Team 10
Pricing and Advertising

Daglio Gabriele, Di Cesare Federico, Germano Jacopo
23th July 2021

Environment

The users

Features: F_1 and F_2, independent.

$$\Pr(F_1 = True) = \theta_1$$
$$\Pr(F_2 = True) = \theta_2.$$

Likelihoods:

$$\tilde{l}_{TT} = \theta_1\theta_2$$
$$\tilde{l}_{TF} = \theta_1(1 - \theta_2)$$
$$\tilde{l}_{FT} = (1 - \theta_1)\theta_2$$
$$\tilde{l}_{FF} = (1 - \theta_1)(1 - \theta_2)$$

Example

C3
l=0.15
C1
l=0.24
C2
l=0.23
C1
l=0.38
True
False
Feature 2
False
True
Feature 1

One round

λ_a = Average number of daily auctions.
$v(b)$ = Probability of winning an auction with bid b
$r_{comb}(p)$ = Conversion rate of *comb* users for price p
f_{comb} = Average future visits of *comb* users

Each day:

1. $A \sim Poisson(\lambda_a)$ # number of auctions
2. $A_{TT}, A_{TF}, A_{FT}, A_{FF} \sim Multinomial(A, (\tilde{l}_{TT}, \tilde{l}_{TF}, \tilde{l}_{FT}, \tilde{l}_{FF}))$
3. $N_{comb} \sim Binomial(A_{comb}, v(b))$ # the new clicks
4. $C_{comb,j} \sim CostPerClick(k_{comb}(b))$ # cost of click j
5. $D_{comb,i} \sim Binomial(N_{comb}, r_{comb}(p))$ # whether user i purchases
6. $F_{comb,i} \sim Poisson(f_{comb})$ # future visits user i

Step 1

Offline algorithm

Without context generation

Expected profit

$$ExpectedProfit(p, b) = E\left[\sum_{c \in C} \sum_{i=1}^{N_{c,b}} \Big(D_{c,p,i}(1 + F_{c,i})m(p) - C_{c,b,i} \Big)\right]$$

$$\Downarrow$$

$$ExpectedProfit(p, b) = \lambda_a v(b) \sum_{c \in C} l_c \Big(r(c, p)(1 + f(c))m(p) - k(c, b) \Big)$$

Independent Price Hierarchy

$$p_1 \succeq p_2 \quad \text{for some bid}$$

$$\Downarrow$$

$$p_1 \succeq p_2 \quad \text{for every bid}$$

The algorithm

1. $\bar{b} \leftarrow median(B)$

2. $p^* \leftarrow optimal\_price\_fixed\_bid(\bar{b})$ # linear scan of the prices

3. $b^* \leftarrow optimal\_bid\_fixed\_price(p^*)$ # linear scan of the bids

4. $return\ (p^*, b^*)$

Step 2

Online problem formalization

Online setting

- The averages are not know
- The feedback is partially delayed
  - The future visits come with a delay of 30 rounds.

Online setting - Feedback

- After round i, the learner receives:
  - The auctions run on round i
  - The new clicks obtained on round i
  - The purchases perfomed on round i
  - The cost paid for the clicks of round i
  - The future visits of the purchases of round i - 30

Online setting - Estimates

- The averages are not known
- Need to estimate:
  - Auctions --> Sample mean
  - New clicks --> Bernoulli variable, auction won (1) or lost (0)
  - Purchases --> Bernoulli variable, click converted (1) or not (0)
  - Cost per click --> Sample mean
  - Future visits --> Sample mean of complete samples

Estimate the profit

$$\widehat{exp(p,b)}_i = \overline{a}_i \overline{w_{bi}} \left(m(p) \overline{r_{p_i}} (1 + \overline{f}_{p_i}) - \overline{c}_b \right)$$

Optimistically explore w and r

Optimistic exploration

Two approaches:

Upper Confidence Bound

- Conversion rate --> Conversion rate upper bound
- Bid win rate --> Bid win rate upper bound

Thompson Sampling

- Conversion rate --> Sample from Betas, update the Betas
- Bid win rate --> Sample from Betas, update the Betas

Step 3

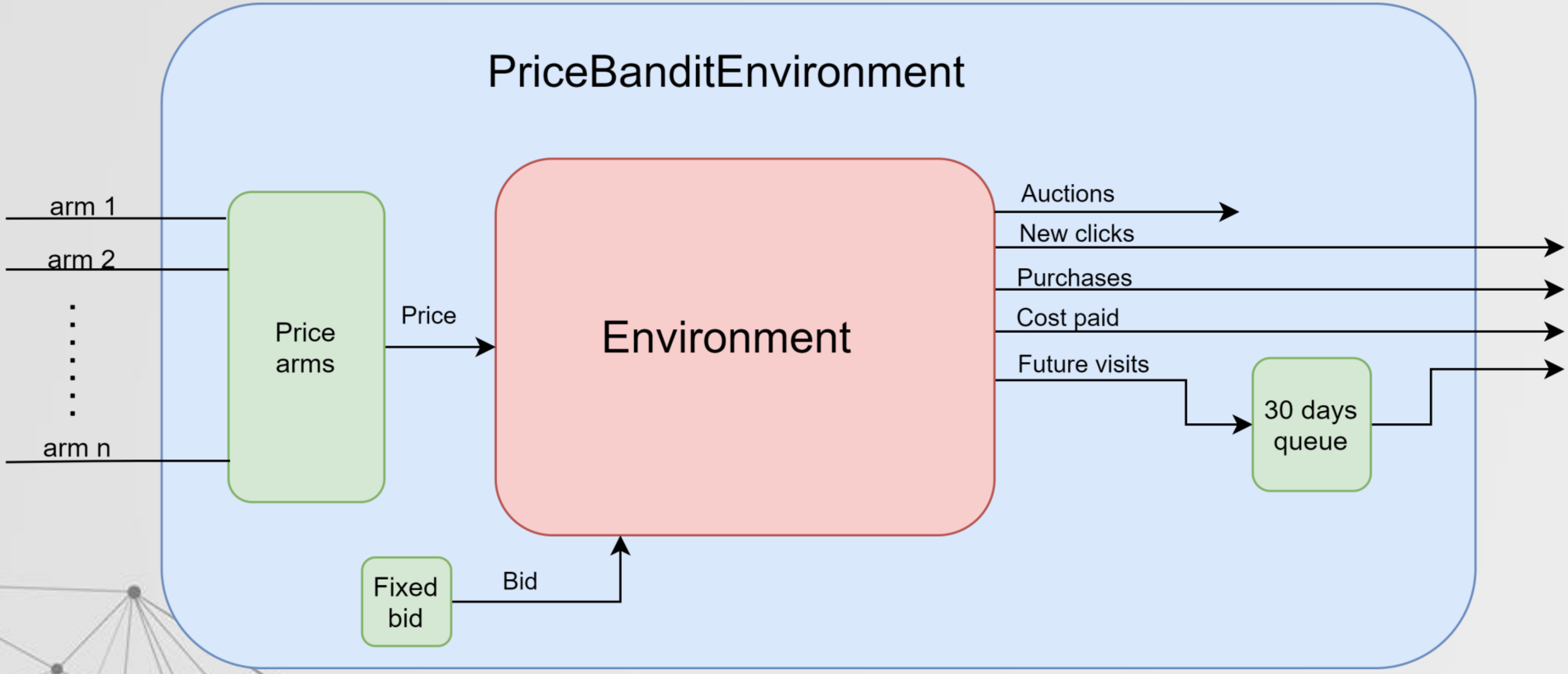
Online pricing

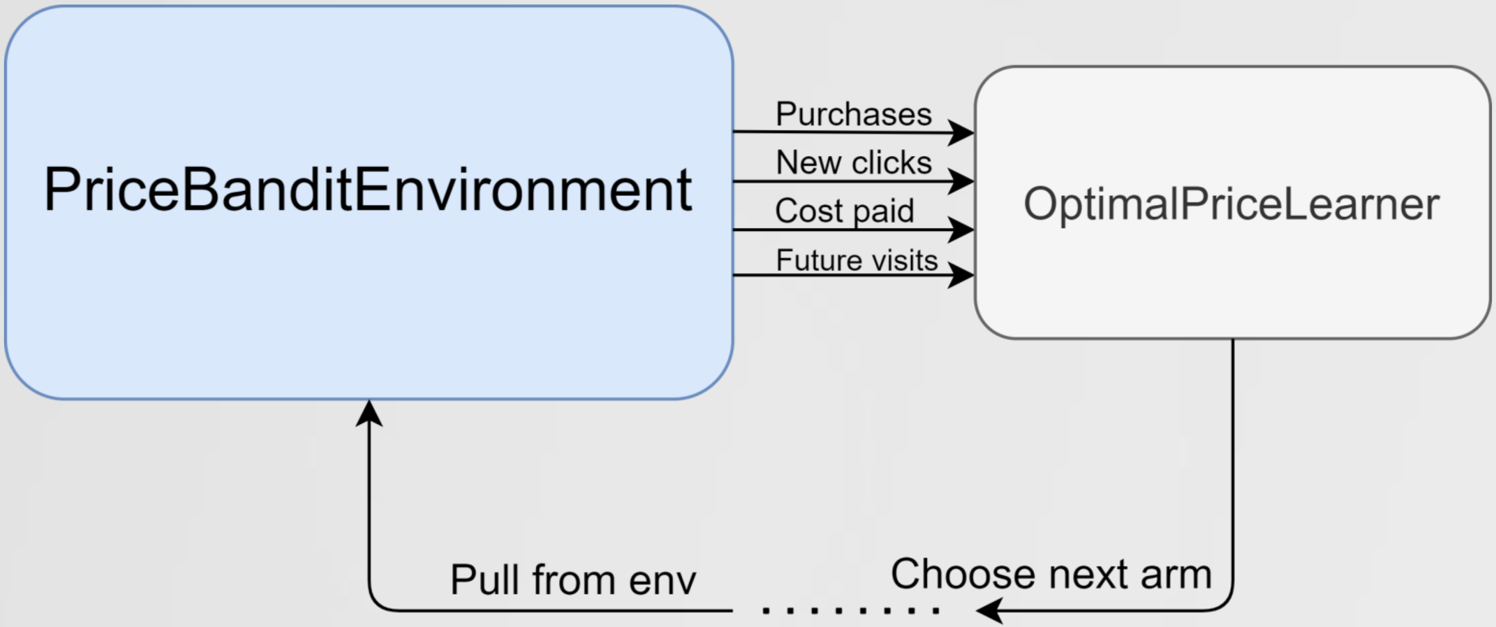
Without discriminating between customer classes

OUR NUMBERS

10 PRICES

1 BID

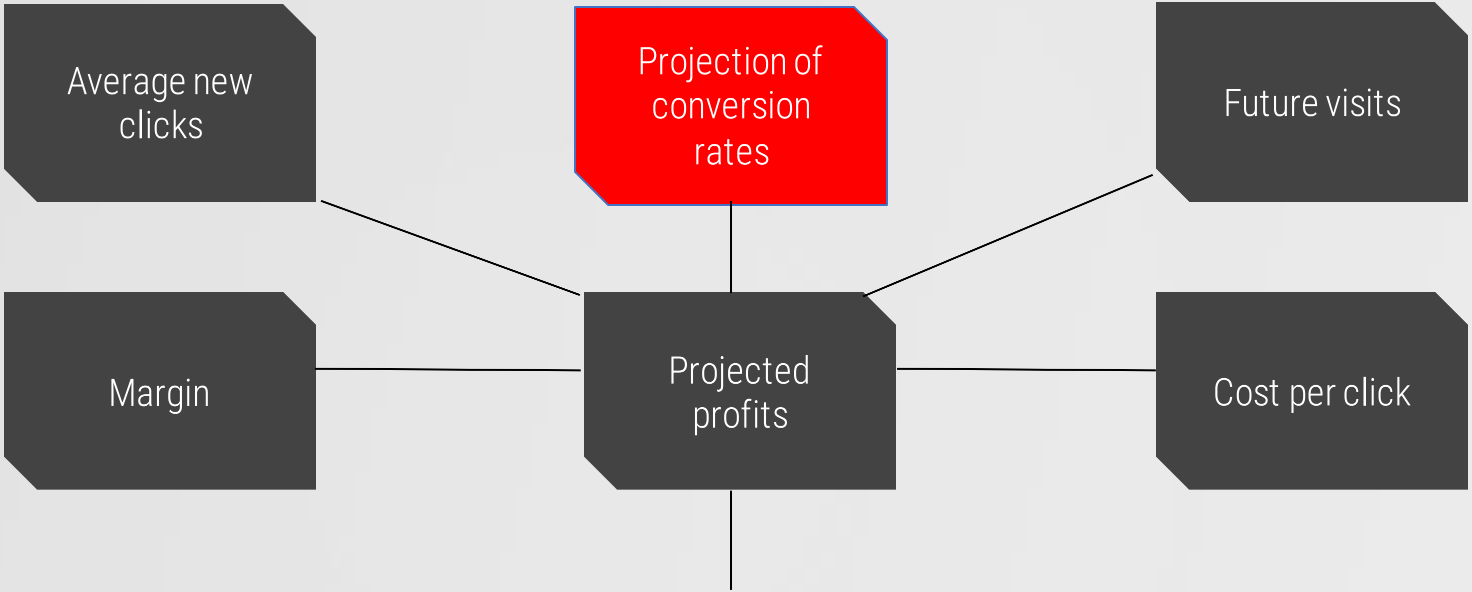

PriceBanditEnvironment
arm 1
arm 2
arm n
Price
arms
Price
Environment
Auctions
New clicks
Purchases
Cost paid
Future visits
30 days
queue
Fixed
bid
Bid


PriceBanditEnvironment
Purchases
New clicks
Cost paid
Future visits
OptimalPriceLearner
Pull from env
Choose next arm

INITIAL ROUND-ROBIN

- Round robin for 30 + *number of arms* rounds
- Until 1 full feedback for each arm

CHOICE OF THE NEXT ARM



NEXT ARM <- ARGMAX

PROJECTION OF CONVERSION RATES

OPTIMAL PRICE LEARNER

UCB OPTIMAL PRICE LEARNER

TS OPTIMAL PRICE LEARNER

UCB OPTIMAL PRICE LEARNER

Conversion rates
<u>AVERAGES</u>

+

Conversione rates
<u>RADII</u>

Conversion rates
<u>UPPER BOUNDS</u>

UCB OPTIMAL PRICE LEARNER

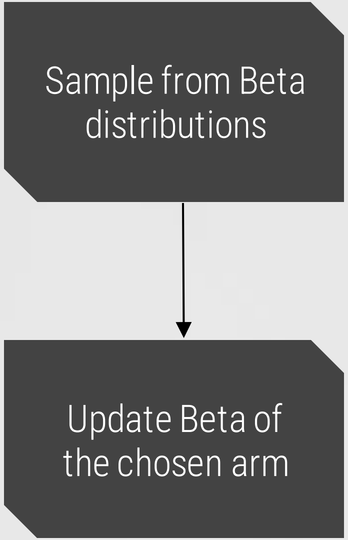
$$\frac{purchases_a}{clicks_a} + \sqrt{\frac{2log(t)}{clicks_a}}$$

AVERAGES

RADII

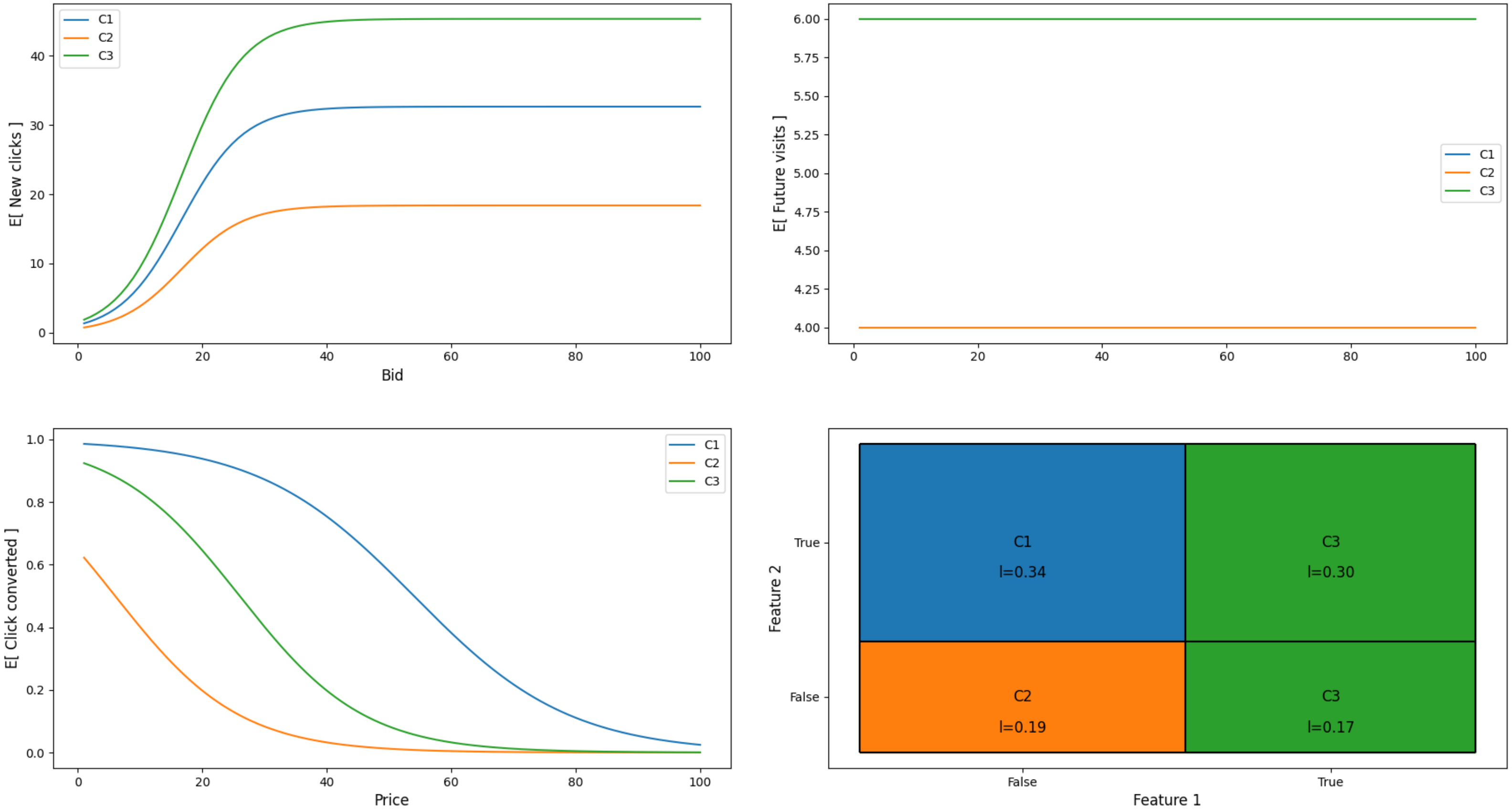
The updates happen in ***batches*** since, at each round, we see the realization of ***new_clicks*** tries

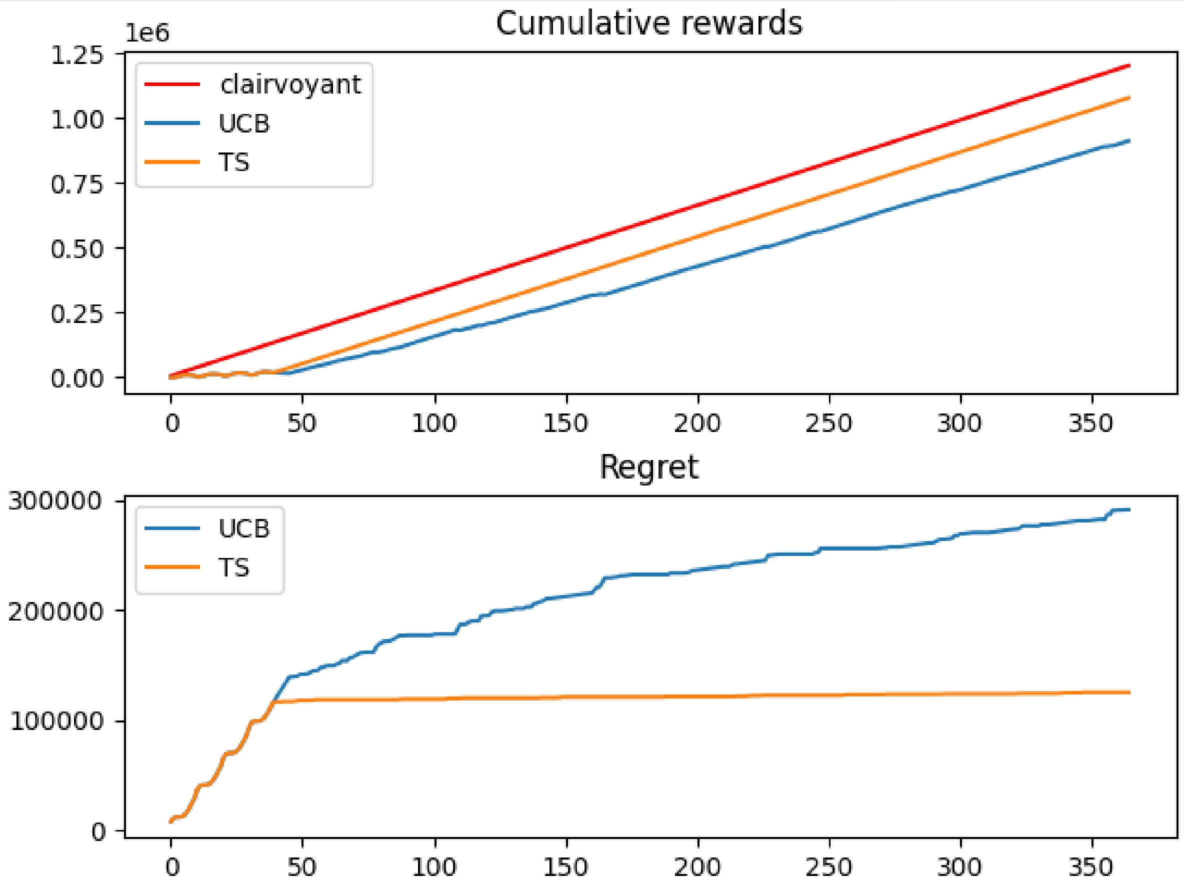
TS OPTIMAL PRICE LEARNER



The updates of the parameters α and β of the Beta distributions happen in ***batches*** since, at each round, we see the realization of ***new_clicks*** tries

Seed: 3144548588

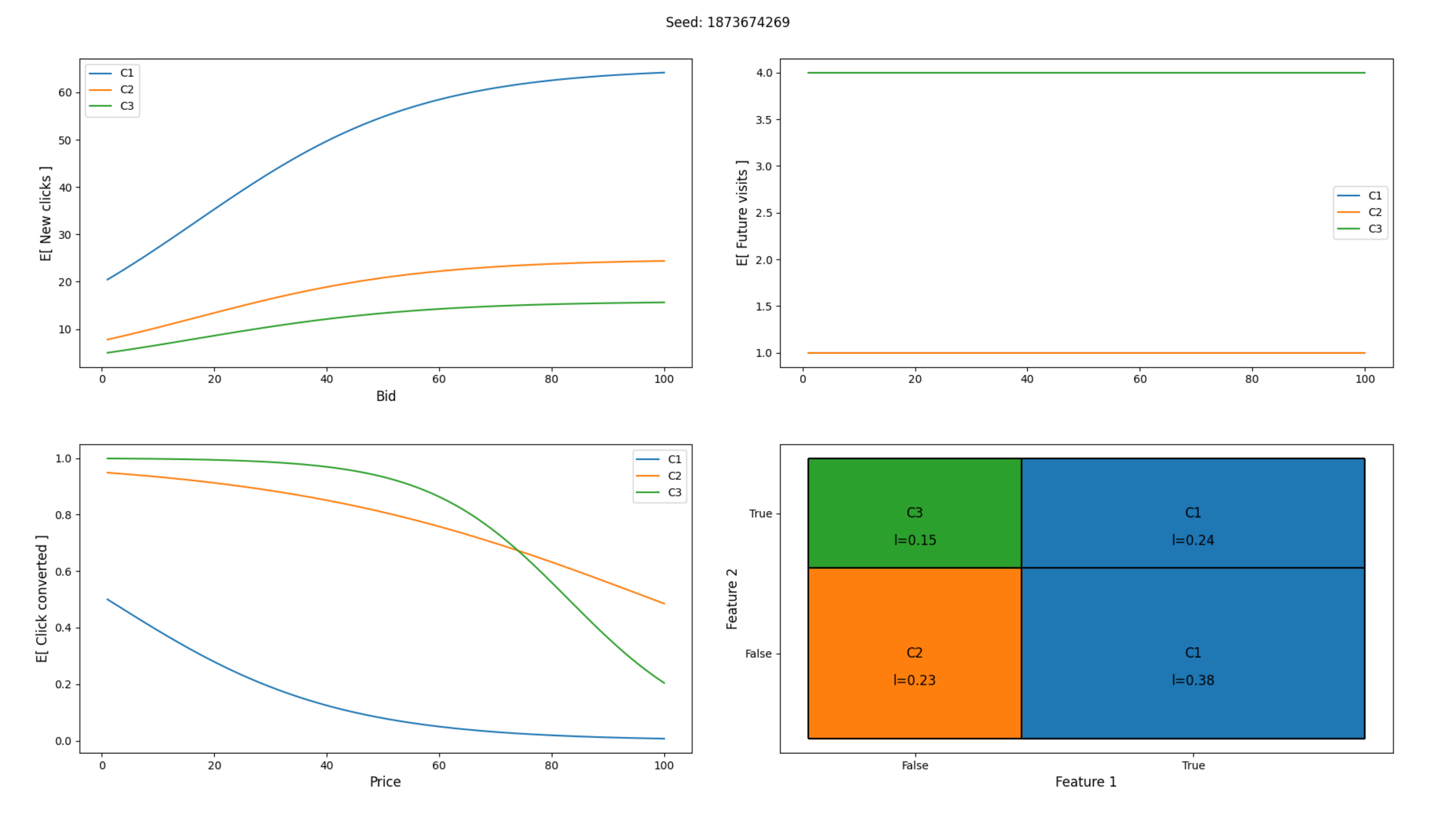



Cumulative rewards
1e6
1.25
1.00
0.75
0.50
0.25
0.00
0
50
100
150
200
250
300
350
clairvoyant
UCB
TS
Regret
300000
200000
100000
0
0
50
100
150
200
250
300
350
UCB
TS

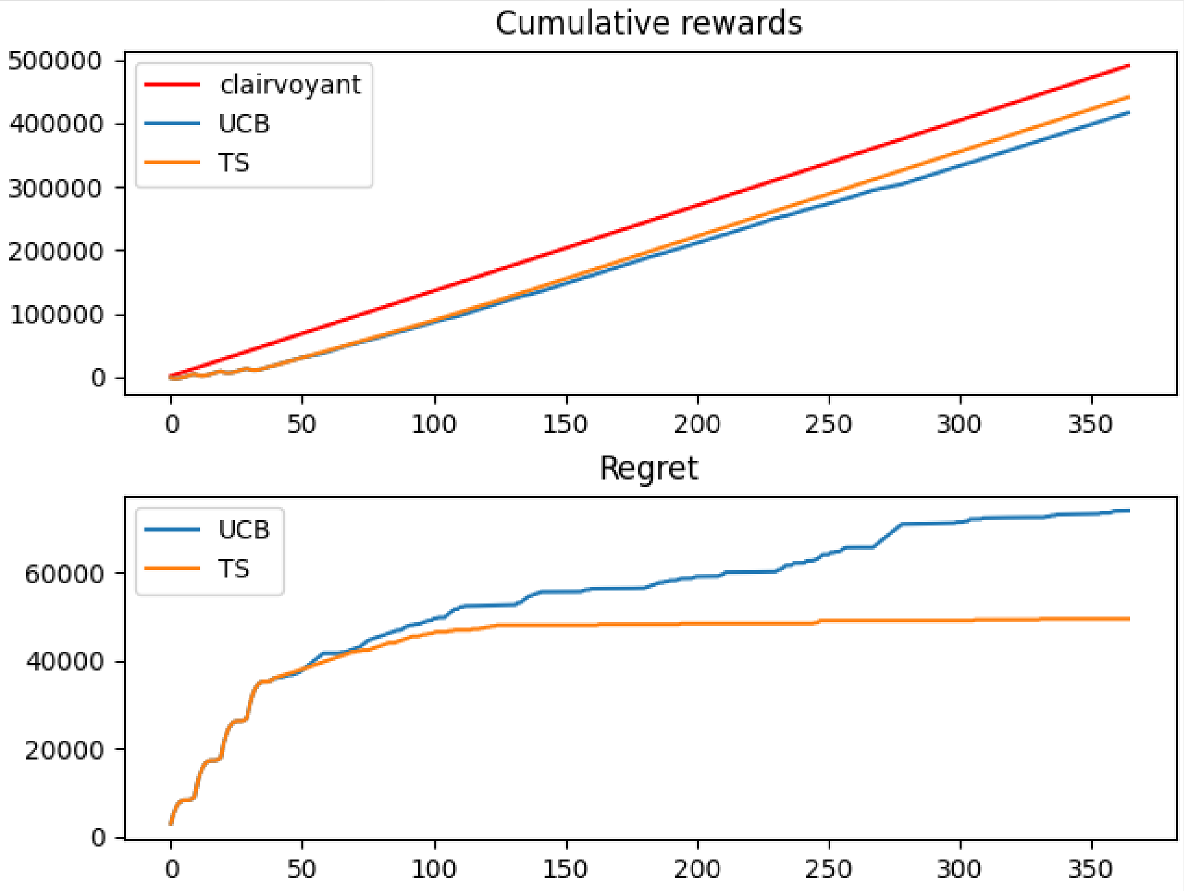
Seed: 3144548588

Price	Expected	Norm. gaps	UCB Pulls	UCB Expected	TS Pulls	TS Expected
10.00	-4309.67	100.0	4	-4287.03	4	-4447.41
20.00	-48.64	43.9	4	16.98	4	28.71
30.00	2460.29	10.9	10	2650.07	4	2066.38
40.00	3291.72	0.0	141	3266.95	302	3344.78
50.00	2977.05	4.1	131	3045.23	31	3016.02
60.00	1924.81	18.0	34	2020.80	4	1859.43
70.00	632.38	35.0	13	218.62	4	-180.93
80.00	-440.08	49.1	13	-435.83	4	-165.58
90.00	-1144.79	58.4	11	-1236.55	4	-1492.98
100.00	-1550.37	63.7	4	-1806.97	4	-1715.24

Optimal price: 40.00, Optimal bid: 27.00



Seed: 1873674269
E[New clicks]
Bid
E[Future visits]
E[Click converted]
Price
C1
C2
C3
Feature 2
Feature 1
True
False
C3
l=0.15
C1
l=0.24
C2
l=0.23
C1
l=0.38


Cumulative rewards
clairvoyant
UCB
TS
500000
400000
300000
200000
100000
0
0
50
100
150
200
250
300
350
Regret
UCB
TS
60000
40000
20000
0
0
50
100
150
200
250
300
350

Seed: 1873674269

Price	Expected	Norm. gaps	UCB Pulls	UCB Expected	TS Pulls	TS Expected
10.00	-1564.32	100.0	4	-1677.11	4	-1704.78
20.00	-806.12	73.9	4	-775.53	4	-778.56
30.00	-172.82	52.1	4	-200.15	4	-153.46
40.00	361.10	33.8	4	264.64	4	414.62
50.00	808.49	18.4	5	765.44	4	993.70
60.00	1152.60	6.5	14	1044.73	27	1177.04
70.00	1342.60	0.0	95	1351.48	261	1351.32
80.00	1327.18	0.5	122	1336.60	7	825.18
90.00	1131.00	7.3	65	1138.87	45	1097.34
100.00	865.62	16.4	48	967.43	5	666.85

Optimal price: 70.00, Optimal bid: 9.00

Step 4

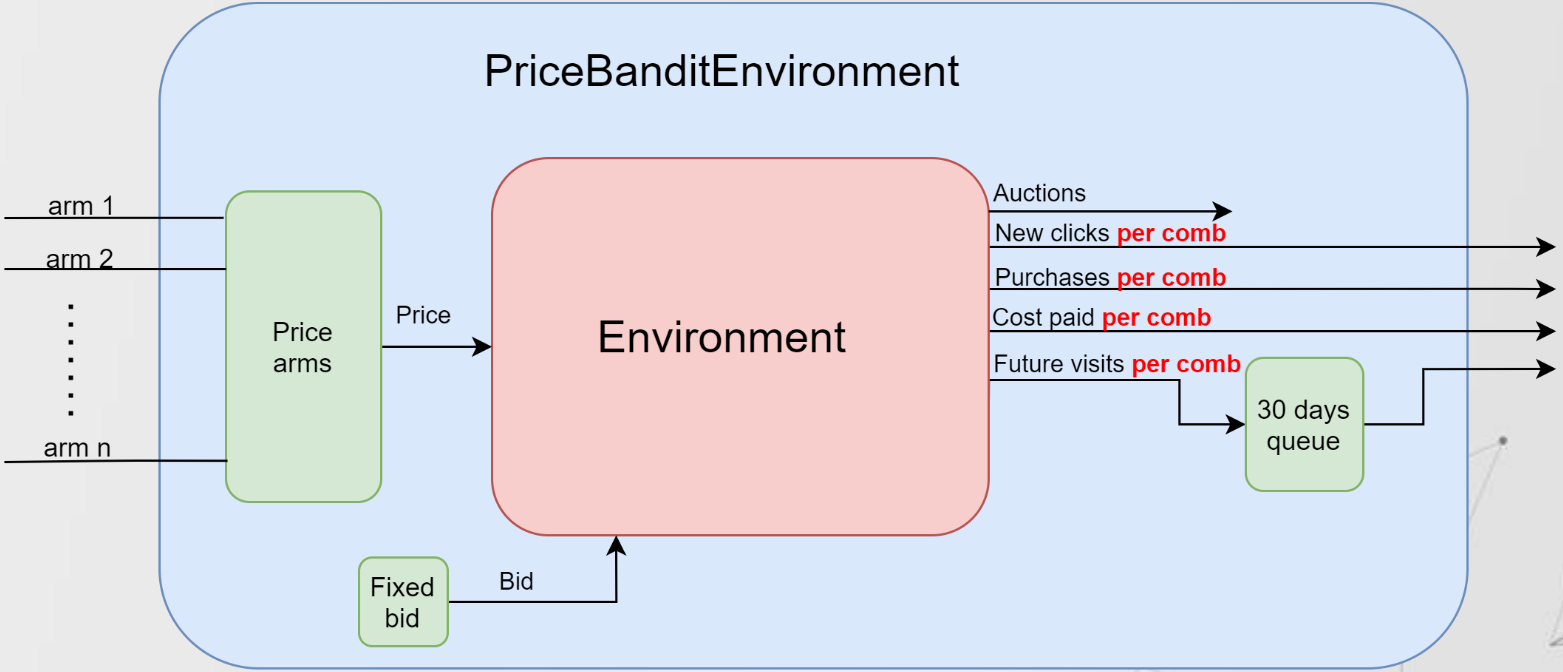
Online pricing

Discriminating between customer classes

OUR NUMBERS

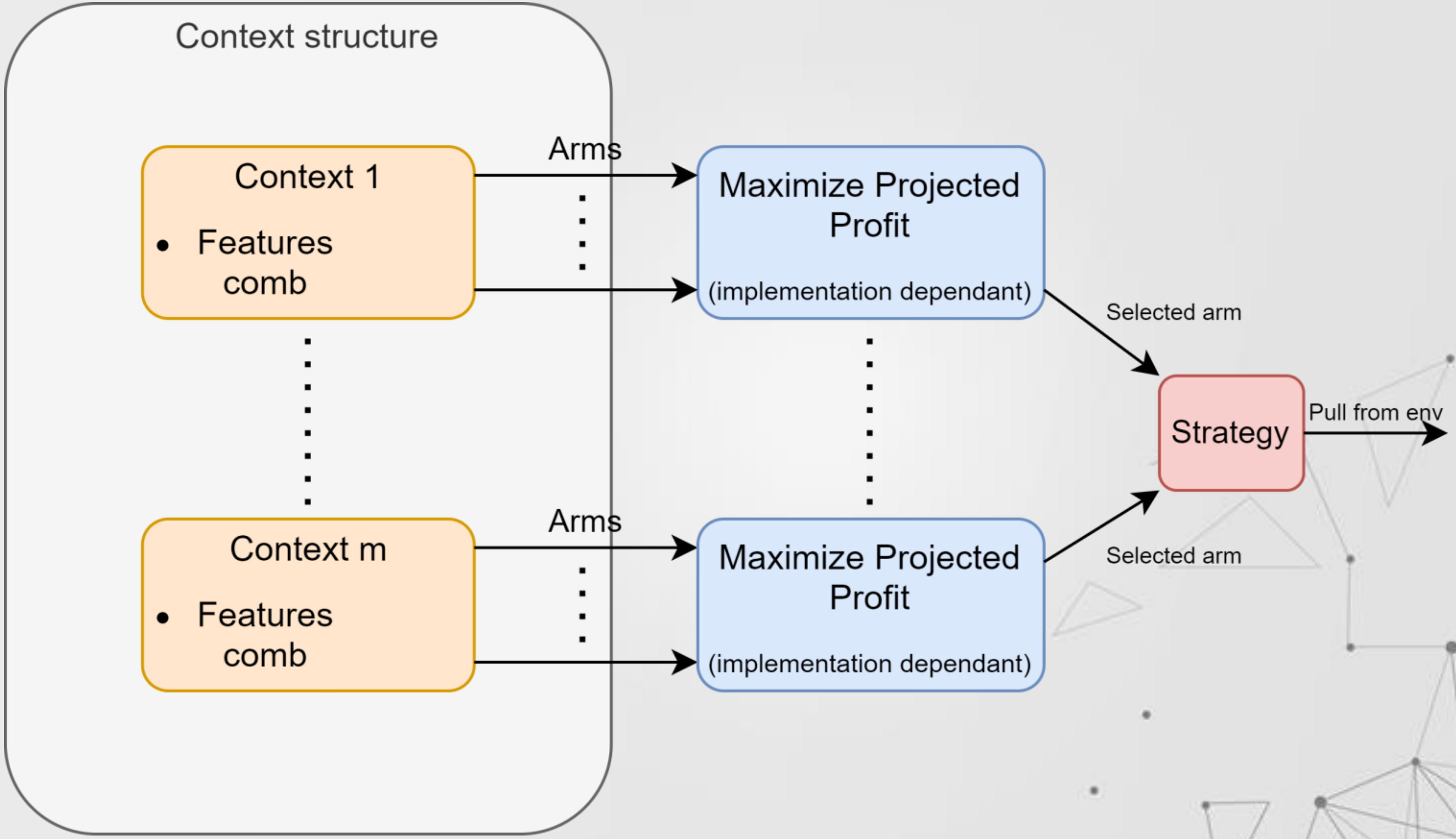
10 PRICES

1 BID



- Now, PriceBanditEnvironment returns new clicks, purchases, cost paid and future visits **per comb**

OPTIMAL PRICE DISCRIMINATING LEARNER



CHOICE OF THE NEXT STRATEGY

- Just like Step 3
- UCB1 and Thompson Sampling are applied **context-wise**

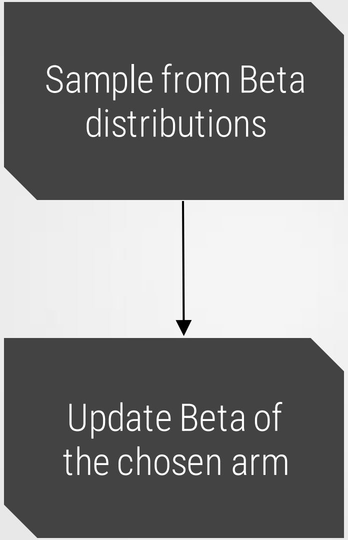
UCB OPTIMAL PRICE DISCRIMINATING LEARNER

$$\frac{purchases_a}{clicks_a} + \sqrt{\frac{2log(t)}{clicks_a}}$$

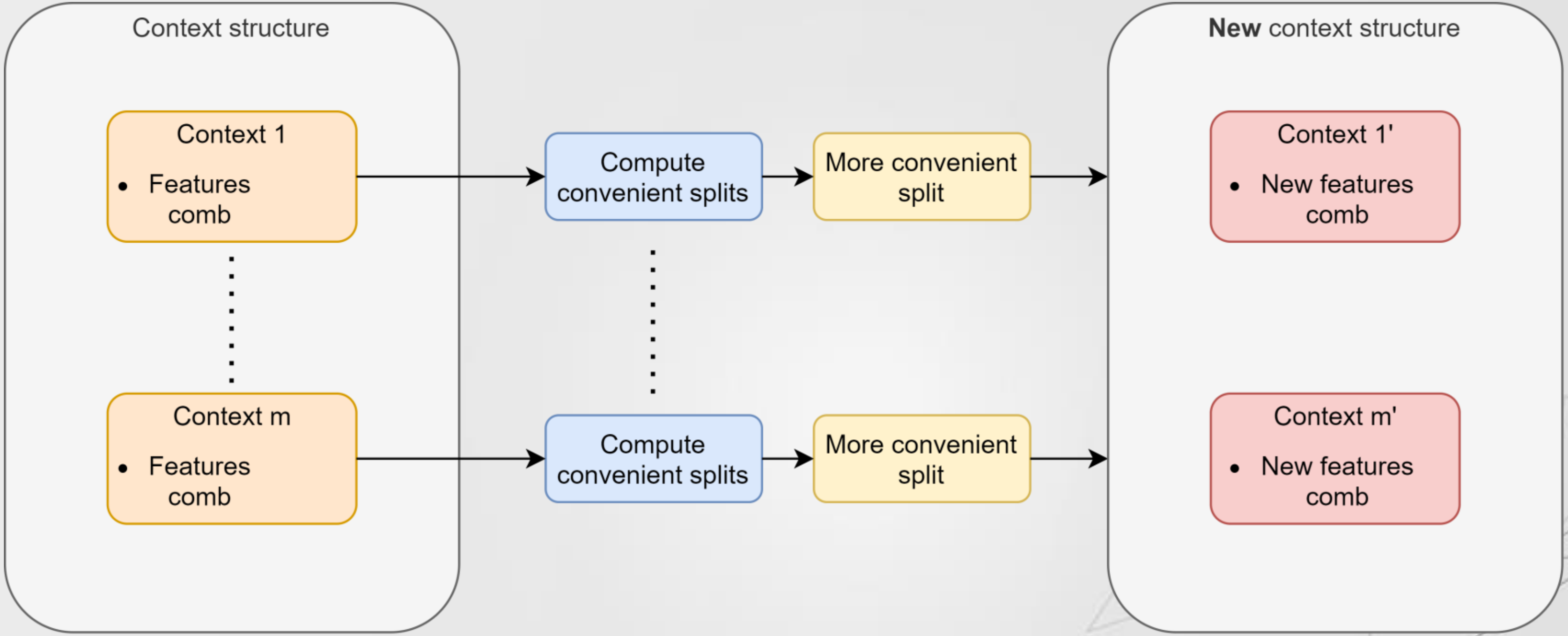
AVERAGES

RADII

TS OPTIMAL PRICE DISCRIMINATING LEARNER



CONTEXT STRUCTURE UPDATE



- For each context, compute the **convenient splits** (the lower bound of expected profit after the split is **higher** than the current lower bound)
- For each context, choose the **more convenient** split and create a new context structure

EXPLORATION

- Splits that could not be detected after the initial round robin are never detected, if their optimal arm is very sub-optimal for the current context

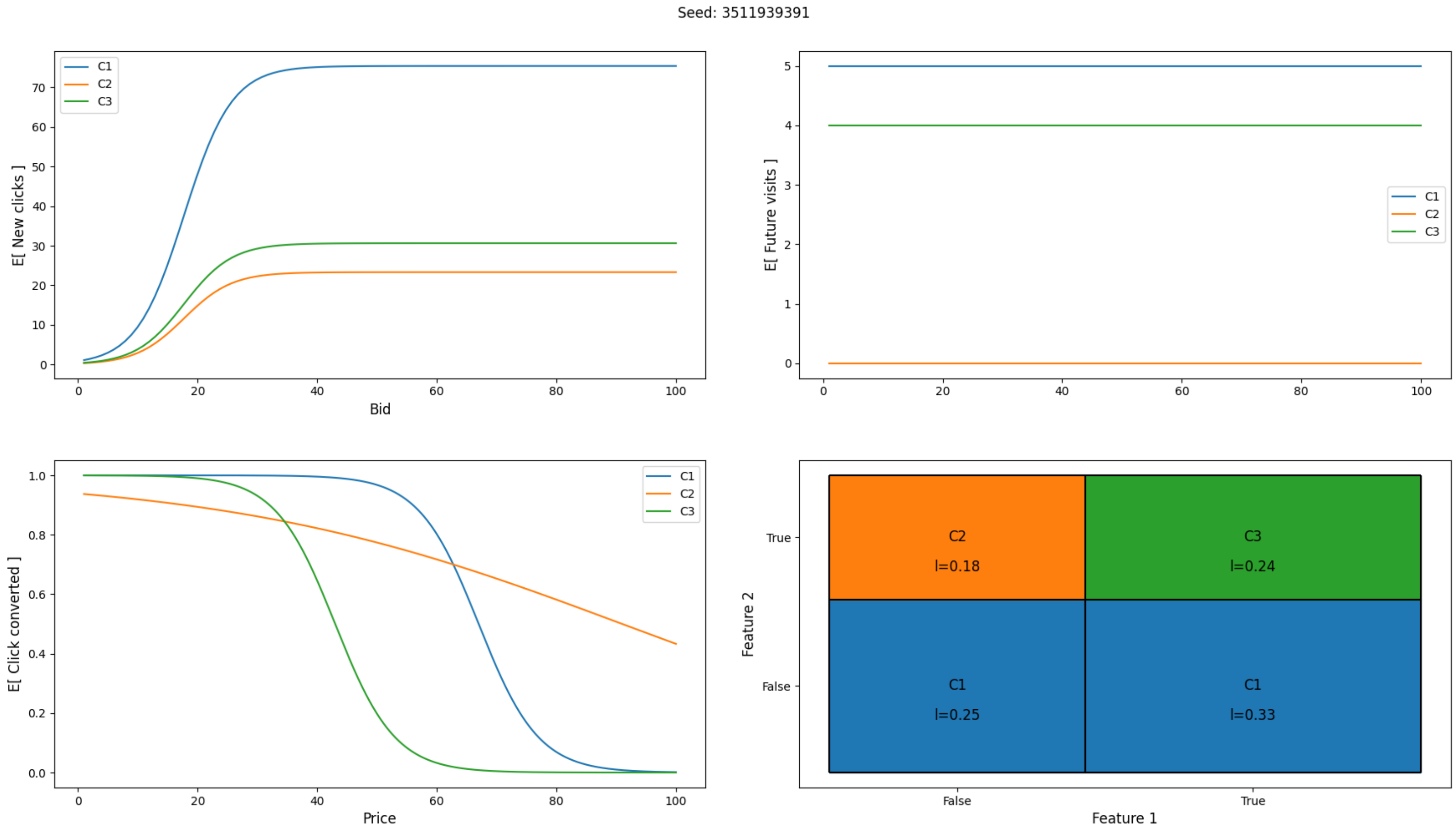
- Lower confidence bound never gets tighter

EXPLORATION

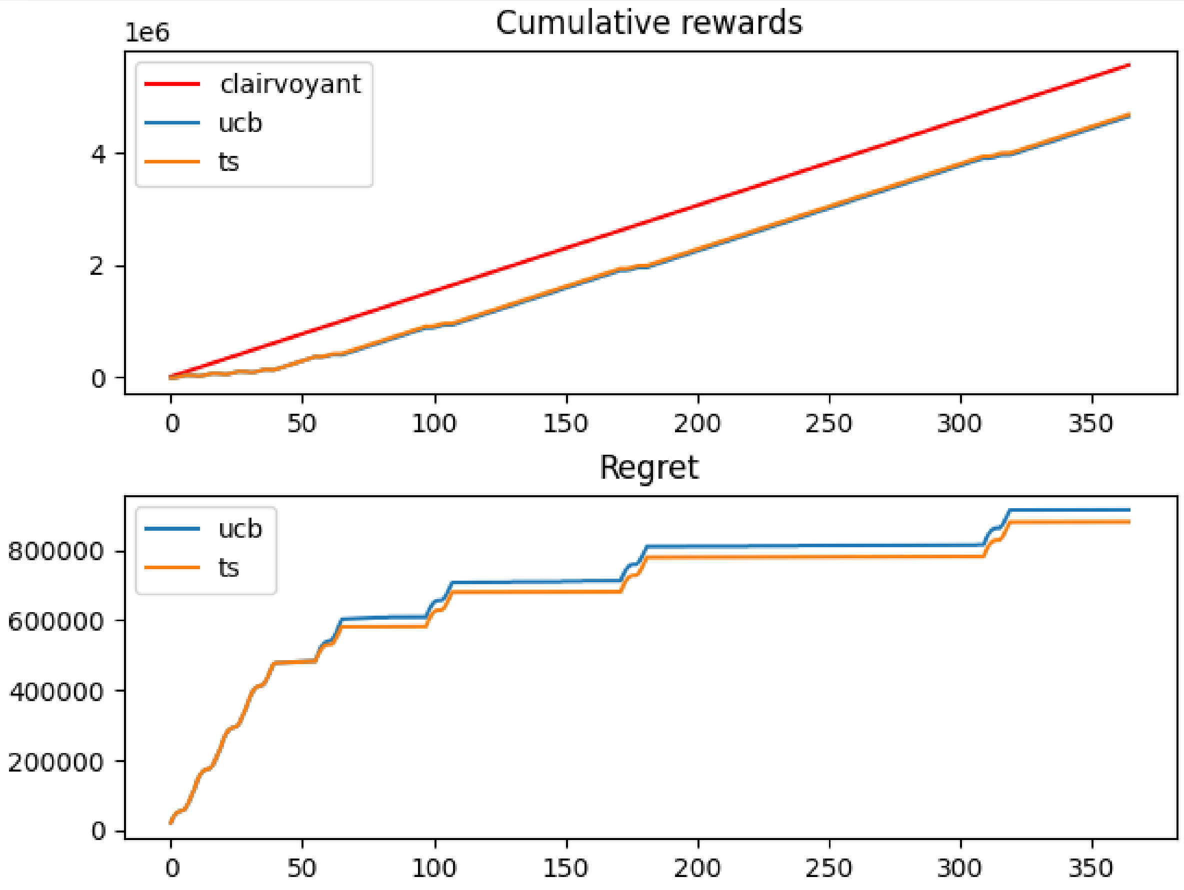
- **Explorative rounds:** contexts that could split perform round-robin selection
- Tightens the bounds of the current sub-optimal arms

HOW MANY EXPLORATIVE ROUNDS?

- Must be logarithmic
- After the completion of i^{th} explorative round, 2^i normal rounds must be completed before the next explorative round



Seed: 3511939391
E[New clicks]
Bid
C1
C2
C3
E[Future visits]
E[Click converted]
Price
Feature 2
Feature 1
True
False
C2
l=0.18
C3
l=0.24
C1
l=0.25
C1
l=0.33



Cumulative rewards
1e6
clairvoyant
ucb
ts
0
2
4
0
50
100
150
200
250
300
350
Regret
ucb
ts
0
200000
400000
600000
800000
0
50
100
150
200
250
300
350

Seed: 3511939391

Legend: context, normalized gap, number of pulls

Optimal pricing strategy: C1(TF, FF): 60.00, C2(FT): 90.00, C3(TT): 40.00

UCB with context generation:

Price	TF,FF	Gaps	Pulls	TT	Gaps	Pulls	FT	Gaps	Pulls
10.00		100.0	8		100.0	5		100.0	5
20.00		75.8	8		53.7	5		78.9	5
30.00		51.6	8		11.5	8		59.2	5
40.00		27.6	8		0.0	315		41.4	32
50.00		5.8	14		40.8	6		26.1	5
60.00		0.0	287		65.5	6		13.9	8
70.00		39.1	8		71.0	5		5.4	42
80.00		74.8	8		71.9	5		0.7	43
90.00		83.7	8		72.1	5		0.0	143
100.00		85.2	8		72.1	5		2.8	77

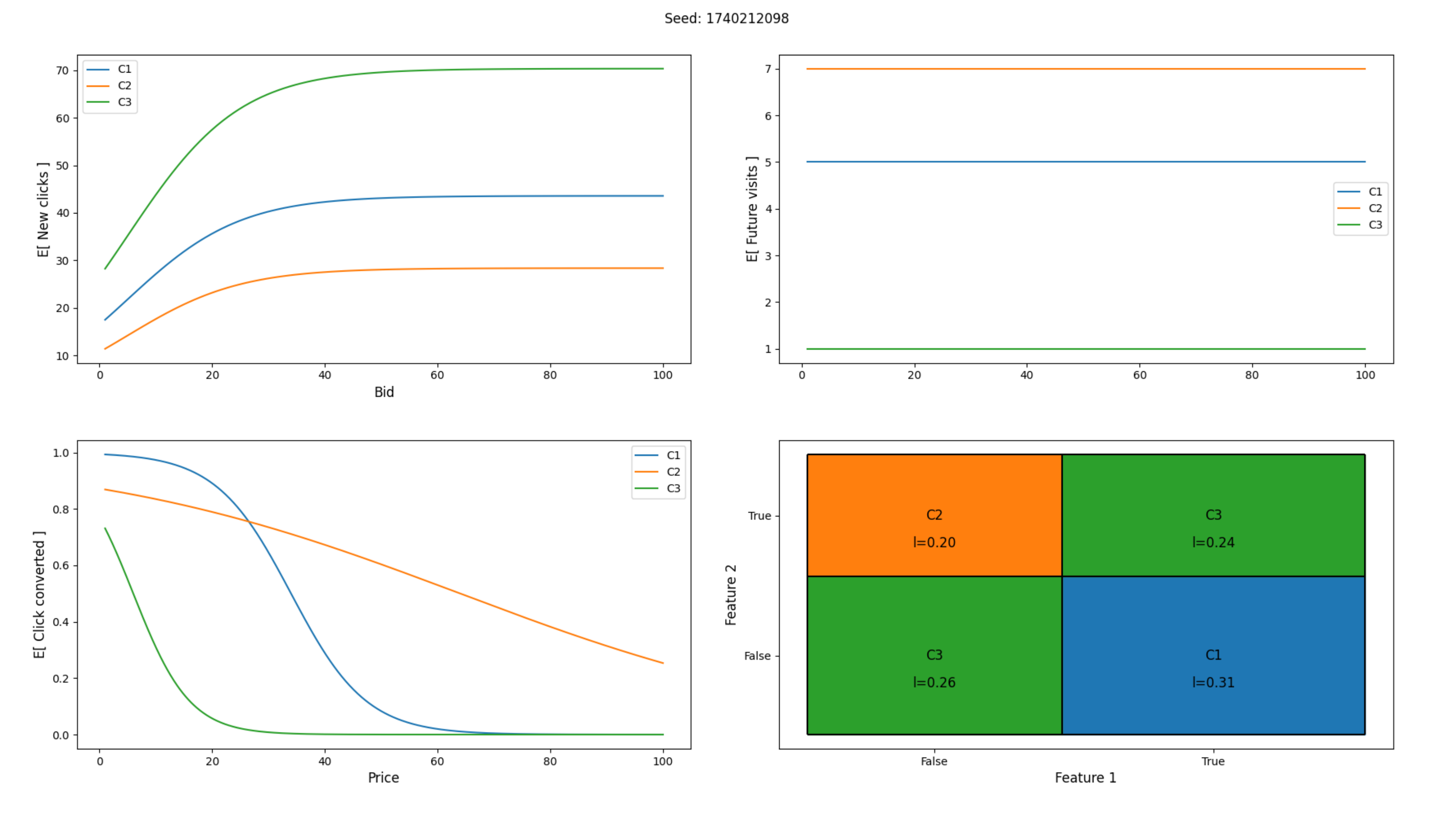
Performed splits:
Round 41 split on feature 2 with incentive 1147.21
Round 78 split on feature 1 with incentive 32.89

Optimal pricing strategy: C1(TF, FF): 60.00, C2(FT): 90.00, C3(TT): 40.00

TS with context generation:

Price	TF,FF	Gaps	Pulls	TT	Gaps	Pulls	FT	Gaps	Pulls
10.00		100.0	8		100.0	4		100.0	4
20.00		75.8	8		53.7	4		78.9	4
30.00		51.6	8		11.5	4		59.2	4
40.00		27.6	8		0.0	328		41.4	17
50.00		5.8	8		40.8	4		26.1	4
60.00		0.0	293		65.5	5		13.9	5
70.00		39.1	8		71.0	4		5.4	17
80.00		74.8	8		71.9	4		0.7	99
90.00		83.7	8		72.1	4		0.0	53
100.00		85.2	8		72.1	4		2.8	158

Performed splits:
Round 41 split on feature 2 with incentive 2994.85
Round 54 split on feature 1 with incentive 122.42



Seed: 1740212098
E[New clicks]
Bid
C1
C2
C3
E[Future visits]
E[Click converted]
Price
Feature 2
Feature 1
True
False
C2
l=0.20
C3
l=0.24
C3
l=0.26
C1
l=0.31

Cumulative rewards
1e6
2.0
1.5
1.0
0.5
0.0
clairvoyant
ucb
ts
0
50
100
150
200
250
300
350
Regret
250000
200000
150000
100000
50000
0
ucb
ts
0
50
100
150
200
250
300
350

Seed: 1740212098

Optimal pricing strategy: C1(TF): 30.00, C2(FT): 70.00, C3(TT, FF): 20.00

UCB with context generation:

Price	TF	Gaps	Pulls	FF	Gaps	Pulls	TT	Gaps	Pulls	FT	Gaps	Pulls
10.00		100.0	4		100.0	4		100.0	4		100.0	4
20.00		32.1	4		0.0	4		0.0	6		71.3	4
30.00		0.0	297		43.7	4		43.7	12		46.5	4
40.00		29.3	17		58.6	7		58.6	10		26.5	4
50.00		68.2	6		61.8	125		61.8	20		12.0	12
60.00		85.3	7		62.3	23		62.3	30		3.2	30
70.00		90.6	4		62.4	31		62.4	44		0.0	91
80.00		92.1	18		62.5	43		62.5	59		1.7	89
90.00		92.5	4		62.5	54		62.5	79		7.3	100
100.00		92.6	4		62.5	70		62.5	101		15.5	27

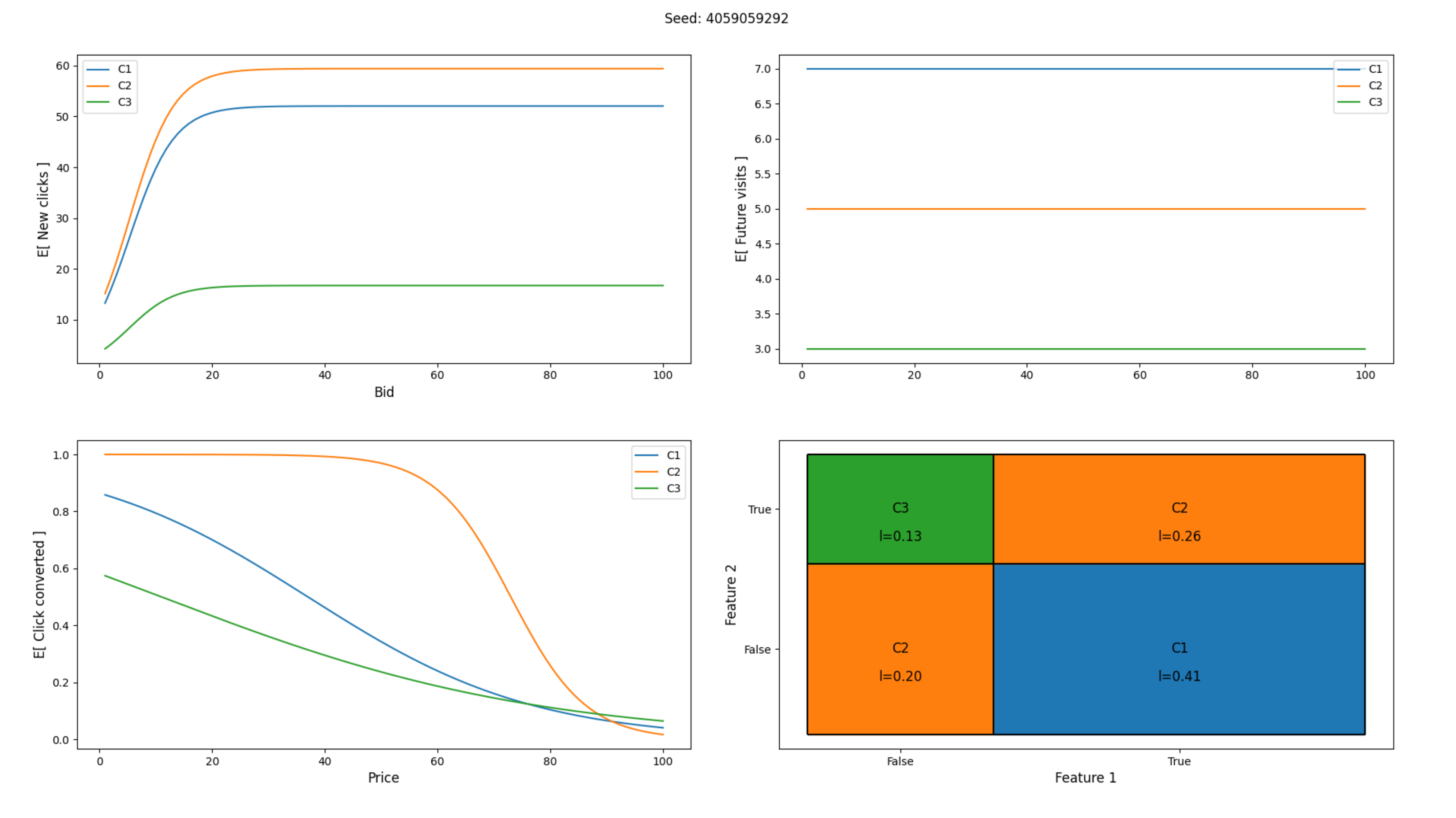
Performed splits:
Round 41 split on feature 2 with incentive 2983.76
Round 42 split on feature 1 with incentive 149.32
Round 44 split on feature 1 with incentive 434.91

Optimal pricing strategy: C1(TF): 30.00, C2(FT): 70.00, C3(TT, FF): 20.00

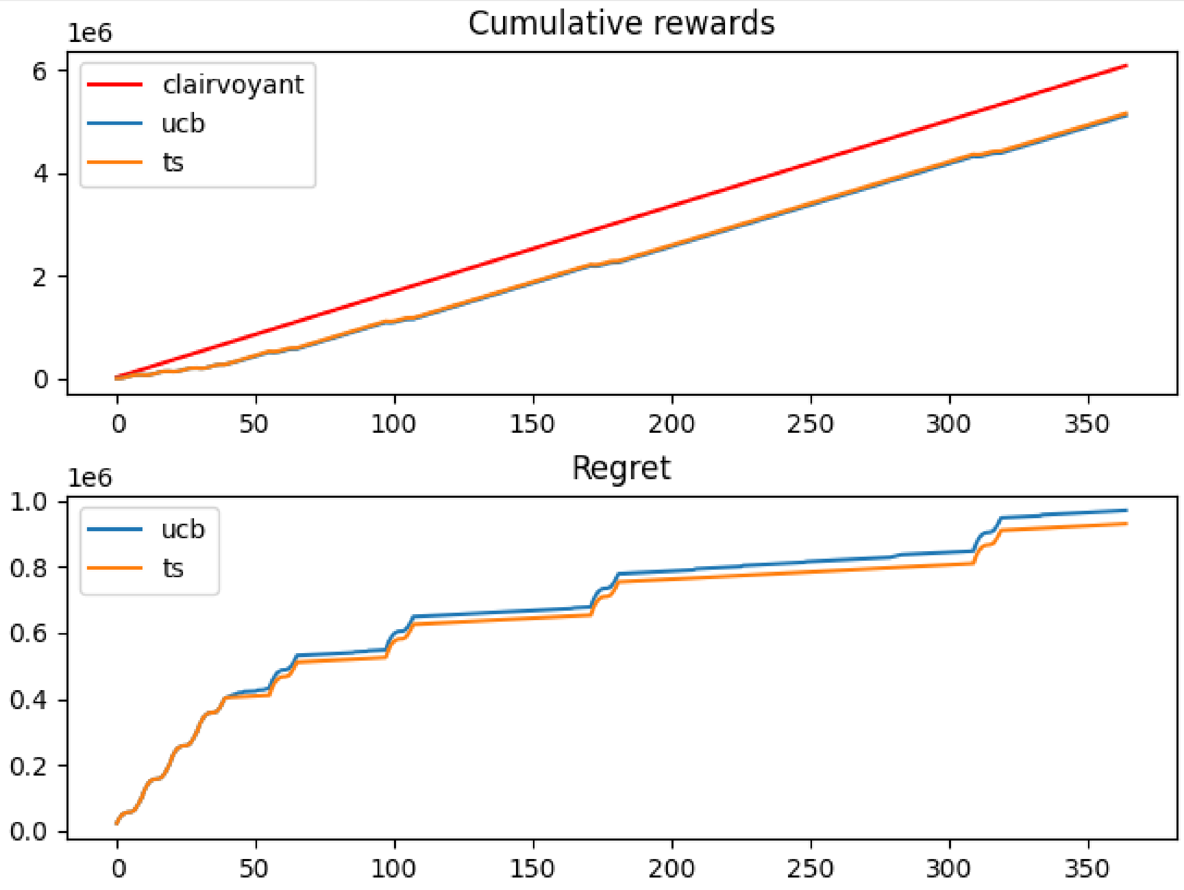
TS with context generation:

Price	TT	Gaps	Pulls	FT	Gaps	Pulls	TF	Gaps	Pulls	FF	Gaps	Pulls
10.00		100.0	4		100.0	4		100.0	4		100.0	4
20.00		0.0	263		71.3	4		32.1	4		0.0	262
30.00		43.7	11		46.5	4		0.0	328		43.7	27
40.00		58.6	19		26.5	4		29.3	4		58.6	5
50.00		61.8	8		12.0	15		68.2	4		61.8	10
60.00		62.3	11		3.2	11		85.3	5		62.3	7
70.00		62.4	11		0.0	245		90.6	4		62.4	10
80.00		62.5	10		1.7	6		92.1	4		62.5	13
90.00		62.5	13		7.3	60		92.5	4		62.5	13
100.00		62.5	15		15.5	12		92.6	4		62.5	14

Performed splits:
Round 41 split on feature 2 with incentive 5179.85
Round 42 split on feature 1 with incentive 2768.04
Round 43 split on feature 1 with incentive 78.48



Seed: 4059059292
E[New clicks]
Bid
C1
C2
C3
E[Future visits]
E[Click converted]
Price
Feature 1
Feature 2
True
False
C3
l=0.13
C2
l=0.26
C2
l=0.20
C1
l=0.41



Cumulative rewards
1e6
6
4
2
0
clairvoyant
ucb
ts
0
50
100
150
200
250
300
350
Regret
1e6
1.0
0.8
0.6
0.4
0.2
0.0
ucb
ts
0
50
100
150
200
250
300
350

Seed: 4059059292

Optimal pricing strategy: C1(TF): 50.00, C2(TT, FF): 60.00, C3(FT): 60.00

UCB with context generation:

Price	TT,TF,FT,FF	Gaps	Pulls
10.00		100.0	8
20.00		69.4	8
30.00		43.2	8
40.00		22.1	8
50.00		6.7	16
60.00		0.0	274
70.00		12.1	19
80.00		41.3	8
90.00		62.2	8
100.00		71.1	8

TS with context generation:

Price	TT,TF,FT,FF	Gaps	Pulls
10.00		100.0	8
20.00		69.4	8
30.00		43.2	8
40.00		22.1	8
50.00		6.7	9
60.00		0.0	292
70.00		12.1	8
80.00		41.3	8
90.00		62.2	8
100.00		71.1	8

NO SPLITS FOUND

FINAL CONSIDERATIONS

- We found some discrepancies between the two algorithm, leading to potential differences in the early convergence
- We are estimating a conversion rate, so a small change can lead to dramatic changes in the expected profit
- If a small amount of people buy at a very high price, the revenue increases a lot: this happen when we overestimate the conversion rate for high prices

UCB VS TS

- UCB tends to be optimistic, since it uses upper bounds, hence it gives more shots to higher prices

- TS converges faster, while UCB "wastes" time exploring expensive arms

- Thanks to the nature of the bounds, the regret remains logarithmic, so it converges in a decent number of rounds

Step 5

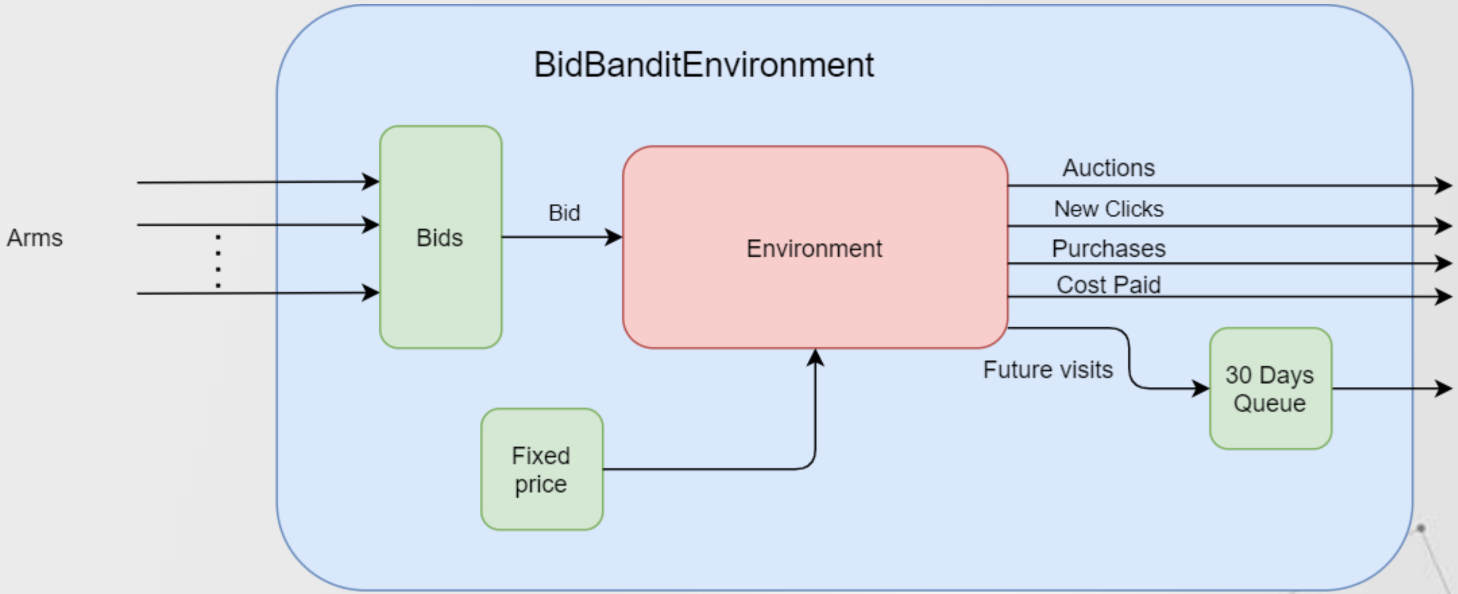
Online bidding

Without discriminating between customer classes

OUR NUMBERS

10 BIDS

1 PRICE



- Symmetric to PriceBanditEnvironment.
- Now environment returns also the total number of auctions
- Our random variable is now the probability of winning an auction


arm1
arm2
arm n
Safe arms logic
Safe Arms
BidBanditEnvironment
Auctions
New clicks
Purchases
Cost paid
Future visits

SAFETY CONSTRAINT

We don't want to pull arms that could give us negative profit!

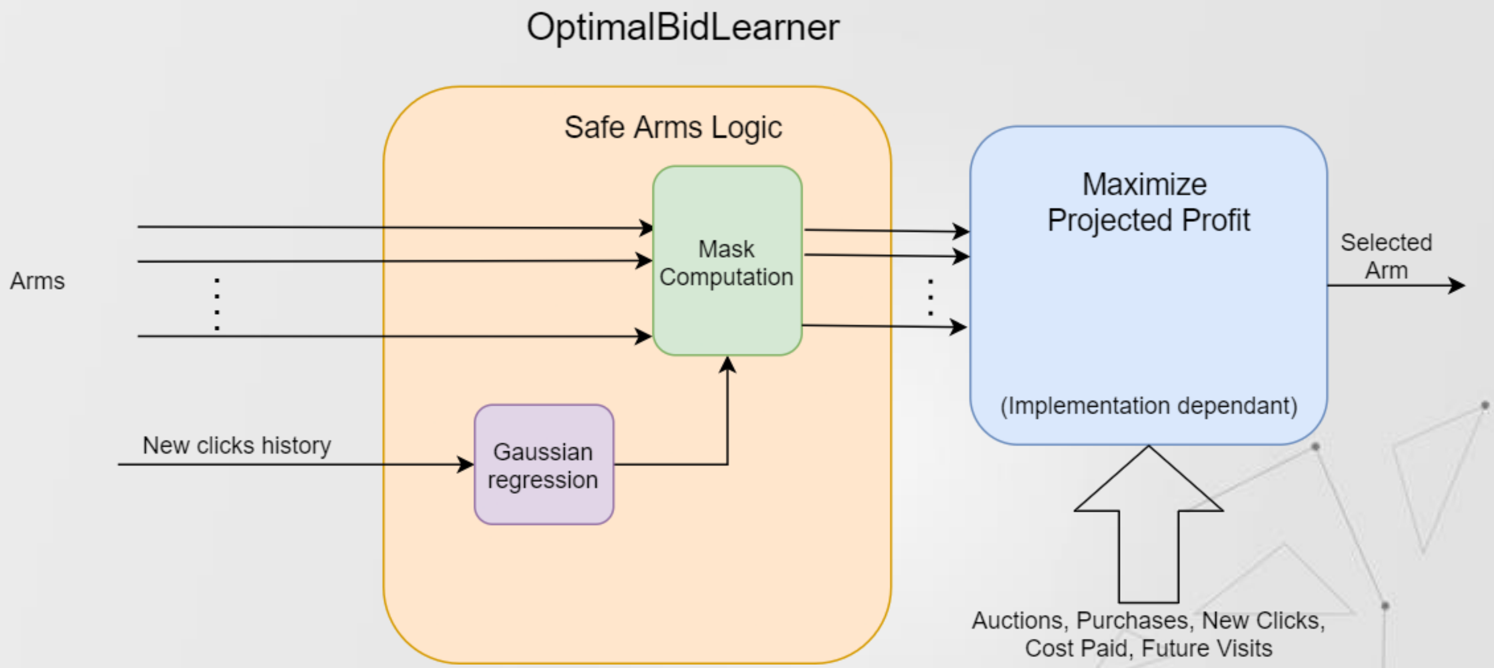
- The main difference between this step and step 3 is the introduction of a safety constraint.
- A *Safe arm* is defined as an arm which revenue won't be negative with a certain probability.
- We use Gaussian regression to approximate the data on our random variable on a normal curve.
- Given the mean and variance of the normal curve, we extract the n-th percentile.

SAFETY CONSTRAINT

As a result, we have a value for the random variable which is the lowest on (100-n)% of the cases!

For example if we take the 20th percentile, we are sure that 80% of the times that's the lowest value we will find.

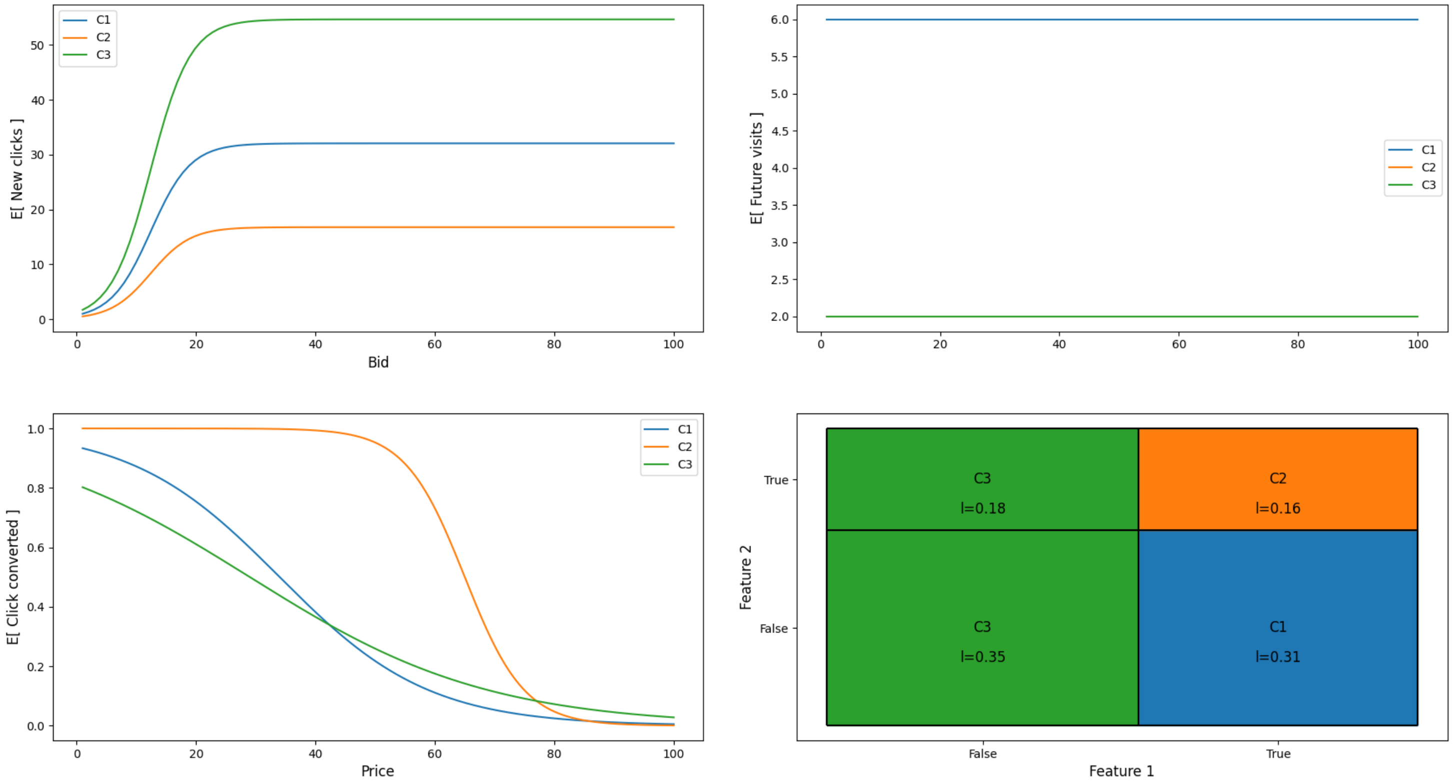
- Given this safety-bound value, we can compute the expected profit with the usual formula.
- An arm is safe if and only if its new expected profit is strictly greater than 0.
- The learner will only pull and arm which is tagged as *safe*.

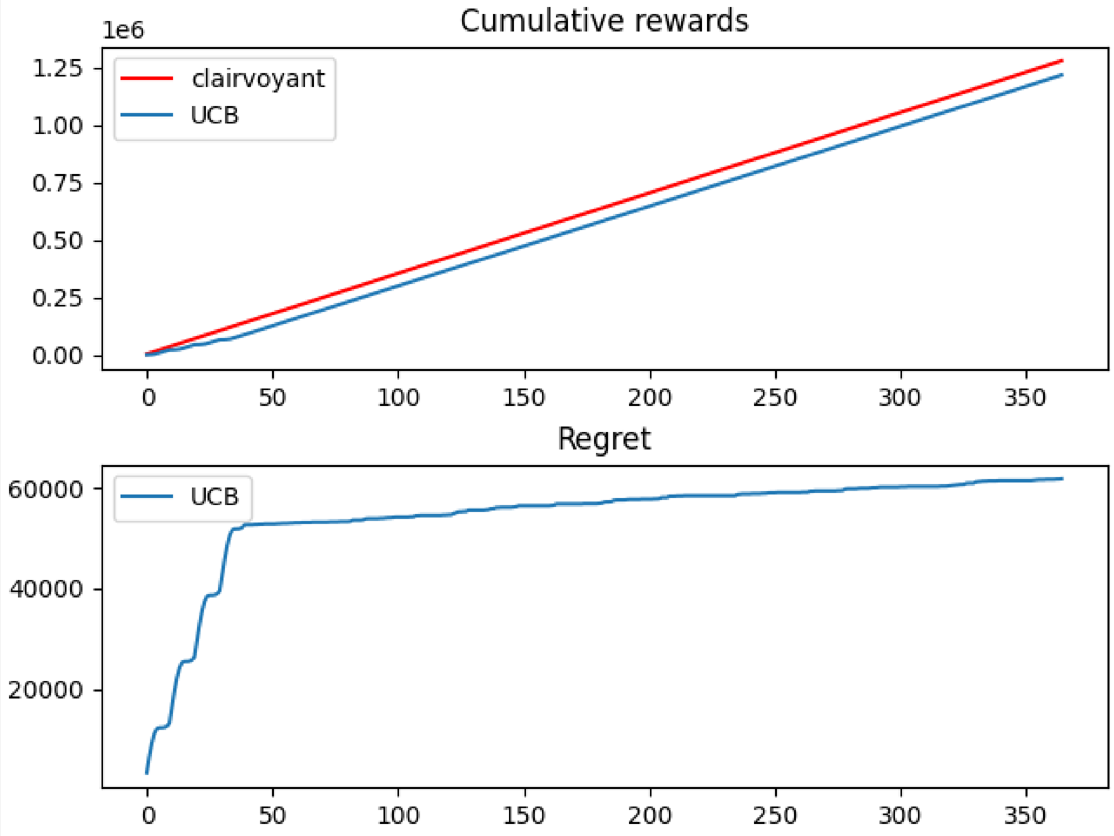

OptimalBidLearner
Safe Arms Logic
Mask Computation
Arms
New clicks history
Gaussian regression
Maximize Projected Profit
(Implementation dependant)
Selected Arm
Auctions, Purchases, New Clicks, Cost Paid, Future Visits

$$\frac{New\ Clicks_a}{Auctions_a} + \sqrt{\frac{2 \cdot \log(t)}{Auctions_a}}$$

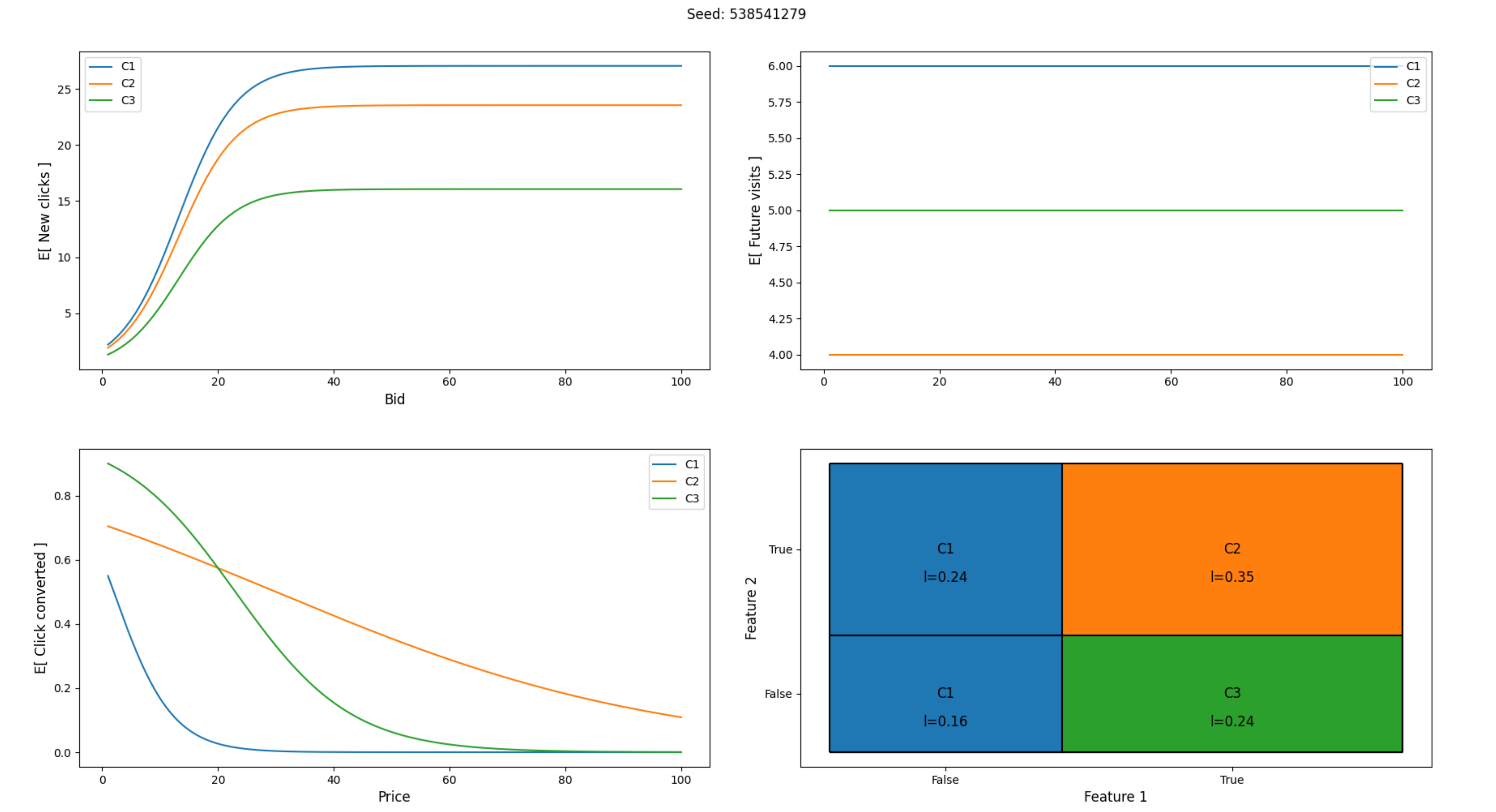
- Standard UCB computation
- The denominator is the number of auctions, since we are approximating the probability of winning an auction

Seed: 3077083550

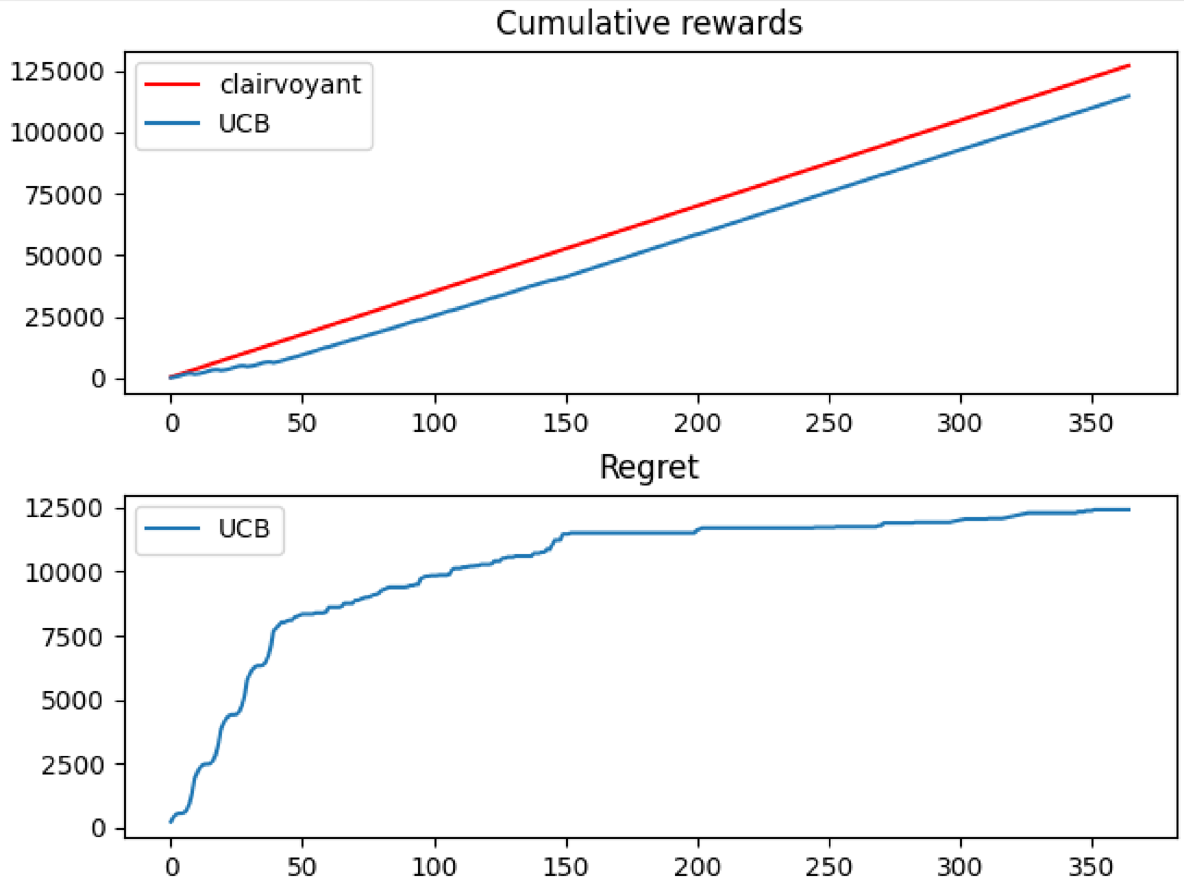




Cumulative rewards
1e6
1.25
1.00
0.75
0.50
0.25
0.00
clairvoyant
UCB
0
50
100
150
200
250
300
350
Regret
60000
40000
20000
UCB
0
50
100
150
200
250
300
350



Seed: 538541279
E[New clicks]
Bid
C1
C2
C3
E[Future visits]
E[Click converted]
Price
Feature 2
Feature 1
True
False
C1
l=0.24
C2
l=0.35
C1
l=0.16
C3
l=0.24



Cumulative rewards
clairvoyant
UCB
125000
100000
75000
50000
25000
0
0
50
100
150
200
250
300
350
Regret
UCB
12500
10000
7500
5000
2500
0
0
50
100
150
200
250
300
350

Step 6

Online joint pricing/bidding

Without discriminating between customer classes

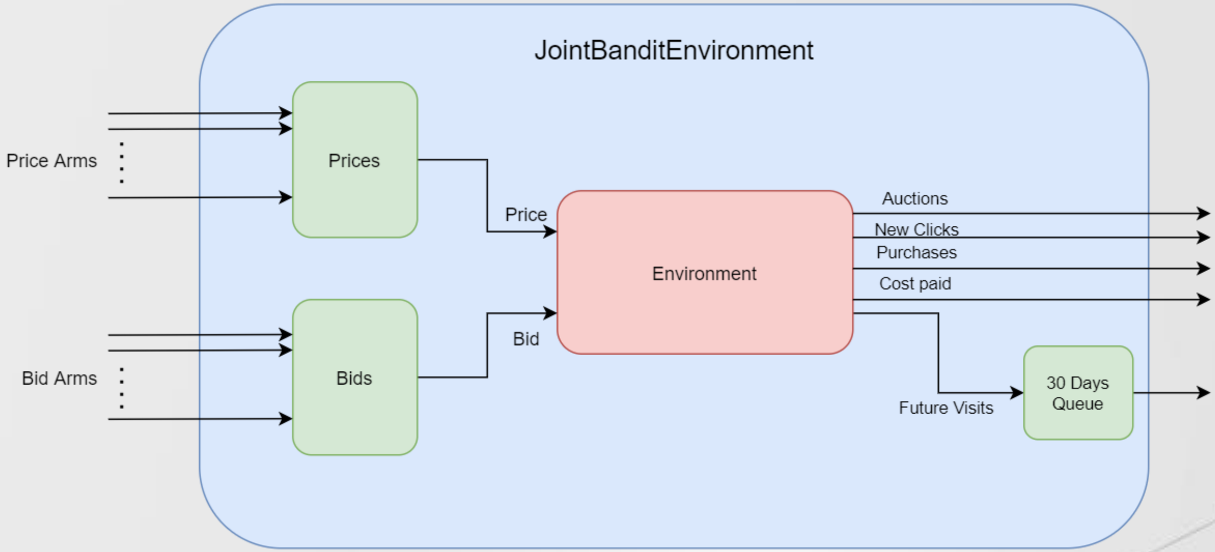
OUR NUMBERS

10

PRICES

10

BIDS



- The environment is similar to *PriceBanditEnvironment* and *BidBanditEnvironment*, but incorporates the features of both.
- Accepts both a price and a bid arm.
- No fixed values.

NEXT ARMS CHOICE

Fix bid

We can select whatever bid we want, thanks to the results of Step 1

Obtain optimal price

Fix optimal price

Fix the price to our optimal one

Obtain optimal bid

Pull joint arms

Pull from the environment, using both optimal arms

Axis projection

- We noticed that every quantity, depends only on one of our set of arms.
- This means that if we are smart, we can re-use all the functionalities from the previous learners.
- If we want to get the average of that quantity for each price arm, we can fix a price and sum together the realizations even if coming from different bids. Of course this remains valid for the specular case when a quantity depends only on the bid.
- This allows us to avoid exploring all the 10x10 possible joint arms.
- All the computations we have to do, can be recycled from *OptimalPriceLearner* and *OptimalBidLearner*, we just need to properly project the data!

QUANTITIES DEPENDENCY

Conversion rate, future visits

All depend on the price

New clicks, auctions, cost per click

All depend on the bid

Same features as before

- As we already said, the projection technique allowed us to re-use efficiently the code (and the samples).

- The computation of the confidence bound for our random variables is the same as before.

- We still have our safety constraint mechanism which operates only on the bids.

Same results as before

- The convergence results are very similar to the previous cases.

- The absence of cross-dependency allowed our learner to learn as fast as before even though now there are two different random variables to estimate.

Round Robin

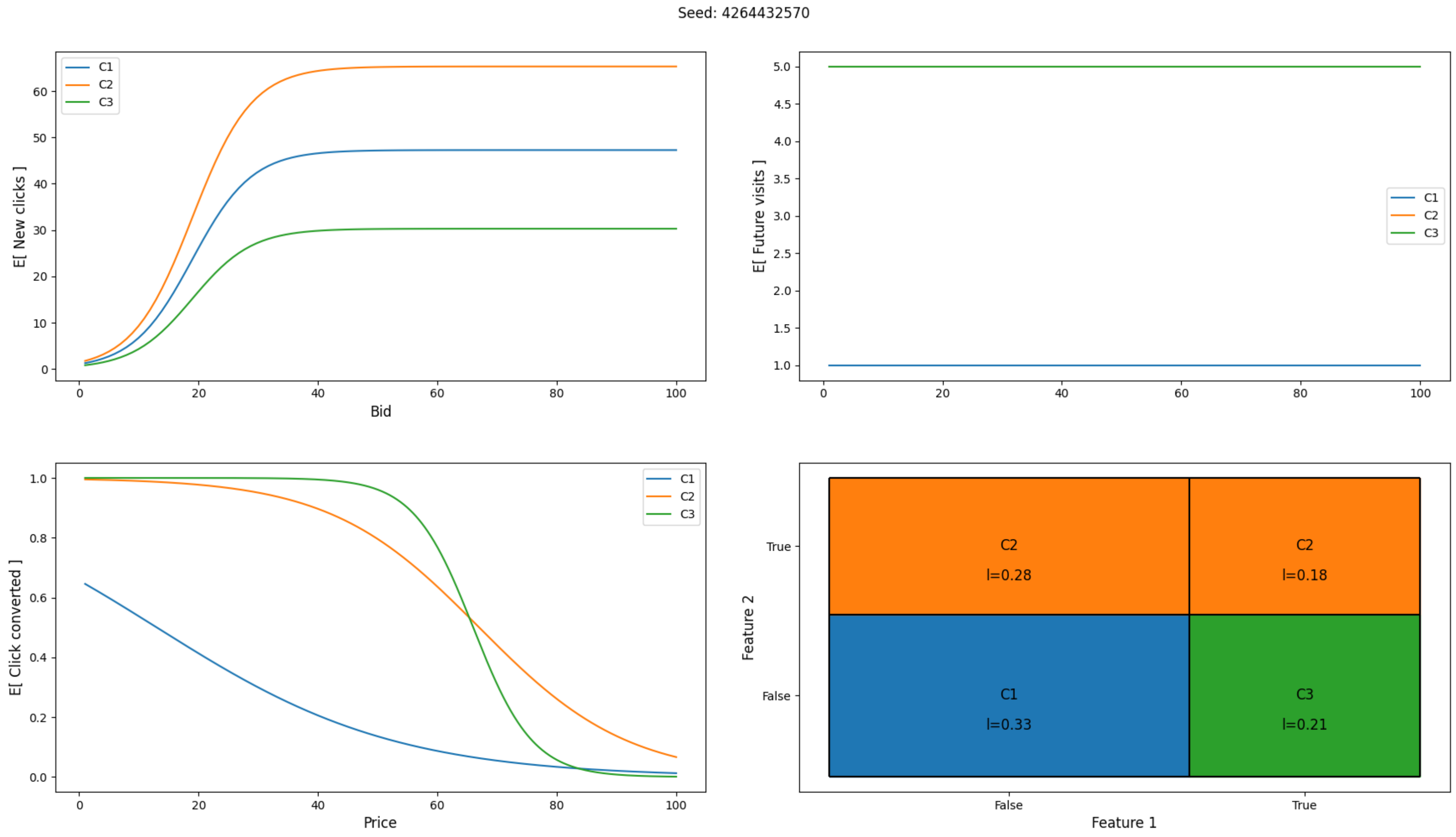
- As always, some Round Robin rounds must be performed at the beginning of the algorithm to have some initial data to work with.

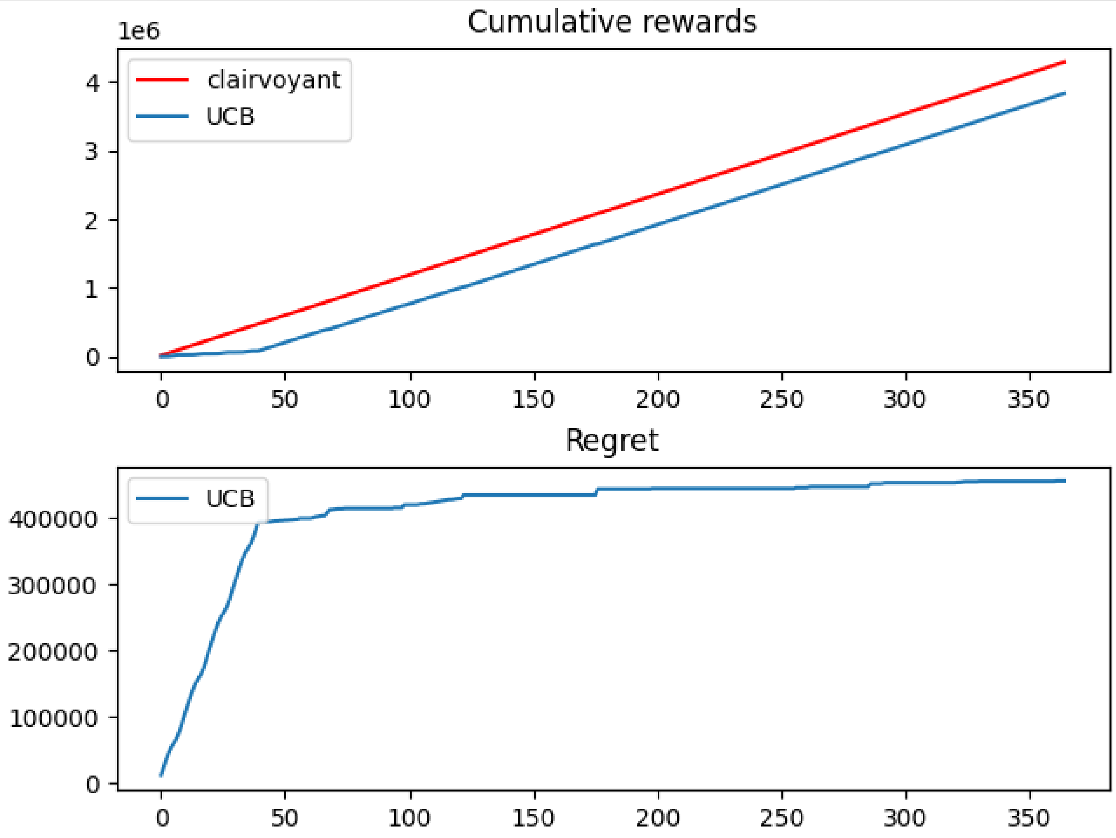
- One naïve approach would be to cycle through all the price/bids combinations.

- This would imply to run 100 round robin rounds. Not really smart.

Round Robin

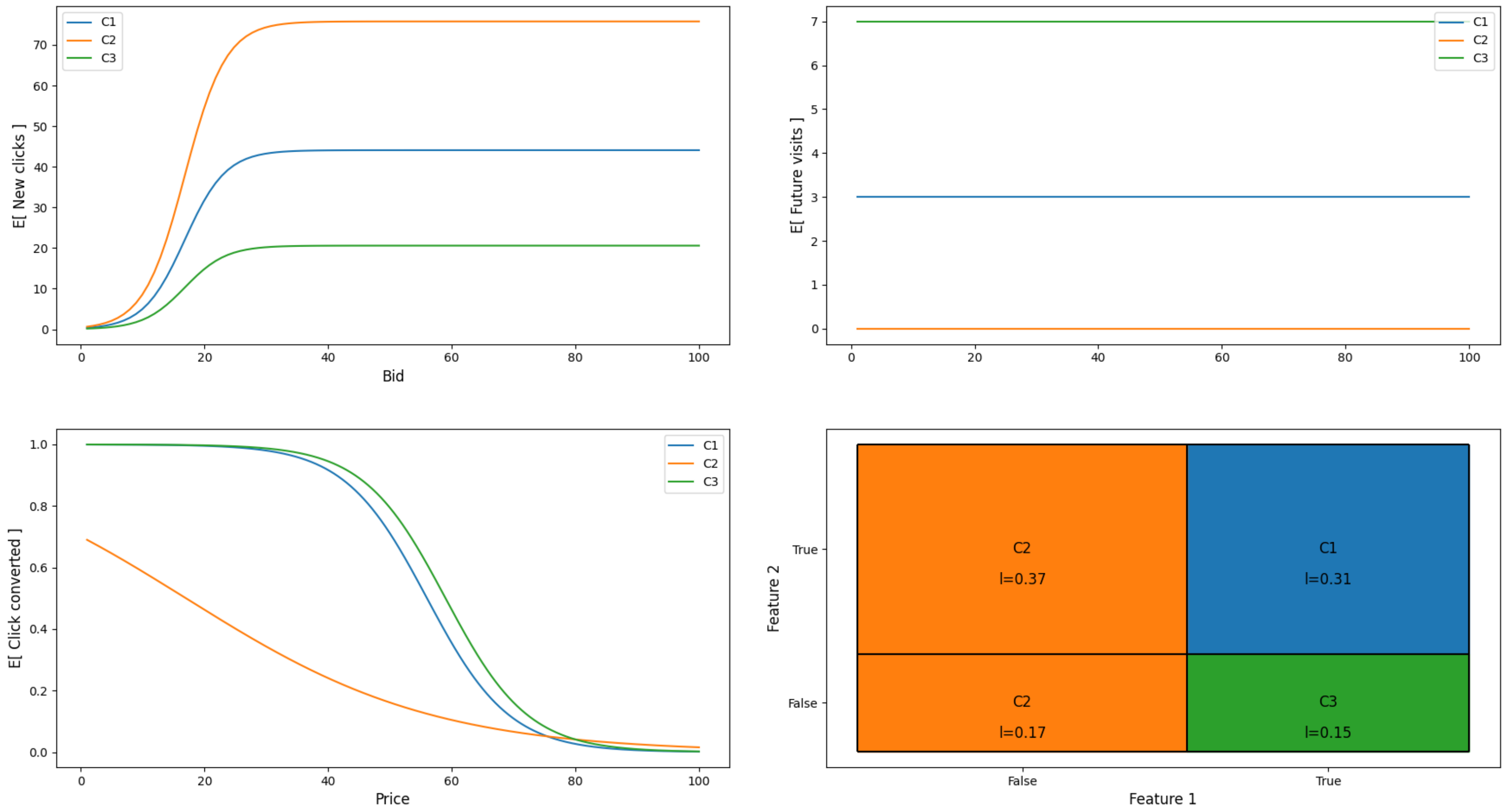
- We can exploit the single arm dependency of all the quantities to avoid pulling unnecessary combinations.
- Then we only have to cycle through 10 prices and 10 bids, which can be done in 10 rounds. Due to the reward delay, actually we have to Round Robin for at least 40 rounds.
- In principle we could select whatever arm we want, the important thing is that after 10 rounds we must have pulled all the prices and all the bids.
- For simplicity we opted to "explore" the diagonal, so pulling (price 1, bid 1), then (price 2, bid 2) and so on.

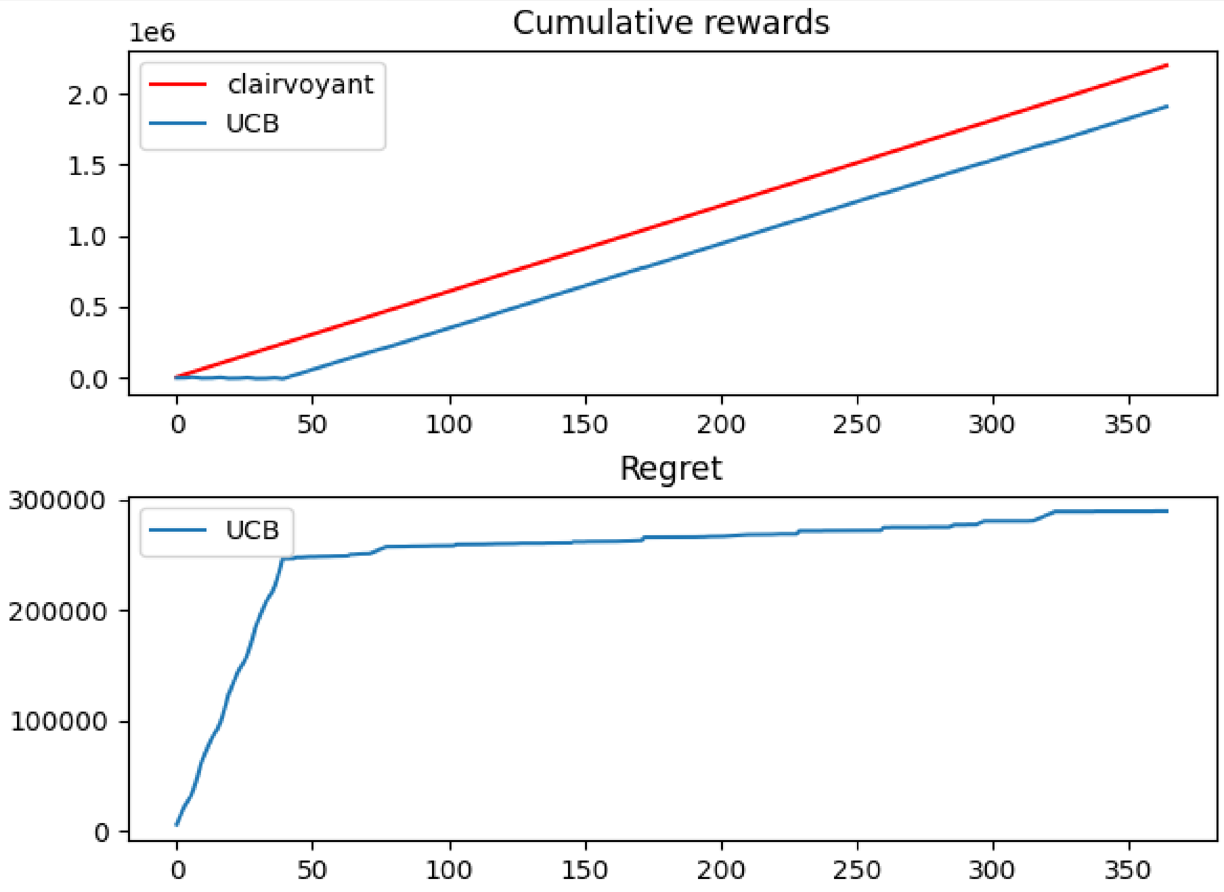

Seed: 4264432570
E[New clicks]
Bid
C1
C2
C3
E[Future visits]
E[Click converted]
Price
Feature 2
Feature 1
True
False
C2
l=0.28
C2
l=0.18
C1
l=0.33
C3
l=0.21



Cumulative rewards
1e6
clairvoyant
UCB
4
3
2
1
0
0
50
100
150
200
250
300
350
Regret
UCB
400000
300000
200000
100000
0
0
50
100
150
200
250
300
350

Seed: 1966222620




Cumulative rewards
1e6
clairvoyant
UCB
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200
250
300
350
Regret
UCB
300000
200000
100000
0
0
50
100
150
200
250
300
350

Step 7

Online joint pricing/bidding

Discriminating between customer classes

OUR NUMBERS

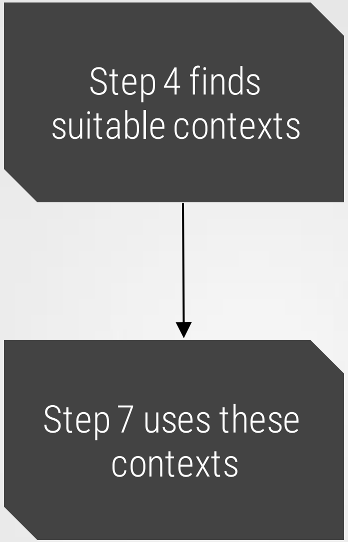
10

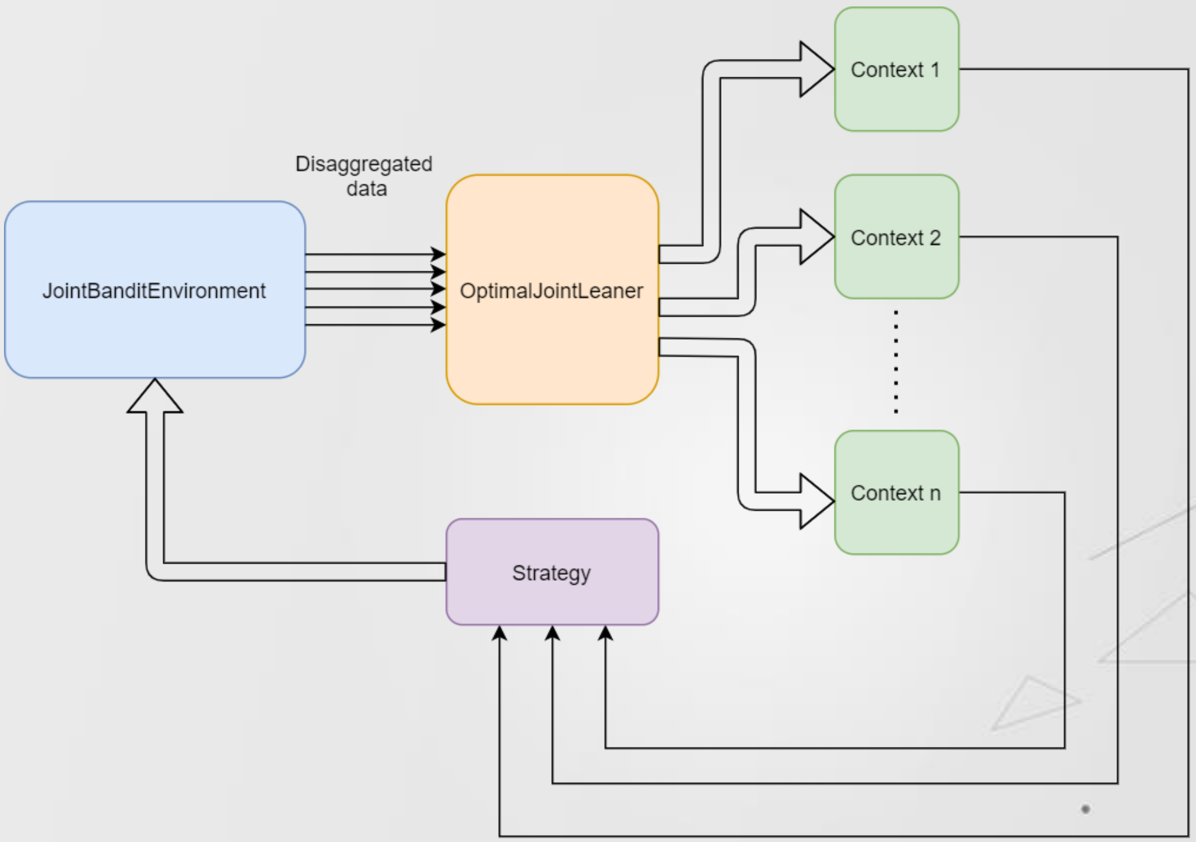
PRICES

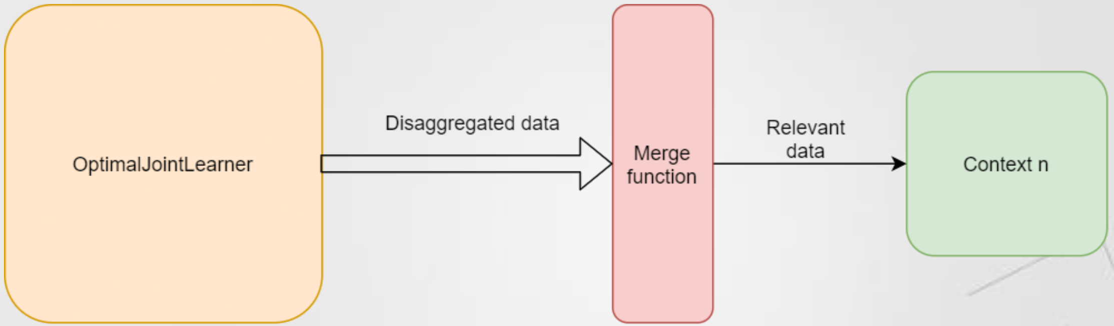
10

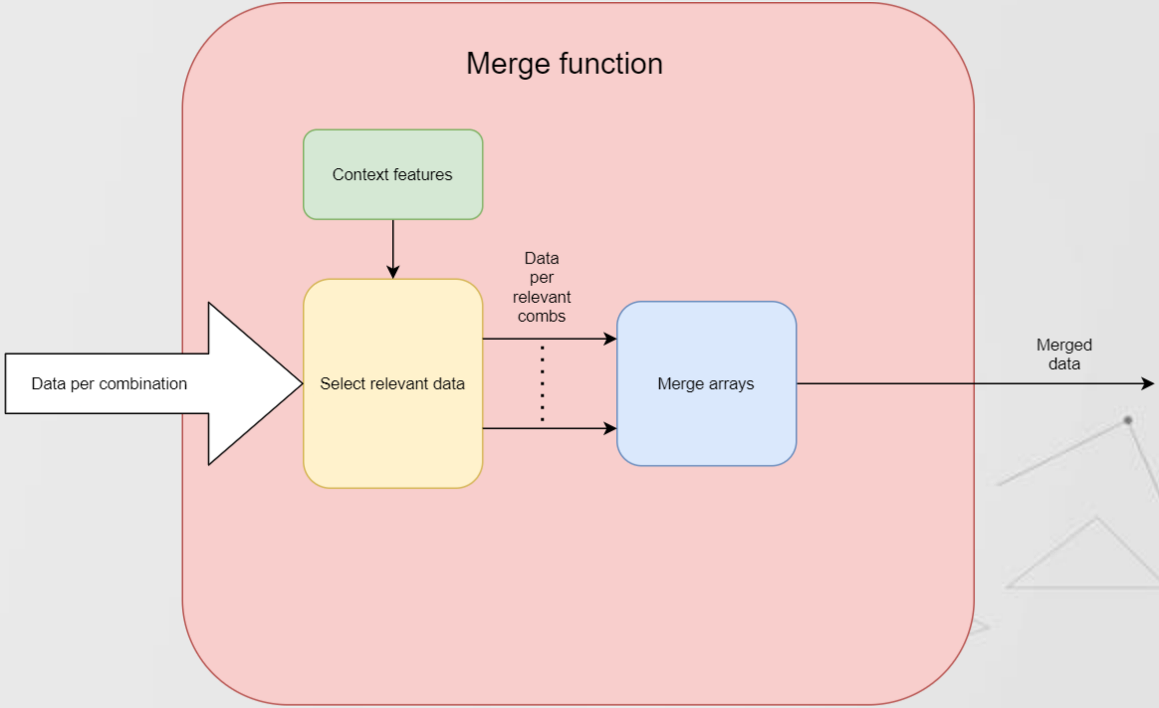
BIDS

CONTEXTS




Disaggregated data
JointBanditEnvironment
OptimalJointLeaner
Context 1
Context 2
Context n
Strategy


OptimalJointLearner
Disaggregated data
Merge function
Relevant data
Context n


Merge function
Context features
Data per combination
Select relevant data
Data
per
relevant
combs
Merge arrays
Merged
data

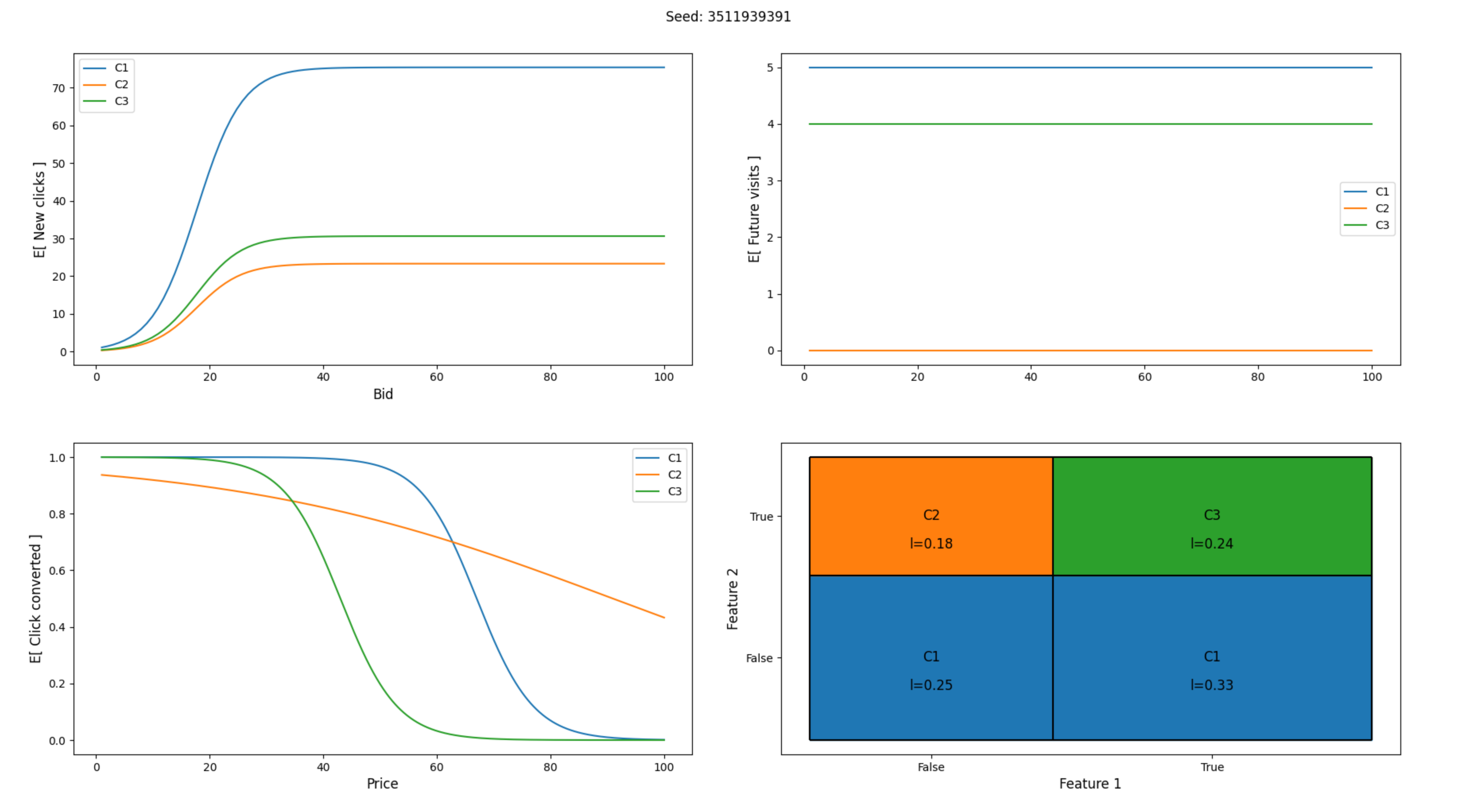
LEARNING

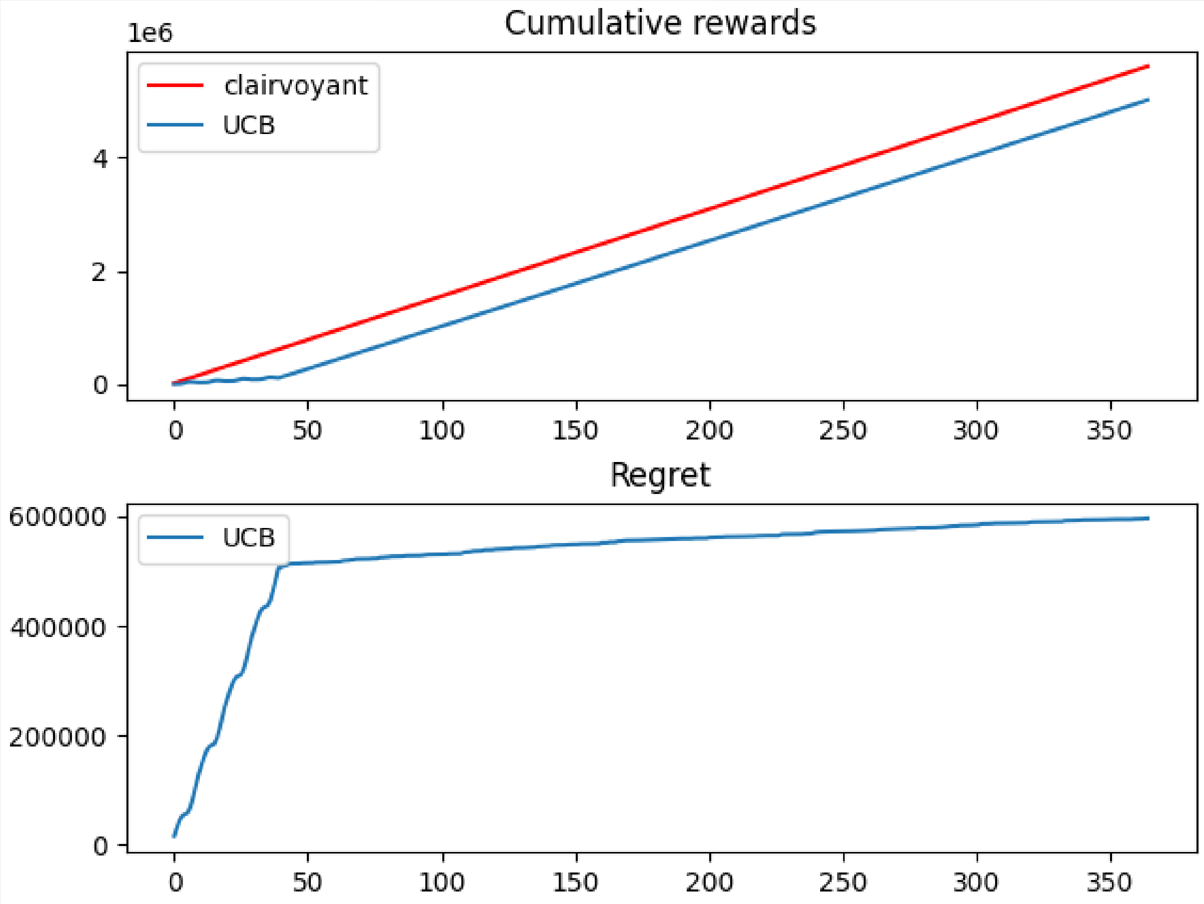
- Given the merge, all the contexts will run exactly like a small step 6. Indeed we took the very same code from step 6 as after the merge we have aggregated data which matches the features of the context.
- We still choose the arms in two steps.
- There still is the safety constraint.
- Every context provides its best choice and the learner will pack all the choices in a strategy.
- Also our UCB implementation resembles the one of step 6.

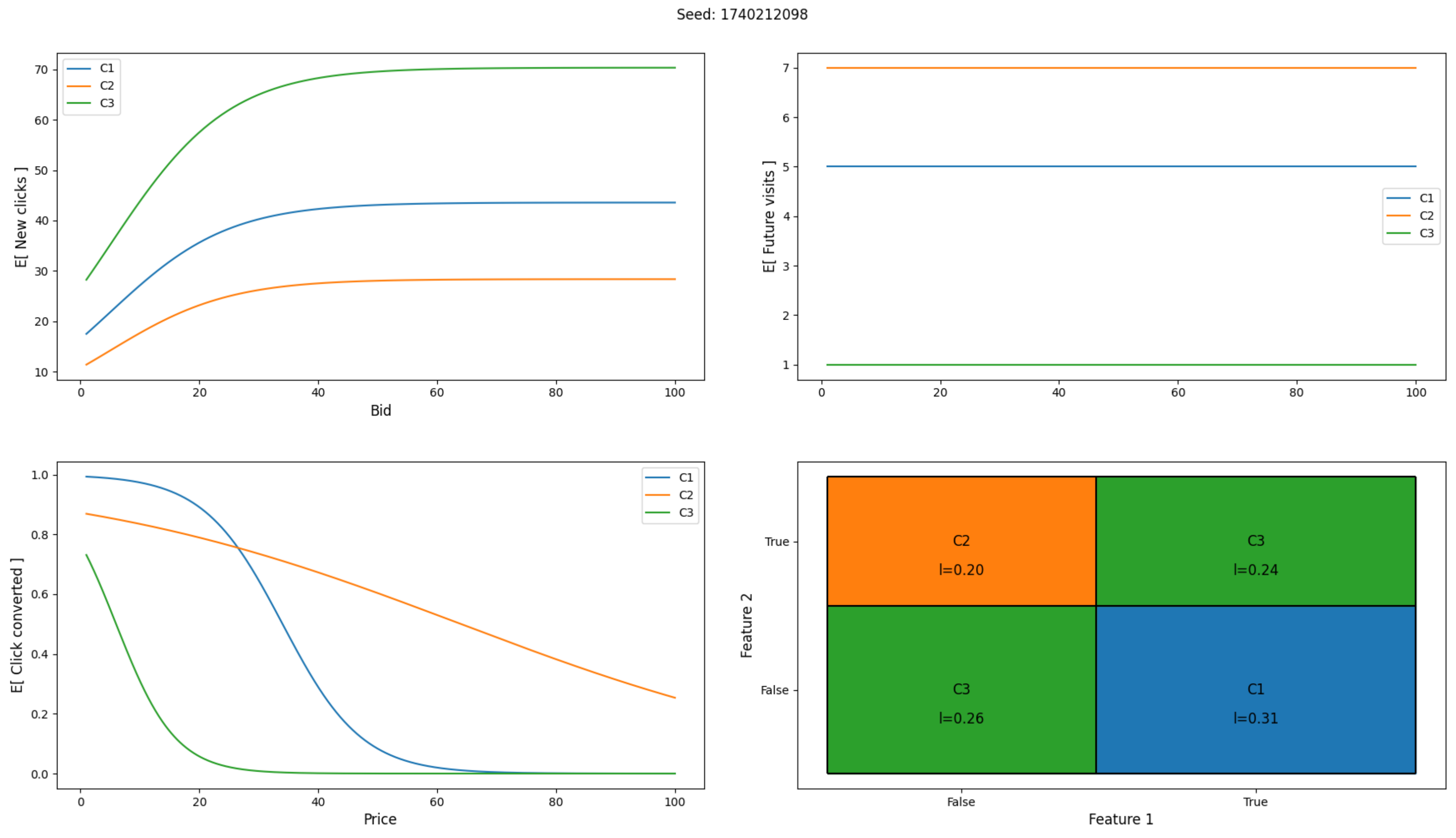
Results

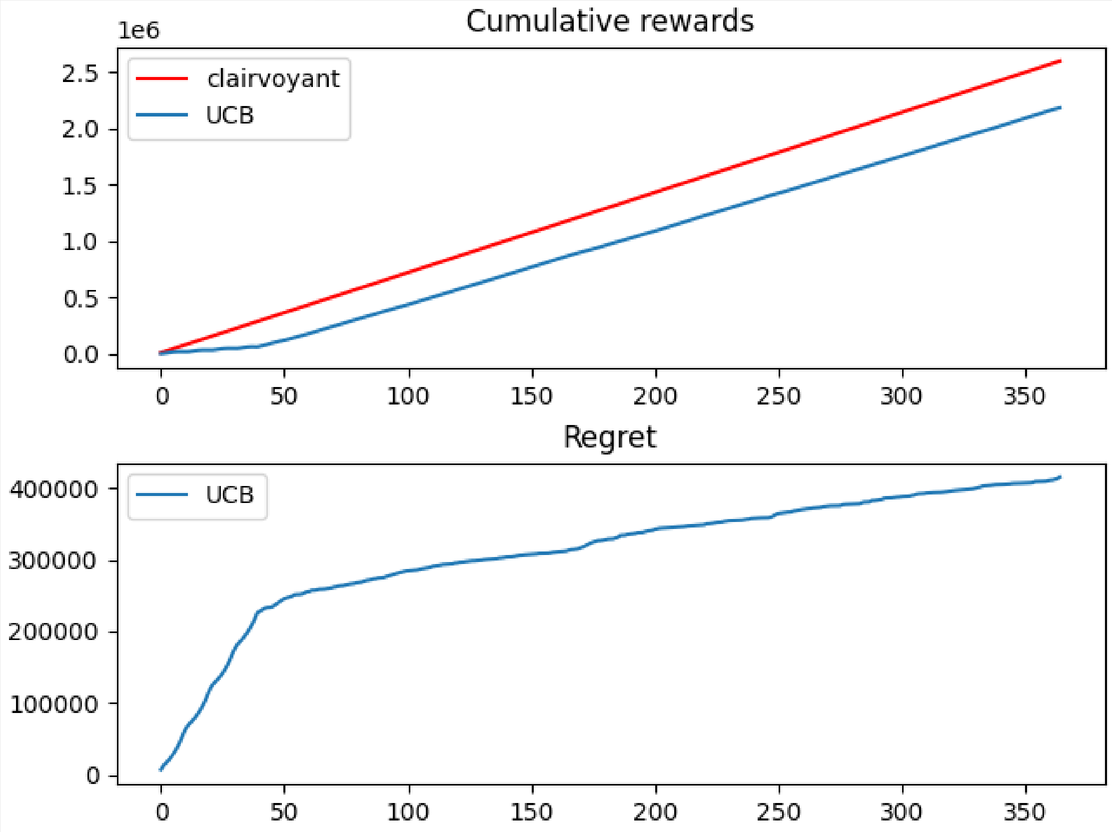
- As expected, the results are very similar to the previous steps.

- The overall regret depends also on the results of the step 4 that provided the contexts.


Seed: 3511939391
E[New clicks]
Bid
C1
C2
C3
E[Future visits]
E[Click converted]
Price
Feature 2
Feature 1
True
False
C2
l=0.18
C3
l=0.24
C1
l=0.25
C1
l=0.33


Cumulative rewards
1e6
clairvoyant
UCB
0
2
4
0
50
100
150
200
250
300
350
Regret
UCB
0
200000
400000
600000
0
50
100
150
200
250
300
350


Seed: 1740212098
E[New clicks]
Bid
C1
C2
C3
E[Future visits]
C1
C2
C3
E[Click converted]
Price
C1
C2
C3
Feature 2
Feature 1
True
False
C2
l=0.20
C3
l=0.24
C3
l=0.26
C1
l=0.31



Cumulative rewards
1e6
2.5
2.0
1.5
1.0
0.5
0.0
clairvoyant
UCB
0
50
100
150
200
250
300
350
Regret
400000
300000
200000
100000
0
UCB
0
50
100
150
200
250
300
350

The end!