ONLINE LEARNING APPLICATIONS

PRICING & SOCIAL INFLUENCE PROJECT

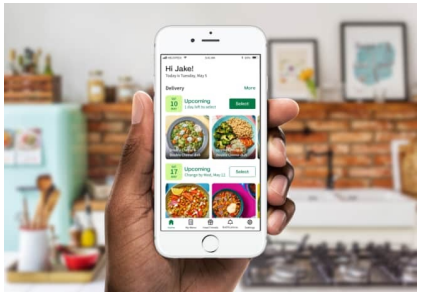
ACADEMIC YEAR 2021/2022

Gattinoni Valentina
Manzoni Claudio
Marchi Nicolò
Masatti Gabriele
Moneta Alessandro Marco Cesare

THE IDEA

1. Pick a plan

Whether cooking for yourself or someone else, you can choose a **weekly box**. Several plans are available to match **every kind of lifestyle.**



2. Compose your weekly box

In your **weekly box** you can select, every time: the **number of recipes**, and their **type** according to specific needs

3. Get your delivery

Each week, open simple **step-by-step recipes** complete with nutritional information and **fresh, pre-measured ingredients** to get easy to prepare delicious meals in no time.

00

PRODUCTS & USERS

OUR USERS

Our users are divided on the basis of three features of the recipes:

- Price
- Difficulty
- Preparation time

	YOUNG	OLD
HOME COOKS		
EXPERIENCED CHEF		

YOUNG HOME COOK

ELDER HOME COOK

EXPERIENCED COOK

OUR PRODUCTS

Our delicious weekly boxes can be composed with **five different products**, which are thought to be suitable to all kinds of users.
Indeed they are characterized by **3 parameters** that allow them to match with all types of customers, these are:

- **Price**: the price at which the product is offered to the client;
- **Difficulty**: it denotes the experience and the means required to follow the recipe;
- **Preparation time**: time required to complete the recipe;

VERMICELLI WITH BROCCOLI AND BACON

Ingredients (for two) :

- Bacon (100g)
- Vermicelli (a packet)
- Sage (1 piece)
- Grated hard cheese (a packet)
- Garlic (1 piece)
- Broccoli (500g)

The key to making this One-Pot Bacon Broccoli Pasta successfully is to cook the bacon, drain it, then cook the rest of the dish in the same pan. So to get a little of that bacon flavour without soggy bacon. You can use any small shaped pasta such as penne or farfalle pasta. You can add rotisserie chicken or cook cubed chicken after you've cooked the bacon.

PINSA WITH ANCHOVIES AND RICOTTA

Ingredients (for two) :

- Pinsa (a packet)
- Tomato (1 piece)
- Basil (5g)
- Onion (1 piece)
- Anchovies (46g)
- Ricotta cheese (a packet)

With the back of a metal spoon, spread an even layer of ricotta across the base of the pinsa. Add ingredients until the ricotta is fully covered, as this will protect the cheese from burning. Next, lay over an anchovy where each slice will be. If you desire, top with a generous handful of fior di latte cheese and finish with a drizzle of olive oil.

CHICKEN CURRY WITH GREEK YOGURT

If you're vegetarian, or not a chicken fan, you can use chickpeas instead! Tips on cooking Indian inspired cuisine with Greek yogurt, suggest that you should whisk together a little bit of your water with the yogurt, and make sure your heat is on low when you add it.

Ingredients (for two) :

- Chicken tender (250g)
- Chard (200g)
- Curry (a packet)
- Yellow curry paste (a packet)
- Greek yogurt (a packet)
- Coriander (5 g)
- Onion (1 piece)
- Jasmine Rice (a packet)
- Garlic (1 piece)
- Granular Vegetable Broth (2 packets)
- Chili Pepper (1 piece)

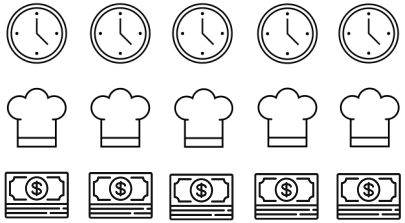
TENDER LOIN OF PORK WITH HONEY

The best method for truly juicy pork tenderloin is to first pan-sear it and then finish it in the oven. A lean cut like this can become very dry if overcooked, so the move is to go hot and fast. Grab an instant-read thermometer—it's a great tool to take the guesswork out of cooking perfectly done pork.

Ingredients (for two) :

- Pork loin (300g)
- Potatoes (8 pieces)
- Honey (4 packets)
- Butter (3 packets)
- Cooking cream (a packet)
- Thyme (5 g)
- Marjoram (5 g)
- Shallot (1 piece)
- Carrot (5 pieces)
- Apple (1 piece)
- Mustard (3 packets)
- Rosemary (1 piece)

BAKED SEA BREAM



Ingredients (for two) :

- Sea bream (200g)
- Tomato (2 pieces)
- Olives (a packet)
- Capers (a packet)
- Basil (5 g)
- Garlic (1 piece)
- Couscous (a packet)
- Zucchini (1 piece)

Kalamata olives work particularly well for this recipe as they pit easily. If you can't find them, try another olive that pits well (one good way to find out is to eat one) or purchase pitted ones. Remove the pits from the olives. Give the capers a good rinse, especially if they've been preserved in salt.

THE INTERFACE

1. Proposed products and primary selection

Once the user has landed on the webpage of the **preferred product** (technically speaking, the **primary**) he/she starts to compose the **weekly box,** as he/she is allowed to include up to 5 products and choose the number of items of each of them.

PRIMARY

SECONDARIES

2. Secondaries

After that, **two suggested secondaries** will be shown to him/her. The first secondary will be for sure seen by the user, while the second with a probability λ, that contributes to the likelihood that a secondary is clicked.

Clearly, if a product has been selected and labelled as primary it is not possible to select it again if displayed as secondary and cannot become primary again.

TABLE OF CONTENTS

01

ENVIRONMENT

01 ENVIRONMENT: Description

GOAL

The goal of our work is to find **the optimal prices of primary products** such that the **expected reward is maximised.**

RATIONALE

We have created an environment where **all the parameters of the distributions** that model the problem **are defined**.

RESULT: EASIER WORK

In the following steps it will be only necessary to **simulate separately the unknown parameters one at a time** to find the optimal ones.

01 ENVIRONMENT: Description

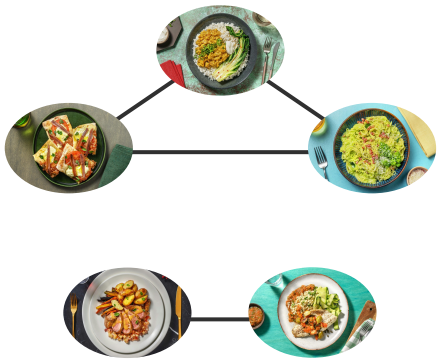
ASSUMPTIONS

- The expected values of the **α-*ratios*** are known;
- For every primary **the secondaries are fixed**;

TWO GRAPHS

Two kinds of graphs have been considered:

- **Fully connected graph** where all the products are linked;
- **Partially connected** graph where not all products are linked;

FULLY CONNECTED	PARTIALLY CONNECTED
	

01 ENVIRONMENT: Products

The products sold on the website are the recipes to compose the weekly box.
For each of them a **range of prices** has been defined considering real data of products of this kind sold by companies of the market segment. A **margin** is defined considering the difference between the **price offered to customers** and the **costs** for the company;

Product	Range of prices	Costs	Margins
Vermicelli	6,5 - 8 – 9,50 - 11	2,75	3,75 – 5,25 – 6,75 – 8,25
Pinsa	8,50 – 9 – 10,50 - 12	5,25	3,25 - 3,75 – 5,25 – 6,75
Chicken curry	12 – 13,50 – 15 – 17	7	5 - 6,50 – 8 – 10
Loin of pork	14 – 17 – 19,50 - 23	9,5	4,5 – 7,50 – 10 – 13,50
Baked Sea Bream	18 – 21 – 25 – 25 - 29	12,5	5,5 – 8,5 – 12,5 – 16,50

01 ENVIRONMENT: Users

Three classes of users are designed with two binary features that are **cooking experience** and **age**: young/old without experience and experienced cookers. The probability of a user to be old is 0.3 while the probability to be expert is 0.35 .

	YOUNG p = 0.7	**OLD** p = 0.3
HOME COOKS p = 0.65	 p = 0.455	 p = 0.195
EXPERIENCED CHEF p = 0.35	 p = 0.245	 p = 0.105

USER PROPERTIES

A user is identified by:

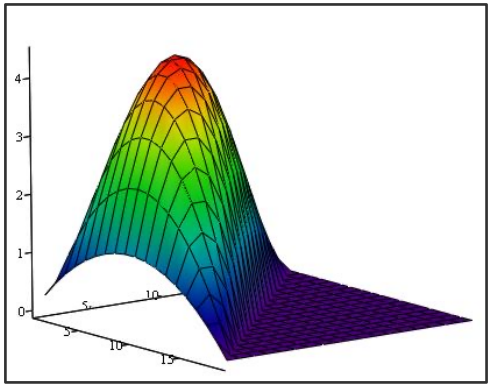
- The **cluster**;
- The **α-ratios;**
- The **reservation price;**
- **Poisson parameter;**
- **Probability matrix**

01 ENVIRONMENT: Users

In the following are summarised the **basic assumptions** on the probability distributions governing the relevant quantities considered in the development of the environment. **The parameters** of these distributions **will change accordingly to the user and the product considered**.

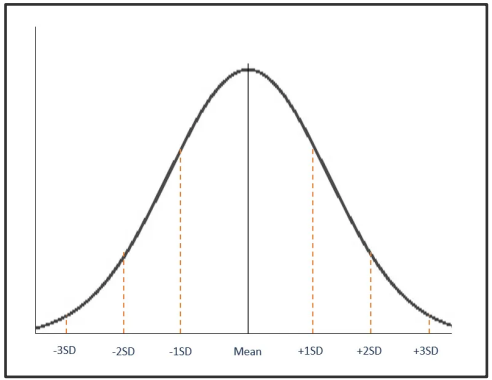
α-RATIOS

Sampled from dirichelet distributions ~Dir(a1, a2, ..., a5)



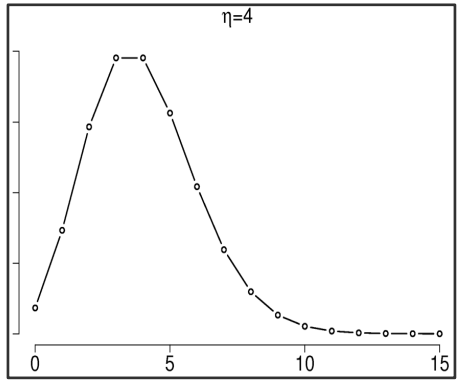
RESERVATION PRICES

Sampled from Gaussian distributions ~N(μ, σ)



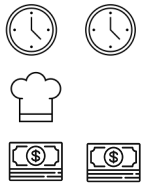
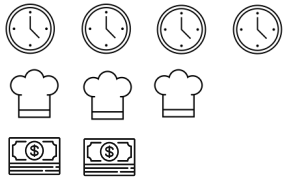
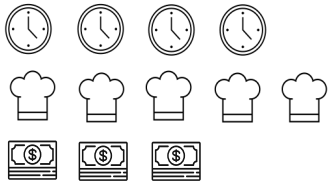
Nº OF ITEMS

Sampled from Poisson distributions ~P(λ)



01 ENVIRONMENT: Users

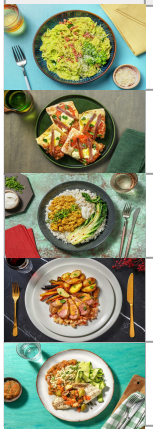
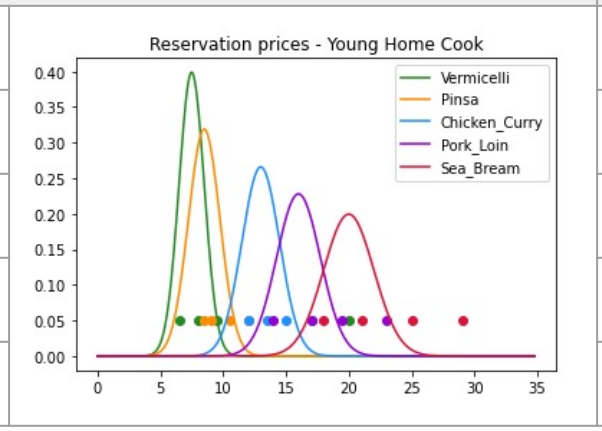
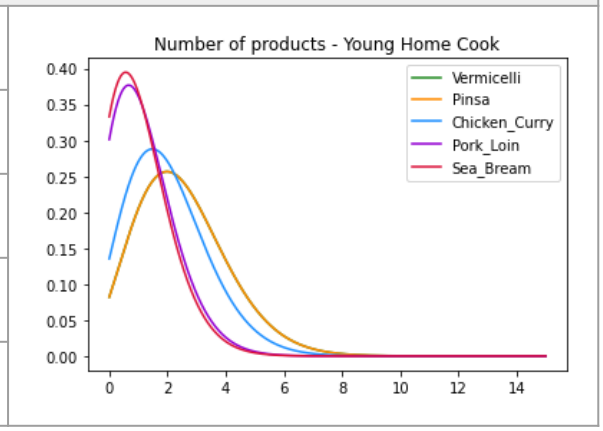
In the following are synthetized the characteristics that are looked for in the products by users and how they can be intuitively translated into the assumptions at the basis of the environment

Young home cook	 	• **Does not have much time**; • Has a **little experience**; • **Not** willing to **pay a lot**; • Enjoys cooking **in groups**.
Old home cook	 	• Enjoys **time-consuming receipts**; • Has **some experience**; • **Not** willing to **pay a lot**.
Experienced cook	 	• Enjoys spending **a lot of time** cooking; • Has a **lot of experience**; • Doesn't mind spending **a lot to have high quality products**

01 ENVIRONMENT: Users – Young Home Cook

The *young unexperienced cooks* are more likely to behave in the following way:

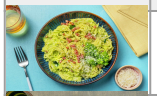
- Look for **low price recipes** → relative **higher μ** in cheaper recipes
- **Not** willing to spend **a lot of time** → relative **higher μ** in easy-to-do recipes
- Buy **more than one recipe** (to cook in groups) → **higher λ** in cheap and easy recipes

	RESERVATION PRICE		N° OF ITEMS	
	~ N (7.5, 1)		~ P (1.5)	
	~ N (8.5, 1.25)		~ P (1.5)	
	~N (13, 1.5)		~ P (1)	
	~ N (16 , 1.75)		~ P (0.2)	
	~ N (20 , 2)		~ P (0.1)	

01 ENVIRONMENT: Users – Young Home Cook

The *young unexperienced cooks* are more likely to behave in the following way:

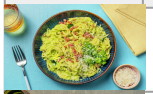
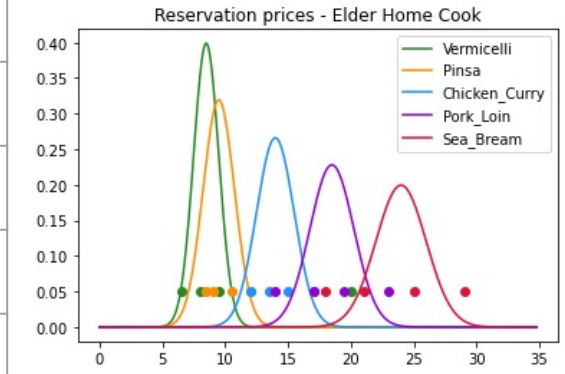
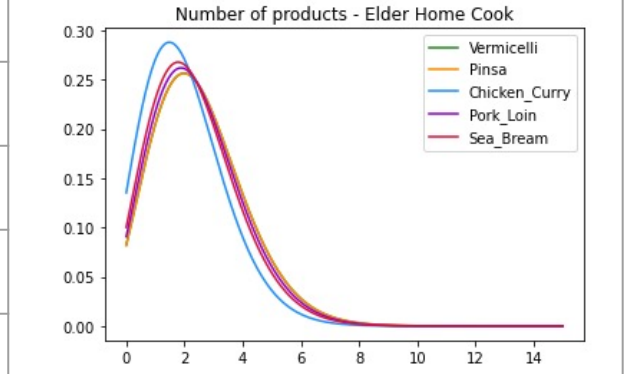
- Look for **low prices recipes** → **higher prob** to pass to cheap secondaries;
- **Not** willing to spend **a lot of time** → **higher prob** to pass to quick secondaries
- Start to search from **cheaper recipes** → **higher α-ratios** for cheap products

	PROBABILITY MATRIX		α - RATIOS
	*I,J = 1*	$Q = \begin{bmatrix} 0 & 0.4000 & 0.3000 & 0 & 0 \\ 0.4000 & 0 & 0.3000 & 0 & 0 \\ 0.5000 & 0 & 0 & 0.2000 & 0 \\ 0 & 0 & 0.4000 & 0 & 0.1000 \\ 0 & 0 & 0.4000 & 0.1000 & 0 \end{bmatrix}$	*~Dir(0.3, 0.3, 0.2, 0.1, 0.1)*
	*I,J = 2*		
	*I,J = 3*		
	*I,J = 4*		
	*I,J = 5*		

01 ENVIRONMENT: Users – Old Home Cook

The old unexperienced cookers are more likely to behave in the following way:

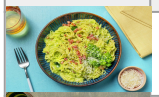
- Look for **low price recipes** → relative **higher μ** for cheaper recipes
- **Enjoys** spending **some time cooking** → relative **higher μ** for long recipes
- Enjoys **testing her skills** → relative **higher μ** for difficult recipes
- Cooks **alone but all possible product** → **average λ**

	RESERVATION PRICE		N° OF ITEMS	
	~ N (8.5 , 1)		~ P (1.5)	
	~ N (9.5, 1.25)		~ P (1.5)	
	~N (14, 1.5)		~ P (1)	
	~ N (18.5, 1.75)		~ P (1.4)	
	~ N (24, 2)		~ P (1.3)	

01 ENVIRONMENT: Users – Old Home Cook

The old unexperienced cookers are more likely to behave in the following way:

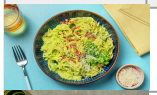
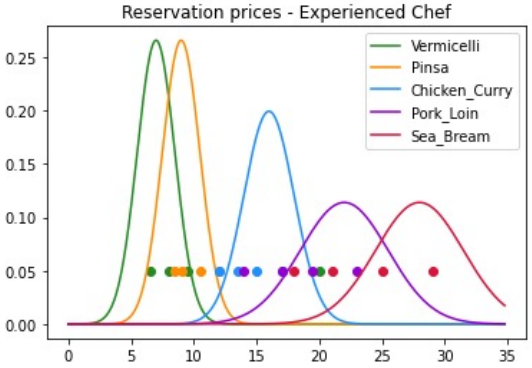
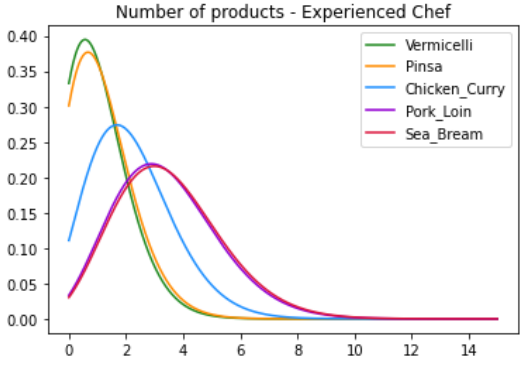
- Look for **low prices recipes** → **high prob** to pass to cheap secondaries
- **Enjoys** spending **some time cooking** → **high prob** time-consuming secondaries
- Enjoys **testing her skills** → **high prob**
- Start to search from **cheaper recipes** → **higher α-ratios** for cheap products

	PROBABILITY MATRIX		α - RATIOS
	*I,J = 1*	$Q = \begin{bmatrix} 0 & 0.3000 & 0.4000 & 0 & 0 \\ 0.3000 & 0 & 0.4000 & 0 & 0 \\ 0.3000 & 0 & 0 & 0.4000 & 0 \\ 0 & 0 & 0.3000 & 0 & 0.4000 \\ 0 & 0 & 0.3000 & 0.4000 & 0 \end{bmatrix}$	*~Dir(0.15, 0.15, 0.23, 0.23, 0.23)*
	*I,J = 2*		
	*I,J = 3*		
	*I,J = 4*		
	*I,J = 5*		

01 ENVIRONMENT: Users – Experienced Cook

The expert cookers are more likely to behave in the following way:

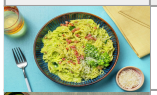
- Look for **high quality recipes** → **higher μ** for expensive recipes
- **Enjoys** spending **a lot of time cooking;** → **higher μ** for high-prep time recipes
- Enjoys **testing her skills** → **higher μ** for difficult recipes
- Cooks **both in groups and alone** → **variable λ**

	RESERVATION PRICE		N° OF ITEMS	
	~ N (7, 1.5)		~ P (0.1)	
	~ N (9, 1.5)		~ P (0.2)	
	~N (16, 2)		~ P (1.2)	
	~ N (22, 3.5)		~ P (2.4)	
	~ N (28, 3.5)		~ P (2.5)	

01 ENVIRONMENT: Users – Experienced Cook

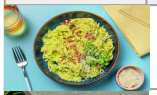
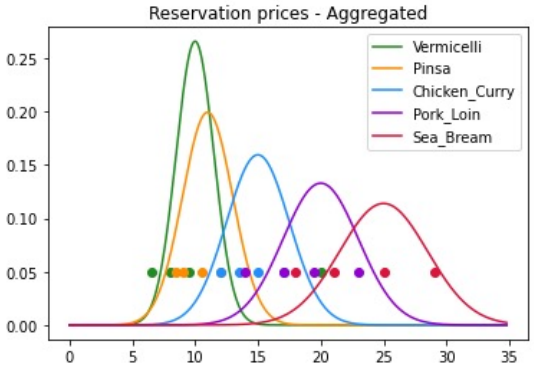
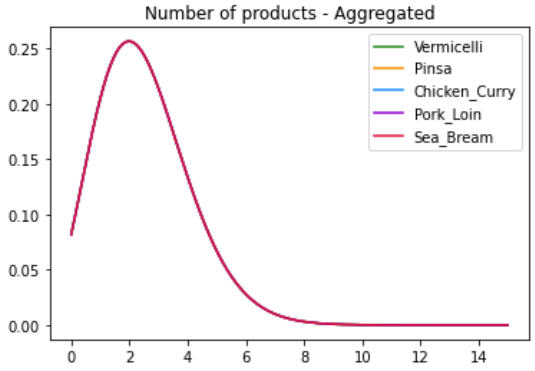
The expert cookers are more likely to behave in the following way:

- Look for **high quality recipes** → **high prob** to pass to expensive secondaries;
- **Enjoys** spending **a lot of time cooking** → **high prob** long lasting secondaries;
- Enjoys **testing her skills** → **high prob difficult secondaries**
- Start to search from **expensive recipes** → **higher α-ratios** for expensive products

	PROBABILITY MATRIX		- RATIOS
	*I,J = 1*	$Q = \begin{bmatrix} 0 & 0.2000 & 0.4000 & 0 & 0 \\ 0.2000 & 0 & 0.4000 & 0 & 0 \\ 0.2000 & 0 & 0 & 0.5000 & 0 \\ 0 & 0 & 0.1000 & 0 & 0.6000 \\ 0 & 0 & 0.2000 & 0.5000 & 0 \end{bmatrix}$	*~Dir(0.08, 0.08, 0.17, 0.33, 0.33)*
	*I,J = 2*		
	*I,J = 3*		
	*I,J = 4*		
	*I,J = 5*		

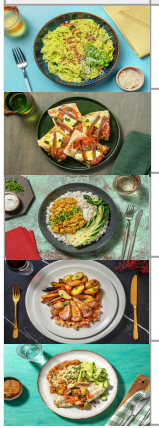
01 ENVIRONMENT: Aggregated User

The aggregated user is built given the probability of assigned to the two binary features that are **cooking experience** and **age.** Each parameter of the aggregated user is found by representing an average of the corresponding parameter for the 3 user categories selected.

	RESERVATION PRICE		N° OF ITEMS	
	~ N (10, 1,5)		~ P (1.5)	
	~ N (11, 2)		~ P (1.5)	
	~N (15 , 2.5)		~ P (1.5)	
	~ N (20, 3)		~ P (1.5)	
	~ N (25 , 3.5)		~ P (1.5)	

01 ENVIRONMENT: Aggregated User

The same as before holds for these parameters of the aggregated user.

	PROBABILITY MATRIX		α - RATIOS
	*I,J = 1*	$Q = \begin{bmatrix} 0 & 0.4000 & 0.3000 & 0 & 0 \\ 0.4000 & 0 & 0.4000 & 0 & 0 \\ 0.4000 & 0 & 0 & 0.4000 & 0 \\ 0 & 0 & 0.3000 & 0 & 0.4000 \\ 0 & 0 & 0.3000 & 0.4000 & 0 \end{bmatrix}$	$\sim Dir(0.2, 0.2, 0.2, 0.2, 0.2)$
	I,J = 2		
	I,J = 3		
	I,J = 4		
	I,J = 5		

02

OPTIMIZATION ALGORITHM

02 OPTIMIZATION ALGORITHM: Notation

Object	Name	Description
rp_i	Reservation price	On the demand side, it is the highest price that a buyer is willing to pay
p_i	Purchase probability	Likelihood that the product i price is greater that the reservation price
Q	Probability matrix	The element (i, j) is the probability to click on the first secondary product j once the primary i has been displayed
α_i	Alpha-ratios	Ratio of customers lending on the webpage in which an item is primary
n_i	Quantity	Number of items that are bought of a specific product
m_i	Margin	Difference between price and production cost
r_i	Return	Product between the margin and the quantity sold for an item
λ	Scaling factor	Likelihood to see the second secondary product (by assumption, the likelihood to see the first secondary is 1)

02 OPTIMIZATION ALGORITHM: Goal

The company's **objective** is to find the **optimal combination of prices of primary products** among the given ones in order to **maximize its revenues**, so the objective function we want to maximize is:

$$\sum_{j=1}^{3} \mathbb{P}(user_j) \sum_{i=1}^{5} \alpha_i \mathbb{E}[revenue\ path_{ij}]$$

where $\mathbb{E}[revenue\ path_{ij}]$ is the **expected revenue** of the path of visited products starting from product i for user j, thus:

$$\mathbb{E}[revenue\ path_{ij}] = \mathbb{P}(path_{ij}) \cdot revenue\ path_{ij}$$

where the probability of following a specific path is found by adding all the probabilities found considering the users' **purhcasing behaviours** at each visited product in the path and the associated collected **revenues**.

02 OPTIMIZATION ALGORITHM: Secondary Products

For each primary product, the two **secondaries are fixed** in this way:

Primary		Secondary – 1	Secondary - 2
Vermicelli	⟹	Pinsa	Chicken curry
Pinsa	⟹	Vermicelli	Chicken curry
Chicken curry	⟹	Vermicelli	Loin of pork
Loin of pork	⟹	Baked sea bream	Chicken curry
Baked sea bream	⟹	Loin of pork	Chicken curry

02 OPTIMIZATION ALGORITHM: Secondary Products

In the **not fully connected** case we consider that some products are not connected, so we have:

Primary		Secondary – 1	Secondary - 2
Vermicelli	⟹	Pinsa	Chicken curry
Pinsa	⟹	Vermicelli	Chicken curry
Chicken curry	⟹	Vermicelli	
Loin of pork	⟹	Baked sea bream	
Baked sea bream	⟹	Loin of pork	

This yields to the following **adjacency matrix** to describe our graph in the not fully connected case:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

02 OPTIMIZATION ALGORITHM: Purchasing Behaviour

To correctly compute the **expected revenue** to successfully perform the optimization of the product prices using the optimization algorithms we have to consider how a user can navigate the website and so all the possible **purchasing behaviours** (PB) of the users while doing so.

	PB0	PB1	PB2	PB3	PB4
Primary seen	✓	✓	✓	✓	✓
Primary bought		✓	✓	✓	✓
First Secondary clicked			✓		✓
Second Secondary clicked				✓	✓

Then the graph is explored considering **iteratively** the secondary products clicked as the new primary using the same structure of purchasing behaviour again.

02 OPTIMIZATION ALGORITHM: Probabilities

In the following table we show the **probability** of each **purchasing behaviour** (PB) of the users and the associated **collected revenues**.

Purchasing Behavior	Probability	Collected Revenues
PB0	$1 - p_i$	0
PB1	$p_i \cdot (1 - q_{i1}) \cdot (1 - \lambda q_{i2})$	r_i
PB2	$p_i \cdot q_{i1} \cdot (1 - \lambda q_{i2})$	r_i
PB3	$p_i \cdot (1 - q_{i1}) \cdot \lambda q_{i2}$	r_i
PB4	$p_i \cdot q_{i1} \cdot \lambda q_{i2}$	r_i

02 OPTIMIZATION ALGORITHM: Exploration Summary

To **summarize**, for each path, we compute the **probability of following it** and the **total revenue collected** while visiting the various recipes; then the **path's expected reward** is found as this whole collected revenue times the probability of following it.
Then, this is multiplied by the probability of **starting in a specific product** and the **probability of observing a specific user type**. Lastly, this quantity is sum over all starting products and all possible user types and the **total expected reward** of the website is found.

In the implementation we need to **be careful** of the following facts:
- a secondary product (recipe) becomes primary when it is clicked, and so we apply iteratively the algorithm to the new path.
- a primary recipe cannot become primary again. Technically speaking, a node cannot activate another time. Thus the exploration comes to an end in a finite time.

02 OPTIMIZATION ALGORITHM: Brute Force

We implemented a brute force algorithm that explores all possible paths and evaluates, in terms of expected reward, each price combination.

This algorithm has the following pros and cons:

CONS

Computationally **expensive** (more than the greedy algorithm)

PROS

Can be used to **compute** the **regret** of other smarter algorithms

Can **always find** the **optimal price** given the chosen candidates

02 OPTIMIZATION ALGORITHM: Greedy

[Iteration **0**] Initial configuration: each item is associated with its **lowest price**

[Iteration ***i***]

1. Create 5 new configurations among which to choose: they are obtained by increasing at each time the price of a single product of a single level.
2. Compute the marginal increase in each scenario.
3. Check if one of the scenarios has reward higher than the current highest reward.
4. If yes, update the optimal configuration and continue the algorithm.

   Otherwise stop the search.

The algorithm **monotonically increases** the prices and consequently also the expected reward

02 OPTIMIZATION ALGORITHM: Results (1/4)

Results for the **fully connected case** for both algorithms (brute force and greedy), considering the aggregated user

Aggregated User		
Product	Price *brute force*	Price *greedy*
Vermicelli	8	8
Pinsa	9	9
Chicken Curry	12	12
Loin of pork	17	17
Baked Sea Bream	21	21

OPTIMAL REWARD	*22.39*	*22.39*

02 OPTIMIZATION ALGORITHM: Results (2/4)

Results for the **fully connected case** for brute force algorithm, considering the three user categories separately.

			
Vermicelli	6.5	6.5	6.5
Pinsa	8.5	8.5	8.5
Chicken curry	12	12	13.5
Loin of Pork	14	17	17
Baked Sea Bream	18	21	25

OPTIMAL REWARD (disaggregated)	*9.26*	*19.90*	*36.30*
OPTIMAL REWARD (as mean)	*20.80*		
OPTIMAL REWARD (aggregated)	*19.37*		

02 OPTIMIZATION ALGORITHM: Results (3/4)

Results for the **not fully connected case** for both algorithms (brute force and greedy) considering the aggregated user

Aggregated User		
Product	Price *brute force*	Price *greedy*
Vermicelli	8	8
Pinsa	9	9
Chicken curry	13.5	13.5
Loin of pork	17	17
Baked sea bream	21	21

OPTIMAL REWARD	19.15	19.15

02 OPTIMIZATION ALGORITHM: Results (4/4)

Results for the **not fully connected case** for brute force algorithm, considering the three user categories separately

			
Vermicelli	6.5	6.5	6.5
Pinsa	8.5	8.5	8.5
Chicken curry	12	12	13.5
Loin of Pork	14	17	17
Baked Sea Bream	18	21	25

OPTIMAL REWARD (disaggregated)	*8.53*	*16.84*	*32.34*
OPTIMAL REWARD (as mean)	*17.71*		
OPTIMAL REWARD (aggregated)	*16.62*		

03

OPTIMIZATION WITH UNCERTAIN CONVERSION RATES

03 UNCERTAIN CONVERSION RATES: Introduction

Our aim in this section is to compute the optimal price combination in the case in which we have uncertain conversion rates, we will proceed with two different algorithms: **Thompson sampling** and **UCB1.** Both are described in the following slides, and followed by an overview of obtained results, in the cases of **Fully Connected** and **Not Fully Connected** graphs

03 UNCERTAIN CONVERSION RATES: Notation

Object	Name	Description
$p_{i,j}$	Prices Matrix	

Product	Price 1	Price 2	Price 3	Price 4
Vermicelli	6.5	8	9.5	11
Pinsa	8.5	9	10.5	12
Chicken curry	12	13.5	15	17
Loin of pork	14	17	19.5	23
Baked sea bream	18	21	25	29

03 UNCERTAIN CONVERSION RATES: Notation

Object	Name	Description
$P\ (rp > p_{i,j})$	Theoretical matrix of conversion rates	

Product	CR1	CR2	CR3	CR4
Vermicelli	0.99	0.91	0.63	0.25
Pinsa	0.89	0.84	0.60	0.31
Chicken curry	0.88	0.73	0.50	0.21
Loin of pork	0.98	0.84	0.57	0.16
Baked sea bream	0.98	0.87	0.50	0.13

NOTE:
Only **conversion rates** are considered **unknown** in this step, scaling factor λ, α – ratios, and parameters from Poisson distributions and from the graph are known.

03 UNCERTAIN CONV RATES: Thompson sampling

We created a learner that follows TS algorithm in the following way, simulating the website evolution differently every day:

1. We sample conversion rates from a Beta Distribution
2. Greedy optimizer runs and computes the optimal price configuration
3. We receive information about the obtained price configuration and we update parameters accordingly:
    - α gets increased by the number of purchased items with new prices configuration
    - β gets increased by the number of visits without purchase
4. Simulate following day with new sampling from updated Beta's distribution

$PseudoCode$: 1) $at\ time\ t\ for\ every\ price\ configuration\ p$: $Sample\ \mathrm{P}\ (\mu_p = \theta_p) \rightarrow \widetilde{\theta_p}$

2) $run\ greedy\ optimization\ to\ obtain\ best\ p_t$: $ArgMax_{p \in P}(\widetilde{\theta_p}) \rightarrow p_t$

3) $Update\ Beta\ distribution\ given\ p_t$: $(\alpha_{p_t}, \beta_{p_t}) + (x_{p_t,t}, 1 - x_{p_t,t}) \rightarrow (\alpha_{p_t}, \beta_{p_t})$

03 UNCERTAIN CONV RATES: UCB1

We created a learner that follows UCB1 algorithm in the following way, simulating the website evolution differently every day:

1. We create a conversion rates matrix of zeros and upper bounded by one
2. Greedy optimizer runs and computes the optimal price configuration and optimal reward
3. The learner pulls this arm
4. We receive information about the obtained price configuration at the end of the day and we update parameters (mean and width of the interval of conversion rates) accordingly:
   - $New\ mean = \frac{\#items\ bought}{\#\ times\ item\ visited\ as\ first}$ *(according to the pulled arm)*
   - *New width: Play once every price configuration p*

     $$At\ every\ time\ t\ play\ price\ conf\ p_t\ s.t.: ArgMax_{p \in P}\{\overline{x_p} + \sqrt{\frac{2\log(t)}{n_p(t-1)}}\}$$

     Where p_t is the price configuration at time t , n_p is the number of times a product is observed given price conf p and t is the number of days already passed
5. Simulate following day with updated conversion rates

03 UNCERTAIN CONV RATES: Results

RESULTS FOR FULLY CONNECTED GRAPH

03 UNCERTAIN CONVERSION RATES: Results - FC

We give a look at main results for the **Fully Connected** case obtained with the two algorithms:

TS

Product	CR1	CR2	CR3	CR4
Vermicelli	0.96	0.91	0.68	0.52
Pinsa	0.90	0.84	0.64	0.52
Chicken curry	0.89	0.74	0.60	0.49
Loin of pork	0.96	0.84	0.57	0.41
Baked sea bream	0.96	0.87	0.54	0.5

UCB1

Product	CR1	CR2	CR3	CR4
Vermicelli	1	0.90	0.54	0.21
Pinsa	0.91	0.85	0.62	0.26
Chicken curry	0.89	0.71	0.57	0.23
Loin of pork	1	0.85	0.60	0.15
Baked sea bream	1	0.87	0.44	0.09

03 UNCERTAIN CONVERSION RATES: Results - FC

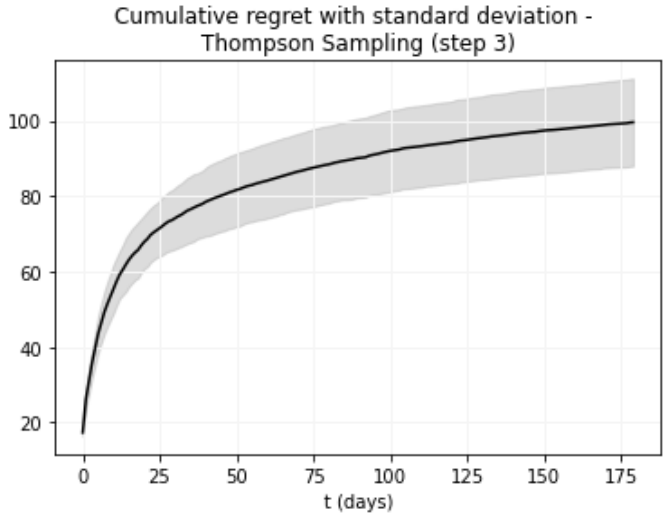
The ratio with respect to theoretical upper bound for the two algorithms are the following:

TS	UCB1
0.1977	0.0723

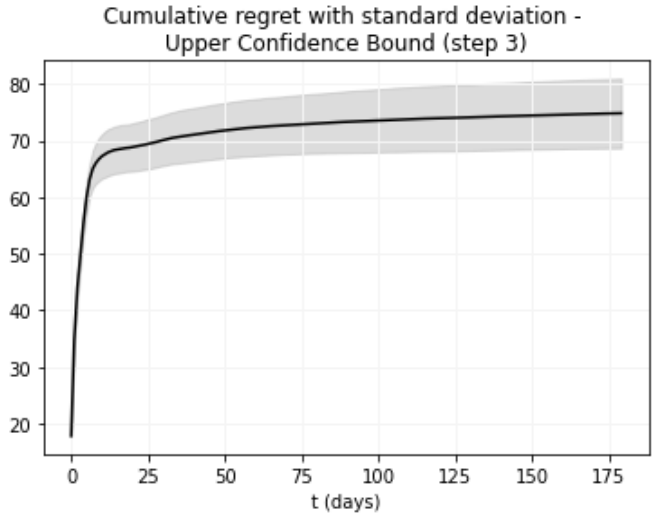
03 UNCERTAIN CONVERSION RATES: Results - FC

We also plot some other important information regarding the obtained results, starting with cumulative regret:

TS



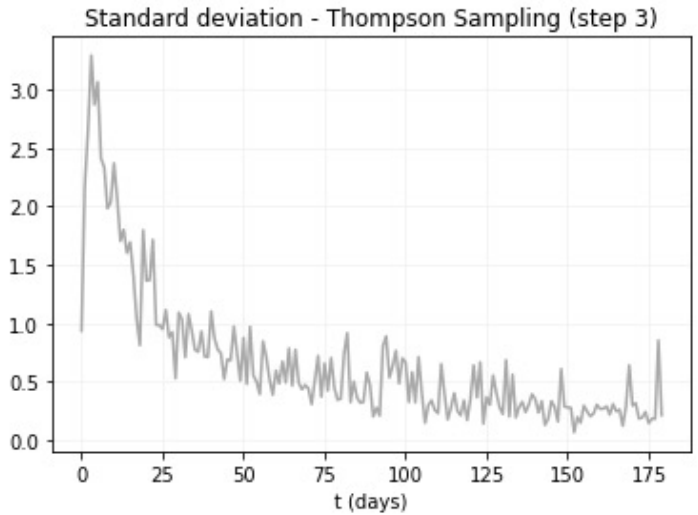
UCB1



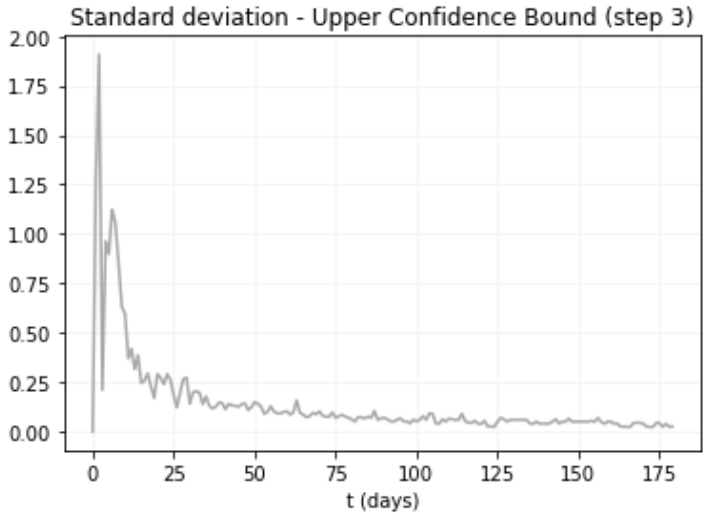
03 UNCERTAIN CONVERSION RATES: Results - FC

Standard Deviation of Cumulative Regret:

TS



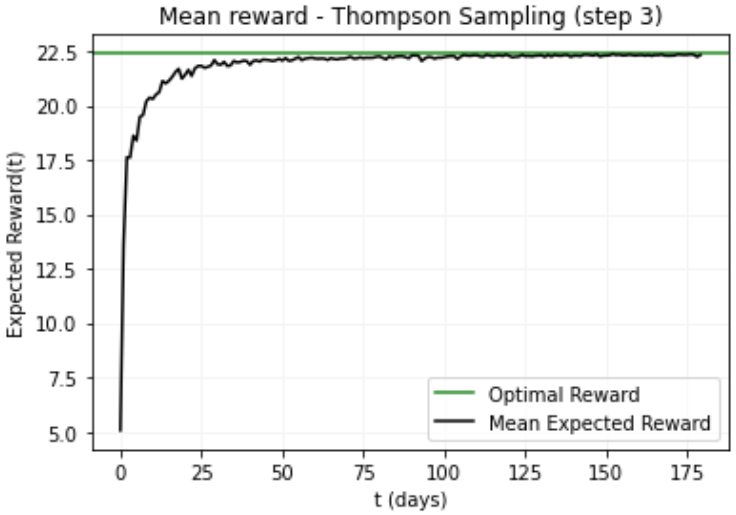
UCB1



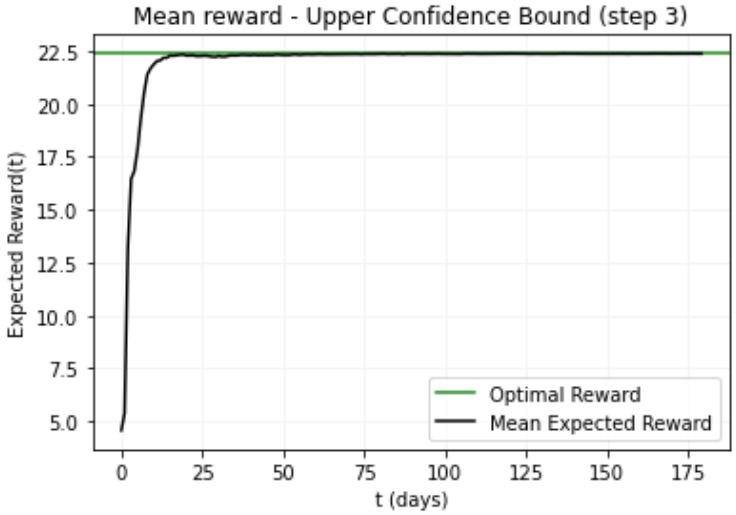
03 UNCERTAIN CONVERSION RATES: Results - FC

Comparison between optimal reward and expected one:

TS



UCB1



03 UNCERTAIN CONV RATES: Comparison TS vs UCB1

What we can observe from these result is:

- The two models in general have a comparable performance
- TS and UCB1 seem to obtain conversion rates values which are exactly as the theoretical ones for the selected configuration (the same selected by the greedy algorithm, indeed we reach the same reward)
- On the other hand for other configurations the obtained conversion rates from UCB1 seem to be a little more precise
- This can find a reasonable explanation if looking at the standard deviation plots, indeed we can deduct that UCB1 is focusing on some combination of prices, and this leads to a reduced standard deviation with respect to TS. UCB1 trying less prices combinations will have better overall results but less precise for the final selected configuration.
- For cumulative regrets instead they are both increasing which is trivial, and seem comparable, even though the one from UCB1 seems to grow more at the beginning and then seems to flatten much more if compared to the one of TS
- We plot also the mean expected reward against the optimal one and we see that it converges.

03 UNCERTAIN CONV RATES: Results

RESULTS FOR NOT FULLY CONNECTED GRAPH

03 UNCERTAIN CONVERSION RATES: Results - NFC

We analysed now the estimated conversion rates for the **Not Fully Connected** case, which gives similar results to the previous one:

TS

Product	CR1	CR2	CR3	CR4
Vermicelli	0.96	0.92	0.70	0.47
Pinsa	0.90	0.84	0.63	0.46
Chicken curry	0.89	0.72	0.57	0.39
Loin of pork	0.96	0.83	0.60	0.40
Baked sea bream	0.96	0.88	0.54	0.37

UCB1

Product	CR1	CR2	CR3	CR4
Vermicelli	1	0.91	0.62	0.32
Pinsa	0.89	0.84	0.59	0.27
Chicken curry	0.89	0.65	0.55	0.17
Loin of pork	1	0.84	0.59	0.15
Baked sea bream	1	0.87	0.47	0.05

03 UNCERTAIN CONVERSION RATES: Results - NFC

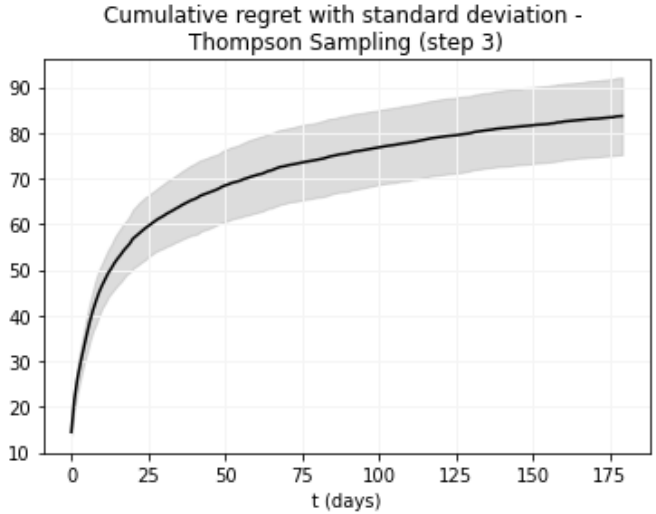
The ratio with respect to theoretical upper bound for the two algorithms are the following:

TS	UCB1
0.1395	0.0739

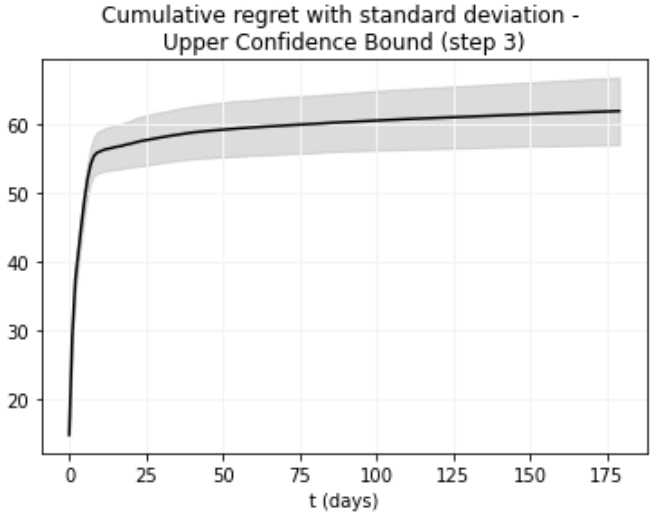
03 UNCERTAIN CONVERSION RATES: Results - NFC

We also plot some other important information regarding the obtained results, starting with cumulative regret:

TS



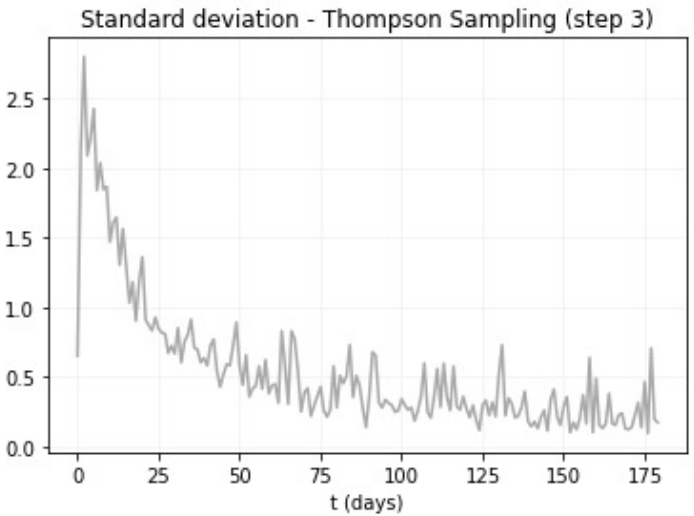
UCB1



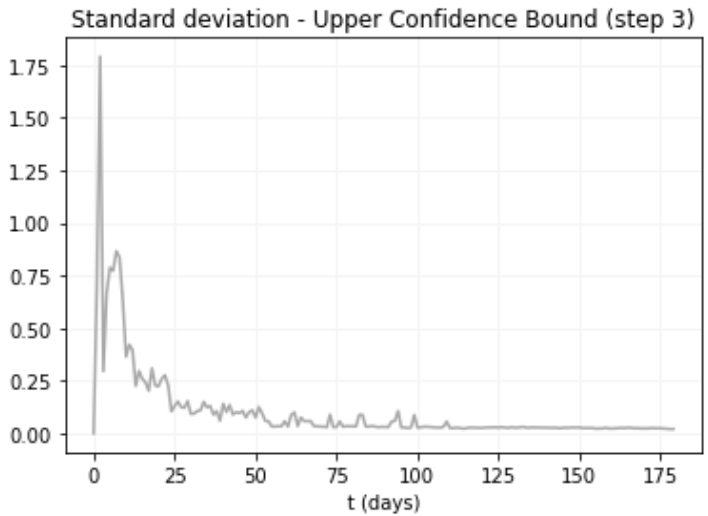
03 UNCERTAIN CONVERSION RATES: Results - NFC

Standard deviation of Cumulative Regret:

TS



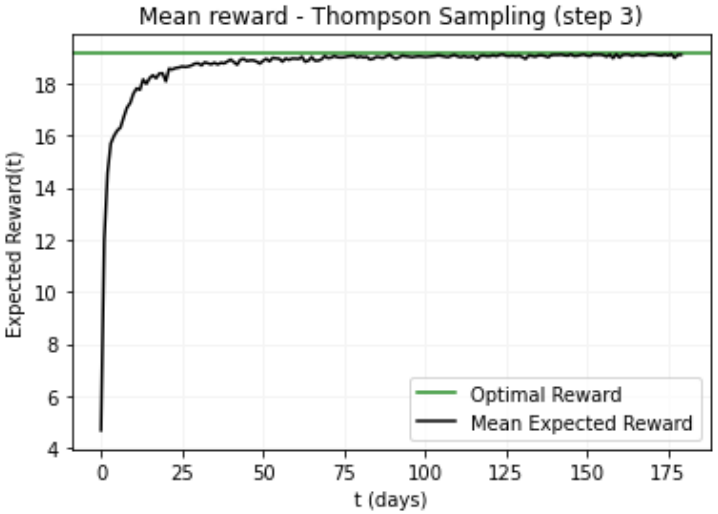
UCB1



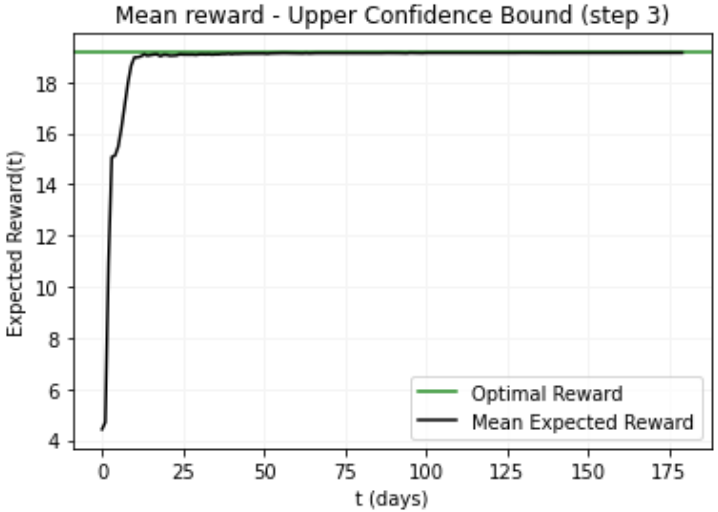
03 UNCERTAIN CONVERSION RATES: Results - NFC

Comparison between optimal reward and expected one:

TS



UCB1



03 UNCERTAIN CONV RATES: Comparison FC vs NFC

The results obtained for the not fully connected case are very similar to the ones obtained with the fully connected ones, but it can be noticed that:

- The estimation of conversion rates is a bit less precise in the NFC case
- Optimal reward with both models (TS and UCB1) is decreased passing from FC to NFC case

04

OPTIMIZATION WITH UNCERTAIN CONVERSION RATES, α-RATIOS & NUMBER OF ITEMS SOLD

04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Notation

Object	Name	Description
$p_{i,j}$	Prices Matrix	

Product	Price 1	Price 2	Price 3	Price 4
Vermicelli	6.5	8	9.5	11
Pinsa	8.5	9	10.5	12
Chicken curry	12	13.5	15	17
Loin of pork	14	17	19.5	23
Baked sea bream	18	21	25	29

04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Notation

In the following step the aim is to compute the optimal prices when 3 quantities are unknown:

- Conversion rates
- $\boldsymbol{\alpha}$-ratios
- Number of items sold per product

CONVERSION RATES

They are computed as step 3:

$P\ (rp > p_{i,j}) = \begin{bmatrix} 0.99 & 0.91 & 0.63 & 0.25 \\ 0.89 & 0.84 & 0.60 & 0.31 \\ 0.88 & 0.73 & 0.50 & 0.21 \\ 0.98 & 0.84 & 0.57 & 0.16 \\ 0.98 & 0.87 & 0.50 & 0.13 \end{bmatrix}$

α-RATIOS

Sampled from a multinomial Dirichelet distribution:

$\sim Dir\,(0.2, 0.2, 02, 0.2, 0.2)$

Nº ITEMS SOLD

Sampled from five Poisson Distributions which have these parameters

- P(2.5)
- P(2.5)
- P(2.5)
- P(2.5)
- P(2.5)

Remark: The values shown above are the theoretical quantities

04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Thompson Sampling

TS algorithm in this step must learn conversion rates, α-ratios and number of products at the same time:

1. The **initial conversion rates** and ***α-ratios*** are sampled from a Beta distribution $\beta(25,1)$.
   The means of the Poisson distributions are sampled from a Gamma distribution $\Gamma(a, 1/b)$
   a:=number of products bought during a purchase; b:=number of purchases, $b \leq a$.
2. Compute optimal configuration and update distributions' parameters at the end of the day :

3. For **conversion rates**
   - α gets increased by the number of purchased items with new prices configuration
   - β gets increased by the number of visits without purchase

4. For ***α-ratios :***
   - α gets increased by the number of times the product has been displayed as a primary
   - β gets increased by the number of times another product different from the one we are learning is displayed as a primary.

5. For **N° of items** the parameter a gets increased by the number of products bought for kind of receipt and the parameter b is increased by the number of purchases.

04 UNCERTAIN RATES, α-RATIOS & No OF PRODUCTS : Thompson Sampling

The above procedure is performed a great-enough number of times until the result is obtained

$PseudoCode$:

- $@\ time\ t\ for\ every\ price\ configuration\ p$: $Sample\ \beta_1(\mu_p = \theta_p) \rightarrow \widetilde{\theta_p}$
  $Sample\ \beta_2(\mu_p = \theta_p) \rightarrow \widetilde{\theta_p}$
  $Sample\ \Gamma\ (\mu_p = \theta_p) \rightarrow \widetilde{\theta_p}$
- $Run\ greedy\ optimization\ to\ obtain\ best\ p_t$: $ArgMax_{p \in P}(\widetilde{\theta_p}) \rightarrow p_t$
- $Update\ Beta\ and\ Gamma\ distributions\ given\ p_t\ as\ follows$:

  $-(\alpha 1_{p_t}, \beta 1_{p_t}) + (x1_{p_t,t}, 1 - x1_{p_t,t}) \rightarrow (\alpha 1_{p_t}, \beta 1_{p_t})$
  $-(\alpha 2_{p_t}, \beta 2_{p_t}) + (x2_{p_t,t}, 1 - x2_{p_t,t}) \rightarrow (\alpha 2_{p_t}, \beta 2_{p_t})$
  $-(a_{p_t}, b_{p_t}) + (x_{p_t,t}, 1 - x_{p_t,t}) \rightarrow (a_{p_t}, b_{p_t})$

04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : UCB1

We created a learner that follows UCB1 algorithm in the following way, simulating the website evolution differently every day:

1. A conversion rates matrix is initialized to zeros with upper bound ones;
2. Greedy optimizer runs and computes the optimal price configuration and optimal reward
3. The learner pulls this arm
4. For **conversion rates** we update parameters (mean and width of the interval) accordingly:
   - $New\ mean = \frac{\#items\ bought}{\#\ times\ item\ visited\ as\ first}$ *(according to the pulled arm)*
   - *New width: Play once every price configuration p*

   At every time t play price conf p_t *s.t. :* $ArgMax_{p \in P}\ \{\overline{x_p} + \sqrt{\frac{2\log(t)}{n_p(t-1)}}\}$

   Where p_t *is the price configuration at time t ,* n_p *is the number of times a product is observed given price conf p and t is the number of days already passed*
5. For **α - ratios** we update the mean accordingly: $New\ mean = \frac{\#\ product\ displayed\ as\ primary}{\#\ times\ item\ is\ dysplayed}$
6. For **Nº of products**: $New\ mean = \frac{\#items\ bought}{\#\ n^o purchases}$

RESULTS FOR FULLY CONNECTED GRAPH

04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Results - FC

We analysed the results for the **Fully Conncted** case, starting with the estimated conversion rates:

TS

Product	CR1	CR2	CR3	CR4
Vermicelli	0.96	0.91	0.68	0.56
Pinsa	0.89	0.83	0.63	0.46
Chicken curry	0.88	0.74	0.60	0.46
Loin of pork	0.96	0.84	0.63	0.38
Baked sea bream	0.96	0.87	0.55	0.35

UCB1

Product	CR1	CR2	CR3	CR4
Vermicelli	1	0.90	0.65	0.29
Pinsa	0.89	0.83	0.52	0.29
Chicken curry	0.89	0.66	0.57	0.34
Loin of pork	1	0.84	0.59	0.17
Baked sea bream	1	0.87	0.38	0.05

04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Results - FC

The estimated α-ratios and the average N°of products are:

TS

Product	α	λ
Vermicelli	0.2022	2.4982
Pinsa	0.1969	2.4876
Chicken curry	0.2011	2.5060
Loin of pork	0.1986	2.5100
Baked sea bream	0.2013	2.4924

UCB1

Product	α	λ
Vermicelli	0.2101	2.4969
Pinsa	0.2060	2.5082
Chicken curry	0.1939	2.4922
Loin of pork	0.1989	2.5504
Baked sea bream	0.1912	2.4889

04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Results - FC

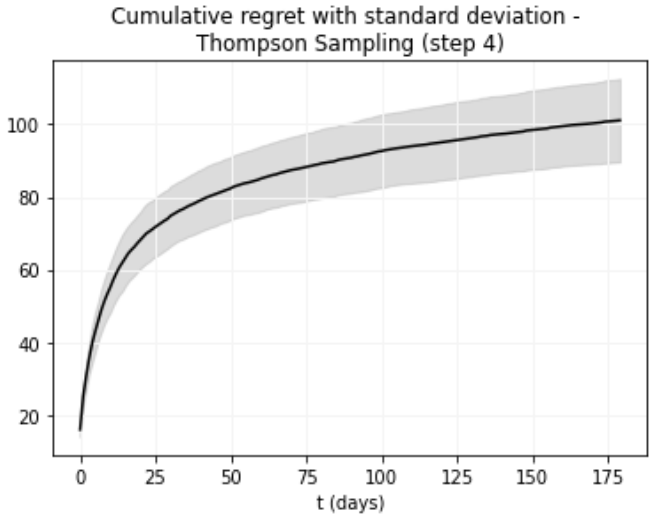
The ratio with respect to the theoretical upper bound for the two algorithms are the following:

TS	UCB1
0.1987	0.0727

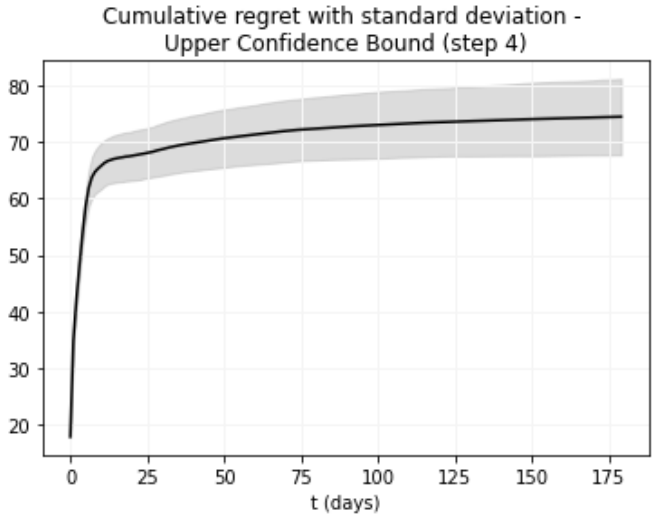
04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Results - FC

We also plot some other important informations regarding the obtained results, starting with cumulative regret:

TS



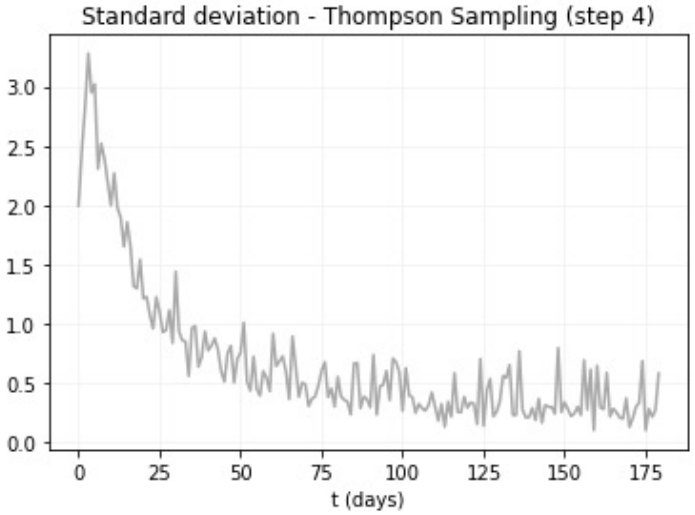
UCB1



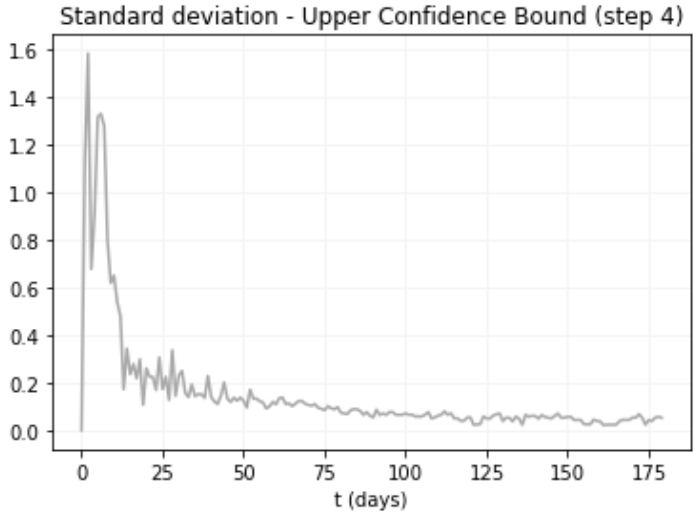
04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Results - FC

Standard deviation of Cumulative Regret:

TS



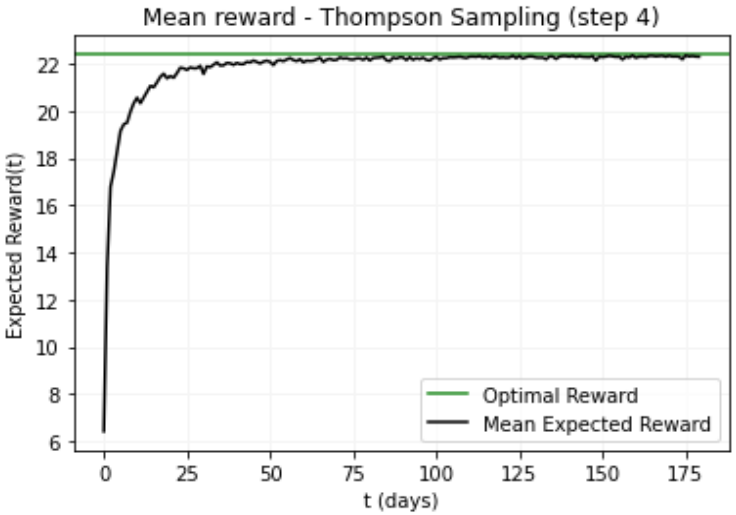
UCB1



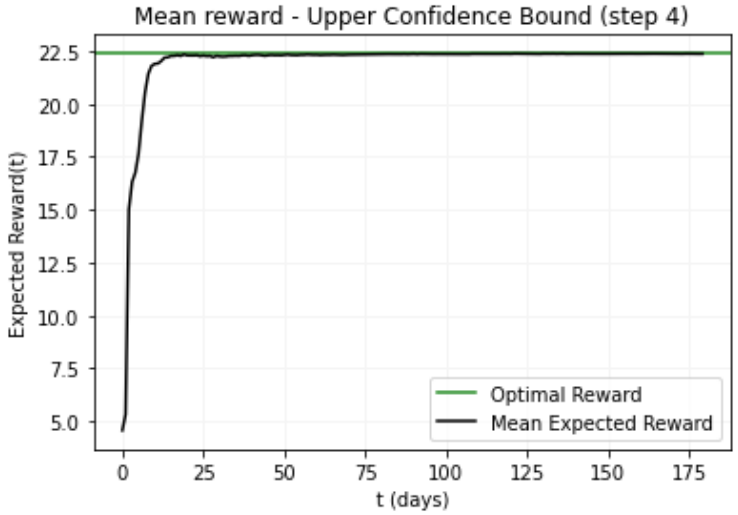
04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Results - FC

Comparison between optimal reward and expected one:

TS



UCB1



04 UNCERTAIN RATES, α-RATIOS & No OF PRODUCTS : TS vs UCB1

- The two models in general have a comparable performance regarding conversion rates.
- TS seems to have better performance for the α – ratios and for the average Nº of products

From the graphs we observe that:

- TS has a higher standard deviation (because it explores more configurations) and slower convergence.
- In contrast, UCB1 has a lower standard deviation, and we deduct that it is focusing on less configurations .
- The ratio with respect to theoretical upper bound for the TS significantly decreases in the NFC case.

RESULTS FOR NOT FULLY CONNECTED GRAPH

04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Results - NFC

We analysed the results for the **NOT Fully Conncted** case, starting with the estimated conversion rates:

TS

Product	CR1	CR2	CR3	CR4
Vermicelli	0.96	0.91	0.64	0.54
Pinsa	0.90	0.85	0.61	0.57
Chicken curry	0.87	0.73	0.54	0.46
Loin of pork	0.96	0.85	0.60	0.38
Baked sea bream	0.96	0.87	0.54	0.41

UCB1

Product	CR1	CR2	CR3	CR4
Vermicelli	1	0.90	0.68	0.19
Pinsa	0.90	0.83	0.56	0.23
Chicken curry	0.88	0.70	0.51	0.13
Loin of pork	1	0.84	0.59	0.15
Baked sea bream	1	0.87	0.52	0.04

04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Results - NFC

The estimated **α**-ratios and average N° of products are:

TS

Product	α	λ
Vermicelli	0.2019	2.4956
Pinsa	0.2020	2.4761
Chicken curry	0.1952	2.5044
Loin of pork	0.2050	2.5169
Baked sea bream	0.1960	2.4679

UCB1

Product	α	λ
Vermicelli	0.2025	2.4990
Pinsa	0.1954	2.5197
Chicken curry	0.2044	2.5283
Loin of pork	0.1959	2.4893
Baked sea bream	0.2018	2.4844

04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Results - NFC

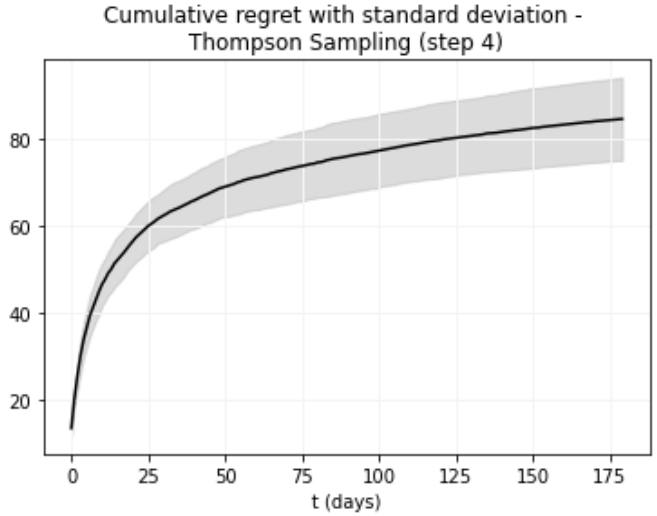
The ratio with respect to theoretical upper bound for the two algorithms are the following:

TS	UCB1
0.1392	0.0740

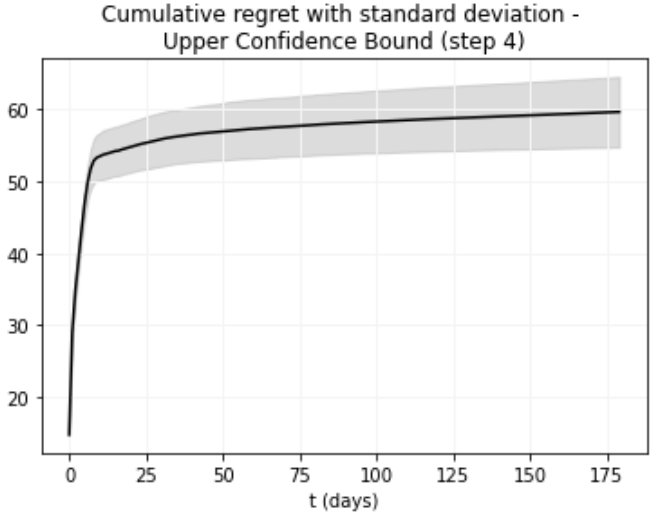
04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Results - NFC

We also plot some other important information regarding the obtained results, starting with cumulative regret:

TS



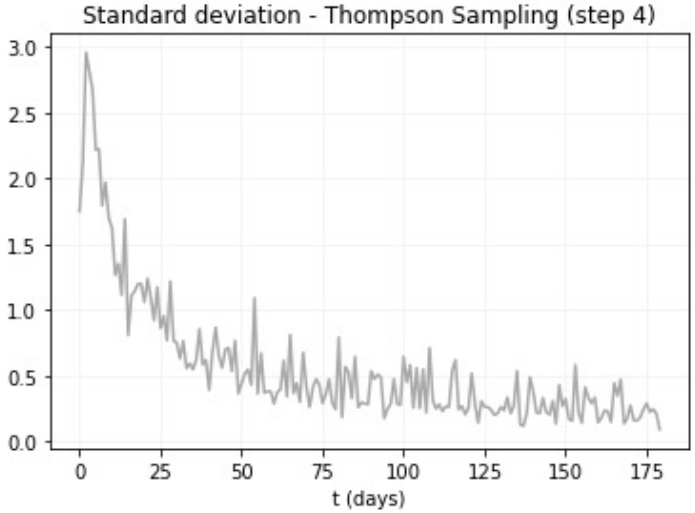
UCB1



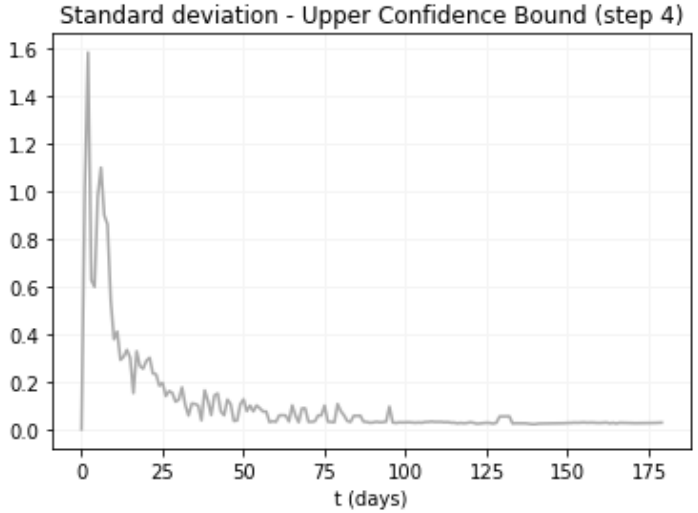
04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Results - NFC

Standard deviation of Cumulative Regret:

TS



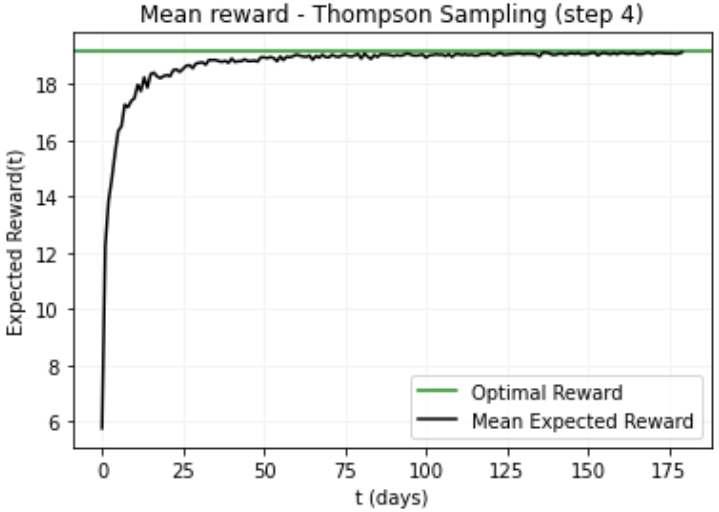
UCB1



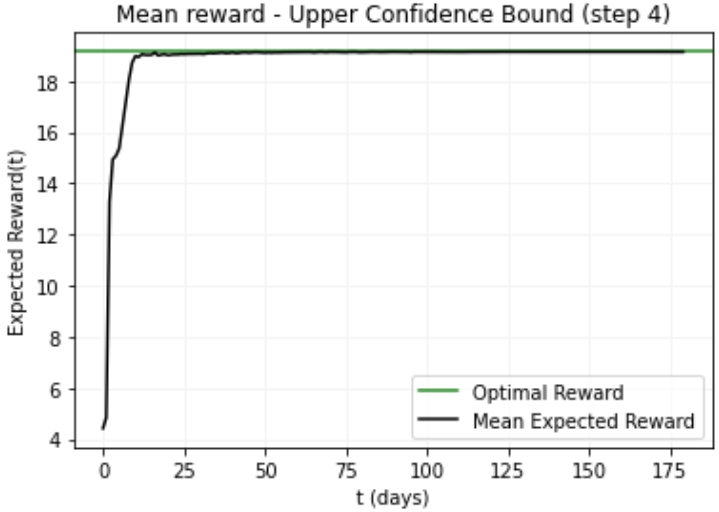
04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Results - NFC

Comparison between optimal reward and expected one:

TS



UCB1



04 UNCERTAIN RATES, α-RATIOS & Nº OF PRODUCTS : Results - NFC

The results obtained for the not fully connected case are very similar to the ones obtained with the fully connected ones, but it can be noticed that:

- Optimal reward and Regret with both models (TS and UCB1) is decreased passing from FC to NFC case
- The ratio with respect to theoretical upper bound for the TS significantly decreases in the NFC case.

05

OPTIMIZATION WITH UNCERTAIN CONVERSION RATES AND GRAPH WEIGHTS

05 UNCERTAIN GRAPH WEIGHTS: Notation

Object	Name	Description
$P\ (rp > p_{i,j})$	Theoretical matrix of conversion rates	

Product	CR1	CR2	CR3	CR4
Vermicelli	0.99	0.91	0.63	0.25
Pinsa	0.89	0.84	0.60	0.31
Chicken curry	0.88	0.73	0.50	0.21
Loin of pork	0.98	0.84	0.57	0.16
Baked sea bream	0.98	0.87	0.50	0.13

05 UNCERTAIN GRAPH WEIGHTS: Notation

Object	Name	Description
$Q_{i,j}$	Theoretical matrix of graph weights	

Product	Vermicelli	Pinsa	Chicken	Pork	Sea Bream
Vermicelli	0	0.4	0.21	0	0
Pinsa	0.4	0	0.28	0	0
Chicken curry	0.4	0	0	0.28	0
Loin of pork	0	0	0.21	0	0.4
Baked sea bream	0	0	0.21	0.4	0

NOTE:
Given λ the final matrix to estimate is $Q_{i,j}$ where for every row (corresponding to a product) the entries of the secondaries for that product are multiplied by λ.

05 UNCERTAIN GRAPH WEIGHT: Thompson sampling

Similarly to what we did in step 3:

1. We sample **conversion rates** and **graph weights** from a Beta Distribution
2. Greedy optimizer runs and computes the optimal price configuration and the corresponding reward
3. We receive information about the obtained price configuration and we update the parameters accordingly:
    - For conversion rates: α and β are updated as shown in step 3.
    - For graph weights: α get increased by the number of times the users click on a product and β by the number of times the product is visualized but the user does not click on it, even if he could.
4. Simulate the following day with new sampling, starting from the updated Beta's distributions

05 UNCERTAIN GRAPH WEIGHT: UCB1

As in step 3 UCB1 algorithm works in the following way, simulating the website evolution differently every day:

1. The **conversion rates** and **graph weights** are initialized to zeros with upper bound ones.
2. Greedy optimizer runs and computes the optimal price configuration and optimal reward.
3. The learner pulls this arm
4. For **conversion rates** we update parameters (mean and width of the interval) accordingly:

   - $New\ mean = \frac{\#items\ bought}{\#\ times\ item\ visited\ as\ first}$ *(according to the pulled arm)*
   - *New width: - Play once every price configuration p*

     - At every time t play price conf p_t *s.t. :* $ArgMax_{p \in P}\{\overline{x_p} + \sqrt{\frac{2\log(t)}{n_p(t-1)}}\}$

   Where p_t *is the price configuration at time t ,* n_p *is the number of times a product is observed given price conf p and t is the number of days already passed*

05 UNCERTAIN GRAPH WEIGHT: UCB1

5. For **graph weights** we update parameters (mean and width of the interval) accordingly:

   - $New\ mean = \frac{\#clicks\ on\ a\ product}{\#\ times\ user\ didn't\ click}$ *(according to the pulled arm)*
   - *New width: Updated with same formula:* $\sqrt{\frac{2\log(t)}{n_p(t-1)}}$ *so we will have the same upper bound for both conversion rates and graph weights*

   Where p_t *is the price configuration at time t ,* n_p *is the number of times a product is observed given price conf p and t is the number of days already passed*

6. Simulate following day with updated conversion rates and graph weights

05 UNCERTAIN GRAPH WEIGHTS: Results

RESULTS FOR FULLY CONNECTED GRAPH

05 UNCERTAIN GRAPH WEIGHTS: Results - FC

We give a look at main results for the **Fully Connected** case obtained with the two algorithms, starting with the estimated conversion rates:

TS

Product	CR1	CR2	CR3	CR4
Vermicelli	0.98	0.91	0.64	0.64
Pinsa	0.88	0.84	0.62	0.42
Chicken curry	0.89	0.73	0.57	0.49
Loin of pork	0.96	0.84	0.61	0.41
Baked sea bream	0.96	0.87	0.56	0.39

UCB1

Product	CR1	CR2	CR3	CR4
Vermicelli	1	0.91	0.63	0.18
Pinsa	0.89	0.84	0.53	0.35
Chicken curry	0.88	0.74	0.40	0.30
Loin of pork	1	0.85	0.61	0.20
Baked sea bream	1	0.88	0.51	0.1

05 UNCERTAIN GRAPH WEIGHTS: Results - FC

Now we analyse estimated graph weights:

TS

Product	P1	P2	P3	P4	P5
P1	0	0.39	0.21	0	0
P2	0.40	0	0.29	0	0
P3	0.40	0	0	0.28	0
P4	0	0	0.23	0	0.41
P5	0	0	0.22	0.40	0

UCB1

Product	P1	P2	P3	P4	P5
P1	0	0.40	0.20	0	0
P2	0.41	0	0.29	0	0
P3	0.41	0	0	0.29	0
P4	0	0	0.22	0	0.41
P5	0	0	0.20	0.40	0

05 UNCERTAIN GRAPH WEIGHTS: Results - FC

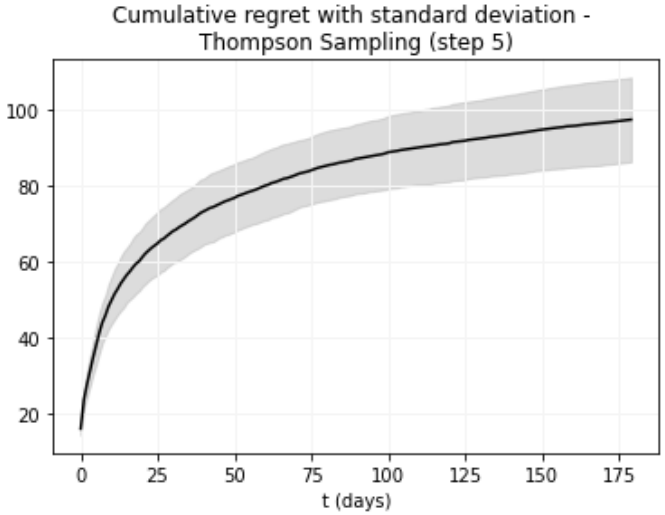
The ratio with respect to theoretical upper bound for the two algorithms are the following:

TS	UCB1
0.1932	0.0677

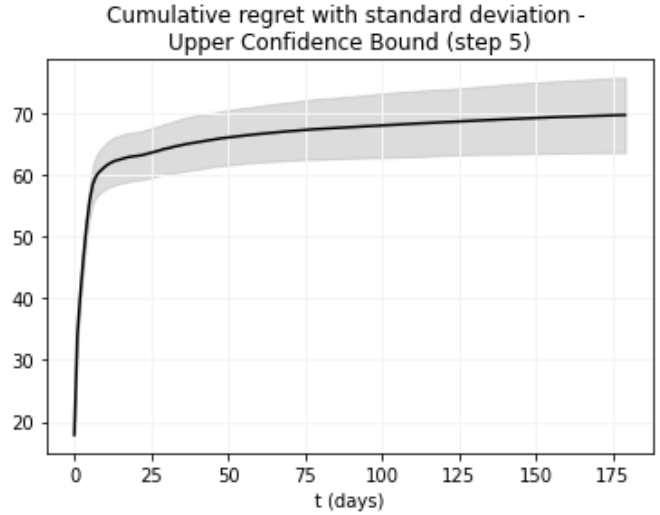
05 UNCERTAIN GRAPH WEIGHTS: Results - FC

We also plot some other important information regarding the obtained results, starting with cumulative regret:

TS



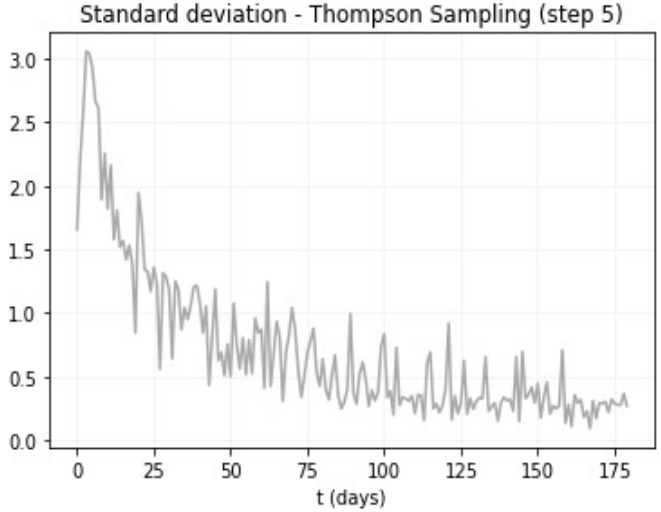
UCB1



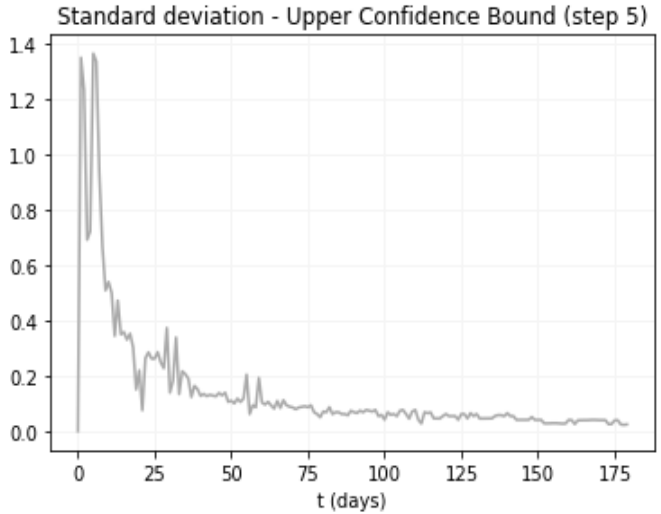
05 UNCERTAIN GRAPH WEIGHTS: Results - FC

Standard Deviation of Cumulative Regret:

TS



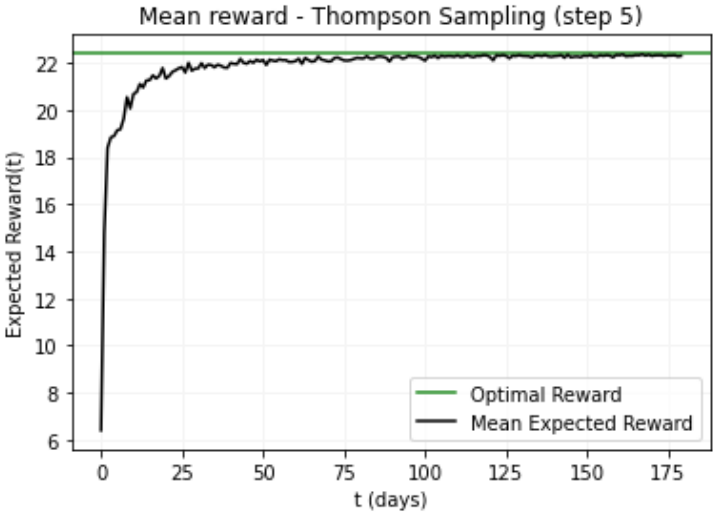
UCB1



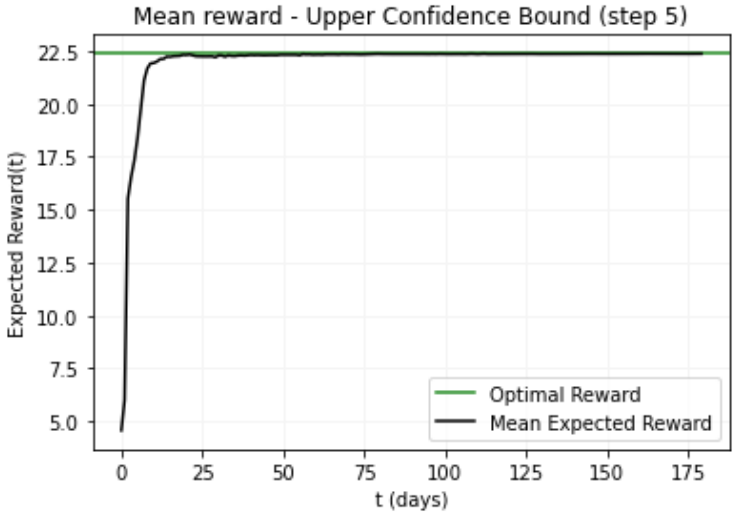
05 UNCERTAIN GRAPH WEIGHTS: Results - FC

Comparison between optimal reward and expected one:

TS



UCB1



05 UNCERT GRAPH WEIGHT: Comparison TS vs UCB1

What we can observe from these results is:

- The two models in general have a comparable performance both for graph weights and conversion rates.
- TS has a standard deviation much greater with respect to UCB1
- The UCB1 algorithm seems to converge sooner, indeed cumulative regret flattens before
- The expected reward, indeed, tends to the optimal value in the both cases.

05 UNCERTAIN GRAPH WEIGHTS: Results

RESULTS FOR NOT FULLY CONNECTED GRAPH

05 UNCERTAIN GRAPH WEIGHTS: Results - NFC

We analysed now the same results but for the **Not Fully Conncted** case, which gives similar results to the previous one. We start again from estimated conversion rates:

TS

Product	CR1	CR2	CR3	CR4
Vermicelli	0.96	0.91	0.68	0.29
Pinsa	0.89	0.85	0.64	0.47
Chicken curry	0.87	0.72	0.55	0.39
Loin of pork	0.96	0.85	0.61	0.42
Baked sea bream	0.96	0.88	0.55	0.37

UCB1

Product	CR1	CR2	CR3	CR4
Vermicelli	1	0.90	0.60	0.19
Pinsa	0.90	0.84	0.62	0.39
Chicken curry	0.88	0.74	0.53	0.20
Loin of pork	1	0.84	0.50	0.11
Baked sea bream	1	0.88	0.50	0.04

05 UNCERTAIN GRAPH WEIGHTS: Results - NFC

Now we analysed estimated graph weights:

TS

Product	P1	P2	P3	P4	P5
P1	0	0.41	0.23	0	0
P2	0.41	0	0.27	0	0
P3	0.41	0	0	0.02	0
P4	0	0	0.01	0	0.40
P5	0	0	0.01	0.40	0

UCB1

Product	P1	P2	P3	P4	P5
P1	0	0.40	0.20	0	0
P2	0.40	0	0.29	0	0
P3	0.41	0	0	0	0
P4	0	0	0	0	0.41
P5	0	0	0	0.40	0

05 UNCERTAIN GRAPH WEIGHTS: Results - NFC

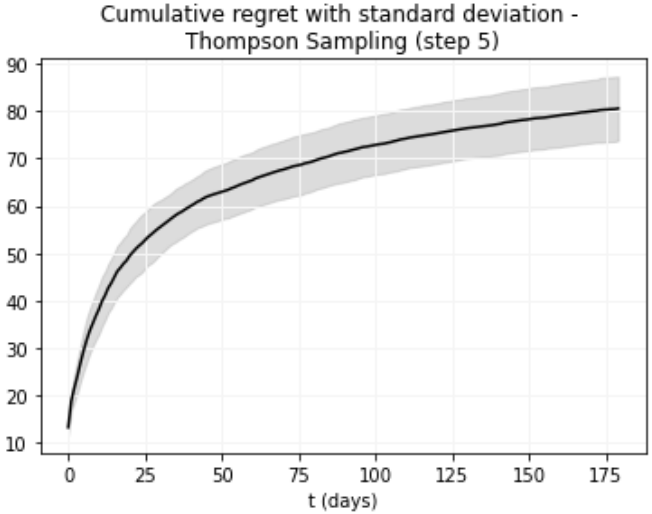
The ratio with respect to theoretical upper bound for the two algorithms are the following:

TS	UCB1
0.1337	0.0678

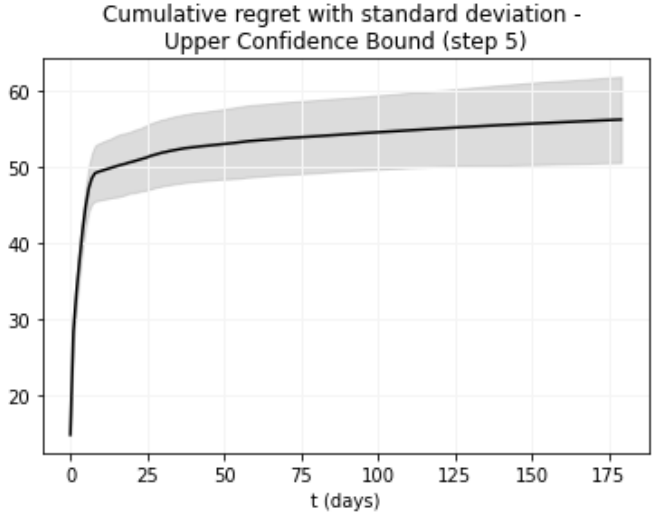
05 UNCERTAIN GRAPH WEIGHTS: Results - NFC

We also plot some other important information regarding the obtained results, starting with cumulative regret:

TS



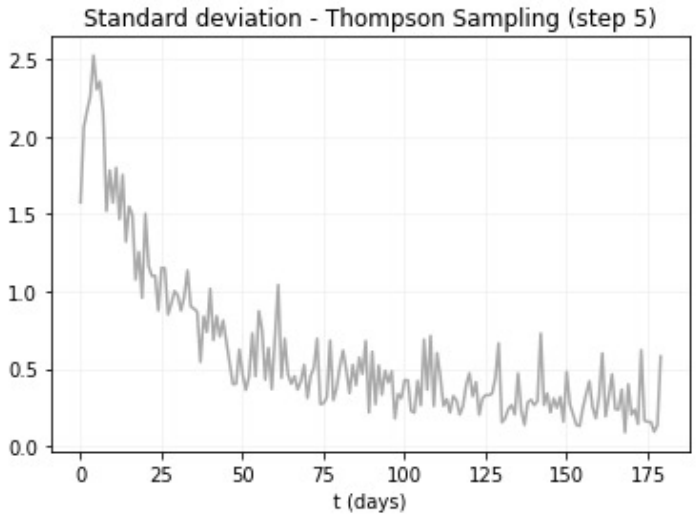
UCB1



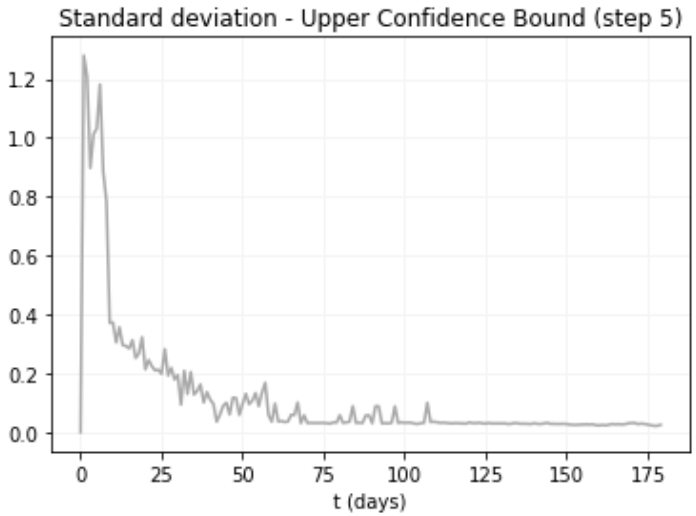
05 UNCERTAIN GRAPH WEIGHTS: Results - NFC

Standard Deviation of Cumulative Regret:

TS



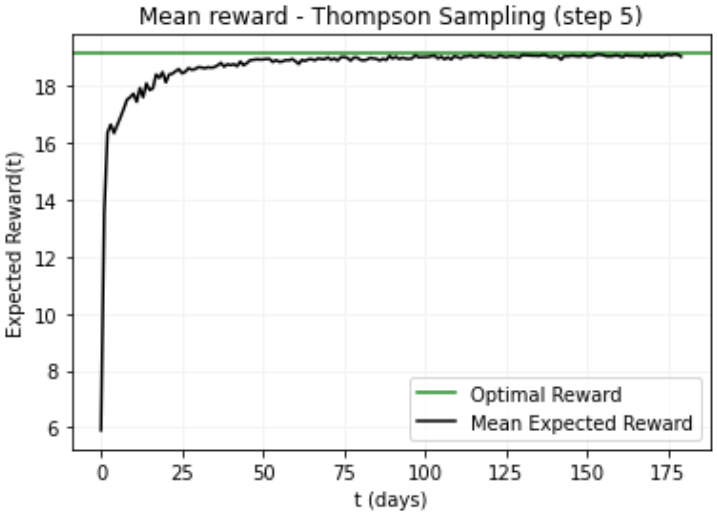
UCB1



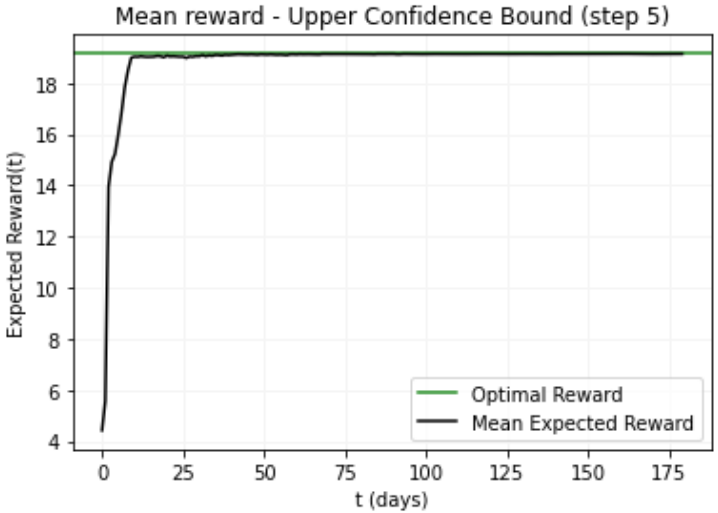
05 UNCERTAIN GRAPH WEIGHTS: Results - NFC

Comparison between optimal reward and expected one:

TS



UCB1



05 UNCERT GRAPH WEIGHT: Comparison FC vs NFC

The results obtained for the NFC case are very similar to the ones obtained in the FC case, the only slight differences stand in these main points:

- Optimal reward and Regret with both models (TS and UCB1) is decreased passing from FC to NFC case
- The ratio with respect to theoretical upper bound for the TS significantly decreases in the NFC case.
- The standard deviation is slightly lower than before and the convergence to the optimal reward is good.

06

OPTIMIZATION WITH NON-STATIONARY DEMAND CURVE

06 NON-STATIONARY DEMAND: METHODOLOGY

We evaluate the optimal prices combination in case of **non-stationary demand curve**. We choose this **abrupt change waterfall** for each user, that leads to a non-stationary demand curve:

Change in the **reservation prices' distributions**

Change in the **conversion rates**

NON-STATIONARY demand curve

Remark: in this framework the conversion rates and the graph weights matrices are uncertain, but only the conversion rates are affected by abrupt changes

06 NON-STATIONARY DEMAND: ALGORITHMS

We solve the problems using an **UCB-like algorithm** with **change detection** and a **sliding-window UCB-like algorithm**, and then we compare the results.

We assume an observation period of 180 days and an abrupt change every 60 days in order to capture a sort of seasonal change of the demand curve.

In our model the change in the demand curve arises from the change in the reservation prices' distributions (one for each product) of the aggregated user. We consider this modelling choice not unrealistic.

06 NON-STATIONARY DEMAND : SLIDING WINDOW UCB (1/5)

Theory suggests to take a sliding window which is proportional to the square root of the observation period, and we choose ***w = 45 days*** (equivalent to a proportionality constant between 3.35). We consider this value as a correct trade – off between the ability of the learner to adapt to the new condition and the possibility to have a lot of data from which the learner can learn.

The algorithm is the following:

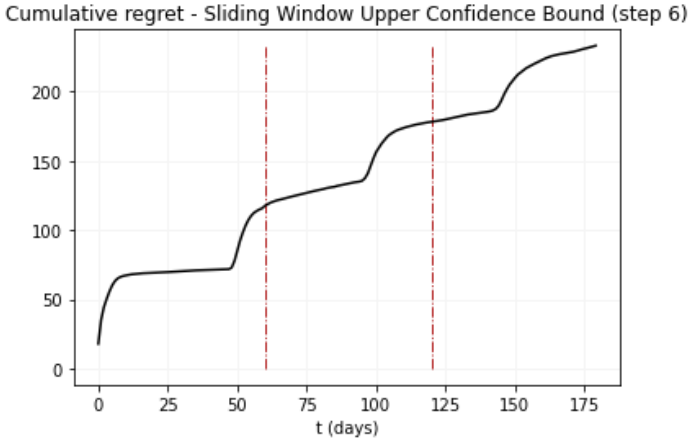
- For the first *w* days we perform the same as of step 5
- From the day *w+1* the learner updates the conversion rates only by exploiting the last *w* days

The result is that when an abrupt change arrives the learner only uses the previous *w* observations, thus neglecting part of previous data.
The optimizer computes the optimal reward associated to the new distributions of reservation prices.
This operation is performed for each abrupt change.

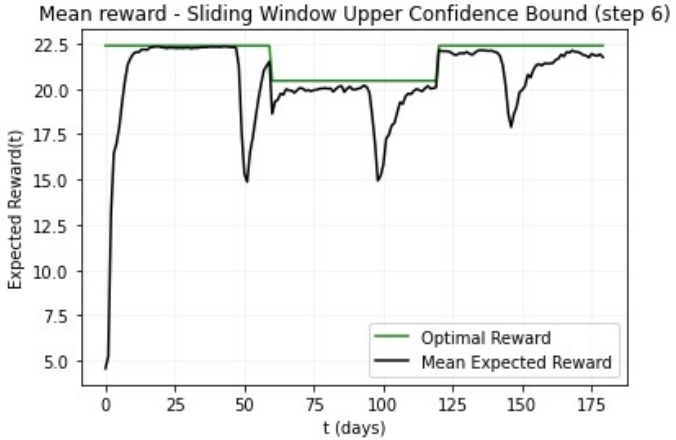
06 NON-STATIONARY DEMAND : SLIDING WINDOW UCB (2/5)

Here the results for the **FULLY CONNECTED** case:

Cumulative regret



Reward

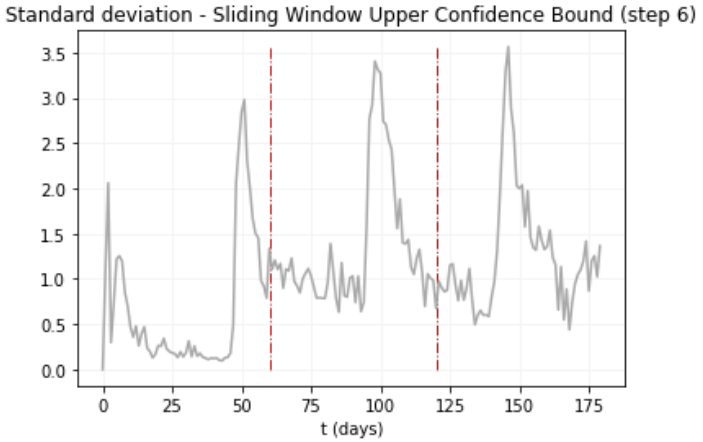


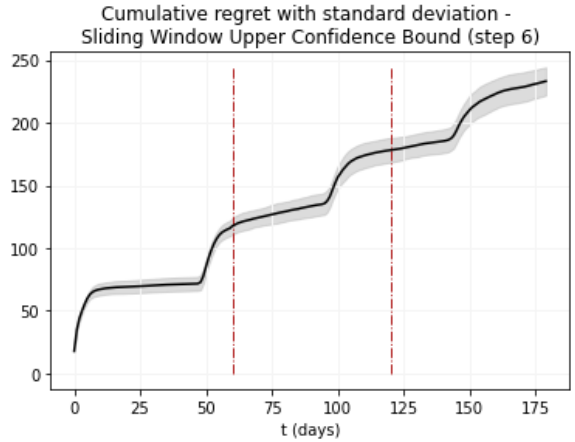
Especially from the reward plot we can observe in how many days the mean expected reward converges to the optimal one exploiting the sliding window UCB-like algorithm.

06 NON-STATIONARY DEMAND : SLIDING WINDOW UCB (3/5)

Here the results for the **FULLY CONNECTED** case:

Standard Deviation

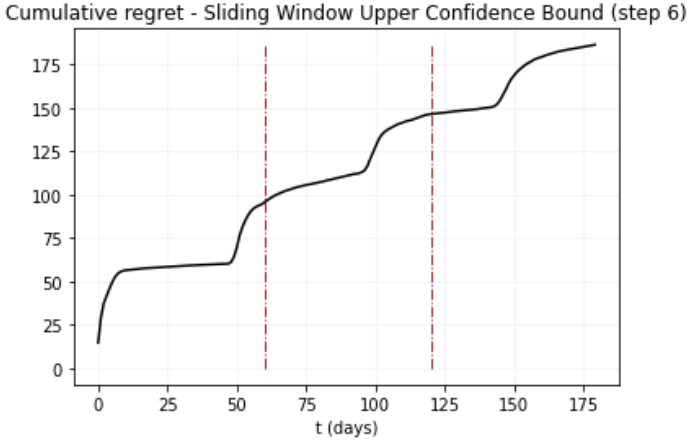




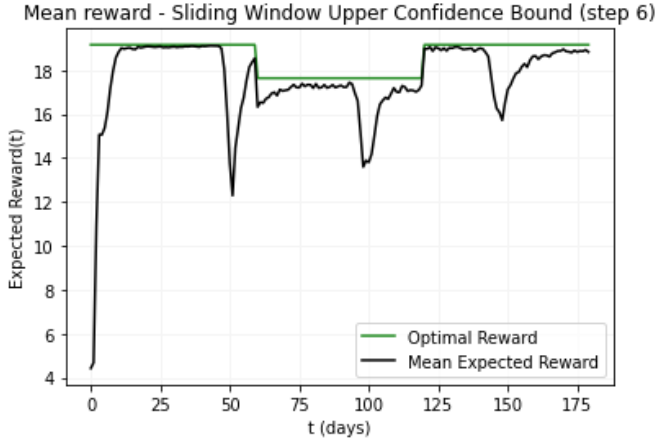
06 NON-STATIONARY DEMAND : SLIDING WINDOW UCB (4/5)

Here the results for the **NOT FULLY CONNECTED** case:

Cumulative regret



Reward

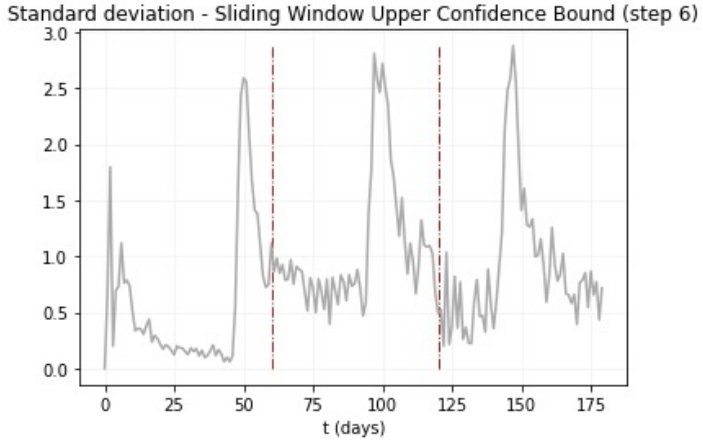


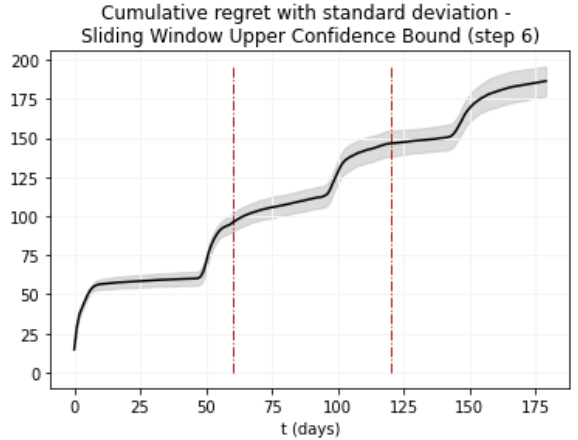
Especially from the reward plot we can observe in how many days the mean expected reward converges to the optimal one exploiting the sliding window UCB-like algorithm.

06 NON-STATIONARY DEMAND : SLIDING WINDOW UCB (5/5)

Here the results for the **NOT FULLY CONNECTED** case :

Standard Deviation





06 NON-STATIONARY DEMAND : Change Detection UCB (1/7)

During the optimization we must learn the graph weights, therefore the UCB1 algorithm needs to estimate both the conversion rates and the probability matrix.

The change detection algorithm must be run 20 times (4 prices for each product): the change in the demand curve can be caused by a change of one of the 20 conversion rates.

Therefore:

- Until no changes are detected, the UCB1 algorithm runs as usually (we update the mean and the width of the conversion rates and the graph weights in the same way of step 5)
- **When a change is detected**, historical data are discarded, and we start to record the new information and then we exploit them.

06 NON-STATIONARY DEMAND : Change Detection UCB (2/7)

Here some technicalities about the ***Change-Detection algorithm*** that we decided to use:

The **CUSUM** algorithm:

Introduction:

It is a sequential analysis technique, developed by *E. S. Page*, typically used for monitoring change detection.

The objective of the algorithm is to monitor a parameter (let us call it β) of a probability distribution (for example, the mean of a given distribution) and for determining changes in it. It is basically an automatic criterion to decide when some corrective actions need to be taken.

To be precise, when the CUSUM algorithm is applied to detect changes in the **mean**, it can **detect the steps** (discontinuities) of a given time series.

06 NON-STATIONARY DEMAND : Change Detection UCB (3/7)

CUSUM technicalities:

As the name the procedure involves the computation of a cumulative sum. This is also the reason for which this is a sequential algorithm.

- Consider a discrete process X_n
- Assign to each sample X_n a weight w
- Set a threshold T
- Normalize the process X_n by subtracting its (expected) mean and dividing by its (known) standard deviation and obtain the process Z_n
- To detect the anomalies (both positive and negative) initialize two sums equal to zero: $S_{H_0} = 0$ and $S_{L_0} = 0$
- To detect a positive anomaly, compute $S_{H_{n+1}} = max(0, S_{H_n} + Z_{n+1} - w)$
- To detect a negative anomaly, compute $S_{L_{n+1}} = max(0, S_{L_n} - Z_{n+1} - w)$
- If one of these sums exceeds the threshold T, we have **detected a change in the mean of the process**.

The hyperparameters T and w determine the sensitivity of the algorithm, for example larger w makes the CUSUM less sensitive to the change and viceversa.

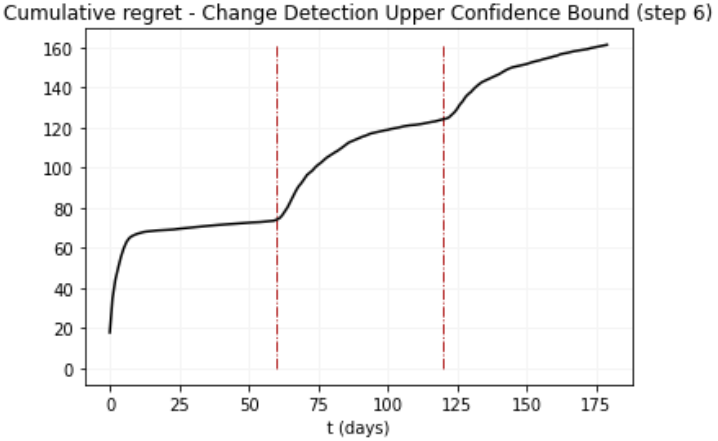
CUSUM – our application:

The " Z – process " that we monitor is the difference between the estimated conversion rate and the realized conversion rate and we look for when this process' realization significantly differs from 0.

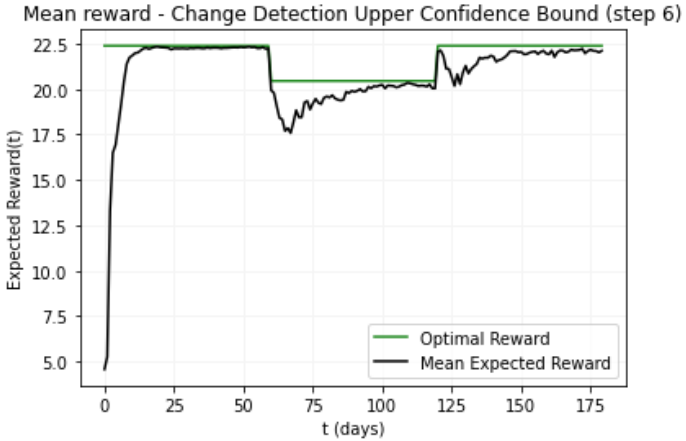
06 NON-STATIONARY DEMAND : Change Detection UCB (4/7)

Here the results for the **FULLY CONNECTED** case:

Cumulative regret



Reward

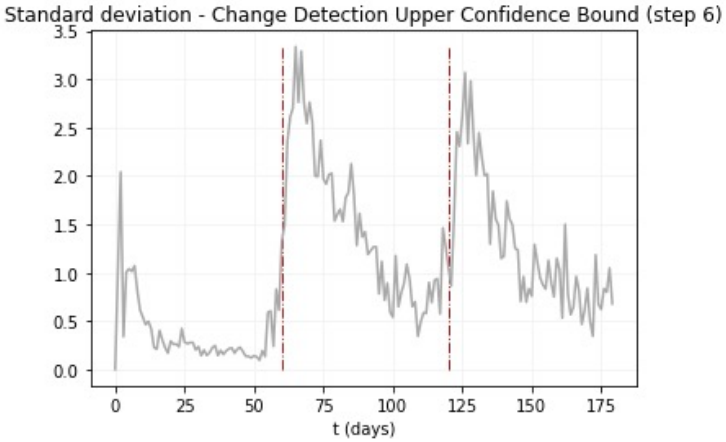


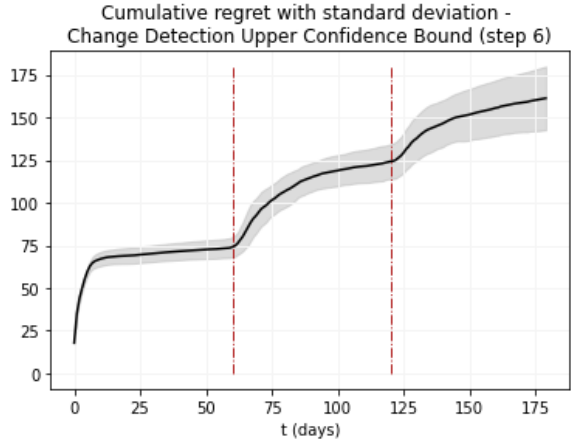
Especially from the reward plot we can observe in how many days the mean expected reward converges to the optimal one exploiting the change detection UCB-like algorithm.

06 NON-STATIONARY DEMAND : Change Detection UCB (5/7)

Here the results for the **FULLY CONNECTED** case:

Standard Deviation

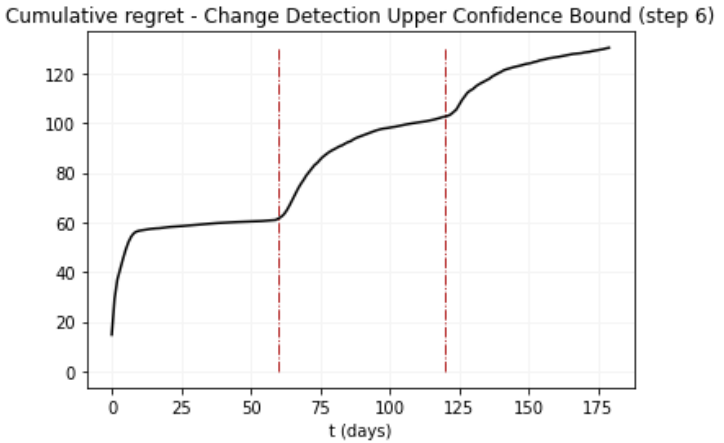




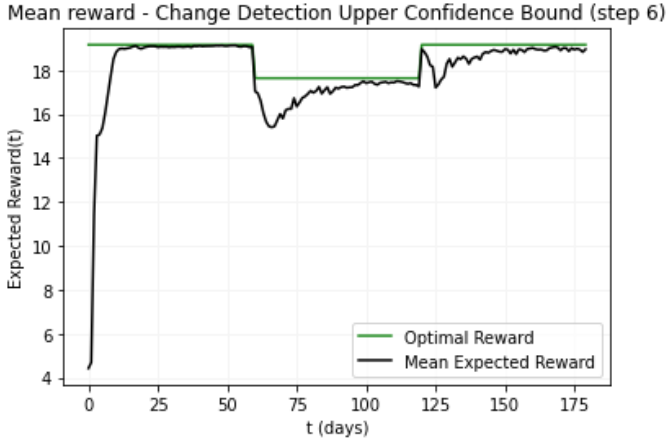
06 NON-STATIONARY DEMAND : Change Detection UCB (6/7)

Here the results for the **NOT FULLY CONNECTED** case:

Cumulative regret



Reward

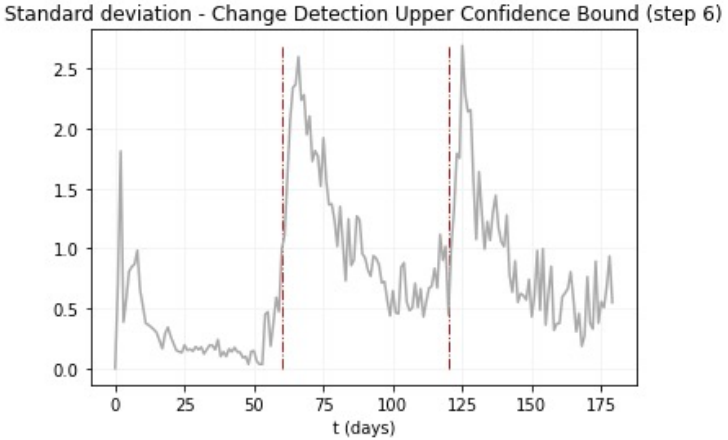


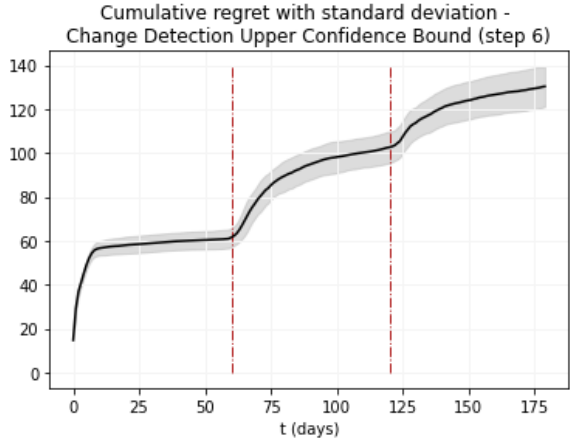
Especially from the reward plot we can observe in how many days the mean expected reward converges to the optimal one exploiting the change detection UCB-like algorithm.

06 NON-STATIONARY DEMAND : Change Detection UCB (7/7)

Here the results for the **NOT FULLY CONNECTED** case:

Standard Deviation



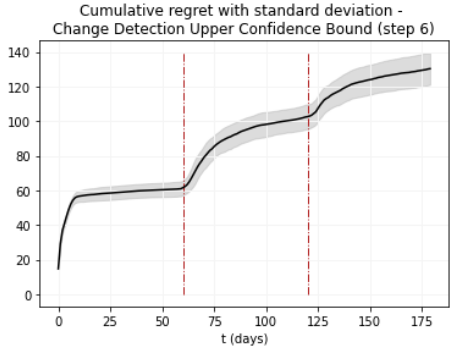


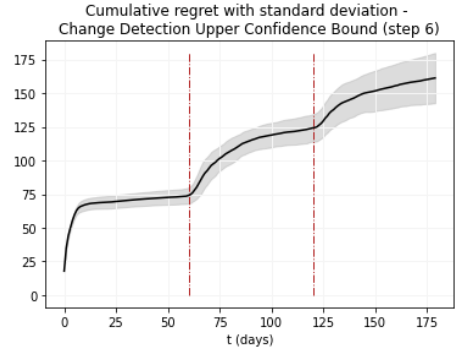
06 NON-STATIONARY DEMAND : SW UCB and CD UCB comparison

The **change detection** algorithm works **better** than the sliding window approach.

Indeed, in the SW approach we notice some downwards in correspondence of time=45 and its multiples (correspondent to the switch of the windows exploited), probably since the algorithm in that moment neglects some data related to the first exploration.

The CD takes less time to converge. Indeed, the cumulative regret of CD is dominated by the cumulative regret of SW approach.





06 NON-STATIONARY DEMAND : POSSIBLE IMPROVEMENT

In both *Sliding Window* and *Change Detection – CUSUM algorithms* should require hyperparameter tuning (width of the window and threshold to detect a change), but we did not perform this kind of tuning.

In fact, we are convinced that in real world scenario the primary objective is the regret minimization, and it is unfeasible to perform an hyperparameter tuning through a grid – search.

07

CONTEXT GENERATION

07 CONTEXT GENERATION : Possible Improvement

The first thing is to understand whether there could be some improvements arising from the use of context generation, that is to **perform price discrimination between different categories of users**.
In order to do that it must be considered the performance of a clairvoyant algorithm in terms of **expected reward** and **convergence time**:

- **Few data available** → clairvoyant algorithm on **aggregate user;**

- **Many data available** → clairvoyant algorithm on **disaggregate user;**

Thus the best thing is to start using an aggregate model and then switch to a disaggregate one once enough data are collected (**combination of the two approaches**)

07 CONTEXT GENERATION : Possible Improvement

In our problem, contexts can be created splitting upon 2 features: **age** and **cooking experience** of the potential user.
The biggest possible space of features is represented with a table:

	YOUNG	OLD
HOME COOKS		
EXPERIENCED CHEF		

For every possible partition of the space of features we evaluate whether partitioning is better than not doing that.
Since partitioning makes sense only when a **strong evidence** can be provided, in the comparison it is better not to use mean values, but the lower confidence bound.

07 NON-STATIONARY DEMAND : Disaggregated Users

Improvement

No more one single demand curve → **One demand curve** for each **cluster** of **users**

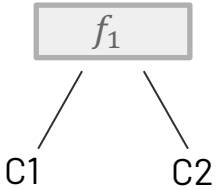
Probability associated to each **category** → A new user has a predetermined probability of being part of the following classes according to the probability matrix on the left;

	YOUNG p = 0.7	OLD p = 0.3
HOME COOKS p = 0.65	 p = 0.455	 p =0.195
EXPERIENCED CHEF p = 0.35	 p = 0.245	 p = 0.105

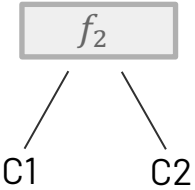
07 NON-STATIONARY DEMAND : Our Procedure's Steps

- Start as usual with the **aggregate user** (consider all the categories together with the initial distribution of prices);
- Continue with the computation of expected reward, splitting according to the already mentioned categories. Thus, it is performed the computation of:
  - Lower bound of the expected reward considering users only divided by **age**;
  - Lower bound of the expected reward considering users only divided by **experience**;

- Those values are **compared** with the expected reward of the aggregated user. If a split is possible on both features, the one with **higher value** is selected first.



$$\underline{p}_{c_1}\underline{\mu}_{a^*_{c_1},c_1} + \underline{p}_{c_2}\underline{\mu}_{a^*_{c_2},c_2} \geq \underline{\mu}_{a^*_{c_0},c_0}$$



$$\underline{p}_{c_1}\underline{\mu}_{a^*_{c_1},c_1} + \underline{p}_{c_2}\underline{\mu}_{a^*_{c_2},c_2} \geq \underline{\mu}_{a^*_{c_0},c_0}$$

07 NON-STATIONARY DEMAND : Our Procedure's Steps

AGE

f_1

C1 C2

$p_{c1}\mu_{a^*_{c1},c1} + p_{c2}\mu_{a^*_{c2},c2} \geq \mu_{a^*_{c0},c0}$

- f_1 := AGE
- C1:= YOUNG
- C2:= OLD

EXPERIENCE

f_2

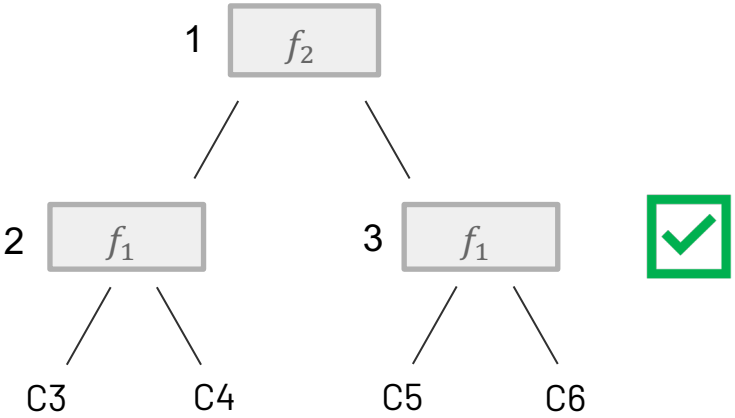
C1 C2

$p_{c1}\mu_{a^*_{c1},c1} + p_{c2}\mu_{a^*_{c2},c2} \geq \mu_{a^*_{c0},c0}$

- f_2 := EXPERIENCE
- C1:= EXPERIENCED
- C2:= UNEXEPERIENCED

Experience is selected at first stage based on the expected reward criterion

07 NON-STATIONARY DEMAND : Our Procedure Steps



$p_{c3}\mu_{a^*_{c3},c3}+p_{c4}\mu_{a^*_{c4},c4} \geq \mu_{a^*_{c1},c1}$

$p_{c5}\mu_{a^*_{c5},c5}+p_{c6}\mu_{a^*_{c6},c6} \geq \mu_{a^*_{c2},c}$

- C3 := YOUNG UNEXPERIENCED
- C4 := OLD UNEXPERIENCED
- C5 := YOUNG EXPERIENCED
- C6 := OLD EXPERIENCED

Age splitting satisfies the bound check only for unexperienced users (node 2);

07 NON-STATIONARY DEMAND : Result

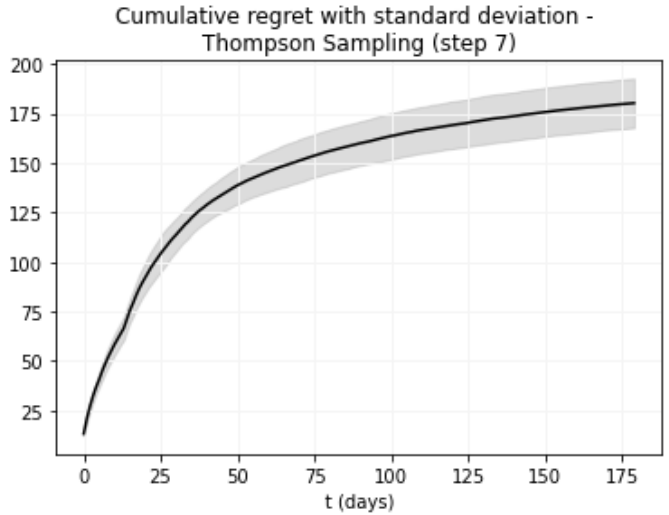
Thus, in the end our feature selection is the following one:

	YOUNG	OLD
HOME COOKS		
EXPERIENCED CHEF		

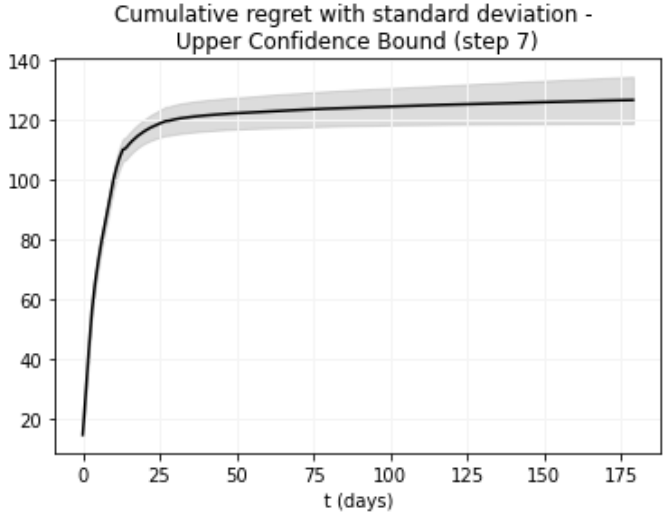
07 NON-STATIONARY DEMAND : Results- FC

We plot some other important informations regarding the obtained results, starting with cumulative regret:

TS



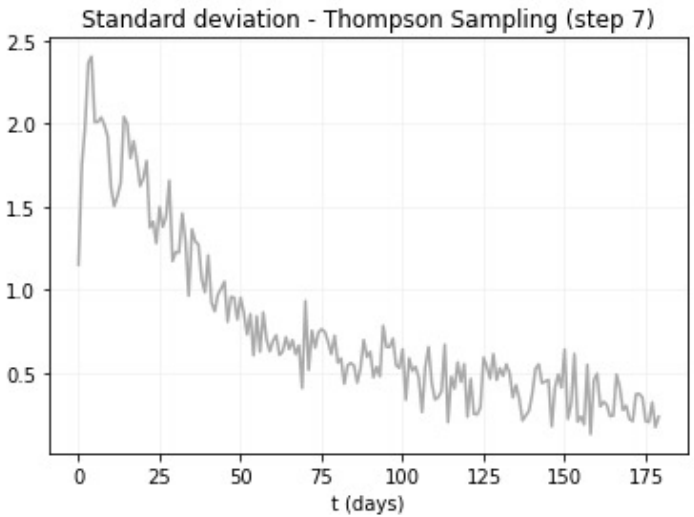
UCB1



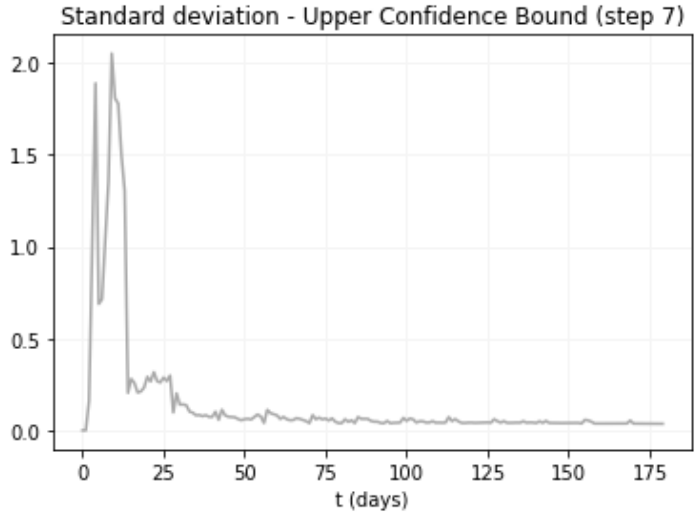
07 NON-STATIONARY DEMAND : Results - FC

Standard deviation of Cumulative Regret:

TS



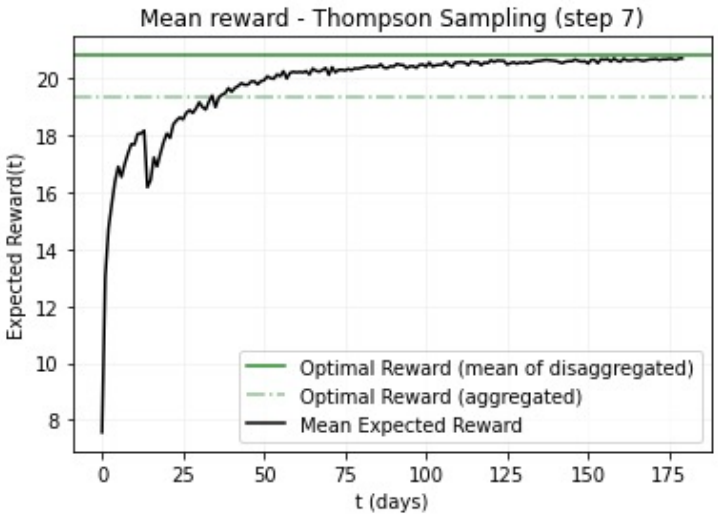
UCB1



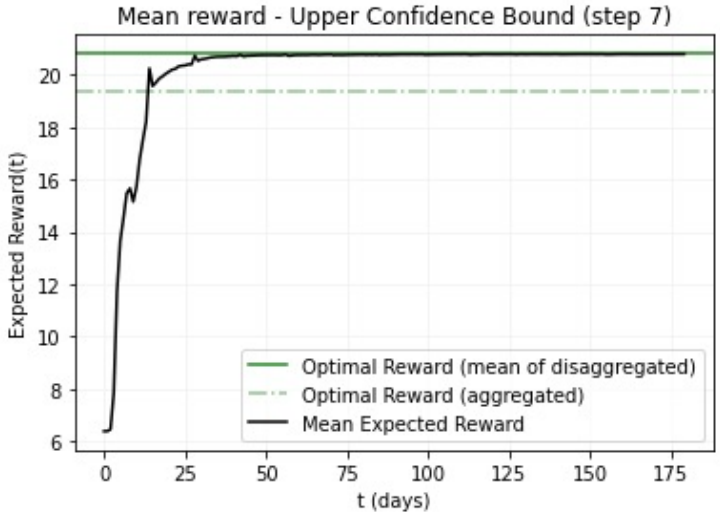
07 NON-STATIONARY DEMAND : Results - FC

Comparison between optimal reward and expected one:

TS



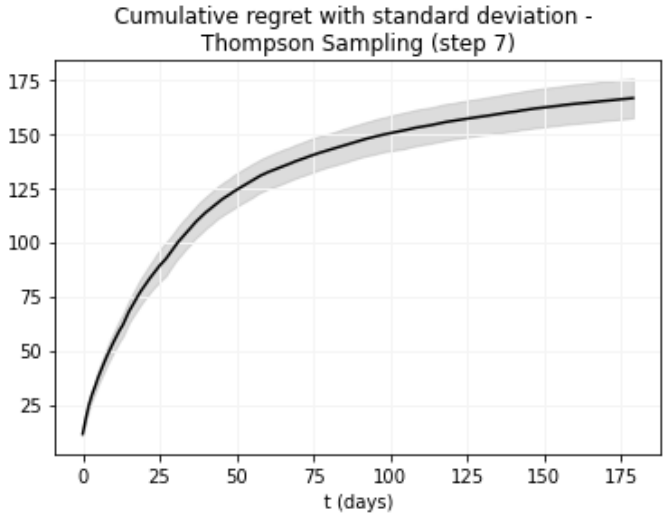
UCB1



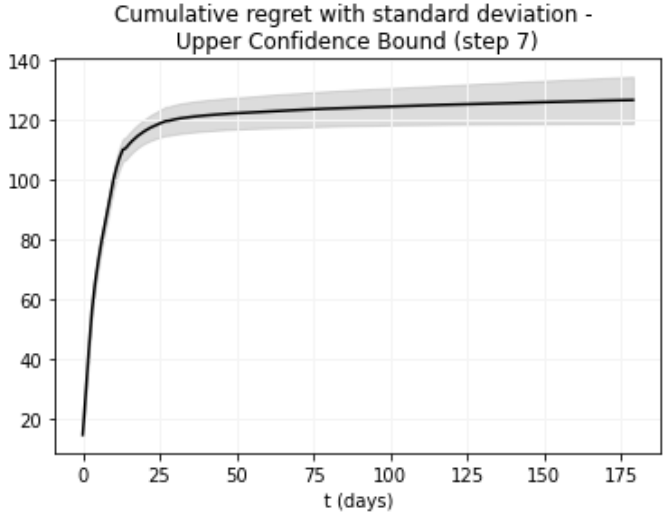
07 NON-STATIONARY DEMAND : Results- NFC

We also plot some other important informations regarding the obtained results, starting with cumulative regret:

TS



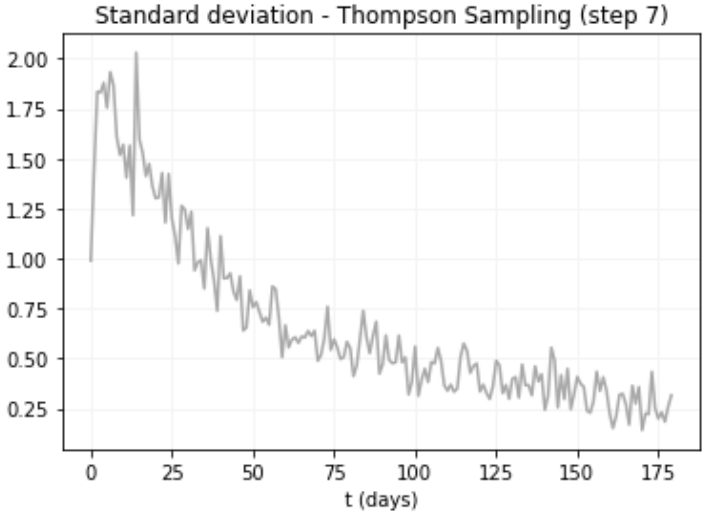
UCB1



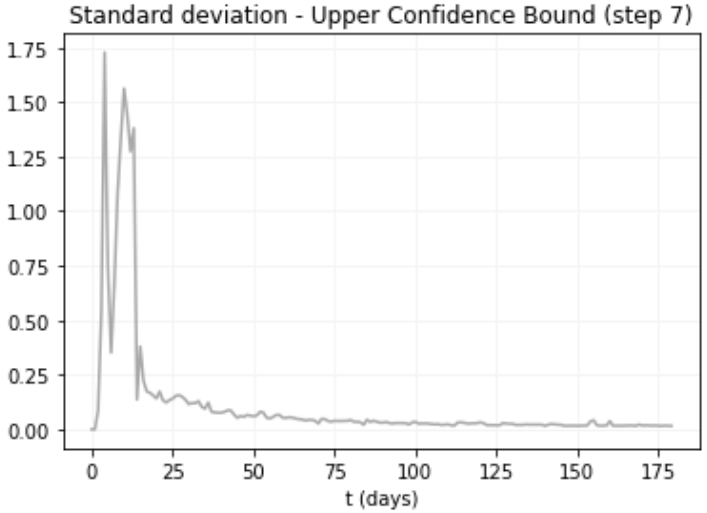
07 NON-STATIONARY DEMAND : Results - NFC

Standard deviation of Cumulative Regret:

TS



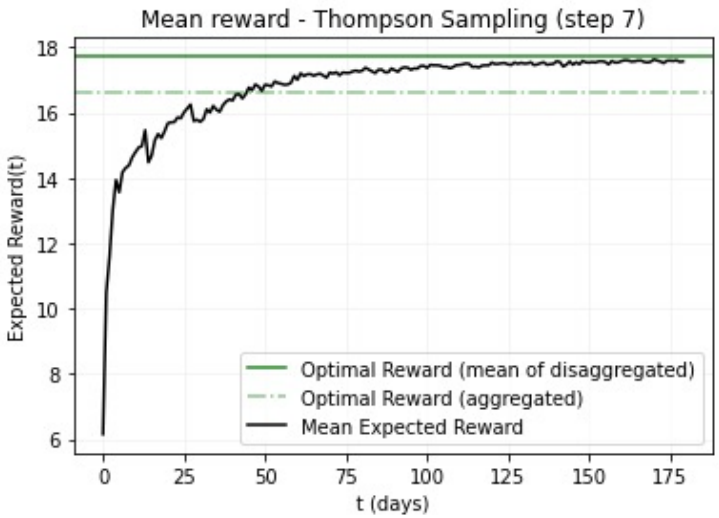
UCB1



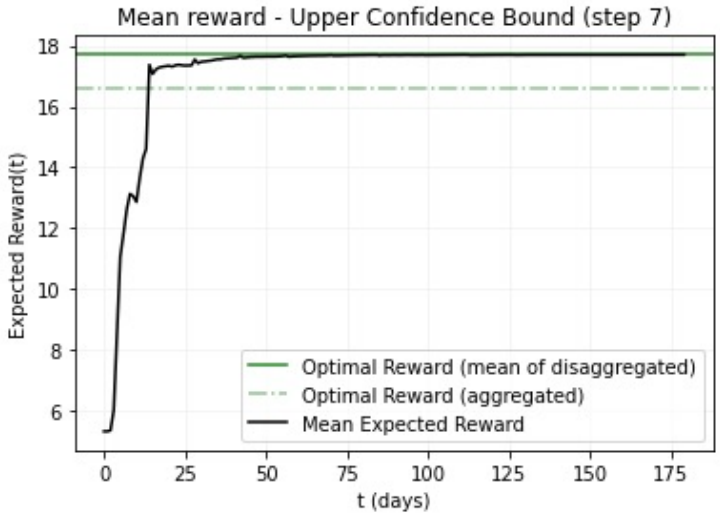
07 NON-STATIONARY DEMAND : Results - NFC

Comparison between optimal reward and expected one:

TS



UCB1



THANK YOU FOR YOUR ATTENTION