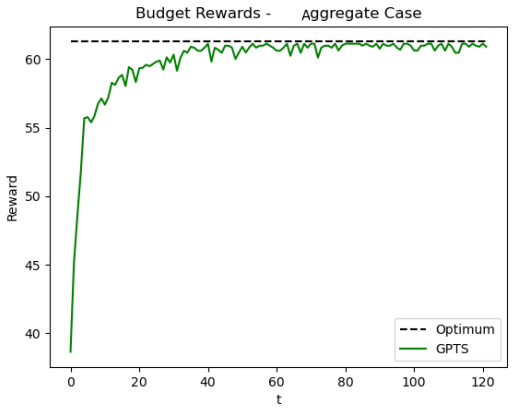


Figure 6.1

Aggregate case:







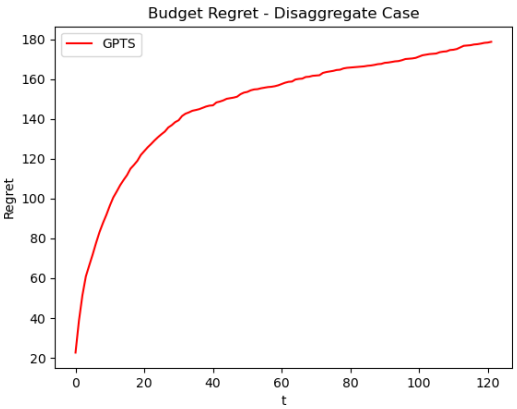


Figure 6.2

Running the experiments, we decided to adopt a time horizon corresponding to the length of the phase of the advertising problem, choosing the values of the first. The choice has been forced by the fact that at each time instant we solve the pricing problem and we fit the results in the knapsack algorithm, in which the arms are modelled through a GP-TS. The computational time needed to solve the entire algorithm is particularly relevant and a study with a more extended time interval would need a huge amount of time.

The analysis points out how the aggregate case and the disaggregate one behave differently:

- As we expected, the optimum of the price reward is slightly higher in the disaggregate experiment since we better exploit each single conversion rate curve.
- The pricing regret curve shows clearly how the convergence is faster in the aggregate case.
- From the observation made above it is possible also to understand how the two approaches influence the advertising problem: a slow convergence of the price learner leads to a slower convergence also of the budget learner, increasing the regret with respect to the optimum.

Completing the analysis under the described environment setting, it is possible to assert that looking for an optimum for each class of users has the potential to reach higher rewards, but it is necessary to collect way more samples w.r.t. choosing to adopt a single fixed price for everyone.

To conclude, then, there is not a strong evidence that the company should behave differently on the class of users.