Yandex

Yandex

Machine Learning with ClickHouse

Nikolai Kochetov, ClickHouse developer

Experimental dataset

NYC Taxi and Uber Trips

- Where to download:
  https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
- How to import data into ClickHouse:
  https://clickhouse.yandex/docs/en/getting_started/example_datasets/nyc_taxi/
- What you can also read:
  https://toddwschneider.com/posts/
  analyzing-1-1-billion-nyc-taxi-and-uber-trips-with-a-vengeance/

External Tools

Tools you got used to

Small sample of data is enough to start

All you need is to get it from ClickHouse

Couple of lines for Python + Pandas

```
import requests
import io
import pandas as pd

url = 'http://127.0.0.1:8123?query='
query = 'select * from trips limit 1000 format TSVWithNames'
resp = requests.get(url, data=query)
string_io = io.StringIO(resp.text)
table = pd.read_csv(string_io, sep="\t")
```

Table (part)

	dropoff_ntaname	trip_distance	total_amount	pickup_datetime	pickup_longitude	pickup_latitude
0	West Village	0.9	5.40	2009-01-01 00:00:00	-73.997484	40.725954
1	Battery Park City-Lower Manhattan	0.8	4.60	2009-01-01 00:00:04	-74.011594	40.708831
2	Washington Heights North	9.1	23.40	2009-01-01 00:00:21	-74.007649	40.742992
3	Lenox Hill-Roosevelt Island	1.2	5.80	2009-01-01 00:00:27	-73.974475	40.764890
4	Hudson Yards-Chelsea-Flatiron-Union Square	2.1	9.00	2009-01-01 00:00:28	-73.972977	40.762695
5	Upper East Side-Carnegie Hill	4.4	14.60	2009-01-01 00:00:29	-74.005849	40.740222
6	North Side-South Side	0.9	5.80	2009-01-01 00:00:30	-73.952676	40.726926
7	Upper West Side	3.0	12.24	2009-01-01 00:00:36	-73.955911	40.764011
8	Lincoln Square	0.7	5.00	2009-01-01 00:00:39	-73.975878	40.782039
9	Clinton	5.3	16.00	2009-01-01 00:00:46	-73.953738	40.806762
10	West Village	1.3	6.60	2009-01-01 00:00:48	-73.987710	40.732993

How to sample data

You already know it!

- LIMIT N
- WHERE condition
- SAMPLE x OFFSET y

How to sample data

LIMIT N

```
SELECT
    min(pickup_date),
    max(pickup_date)
FROM
(
    SELECT pickup_date
    FROM trips_mergetree_third
    LIMIT 1000
)

┌─min(pickup_date)─┬─max(pickup_date)─┐
│       2009-01-01 │       2009-01-01 │
└──────────────────┴──────────────────┘
```

How to sample data

WHERE rand() % N < M

```
SELECT trip_id FROM trips WHERE (rand() % 1000) = 0
LIMIT 1 SETTINGS max_threads = 1

┌─trip_id─┐
│ 960186089 │
└─────────┘

SELECT trip_id FROM trips WHERE (rand() % 1000) = 0
LIMIT 1 SETTINGS max_threads = 1

┌─trip_id─┐
│ 335608036 │
└─────────┘
```

How to sample data

WHERE hash(...) % N < M

```
SELECT trip_id FROM trips WHERE (sipHash64(trip_id) % 1000) = 0
LIMIT 1 SETTINGS max_threads = 1

┌─trip_id─┐
│ 48203111 │
└─────────┘

SELECT trip_id FROM trips WHERE (sipHash64(trip_id) % 1000) = 0
LIMIT 1 SETTINGS max_threads = 1

┌─trip_id─┐
│ 48203111 │
└─────────┘
```

How to sample data

SAMPLE x OFFSET y

Must specify an expression for sampling

- Optimized by PK
- Fixed dataset for fixed sample query
- Only for MergeTree

How to sample data

SAMPLE x OFFSET y

```
CREATE TABLE trips_sample_time
(
    pickup_datetime DateTime
)
ENGINE = MergeTree
ORDER BY sipHash64(pickup_datetime) -- Primary Key
SAMPLE BY sipHash64(pickup_datetime) -- expression for sampling
```

SAMPLE BY expression must be evenly distributed!

How to sample data

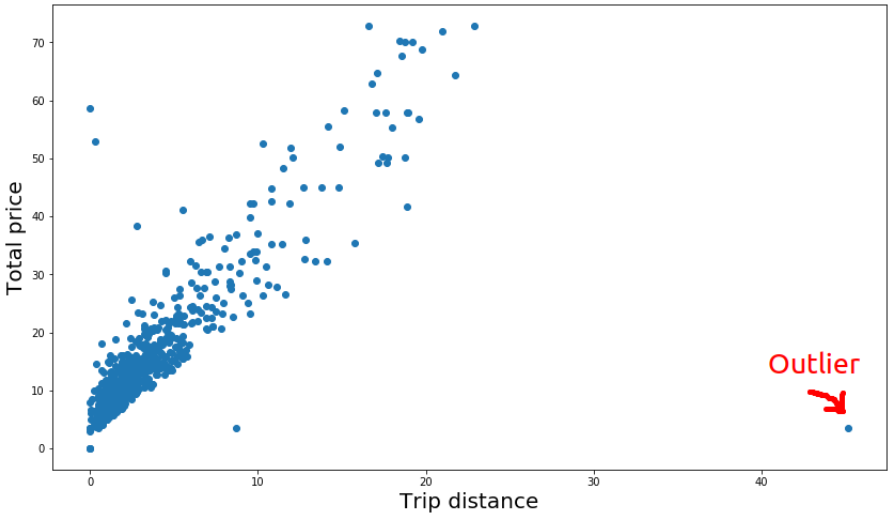
SAMPLE x OFFSET y

```
SELECT count() FROM trips_sample_time

432992321

1 rows in set. Elapsed: 0.413 sec. Processed 432.99 million rows
```

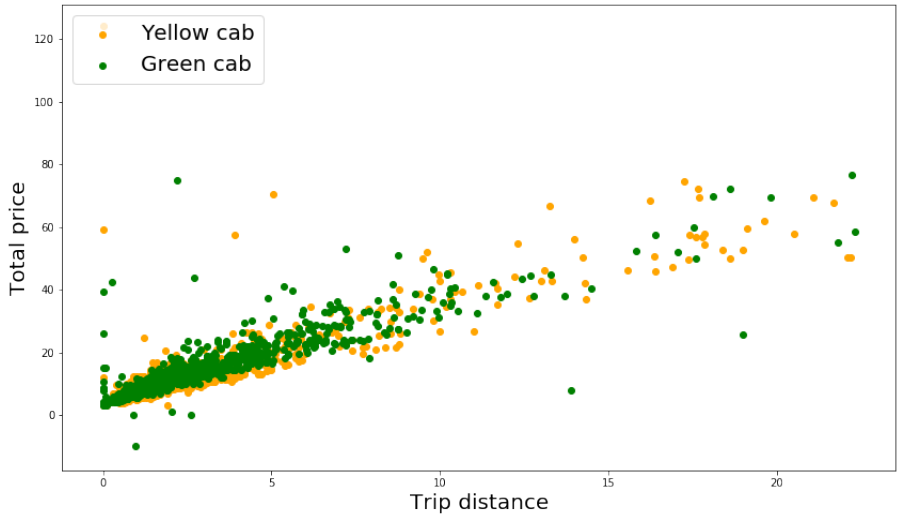
Query with sampling reads less rows!

```
SELECT count() FROM trips_sample_time SAMPLE 1 / 3 OFFSET 1 / 3

144330770

1 rows in set. Elapsed: 0.276 sec. Processed 144.33 million rows
```

Explore your data



Explore your data



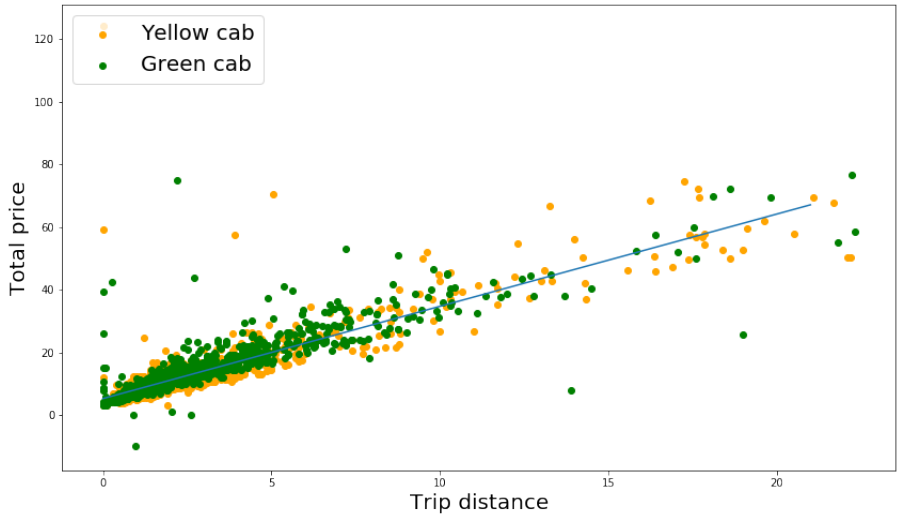
Linear regression in ClickHouse

You can use all your data in one SQL query!

```
SELECT simpleLinearRegression(trip_distance, total_amount)
FROM trips
WHERE ((dropoff_datetime - pickup_datetime) < 5000)
AND (trip_distance < 25)
AND (total_amount != 0)
AND (trip_distance != 0)
AND (cab_type != 'uber')

┌─simpleLinearRegression(trip_distance, total_amount)─┐
│ (2.9427761862952964,5.24205006595278)               │
└─────────────────────────────────────────────────────┘
```

total_amount = 2.94 * trip_distance + 5.24

Linear regression result



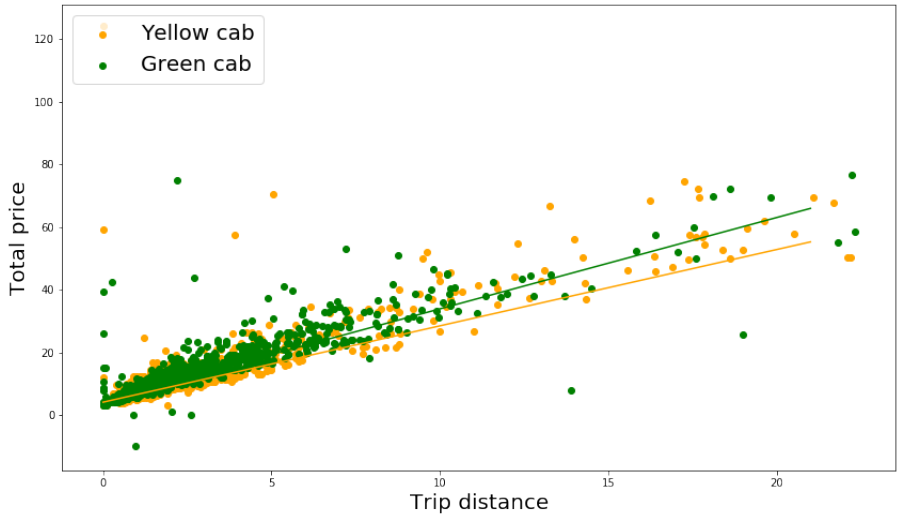
Linear regression in ClickHouse

Is implemented as Aggregate Function

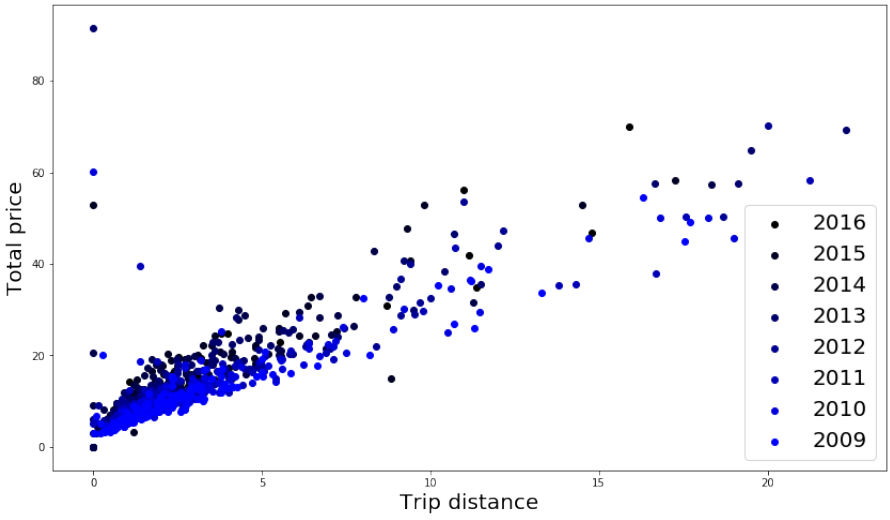
Means you can train multiple models in single SQL query

```
SELECT
    cab_type,
    simpleLinearRegression(trip_distance, total_amount)
FROM trips
WHERE <...>
GROUP BY cab_type

┌─cab_type─┬─simpleLinearRegression(trip_distance, total_amount)─┐
│ yellow   │ (2.4343401638740527,4.093962797316789)              │
│ green    │ (3.0585805837026796,4.433117172403966)              │
└──────────┴─────────────────────────────────────────────────────┘
```

Linear regression result



The influence of time

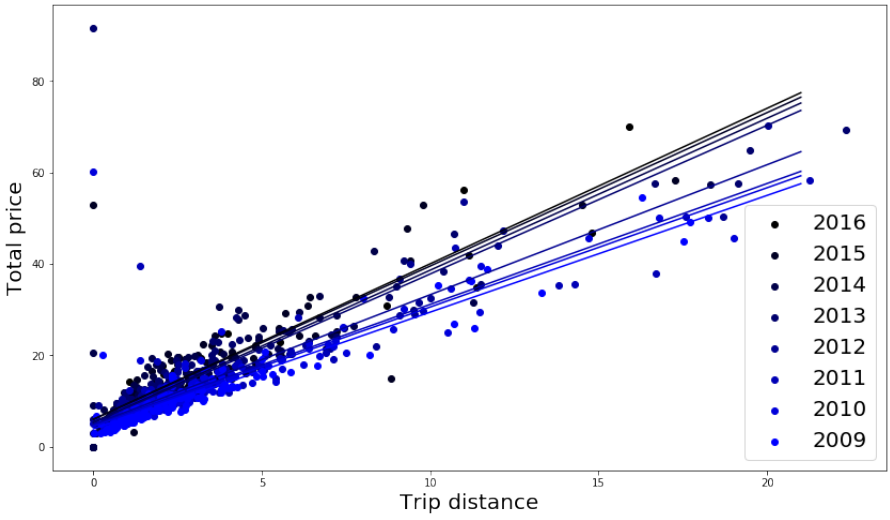


Let's train more models!

```
SELECT toYear(pickup_datetime) AS y,
       simpleLinearRegression(trip_distance, total_amount)
FROM trips WHERE <...> GROUP BY y
```

y	simpleLinearRegression(trip_distance, total_amount)
2009	(2.553562453857034,3.8870750415729884)
2010	(2.6092185690577705,4.446715211300636)
2011	(2.6442522510248594,4.6764411907968695)
2012	(2.8429103158356224,4.809762047068699)
2013	(3.2509270548953855,5.262739563061447)
2014	(3.3224390615105084,5.3788910690904395)
2015	(3.356450828179191,5.94053817327741)
2016	(3.4008040367471857,5.99253781758988)

Let's train more models!



Linear regression in ClickHouse

simpleLinearRegression supports only single factor

- You are welcome to contribute to https://github.com/clickhouse/ClickHouse

There are stochastic regression methods in ClickHouse

- stochasticLinearRegression
- stochasticLogisticRegression

Stochastic methods do support multiple factors.
That's not the most important difference.

Stochastic linear regression in ClickHouse

stochasticLinearRegression(parameters)(target, x_1, ..., x_N)

Available parameters:

- learning_rate
- l2_regularization
- batch_size
- optimizer: Adam, SGD, Momentum, Nesterov

All parameters are specified for **stochastic gradient descent**.
Related wiki page: https://en.wikipedia.org/wiki/Stochastic_gradient_descent

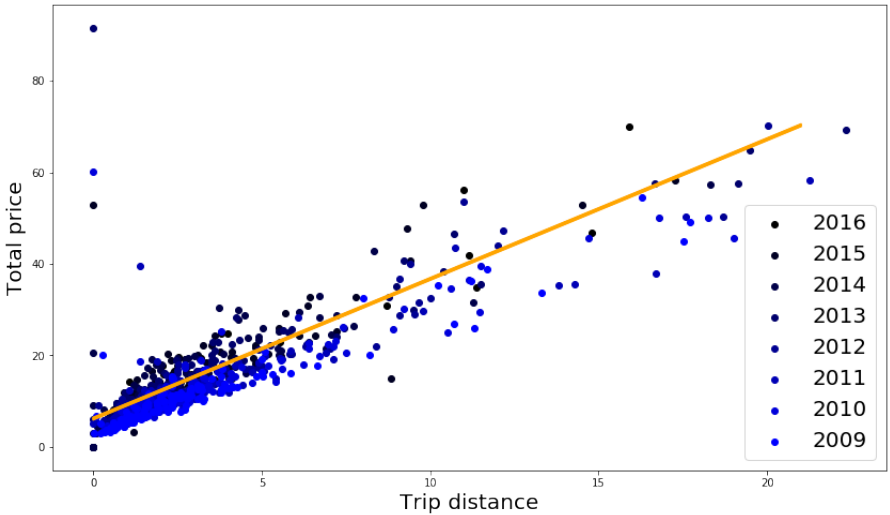
Stochastic model with default parameters

```
SELECT stochasticLinearRegression(
    total_amount, trip_distance, toYear(pickup_datetime) - 2009)
FROM trips WHERE <...>

 [3.050791112773074,0.07878654160131655,5.910850376181039]
```

total_amount = trip_distance * 3.05 + (year - 2009) * 0.08 + 5.91

Year doesn't seem to matter a lot for trained model

Stochastic model with default parameters



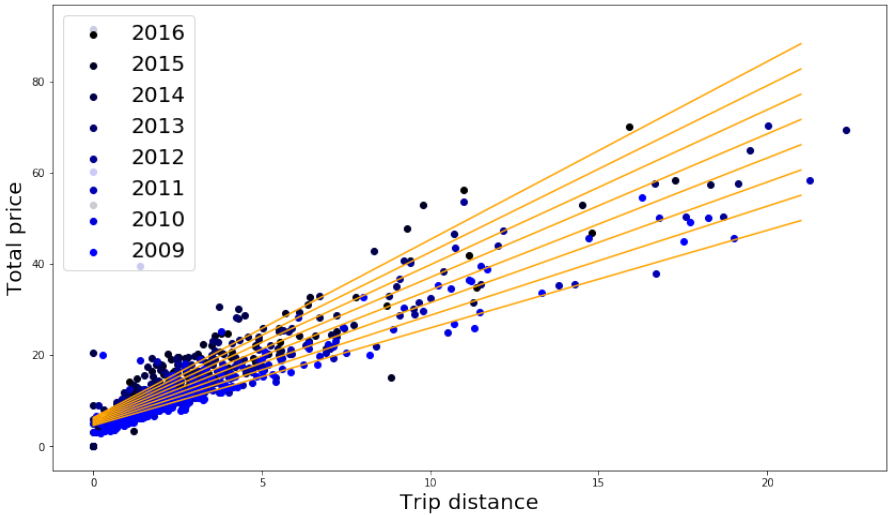
Stochastic model with default parameters

Actually, our last feature was not good
Annual price was increased not in total, by per mile!

```
SELECT stochasticLinearRegression(
    total_amount,
    trip_distance,
    (toYear(pickup_datetime) - 2009) * (trip_distance + 1))
FROM trips
WHERE <...> ORDER BY sipHash64(trip_id) ASC

[2.1387068697017 64,0.25152600248358253,4.5418692076782445]
```

That's better!

Stochastic model with default parameters



Models management in ClickHouse

How to store trained model

> You can store model as aggregate function state in a separate table

Example

```
CREATE TABLE models
ENGINE = MergeTree ORDER BY tuple() AS
SELECT
    stochasticLinearRegressionState(total_amount, trip_distance)
    AS model
FROM trips WHERE <...>
```

Note: stochasticLinearRegression**State** is used

Aggregate function state in ClickHouse

```
SELECT sum(number) FROM numbers(5) FORMAT TSV  --  0 + 1 + 2 + 3 + 4

10
```

Use sum**State** to get aggregate function state

```
SELECT sumState(number) FROM numbers(5) FORMAT TSV

\n\0\0\0\0\0\0\0  --  Binary data. ASCII for \n is 10.
```

Column has special type

```
SELECT toTypeName(sumState(number)) FROM numbers(5) FORMAT TSV

AggregateFunction(sum, UInt64)
```

Aggregate function state in ClickHouse

You can save aggregate function result into table.

```
CREATE TABLE tab ENGINE = Memory AS
SELECT sumState(number) AS x FROM numbers(5)
```

Use sum**Merge** to get final result

```
SELECT sumMerge(x) FROM tab FORMAT TSV

10
```

Function **finalizeAggregation** does the same

```
SELECT finalizeAggregation(x) FROM tab FORMAT TSV

10
```

How to store trained model

You can store model as aggregate function state in a separate table

Example

```
CREATE TABLE models
ENGINE = MergeTree ORDER BY tuple() AS
SELECT
    stochasticLinearRegressionState(total_amount, trip_distance)
    AS model
FROM trips WHERE <...>
```

Note: stochasticLinearRegression**State** is used

You can continue feeding model with new data!

How to read trained model

Let's read our models from table.

```
SELECT stochasticLinearRegressionMerge(model) FROM models

┌─stochasticLinearRegressionMerge(model)─┐
│ [3.098869825901077,5.428854306127582]  │
└────────────────────────────────────────┘
```

Note: stochasticLinearRegression**Merge** is used.

```
SELECT finalizeAggregation(model) FROM models

┌─stochasticLinearRegressionMerge(model)─┐
│ [3.098869825901077,5.428854306127582]  │
└────────────────────────────────────────┘
```

How to update trained model

```
CREATE TABLE models
ENGINE = AggregatingMergeTree ORDER BY tuple() AS
SELECT stochasticLinearRegressionState(...) AS model
FROM trips
WHERE <...> AND (toYear(pickup_date) = 2009)
```

Note: **Aggregating**MergeTree is used.

```
SELECT finalizeAggregation(model) FROM models

┌─finalizeAggregation(model)─┐
│ [2.0363653965934154,4.815617789622539] │
└────────────────────────────┘
```

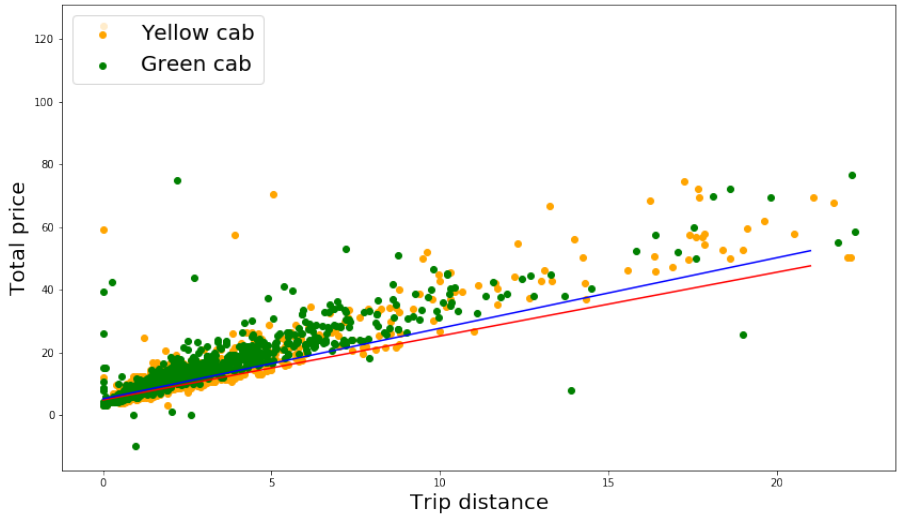
How to update trained model

```
INSERT INTO models
SELECT stochasticLinearRegressionState(...) AS model
FROM trips
WHERE <...> AND (toYear(pickup_date) = 2010)
```

FINAL allows to merge all data into single model

```
SELECT finalizeAggregation(model) FROM models FINAL

┌─finalizeAggregation(model)─┐
│ [2.250130791840091,5.16166254336767] │
└────────────────────────────┘
```

How to update trained model



How to apply trained model

There is a function **evalMLMethod(model, x_1, ..., x_N)**.

```
WITH (SELECT model FROM models) AS model
SELECT
    evalMLMethod(model, trip_distance),
    total_amount
FROM trips LIMIT 5
```

evalMLMethod(model, trip_distance)	total_amount
8.217837149438552	5.4
7.907950166848444	4.6
33.62856972182739	23.4
9.147498097208874	5.8
11.936480940519843	9

Store several trained models

```
CREATE TABLE models
ENGINE = MergeTree ORDER BY tuple() AS
SELECT
    toYear(pickup_datetime) AS year,
    stochasticLinearRegressionState(total_amount, trip_distance)
                            AS model
FROM trips WHERE <...>
GROUP BY year

Ok.
```

Apply several trained models

```
SELECT
    evalMLMethod(model, trip_distance),
    total_amount
FROM trips
LEFT JOIN models ON year = toYear(pickup_datetime)
LIMIT 5

┌─evalMLMethod(model, trip_distance)─┬─total_amount─┐
│                  8.087692004204174 │          5.4 │
│                  7.861181608305352 │          4.6 │
│                 26.661544467907536 │         23.4 │
│                  8.767223191900637 │          5.8 │
│                  10.80581675499003 │            9 │
└────────────────────────────────────┴──────────────┘
```

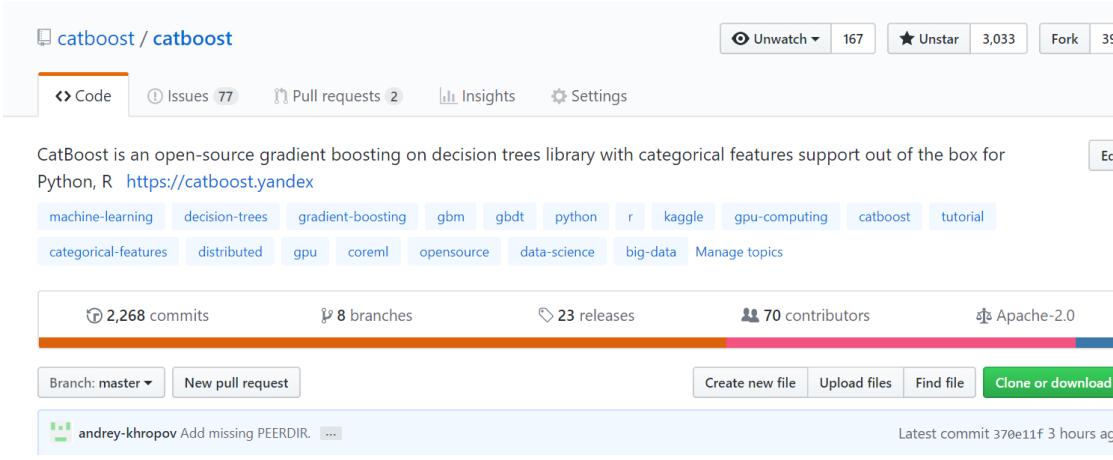
Evaluate error

```
SELECT sqrt(avg(diff * diff)) AS MSE
FROM
(
    SELECT evalMLMethod(model, trip_distance) - total_amount AS diff
    FROM trips_mergetree_third
    LEFT JOIN models ON year = toYear(pickup_datetime)
)
```

```
┌──────────────────MSE─┐
│ 4.145554613376103    │
└──────────────────────┘
```

External Models

CatBoost


catboost / catboost
Unwatch 167
Unstar 3,033
Fork 399
Code
Issues 77
Pull requests 2
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
kaggle
gpu-computing
catboost
tutorial
categorical-features
distributed
gpu
coreml
opensource
data-science
big-data
Manage topics
2,268 commits
8 branches
23 releases
70 contributors
Apache-2.0
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
andrey-khropov Add missing PEERDIR. ...
Latest commit 370e11f 3 hours ago

Gradient Boosting

General advantages

- Best solution for heterogeneous data
- Works well for small data
- Easy to use

Yandex
CatBoost

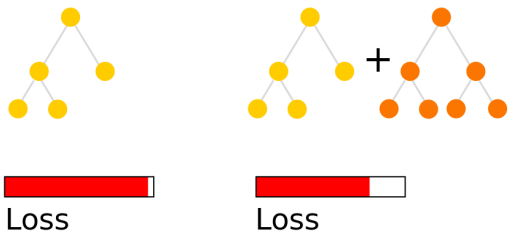
CatBoost advantages

- Good quality for default parameters
- Sophisticated categorical features support
- Models analysis tools

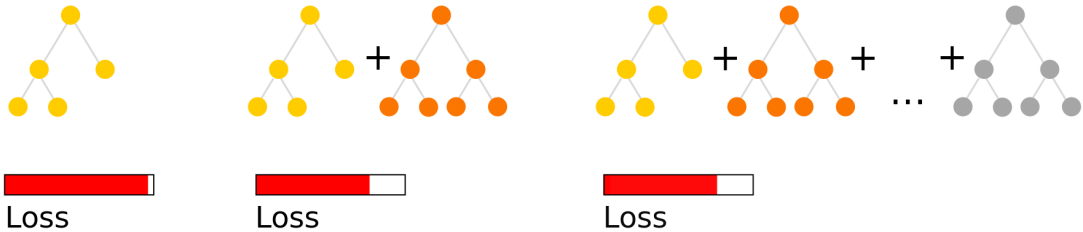
Gradient Boosting

Loss

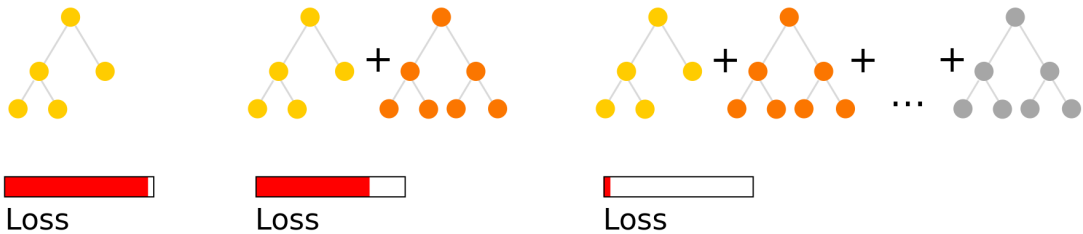
Gradient Boosting



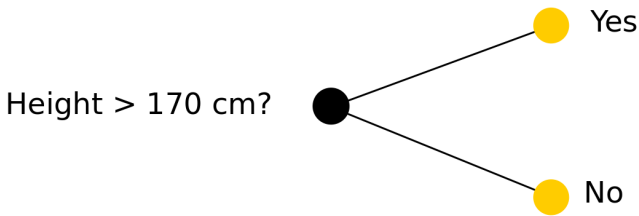
Gradient Boosting



Gradient Boosting

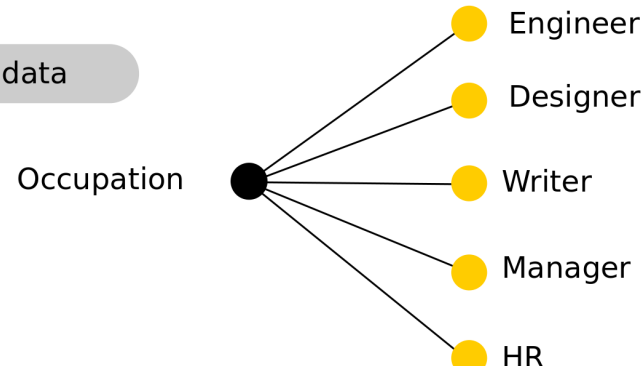


Numerical features



Categorical features

Categorical data

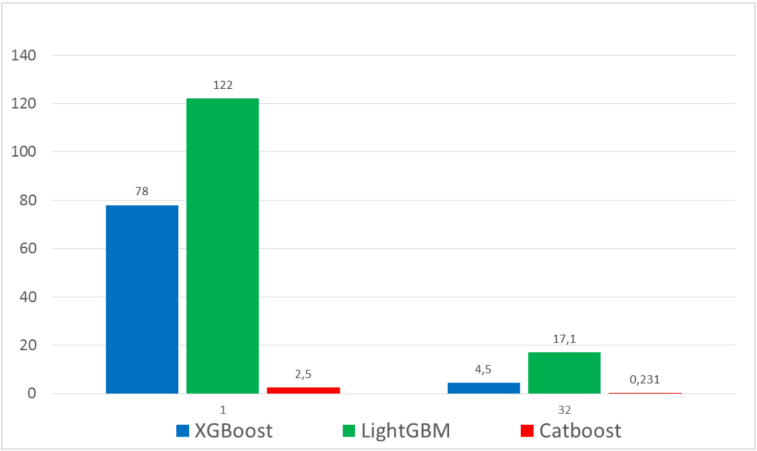


Algorithm comparison

	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

Prediction time



Applying CatBoost models in ClickHouse

CatBoost models in ClickHouse

Steps to do:

- Train model and save it as `my_favorite_model.bin`
- Build CatBoost evaluation library. Follow the instruction at https://catboost.ai/docs/concepts/c-plus-plus-api_dynamic-c-pluplus-wrapper.html
  You need to get `libcatboostmodel.so`
- Update ClickHouse configuration file

```
cat /etc/clickhouse-server/conf.d/models_config.xml
<yandex>
    <catboost_dynamic_library_path>/pah/to/libcatboostmodel.so
    </catboost_dynamic_library_path>
    <models_config>/path/to/models/*_model.xml</models_config>
</yandex>
```

CatBoost models in ClickHouse

Steps to do:

- Train model and save it as `my_favorite_model.bin`
- Build CatBoost evaluation library
- Update ClickHouse configuration file
- Add model description which matches `models_config`

```
<models>
    <model>
        <!-- Model type. Now catboost only. -->
        <type>catboost</type>
        <!-- Model name. -->
        <name>my_model</name>
        <!-- Path to trained model. -->
        <path>/path/to/my_favorite_model.bin</path>
    </model>
</models>
```

CatBoost models in ClickHouse

Steps to do:

- Train model and save it as `my_favorite_model.bin`
- Build CatBoost evaluation library
- Update ClickHouse configuration file
- Add model description which matches `models_config`

More details:

- Tutorial
  https://github.com/ClickHouse/clickhouse-presentations/blob/master/tutorials/catboost_with_clickhouse_en.md
- Documentation

Train CatBoost model

```
from catboost import CatBoostRegressor

# Initialize data
train_data = df[['trip_distance', 'cab_type', 'year']]
train_labels = df['total_amount']

# Initialize CatBoostRegressor
model = CatBoostRegressor(iterations=1000, learning_rate=0.1, depth=6)
# Fit model
model.fit(train_data, train_labels, cat_features=[1])
# Save model
model.save_model('trip_price.bin')
```

Apply CatBoost model in ClickHouse

```
SELECT   -- specify cat features at the end
    modelEvaluate('trip_price', trip_distance,
                  toYear(pickup_datetime) - 2006,
                  cab_type) AS prediction,
    total_amount
FROM trips LIMIT 5

┌──────────prediction─┬─total_amount─┐
│  8.096942220719471  │          5.4 │
│ 7.6722147935759955  │          4.6 │
│ 26.433542947767798  │         23.4 │
│  8.506852274026288  │          5.8 │
│ 11.555079604924444  │            9 │
└─────────────────────┴──────────────┘
```

Apply CatBoost model in ClickHouse

```
SELECT sqrt(avg(diff * diff)) AS MSE
FROM
(
    SELECT modelEvaluate('trip_price', ...) - total_amount AS diff
    FROM trips
    WHERE <...>
)

┌──────────────────MSE─┐
│ 3.8519197052953755 │
└──────────────────────┘
```

Models comparison

	MSE
simpleLinearRegression	4.72
simpleLinearRegression, group by year	4.39
stochasticLinearRegression, 3 features	4.43
stochasticLinearRegression, group by year	4.15
CatBoost (trained on 10000 trips)	3.85

TODO

List of features it is good to implement:

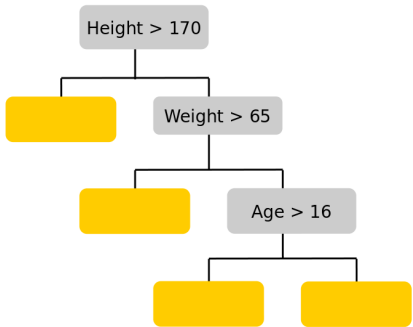
- Loss functions (as aggregate function): MSE, MAE, logloss, ...
- Shuffle for minibatches (as table function)
- Table function to sample data from table with repetition
- Support for multiple features in simpleLinearRegression
- Support for multiple loss functions in stochasticLinearRegression (MSE now)
- Named parameters for aggregate functions
- More aggregate functions for ML

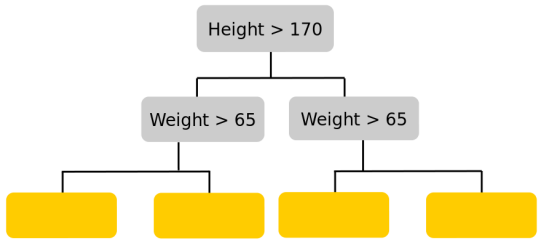
More detailed description https://github.com/ClickHouse/ClickHouse/issues/7345
You are welcome to contribute to https://github.com/clickhouse/ClickHouse

Thank you!

QA

Symmetric trees





Categorical features support

One-hot encoding

Statistics based on category and category plus label value

Usage of several permutations

Greedy constructed feature combinations