



CatBoost и Python: как быстро учить и применять CatBoost модели в Python приложениях

Кириллов Станислав, ведущий разработчик Yandex

Code

Issues 151

Pull requests 4

Security

Insights

Settings

A fast, scalable, high performance Gradient Boosting on Decision Trees library, used for ranking, classification, regression and other machine learning tasks for Python, R, Java, C++. Supports computation on CPU and GPU. <https://catboost.ai>

Edit

machine-learning decision-trees gradient-boosting gbm gbdt python r kaggle gpu-computing catboost tutorial
categorical-features gpu coreml data-science big-data cuda data-mining

Manage topics

● C++ 80.5% ● Python 12.5% ● Cuda 4.5% ● R 0.6% ● Makefile 0.6% ● Java 0.4% ● Other 0.9%

Branch: master

New pull request

Create new file

Upload files

Find File

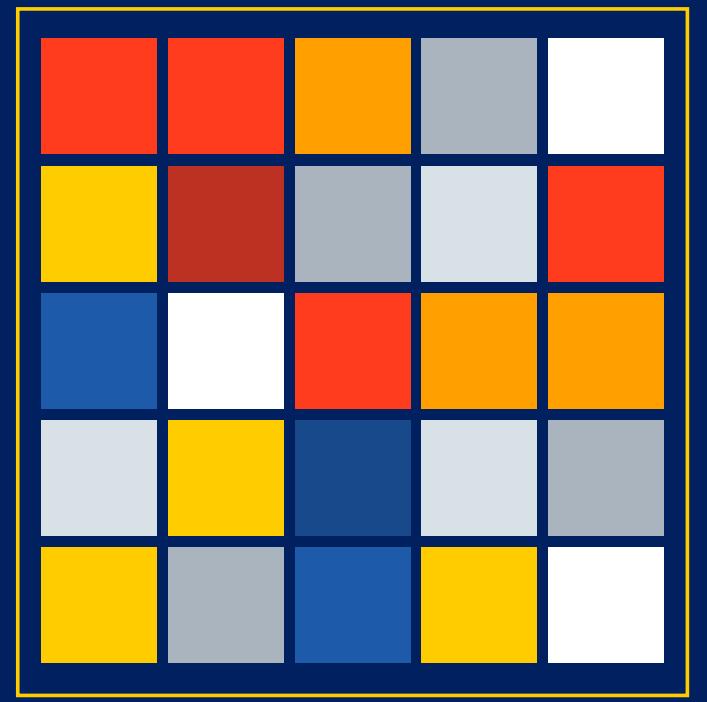
Clone or download

 slon	Replace diffing library. . .	Latest commit f0e40f2 17 minutes ago
 build	Replace diffing library. . .	17 minutes ago
 catboost	Added the ability to save plot from calc_feature_statistics to a file.	4 hours ago
 ci	Fix OpenSSL paths	last month
 contrib	Workaround for MSVC compiler bug.	5 hours ago
 library	intermediate changes	3 days ago
 make	Solution files and make files updated	3 days ago
 msvs	Solution files and make files updated	3 days ago

План доклада

- Решающие деревья и CatBoost. Чем же он так хорош? 😊
- Как устроена Cython обертка
- Как готовить данные для быстрого обучения и применения

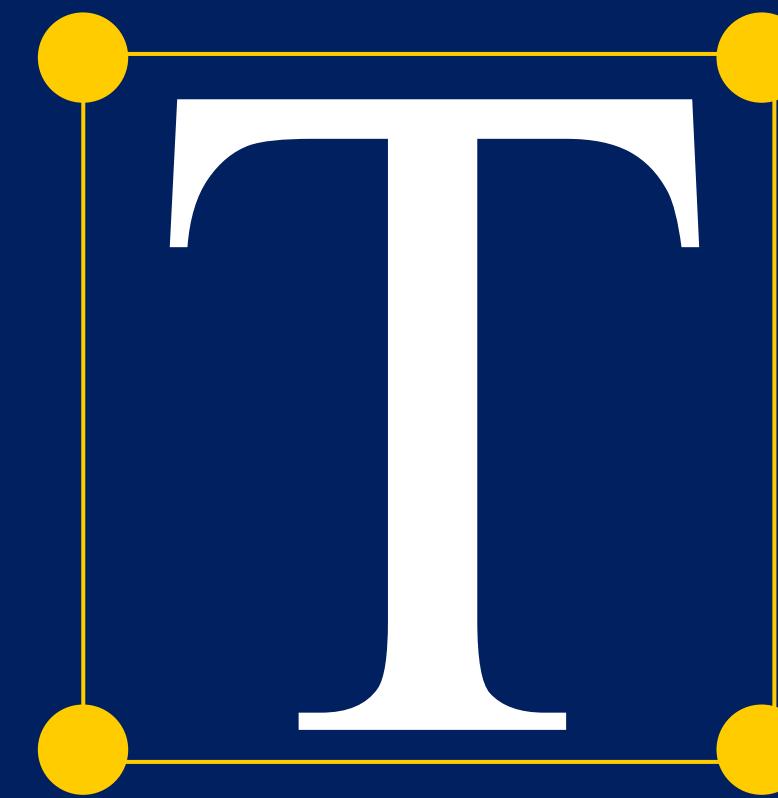
Нейронные сети?



Картинки

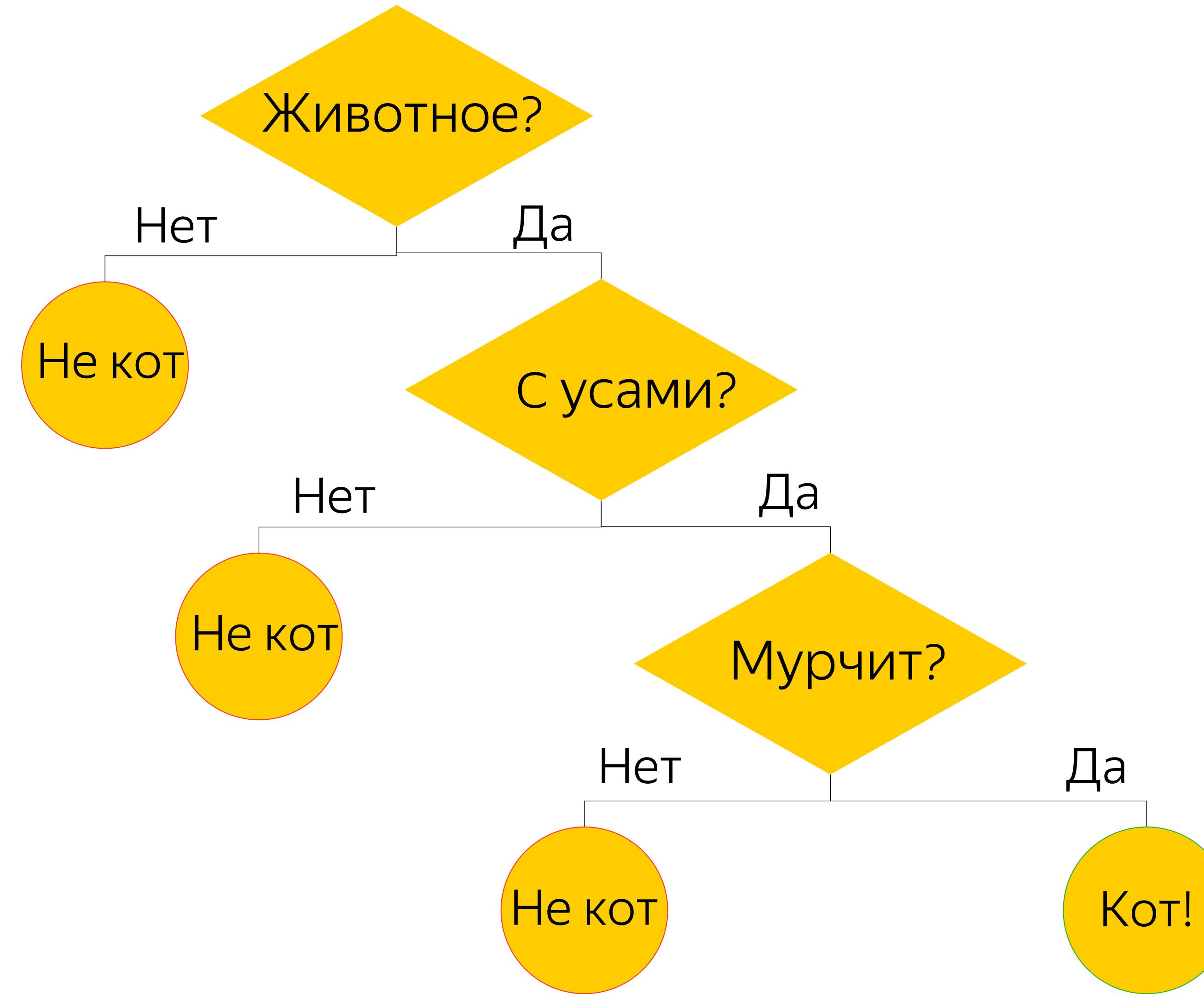


Звук

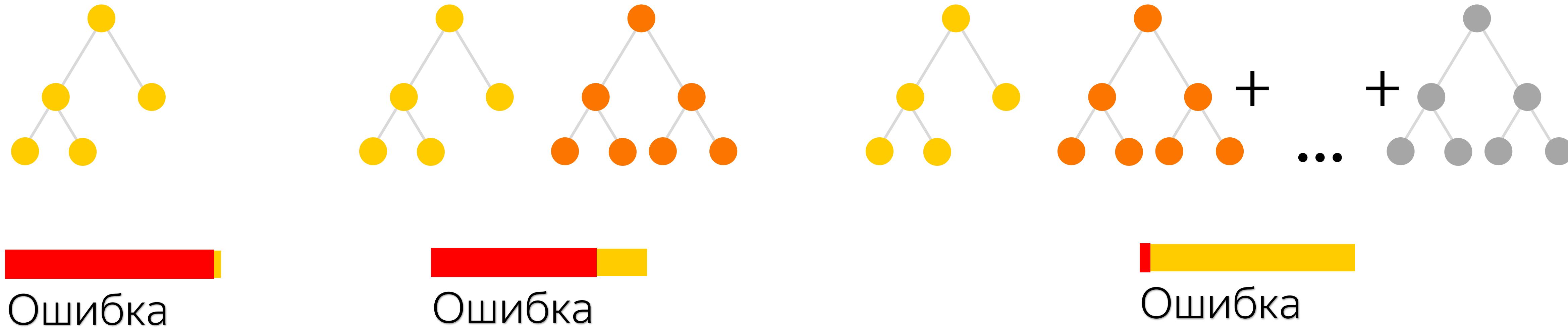


Текст

Решающие деревья



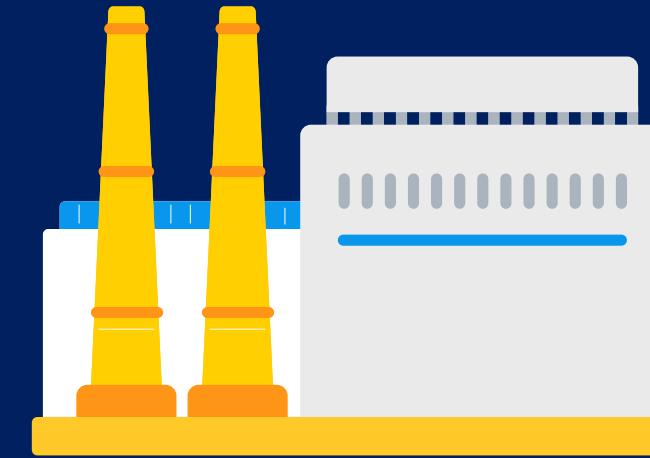
Градиентный бустинг



Применения CatBoost



Медицина



Промышленность



Финансы



Рекомендательные
системы



Предсказания
продаж



Конкурсы

Ключевые особенности

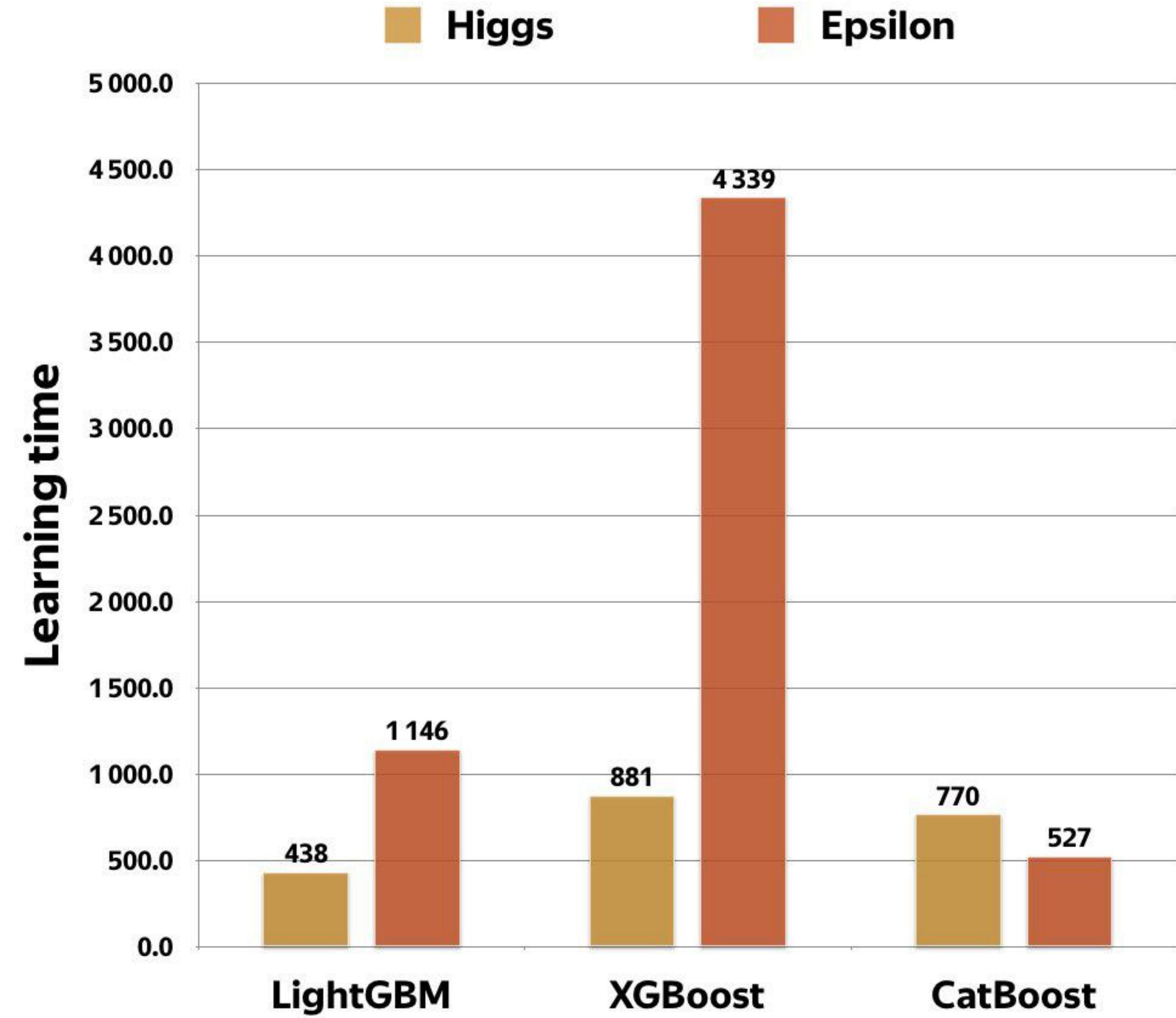
- Хорошее качество с параметрами по умолчанию
- Поддержка категориальных признаков
- Скорость обучения
- Скорость применения моделей
- Инструменты анализа моделей

Сравнение с другими библиотеками

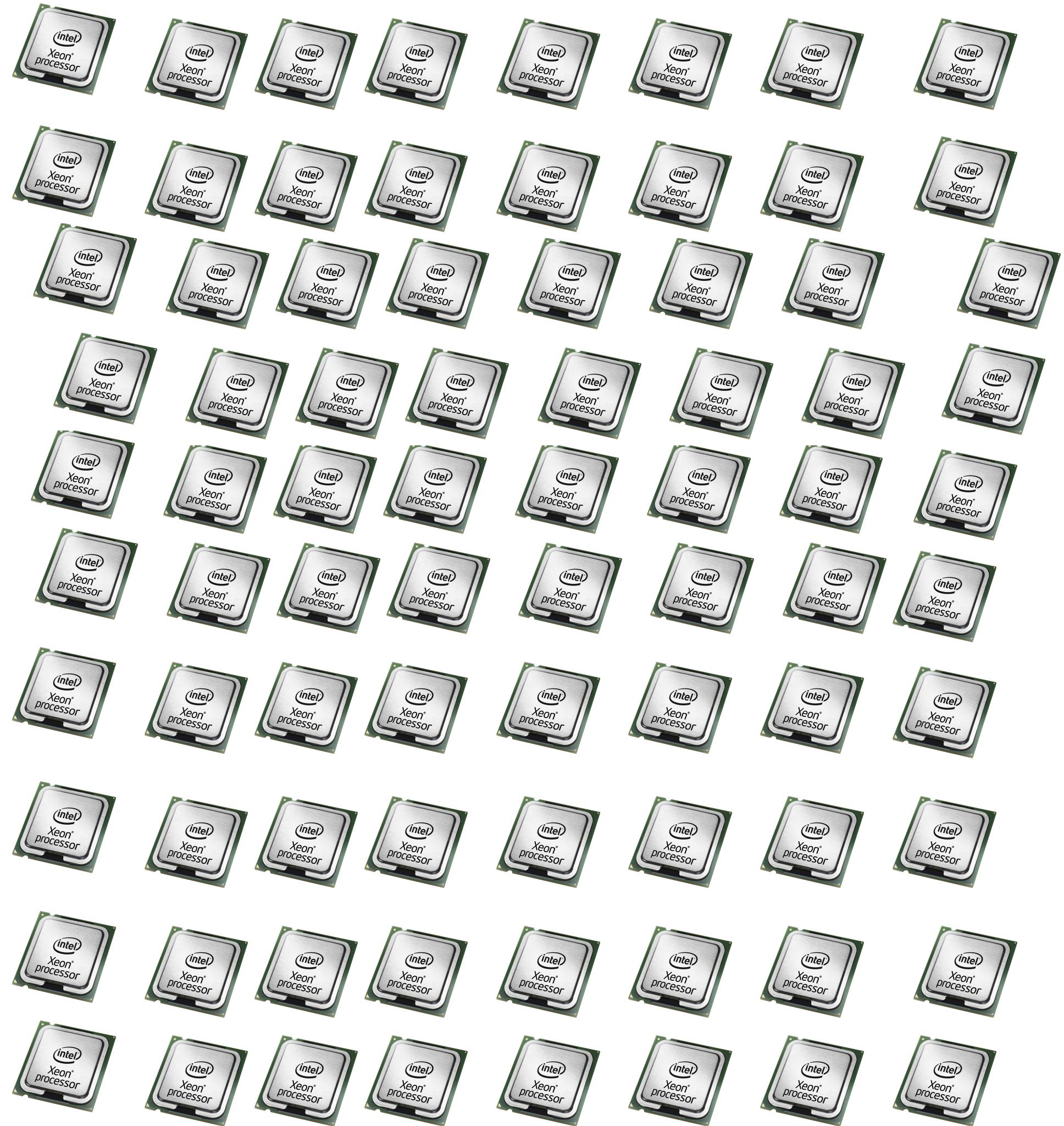
	CatBoost	LightGBM	XGBoost	H2O
Adult	0.269741	0.276018 + 2.33 %	0.275423 + 2.11%	0.275104 + 1.99%
Amazon	0.137720	0.163600 + 18.79 %	0.163271 + 18.55%	0.162641 + 18.09%
Appet	0.071511	0.071795 + 0.40 %	0.071760 + 0.35%	0.072457 + 1.32%
Click	0.390902	0.396328 + 1.39 %	0.396242 + 1.37%	0.397595 + 1.71%
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%

CPU: Скорость обучения

- Параметры:
квантизация на 128 бинов, 64 листа
в дереве, 400 деревьев
- Higgs:
28 фич, 10.5M объектов
- Epsilon:
2000 фич, 400K объектов



Производительность GPU



≈

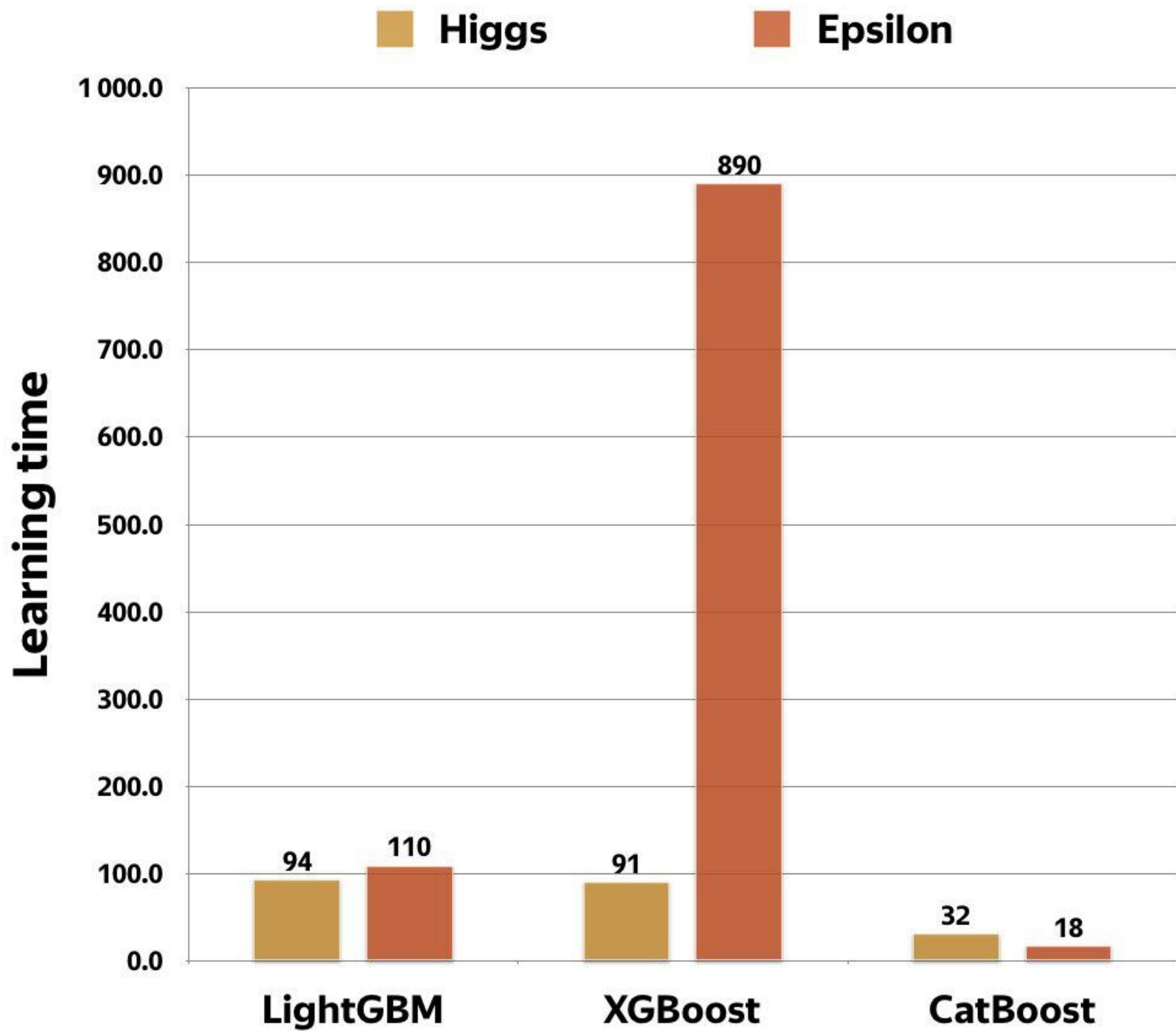


NVIDIA Titan V

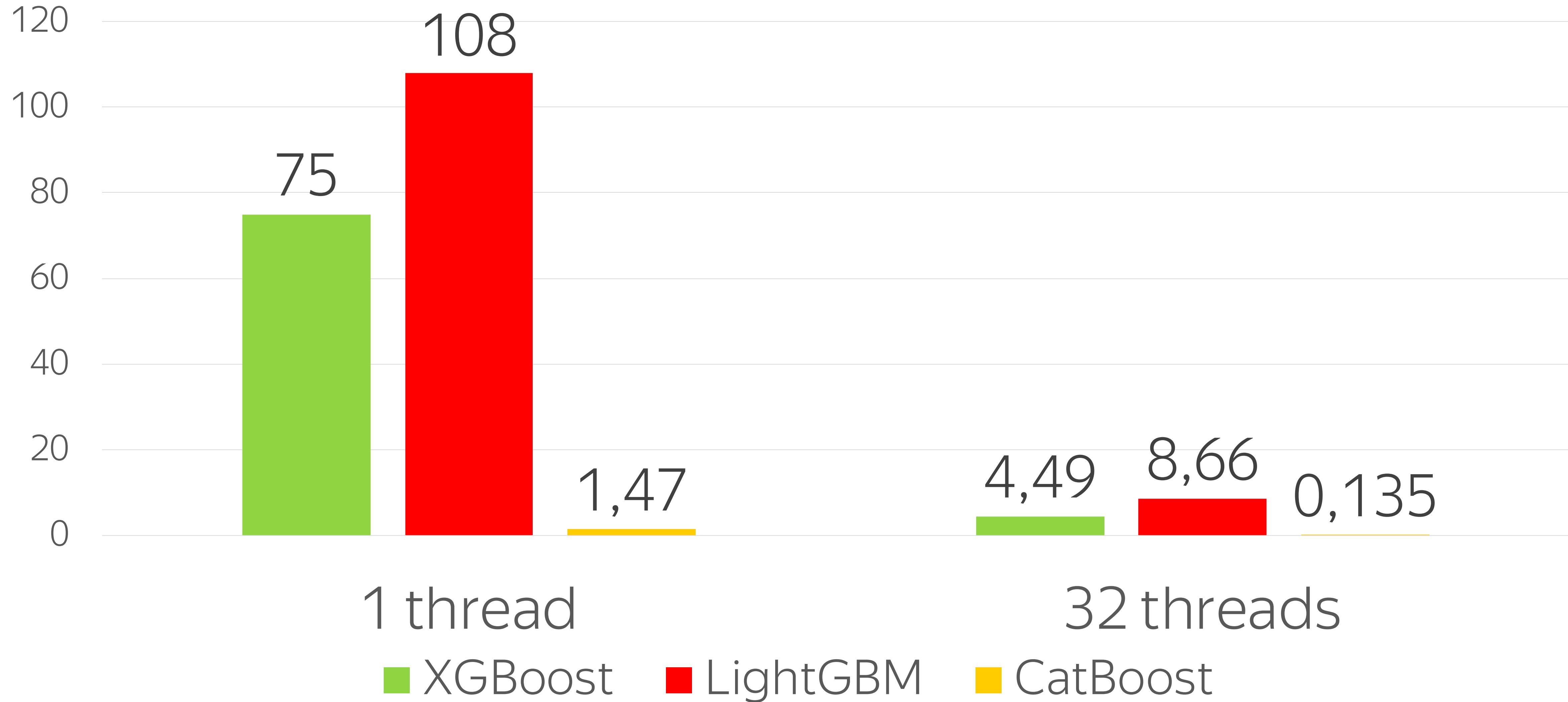
80x Intel Xeon E5-2660v4

GPU: Скорость обучения

- Параметры:
квантизация на 128 бинов, 64 листа
в дереве, 400 деревьев
- Higgs:
28 фич, 10.5М объектов
- Epsilon:
2000 фич, 400К объектов



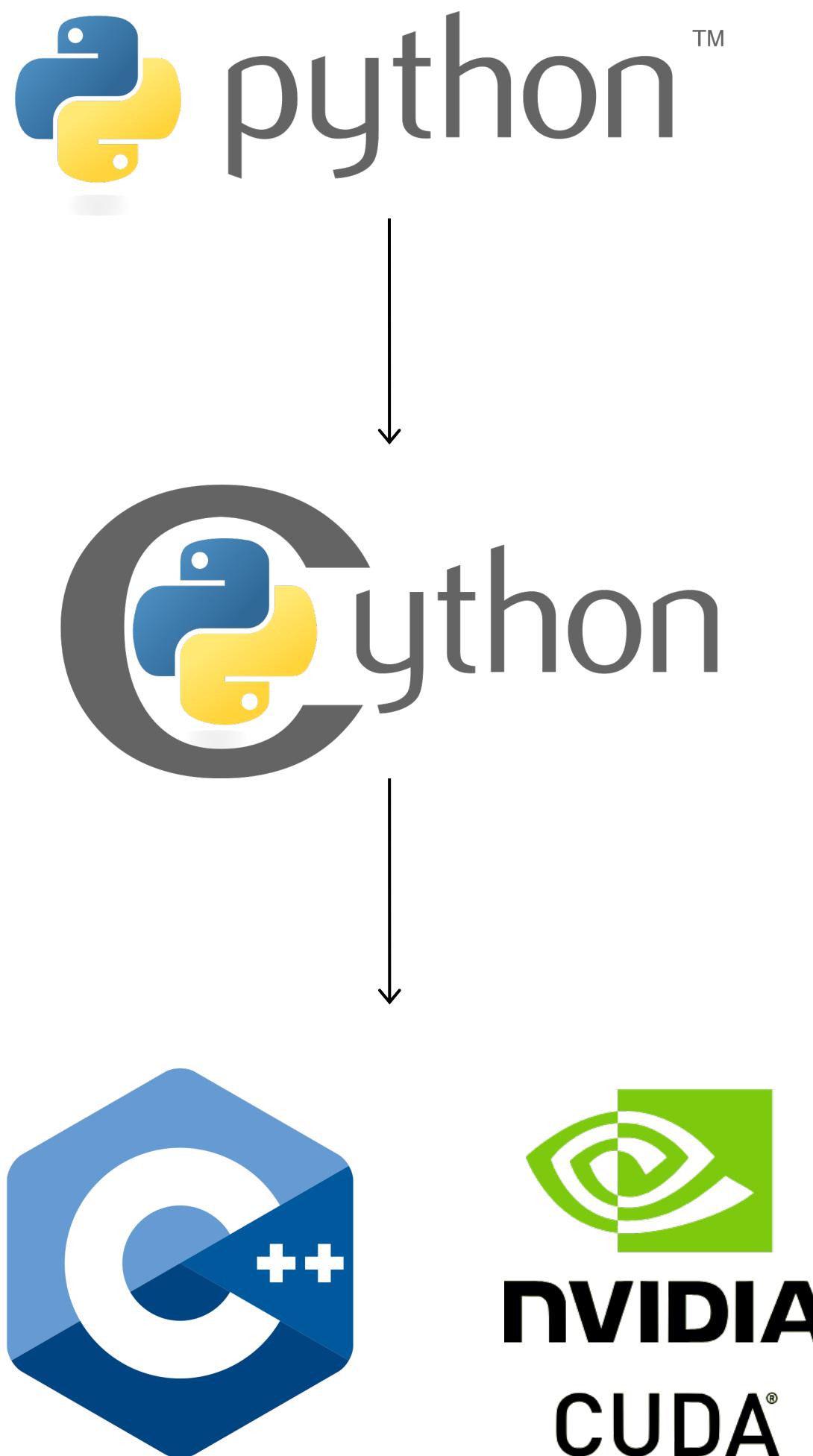
Скорость применения на Epsilon



В ближайших релизах

- Поддержка текстовых фичей
- Поддержка разреженных данных на CPU и GPU
- Применение моделей на GPU
- Monotonic constraints
- Сравнение/объяснение предсказаний для пары объектов

Как устроена Python библиотека CatBoost



Интерфейсы обучения и загрузки
данных

Подготовка и преобразование данных

Главный цикл обучения

Почему мы выбрали Cython

Плюсы:

Простота интеграции C++ кода в Python

Скорость и эффективность

Удобно поддерживать Python2 & Python3 совместимость

MemoryView – скорость и эффективность

Пример функции, которая принимает на вход двумерный вектор и указатель на данные передает в C++:

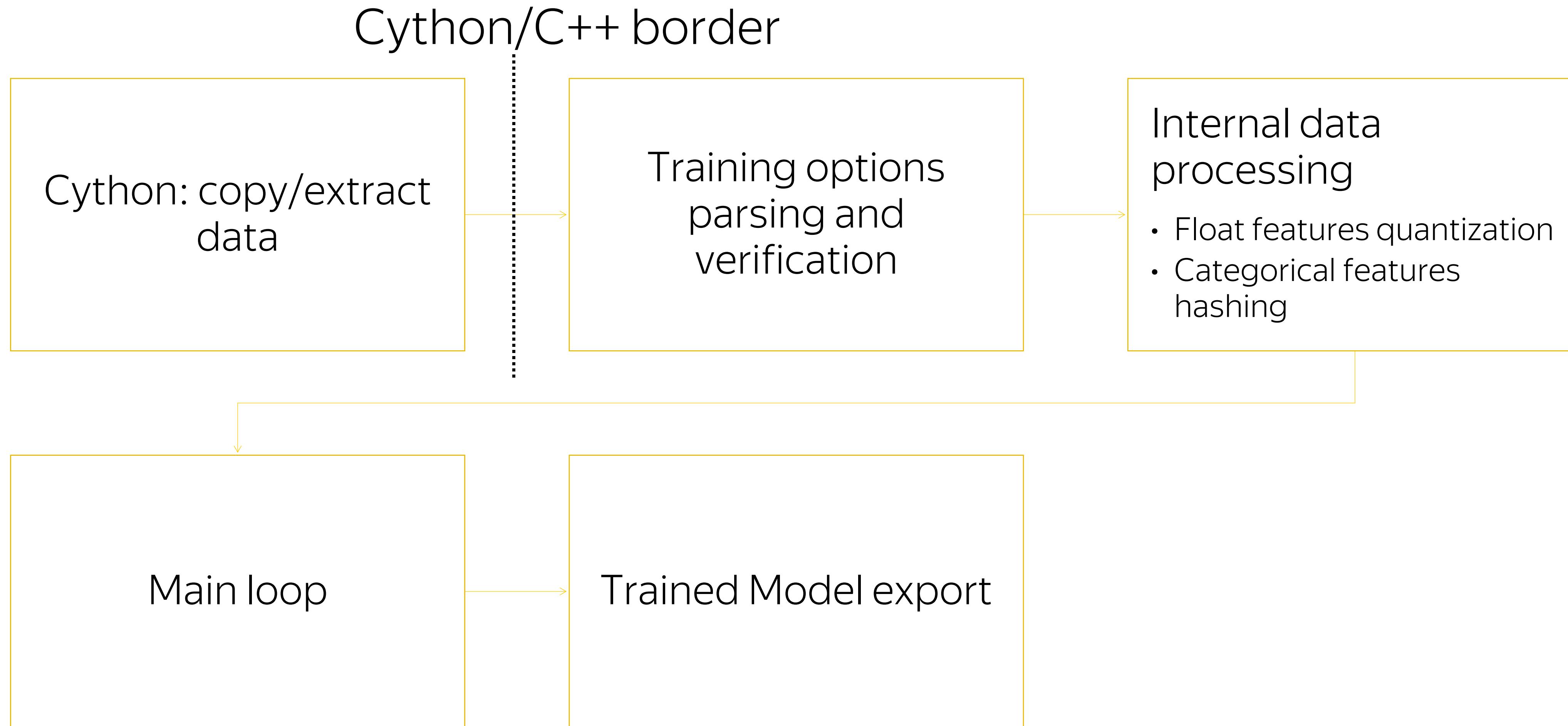
```
cdef _set_features_order_data_np(  
    const float [:,:] num_feature_values,  
    IFeatureConsumeCtx* ctx  
) :  
    ctx[0].FuncOnColumn(  
        &num_feature_values[0, num_feature_idx],  
        num_feature_idx  
)
```

Боль работы со строками

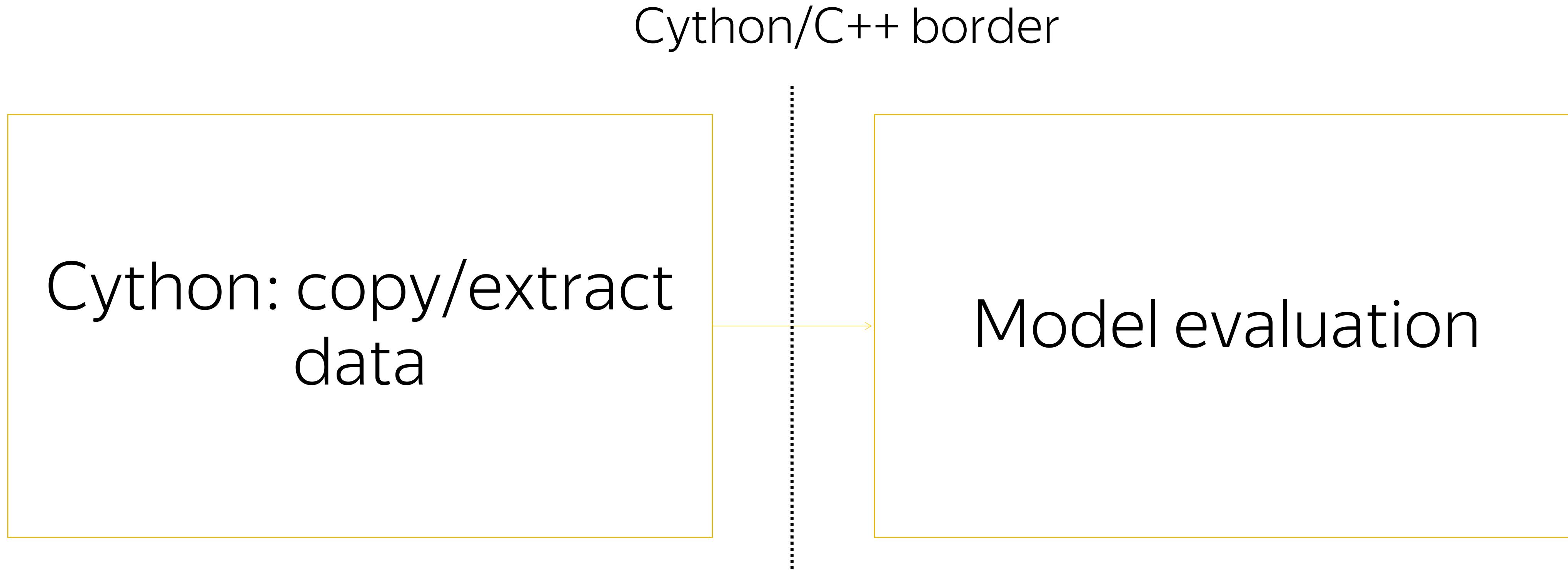
```
cdef TString to_arcadia_string(s):
    cdef const unsigned char[:] bytes_s
    cdef type s_type = type(s)
    if len(s) == 0:
        return TString()
    if s_type is unicode:
        # Fast path for most common case(s).
        tmp = (<unicode>s).encode('utf8')
        return TString(<const char*>tmp, len(tmp))
    elif s_type is bytes:
        return TString(<const char*>s, len(s))

    if PY3 and hasattr(s, 'encode'):
        # encode to the specific encoding used inside of the module
        bytes_s = s.encode()
    else:
        bytes_s = s
return TString(<const char*>&bytes_s[0], len(bytes_s))
```

Процесс обучения



Процесс применения обученной модели



Время обучения

Время обучения состоит из трех частей:

- Время копирования/извлечения данных в Cython
- Время внутренней обработки данных – зависит только от входных данных
- Время на подбор деревьев и значений в листьях

Уменьшаем время подбора деревьев

Ключевые моменты:

- Правильный выбор количества итераций
 - Подходящий тип бустинга (Ordered медленнее чем Plain)
 - Подходящее количество сплитов float фичей (иногда 128/254 избыточны)
 - Пробуйте random subspace method (сэмплирование фичей на дерево)
- Полная инфа <https://catboost.ai/docs/concepts/speed-up-training.html>

Снижаем накладные расходы

Используем структуры данных с возможностью непрерывного хранения численных данных:

- `numpy.array`
- `pandas.DataFrame`

Можно использовать предсозданные из этих типов или из файла `catboost.FeaturesData` и `catboost.Pool`, полезно при переборе параметров

Избегаем копирования float колонок

Чтобы не копировать float данные между python и c++ объектами, должны выполняться два условия.

Нужная строгая типизация массива и/или колонки:

| `numpy.float32`

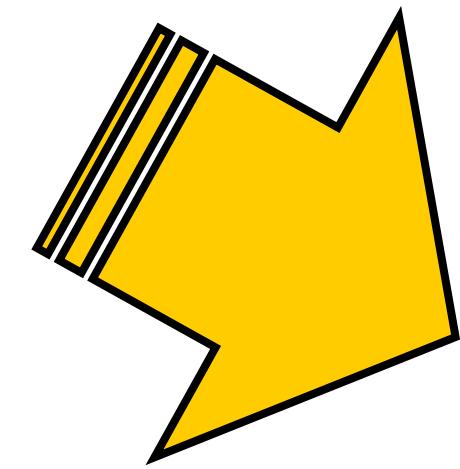
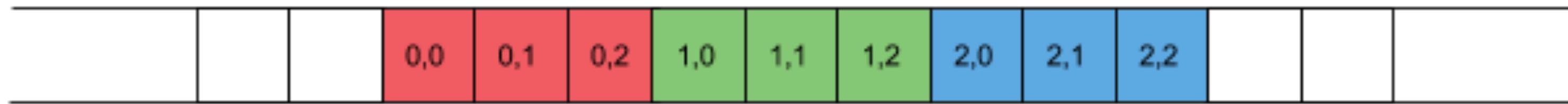
Нужны F contiguous массивы (F as in Fortran)

Почему так

row,col

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

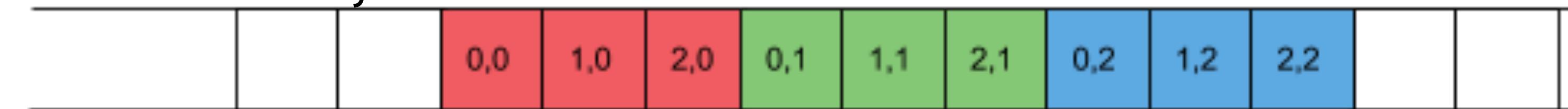
C contiguous
Row major



row,col

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

F contiguous
Column major



Пример (все плохо)

```
first_arr, first_target = np.random.rand(100000, 200), np.random.rand(100000)
```

```
first_arr.dtype, first_arr.flags
```

```
(dtype('float64'), C_CONTIGUOUS : True
 F_CONTIGUOUS : False
 OWNDATA : True
 WRITEABLE : True
 ALIGNED : True
 WRITEBACKIFCOPY : False
 UPDATEIFCOPY : False)
```

```
%timeit catboost.Pool(first_arr, first_target)
```

```
3.27 s ± 96.8 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Пример (стало лучше)

```
first_arr, first_target = np.random.rand(100000, 200).astype(np.float32), np.random.rand(100000)  
first_arr.dtype, first_arr.flags
```

```
(dtype('float32'), C_CONTIGUOUS : True  
 F_CONTIGUOUS : False  
 OWNDATA : True  
 WRITEABLE : True  
 ALIGNED : True  
 WRITEBACKIFCOPY : False  
 UPDATEIFCOPY : False)
```

```
%timeit catboost.Pool(first_arr, first_target)
```

```
868 ms ± 3.91 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Пример (теперь все хорошо)

```
first_arr = np.asfortranarray(np.random.rand(100000, 200), dtype=np.float32)
first_target = np.random.rand(100000).astype(np.float32)
first_arr.dtype, first_arr.flags
```

```
(dtype('float32'),    C_CONTIGUOUS : False
 F_CONTIGUOUS : True
 OWNDATA : True
 WRITEABLE : True
 ALIGNED : True
 WRITEBACKIFCOPY : False
 UPDATEIFCOPY : False)
```

```
%timeit catboost.Pool(first_arr, first_target)
```

```
171 ms ± 1.68 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

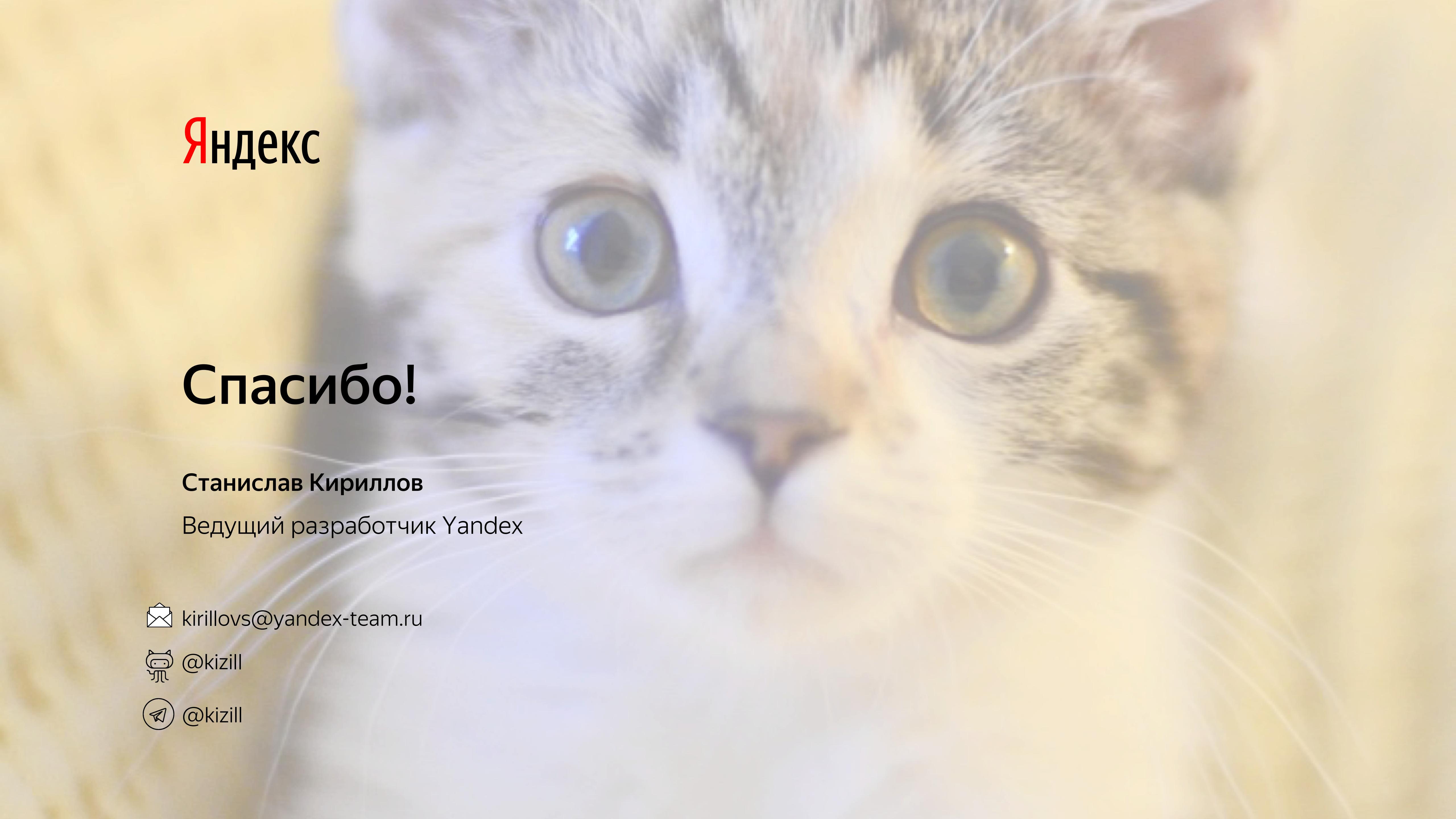
Уменьшаем использование памяти для категориальных фичей

Катбуст всегда переводит значения категориальных колонок в строки, сейчас оптимизации

Для текстовых колонок лучше использовать `pandas.Categorical`

Где нас найти?

- catboost.ai
- github.com/catboost
- twitter.com/CatBoostML
- t.me/catboost_en, t.me/catboost_ru
- ods.ai => slack (30k people community) => tool_catboost channel

A close-up photograph of a cat's face, focusing on its large, bright yellow eyes. The cat has white and light brown fur. The background is blurred.

Яндекс

Спасибо!

Станислав Кириллов

Ведущий разработчик Yandex

✉ kirillovs@yandex-team.ru

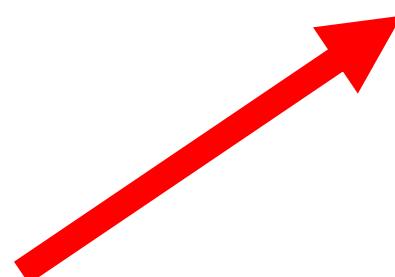
🐙 @kizill

telegram icon @kizill

Пример интересных багов #1

В описании функций импортируемых из C++ **nogil** это просто декларация
`\\(ツ)/*`

```
void ApplyModelMulti(  
    const EPredictionType predictionType,  
    int begin,  
    int end,  
    TVector<double>*& flatApprox,  
    TVector<TVector<double>>*& approx  
) nogil except +ProcessException
```



Пример интересных багов #2

```
numpy.array([[value for value in vec] for vec in rows])
```

rows – это std::vector<std::vector<double>>

расход памяти 15ГБ



```
numpy.empty([dim1, dim2], dtype= numpy.float64) + заполнение в цикле
```

расход памяти 800МБ

Разыменование указателей

```
int* a = new int
derefERENCE(a) = 4 # invalid
a[0] = 4 # valid
print(derefERENCE(a)) # valid
```