# Data Science Game 2018

## Big Russian Loss (MIPT)

Konstantin Gavrilchik
Artur Fattakhov
Evgeny Kononenko
Vasily Ryazanov

# Championship

**Data Science Game** - annual international student data science championship

**Main rules:**

- Student only (maximum 2 PHDs)
- 1 team per university possible to qualified to final
- Maximum 5 team from 1 country
- No more than 20 teams in overall

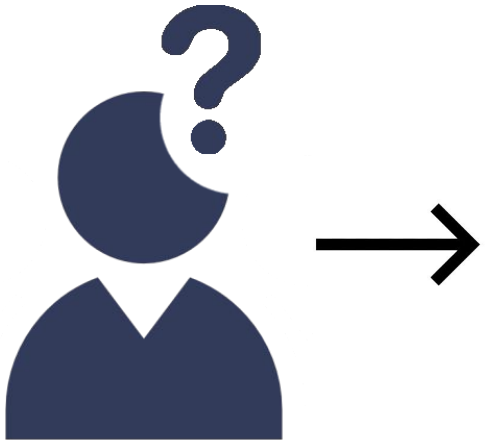
# Qualification stage

# Data

- Bond market
- About 2 years history
- Traders can perform bond transactions
- Several tables describing users, bonds, global market
- Predict traders activity

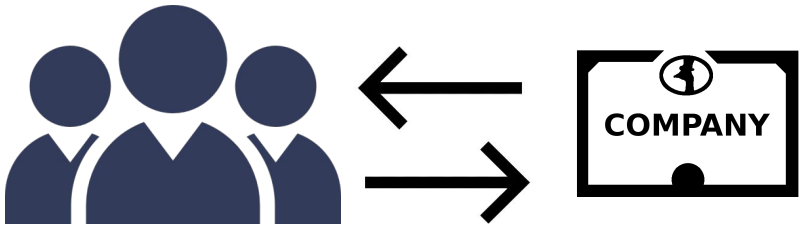

# Data

1. The transaction was completed (that is, the user bought / sold the paper)
2. The user looked at the paper, but did not make a deal
3. User has set aside a paper to buy / sell in the future
4. The transaction was not completed for technical reasons.
5. Holding



BUY/SELL
ADD TO CARD
LEAVE

# Dataset creating process



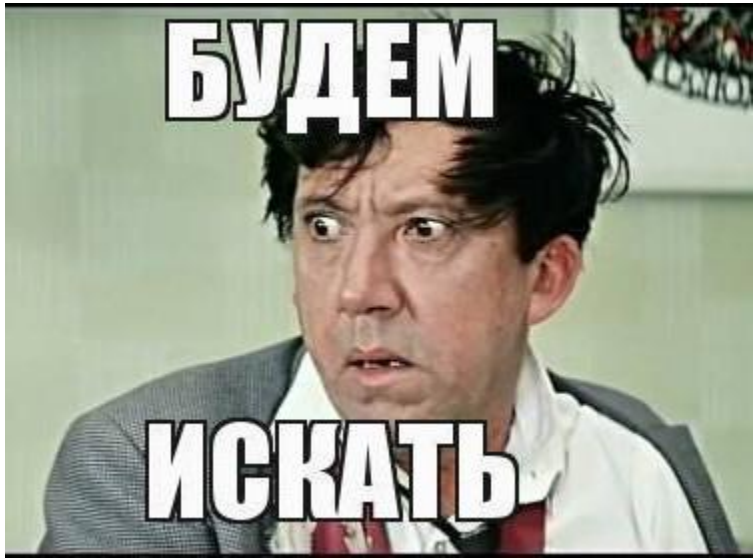

| USER ID | BOND ID | ORDER TYPE | STATUS | EXTRA FIELDS | TARGET |
|---|---|---|---|---|---|
| 15 | 7 | Buy | Postponed for the future | ... | 1 |

# Where to get 0-targets?

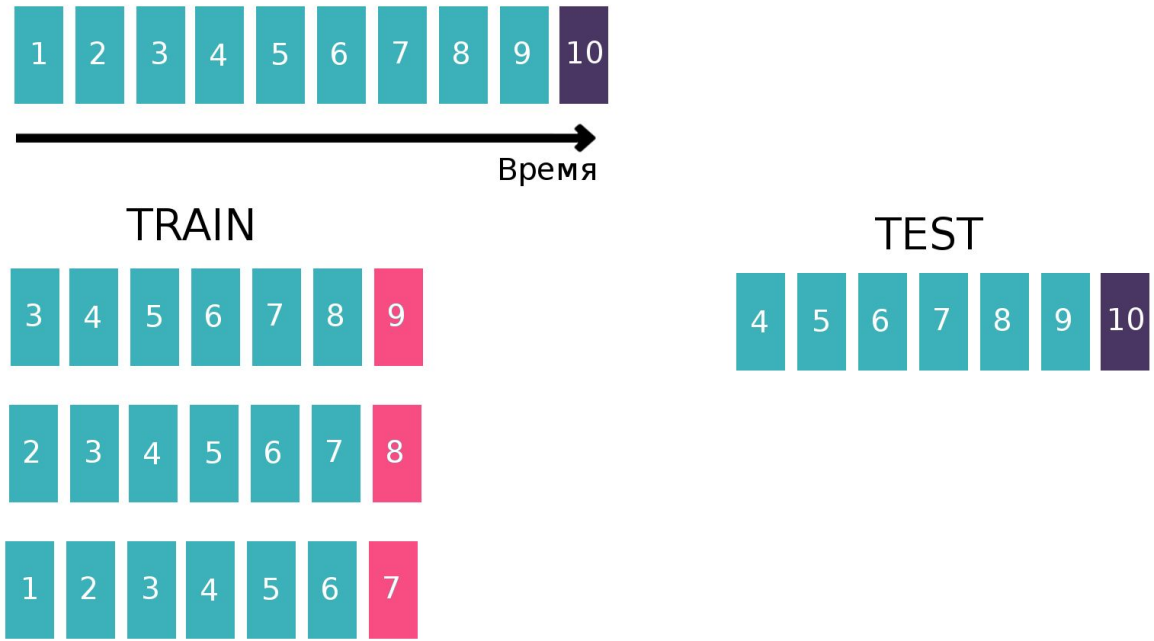
- Take all possible pairs

- Generate user and bond independently

- Take pairs from history



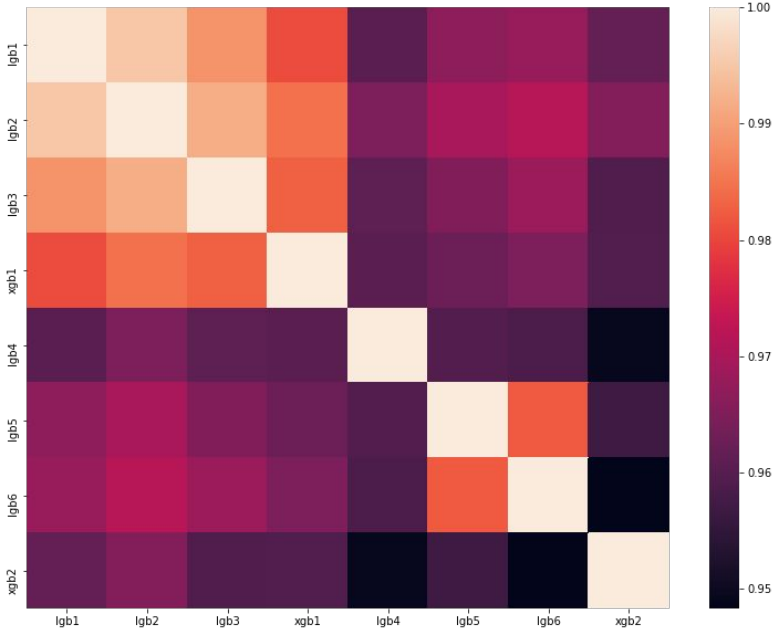
# Validation


1 2 3 4 5 6 7 8 9 10
Время
TRAIN
3 4 5 6 7 8 9
2 3 4 5 6 7 8
1 2 3 4 5 6 7
TEST
4 5 6 7 8 9 10

# Basic features

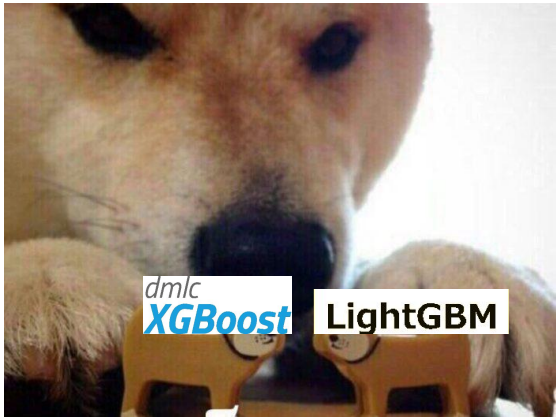
- How often did a pair / triple meet in the last 1, 2, 5, 20, 100 weeks?
- Statistics on the time intervals between pairs / triples in dataset (mean, std, max, min)
- The distance in time to the first / last time the pair / three met
- The share of each TradeStatus value for a pair / triple
- Statistics on how many times a week a pair / triple meets (mean, std, max, min)

# Last chance




Averaging of different models on different datasets:

- Xgboost
- Lightgbm

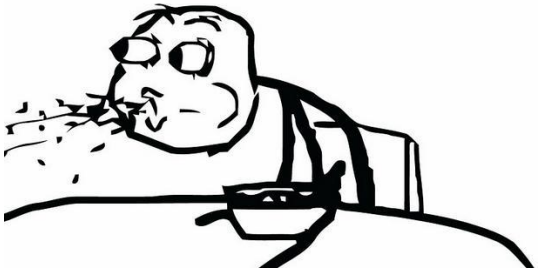


| # | △pub | Team Name | Kernel | Team Members | Score | Entries | Last |
|---|---|---|---|---|---|---|---|
| | — | **Fountainhead - MSU - Russia** | | | 0.80882 | 43 | 4mo |
| 2 | — | **Big Russian Loss - MIPT - Rus...** | | | 0.80272 | 110 | 4mo |
| 3 | ▲ 1 | **ShallowLearning - MIPT - Rus...** | | | 0.80204 | 81 | 4mo |
| 4 | ▼ 1 | **BruteForce - HSE - Russia** | | | 0.80023 | 22 | 4mo |
| 5 | — | **Extra Mile - UNI - Peru** | | | 0.79940 | 85 | 4mo |
| 6 | — | **Artificial Psycho Killer - UNIFA...** | | | 0.79909 | 97 | 4mo |
| 7 | ▲ 1 | **Magic City - SPBSU - Russia** | | | 0.79841 | 19 | 4mo |
| 8 | ▲ 2 | **Chemist - MUCTR - Russia** | | | 0.79716 | 38 | 4mo |
| 9 | ▼ 2 | **Mean Predictors - HU - Germa...** | | | 0.79696 | 97 | 4mo |
| 10 | ▼ 1 | **Data Emissaries - IIM Calcutt...** | | | 0.79536 | 53 | 4mo |

| | Δpub | Team Name | Kernel | Team Members | Score | Entries | Last |
|---|---|---|---|---|---|---|---|
| 1 | | **Fountainhead - MSU - Russia** | | | 0.80882 | 43 | 4mo |
| 2 | — | **Big Russian Loss - MIPT - Rus...** | | | 0.80272 | 110 | 4mo |
| 3 | ▲ 1 | **ShallowLearning - MIPT - Rus...** | | | 0.80204 | 81 | 4mo |
| 4 | ▼ 1 | **BruteForce - HSE - Russia** | | | 0.80023 | 22 | 4mo |
| 5 | — | **Extra Mile - UNI - Peru** | | | 0.79940 | 85 | 4mo |
| 6 | — | **Artificial Psycho Killer - UNIFA...** | | | 0.79909 | 97 | 4mo |
| 7 | ▲ 1 | **Magic City - SPBSU - Russia** | | | 0.79841 | 19 | 4mo |
| 8 | ▲ 2 | **Chemist - MUCTR - Russia** | | | 0.79716 | 38 | 4mo |
| 9 | ▼ 2 | **Mean Predictors - HU - Germa...** | | | 0.79696 | 97 | 4mo |
| 10 | ▼ 1 | **Data Emissaries - IIM Calcutt...** | | | 0.79536 | 53 | 4mo |

# Models on raw data & simple features

| Local validation | Reality |
| --- | --- |
| [0] valid-auc: 0.490002<br>[10] valid-auc: 0.600384<br>[20] valid-auc: 0.687156<br>[30] valid-auc: 0.747321<br>[40] valid-auc: 0.79843<br>[50] valid-auc: 0.83848<br>[60] valid-auc: 0.875837<br>[70] valid-auc: 0.891165<br>[80] valid-auc: 0.91737<br>[90] valid-auc: 0.939562<br>[99] valid-auc: 0.946337 | Big Russian Loss - MIPT - Rus... 0.49712 |



# Main problems to be solved

- Creating the right training set
- Correct validation
    - Train & valid dataset do not look like a test

Difference:

- 500k samples in the test dataset (one week)
- ~15k samples for each previous week in train dataset

# How was dataset really build?

There are **two types of zero-labels:**

- User did not perform transaction (does not contain in dataset)
- Transaction with “Holding” type

Provided dataset contains all 1s and only second type of zeros.

# How was dataset really build?

**Main hint:** each couple of customer - item for the last 6 month was extended by adding both buy and sell appeared before the current week.

Additionally: all transactions before 2016-01-01 was shifted to this date

# Correct validation

- Use only 3 last week as a validation dataset
- There are no information about time during the week (test data contains only start date of the week)
- Use 12 last weeks for training models (about 15% of all data)
- Restrictions on some features

# Bad features

- Features based on all previous period
- “Sum” or “count” features
  - causes overfitting of tree-based algorithms very fast

# Aggregation & statistics features

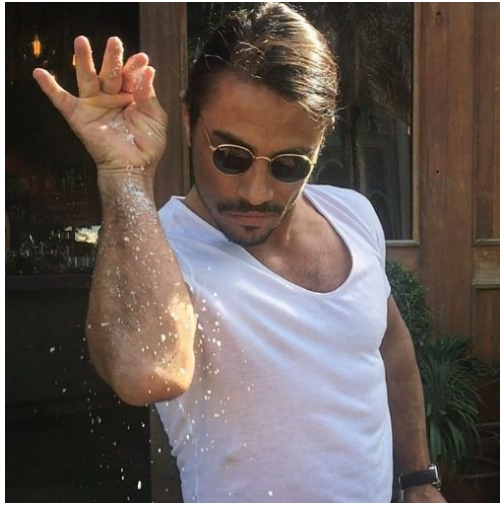
- Compute any statistics for different windows before the considering week grouped by *users* / *items* / *user+items* / *user+TradeStatus*
  - Sum, mean, max, min, median, quantiles of target for a last 1, 2, 3, 6, 12, 24, 27 weeks
  - A lot of features based on categorical data: nunique, size
- Features which describe users activity:
  - Time difference between last and first transaction
  - Time difference between start of current week and first transaction
  - Time difference between end of obligations and first transaction
  - “Density” of user transactions: number of transaction in the last N weeks divided by N

# Model

| Model | Training time | | out of the box | tuned |
|---|---|---|---|---|
| | CPU | GPU | | |
| XGBoost | 30 min | 15 min | 0.792 | 0.795 |
| LightGBM | 10 min | 7 min | 0.788 | 0.790 |
| CatBoost | 2 hour | 5 min | 0.790 | 0.791 |

# Clever features

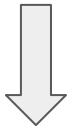
- Customer - item matrix
  - PCA (works only for users)
  - Item2Vec
- Clustering of users & generating the same features for clusters
- Embeddings from *implicit* library (Alternative Least Square algorithm)



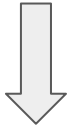
Score: ~0.802 roc-auc

# Very strange but working features

Value counts by the end dates of obligations



Improved from 0.802 to 0.803 (!)

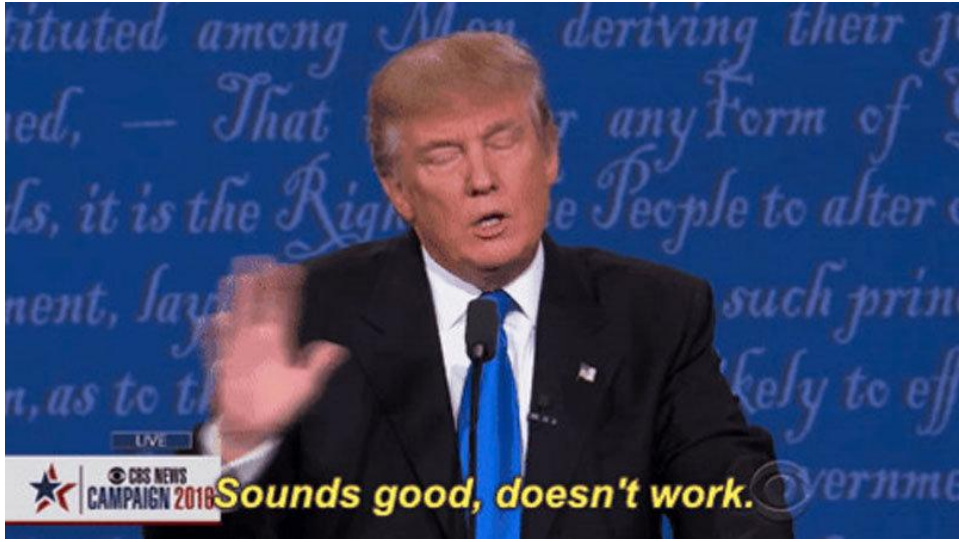


More features based on this dates as the categorical data

Score: ~0.805 roc-auc

# What did not work

- Classifier for each customer, item, customer+item
  - It works with great score (>0.96 roc-auc), but did not help for best model
- Stacking, blending, weak models (KNN / fc-NNs, Random Forest)
- More accurate recommender system approaches:
  - Cosine distance
  - SVD
  - TF-IDF
- Market / Macroeconomic data



# Final stage

# Data and Task

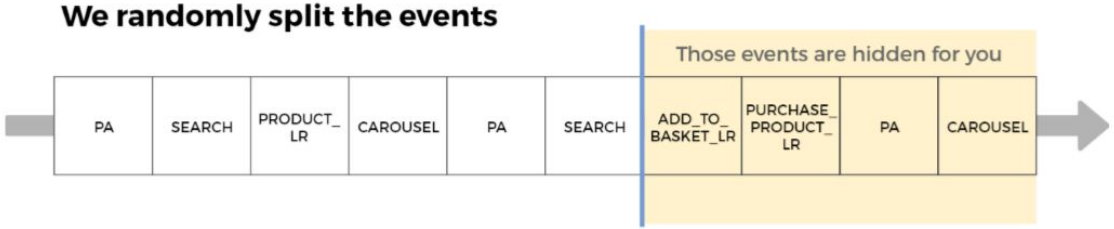
Data Provider: CDiscount
Top-2 Marketplace in France

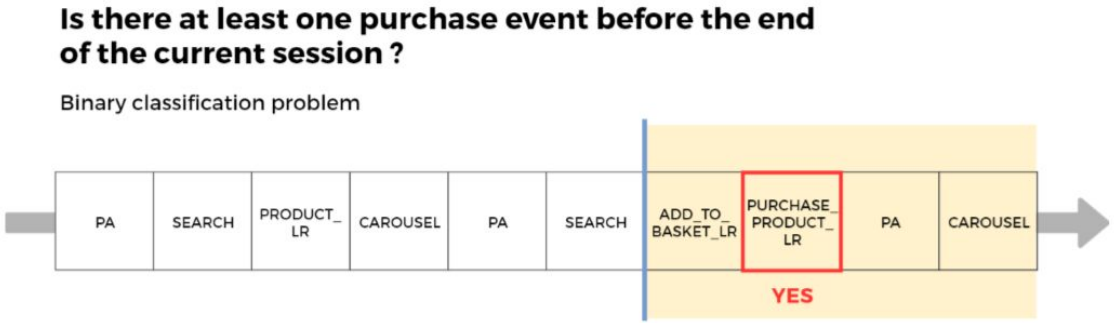
# Data and Task

**Data:** user sessions from CDiscount website randomly splitted in the middle

**Task:** Predict whether the end of session will contain a purchase

**Metric:** LogLoss

# Data descriptions

| Variable | Type | Encrypted | Description |
|---|---|---|---|
| sid | *string* | True | Session ID |
| type | *string* | False | Event type |
| query | *string* | True | Search query written by the customer |
| nb_query_terms | *int* | False | Number of query terms |
| rcount | *int* | False | Number of results returned |
| pn | *int* | False | Page result number |
| facets | *string* | True | Filters added |
| products | *stringified JSON* | True | List of product objects returned |
| dproducts | *stringified JSON* | True | List of product IDs returned |
| rh | *int* | False | Screen resolution (height) |
| rw | *int* | False | Screen resolution (width) |
| device | *string* | True | Customer device |
| idcar | *string* | True | Carousel ID where the product has been clicked/added to basket/bought |
| carproducts | *stringified JSON* | True | Product list seen on the carousel where the product has been clicked/added to basket/bought |
| sku | *string* | True | Product ID displayed to the user |
| offerid | *string* | True | Offer ID related to the product ID clicked/added to basket/bought |
| quantity | *int* | False | Number of products added to basket or purchased |
| stype | *string* | True | Seller type : Cdiscount or Marketplace |
| sname | *string* | True | Seller name |
| ff | *boolean* | False | True if the seller is not Cdiscount but storage and delivery are provided by Cdiscount |
| oidcar | *string* | True | Carousel ID where the product has been clicked/added to basket/bought |

| Variable | Type | Encrypted | Description |
|---|---|---|---|
| ocarproducts | *stringified JSON* | True | Product list seen on the carousel when the product has been clicked/added to basket/bought |
| oquery | *string* | True | Search query written by the customer when the product has been clicked/ added to basket/bought |
| orcount | *int* | False | Number of results returned when the product has been clicked/added to basket/bought |
| ofacets | *stringified JSON* | True | Filters added when the product has been clicked/added to basket/bought |
| opn | *int* | False | Page result number when the product has been clicked/added to basket/ bought |
| odproducts | *stringified JSON* | True | List of product IDs return when the product has been clicked/added to basket/bought |
| oproducts | *stringified JSON* | True | List of products objects return by the search engine when the product has been clicked/added to basket/bought |
| siteid | *string* | True | Website, tablet or phone |
| duration | *string* | False | Event duration |
| type_simplified | *string* | False | Simplified event type |

Need to predict probability for sid
Can use many information in JSON

Validation: StratifiedKfold

Model: LightGBM

Plan: Start separately and then blend our models

# First features: Statistics and aggregations

```
.groupby('sid')
```

```
.groupby(['type', 'sid'])
```

```
.groupby(['type_simplified', 'sid'])
```

+ raw features + features from JSON +

+ `['max', 'min', 'median', 'sum', 'std', ...]`

= Good start and 1st place on LB

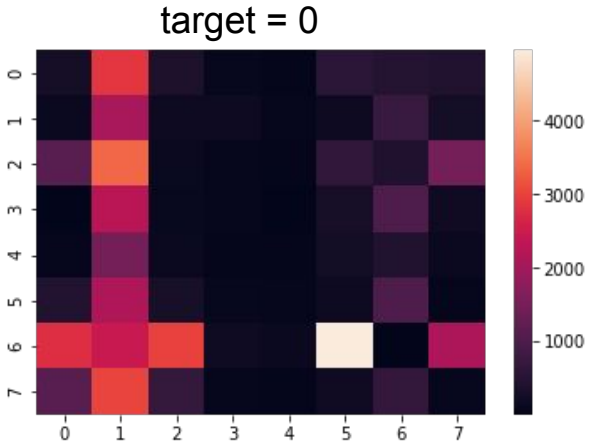
# Time features

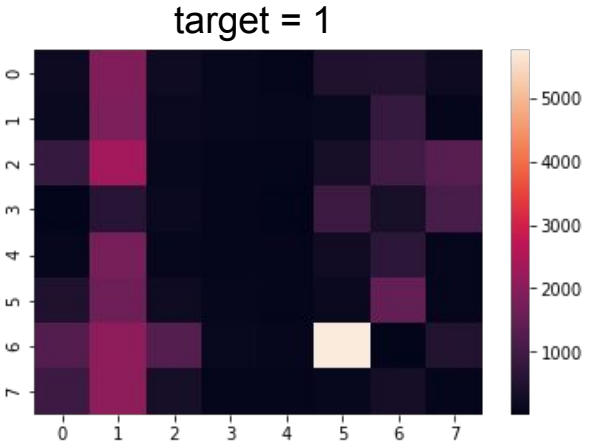
Analyze the transition matrix:

- transition frequency
- average transition time (to page / from page)
- Other statistics based on transition matrices

Type simplified

| | |
|---|---|
| 0 | Carousel |
| 1 | Landing page |
| 2 | Search |
| 3 | Add to basket |
| 4 | Product |
| 5 | Show case |
| 6 | Purchase product |
| 7 | List product |



# Other difficulties

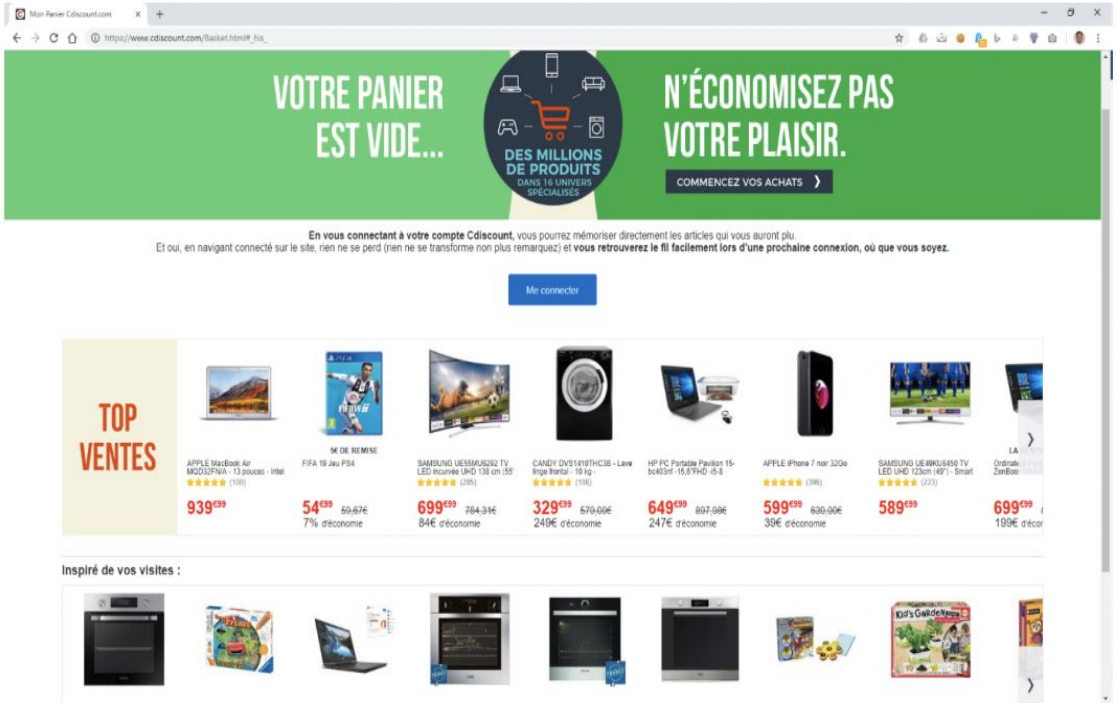
- Motivation
  - when you drop on leaderboard
- Dispute
  - when you don’t know what to add to model
- Lack of sleep => productivity
  - (sometimes it is better to sleep)



# Other features

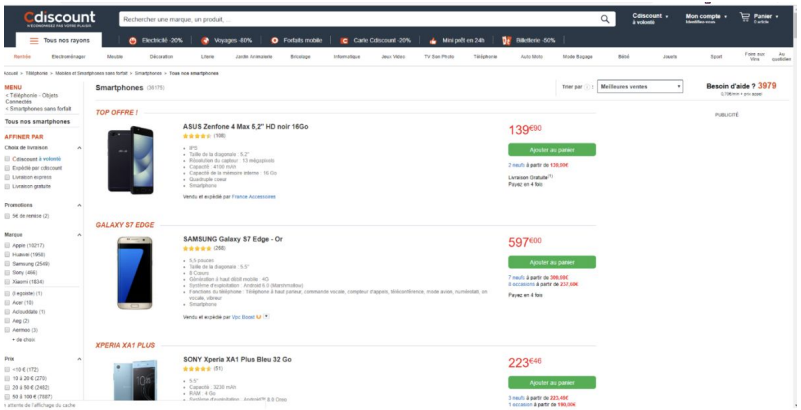
- Ratings
- Number of votes
- Screen resolution
- Order of goods on the page
- Carousel id

## Carousel

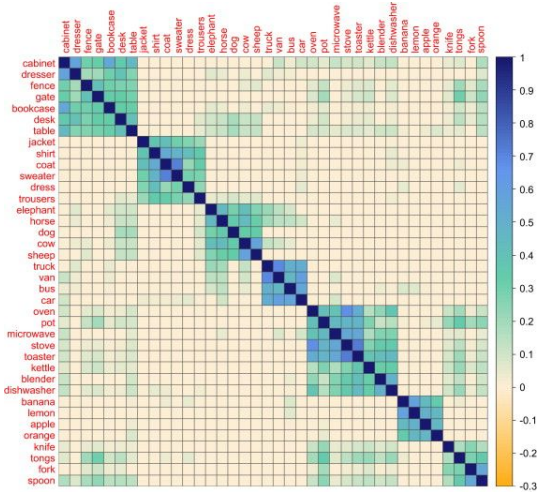


# Other features

- Average precision@K, NDCG
  - for search and list pages
  - price as relevance




- Cosine similarities statistics for events embeddings
  - use  standing in same session as local context for events
  - train gensim W2V and obtain embeddings for events
  - having N events in session make NxN matrix of cosine similarities
  - statistics on matrix (overall, row by column, column by row)



# Tf-Idf for text analysis

Usually applied for text embeddings for collection of documents:

seq1.txt
seq2.txt
....

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

**TF-IDF**

Term $x$ within document $y$

$tf_{x,y}$ = frequency of $x$ in $y$

$df_x$ = number of documents containing $x$

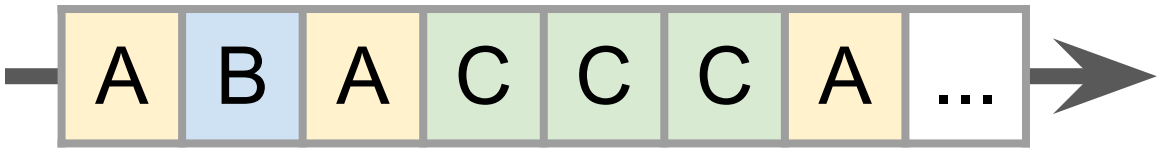
$N$ = total number of documents

| id | men | entered | bank | charlotte | missiles | masks | aryan | guns | witnesses | reported | silver | suv | august |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| seg1.txt | 0.239441 | 0 | 0.153457 | 0.195243 | 0 | 0.237029 | 0 | 0.195243 | 0.237029 | 0.140004 | 0.195243 | 0.237029 | 0 |
| seg13.txt | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg14.txt | 0 | 0.192197 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.172681 |
| seg15.txt | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.149652 |
| seg16.txt | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg17.txt | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg18.txt | 0 | 0.158432 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg19.txt | 0 | 0 | 0 | 0.197255 | 0 | 0 | 0 | 0 | 0 | 0.141447 | 0 | 0 | 0.155038 |
| seg2.txt | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg20.txt | 0 | 0.234323 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg21.txt | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg22.txt | 0 | 0 | 0 | 0 | 0.139629 | 0 | 0.127389 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg23.txt | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.180656 | 0 | 0 | 0 |
| seg24.txt | 0 | 0 | 0 | 0 | 0 | 0 | 0.117966 | 0 | 0 | 0.117966 | 0 | 0 | 0 |
| seg25.txt | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg26.txt | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg27.txt | 0 | 0 | 0.235418 | 0 | 0 | 0 | 0.214781 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg28.txt | 0 | 0 | 0 | 0 | 0.151753 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg29.txt | 0 | 0 | 0 | 0 | 0 | 0 | 0.129852 | 0 | 0 | 0 | 0 | 0 | 0.142329 |
| seg3.txt | 0 | 0 | 0 | 0 | 0.18432 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg30.txt | 0.078262 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg31.txt | 0 | 0 | 0.213409 | 0 | 0 | 0 | 0.194701 | 0 | 0 | 0 | 0 | 0 | 0 |
| seg32.txt | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# But we can apply it for sequential data!

Document of words ~ Sequence of events

Word ~ Some event / **event n-gram**




Unigrams: A, B, A, …

Bigrams: AB, BA, AC, …

Trigrams: ABC, BAC, ACC, ...

Then calculate tf-idf for bigrams/trigrams inside each sequence and use it as features

# Then apply tf-idf for our task

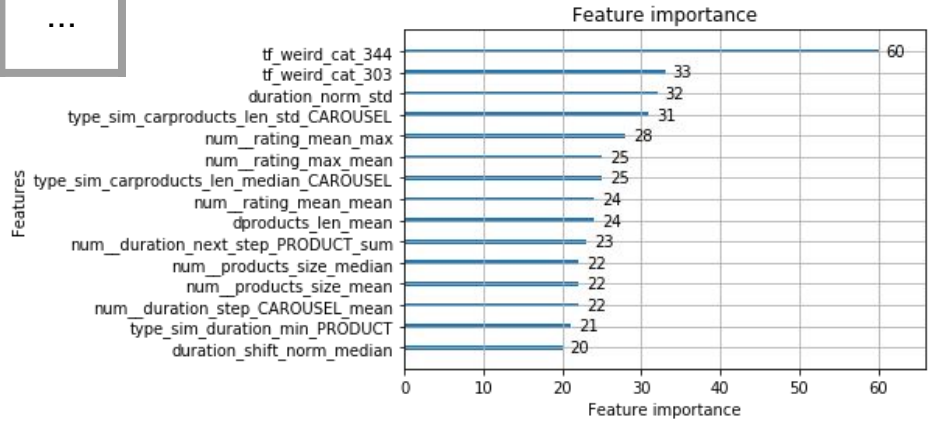
- Sequence of event types (carousel, add_to_basket, search, … )

| A | B | A | C | C | C | A | ... |
|---|---|---|---|---|---|---|---|

- Sequence of event types + sku (carousel, sku1, add_to_basket, sku2, search, sku3, … )

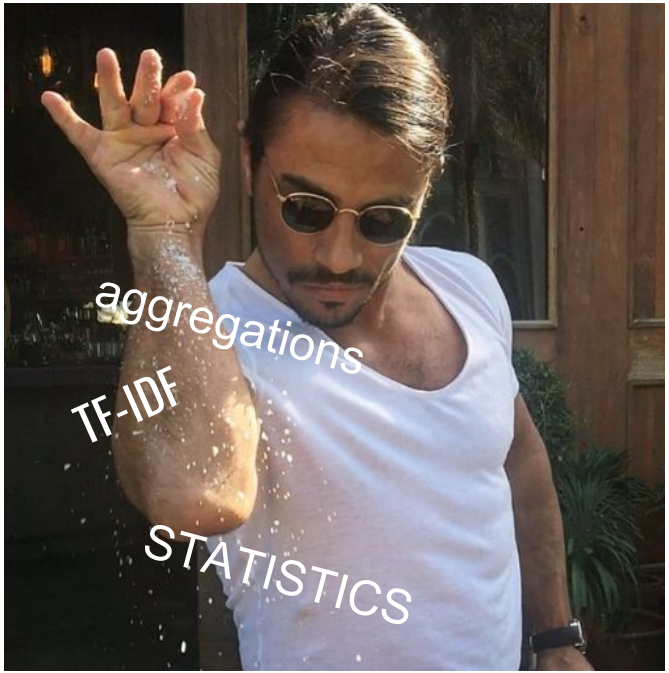
| A | sku1 | A | sku2 | C | sku3 | A | ... |
|---|---|---|---|---|---|---|---|

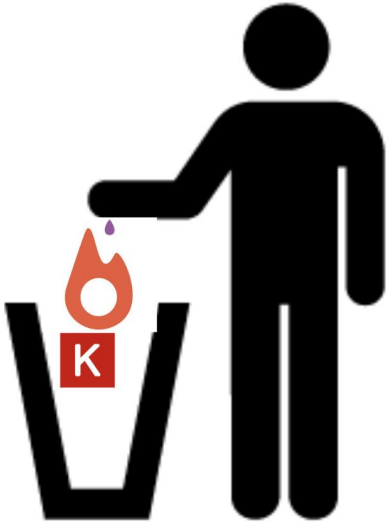
- Used bigrams + 500 max_num_features



# Unsuccessful or didn’t try

- Mean target
- NN (CNN, RNN, … )

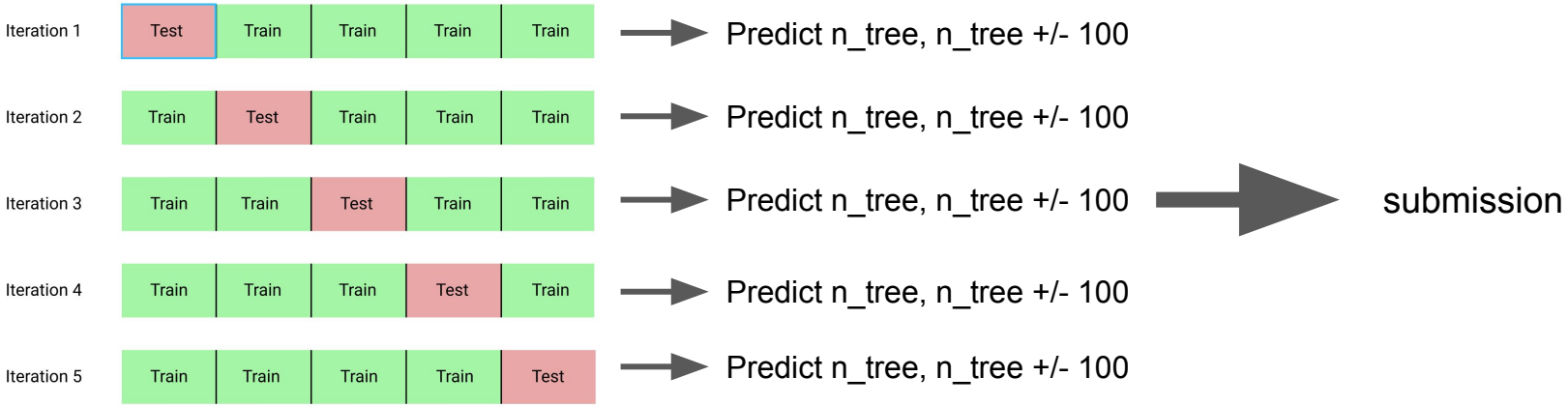


# Final Solution

Two submits:

- Overfitting for public (0.26546 LB)
  - Blending of best models
  - Coefficient selection according public LB
- Trust CV (0.26560 LB)
  - Out-of-fold predictions (LGBM)
  - gathering all features



# Final standings

| # | Name | Score | Count | Last |
|---|---|---|---|---|
| 1 |  Big Russian Loss | 0.25397 | 2 | 29/09/18 12:17:56 |
| 2 |  Magic City | 0.25451 | 3 | 29/09/18 12:10:59 |
| 3 | FO Fountainhead | 0.25461 | 3 | 29/09/18 12:53:25 |
| 4 | AP Artificial Psycho Killers | 0.25600 | 3 | 29/09/18 12:22:01 |
| 5 | EN Enough Naughty Statistics | 0.25602 | 2 | 29/09/18 12:24:47 |
| 6 | MO Mosiam | 0.25660 | 3 | 29/09/18 12:04:57 |
| 7 |  np.argmax | 0.25713 | 3 | 29/09/18 12:03:15 |
| 8 |  LiquoriceSolution | 0.25714 | 3 | 29/09/18 12:39:40 |

Impressions and solution(8 place): https://towardsdatascience.com/why-you-should-not-code-30-hours-in-a-row-a3a471301826

# Thanks for attention!


| Konstantin Gavrilchik | Artur Fattakhov |
| --- | --- |
| Telegram/ODS slack: @kgavrilchik | Telegram/ODS slack: @fartuk |
| Mail: konstantin.gavrilchik@phystech.edu | Mail: fattahov.ao@phystech.edu |
| **Evgeny Kononenko** | **Vasily Ryazanov** |
| Telegram/ODS slack: @Lenny_nn | Telegram/ODS slack: @ryazanoff |
| Mail: yevgeny.kononenko@phystech.edu | Mail: vasily.ryazanov@phystech.edu |