CatBoost: unbiased boosting with categorical features

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, Andrey Gulin

Yandex

May 15, 2018

CatBoost

- Based on the gradient boosting algorithm with decision trees as base predictors (GBDT)
- Effectively handles categorical features
- Shows best results on many datasets (compared with XGBoost, LightGBM, MatrixNet, H2O)
- Available as an open-source library: `https://github.com/catboost/`
- See also our paper: `https://arxiv.org/abs/1706.09516`

Notation

- Dataset $\mathcal{D} = \{(\mathbf{x}_k, y_k)\}_{k=1..n}$, $\mathbf{x}_k \in \mathbb{R}^m$, $y_k \in \mathbb{R}$
- $(\mathbf{x}_k, y_k)$ — i.i.d. from $P(\cdot, \cdot)$

Notation

- Dataset $\mathcal{D} = \{(\mathbf{x}_k, y_k)\}_{k=1..n}$, $\mathbf{x}_k \in \mathbb{R}^m$, $y_k \in \mathbb{R}$
- $(\mathbf{x}_k, y_k)$ — i.i.d. from $P(\cdot, \cdot)$
- $L(y, \hat{y})$ — loss function
- Looking for $F = \operatorname{argmin}_f \mathcal{L}(f),\ \mathcal{L}(f) := \mathbb{E}(L(y, f(\mathbf{x})))$

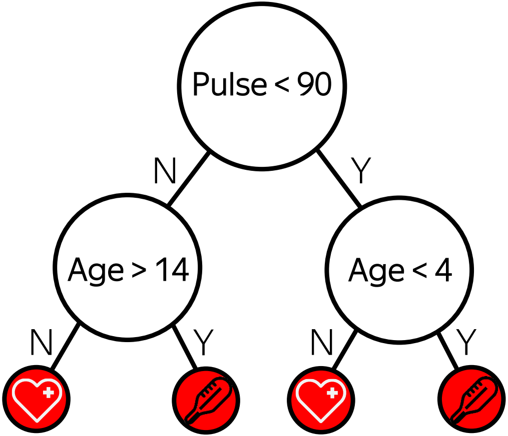
Notation

- Dataset $\mathcal{D} = \{(\mathbf{x}_k, y_k)\}_{k=1..n}$, $\mathbf{x}_k \in \mathbb{R}^m$, $y_k \in \mathbb{R}$
- $(\mathbf{x}_k, y_k)$ — i.i.d. from $P(\cdot, \cdot)$
- $L(y, \hat{y})$ — loss function
- Looking for $F = \operatorname{argmin}_f \mathcal{L}(f),\ \mathcal{L}(f) := \mathbb{E}(L(y, f(\mathbf{x})))$
- Gradient boosting: $F^t = F^{t-1} + \alpha h^t$,
  $h^t = \operatorname{argmin}_{h \in H} \mathcal{L}(F^{t-1} + h)$ (details later)

Notation

- Dataset $\mathcal{D} = \{(\mathbf{x}_k, y_k)\}_{k=1..n}$, $\mathbf{x}_k \in \mathbb{R}^m$, $y_k \in \mathbb{R}$
- $(\mathbf{x}_k, y_k)$ — i.i.d. from $P(\cdot, \cdot)$
- $L(y, \hat{y})$ — loss function
- Looking for $F = \operatorname{argmin}_f \mathcal{L}(f)$, $\mathcal{L}(f) := \mathbb{E}(L(y, f(\mathbf{x})))$
- Gradient boosting: $F^t = F^{t-1} + \alpha h^t$,
  $h^t = \operatorname{argmin}_{h \in H} \mathcal{L}(F^{t-1} + h)$ (details later)
- In CatBoost, H is a family of oblivious decision trees with limited depth

Decision tree



Algorithmic advances

1. The implementation of ordered boosting, a permutation-driven alternative to the classic algorithm
2. Innovative algorithm for processing categorical features

Categorical features

- Have discrete set of values (categories), not comparable with each other
- Cannot be used in binary decision trees directly

Categorical features

- Have discrete set of values (categories), not comparable with each other
- Cannot be used in binary decision trees directly
- **One-hot encoding:** add binary variables identifying categories
- Problems: large memory requirements and computational cost, weak features

Categorical features

- Have discrete set of values (categories), not comparable with each other
- Cannot be used in binary decision trees directly
- **One-hot encoding:** add binary variables identifying categories
- Problems: large memory requirements and computational cost, weak features
- Solution: use *target statistics* (TS) instead
- We replace category x_k^i by some numerical value $\hat{x}_k^i$ which usually approximates $\mathbb{E}(y|x^i = x_k^i)$

Greedy TS

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D}} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j}{\sum_{\mathbf{x}_j \in \mathcal{D}} \mathbb{1}_{\{x_j^i = x_k^i\}}}$$

Greedy TS

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D}} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j}{\sum_{\mathbf{x}_j \in \mathcal{D}} \mathbb{1}_{\{x_j^i = x_k^i\}}}$$

Problem: target leakage leads to a *conditional shift*, i.e., $\hat{x}^i|y$ differs for training and test examples

P1 $\mathbb{E}(\hat{x}^i|y = v) = \mathbb{E}(\hat{x}_k^i|y_k = v)$, *where* $(\mathbf{x}_k, y_k)$ *is the* k*-th training example*

Greedy TS

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D}} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j}{\sum_{\mathbf{x}_j \in \mathcal{D}} \mathbb{1}_{\{x_j^i = x_k^i\}}}$$

Problem: target leakage leads to a *conditional shift*, i.e., $\hat{x}^i | y$ differs for training and test examples

P1 $\mathbb{E}(\hat{x}^i | y = v) = \mathbb{E}(\hat{x}_k^i | y_k = v)$, *where* $(\mathbf{x}_k, y_k)$ *is the* k*-th training example*

Example: i-th feature is categorical, all values are unique, $P(y = 1 | x^i = A) = 0.5$:

$$\mathbb{E}(\hat{x}_k^i | y_k) = y_k \in \{0, 1\}$$

$$\mathbb{E}(\hat{x}^i | y) = 0.5$$

Greedy TS with prior

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D}} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + a\,P}{\sum_{\mathbf{x}_j \in \mathcal{D}} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

Greedy TS with prior

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D}} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + a\,P}{\sum_{\mathbf{x}_j \in \mathcal{D}} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

Still problems with P1:
$\hat{x}_k^i = \frac{aP}{1+a}$ if $y_k = 0$
$\hat{x}_k^i = \frac{1+aP}{1+a}$ if $y_k = 1$

Holdout TS

$\mathcal{D} = \mathcal{D}_0 \sqcup \mathcal{D}_1$, use $\mathcal{D}_0$ to calculate the TS and $\mathcal{D}_1$ to perform training

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_0} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + a\,P}{\sum_{\mathbf{x}_j \in \mathcal{D}_0} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

Holdout TS

$\mathcal{D} = \mathcal{D}_0 \sqcup \mathcal{D}_1$, use $\mathcal{D}_0$ to calculate the TS and $\mathcal{D}_1$ to perform training

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_0} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + a\,P}{\sum_{\mathbf{x}_j \in \mathcal{D}_0} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

P2 *It is desirable for $\hat{x}_k^i$ to have a low variance*

Leave-one-out TS

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D} \setminus \mathbf{x}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + a\,P}{\sum_{\mathbf{x}_j \in \mathcal{D} \setminus \mathbf{x}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

Leave-one-out TS

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D} \setminus \mathbf{x}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + a\,P}{\sum_{\mathbf{x}_j \in \mathcal{D} \setminus \mathbf{x}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

Example: $x_k^i = A$ for all examples
Let n^+ be the number of examples with $y = 1$
$\hat{x}_k^i = \frac{n^+ - y_k + a\,P}{n - 1 + a}$
For a test example: $\hat{x}^i = \frac{n^+ + a\,P}{n + a}$

Leave-one-out TS

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D} \setminus \mathbf{x}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + a\,P}{\sum_{\mathbf{x}_j \in \mathcal{D} \setminus \mathbf{x}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

Example: $x_k^i = A$ for all examples
Let n^+ be the number of examples with $y = 1$
$\hat{x}_k^i = \frac{n^+ - y_k + a\,P}{n-1+a}$
For a test example: $\hat{x}^i = \frac{n^+ + a\,P}{n+a}$

Violates P1:
$\mathbb{E}(\hat{x}_k^i | y_k) = \frac{n\,\mathbb{E}y - y_k + a\,P}{n-1+a}$
$\mathbb{E}(\hat{x}^i | y) = \frac{n\,\mathbb{E}y + a\,P}{n+a}$

Ordered TS

Perform a random permutation σ of the dataset

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j:\sigma(j)<\sigma(k)} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + a\,P}{\sum_{\mathbf{x}_j:\sigma(j)<\sigma(k)} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

Obtained *ordered* TS satisfies the requirement P1, and we also reduce the variance of $\hat{x}_k^i$ (see P2) compared to sliding window TS used in online learning.

CatBoost uses several permutations.

Comparison of TS

Relative change in logloss / zero-one loss:

	Greedy	Holdout	Leave-one-out
Adult	+1.1% / +0.79%	+2.1 % / +2.0%	+5.5% / +3.7%
Amazon	+40% / +32%	+8.3% / +8.3%	+4.5% / +5.6%
Click prediction	+13% / +6.7%	+1.5% / +0.51%	+2.7% / +0.90%
KDD appetency	+24% / +0.68%	+1.6% / -0.45%	+8.5% / +0.68%
KDD churn	+12% / +2.1%	+0.87% / +1.3%	+1.6% / +1.8%
KDD Internet	+33% / +22%	+2.6% / +1.8%	+27% / +19%
KDD upselling	+57% / +50%	+1.6% / +0.85%	+3.9% / +2.9%
Kick prediction	+22% / +28%	+1.3% / +0.32%	+3.7% / +3.3%

Prediction shift

- Gradient boosting: $F^t = F^{t-1} + \alpha h^t$, $h^t = \mathrm{argmin}_{h \in H} \mathcal{L}(F^{t-1} + h)$

Prediction shift

- Gradient boosting: $F^t = F^{t-1} + \alpha h^t$, $h^t = \text{argmin}_{h \in H} \mathcal{L}(F^{t-1} + h)$
- $g^t(\mathbf{x}, y) := \frac{\partial L(y,s)}{\partial s}\Big|_{s=F^{t-1}(\mathbf{x})}$
- $\hat{h}^t = \text{argmin}_{h \in H} \mathbb{E} \left(-g^t(\mathbf{x}, y) - h(\mathbf{x})\right)^2$

Prediction shift

- Gradient boosting: $F^t = F^{t-1} + \alpha h^t$,
  $h^t = \operatorname{argmin}_{h \in H} \mathcal{L}(F^{t-1} + h)$
- $g^t(\mathbf{x}, y) := \frac{\partial L(y,s)}{\partial s}\Big|_{s=F^{t-1}(\mathbf{x})}$
- $\hat{h}^t = \operatorname{argmin}_{h \in H} \mathbb{E}\left(-g^t(\mathbf{x}, y) - h(\mathbf{x})\right)^2$
- $h^t = \operatorname{argmin}_{h \in H} \frac{1}{n} \sum_{k=1}^{n} \left(-g^t(\mathbf{x}_k, y_k) - h(\mathbf{x}_k)\right)^2$

Prediction shift

- Gradient boosting: $F^t = F^{t-1} + \alpha h^t$,
  $h^t = \mathrm{argmin}_{h \in H} \mathcal{L}(F^{t-1} + h)$
- $g^t(\mathbf{x}, y) := \frac{\partial L(y,s)}{\partial s}\Big|_{s=F^{t-1}(\mathbf{x})}$
- $\hat{h}^t = \mathrm{argmin}_{h \in H} \mathbb{E}\left(-g^t(\mathbf{x}, y) - h(\mathbf{x})\right)^2$
- $h^t = \mathrm{argmin}_{h \in H} \frac{1}{n} \sum_{k=1}^{n} \left(-g^t(\mathbf{x}_k, y_k) - h(\mathbf{x}_k)\right)^2$

Shifts:

1. $g^t(\mathbf{x}_k, y_k) \mid \mathbf{x}_k$ is shifted from $g^t(\mathbf{x}, y) \mid \mathbf{x}$
2. So, h^t is biased with respect to $\hat{h}^t$
3. This, finally, affects the generalization ability of the trained model F^t

Theoretical example

- Two features x^1, x^2 — i.i.d. Bernoulli random variables with $p = 1/2$
- $y = f^*(\mathbf{x}) = c_1 x^1 + c_2 x^2$
- Use decision stumps, $\alpha = 1$, $N = 2$
- $F^2 = h^1 + h^2$, h^1 based on x^1 and h^2 based on x^2

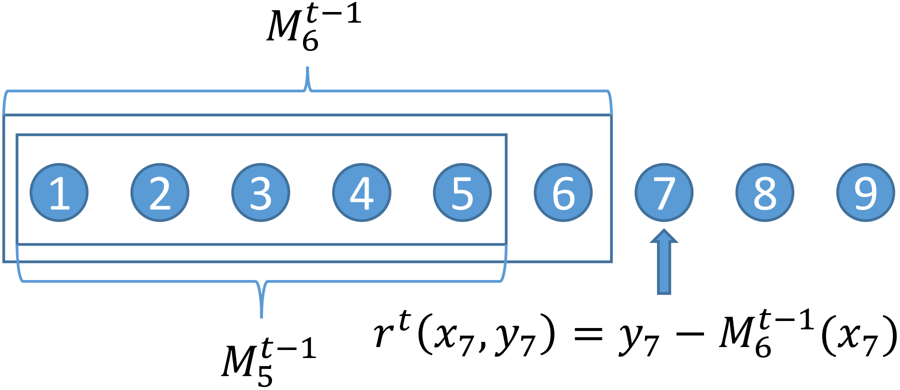
Theoretical example

- Two features x^1, x^2 — i.i.d. Bernoulli random variables with $p = 1/2$
- $y = f^*(\mathbf{x}) = c_1 x^1 + c_2 x^2$
- Use decision stumps, $\alpha = 1$, $N = 2$
- $F^2 = h^1 + h^2$, h^1 based on x^1 and h^2 based on x^2

Proposition

1. If two independent samples $\mathcal{D}_1$ and $\mathcal{D}_2$ of size n are used to estimate h^1 and h^2, respectively, then $\mathbb{E}_{\mathcal{D}_1,\mathcal{D}_2} F^2(\mathbf{x}) = f^*(\mathbf{x})$ for any $\mathbf{x} \in \{0,1\}^2$.
2. If the same dataset $\mathcal{D}$ is used for both h^1 and h^2, then $\mathbb{E}_{\mathcal{D}} F^2(\mathbf{x}) = f^*(\mathbf{x}) - \frac{1}{n-1} c_2 (x^2 - \frac{1}{2}) + O(1/2^n)$.

Ordered boosting



Ordered boosting

Algorithm 1: Ordered boosting

input : $\{(\mathbf{x}_k, y_k)\}_{k=1}^n, I$;

$M_i \leftarrow 0$ for $i = 1..n$;
for $t \leftarrow 1$ **to** I **do**
    for $i \leftarrow 1$ **to** n **do**
        $r_i \leftarrow y_i - M_{i-1}(i)$;
    for $i \leftarrow 1$ **to** n **do**
        $M \leftarrow LearnModel((\mathbf{x}_j, r_j)_{j=1..i})$;
        $M_i \leftarrow M_i + M$;
return M_n

Implementation

Two phases: choosing the tree structure and choosing the values in leaves

Second phase:

- This phase uses the standard GBDT scheme
- σ_0 — random permutation used for computing ordered TS

Implementation

Two phases: choosing the tree structure and choosing the values in leaves

Second phase:

- This phase uses the standard GBDT scheme
- σ_0 — random permutation used for computing ordered TS

First phase:

- Two modes: *Ordered* and *Plain*
- $\sigma_1, \ldots, \sigma_s$ — random permutation used for computing ordered TS, also used in Ordered mode
- At each step we construct a tree based on a randomly sampled permutation σ_r

Implementation

Ordered mode:

- For simplicity of notation order examples according to σ_r
- $M_{r,j}(i)$ — current prediction for i-th example based on examples $1..j$
- $grad_{r,j}(i)$ is computed based on $M_{r,j}(i)$

Implementation

Ordered mode:

- For simplicity of notation order examples according to σ_r
- $M_{r,j}(i)$ — current prediction for i-th example based on examples $1..j$
- $grad_{r,j}(i)$ is computed based on $M_{r,j}(i)$
- Target gradient: $G = (grad_{r,0}(1), \dots, grad_{r,n-1}(n))$
- Choosing a split: for i-th example average $grad_{r,i-1}(j)$ for $j < i$ in the same leaf and compare the obtained vector with G

Implementation

Ordered mode:

- For simplicity of notation order examples according to σ_r
- $M_{r,j}(i)$ — current prediction for i-th example based on examples $1..j$
- $grad_{r,j}(i)$ is computed based on $M_{r,j}(i)$
- Target gradient: $G = (grad_{r,0}(1), \ldots, grad_{r,n-1}(n))$
- Choosing a split: for i-th example average $grad_{r,i-1}(j)$ for $j < i$ in the same leaf and compare the obtained vector with G
- $M_{r,j}(i) \leftarrow M_{r,j}(i) - \alpha \operatorname{avg}(grad_{r,i-1}(j)$ for $j < i$ in the same leaf)

Comparison with baselines

Logloss / zero-one loss, relative increase is presented in the brackets:

	CatBoost	LightGBM	XGBoost
Adult	**0.2695 / 0.1267**	0.2760 (+2.4%) / 0.1291 (+1.9%)	0.2754 (+2.2%) / 0.1280 (+1.0%)
Amazon	**0.1394 / 0.0442**	0.1636 (+17%) / 0.0533 (+21%)	0.1633 (+ 17%) / 0.0532 (+ 21%)
Click prediction	**0.3917 / 0.1561**	0.3963 (+1.2%) / 0.1580 (+1.2%)	0.3962 (+1.2%) / 0.1581 (+1.2%)
Epsilon	**0.2647 / 0.1086**	0.2703 (+1.5%) / 0.114 (+4.1%)	0.2993 (+11%) / 0.1276 (+12%)
KDD appetency	**0.0715 / 0.01768**	0.0718 (+0.4%) / 0.01772 (+0.2%)	0.0718 (+0.4%) / 0.01780 (+0.7%)
KDD churn	**0.2319 / 0.0719**	0.2320 (+0.1%) / 0.0723 (+0.6%)	0.2331 (+0.5%) / 0.0730 (+1.6%)
KDD Internet	**0.2089 / 0.0937**	0.2231 (+6.8%) / 0.1017 (+8.6%)	0.2253 (+7.9%) / 0.1012 (+8.0%)
KDD upselling	**0.1662 / 0.0490**	0.1668 (+0.3%) / 0.0491 (+0.1%)	0.1663 (+0.04%) / 0.0492 (+0.3%)
Kick prediction	**0.2855 / 0.0949**	0.2957 (+3.5%) / 0.0991 (+4.4%)	0.2946 (+3.2%) / 0.0988 (+4.1%)

Ordered vs Plain

Table: Plain mode: logloss, zero-one loss and their relative change compared to Ordered mode

	Logloss	Zero-one loss
Adult	0.2723 (+1.1%)	0.1265 (-0.1%)
Amazon	0.1385 (-0.6%)	0.0435 (-1.5%)
Click prediction	0.3915 (-0.05%)	0.1564 (+0.19%)
Epsilon	0.2663 (+0.6%)	0.1096 (+0.9%)
KDD appetency	0.0718 (+0.5%)	0.0179 (+1.5%)
Kdd churn	0.2317 (-0.06%)	0.0717 (-0.17%)
KDD internet	0.2170 (+3.9%)	0.0987 (+5.4%)
KDD upselling	0.1664 (+0.1%)	0.0492 (+0.4%)
Kick prediction	0.2850 (-0.2%)	0.0948 (-0.1%)

Ordered vs Plain, effect of size

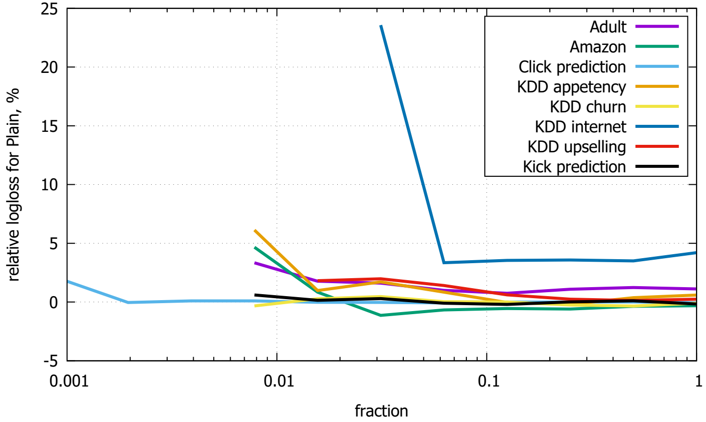


Figure: Relative error of Plain compared to Ordered depending on the fraction of the dataset

Number of permutations

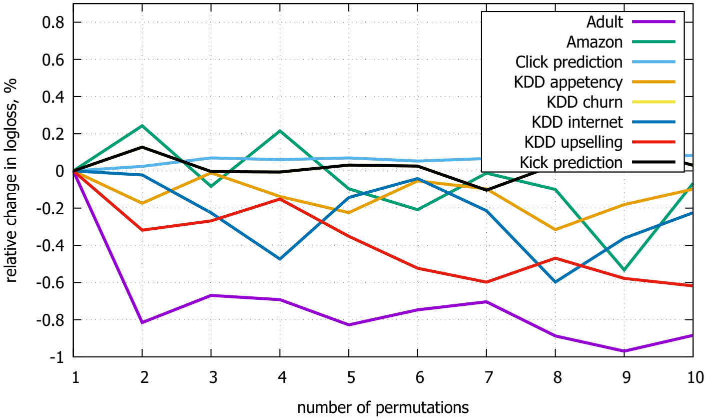


Figure: Relative change in logloss for a given number of permutations s compared to $s = 1$

Feature combinations

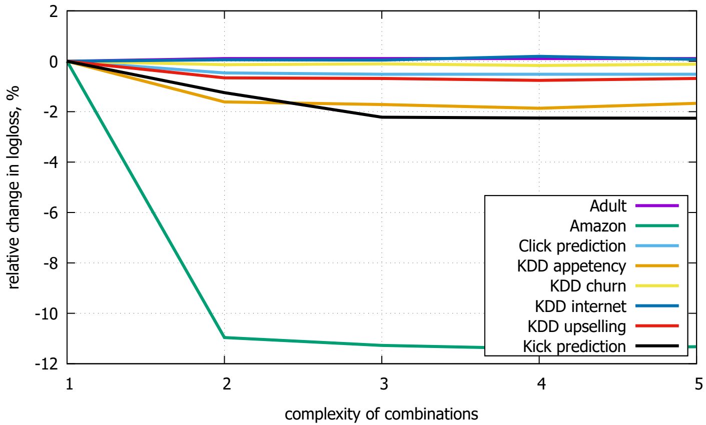


Figure: Relative change in logloss for a given allowed complexity compared to the absence of combinations

Yandex Research

Areas:

- Machine learning
- Computer Vision
- NLP
- Web Mining and Search
- Computational Economics

Conferences: NIPS, ICML, CVPR, ACL, KDD, SIGIR, etc.

https://research.yandex.com
https://yandex.ru/jobs/vacancies/research