Data Science for e-commerce

Images classification & recommender system at Veepee.

Veepee

Thibault ALLART
Senior Data Scientist, PhD @ **Veepee**
Teaching AI for e-commerce at Telecom ParisTech

Mohamed Amine ZGHAL
Data Scientist @ **Veepee**
Teaching Streaming algorithms at Telecom SudParis

Summary

01

Image classification at Veepee. A gold mine of 40M labelled images

02

Real time Recommender system using triplet loss siamese networks

03

Open questions on recommender systems

3

Part 01

Image classification

How to automate image classifications?

Context

At Veepee we sell between 10 000 to 80 000 new products per week.
Most of them are shoot in Veepee studio (one of the biggest in Europe).
For each product we takes multiple views.




Product description

Coloris

Blanc

Description

Top

Manches courtes

Col rond

Fermeture boutonnée au dos

Guipure

Volanté sur les manches

Effet de superposition

Composition

76% coton et 24% polyamide

Doublure : 95% coton et 5% élasthanne

Conseils d'entretien

Lavage délicat en machine à 30°C

Data sample

Product images




Text description

“We definitely fall in love with this pretty pair of women's ballerinas! This model accompanies you in all circumstances with comfort and femininity. We love its polished look with metallic nails for a rock'n roll look! ...”

Product description

103 (number of labels)

	Type of product	Lifestage	Gender	Color	Type of collar	Tip type	...
Classes	*Tee shirt*	*Adult*	*Woman*	*Pink*	*Straight collar*	*Round toe*	...
	Pant	*Children*	*Man*	*Red*	*Club collar*	...	...
	Ballerinas	*Both*	*Unisex*	*Salmon*	...	...	...
	...	...	...	...	...	...	...

Between 3 and 2000

Losses

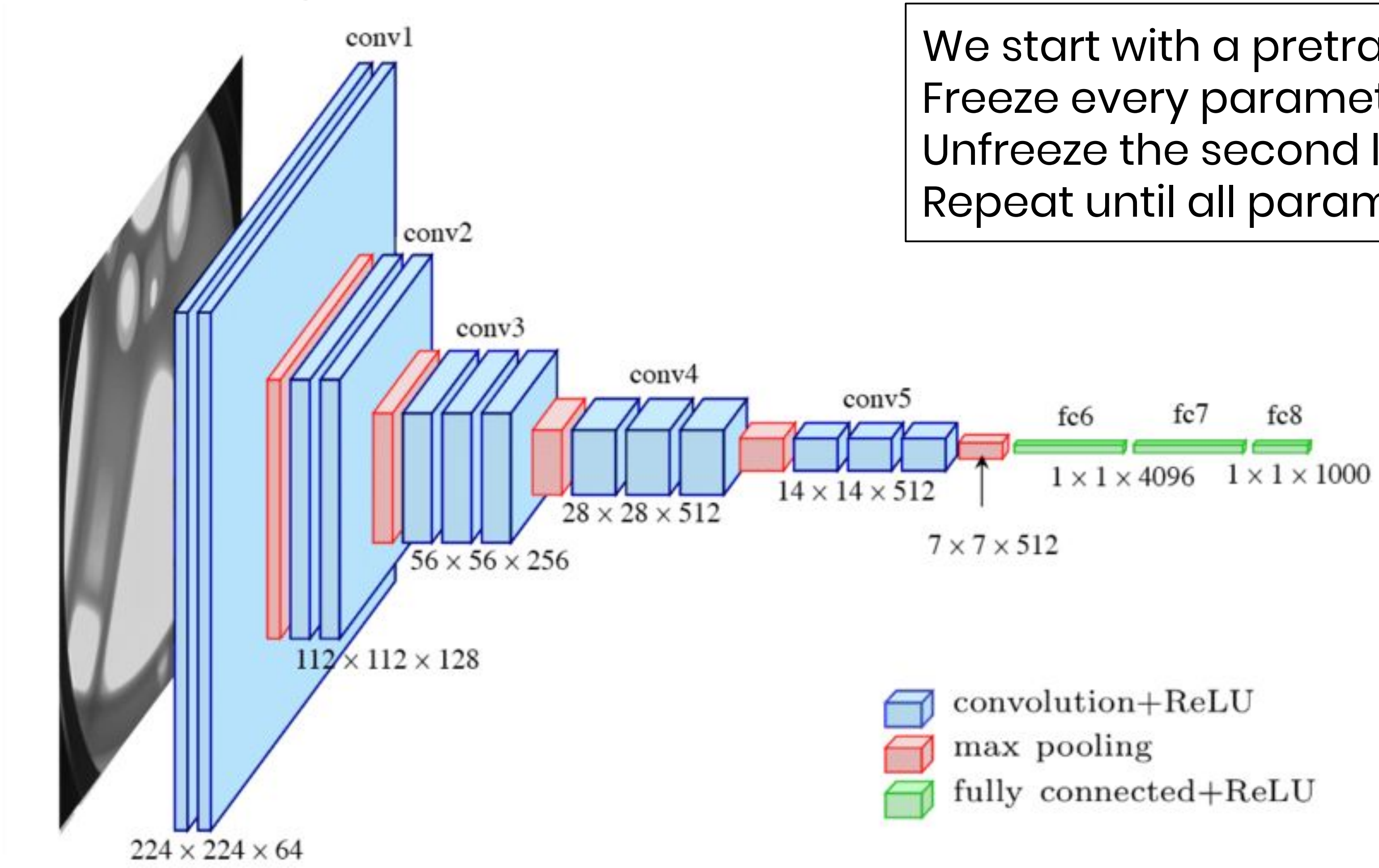
Binary Classification

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$$

Multi-class classification

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{c,i}\log(\hat{y}_{c,i})$$

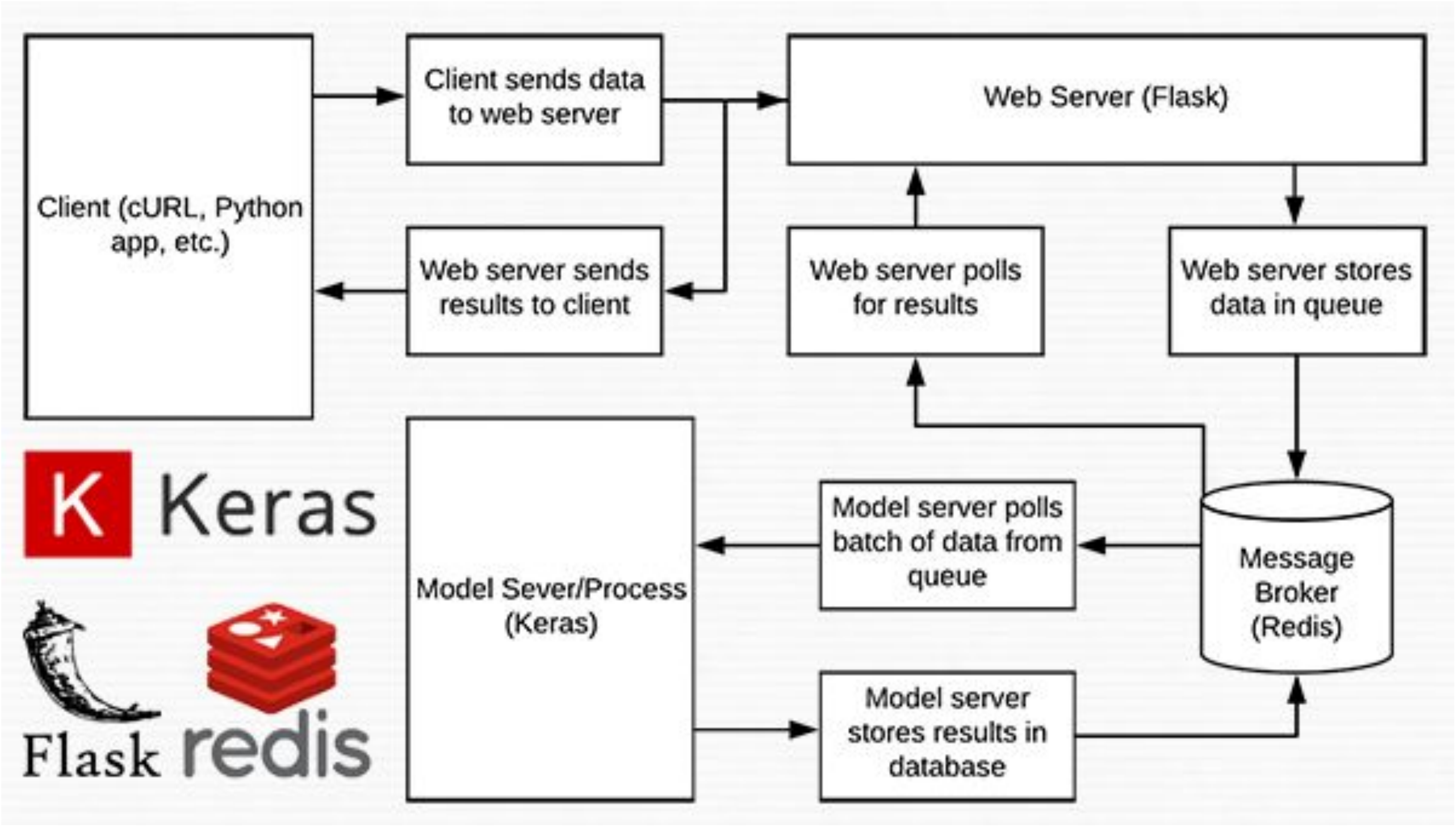
Training



We start with a pretrained network.
Freeze every parameters except le last layer —> Train.
Unfreeze the second last layer —> Train.
Repeat until all parameters have been trained.

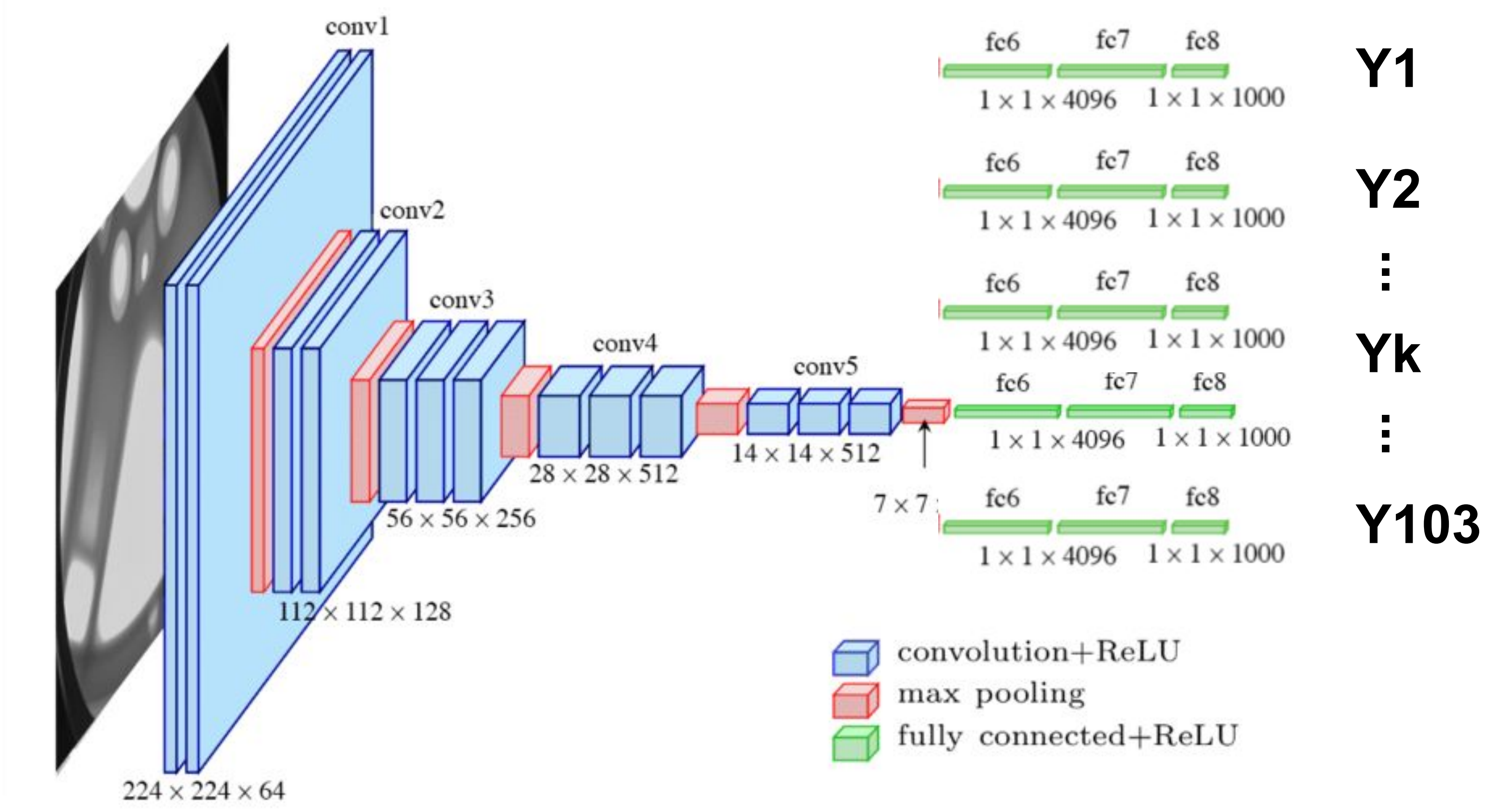
← This is VGG.
We use ResNet50.

Serving



We use a kind of similar architecture but on GCP and with some optimizations.

One model to rule them all



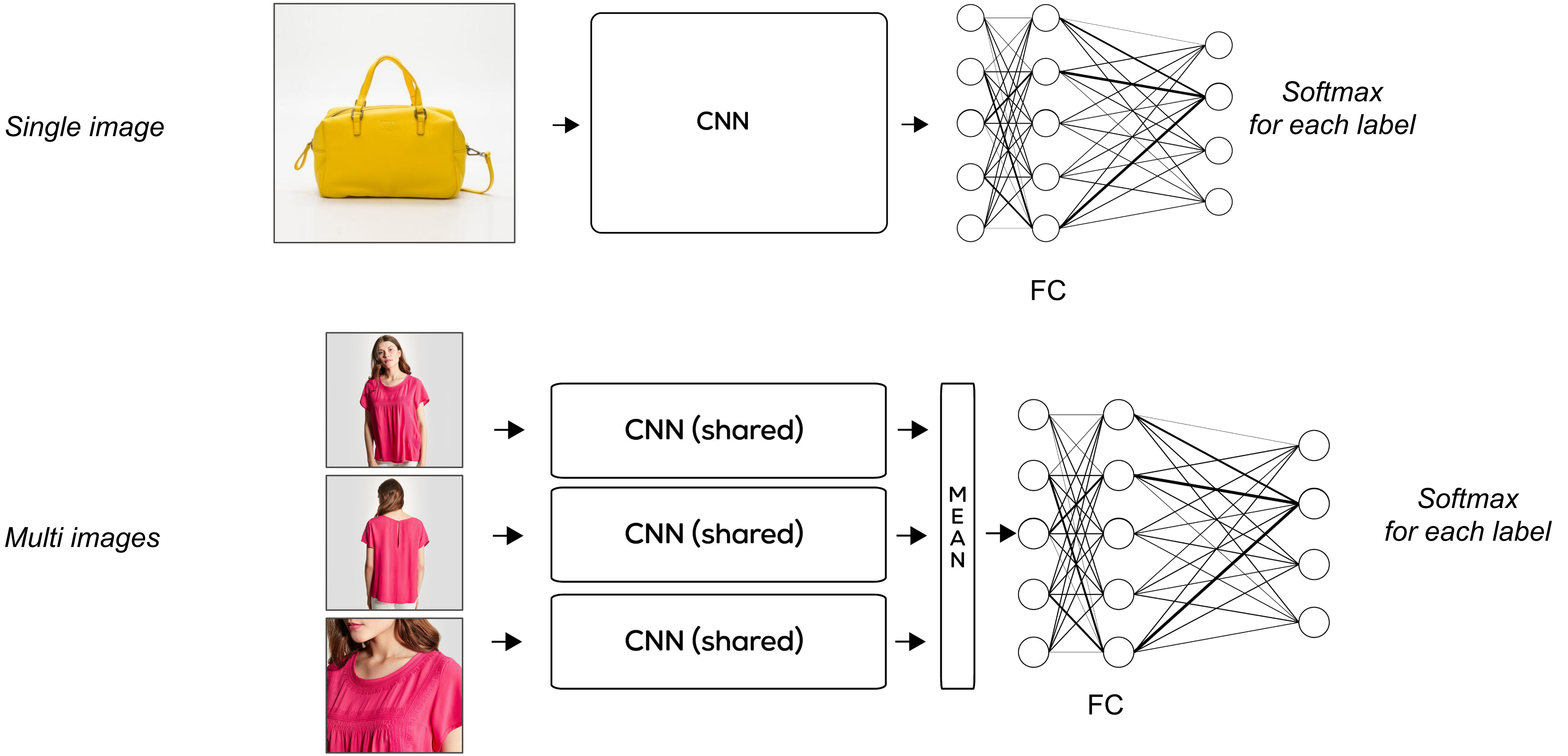
Multi-task classifier loss

Multi-task and multi-class classification

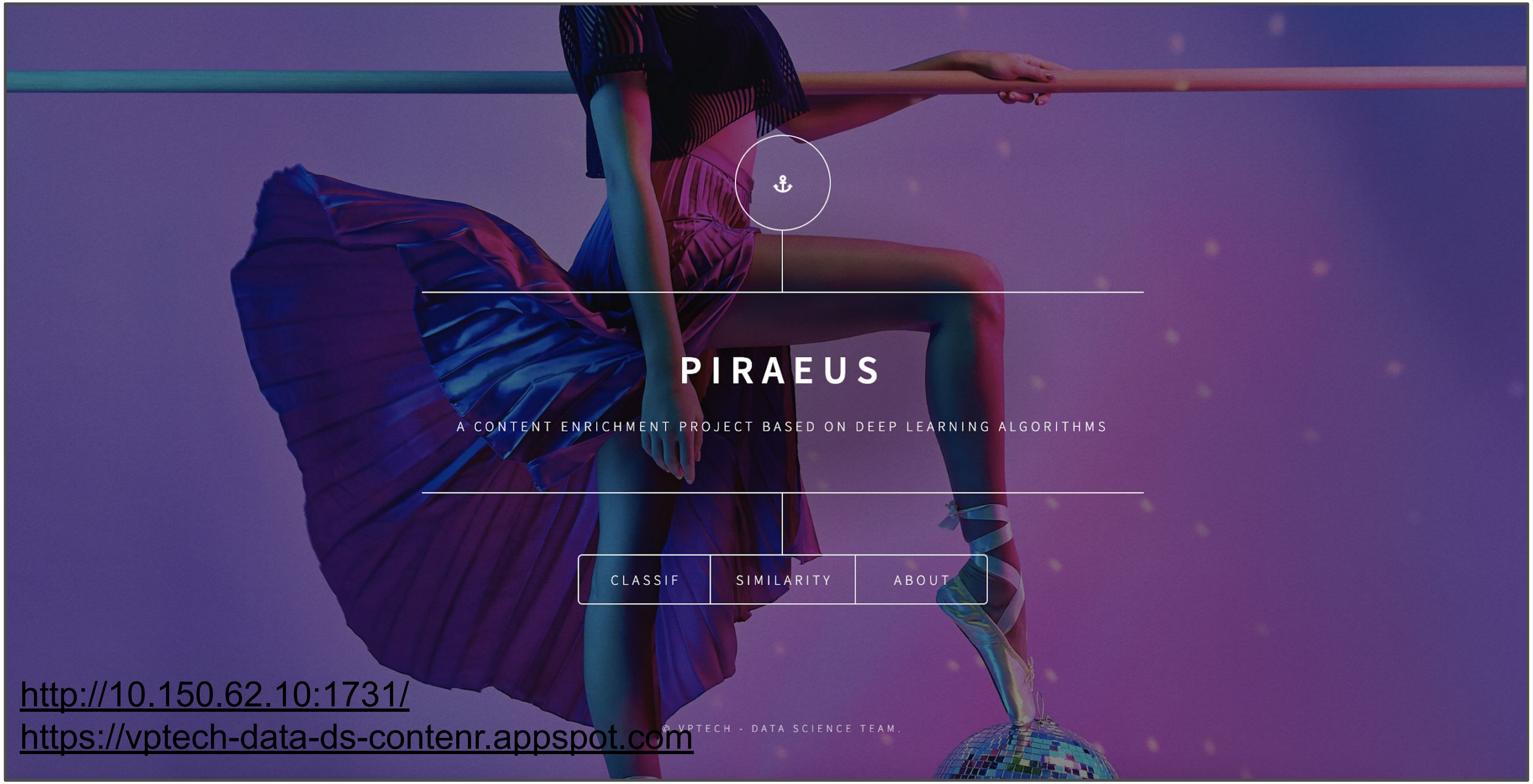
$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{c=1}^{C} w_k \cdot y_{k,c,i} \log(\hat{y}_{k,c,i})$$

The weight here allow to deal with task imbalanced number of classes.

Multi-input network



Demo API



http://10.150.62.10:1731/

https://vptech-data-ds-contenr.appspot.com

Part 02

Recommender system

Home page

How to Place the best sales for each user?

Context

- Revenue 3 billion € per year.

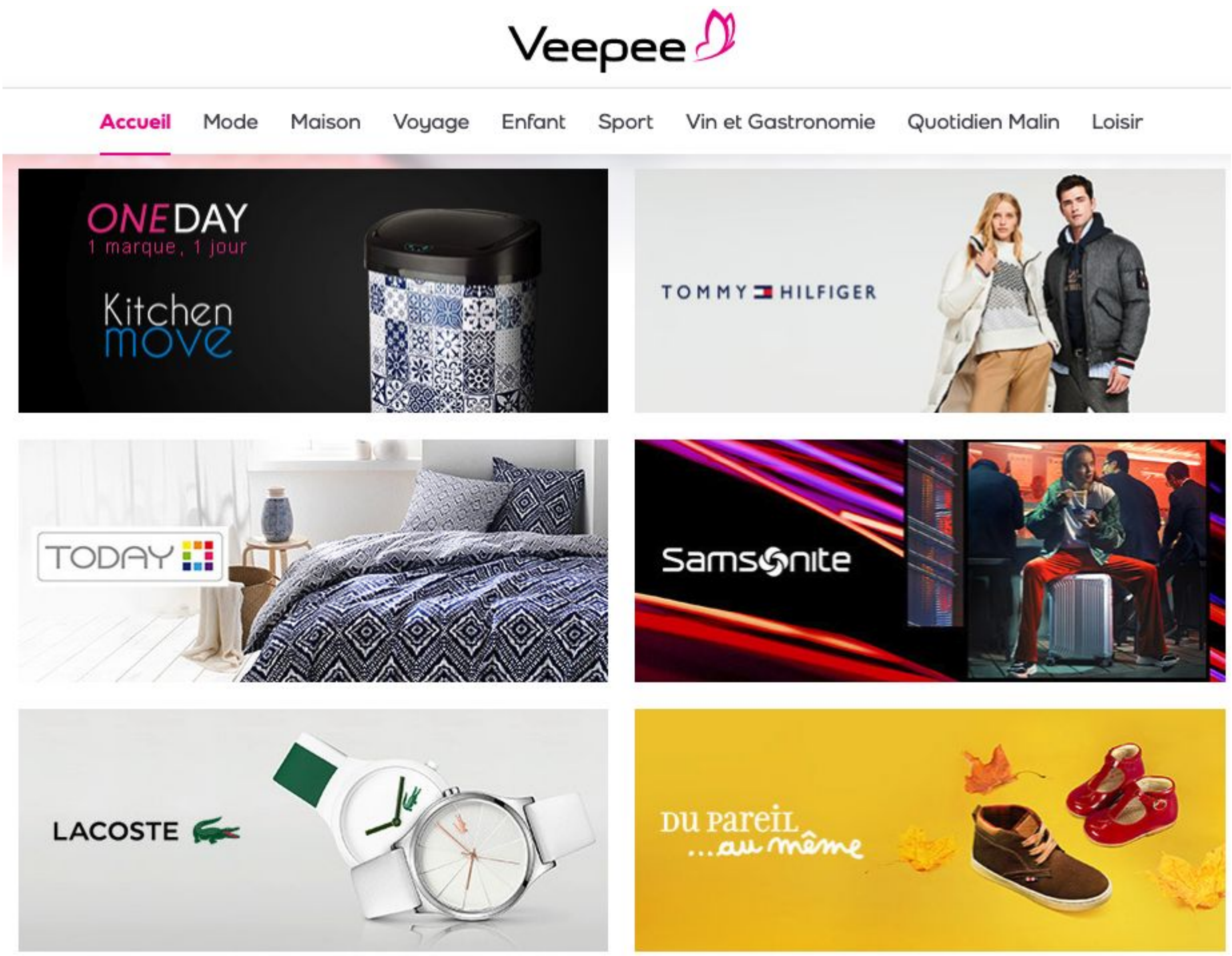
Veepee website has a hierarchical structure.

Brand —> Universe —> Product

We'll present the home page (brand level) recommender system.

There is around 250 brands at the same time.

Sales last between 24h and 15 days.



Goal

Business

Ease users navigation by showing them brands they are more likely to purchase on top.

Data Science

Every time a user connects to the home page, we want to display available brands, in a way that **maximizes the conversion rate**

Service Level Agreement

- 50ms response time
- Peaks up to 150 requests per second

Formalism

Lets note $\mathbf{U}$ and $\mathbf{I}$ ($\subset \mathbb{N}$) the sets of all users and items (past, present and future).

At time t :

- A user u^t connect on the website.
- There is N^t items available for sales denoted by $\{i_1^t, i_2^t, \ldots, i_{N^t}^t\}$.
- We observe a list of p-dimensinal features for every (user, item) pairs that possibly depend on time t, denoted by $[X_{u,1}^t, X_{u,2}^t, \cdots, X_{u,N^t}^t]$.
- The recommender system returns an ordered set of items s^t to user u^t denoted by $[i_{\phi(1)}^t, i_{\phi(2)}^t, \ldots, i_{\phi(N^t)}^t]$ where ϕ is a permutation of $[\![1, N^t]\!]$.

At time $t + \delta_t$ we observe a list of rewards $[r_1, r_2, \cdots, r_{N^t}]$.

Removing list interaction

First assumption: Item reward is not influenced by side display (but eventually by its position).

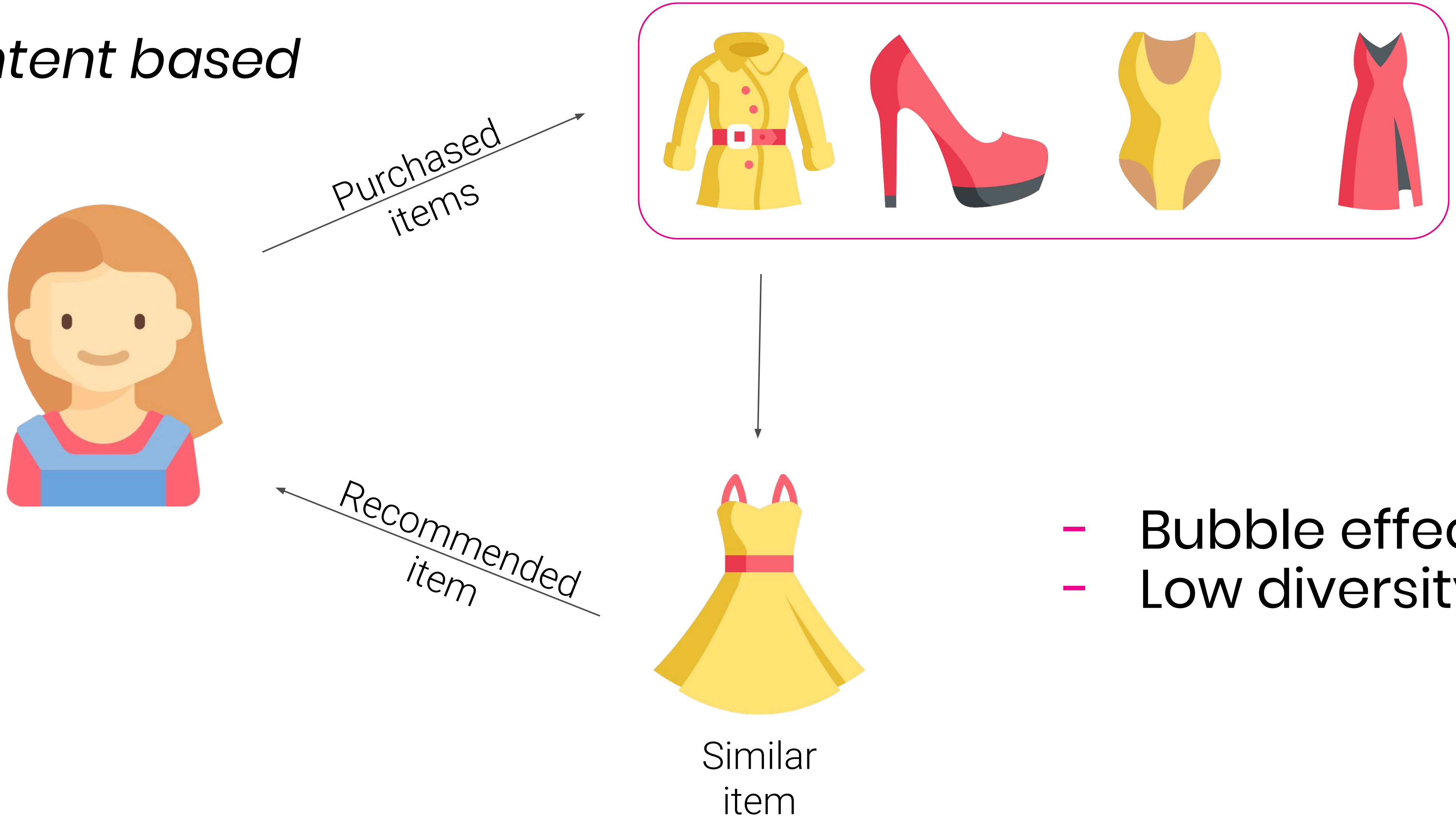
$$p(R_k|[i_1, \cdots, i_k, \cdots, i_N]) = p(R_k|[i_k], k, N) \tag{1}$$

Considering all permutation in a list of size N would require $N!$ cases.

Thanks to this assumption the problem is reduced to estimating N rewards and rank the items accordingly.

Classic approaches

Content based

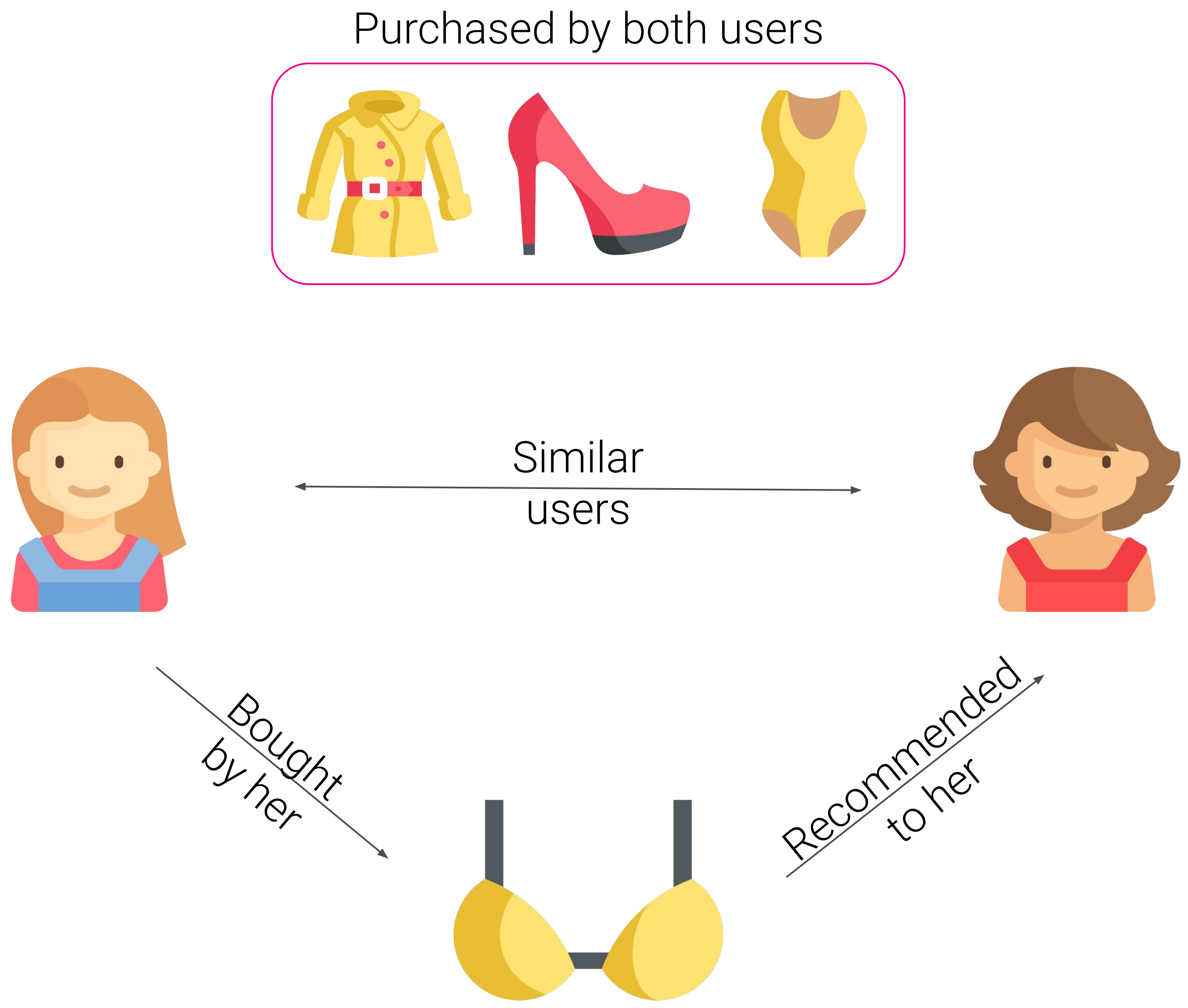


- Bubble effect
- Low diversity

Classic approaches

Collaborative filtering

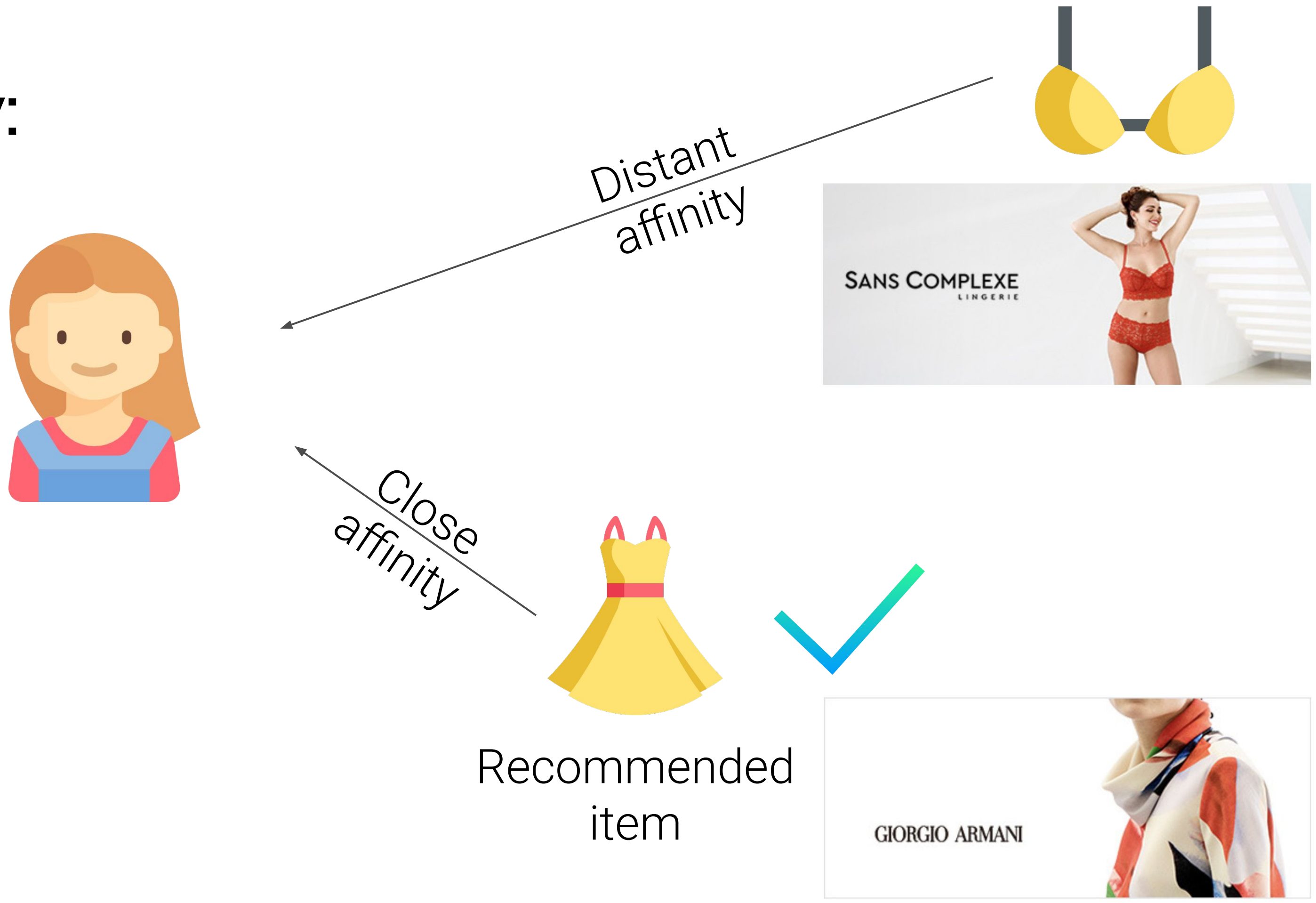
- “Cold start”
- Lack of context



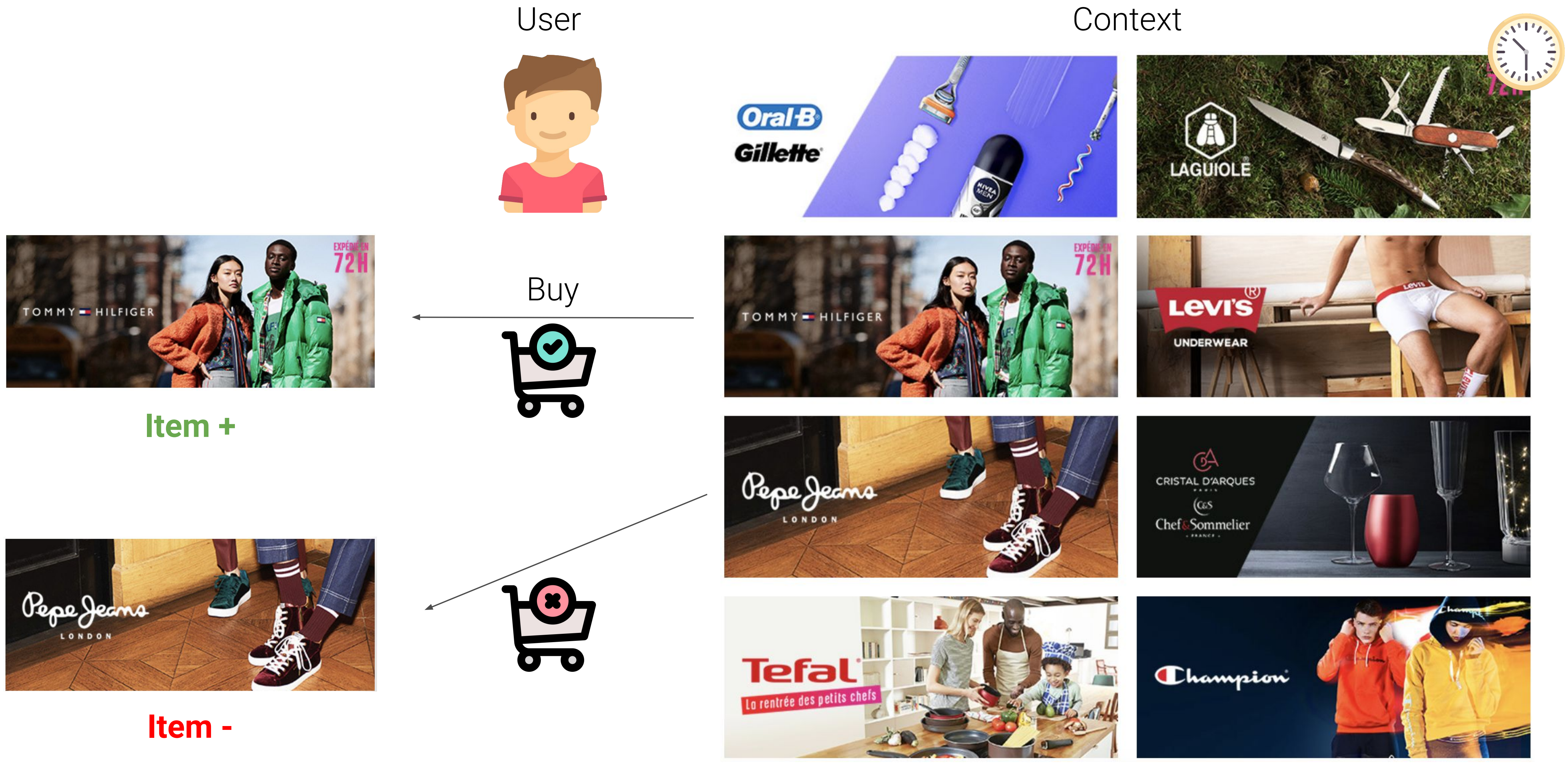
Our approach - Hybrid

Recommendations driven by:

- User description
- User habits
- Item description
- Item performance
- Context



Our approach

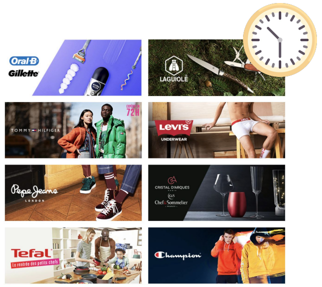


Our model

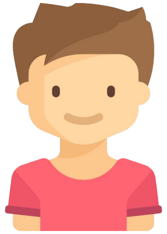
Train

- Siamese neural network
- Triplet loss

Item +

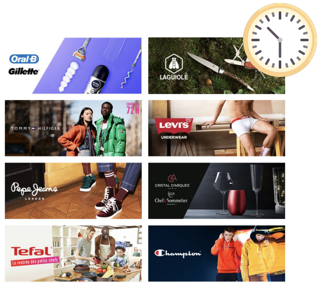


Same context

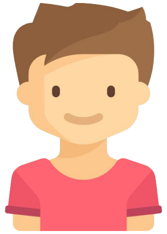


Same user

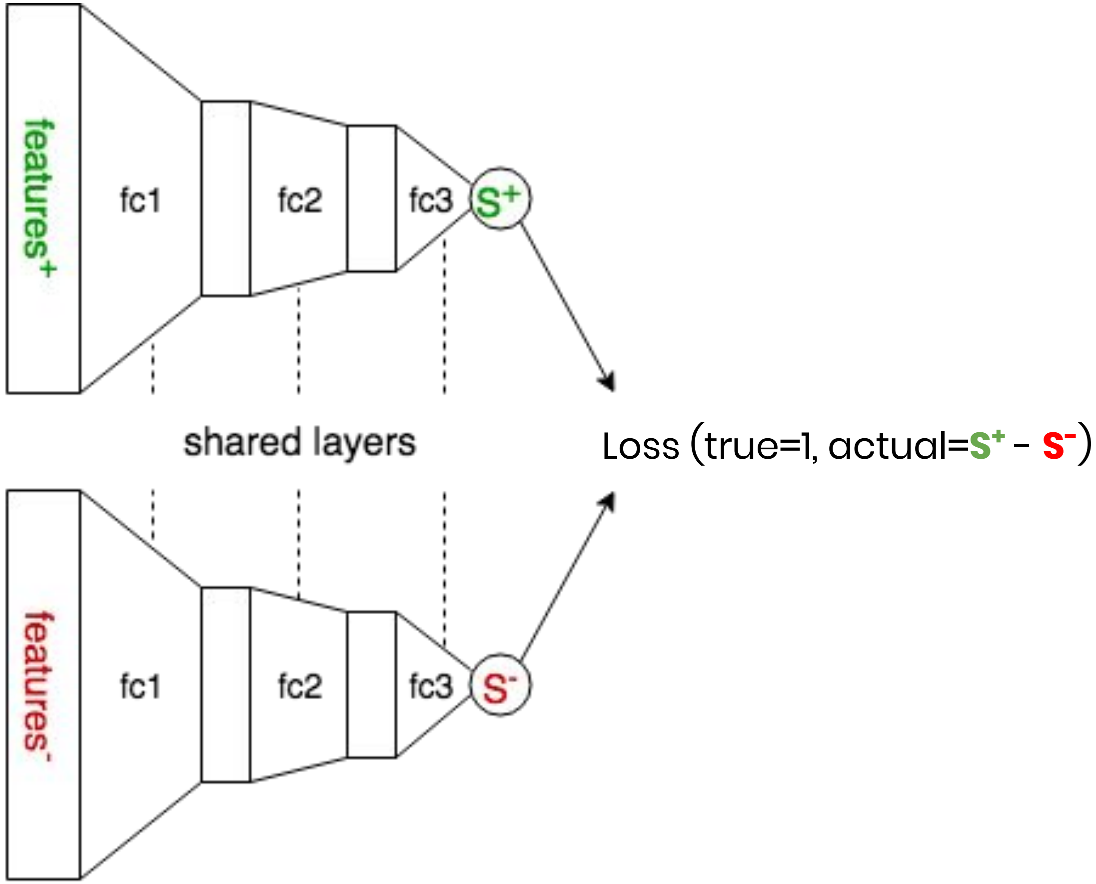
Item -



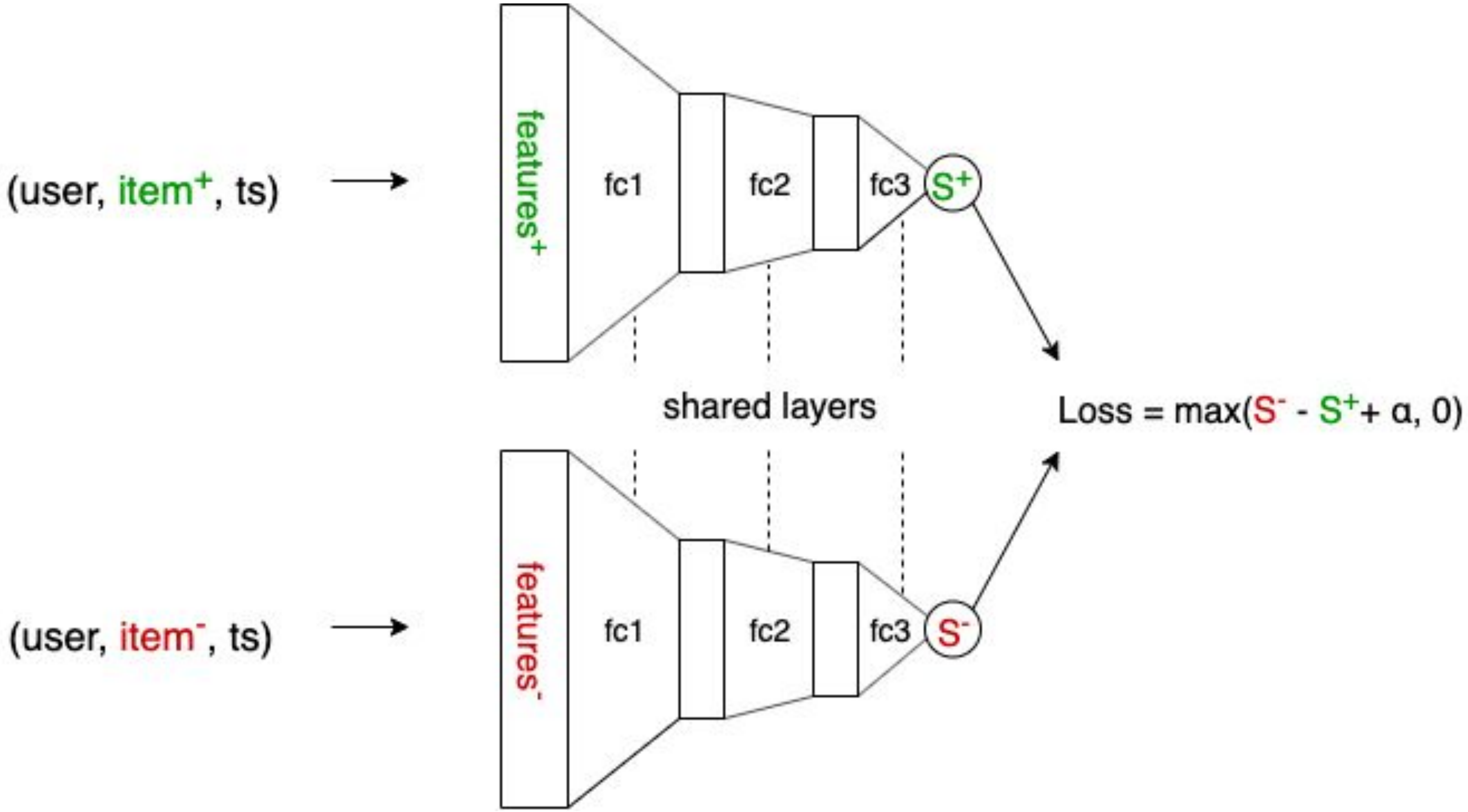
Same context



Same user

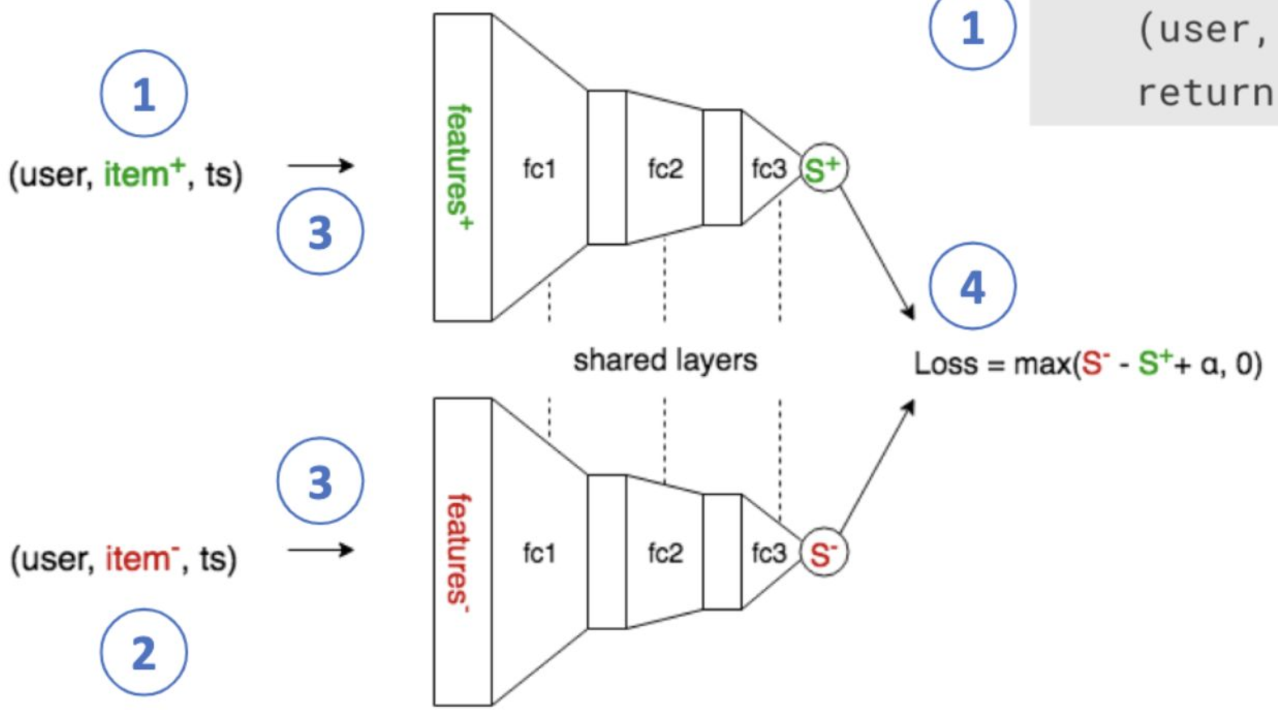


Model



A good tutorial by Charles Ollion and Olivier Grisel
https://github.com/m2dsupsdlclass/lectures-labs

Training



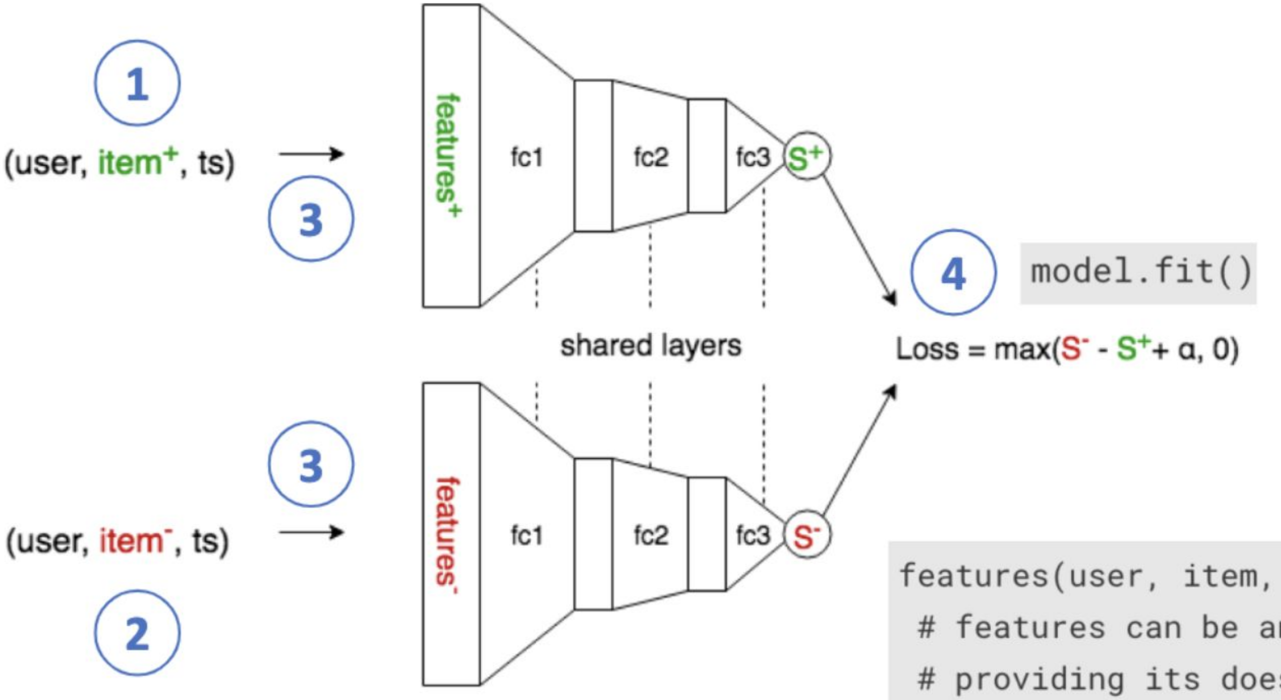
1

```
generate_positive_sample(purchase_table):
    (user, item+, timestamp) = select a random purchase
    return (user, item+, timestamp)
```

2

```
generate_negative_sample(timestamp, item+, item_table):
    candidates = items at sales at timestamp remove item+ from candidates
    item- = select a random candiate return item-
```

Training



3

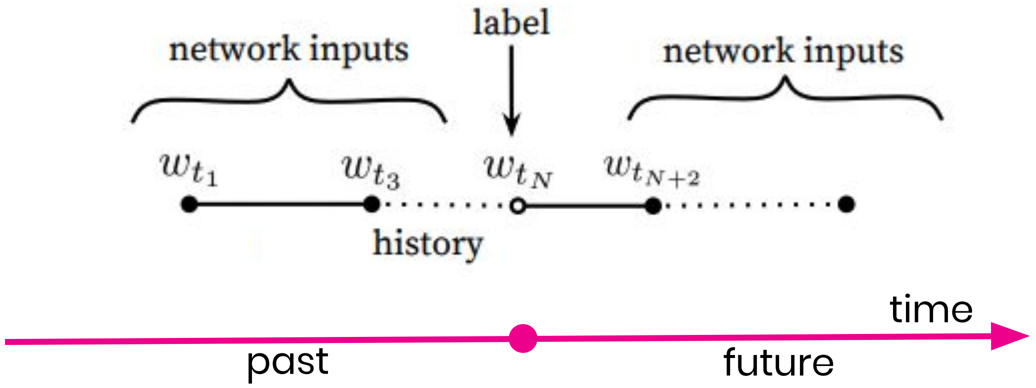
```
features(user, item, timestamp, tables)
 # features can be any function that depend on user, item and/or timestamp
 # providing its doesn't include information at time >= timestamp. i.e
    user_age
    item_brand
    item_ranking_in_the_last_hour
    number_of_time_user_bought_this_item_category

    return features
```

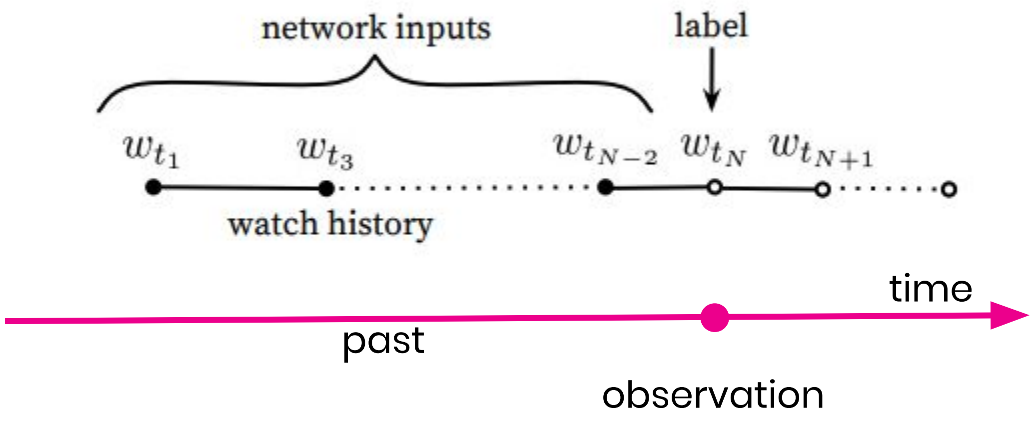
Time importance

4	3			5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

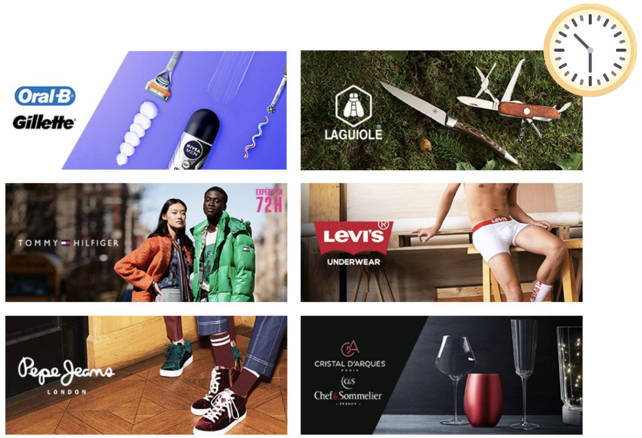
user_id	item_id	rating	timestamp
196	242	3	881250949
186	302	3	891717742
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596



observation



The features



Context

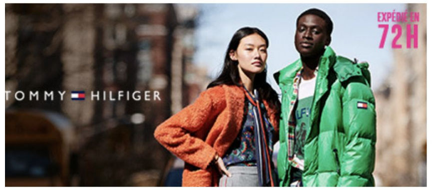
- Hour of the day
- Day of the week
- Day of year
- Section
- Minutes since sales opening
- etc.

User description

- Socio-professional group
- Age
- Gender
- etc.

User activity

- Purchased items
- Viewed items
- Days since last purchase
- Turnover by sector
- Purchase power
- etc.

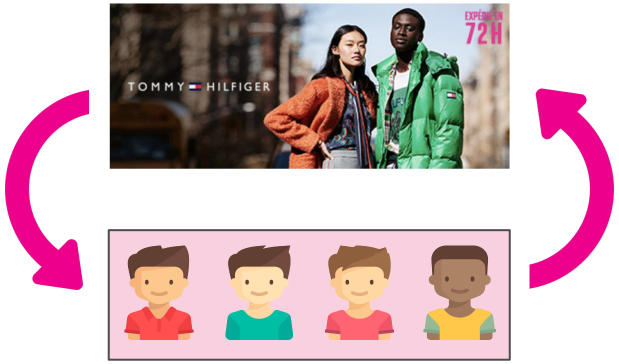


Campaign description

- Brand
- Business type
- Sector
- Sub-sector
- Duration
- Country
- etc.

Campaign performance

- Turnover - last 15 min.
- Conversion rate - last hour
- % of total purchases - last 4 hours
- Brand past performance
- etc.



Popularity by clusters

- Turnover - last 15 minutes - per demographic cluster
- % of total purchases - last hour - per demographic cluster
- etc.

Continuously updated

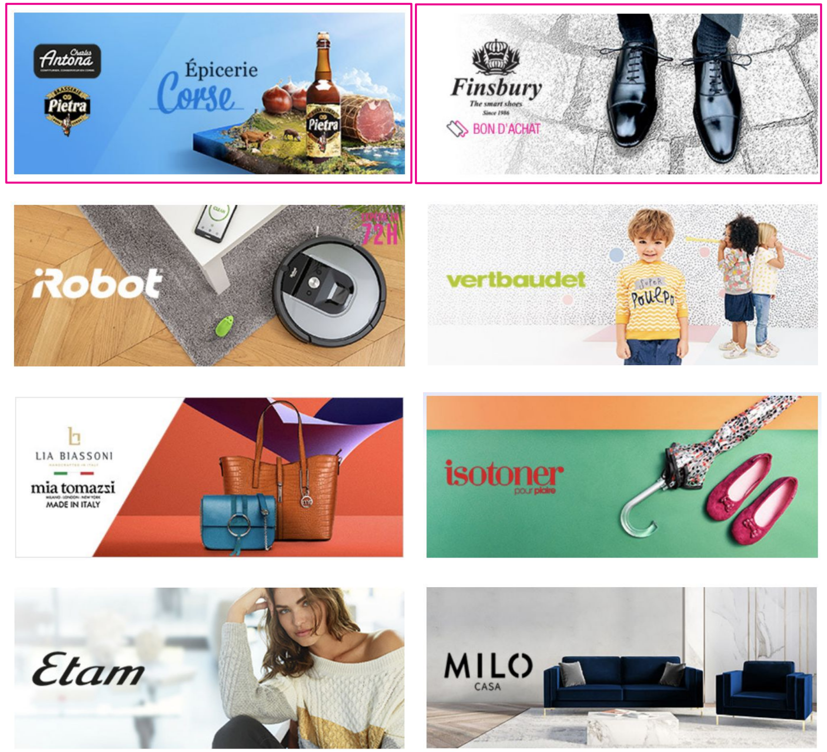
Updated once a day

Our model

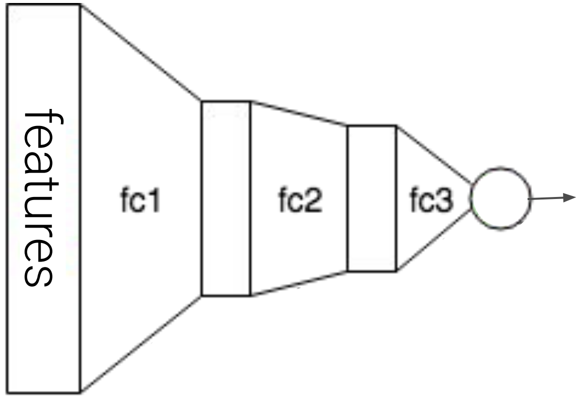
Predict

Context = Now

User



Available sales
(Not ordered)



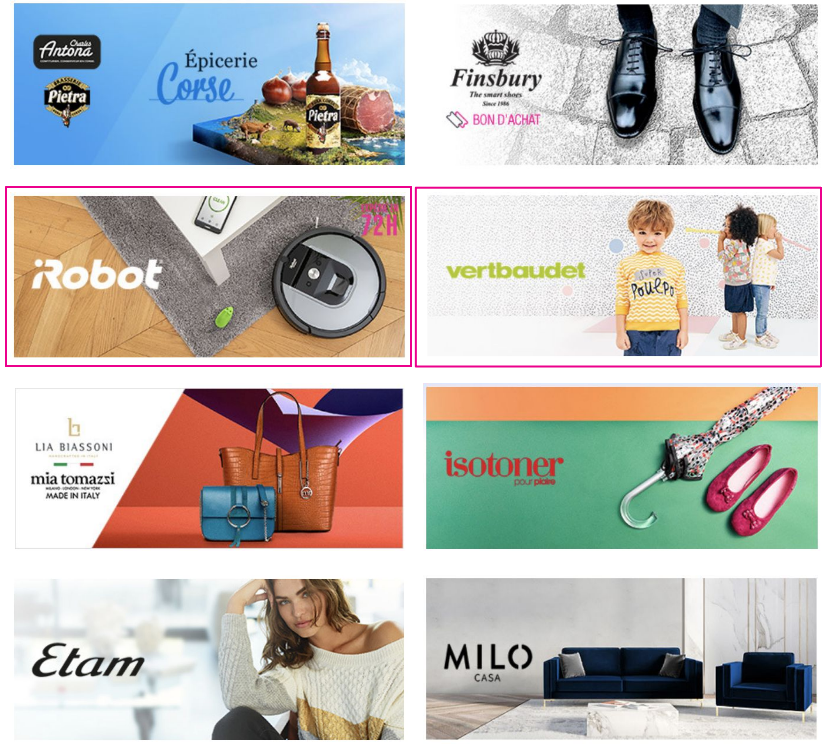
0.52

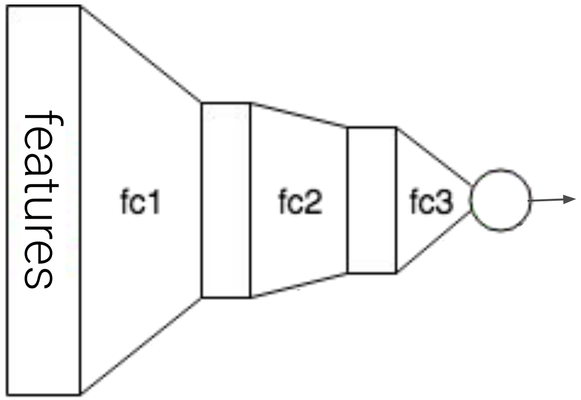
Our model

Predict

Context = Now

User





0.26

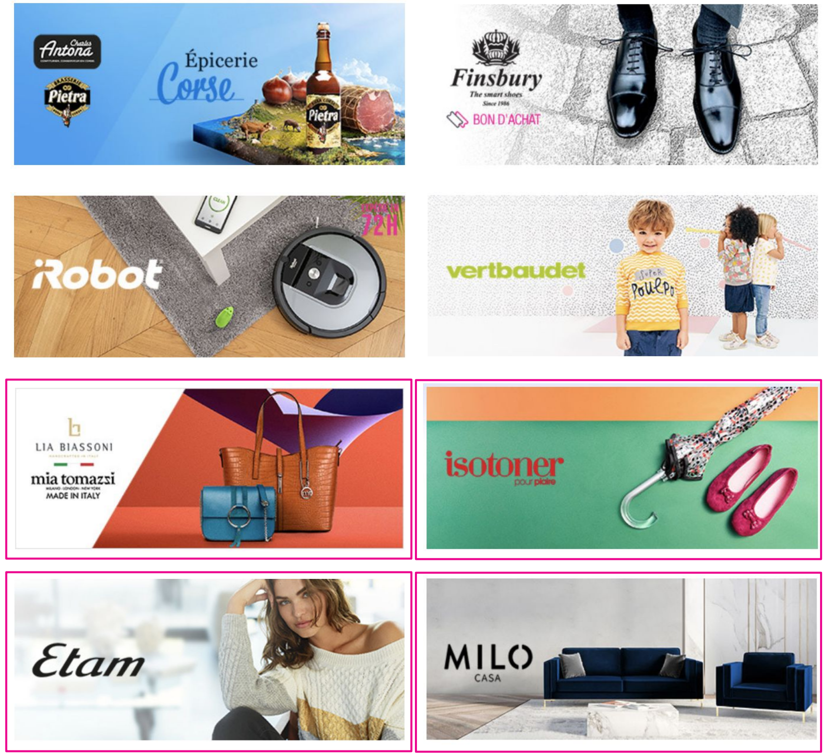
Available sales

Our model

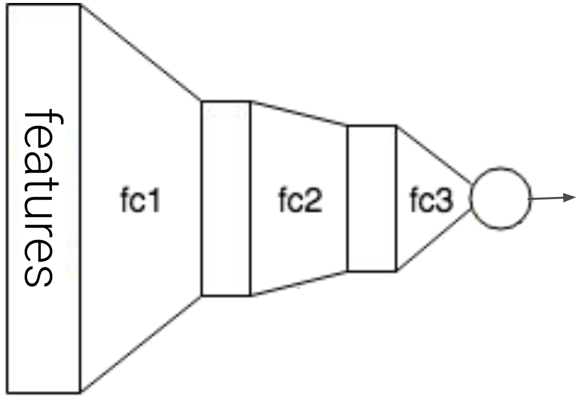
Predict

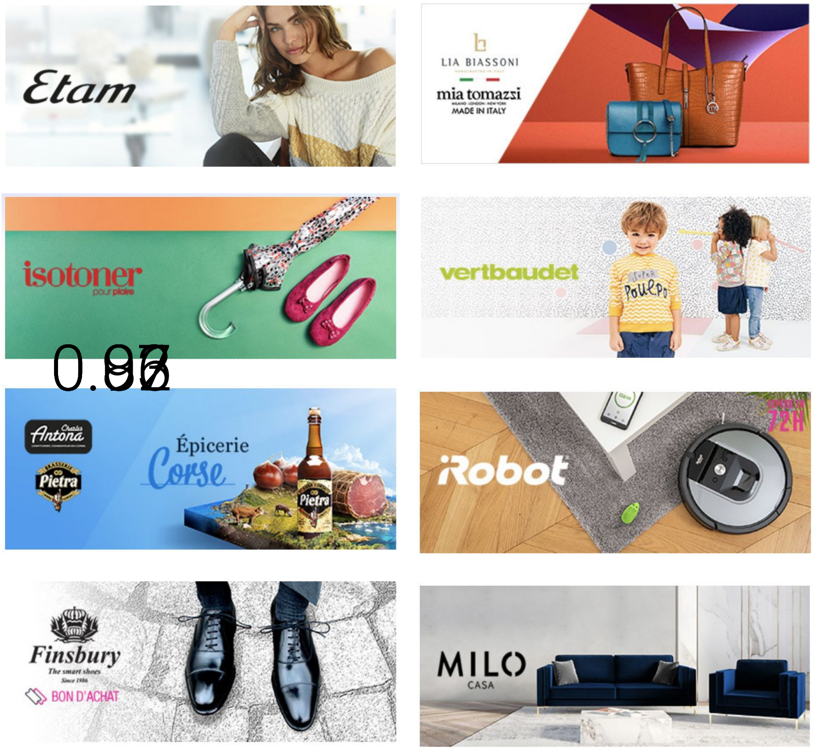
Context = Now 

User



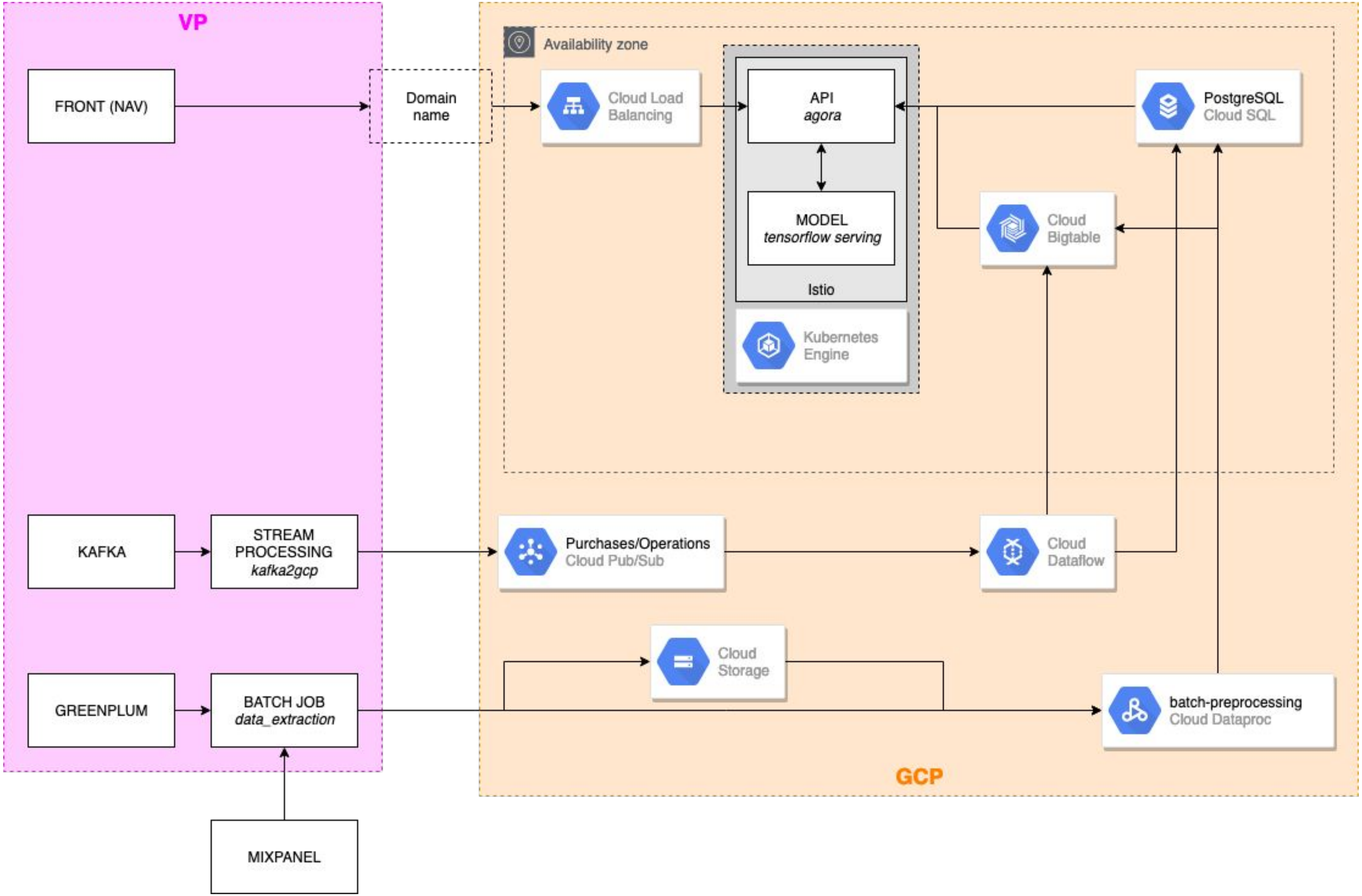
Available sales





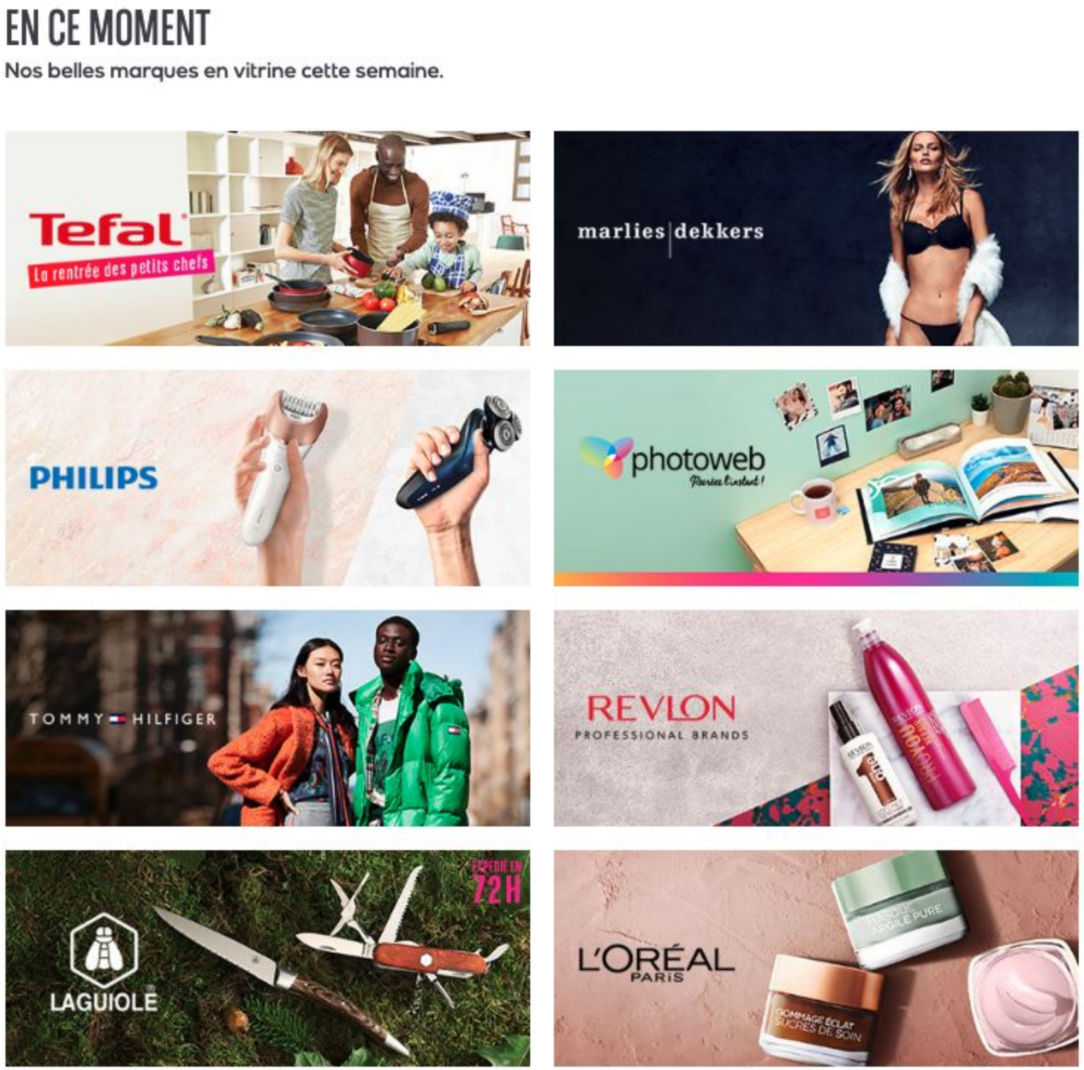
Personalized order

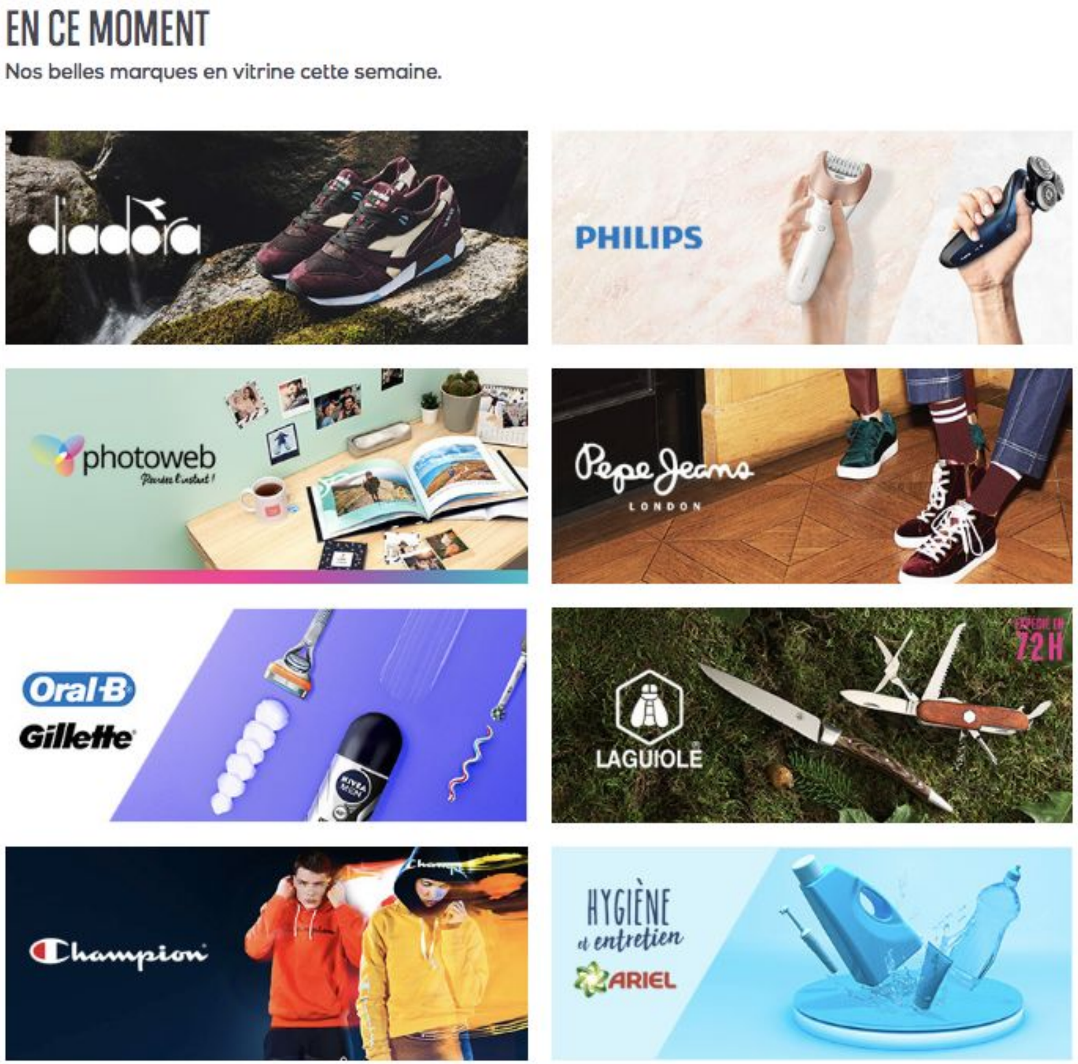
Architecture

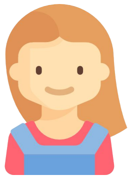


Demo

www.veepee.fr






Betty’s home page


Amine’s home page

Achievements

December 2018

V1

Members description, operation descriptions and operations popularity

+3 %*

5% - 20%**

April 2019

V2

Purchase history, migration to GCP

30%**

+4 %*

June 2019

V3

Demographic popularity, observations redefinition and new sales boost

+5 %*

50%**

July 2019

V4

View history, stop sampling

70%**

+6 %*

August 2019

New Sections A/B test

** Personalization Conversion Rate Vs. Merchandising version*

*** Traffic allocation for personalization*

**** Personalization turnover Vs. merchandising turnover for the same number of unique visitors over the period*

Achievements

New Sections
A/B test

September
2019

70%**

+9 %*

October
2019

V5

Performance history, loss
optimisation, delayed reward
and position de-bias.

Ongoing AB test

95%**

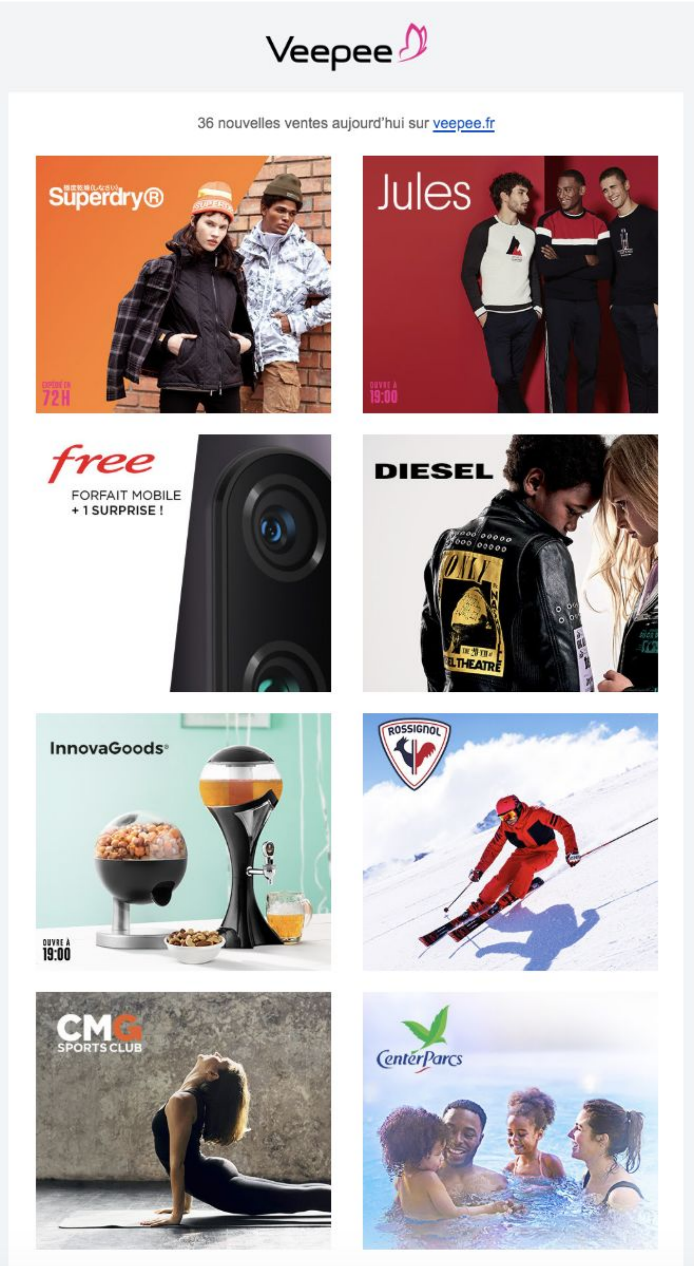
** Personalization Conversion Rate Vs. Merchandising version*

*** Traffic allocation for personalization*

Personalization is an essential growth driver

International home page personnalisation

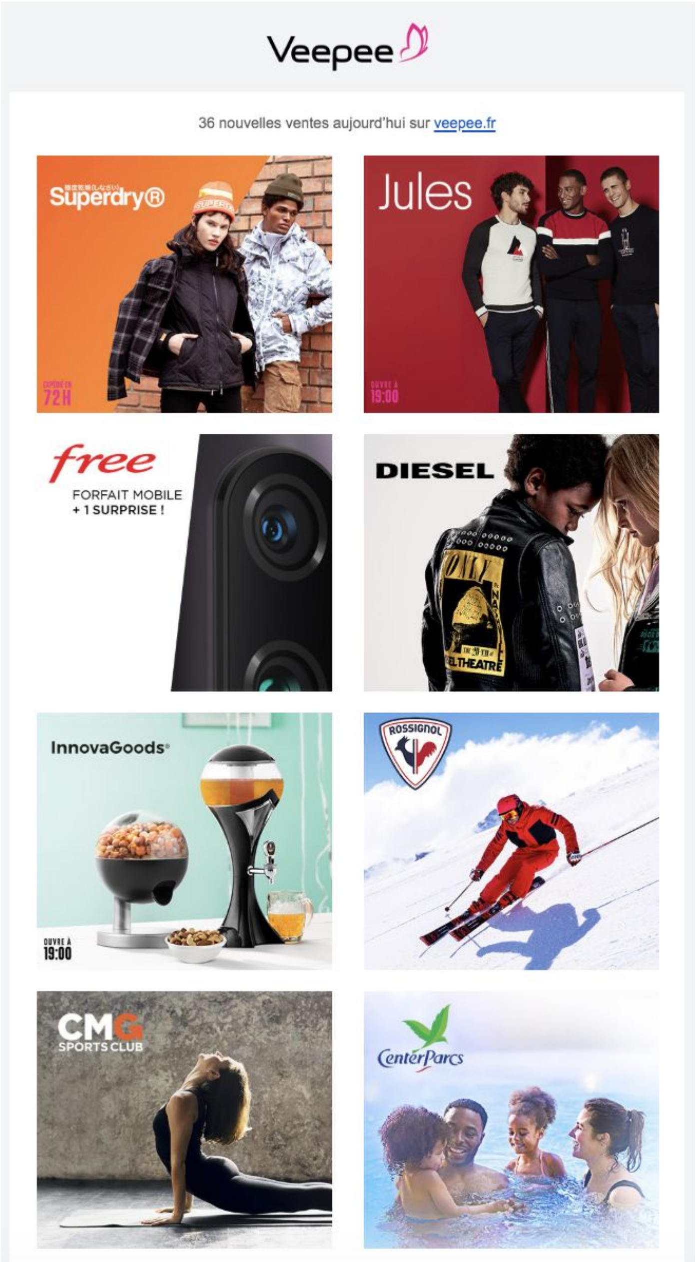
Email personalization



IInternships @ Veepee

Email personalization

mazghal@vente-privee.com

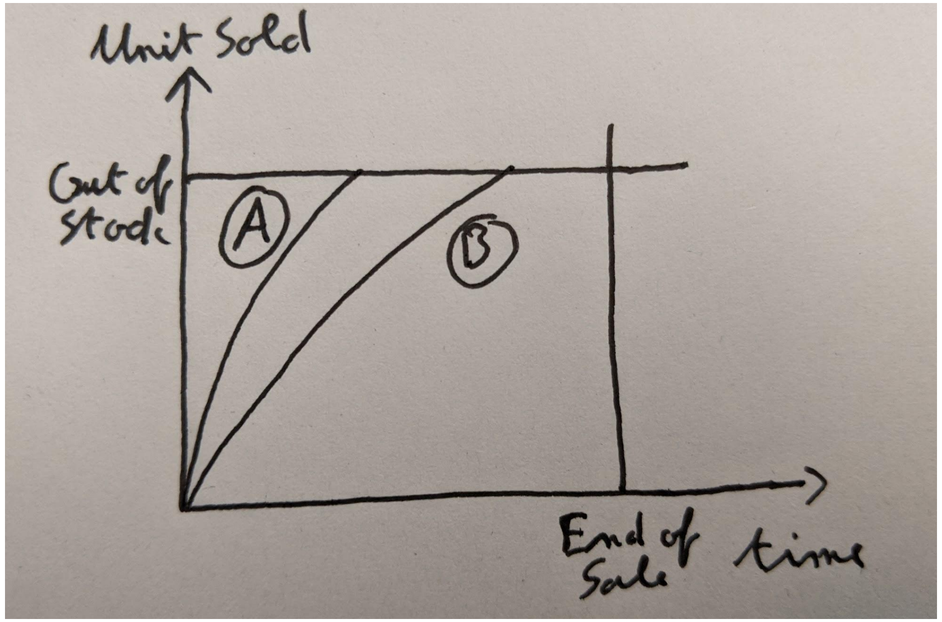


Contextual Bandit approach

Limited stock

Limited stock

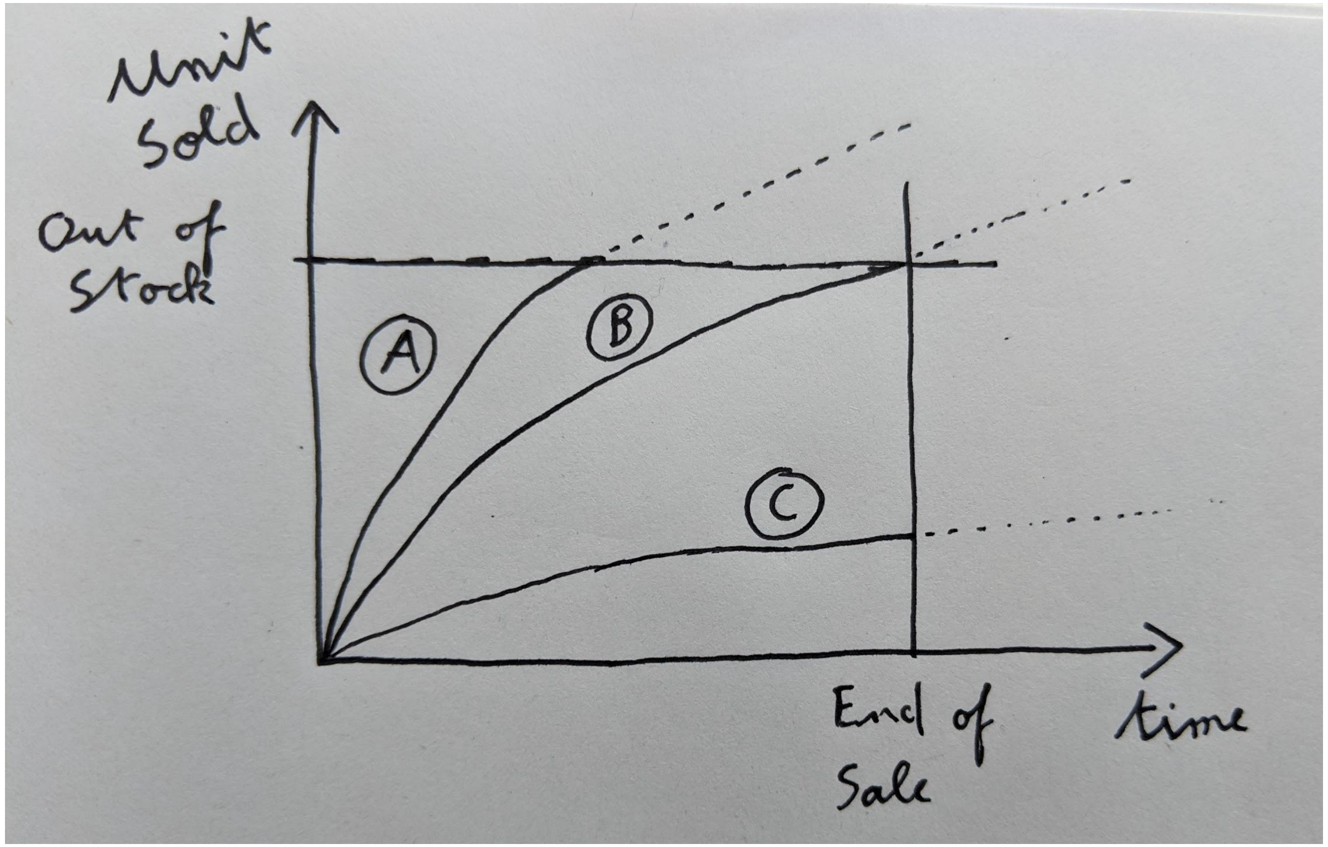
Under unlimited stock the optimal strategy is to put the best item on top.
But with stock constraint it can be the opposite.



E.g. If the stock is small compared to the demand the item will go out of stock whatever the position. In such case we should not spend our *energy* on putting this item on top but instead focus on other items.

Limited stock

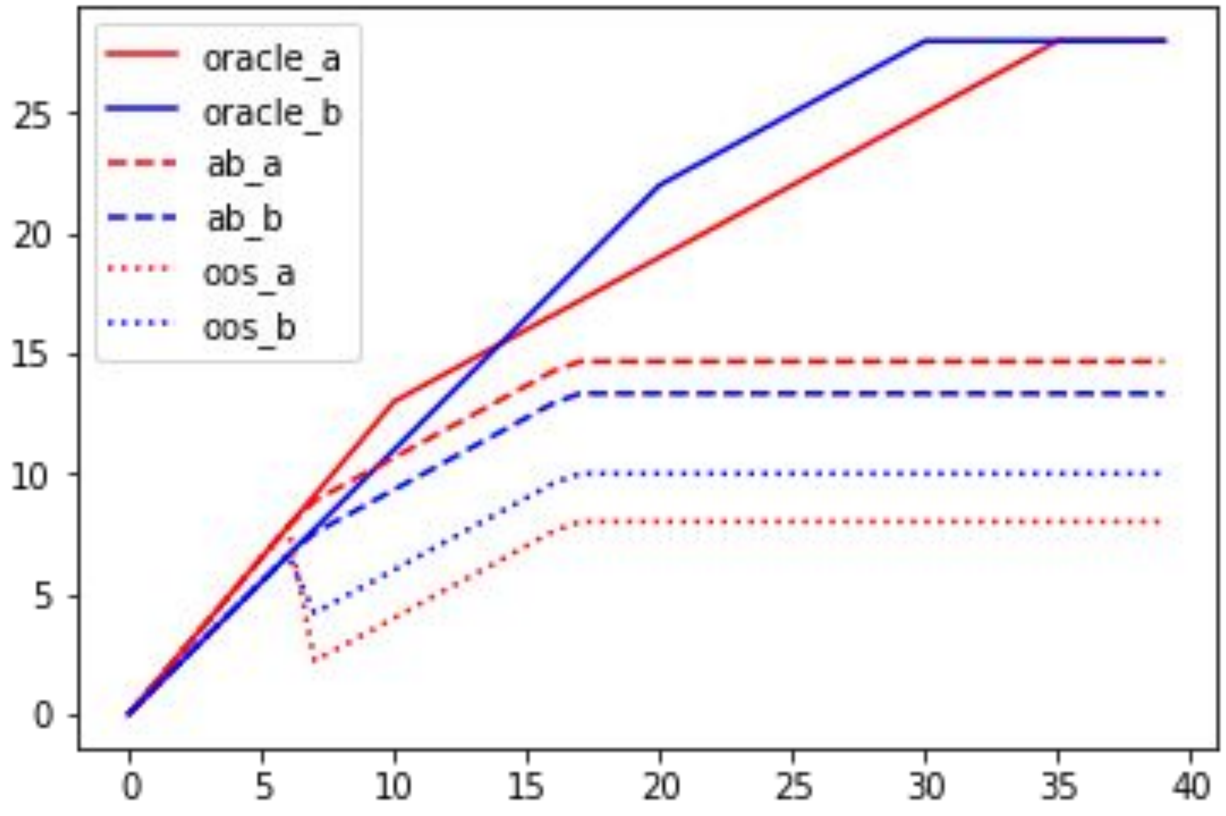
Recommender systems doesn’t take stock limitation into account leading to a loss of revenue.



Item display position:

A. On top
B. Middle
C. Bottom

How to do A/B-test with limited stock ?



- Splitting by users lead to wrong conclusion due to cannibalization.
- Split stocks is hard to implement and no consumer friendly.
- Splitting by sales lack of statistical power.


Thank you