Yandex
Data Factory

# RecSys Challenge 2015: ensemble learning with categorical features

Peter Romov, Evgeny Sokolov

The ACM Conference Series on
**Recommender Systems**

# Problem statement

- Logs from e-commerce website: collection of sessions
- Session
  - sequence of clicks on the item pages
  - could end with or without purchase
- Click
  - Timestamp
  - ItemID (≈54k unique IDs)
  - CategoryID (≈350 unique IDs)
- Purchase
  - set of bought items with price and quantity
- Known purchases for the train-set, need to predict on the test-set

# Problem statement

Clicks from session ***s*** $c(s) = (c_1(s), \ldots, c_{n(s)}(s))$

Purchase (actual) $y(s) = \begin{cases} \emptyset & \text{— no purchase} \\ \{i_1, \ldots, i_{m(s)}\} & \text{(bought items) — otherwise} \end{cases}$

Purchase (predicted) $h(s) \approx y(s)$

**Evaluation measure:**

$$Q(h, S_{\text{test}}) = \sum_{\substack{s \in S_{\text{test}}: \\ |h(s)|>0}} \begin{cases} \frac{|S^b_{\text{test}}|}{|S_{\text{test}}|} + J(y(s), h(s)), & \text{if } y(s) \neq \emptyset \\ -\frac{|S^b_{\text{test}}|}{|S_{\text{test}}|}, & \text{otherwise} \end{cases}$$

where $J(y(s), h(s)) = \frac{|y(s) \cap h(s)|}{|y(s) \cup h(s)|}$ — Jaccard distance between two sets

$S_{test}$ — all sessions from test-set

$S^b_{test}$ — sessions from test-set with purchase

# Problem statement

First observations (from the task):

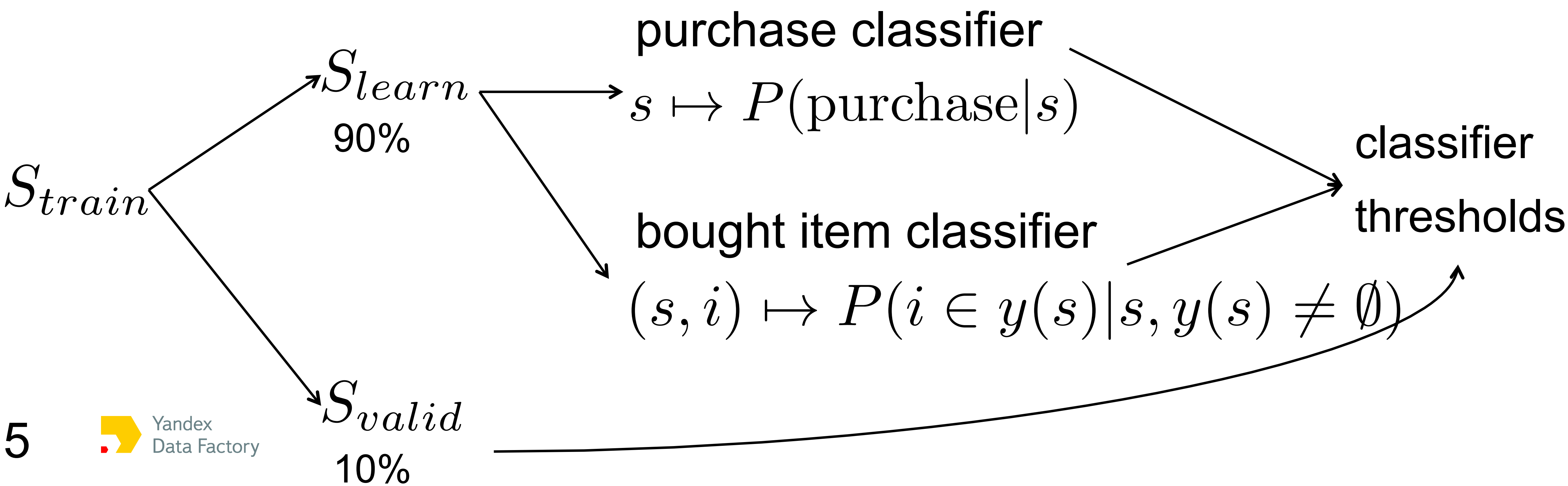
- the task is uncommon (set prediction with specific loss function)
- evaluation measure could be rewritten

$$Q(h, S_{\text{test}}) = \underbrace{\frac{|S^b_{\text{test}}|}{|S_{\text{test}}|}(\text{TP} - \text{FP})}_{\text{purchase score}} + \underbrace{\sum_{s \in S_{\text{test}}} J(y(s), h(s))}_{\text{Jaccard score}},$$

- the original problem can be divided into two well-known binary classification problems;
  1. predict purchase given session optimize Purchase score $P(y(s) \neq \emptyset | s)$
  2. predict bought items given session with purchase optimize Jaccard score $P(i \in y(s) | s, y(s) \neq \emptyset)$

# Solution schema

- Two-stage prediction
  - Two binary classification models learned on the train-set
  - Both classifiers require thresholds
  - Set up thresholds to optimize Purchase score and Jaccard score using hold-out subsample of the train-set

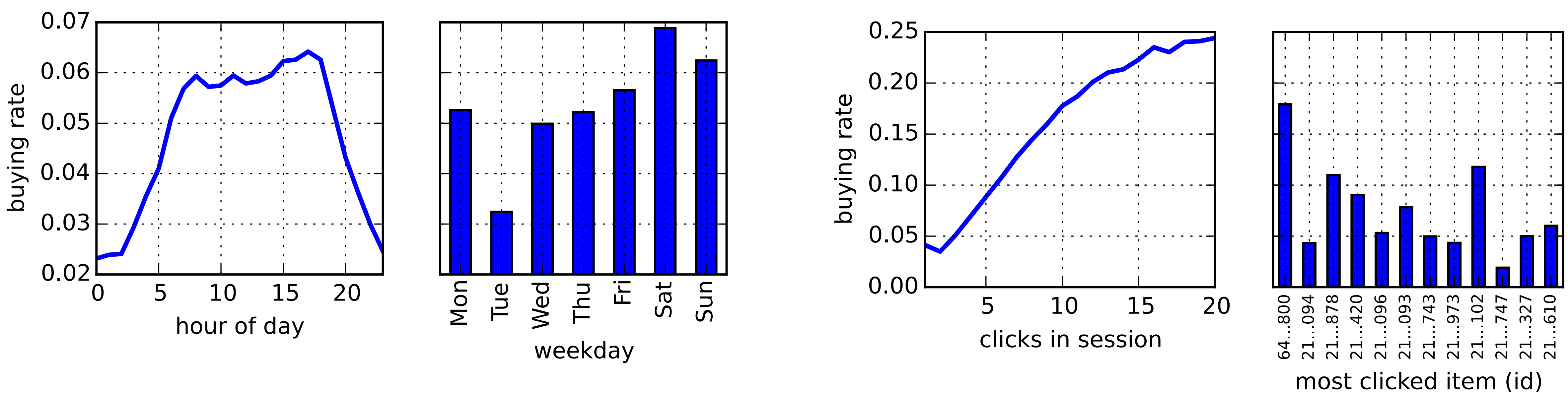


# Some relationships from data

Next observations (from the data):

- Buying rate strongly depends on time features
- Buying rate varies highly between categorical features

Buying rate — fraction of buyer sessions in some subset of sessions



# Feature extraction

- Purchase classifier: features from session (sequence of clicks)
- Bought item classifier: features from pair session+itemID
  - Observation: bought item is a clicked item

- **We use two types of features**
  - Numerical: real number, e.g. seconds between two clicks
  - Categorical: element of the unordered set of values (levels), e.g. ItemID

# Feature extraction: session

1. Start/end of the session (month, day, hour, etc.)
   [numerical + categorical with few levels]
2. Number of clicks, unique items, categories, item-category pairs
   [numerical]
3. Top 10 items and categories by the number of clicks
   [categorical with ≈50k levels]
4. ID of the first/last item clicked at least *k* times
   [categorical with ≈50k levels]
5. Click counts for 100 items and 50 categories that were most popular in the whole training set
   [sparse numerical]

# Feature extraction: session+ItemID

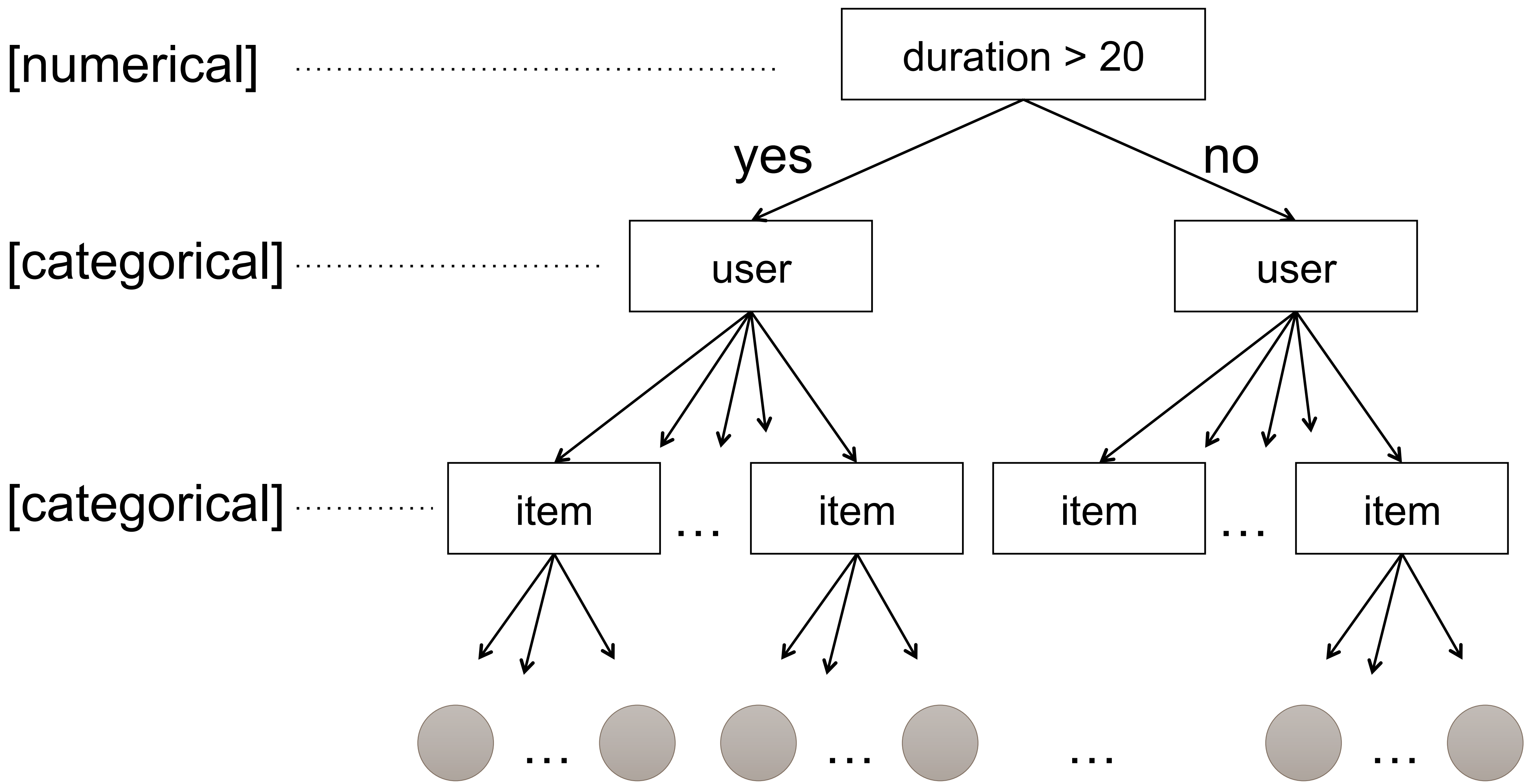
1. All session features
2. ItemID
   [categorical with ≈50k levels]
3. Timestamp of the first/last click on the item (month, day, hour, etc.)
   [numerical + categorical with few levels]
4. Number of clicks in the session for the given item
   [numerical]
5. Total duration (by analogy with dwell time) of the clicks on the item
   [numerical]

# Classification method

- **GBM and similar ensemble learning techniques**
  - useful with numerical features
  - one-hot encoding of categorical features doesn’t perform well
- **Matrix decompositions, FM**
  - useful with categorical features
  - hard to incorporate numerical features because of rough (bi-linear) model
- **We used our internal learning algorithm: MatrixNet**
  - GBM with oblivious decision trees
  - trees properly handle categorical features (multi-split decision trees)
  - SVD-like decompositions for new feature value combinations

# Classification method

Oblivious decision tree with categorical features



# Classification method: speed

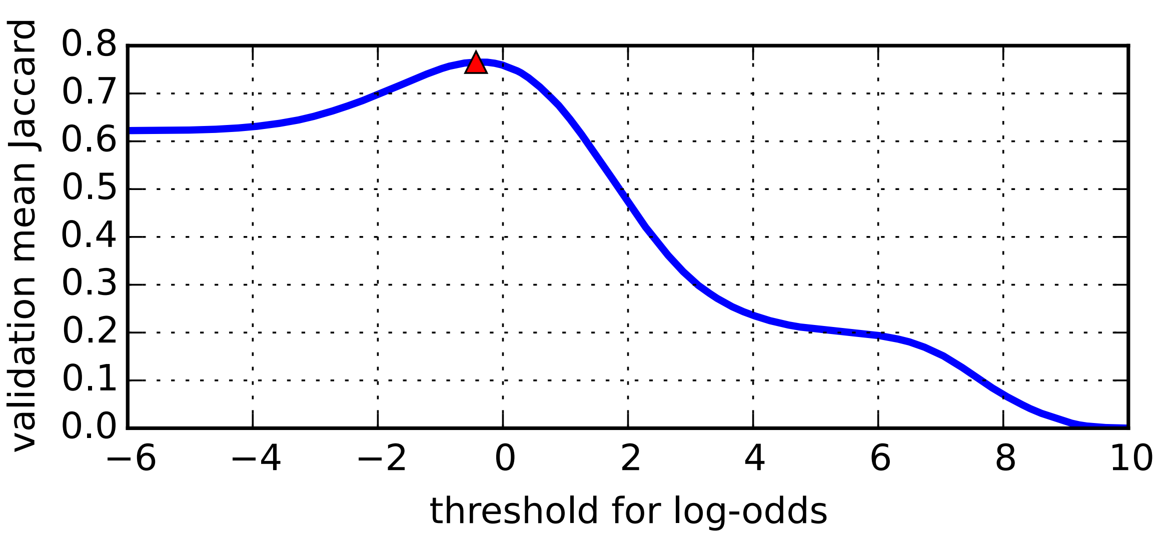
- Training classifiers
  - GB with 10k trees for each classifier
  - ≈12 hours to train both models on 150 machines
- Making predictions
  - We made 4000 predictions per second per thread

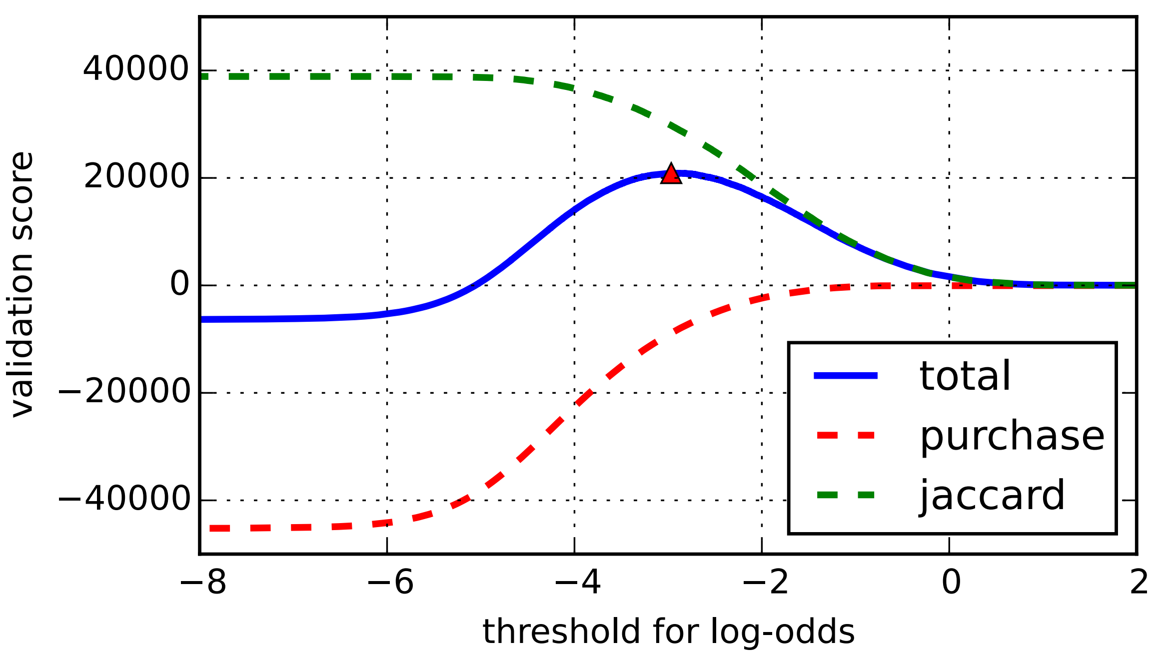
# Threshold optimization

$$Q(h, S_{\text{valid}}) = \underbrace{\frac{|S^b_{\text{valid}}|}{|S_{\text{valid}}|}(\text{TP} - \text{FP})}_{\text{purchase score}} + \underbrace{\sum_{s \in S_{\text{valid}}} J(y(s), h(s))}_{\text{Jaccard score}},$$

We optimized thresholds using validation set (10% hold-out from train-set)

1) Maximize Jaccard score
2) Maximize Purchase+Jaccard scores using fixed bought item threshold





Yandex
Data Factory

# Final results

- Leaderboard: **63102** (1st place)
- Purchase detection on validation (10% hold-out):
  - 16% precision
  - 77% recall
  - AUC 0.85
- Purchased item detection on validation:
  - Jaccard measure 0.765
- Features / datasets used to learn classifiers / evaluation process can be reproduced, see our code[1]

 Yandex Data Factory

[1] https://github.com/romovpa/ydf-recsys2015-challenge

# Summary / Questions?

1. **Observations from the problem statement**
   - The task is complex but decomposable into two well-known: binary classification of sessions and (session, ItemID)-pairs
2. **Observations from the data (user click sessions)**
   - Features from sessions and (session, ItemID)-pairs
   - Easy to develop many meaningful categorical features
3. **The algorithm**
   - Gradient boosting on trees with categorical features
   - No sophisticated mixtures of Machine Learning techniques: one algorithm to work with many numerical and categorical features