

Яндекс



Библиотека градиентного бустинга

CatBoost

catboost / catboost

Unwatch

149

Unstar

2,418

Fork

308

Code

Issues45

Pull requests1

Insights

Settings

CatBoost is an open-source gradient boosting on decision trees library with categorical features support out of the box for Python, R <https://catboost.yandex>

Edit

machine-learning

decision-trees

gradient-boosting

gbm

gbdt

python

r

Manage topics

569 commits

3 branches

11 releases

46 contributors

Apache-2.0

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

arcadia-devtools Update generated

Latest commit 371359b 44 minutes ago

build

Pull-request for branch users/heretic/java-gen-script-support

44 minutes ago

catboost

move model_build_helper to proper place

44 minutes ago

ci

Fix

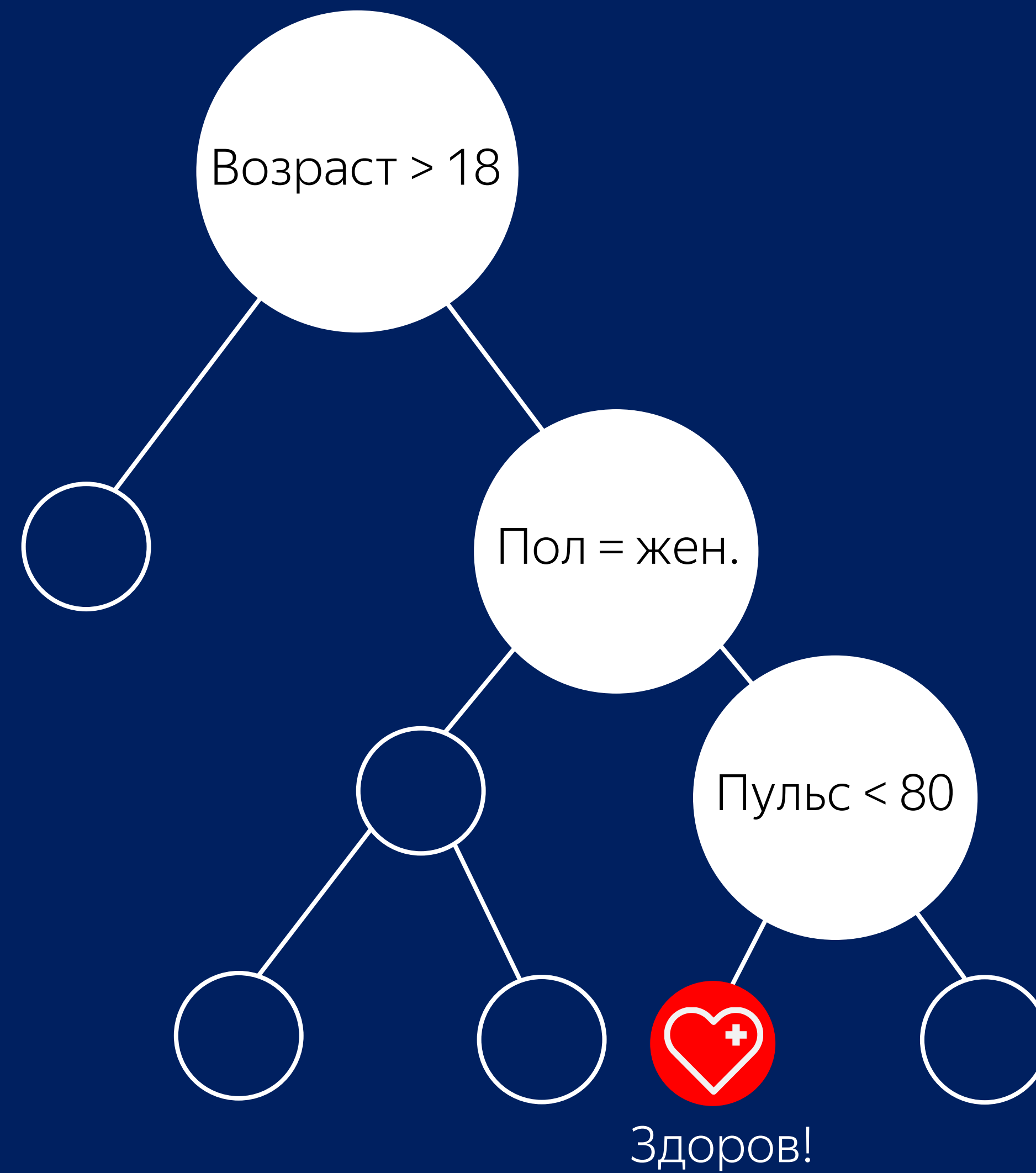
7 days ago

contrib

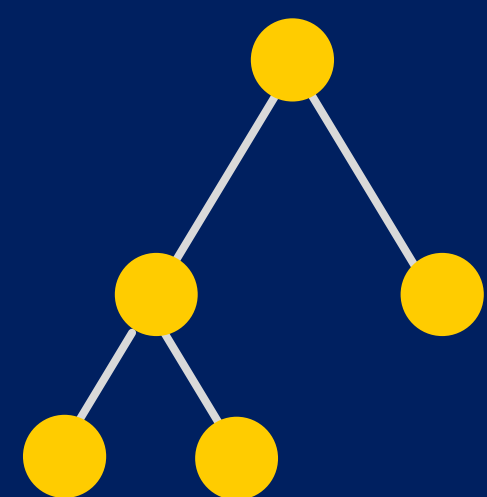
[rcutill] Suppress unaligned read errors reported by ubsan

2 days ago

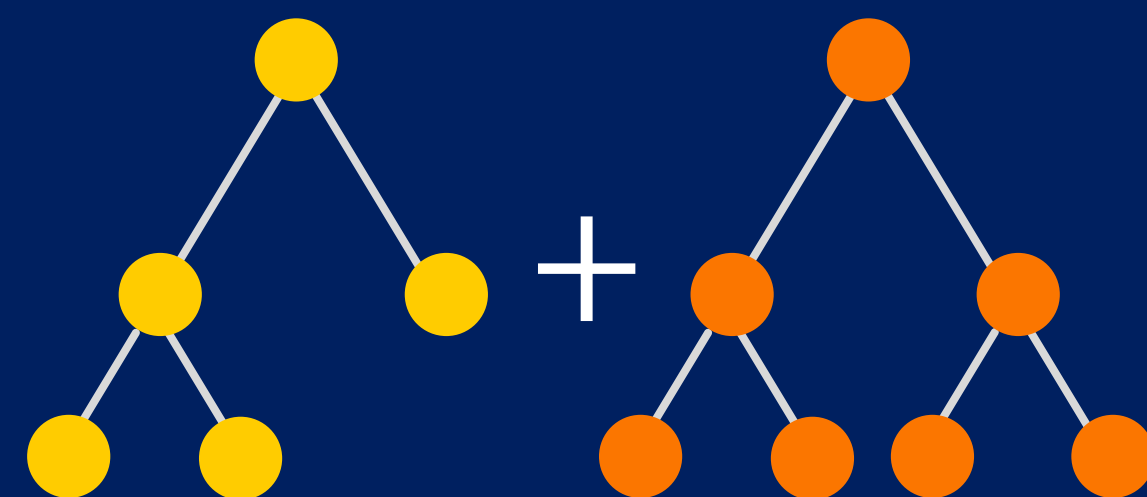
Дерево решений



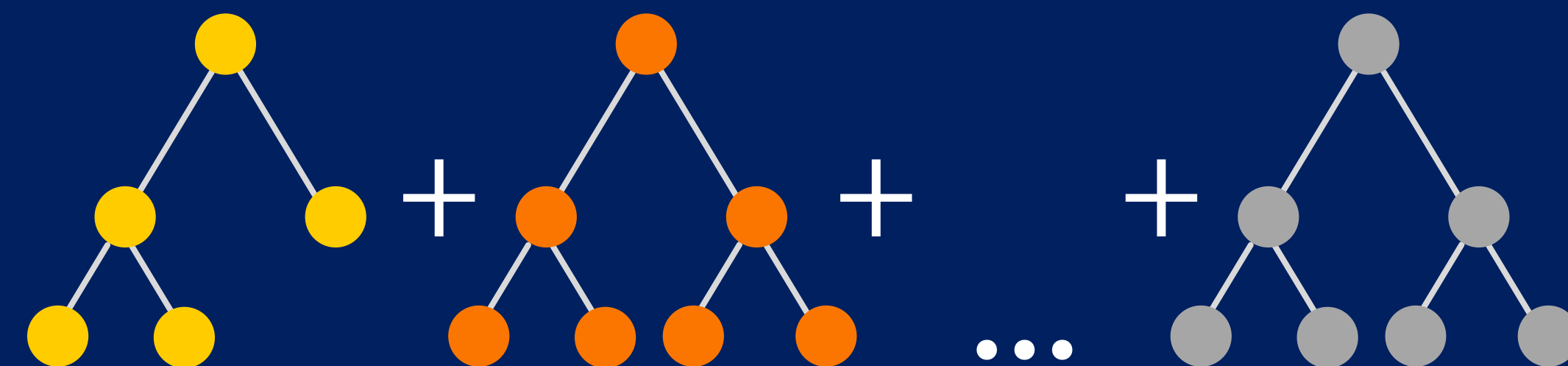
Градиентный бустинг



Ошибка



Ошибка



Ошибка

Режимы

- Регрессия
- Классификация
- Мультиклассификация
- Ранжирование
- Оптимизируемая функция
- Метрики

CatBoost Viewer

```
In [11]: model.fit(  
    X_train, y_train,  
    cat_features=categorical_features_indices,  
    eval_set=(X_validation, y_validation),  
    # verbose=True, # you can uncomment this for text output  
    plot=True  
)
```

× ☒ --- Learn ☒ — Test

Logloss Accuracy

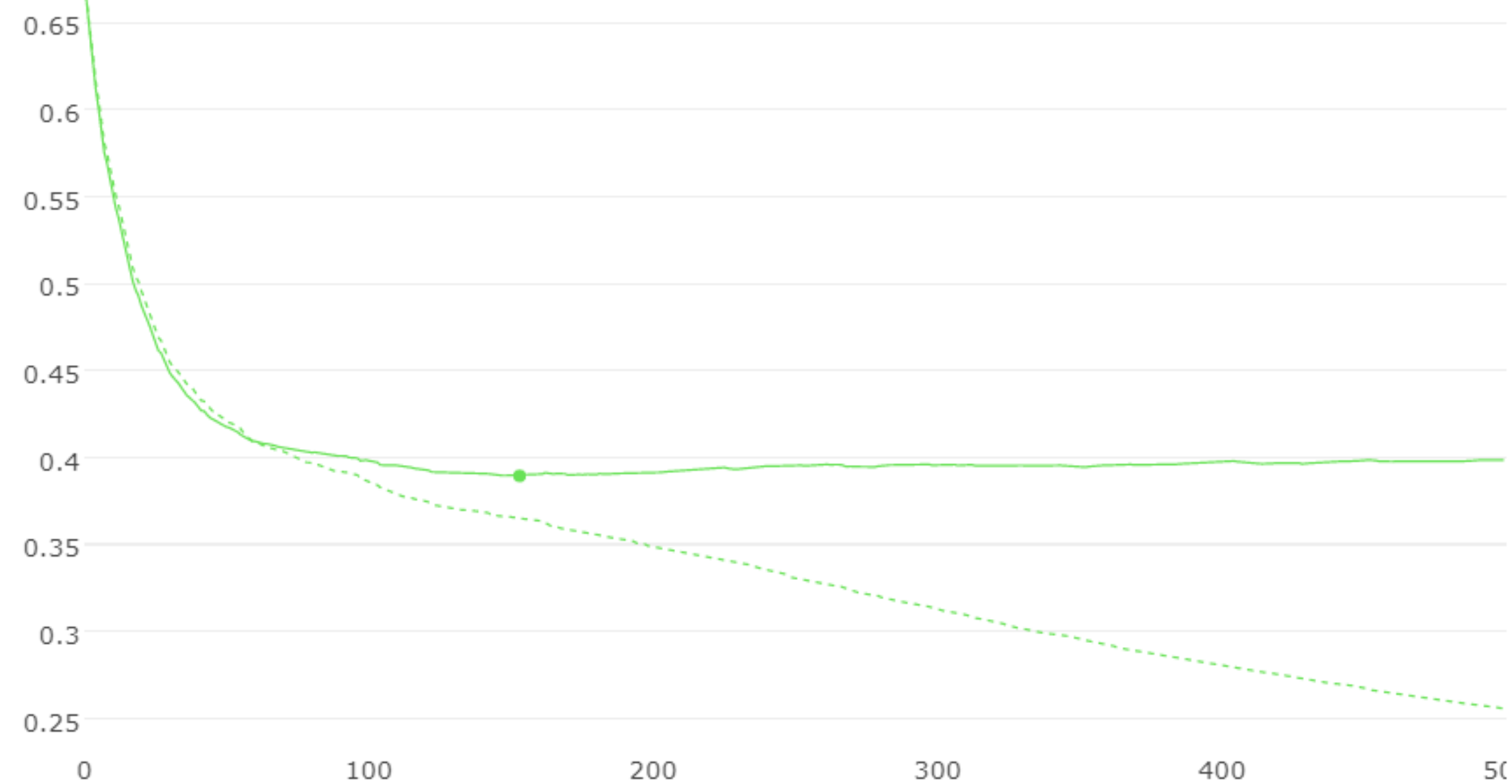
☒ current 19s
curr --- 0.2555652... — 0.3988282... 498
best — 0.3894004... 153

☐ Click Mode

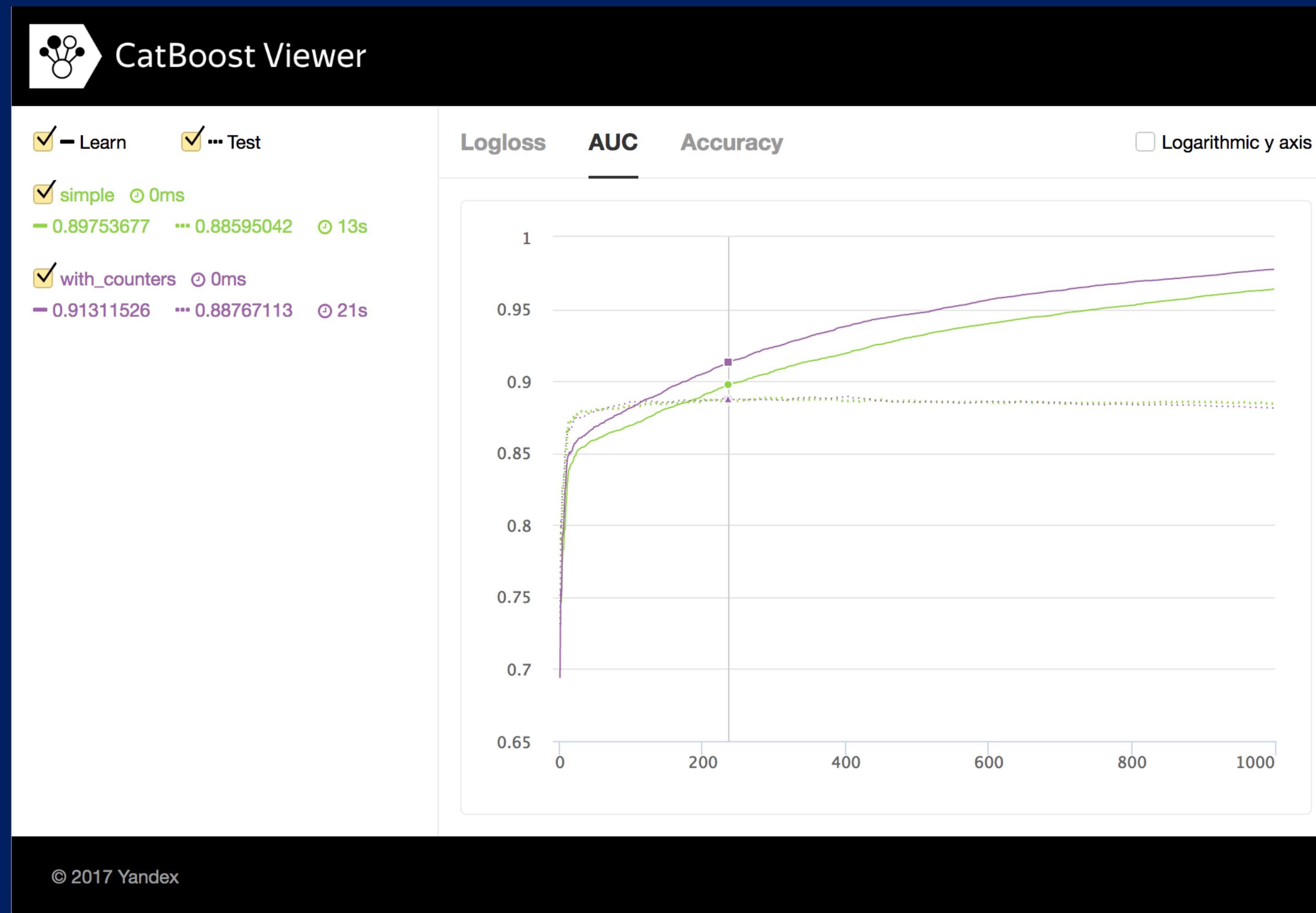
☐ Logarithm

☐ Smooth

0.5



CatBoost Viewer



TensorBoard

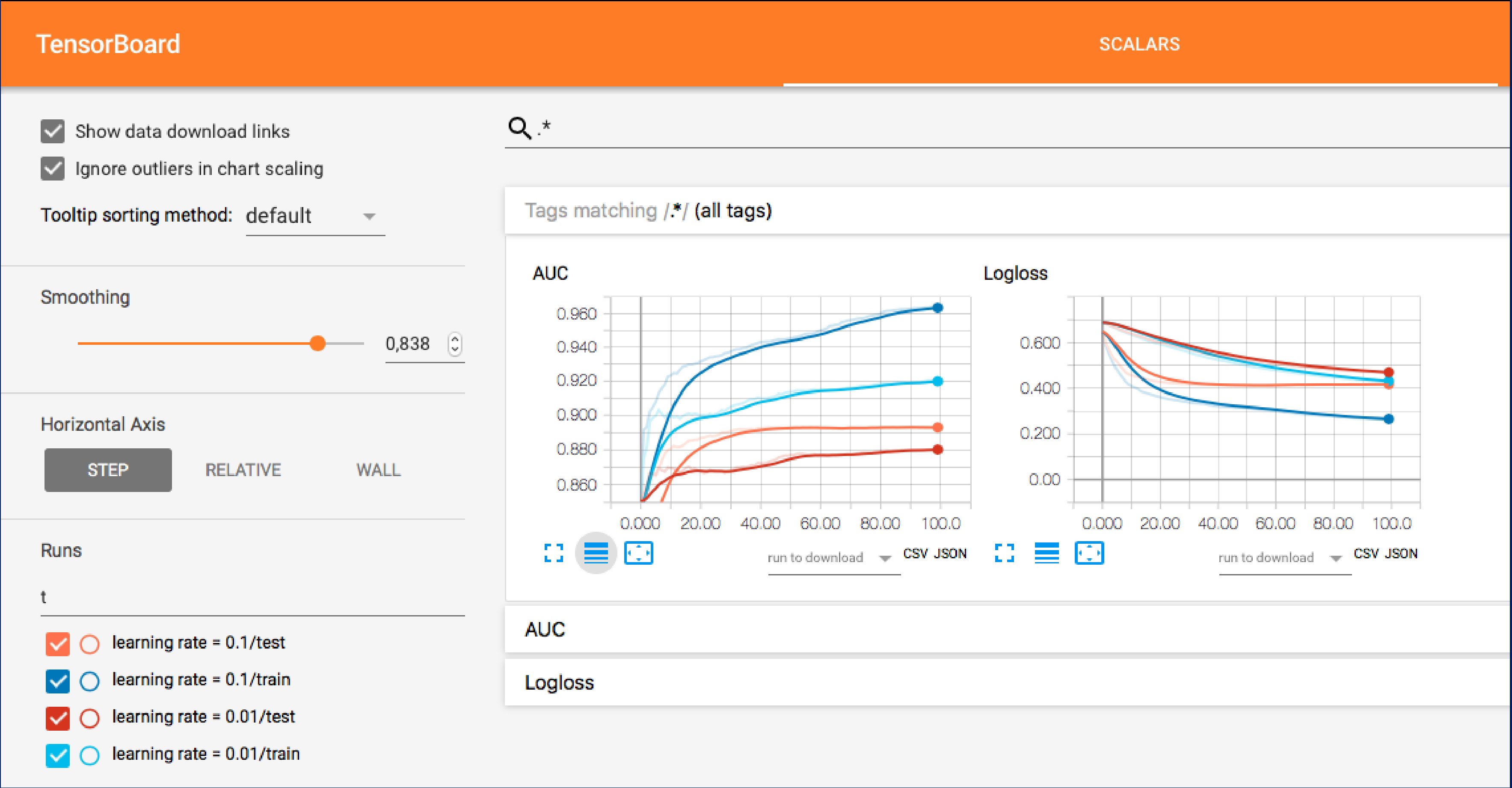


Схема обучения

1. Дерево приближает шаг по (анти-)градиенту

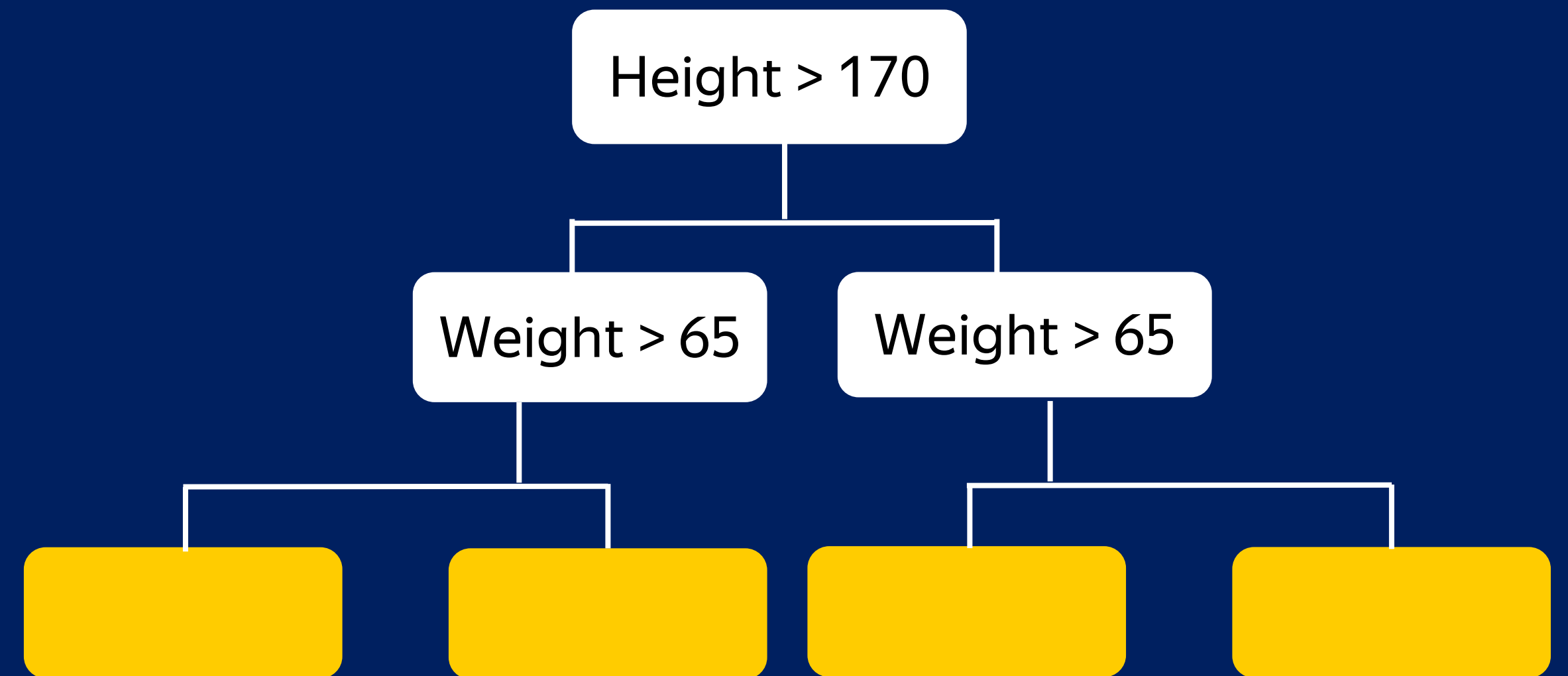
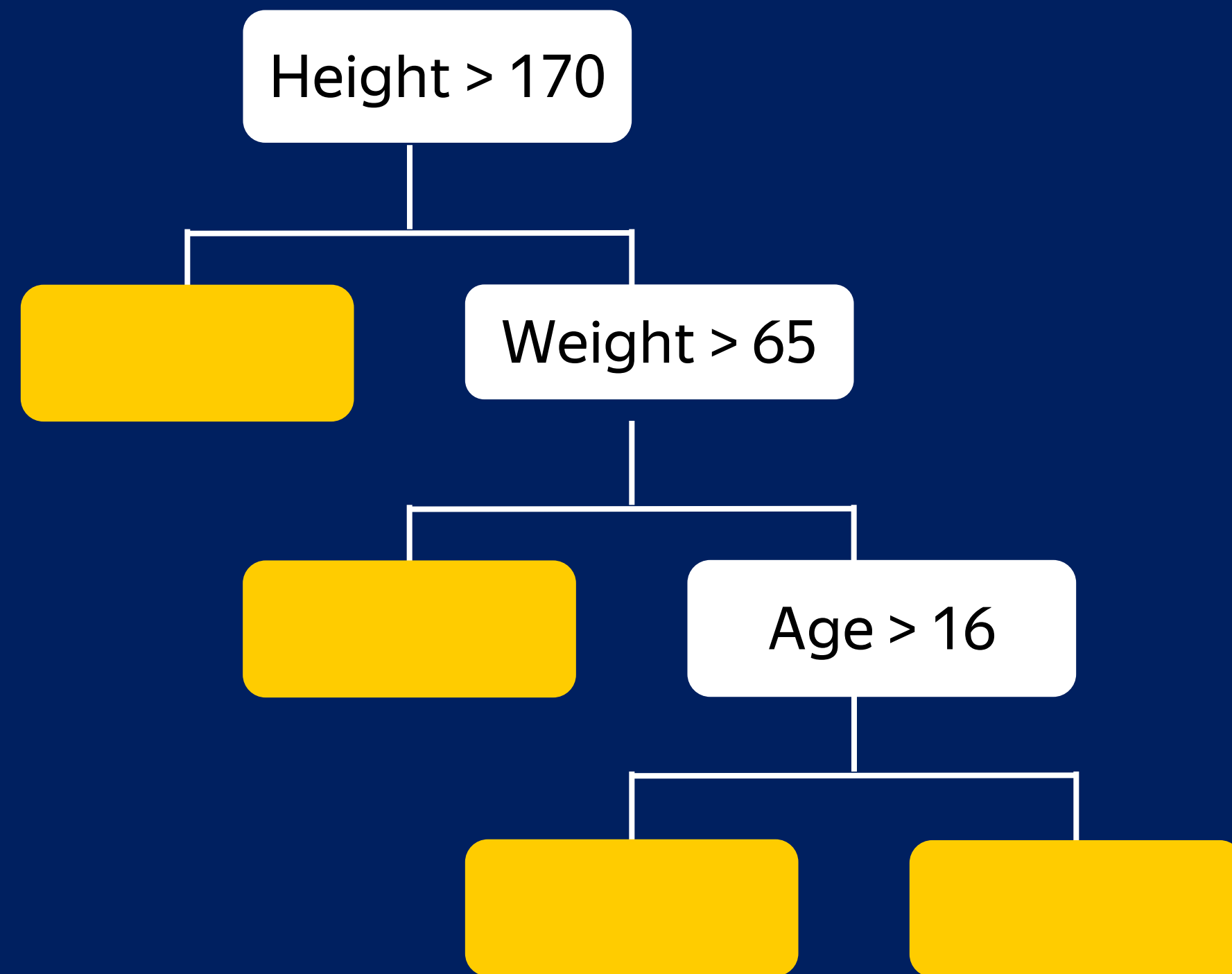
$\text{approx1}, \dots, \text{approxN}$ – значения формулы на документах.

$\text{Err}(\text{approx1}, \dots, \text{approxN}) \Rightarrow -(\text{Der1}, \dots, \text{DerN})$.

2. Жадное построение дерева

3. Подсчет значений в листьях

Symmetric trees



Скор сплита

$$\text{score(split)} = \frac{\sum_{doc} \text{leafValue}(doc) * \text{gradient}(doc) * w(doc)}{\sqrt{\sum_{doc} w(doc) * \text{leafValue}(doc)^2}}$$

$$\text{leafValue}(doc) = \frac{\text{sumWeightedDer}}{\text{sumWeights}}$$

Бутстрап

Бернулли:

$$w(doc) = 0 \text{ or } 1 \ (P(1) = sample_rate)$$

Байесовский:

$$w(doc) *= (-\log(rand(0,1)))^{bagging_temperature}$$

Только на этапе выбора структуры дерева

Рандомизация сора

$$score(f) += random_strength * N(0, RndMult * Sko)$$

Sko – длина вектора градиента

$RndMult$ – множитель, уменьшающийся при увеличении итерации

Бинаризация

Бинаризация плотных факторов



Uniform

Median

UniformAndQuantiles

MaxSumLog

GreedyLogSum

Сравнение бинаризаций

Равномерная сетка:

Только равномерная использует значения фичей


Остальные – только порядок документов

$$\sum W^2$$

Веса: 10,10,1000,1 – как лучше расставить 2 границы?

Бинаризация

Бинаризация счетчиков



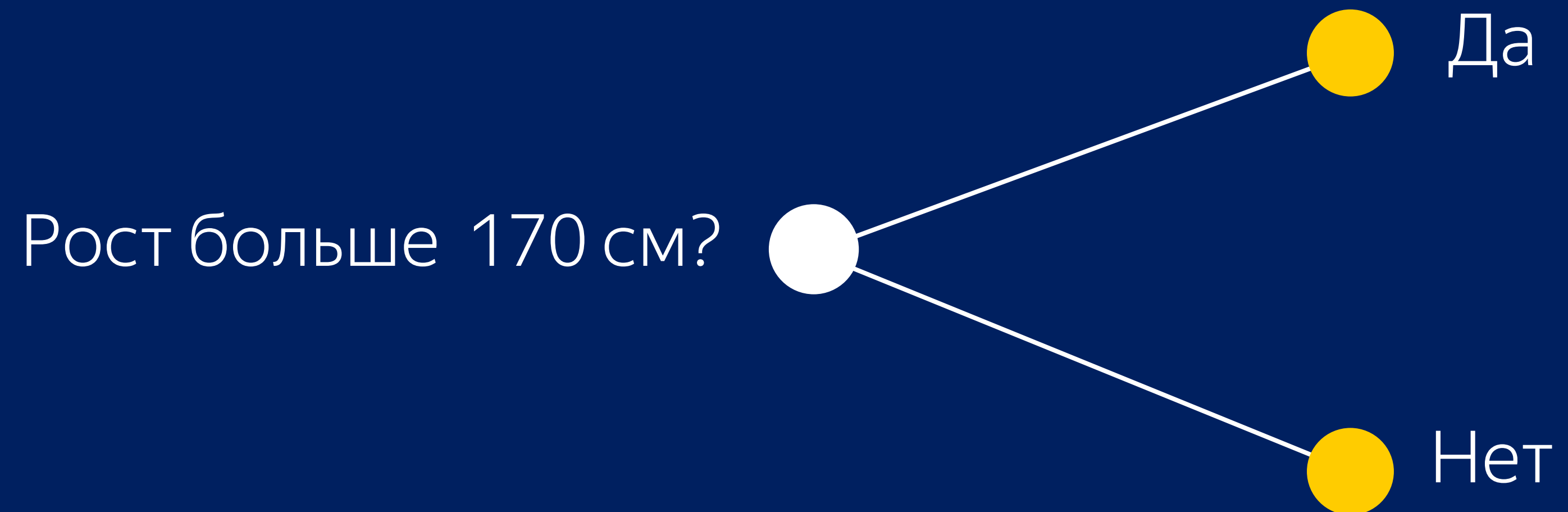
На CPU – Uniform

На GPU – Любая

Вычисление значений в листьях

1. Метод Ньютона или шаг по градиенту
2. Несколько шагов внутри одного дерева

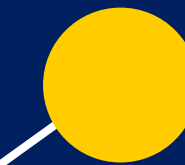
Числовые факторы



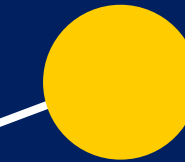
Категориальные факторы

Категориальные данные

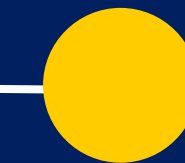
Виды облаков



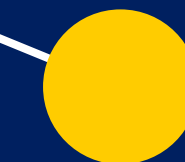
Перистые



Кучевые



Слоистые



Перламутровые



Мезосферные

Работа с категориальными факторами

1. Перенумеровать факторы
2. One-hot encoding
3. Хеширование в несколько корзин

Статистики по катфичам

1. Средний таргет по пулу (пример – один объект с такой фичой)
2. Leave one out (пример – «кот»: 4 успеха, 7 неуспехов. one-hot + ctr = 0.3 и 0.4)
3. Leave bucket out

Такие способы ведут к переобучению

Статистики по катфичам

1. Средний таргет на отложенной выборке

При таком способе меньше данных для обучения и вычисления статистик

Статистики по катфичам

1. Статистики по прошлому в перестановке
2. Несколько перестановок
3. Комбинации факторов

Типы статистик

1. С учетом таргета (CTR)
2. Без учета таргета (Counter)

1. Классификация
2. Регрессия
3. Мультиклассификация

Статистики для бинарной классификации

1. CTR

$$Ctr = \frac{\#Positive + Prior}{\#All + 1}$$

Статистики для регрессии и мультикласса

$$Ctr = \frac{\#CountInClass + Prior}{\#All + 1}$$

1. Borders

- › CountInClass – Число объектов с таргетом больше границы

2. Buckets

- › CountInClass – Число объектов в бакете

3. MeanValue

- › CountInClass - Суммарный таргет

Вычисление статистик при применении

1. CTR

Объект теста дописывается к обучающей выборке.

2. Counter

1. Каунтеры по всему тесту

2. Объект теста дописывается к лерну

One-hot encoding

1. Имеет смысл пробовать, если у катфичи мало значений.

По дефолту используется для катфичей с 2 значениями.

1. Не нужно делать самим. Нужно использовать `one_hot_max_size`

Переобучение в классическом бустинге

1. Оценка градиента для каждого документа делается при помощи модели, обученной с использованием данного документа.

Ordered boosting

1. Квадратичная схема
2. Линейная схема
3. Линейное упрощение квадратичной схемы

Обучение из бейзлайна

1. Не то же самое, что продолжение обучения

Подбор параметров

1. Выбор категориальных факторов

Подбор параметров

1. Число итераций + learning_rate
2. Детектор переобучения

Подбор параметров

1. L2-регуляризация
2. `random_strength`
3. `bagging_temperature`

Подбор параметров

1. Глубина дерева
2. Размер бинаризации

Подбор параметров

1. rsm

Подбор параметров

1. Вычисление значений в листьях

1. Newton vs Gradient

2. Число шагов по градиенту

Полезная функциональность

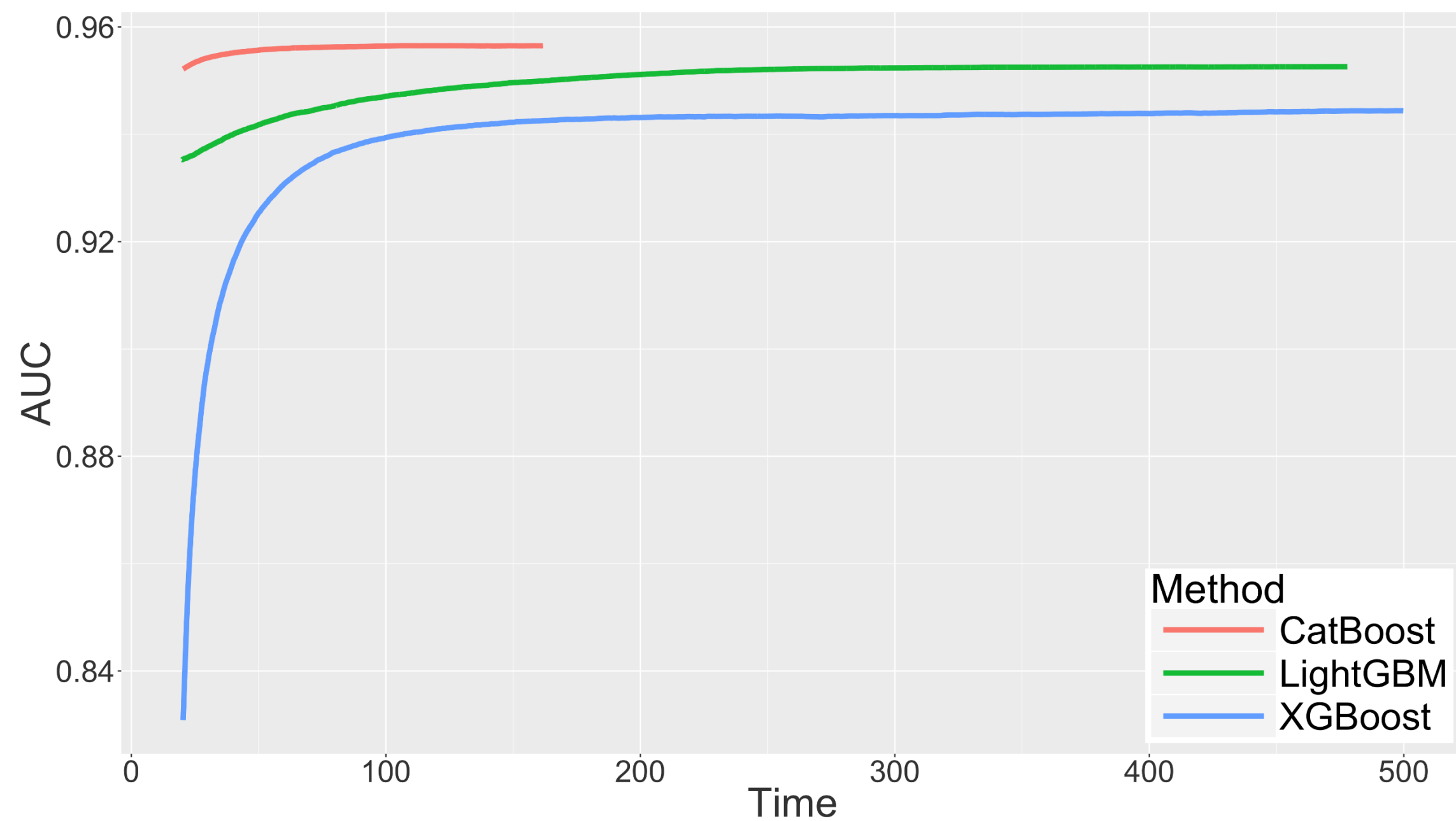
1. `staged_predict()` и `eval_metrics()`
2. `cv()`
3. `snapshots`

CPU vs GPU

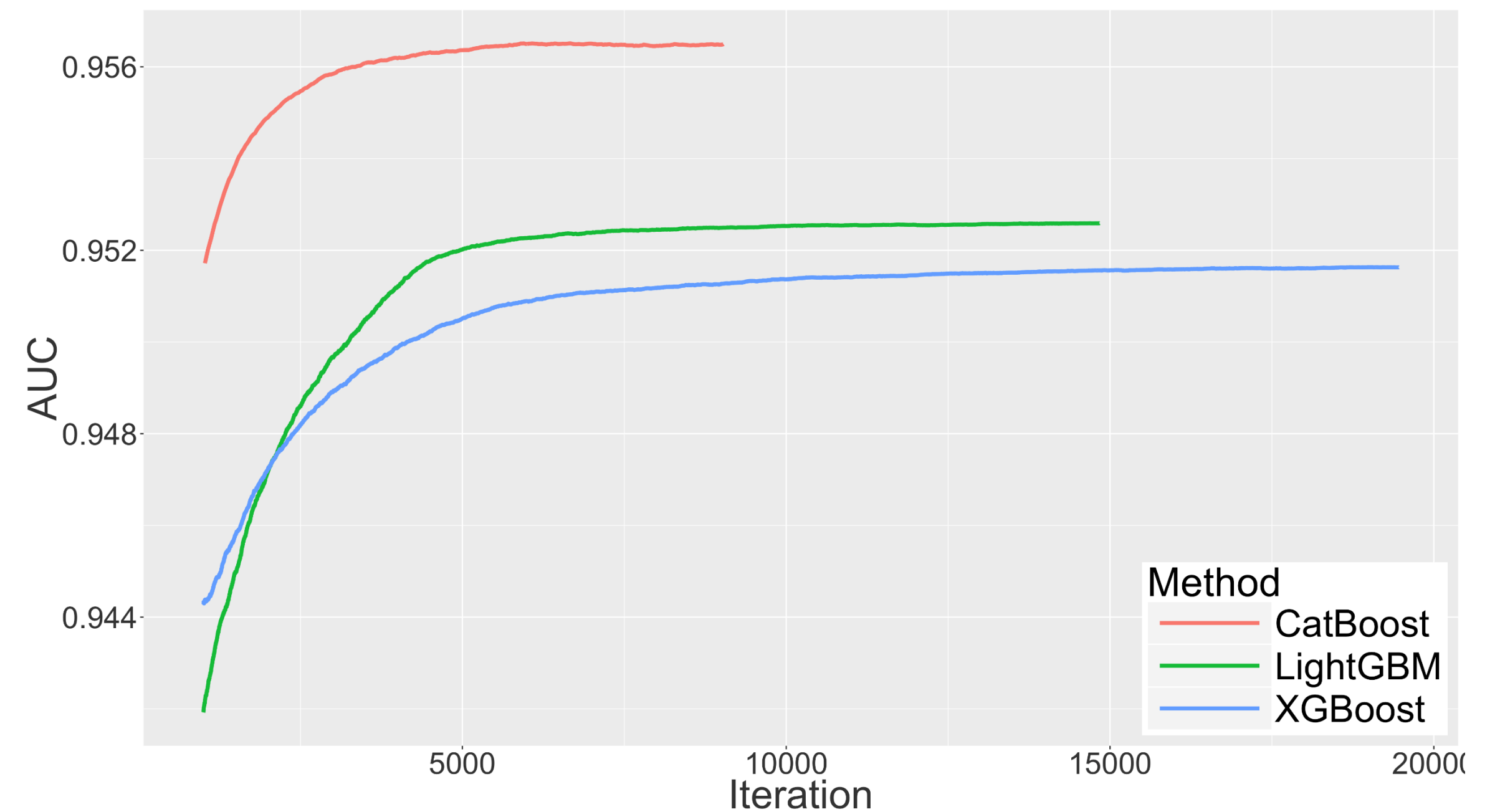
Epsilon dataset	128 bins	32 bins
CPU	713 sec (1.0x)	653 sec (1.0x)
K40	547 sec (1.3x)	248 sec (2.6x)
GTX 1080	194 sec (3.67x)	120 sec (5.4x)
P40	162 sec (4.4x)	91 sec (7.1x)
GTX 1080Ti	145 sec (4.9x)	88 sec (7.4x)
P100-PCI	127 sec (5.6x)	70 sec (9.3x)
V100-PCI	77 sec (9.25x)	49 sec (13.3x)

Criteo dataset	128 bins
CPU	1060 sec (1.0x)
K40	373 sec (2.84x)
GTX 1080Ti	301 sec (3.5x)
GTX 1080	285 sec (3.7x)
P40	123 sec (8.6x)
P100-PCI	82 sec (12.9x)
V100-PCI	69.8 sec (15x)

GPU: Comparison with other libraries



AUC vs Training time



AUC vs Training iterations

Сравнение с другими библиотеками

	CatBoost	LightGBM		XGBoost		H2O	
Adult	0.269741	0.276018	+ 2.33 %	0.275423	+ 2.11%	0.275104	+ 1.99%
Amazon	0.137720	0.163600	+ 18.79 %	0.163271	+ 18.55%	0.162641	+ 18.09%
Appet	0.071511	0.071795	+ 0.40 %	0.071760	+ 0.35%	0.072457	+ 1.32%
Click	0.390902	0.396328	+ 1.39 %	0.396242	+ 1.37%	0.397595	+ 1.71%
Internet	0.208748	0.223154	+ 6.90 %	0.225323	+ 7.94%	0.222091	+ 6.39%
Kdd98	0.194668	0.195759	+ 0.56 %	0.195677	+ 0.52%	0.195395	+ 0.37%
Kddchurn	0.231289	0.232049	+ 0.33 %	0.233123	+ 0.79%	0.232752	+ 0.63%
Kick	0.284793	0.295660	+ 3.82 %	0.294647	+ 3.46%	0.294814	+ 3.52%

Logloss

Информация

- › <https://github.com/catboost/catboost>
- › <https://catboost.yandex/>
- › <https://tech.yandex.com/catboost/doc/dg/concepts/about-docpage/>
- › <https://twitter.com/CatBoostML>

Если хочется поучаствовать

› <https://github.com/catboost/catboost/issues>

Issues с тегами «help wanted», «good first issue»

› Стажировки:

annaveronika@yandex-team.ru