


Яндекс

Яндекс  ClickHouse

Применение моделей CatBoost в ClickHouse

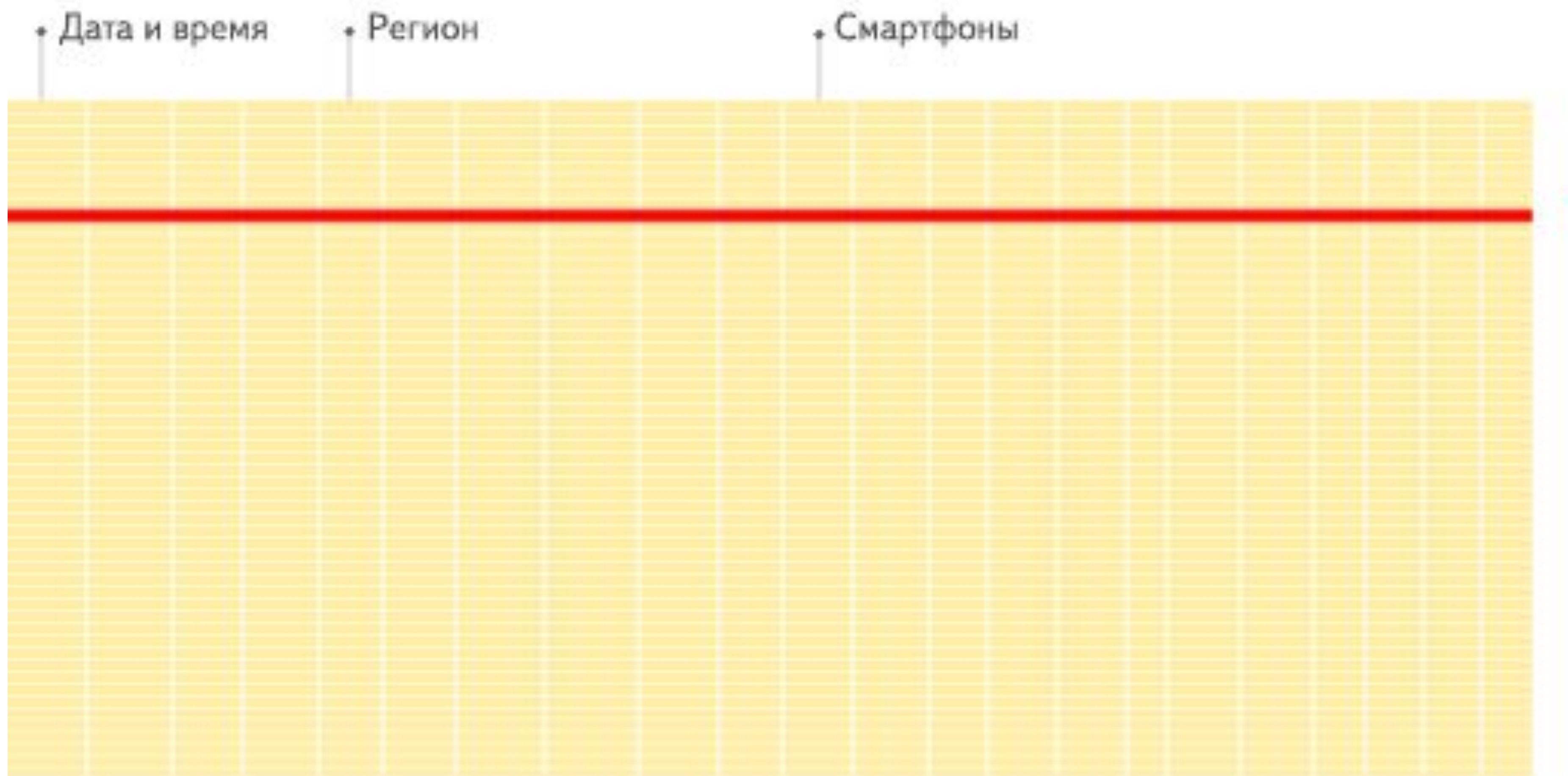
Николай Кочетов

Что такое ClickHouse?

ClickHouse - distributed analytical column-oriented DBMS

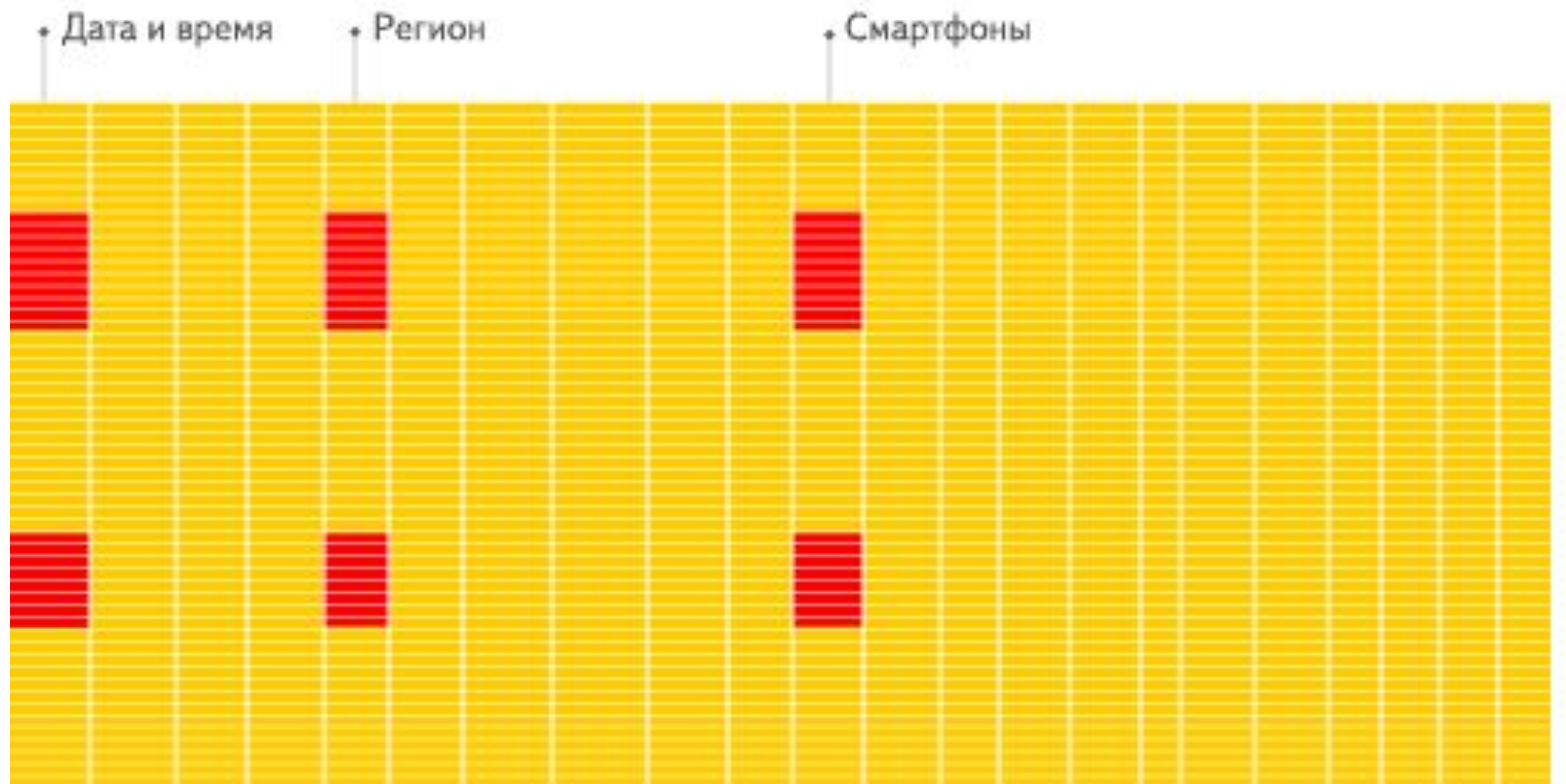
Почему column-oriented?

Так работают row-oriented системы:



Почему column-oriented?

Так работают column-oriented системы:



Особенности ClickHouse

- › column-oriented
- › линейная масштабируемость
- › отказоустойчивость
- › загрузка данных в реальном времени
- › онлайн (sub-second) запросы
- › поддержка диалекта SQL + расширения
(массивы, вложенные структуры данных, domain-specific функции, сэмплирование)

Предсказание вероятности покупки



Предсказание вероятности покупки

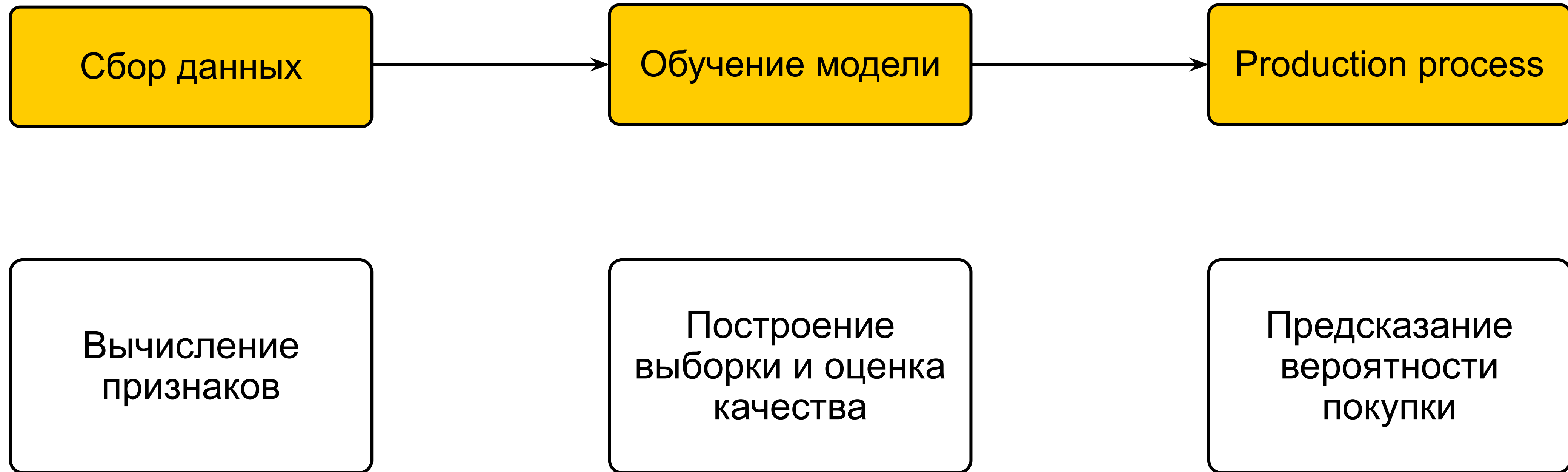
Задача

Узнать больше информации про пользователей

Цель

По поведению пользователей в прошлом, предсказывать их поведение в будущем. Например, ответить на вопрос — с какой вероятностью пользователь совершит заказ.

Рабочий процесс



Сбор данных

Используем данные **Яндекс.Метрики** из **Logs API**

Сырые данные по хитам и визитам

- › Предыдущие покупки
- › Состояние корзины
- › Посещения страниц с описанием товара

 Загружаем данные в ClickHouse

Вычисление признаков

Можем использовать более 60 характеристик

Характеристики пользователя:

Устройство, браузер, регион

Поведение:

Источники трафика, доход,
был ли последний визит отказом и т.д.

Вычисление признаков

- › Храним данные в ClickHouse в неагрегированном виде
- › Можем получить (почти) любые признаки

SELECT

› Например, средняя длительность сессии:
avg(Duration) as avg_duration

FROM visits_all SAMPLE 1/10 OFFSET 2/10

WHERE StartDate BETWEEN '{start_date}' AND '{end_date}'

GROUP BY FirstPartyCookie

Обучение моделей

Собрали выборку и обучили несколько различных моделей:

- › SVM
- › Logistic Regression
- › Random Forest
- › XGBoost
- › CatBoost

Внедрение обученной модели

В результате обучения получили модель и набор скриптов.

Как с этим жить?

Способ 1: Используем код из обучения

Наводим порядок в куче скриптов:

1. Выгружаем данные раз в неделю (каждый понедельник)
2. Применяем модель также, как и тестировали
3. Загружаем вероятность покупки в таблицу ClickHouse

Способ 1: Используем код из обучения

Преимущества

- › Просто, быстро, эффективно. Часть кода уже написана.
- › Полезно. Пригодится при переобучении.

Недостатки

- › Загрузка и выгрузка данных. Может тормозить.
- › Заранее готовим ответы.

Сложно получить ответ для произвольного периода.

Способ 2: препарлируем модель

План работ

- › Смотрим, как устроена модель
- › Переносим процесс применения в хранилище данных
- › Избавляемся от загрузки и выгрузки данных

Чего сможем добиться?

- › Избавимся от перекладывания данных
- › Будем работать с произвольным множеством данных
- › Применение модели — запрос в базу

Способ 2: препарлируем модель

Какие алгоритмы можем перенести в СУБД?

Способ 2: препарируем модель

Линейные классификаторы

Тривиально

```
SELECT ((feature1 * w1) + ... + (featureN * wN)) > threshold  
FROM table
```

Логистическая регрессия

Результаты на тестовой выборке

› ROC AUC: 0.917

› 0.44 sec. 441497 rows/sec. 63.16 MiB/sec.

Способ 2: препарируем модель

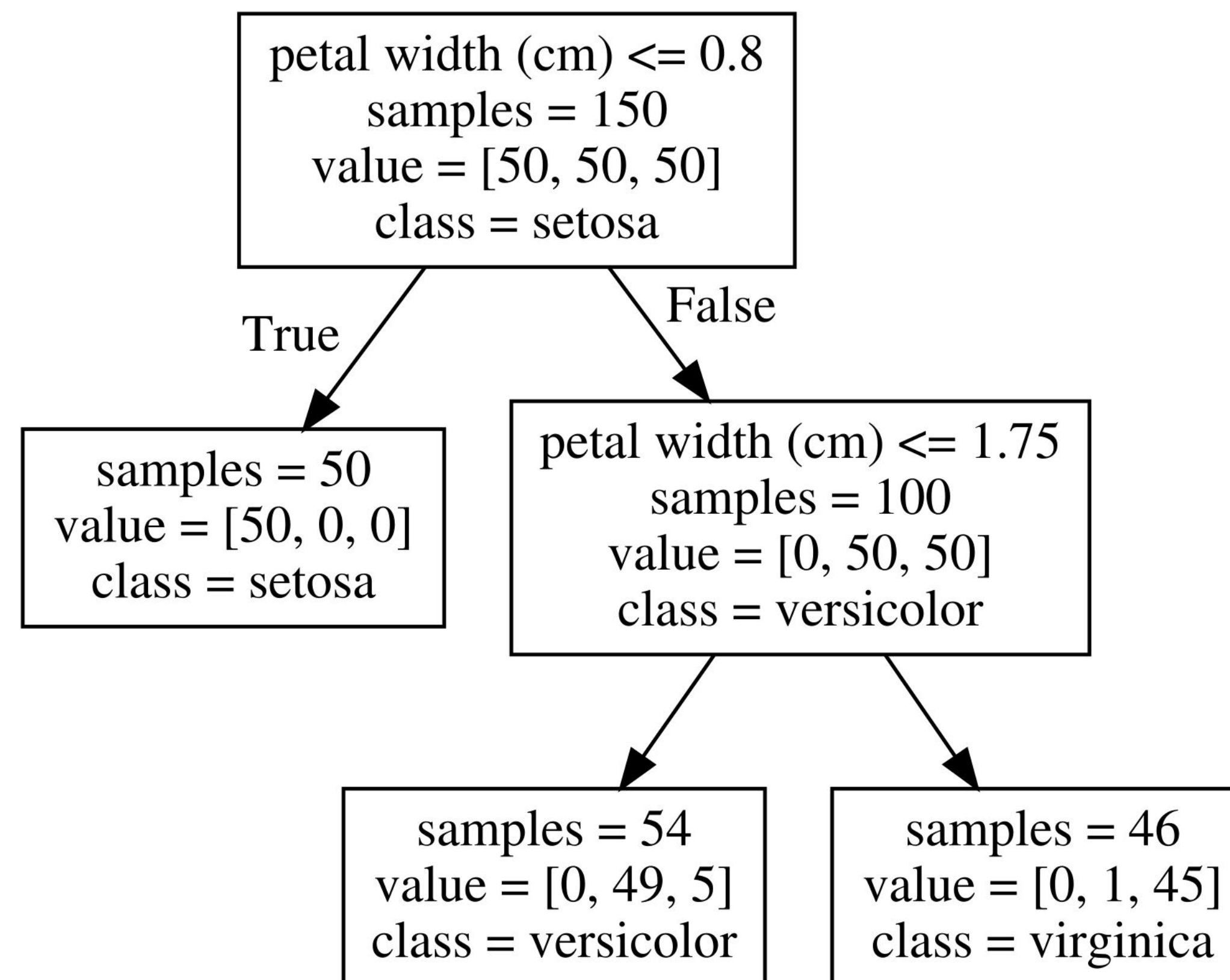
Дерево принятия решений

Вкладываем дерево в цепочку вызовов условных функций

SELECT

```
if(petal_width <= 0.8,  
  'setosa',  
  if(petal_width <= 1.75,  
    'versicolor',  
    'virginica'))
```

FROM iris



Способ 2: препарируем модель

Лес, бустинг — набор деревьев

```
SELECT arrayReduce('avg',[if(...), ..., if(...)])
```

Random Forest, 100 деревьев глубины 3

Результаты на тестовой выборке

› ROC AUC: 0.917 (0.923 для глубины 6)

› 2.54 sec. 59450 rows/sec. 1.59 MiB/sec.

Способ 2: препарлируем модель

Недостатки

- › Сложность преобразования модели в запрос
- › Ограниченная применимость
- › Не для всех алгоритмов машинного обучения
- › Ограничения со стороны СУБД
- › Проблемы с производительностью

Способ 3: встраиваем применение в базу

- › Используем библиотеку машинного обучения внутри базы
- › Перекладываем на базу работу по преобразованию данных
- › Применяем модель как вызов встроенной функции
`SELECT modelEvaluate('iris', sepal_width, petal_width)`
`FROM iris`

Способ 3: встраиваем применение в базу

Преимущества

Те же, что и у предыдущего способа, но

- › Нет неоправданных проблем с производительностью
- › Основная работа — на базе данных
- › Оптимизация работы внутри библиотеки машинного обучения

Недостатки

- › База должна поддерживать работу с конкретным алгоритмом
- › Различия в версиях и форматах хранения

Способ 3: встраиваем применение в базу

CatBoost, 100 деревьев глубины 6

Результаты на тестовой выборке

› ROC AUC: 0.932

› 3.96 sec. 19467 rows/sec. 2.78 MiB/sec.

XGBoost

Результаты на тестовой выборке

› 1000 деревьев глубины 3 — ROC AUC: 0.929

Способ 3: встраиваем применение в базу

Результаты

Алгоритм	ROC AUC	Время	Скорость
Логистическая регрессия	0.917	0.44 sec.	63.16 MiB/sec.
Лес, 100 дер. глубины 3	0.917	2.54 sec.	1.59 MiB/sec.
Лес, 100 дер. глубины 6	0.924		
XGBoost, 1000 дер. глубины 3	0.929		
CatBoost, 100 дер. глубины 6	0.932	3.96 sec.	2.78 MiB/sec.

Интеграция ClickHouse и CatBoost



Модели CatBoost в ClickHouse

1. Описываем конфигурацию модели

```
<models>
  <model>
    <!-- Тип модели. Сейчас только catboost. -->
    <type>catboost</type>
    <!-- Имя модели. Для modelEvaluate(). -->
    <name>purchase_model</name>
    <!-- Путь к обученной модели. -->
    <path>clickhouse/models/model.cbm</path>
    <!-- Период обновления. 0 — не обновляем. -->
    <lifetime>0</lifetime>
  </model>
</models>
```

Модели CatBoost в ClickHouse

1. Описываем конфигурацию модели
2. В config.xml добавляем путь к конфигурации и путь к CatBoost

```
<!-- catboost/catboost/libs/model_interface репозитория CatBoost -->
```

```
<!-- Сборка: ../../../../ya make -r -->
```

```
<catboost_dynamic_library_path>
```

```
    /path/to/libcatboostmodel.so
```

```
</catboost_dynamic_library_path>
```

```
<!-- Используем '*' для маски поиска -->
```

```
<models_config>
```

```
    clickhouse/models/model*.xml
```

```
</models_config>
```

Модели CatBoost в ClickHouse

1. Описываем конфигурацию модели
2. В config.xml добавляем путь к конфигурации и путь к CatBoost
3. Используем функцию

```
modelEvaluate('model_name', feature1, ..., featureN)
```

```
SELECT
```

```
    modelEvaluate('purchase_model', ...) AS prediction
```

```
FROM table
```

Сначала перечисляем числовые признаки, затем категориальные.

Чтение из CatBoost Pool

Формат входных данных для обучения CatBoost — CatBoost Pool

1. Описание столбцов — TSV файл вида

```
column_id data_type feature_id (optional)
```

Пример для двух признаков и Target

```
0 Categ is_yabrowser
```

```
1 Num viewed_products
```

```
2 Target
```

2. Описание датасета — TSV файл с данными

Чтение из CatBoost Pool

Чтобы быстро протестировать работу обученной модели в ClickHouse, добавлена возможность читать данные сразу из пула CatBoost.

■ Табличная функция `catBoostPool`.

Параметры — пути к файлам с описанием столбцов и датасета.

```
catBoostPool('/path/to/column/description', '/path/to/dataset/description')
```

Создает временную таблицу с движком `File('TSV')`.

Файлы должны находиться в директории данных сервера.

Чтение из CatBoost Pool

Описание столбцов пула CatBoost:

0 Categ is_yabrowser

1 Num viewed_products

2 Target

Описание столбцов в catBoostPool

`DESCRIBE TABLE` catBoostPool('test.cd', 'test.csv')

name	type	default_type
Num0	Float64	
Categ0	String	
Target	Float64	
is_yabrowser	String	ALIAS Categ0
viewed_products	Float64	ALIAS Num0

Чтение из CatBoost Pool

Описание столбцов пула CatBoost:

0 Categ is_yabrowser

1 Num viewed_products

2 Target

Описание столбцов в catBoostPool

DESCRIBE TABLE

(

SELECT *

FROM catBoostPool('test.cd', 'test.csv')

)

name	type	default_type	default_expression
Num0	Float64		
Categ0	String		

Использование обученной модели

Предсказываем вероятность покупки

SELECT

modelEvaluate('purchase_model', *) AS prediction,

1. / (1. + exp(-prediction)) AS probability

FROM catBoostPool('test.cd', 'test.csv')

Использование обученной модели

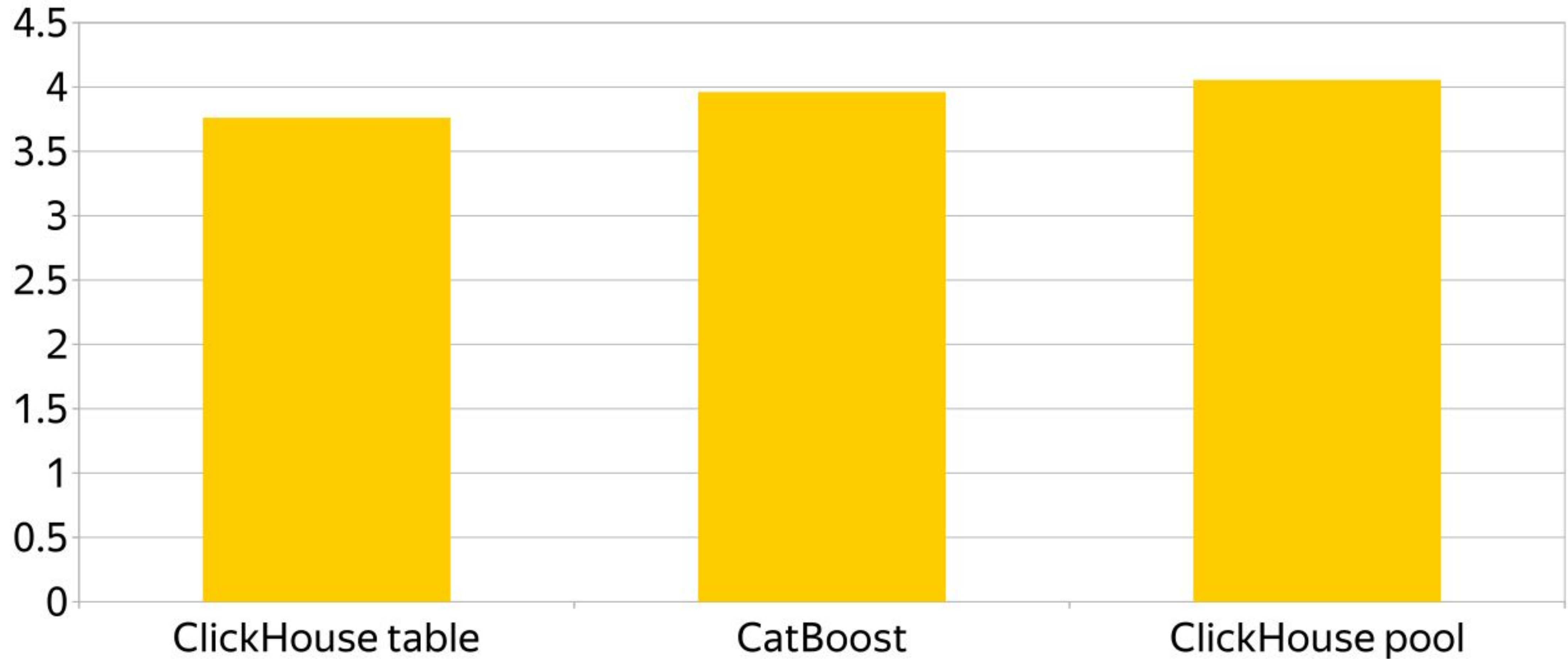
Считаем ошибку на тестовой выборке по метрике Logloss

```
SELECT -avg((Target * log(prob) +  
            ((1. - Target) * log(1. - prob)))) AS logloss  
  
FROM  
  
(  
    SELECT  
        modelEvaluate('purchase_model', *) AS pred,  
        1. / (1. + exp(-pred)) AS prob,  
        Target  
    FROM catBoostPool('test.cd', 'test.csv')  
)
```

```
┌──────────────────┴──┐ logloss ─┐  
└──────────────────┬──┘  
0.02710106000177000
```

Использование обученной модели

■ Время запроса, сек.



ИТОГИ

Интеграция ClickHouse и CatBoost

- › Применение обученных моделей
- › Чтение данных из пула

Дальнейшие планы

- › Другие форматы моделей
- › Встроенное обучение моделей — ?

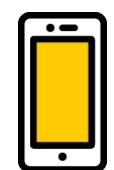
Спасибо!

Николай Кочетов

Разработчик ClickHouse



nik-kochetov@yandex-team.ru



+7 965 124 03 64