Александр Крашенинников - 11 декабря 2019

ПРОГНОЗИРОВАНИЕ ВРЕМЕННЫХ РЯДОВ НА CLICKHOUSE

MagicLab

badoo

bumble

Lumen

CHAPPY


>550 000 000
people all over the world
use our apps

О чём поговорим

- Что такое прогнозы временных рядов
- Какие есть способы предсказания
- Критерии оценки качества предсказаний
- Про будущее

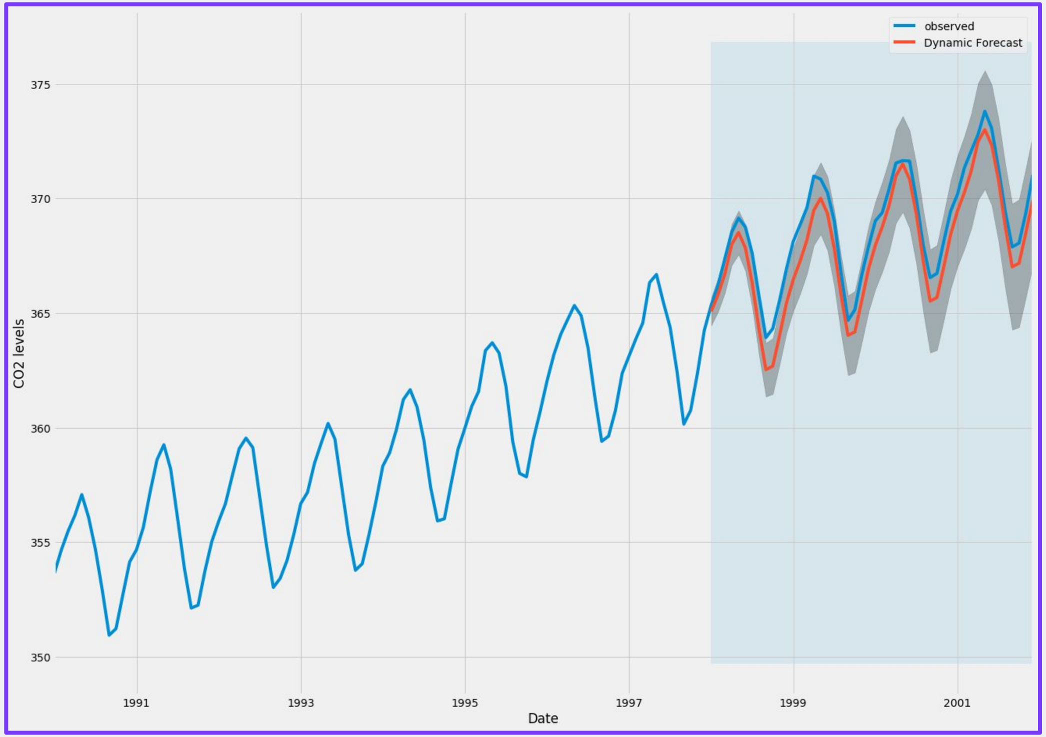
DISCLAIMER

Я – инженер, любящий отец
и отличный парень

Я не data scientist

Все материалы носят ознакомительный характер

Прогнозирование



Прогнозирование

- Предсказание будущего
- Событий или фактов
- Значений показателей

“ The population is constant in size and will remain so right up to the end of mankind. ”

L’Encyclopedie, 1756

“Computers are multiplying at a rapid rate. By the turn of the century there will be 220,000 in the U.S.”

Wall Street Journal, 1966

Прогнозирование временных рядов

- Как будет вести себя метрика в будущем
  - Закупка оборудования
  - Планирование логистики/закупок

Прогнозирование временных рядов

- Как будет вести себя метрика в будущем
  - Закупка оборудования
  - Планирование логистики/закупок
- Обнаружение аномального поведения
  - One-step-ahead forecast

Качественное предсказание

- Сложные модели и технологии
- Рекурсивные алгоритмы
- Индивидуальные модели для каждого ряда
- Хорошие реализации на Python и R

Но почему ClickHouse?

- Не тормозит!
- Подходит для миллионов метрик
- Параллельная обработка
- Батчевый анализ результатов предсказания
- Реализовать недостающее можно всегда!

Подготовка данных

Особенность работы с данными

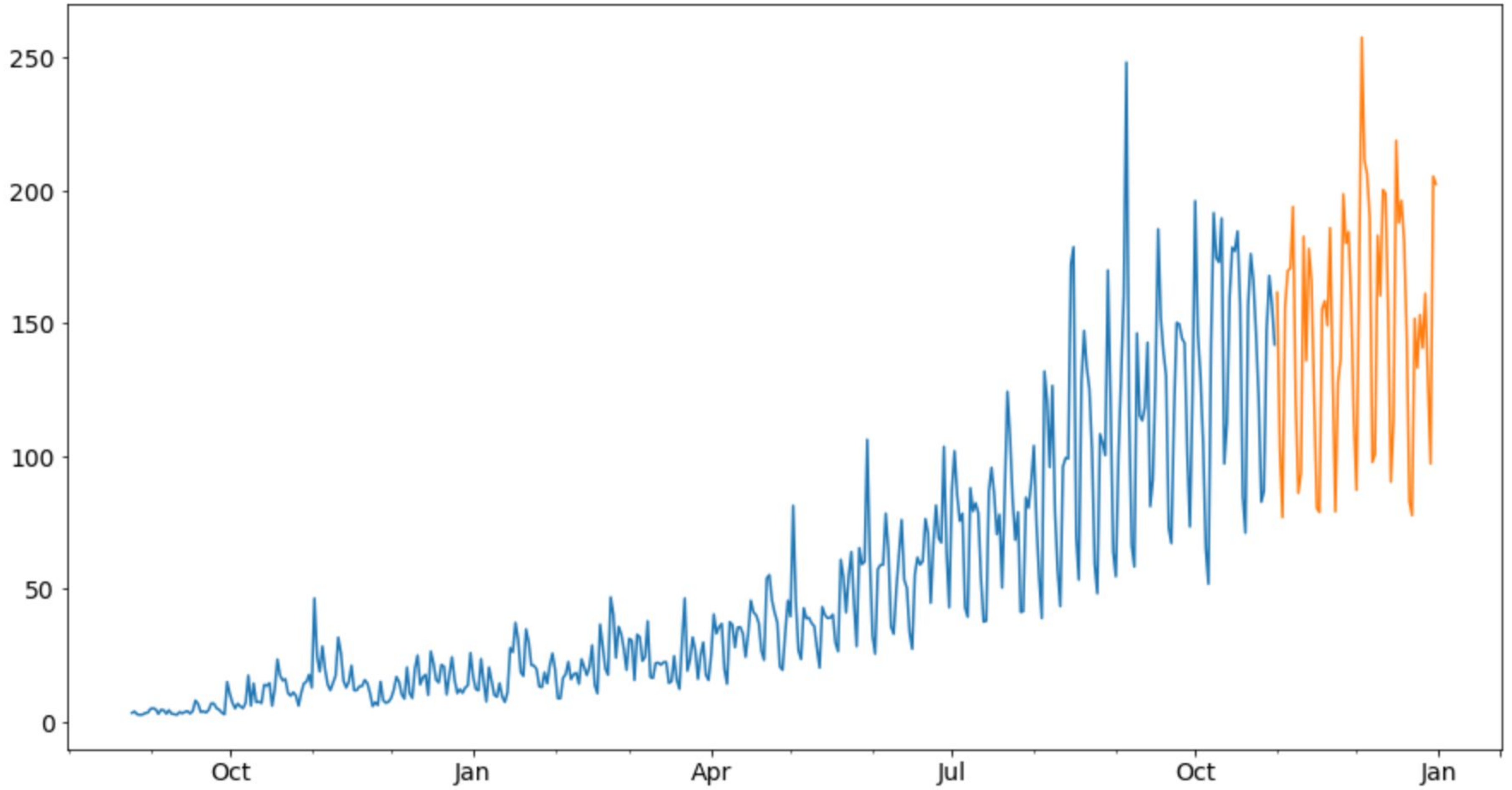
- Могут отсутствовать значения
    - Замена нолем, средним

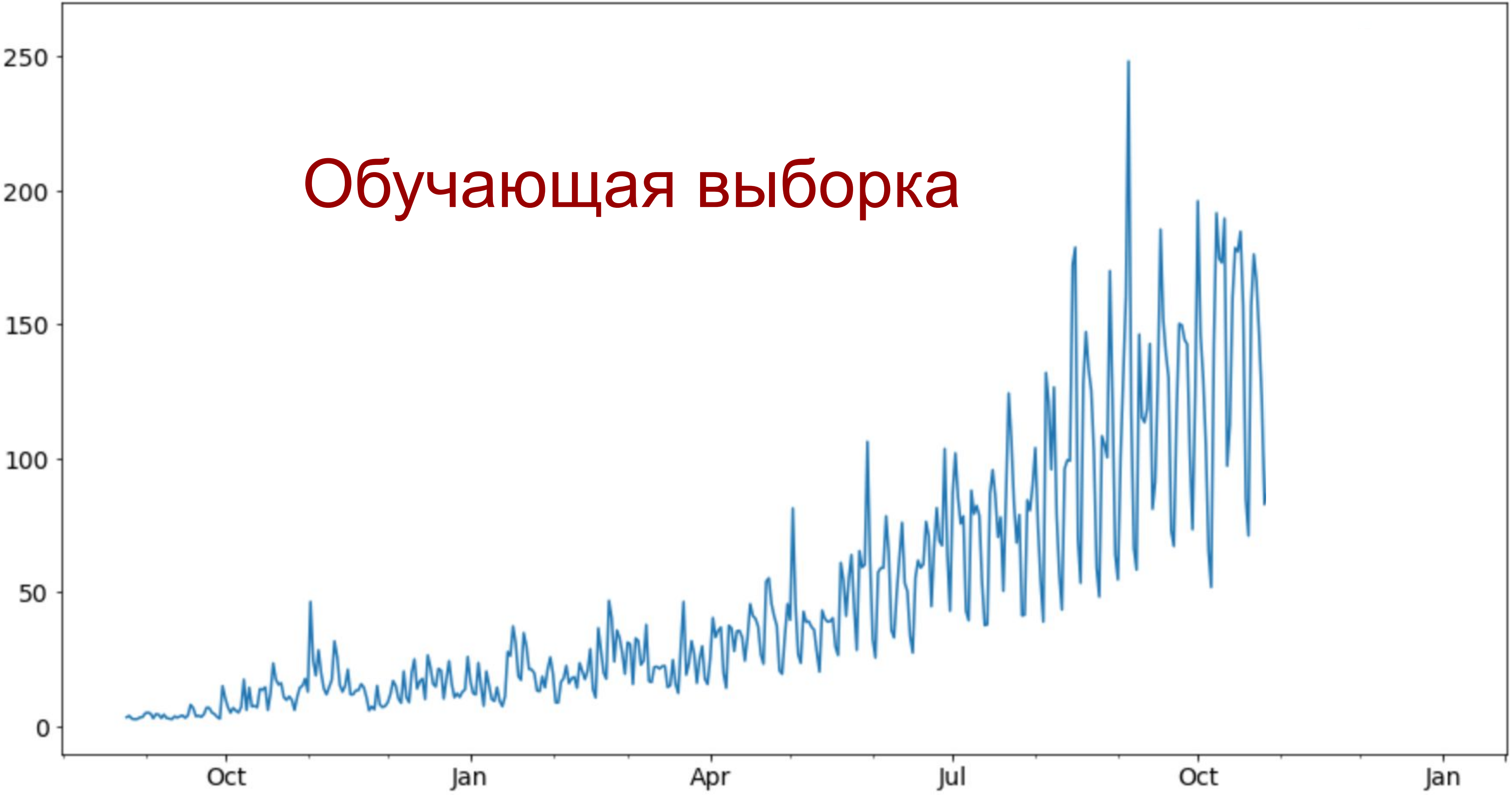
Особенность работы с данными

- Могут отсутствовать значения
  - Замена нолем, средним
- Значения могут поступать нерегулярно
  - Выравниваем по времени

Особенность работы с данными

- Могут отсутствовать значения
  - Замена нолем, средним
- Значения могут поступать нерегулярно
  - Выравниваем по времени
- Необходимость пересчёта
  - Партицируем по выровненным интервалам


250
200
150
100
50
0
Oct
Jan
Apr
Jul
Oct
Jan


Обучающая выборка
250
200
150
100
50
0
Oct
Jan
Apr
Jul
Oct
Jan

```
CREATE TABLE metrics
(
    `dt` MATERIALIZED toDate(ts),
    `ts` DateTime,
    `id` UInt64 CODEC(Delta, ZSTD),
    `value` Float64 CODEC(Gorilla, ZSTD)
)
ENGINE = MergeTree
PARTITION BY (dt, ts)
ORDER BY id
```

```
CREATE TABLE metrics
(
    `dt` MATERIALIZED toDate(ts),
    `ts` DateTime,
    `id` UInt64 CODEC(Delta, ZSTD),
    `value` Float64 CODEC(Gorilla, ZSTD)
)
ENGINE = MergeTree
PARTITION BY (dt, ts)
ORDER BY id
```

Classic

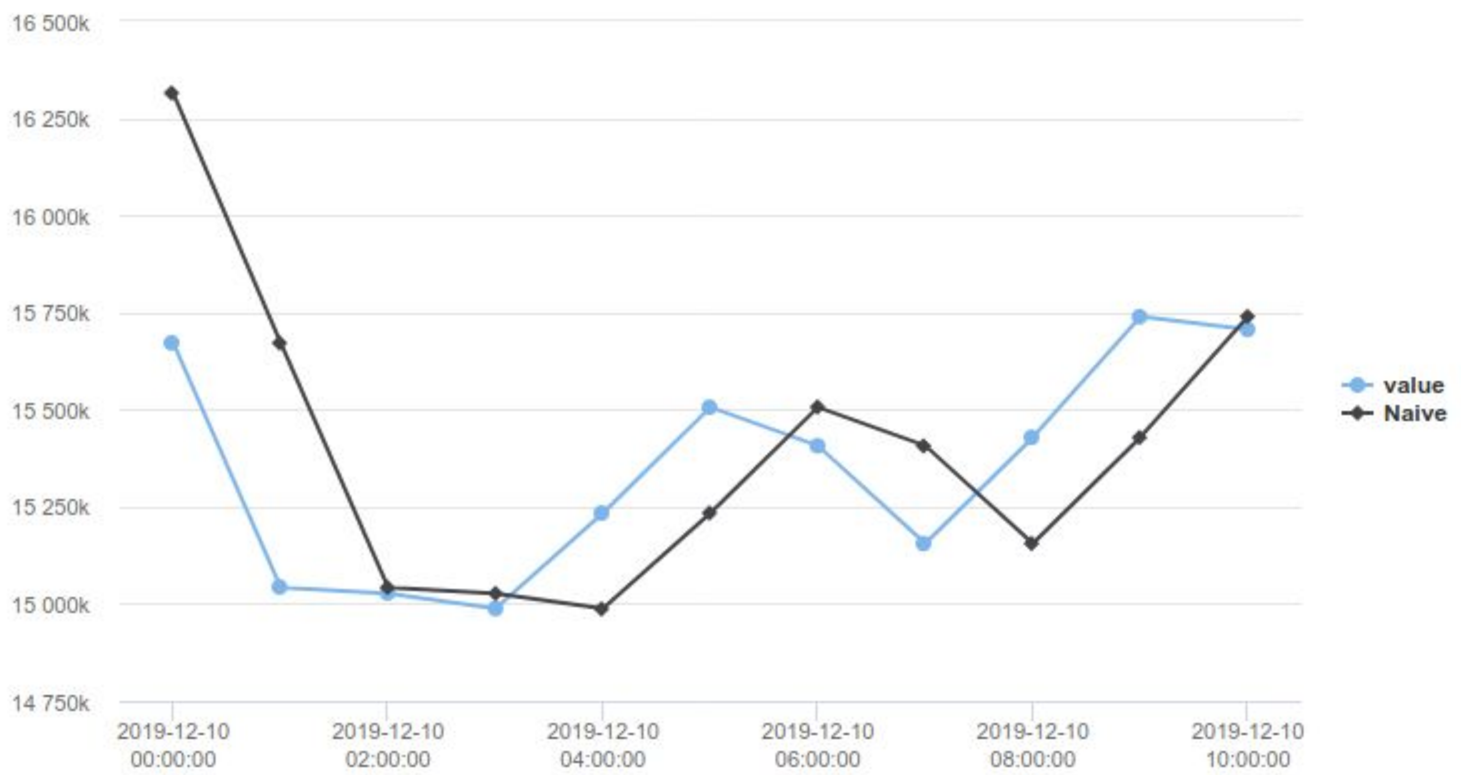
```sql
CREATE TABLE metrics
(
    `dt` MATERIALIZED toDate(ts),
    `ts` DateTime,
    `id` UInt64 CODEC(Delta, ZSTD),
    `value` Float64 CODEC(Gorilla, ZSTD)
)
ENGINE = MergeTree
PARTITION BY (dt, ts)
ORDER BY id
```

Не строка!
Delta + ORDER = 2x
read speed

Модели предсказаний

Naive

- “Завтра будет как вчера”
- $F_t = O_{t-1}$ (**F**orecast, **O**bserved)
- Для оценки качества других моделей
- Или если ничего не происходит :)


16 500k
16 250k
16 000k
15 750k
15 500k
15 250k
15 000k
14 750k
2019-12-10 00:00:00
2019-12-10 02:00:00
2019-12-10 04:00:00
2019-12-10 06:00:00
2019-12-10 08:00:00
2019-12-10 10:00:00
value
Naive

```sql
WITH
    toDateTime('2019-12-12 00:00:00') AS next_time,
    3600 AS frequency
SELECT
    id,
    value AS forecast
FROM metrics
WHERE ts = (next_time - frequency)
```

```
WITH
    toDateTime('2019-12-12 00:00:00') AS next_time,
    3600 AS frequency
SELECT
    id,
    value AS forecast
FROM metrics
WHERE ts = (next_time - frequency)
```

Время предсказания

```
WITH
    toDateTime('2019-12-12 00:00:00') AS next_time,
    3600 AS frequency
SELECT
    id,
    value AS forecast
FROM metrics
WHERE ts = (next_time - frequency)
```

Ширина выровненного интервала

Linear Regression

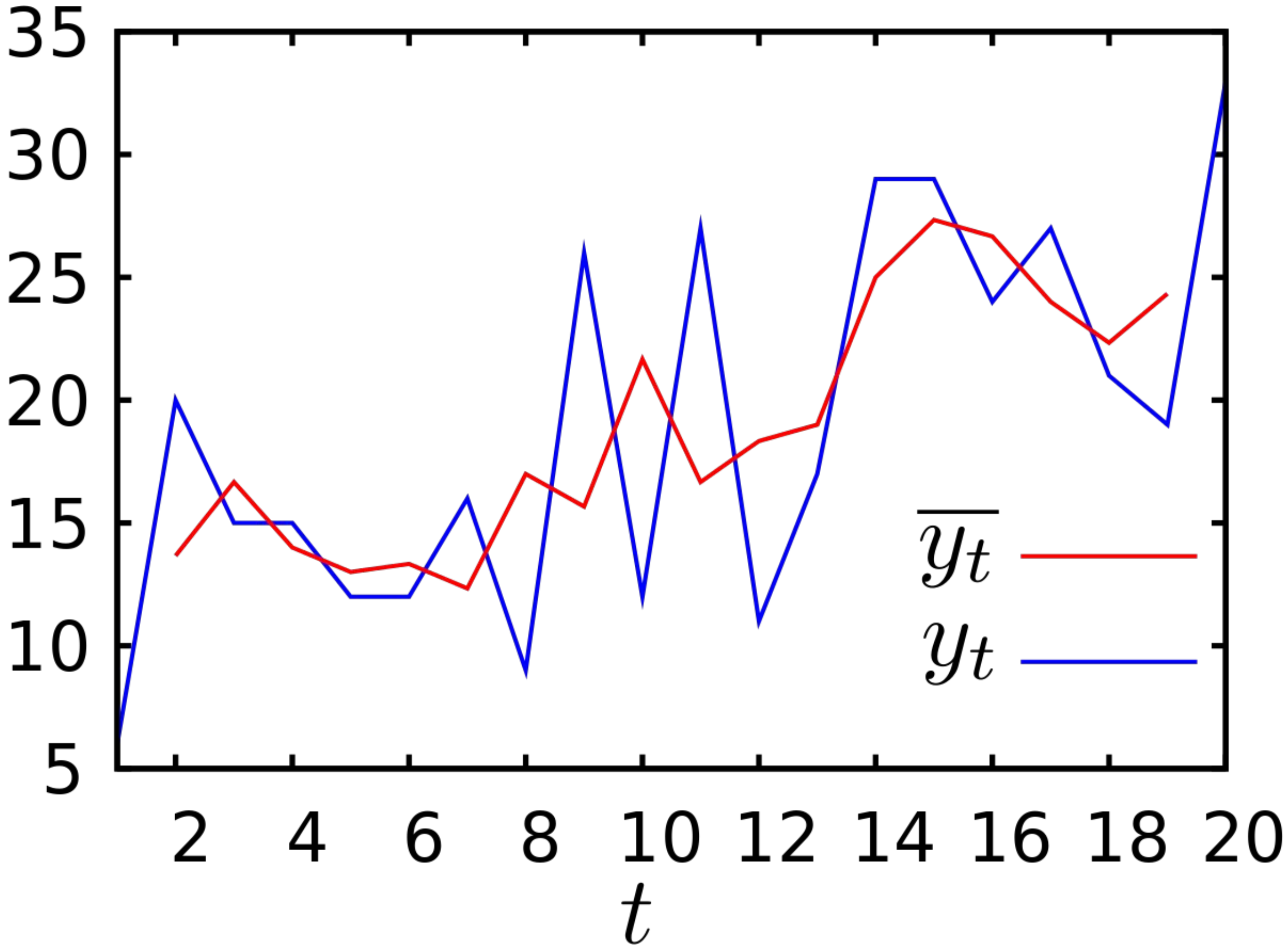
- Аппроксимация значений временного ряда прямой линией
- Также используется для преобразования рядов

```
WITH
    toFloat64(toDateTime('2019-12-12 04:00:00')) AS next_time,
    toFloat64(ts) AS casted,
    simpleLinearRegression(cast, value) AS k

SELECT
    id,
    next_time * k.1 + k.2 AS forecast
FROM metrics
GROUP BY id
```

Moving Average

- Предсказание = среднее среди предыдущих
- $F_t = (O_{t-1} + \ldots + O_{t-n}) / n$


35
30
25
20
15
10
5
2
4
6
8
10
12
14
16
18
20
t
$\overline{y_t}$
y_t

```
WITH
    toDateTime('2019-12-12 00:00:00') AS next_time,
    4 AS num_probes
SELECT
    id,
    avg(value)
FROM metrics
WHERE ts BETWEEN
next_time - (num_probes + 1) * frequency AND next_time
```

Weighted Linear Moving Average

- Предсказание = взвешенное среднее среди предыдущих
- Чем ближе к точке предсказания, тем выше вес

$$WMA_t = \frac{n \cdot p_t + (n-1) \cdot p_{t-1} + \cdots + (n-i) \cdot p_{t-i} + \cdots + 2 \cdot p_{t-n} + 1 \cdot p_{t-n+1}}{n + (n-1) + \cdots + (n-i) + \cdots + 2 + 1} = \frac{2}{n \cdot (n+1)} \sum_{i=0}^{n-1} (n-i) \cdot p_{t-i}$$

ts	value	weight
2019-12-12 00:00:00	100	1
2019-12-12 01:00:00	200	2
2019-12-12 02:00:00	300	3
2019-12-12 03:00:00	400	4

```
            2
WMA = -------  x (4 x 400 + 3 x 300 + 2 x 200 + 1 x 100) = 500
        4 x (4 -1)
```

$$\frac{2}{n \cdot (n+1)} \sum_{i=0}^{n-1} (n-i) \cdot p_{t-i}$$

```sql
WITH
    4 AS num_probes,
    (2 + num_probes) - ((next_time - ts) / frequency)
    AS weight
SELECT
    (2 / (num_probes * (num_probes - 1))) *
    sum(value * weight)
FROM metrics
WHERE ts >= next_time - frequency * (num_probes + 1)
AND next_time
GROUP BY id
```

```sql
WITH
    4 AS num_probes,
    (2 + num_probes) - ((next_time - ts) / frequency)
    AS weight
SELECT
    (2 / (num_probes * (num_probes - 1))) *
    sum(value * weight)
FROM metrics
WHERE ts >= next_time - frequency * (num_probes + 1)
AND next_time
GROUP BY id
```

Соответствие
ts => weight

$$\boxed{\frac{2}{n \cdot (n+1)}} \sum_{i=0}^{n-1} (n-i) \cdot p_{t-i}$$

```
WITH
    4 AS num_probes,
    (2 + num_probes) - (                                    ncy)
    AS weight
SELECT
    (2 / (num_probes * (num_probes - 1))) *
    sum(value * weight)
FROM metrics
WHERE ts >= next_time - frequency * (num_probes + 1)
AND next_time
GROUP BY id
```

$$\frac{2}{n \cdot (n+1)} \boxed{\sum_{i=0}^{n-1} (n-i) \cdot p_{t-i}}$$

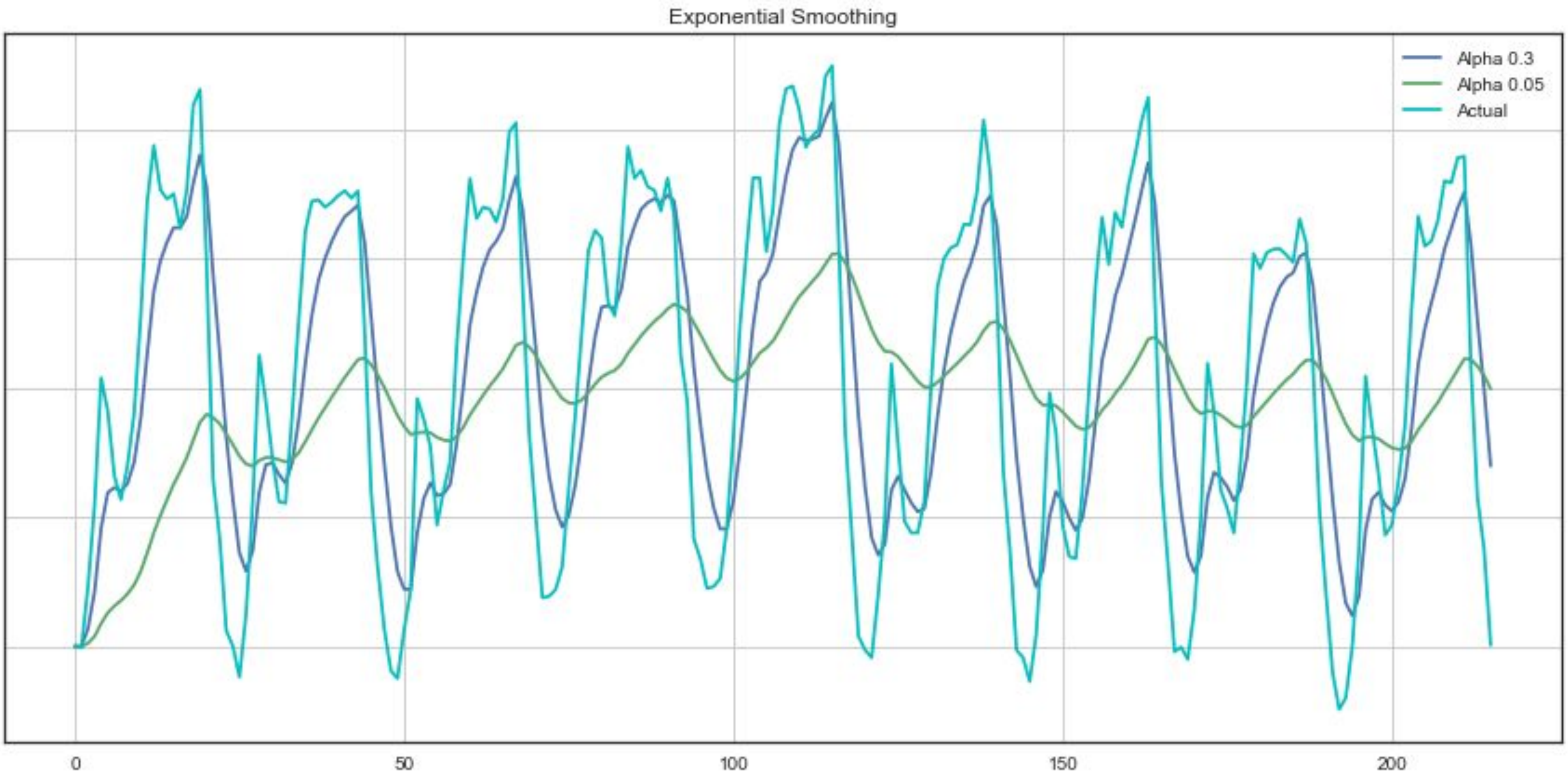
```
WITH
    4 AS num_probes,
    (2 + num_probes) - (                      ncy)
    AS weight
SELECT
    (2 / (num_probes * (num_probes - 1))) *
    sum(value * weight)
FROM metrics
WHERE ts >= next_time - frequency * (num_probes + 1)
AND next_time
GROUP BY id
```

Exponential Smoothing

- Геометрически убывающая сумма предыдущих значений

$$s_t = \begin{cases} c_1 & : t = 1 \\ s_{t-1} + \alpha \cdot (c_t - s_{t-1}) & : t > 1 \end{cases}$$

- Предсказание = “размытое” значение предыдущей точки


Exponential Smoothing
Alpha 0.3
Alpha 0.05
Actual
0
50
100
150
200

$$
\begin{aligned}
\tilde{y}_1 &= \lambda y_1 + (1-\lambda)\tilde{y}_0 \\
\tilde{y}_2 &= \lambda y_2 + (1-\lambda)\tilde{y}_1 = \lambda y_2 + (1-\lambda)(\lambda y_1 + (1-\lambda)\tilde{y}_0) \\
&= \lambda(y_2 + (1-\lambda)y_1) + (1-\lambda)^2\tilde{y}_0 \\
\tilde{y}_3 &= \lambda(y_3 + (1-\lambda)y_2 + (1-\lambda)^2 y_1) + (1-\lambda)^3\tilde{y}_0 \\
&\vdots \\
\tilde{y}_T &= \lambda(y_T + \underline{(1-\lambda)y_{T-1} + \cdots + (1-\lambda)^{T-1}y_1}) + (1-\lambda)^T\tilde{y}_0,
\end{aligned}
$$

```sql
WITH
    0.3 AS a, 1 - a AS b,
    arraySort(groupArray((ts,value))) AS sorted, arrayMap(s -> s.2, sorted) AS values,
    length(values) AS pos, values[1] AS initial, a * values[pos-1] AS last_point,
    arrayMap(
        (x, pos_int)-> pos_int > pos - 2 ? 0 : a * pow(b, pos - pos_int - 1) * x,
        values,
        arrayEnumerate(values)
    ) AS smoo,
    last_point + arraySum(smoo) + pow(b, pos - 1) * initial AS forecast_last_point,
    a * values[pos] + b * forecast_last_point AS forecast
SELECT
    id, forecast
FROM metric
GROUP BY id
```

```
WITH
    0.3 AS a, 1 - a AS b,
    arraySort(groupArray((ts,value))) AS sorted, arrayMap(s -> s.2, sorted) AS values,
    length(values) AS pos, values[1] AS initial, a * values[pos-1] AS last_point,
    arrayMap(
        (x, pos_int)-> pos_int > pos - 2 ? 0 : a * pow(b, pos - pos_int - 1) * x,
        values,
        arrayEnumerate(values)
    ) AS smoo,
    last_point + arraySum(smoo) + pow(b, pos - 1) * initial AS forecast_last_point,
    a * values[pos] + b * forecast_last_point AS forecast
SELECT
    id, forecast
FROM metric
GROUP BY id
```

Параметр сглаживания [0, 1]

```
WITH
    0.3 AS a, 1 - a AS b,
    arraySort(groupArray((ts,value))) AS sorted, arrayMap(s -> s.2, sorted) AS values,
    length(values) AS pos, values[1] AS initial, a * values[pos-1] AS last_point,
    arrayMap(
        (x, pos_int)-> pos_int > pos - 2 ? 0 : a * pow(b, pos - pos_int - 1) * x,
        values,
        arrayEnumerate(values)
    ) AS smoo,
    last_point + arraySum(smoo) + pow(b, pos - 1) * initial AS forecast_last_point,
    a * values[pos] + b * forecast_last_point AS forecast
SELECT
    id, forecast
FROM metric
GROUP BY id
```

Сортируем в пределах id
по timestamp

```
WITH
    0.3 AS a, 1 - a AS b,
    arraySort(groupArray((ts,value))) AS sorted, arrayMap(s -> s.2, sorted) AS values,
    length(values) AS pos, values[1] AS initial, a * values[pos-1] AS last_point,
    arrayMap(
        (x, pos_int)-> pos_int > pos - 2 ? 0 : a * pow(b, pos - pos_int - 1) * x,
        values,
        arrayEnumerate(values)
    ) AS smoo,
    last_point + arraySum(smoo) + pow(b, pos - 1) * initial AS forecast_last_point,
    a * values[pos] + b * forecast_last_point AS forecast
SELECT
    id, forecast
FROM metric
GROUP BY id
```

Just aliases

$$\tilde{y}_T = \lambda(y_T + \boxed{(1-\lambda)y_{T-1} + \cdots} + (1-\lambda)^{T-1}y_1) + (1-\lambda)^T\tilde{y}_0,$$

```
    length(values) AS pos, values[1] AS initial, a * values[pos-1] AS last_point,
    arrayMap(
        (x, pos_int)-> pos_int > pos - 2 ? 0 : a * pow(b, pos - pos_int - 1) * x,
        values,
        arrayEnumerate(values)
    ) AS smoo,
    last_point + arraySum(smoo) + pow(b, pos - 1) * initial AS forecast_last_point,
    a * values[pos] + b * forecast_last_point AS forecast
SELECT
    id, forecast
FROM metric
GROUP BY id
```

$$\tilde{y}_T = \lambda(\boxed{y_T} + (1-\lambda)y_{T-1} + \cdots + (1-\lambda)^{T-1}y_1) + (1-\lambda)^T\tilde{y}_0,$$

```
WITH
        length(values) AS pos, values[1] AS initial, a * values[pos-1] AS last_point,
        arrayMap(
            (x, pos_int)-> pos_int > pos - 2 ? 0 : a * pow(b, pos - pos_int - 1) * x,
            values,
            arrayEnumerate(values)
        ) AS smoo,
        last_point + arraySum(smoo) + pow(b, pos - 1) * initial AS forecast_last_point,
        a * values[pos] + b * forecast_last_point AS forecast
SELECT
        id, forecast
FROM metric
GROUP BY id
```

$$\tilde{y}_T = \lambda(y_T + \boxed{(1-\lambda)y_{T-1} + \cdots} + (1-\lambda)^{T-1}y_1) + (1-\lambda)^T\tilde{y}_0,$$

```
WITH
    ...
    length(values) AS pos, values[1] AS initial, a * values[pos-1] AS last_point,
    arrayMap(
        (x, pos_int)-> pos_int > pos - 2 ? 0 : a * pow(b, pos - pos_int - 1) * x,
        values,
        arrayEnumerate(values)
    ) AS smoo,
    last_point + arraySum(smoo) + pow(b, pos - 1) * initial AS forecast_last_point,
    a * values[pos] + b * forecast_last_point AS forecast
SELECT
    id, forecast
FROM metric
GROUP BY id
```

$$\tilde{y}_T = \lambda(y_T + (1-\lambda)y_{T-1} + \cdots \boxed{+ (1-\lambda)^{T-1} y_1}) + (1-\lambda)^T \tilde{y}_0,$$

```
WITH
        length(values) AS pos, values[1] AS initial, a * values[pos-1] AS last_point,
        arrayMap(
            (x, pos_int)-> pos_int > pos - 2 ? 0 : a * pow(b, pos - pos_int - 1) * x,
            values,
            arrayEnumerate(values)
        ) AS smoo,
        last_point + arraySum(smoo) + pow(b, pos - 1) * initial AS forecast_last_point,
        a * values[pos] + b * forecast_last_point AS forecast
SELECT
        id, forecast
FROM metric
GROUP BY id
```

```sql
WITH
    0.3 AS a, 1 - a AS b,
    arraySort(groupArray((ts,value))) AS sorted, arrayMap(s -> s.2, sorted) AS values,
    length(values) AS pos, values[1] AS initial, a * values[pos-1] AS last_point,
    arrayMap(
        (x, pos_int)-> pos_int > pos - 2 ? 0 : a * pow(b, pos - pos_int - 1) * x,
        values,
        arrayEnumerate(values)
    ) AS smoo,
    last_point + arraySum(smoo) + pow(b, pos - 1) * initial AS forecast_last_point,
    a * values[pos] + b * forecast_last_point AS forecast
SELECT
    id, forecast
FROM metric
GROUP BY id
```

Мы научились готовить

- Naive
- Linear Regression
- Moving Average
- Weighted Moving Average
- Exponential Smoothing

Не вошли в short-list

- Polynomial Regression
- ARIMA models
- GARCH
- Нейронные сети
- Не значит, что всё это нереализуемо на ClickHouse!

Выбор модели

Последовательность

- Зафиксировали набор моделей
- Прогнали все метрики через все модели
- Получили реальные значения показателей
- Оцениваем метрики каждой модели для пар (observed, forecast)

```sql
CREATE TABLE forecast
(
    `dt` MATERIALIZED toDate(ts),
    `ts` DateTime,
    `id` UInt64 CODEC(Delta, ZSTD),
    `forecast` Float64,
    `model` String
)
ENGINE = MergeTree
PARTITION BY (dt, ts)
ORDER BY (model, id)
```

```sql
SELECT
    id,
    sum(v.value - f.forecast)
FROM metrics AS v
INNER JOIN forecast AS f
    ON (f.id = v.id) AND (f.ts = v.ts)
WHERE f.model = 'LinearRegression'
GROUP BY id
```

```sql
SELECT
    id,
    sum(v.value - f.forecast)
FROM metrics AS v
INNER JOIN forecast AS f
    ON (f.id = v.id) AND (f.ts = v.ts)
WHERE f.model = 'LinearRegression'
GROUP BY id
```

Метрика качества

```sql
SELECT
    id,
    sum(v.value - f.forecast)
FROM metrics AS v
INNER JOIN forecast AS f
    ON (f.id = v.id) AND (f.ts = v.ts)
WHERE f.model = 'LinearRegression'
GROUP BY id
```

Модель, которую оцениваем

Проблема

- Для миллиона метрик надо 10 раз сделать SCAN и JOIN
- Distributed x Distributed = ничего хорошего
- ClickHouse не тормозит, если “нормально делай – нормально будет”

```sql
CREATE TABLE values_forecast
(
	`dt` MATERIALIZED toDate(ts),
	`ts` DateTime,
	`id` UInt64 CODEC(Delta, ZSTD),
	`value` Float64,
	`forecast_1` Float64 DEFAULT 0 COMMENT 'LinearRegression(steps=6)',
	`forecast_2` Float64 DEFAULT 0 COMMENT 'WMA(steps=6)',
	`forecast_3` Float64 DEFAULT 0 COMMENT 'Naive',
	`mask` UInt64
)
ENGINE = SummingMergeTree
PARTITION BY (dt, ts)
ORDER BY id
```

```
CREATE TABLE values_forecast
(
    `dt` MATERIALIZED toDate(ts),
    `ts` DateTime,
    `id` UInt64 CODEC(Delta, ZSTD),
    `value` Float64,
    `forecast_1` Float64 DEFAULT 0 COMMENT 'LinearRegression(steps=6)',
    `forecast_2` Float64 DEFAULT 0 COMMENT 'WMA(steps=6)',
    `forecast_3` Float64 DEFAULT 0 COMMENT 'Naive',
    `mask` UInt64
)
ENGINE = SummingMergeTree
PARTITION BY (dt, ts)
ORDER BY id
```

Импорт из values

```
CREATE TABLE values_forecast
(
    `dt` MATERIALIZED toDate(ts),
    `ts` DateTime,
    `id` UInt64 CODEC(Delta, ZSTD),
    `value` Float64,
    `forecast_1` Float64 DEFAULT 0 COMMENT 'LinearRegression(steps=6)',
    `forecast_2` Float64 DEFAULT 0 COMMENT 'WMA(steps=6)',
    `forecast_3` Float64 DEFAULT 0 COMMENT 'Naive',
    `mask` UInt64
)
ENGINE = SummingMergeTree
PARTITION BY (dt, ts)
ORDER BY id
```

Колонка для каждой модели

Магия SummingMergeTree :)

```
CREATE TABLE values_forecast
(
    `dt` MATERIALIZED toDate(ts),
    `ts` DateTime,
    `id` UInt64 CODEC(Delta, ZSTD),
    `value` Float64,
    `forecast_1` Float64 DEFAULT 0 COMMENT 'LinearRegression(steps=6)',
    `forecast_2` Float64 DEFAULT 0 COMMENT 'WMA(steps=6)',
    `forecast_3` Float64 DEFAULT 0 COMMENT 'Naive',
    `mask` UInt64
)
ENGINE = SummingMergeTree
PARTITION BY (dt, ts)
ORDER BY id
```

```sql
INSERT INTO values_forecast (ts, id, value, mask)
WITH
	toDateTime('2019-12-12 00:00:00') AS forecast_time
SELECT
	forecast_time,
	id,
	value,
	1 AS mask
FROM values
WHERE ts = forecast_time
```

Импорт актуальных значений метрик

```sql
INSERT INTO values_forecast (ts, id, value, mask)
WITH
	toDateTime('2019-12-12 00:00:00') AS forecast_time
SELECT
	forecast_time,
	id,
	value,
	1 AS mask
FROM values
WHERE ts = forecast_time
```

Защита от merge в SummingMergeTree

```sql
INSERT INTO values_forecast (ts, id, forecast_1, mask)
WITH
    toDateTime('2019-12-12 00:00:00') AS forecast_time
SELECT
    forecast_time,
    id,
    forecast,
    2 AS mask
FROM (
    /* Подзапрос для модели 1 */
)
```

Вставка данных от модели #1

```sql
INSERT INTO values_forecast (ts, id, forecast_2, mask)
WITH
    toDateTime('2019-12-12 00:00:00') AS forecast_time
SELECT
    forecast_time,
    id,
    forecast,
    4 AS mask
FROM (
    /* Подзапрос для модели 2 */
)
```

Вставка данных от модели #2

```
OPTIMIZE TABLE values_forecast
PARTITION
('2019-12-12', '2019-12-12 00:00:00')
FINAL
```

Time to merge everything!

Реализация на SummingMergeTree

- Просчёт метрик для всех моделей за один проход (без JOIN)
- Легко понять, от каких моделей есть forecast
- Write amplification (do not write zero columns? :)
- Ограниченное число моделей (63)
- Можно исправить?

```sql
CREATE TABLE values_forecast
(
	`dt` MATERIALIZED toDate(ts),
	`ts` DateTime,
	`id` UInt64 CODEC(Delta, ZSTD),
	`value` Float64,
	`forecast_1` Float64 DEFAULT 0 COMMENT 'LinearRegression(steps=6)',
	`forecast_2` Float64 DEFAULT 0 COMMENT 'WMA(steps=6)',
	`forecast_3` Float64 DEFAULT 0 COMMENT 'Naive',
	`models` AggregateFunction(groupUniqArray, String),
	`mask` UInt64
)
ENGINE = SummingMergeTree
PARTITION BY (dt, ts)
ORDER BY id
```

```
CREATE TABLE values_forecast
(
    `dt` MATERIALIZED toDate(ts),
    `ts` DateTime,
    `id` UInt64 CODEC(Delta, ZSTD),
    `value` Float64,
    `forecast_1` Float64 DEFAULT 0 COMMENT 'LinearRegression(steps=6)',
    `forecast_2` Float64 DEFAULT 0 COMMENT 'WMA(steps=6)',
    `forecast_3` Float64 DEFAULT 0 COMMENT 'Naive',
    `models` AggregateFunction(groupUniqArray, String),
    `mask` UInt64
)
ENGINE = SummingMergeTree
PARTITION BY (dt, ts)
ORDER BY id
```

При INSERT - имя модели

Всегда пишем “1”

Метрики
качества
моделей

Текущая ситуация

- Прогнали все модели
- Соединили с реальным значением
- Надо выбрать наиболее адекватную модель
- Выбираем модель с наименьшим значением ошибки

Ошибки

- Mean Absolute Error

$$MAE = \frac{1}{N}\sum_{i=0}^{N-1} |\mathbf{y}_i - \hat{\mathbf{y}}_i|$$

- Root Mean Squared Error

$$MSE = \frac{\sum_{i=0}^{N-1}(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2}{N}$$

- Mean Absolute Percentage Error

$$\mathrm{M} = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

Ещё ошибки

- Symmetric mean absolute percentage error

$$\text{SMAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|F_t - A_t|}{(|A_t| + |F_t|)/2}$$

- Weighted MAPE

$$WMAPE = \frac{\sum |Actual - Forecast|}{\sum Actual}$$

```
/* MAE */
sum(abs(value - forecast))

/* MSE */
sumIf(pow(value - forecast), 2), ts < next_time - frequency ) / count()

/* MAPE */
100 / count() * sum( abs(value - forecast) / value )

/* SMAPE */
100 / count() * sum(  abs(forecast - value) / ( abs(value) + abs(forecast) ) / 2  )

/* WMAPE */
sum( abs(actual - forecast) ) / sum(actual)
```

```sql
WITH
    [
        'LinearRegression(steps=6)',
        'WMA(steps=6)',
        'Naive'
    ] AS model_names,
    [
        sum(abs(value - forecast_1)),
        sum(abs(value - forecast_2)),
        sum(abs(value - forecast_3))
    ] AS quality_metrics
SELECT
    id,
    arraySort((x, y) -> y, model_names, quality_metrics)[1] AS best_model
FROM values_forecast
GROUP BY id
```

$$MAE = \frac{1}{N} \sum_{i=0}^{N-1} |\mathbf{y}_i - \hat{\mathbf{y}}_i|$$

```
WITH
    [
        'LinearRegression(steps=6)',
        'WMA(steps=6)',
        'Naive'
    ] AS model_names,
    [
        sum(abs(value - forecast_1)),
        sum(abs(value - forecast_2)),
        sum(abs(value - forecast_3))
    ] AS quality_metrics
SELECT
    id,
    arraySort((x, y) -> y, model_names, quality_metrics)[1] AS best_model
FROM values_forecast
GROUP BY id
```

```sql
WITH
    [
        'LinearRegression(steps=6)',
        'WMA(steps=6)',
        'Naive'
    ] AS model_names,
    [
        sum(abs(value - forecast_1)),
        sum(abs(value - forecast_2)),
        sum(abs(value - forecast_3))
    ] AS quality_metrics
SELECT
    id,
    arraySort((x, y) -> y, model_names, quality_metrics)[1] AS best_model
FROM values_forecast
GROUP BY id
```

```sql
WITH
    [
        'LinearRegression(steps=6)',
        'WMA(steps=6)',
        'Naive'
    ] AS model_names,
    [
        sum(abs(value - forecast_1)),
        sum(abs(value - forecast_2)),
        sum(abs(value - forecast_3))
    ] AS quality_metrics
SELECT
    id,
    arraySort((x, y) -> y, model_names, quality_metrics)[1] AS best_model
FROM values_forecast
GROUP BY id
```

```sql
WITH
    [
        'LinearRegression(steps=6)',
        'WMA(steps=6)',
        'Naive'
    ] AS model_names,
    [
        sum(abs(value - forecast_1)),
        sum(abs(value - forecast_2)),
        sum(abs(value - forecast_3))
    ] AS quality_metrics
SELECT
    id,
    arraySort((x, y) -> y, model_names, quality_metrics)[1] AS best_model
FROM values_forecast
GROUP BY id
```

Имя модели с минимальным значением ошибки

Выводы

- Уже сегодня доступны операции с timeseries

Выводы

- Уже сегодня доступны операции с timeseries
- Есть потребность в нативной реализации
  - LOSS-функций (MAE, MASE, etc.)
  - моделей (Polynomial Regression, ARIMA)
  - преобразований (Box-Cox, Kalman, Fourier)

Выводы

- Уже сегодня доступны операции с timeseries
- Есть потребность в нативной реализации
  - LOSS-функций (MAE, MASE, etc.)
  - моделей (Polynomial Regression, ARIMA)
  - преобразований (Box-Cox, Kalman, Fourier)
- Но всё это обязательно случится!

СПАСИБО!

badoo

by MagicLab