Yandex

Yandex
CatBoost

CatBoost: Gradient Boosting for data with both numerical and text features

Kirillov Stanislav,
Head of CatBoost team

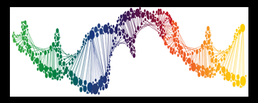
Plan

- Gradient boosting and CatBoost overview
- CatBoost in Yandex and in the wild
- Supported feature types and text support
- Big data support and other new features

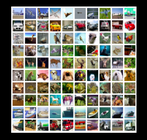
Data at hand

Unstructured data

Music



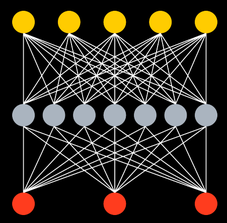
DNA



Images

Text

End2End with Deep NN

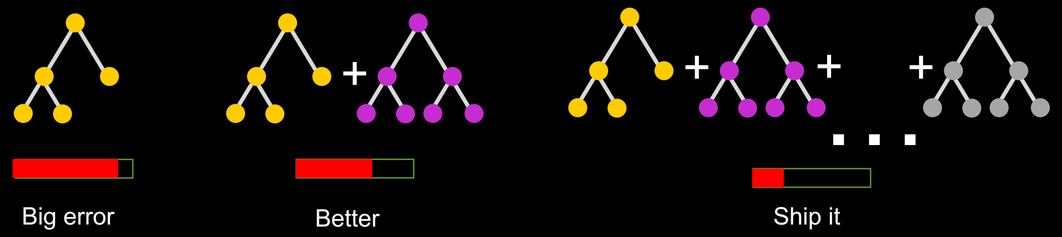


Tabular (or structured) data

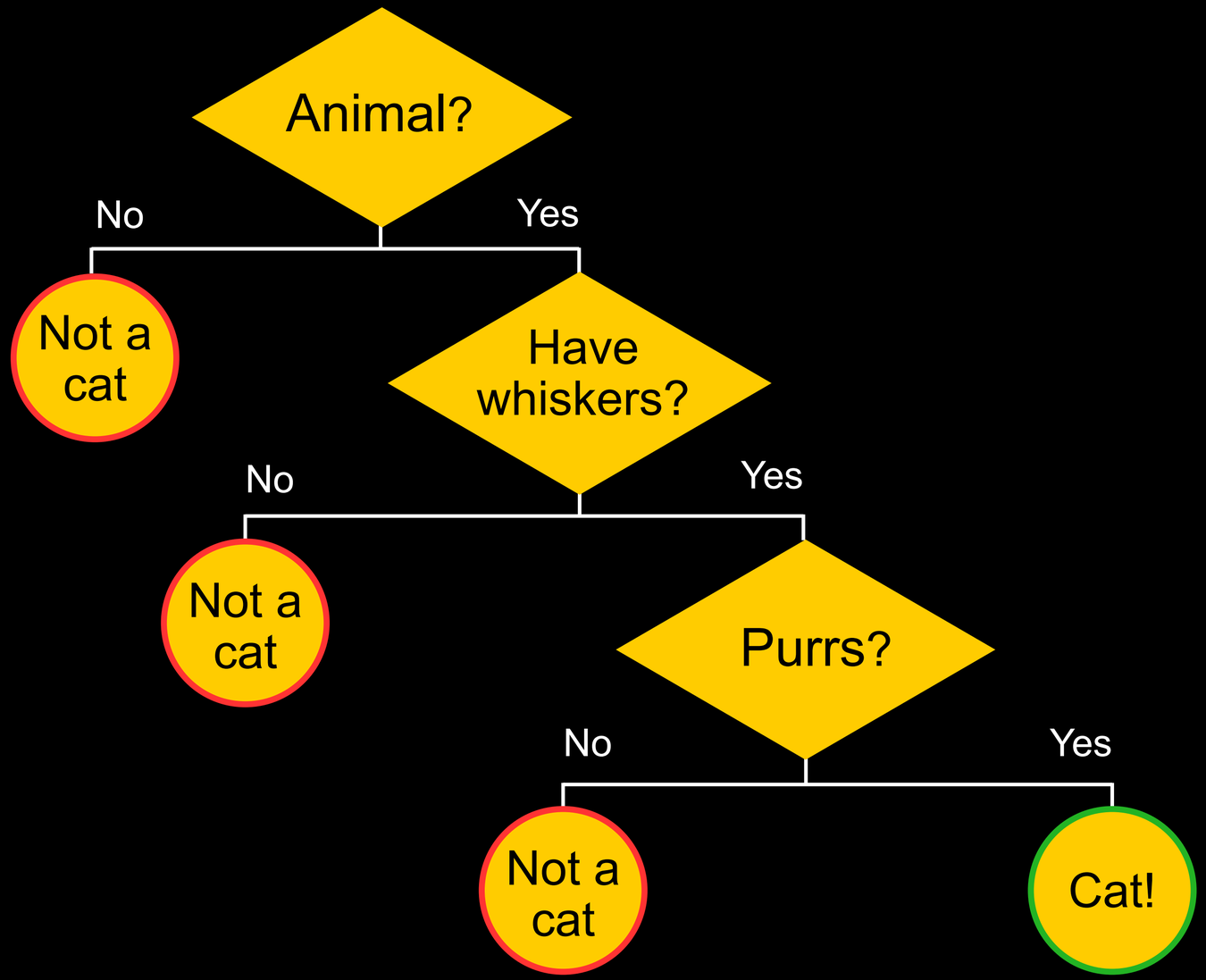
Well engineered features

Music track length	Year	Rating	Label
2	1990	3	1
3	1950	5	0
15	1970	4	1

GBDT

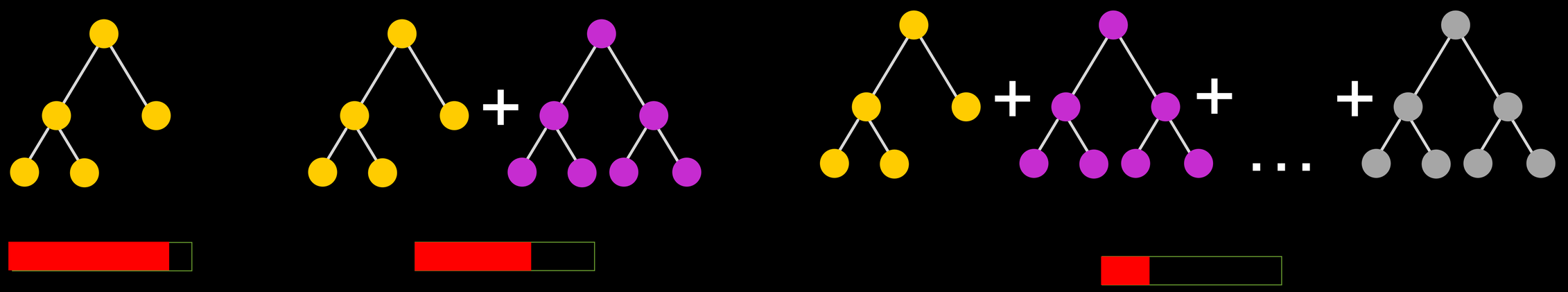


Tabular data? Decision trees!



Gradient boosted decision trees

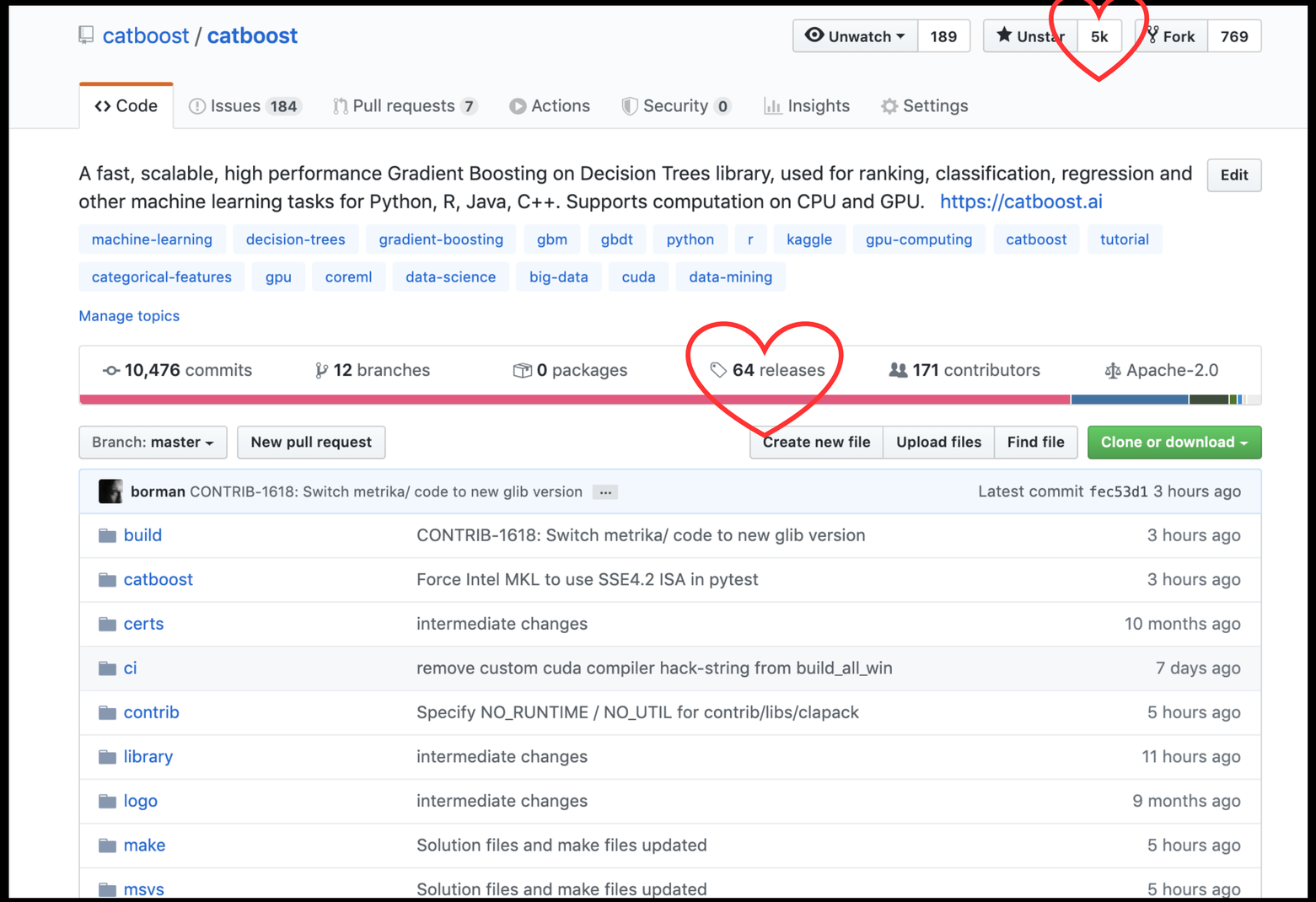
- State-of-the-art quality on tabular data
- Easy to use, no sophisticated parameter tuning
- Works well with small data and scales for big data problems



Main Boosting libraries

CatBoost

- 50K pip installs per week
- 5K stars on github
- 64 releases

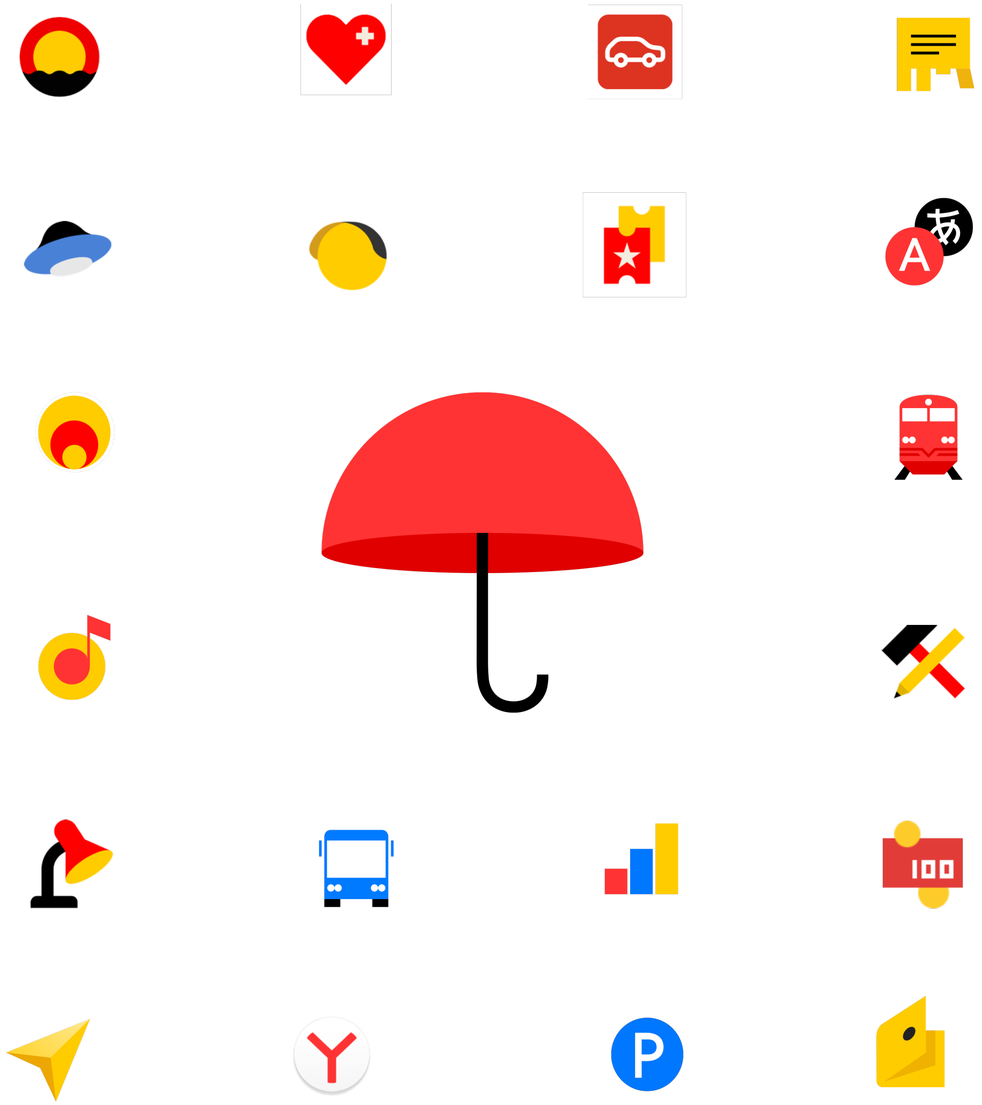


CatBoost advantages

- Good quality with default parameters
- Sophisticated categorical and text features support
- Model analysis tools
- Set of tools to make GBDT usage easier

CatBoost in Yandex

- Yandex.Zen
- Yandex.Music
- Yandex.Self-Driving Cars
- Yandex.Search
- Yandex.Ads
- Yandex.Weather
- Yandex Alice
- Practically everywhere!



Yandex.Search

Task?

- Search document order prediction

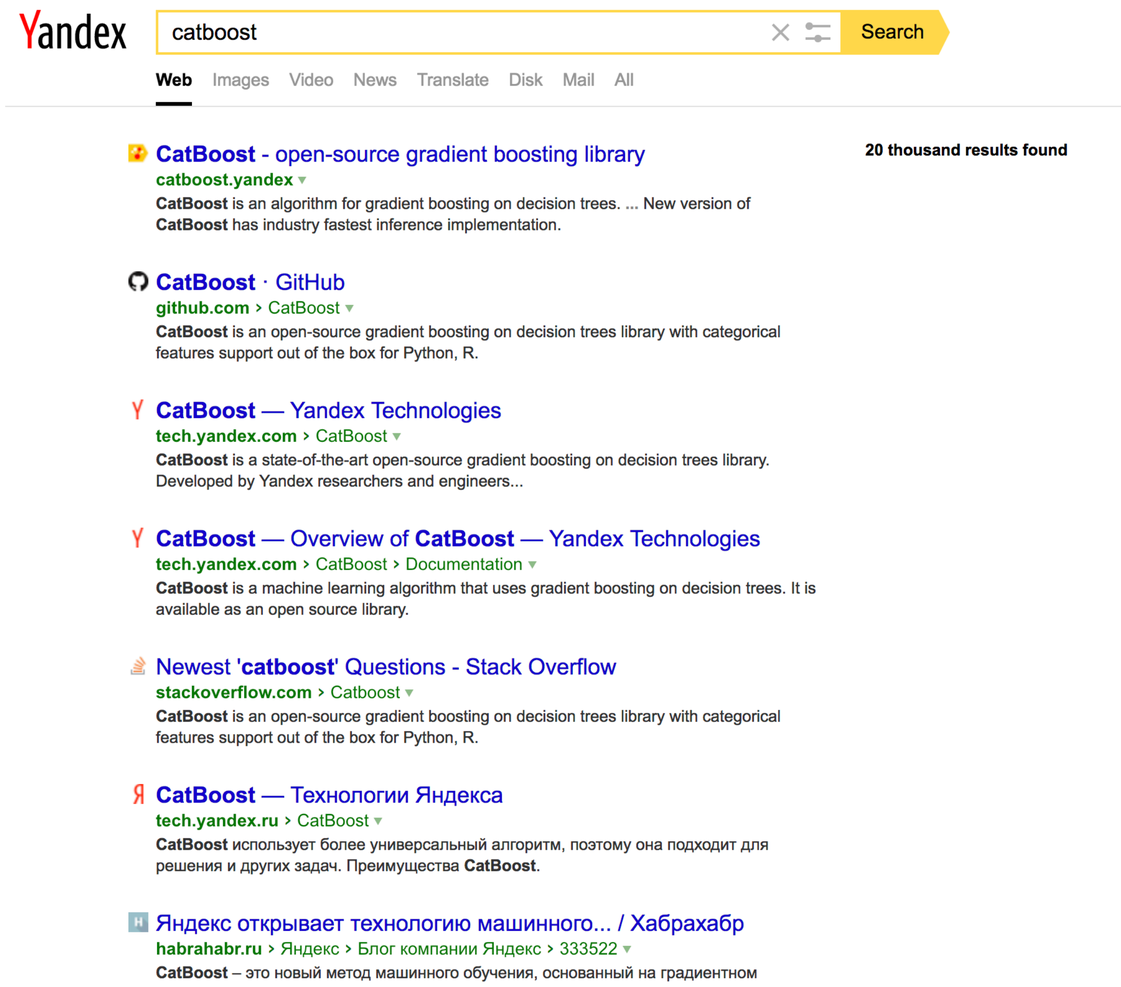
Task type: ranking

Dataset features:

- Classic features (PageRank, BM25 and others)
- Neural Networks output

CatBoost features used:

- YetiRankPairwise target
- Distributed GPU training
- Model blending
- Feature importance analysis
- Ranking analysis



Yandex.Weather

Task?

- Cloudiness type and temperature prediction

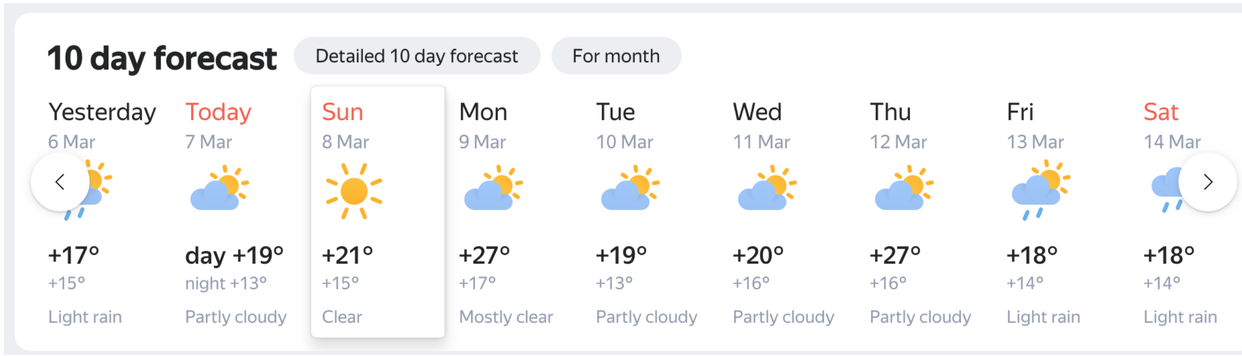
Task type: multiclassification and regression

Dataset features

- Physical weather model output
- Neural network output
- Online-data from weather stations
- Weather historical data

CatBoost features used:

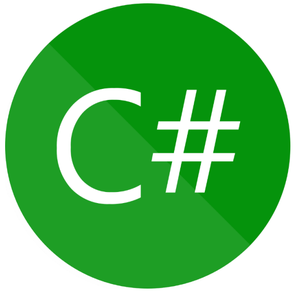
- Multiclassification target and RMSE (for temperature)
- GPU training
- Feature importance analysis
- Training process visualization

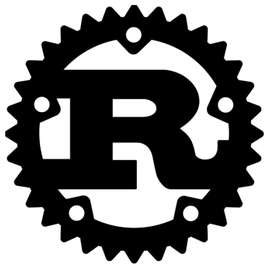


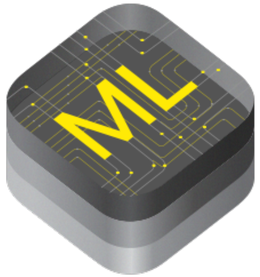
CatBoost in the Wild

- Recommendations at Netflix
- Hotel ranking in Aviasales
- Protection against bots in CloudFlare
- Particle classification in CERN
- Medical research at University of NSW Sydney
- Destination prediction in Careem taxi service
- ML competitions on Kaggle

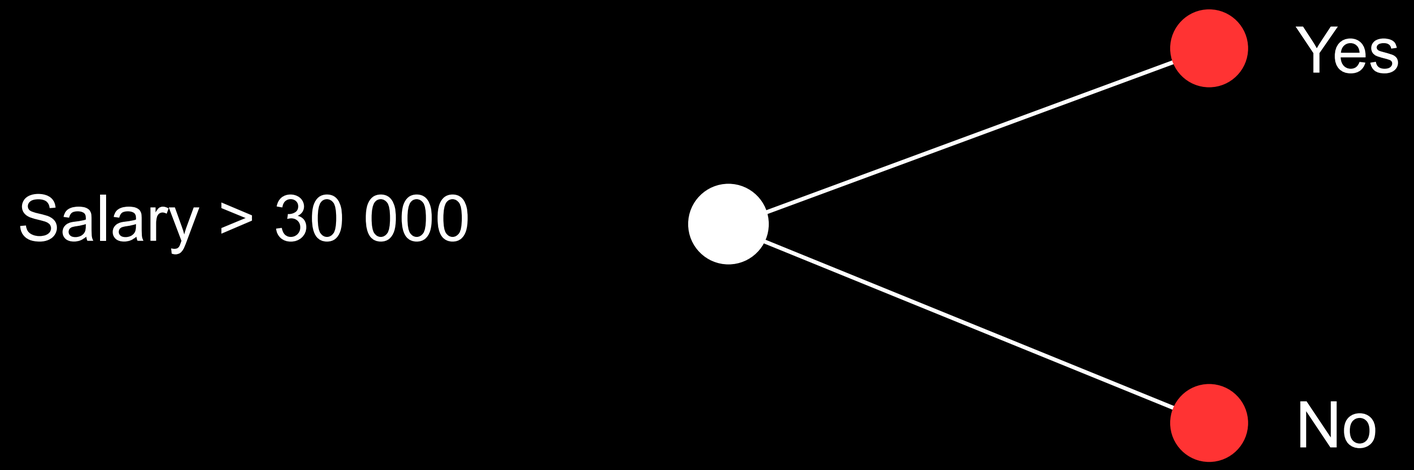
Integration in production







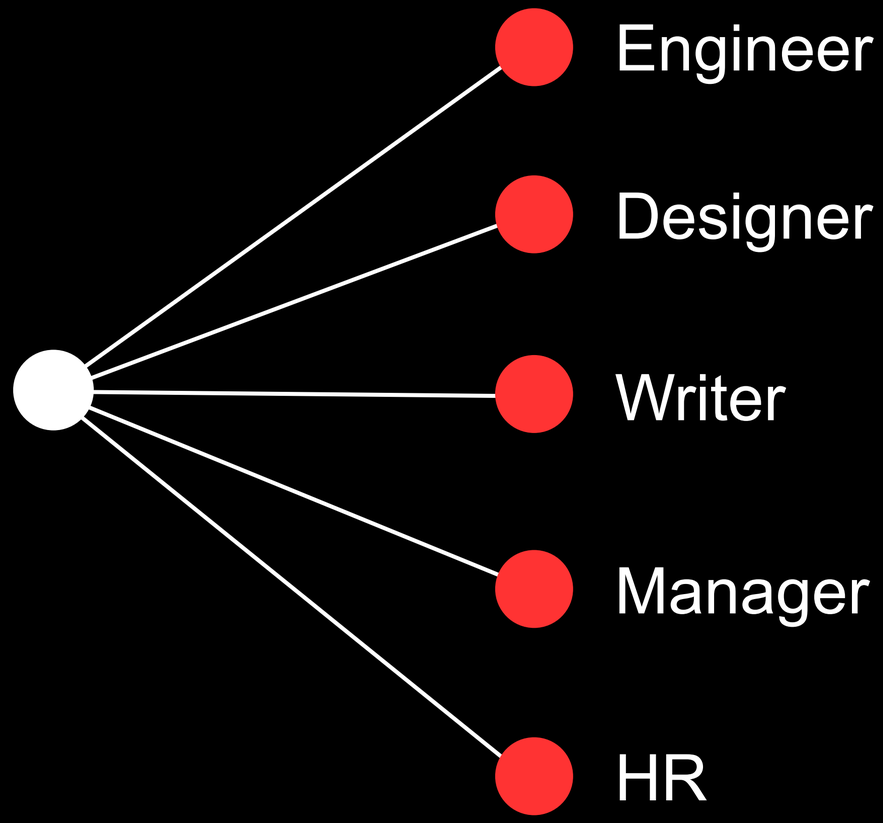
Numerical features



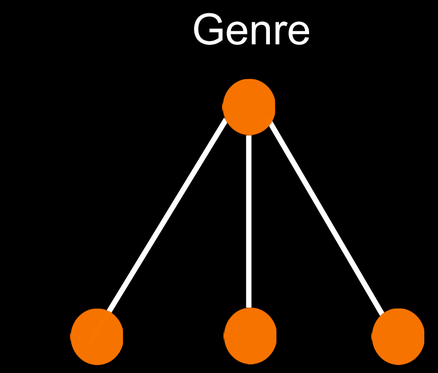
Categorical features

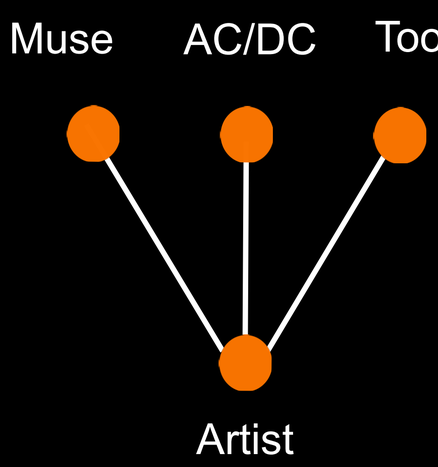
Categorical data

Occupation



Categorical features handling





One-hot encoding

Statistics based on category

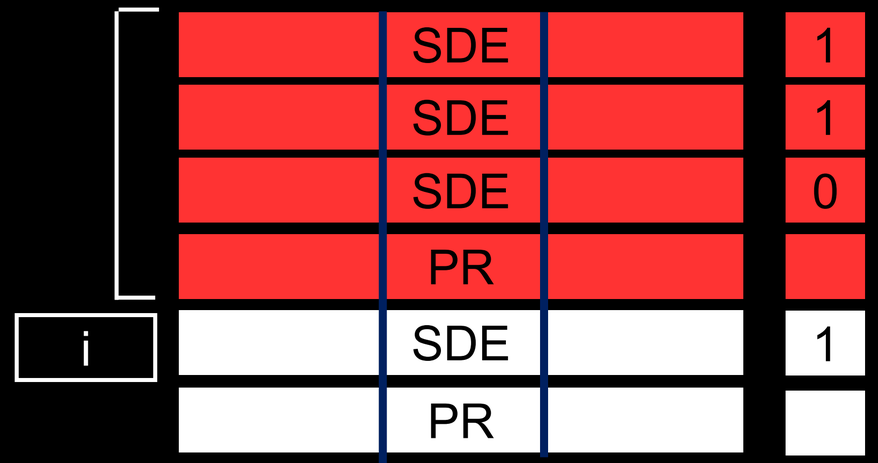
Category-based

Label based:
calculated “online”

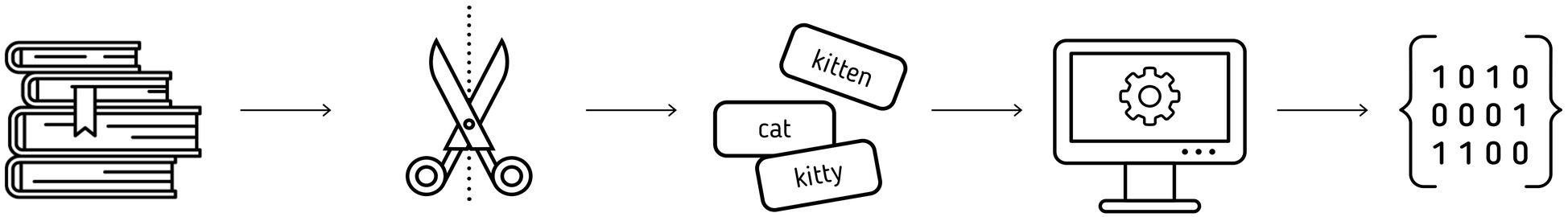
Greedy search for
combinations

Online features for categories

For every sample “online” feature is calculated using objects with the same category before this one



Text features in CatBoost



Input text

Tokenizing

Creating dictionary

Applying feature

These computed estimators features are passed to training procedure

Text features

at first i was afraid i was petrified kept thinkin i could never live without you by my side

Dictionary

[0, 1, 2, 3, 4, 2, 3, 5, 6, 7, 2, 8, 9, 10, 11, 12, 13, 14, 15]

BM25

Naïve Bayes

Bag-of-Words

…

Preprocessing stage

- Split into words
- Process numbers and punctuation
- Build letter and word dictionaries
- Build ngram dictionaries or BPE

Bag of Words

- Default: word unigrams + word bigrams
- For a given dictionary (set of tokens)– is this token present in text?

Dictionary 1:
yesterday, petrified
Dictionary 2:
at+first, i+has,
could+never, …

at first i was afraid i
was petrified kept
thinkin i could never
live without you by
my side

Feature values:
0,1,1,0,1

Naïve Bayes

- For every class a new feature P(Class|Text)
- P(Class|Text) is replaced with P(Class) * Π_i P(word$_i$ | Class)
- **Most importantly**: calculate it “online”

BM-25 for MultiClass

- A new numerical feature for every class of multi-classification
- TF – frequency of a word in text
- IDF – inverted frequency of a word in a “document”, where “document” is a **concatenation of all texts in this class**
- **Most importantly**: calculate it “online”

Text processing in CatBoost

- catboost.Tokenizer
- catboost.Dictionary

Advantages:

- Fast
- Customizable
- Production-ready
- Can be used with other libraries, including Neural Networks

Text features examples

› Rotten Tomatoes: movie review

Numerical
- runtime
- box_office – amount of money raised by ticket sales

Categorical
- critic - name of reviewer
- publisher - journal where the review was published

Text
- review - review of a movie, that was written by a critic
- genres - list of genres that are suitable for this film

review

One very long, dark ride.

genres

Action and Adventure | Art House and International | Drama | Mystery and Suspense

Profit from text features

Accuracy on Rotten Tomatoes

Numerical + Caterogical	+ BOW	+ Online Text Features
0.4592	0.4616	**0.4714**

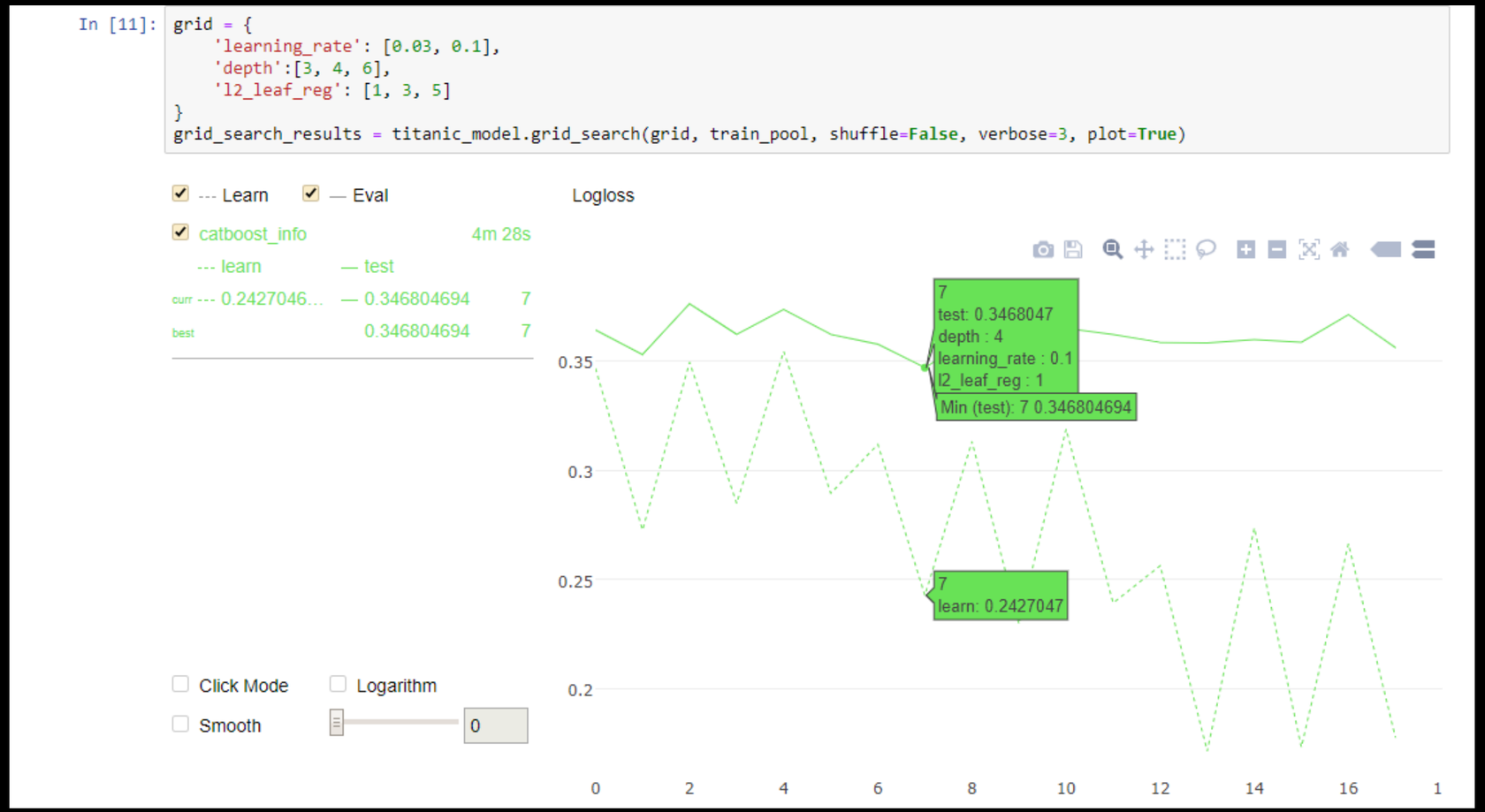
New CatBoost features

New features with effect on quality

- Automatic leaning rate selection for most loss functions
- Different tree growing strategies both on CPU and GPU
- Separate quantization for “golden” features
- Improved MVS sampling
- New ranking objective StochasticRank
- Implemented Stochastic Gradient Langevin Boosting
- Exact calculation for leaf values in some modes

Parameter tuning out of the box

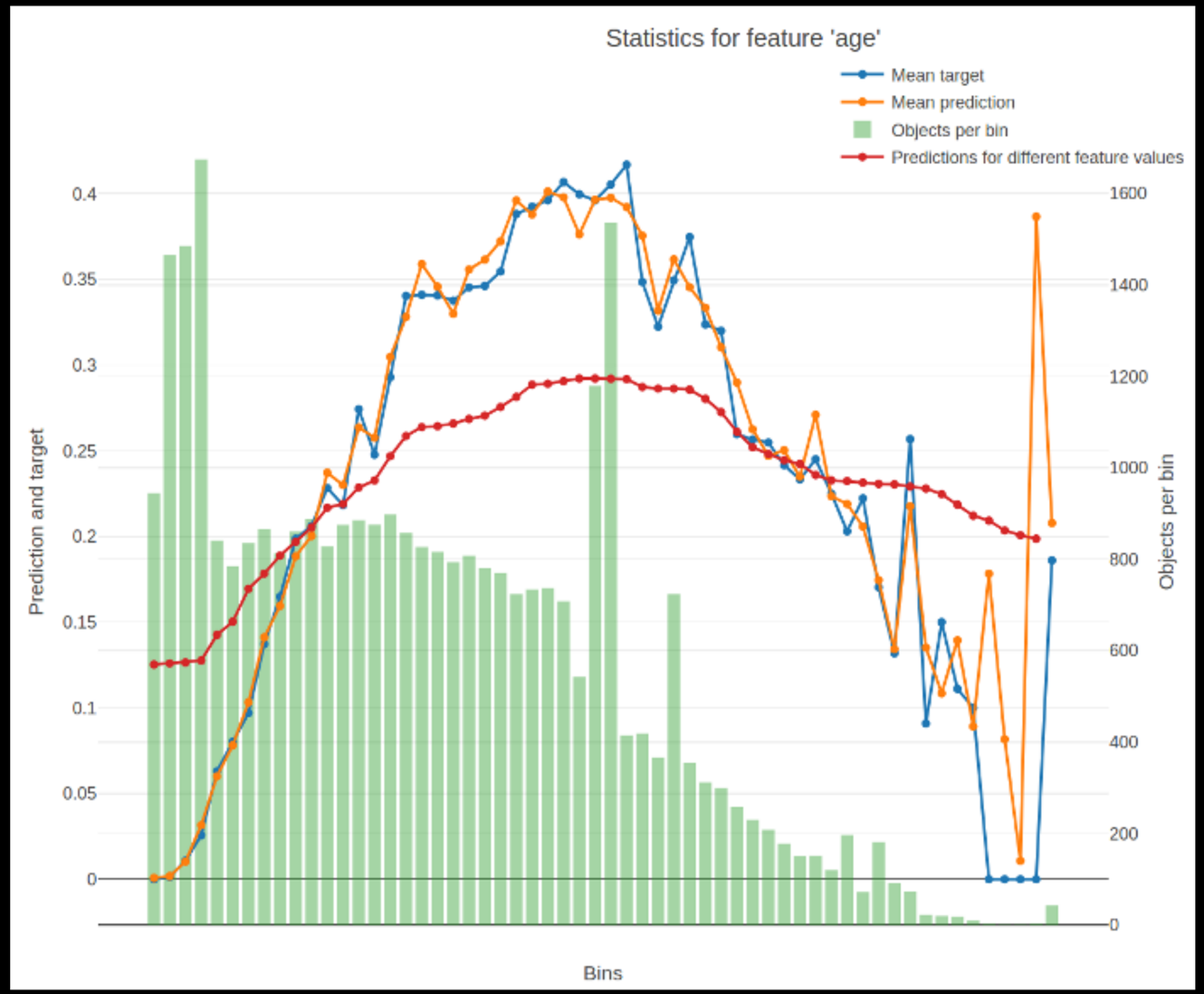
```
In [11]: grid = {
             'learning_rate': [0.03, 0.1],
             'depth':[3, 4, 6],
             'l2_leaf_reg': [1, 3, 5]
         }
         grid_search_results = titanic_model.grid_search(grid, train_pool, shuffle=False, verbose=3, plot=True)
```



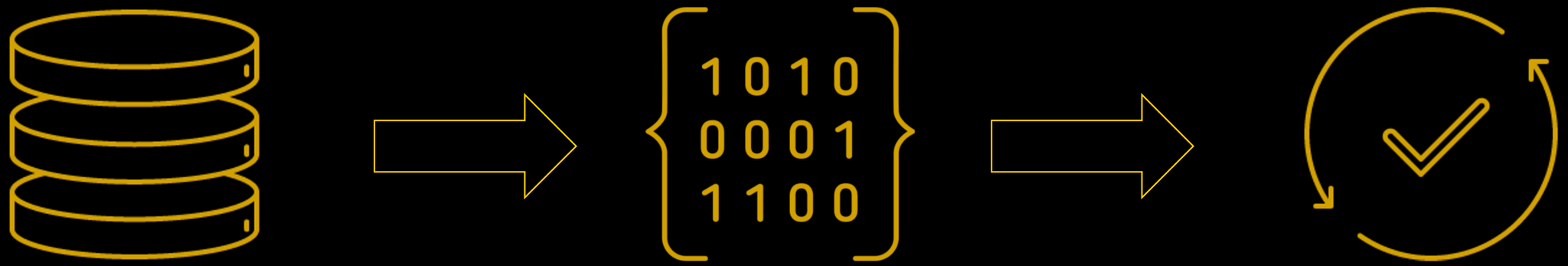
For more information about parameter tuning, visit:
https://catboost.ai/docs/concepts/parameter-tuning.html

New Model Analysis tools

- Per feature model analysis charts
- New types of feature importance
- Tree visualization
- Ranking analysis



Training on Large Data



Load raw

1TB
(tsv data)

Quantize

~160GB
quantized
representation

Train

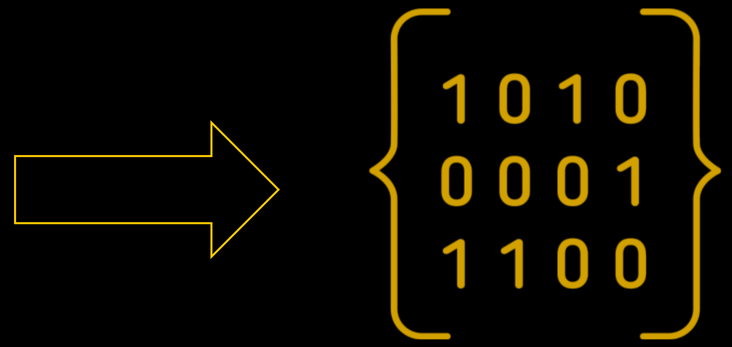
~160GB
in RAM or even less on GPU

Quantization before training

We've introduced special method `catboost.utils.quantize`
This method quantizes pool in chunks to minimize RAM usage

Load raw

1TB
(tsv data)



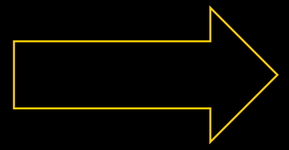
Quantize

Save quantized

~160GB
quantized
representation

Quantization before training

Load quantized



Train

~160GB
quantized
representation

~160GB
in RAM or even less on GPU

Pool quantization code snippets

```
from catboost import Pool, CatBoostClassifier

# save quantized dataset
train_dataset = Pool(train_data, train_labels)
train_dataset.quantize() # set quantization params here
train_dataset.save('quantized_train.bin')

# we can pass quantized pool path directly to fit
model = CatBoostClassifier()
model.fit('quantized://quantized_train.bin')

# load Pool object from quantized file and use multiple times
quantized_pool = Pool('quantized://quantized_train.bin')
for i in range(N):
    model = CatBoostClassifier()
    model.fit(quantized_pool)
    ...
```

Pool quantization code snippets

```
from catboost.utils import quantize

pool = quantize(
    data='really_big_train.csv’,
    delimiter=',’,
    column_description='train.cd',
)
pool.save('big_quantized_train.bin')
```

Speedups

- Huge speedups of preprocessing
- Sparse data support
- Up to 20x speedups of different modes
- Huge speedups for small datasets

Questions?

Kirillov Stanislav

Head of CatBoost team

- catboost.ai
- github.com/catboost
- twitter.com/CatBoostML
- t.me/catboost_en, t.me/catboost_ru
- ods.ai => slack => tool_catboost chanel