Yandex

Yandex
CatBoost

CatBoost: How to train better quality gradient boosting models with object embeddings and text data on GPU

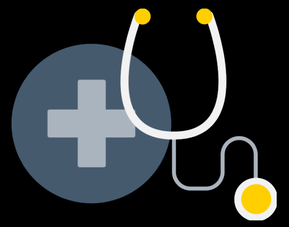
Kirillov Stanislav,
Head of CatBoost team @ Yandex

Plan

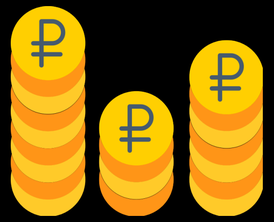
- Gradient boosting and CatBoost overview
- CatBoost: use cases
- CUDA training performance
- Distributed training: multi GPU and multi host
- Supported feature types: categorical, text and embeddings
- Big data support

Applied ML (supervised learning)

Applications



Medicine

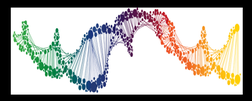


Finance

Sales prediction

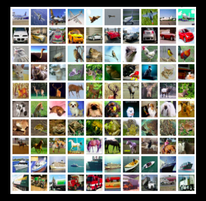
+

Data at hand



DNA

Text



Images

Music

+

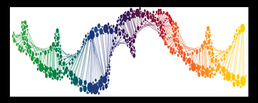
Tool

- Linear models
- Neural nets
- Decision trees
- GBDT
- etc

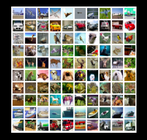
Data at hand

Unstructured data

Music



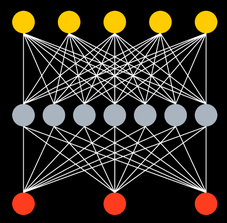
DNA



Images

Text

End2End with Deep NN

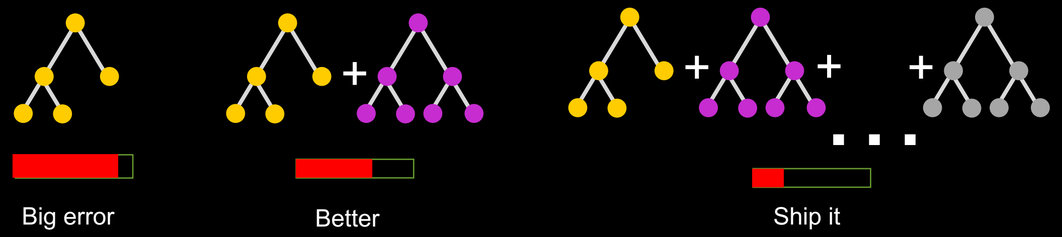


Tabular (or structured) data

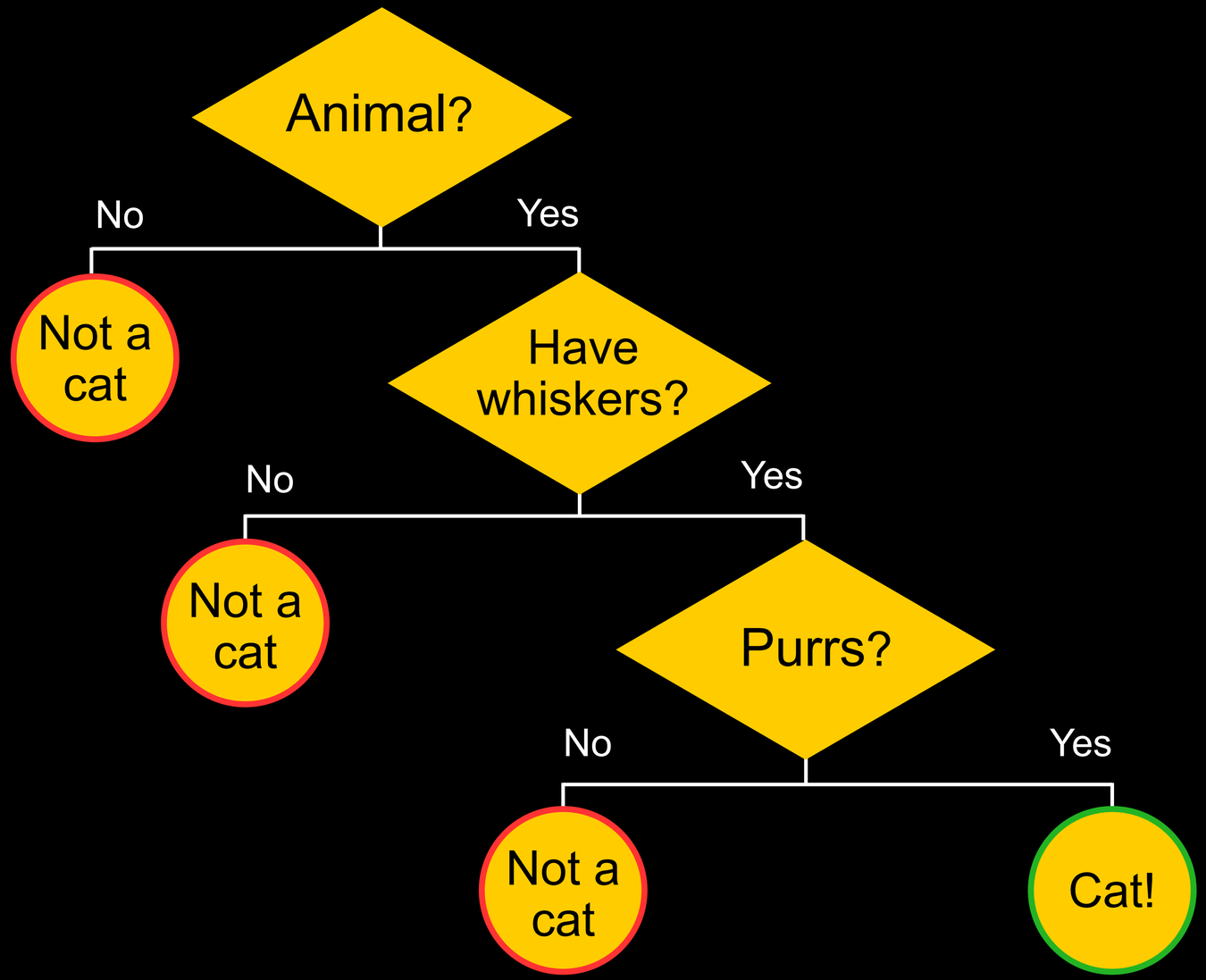
Well engineered features

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

GBDT

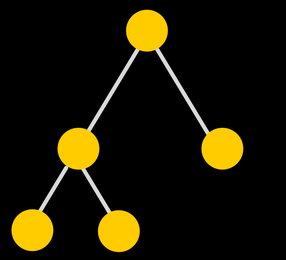


Tabular data? Decision trees!



Gradient boosted decision trees

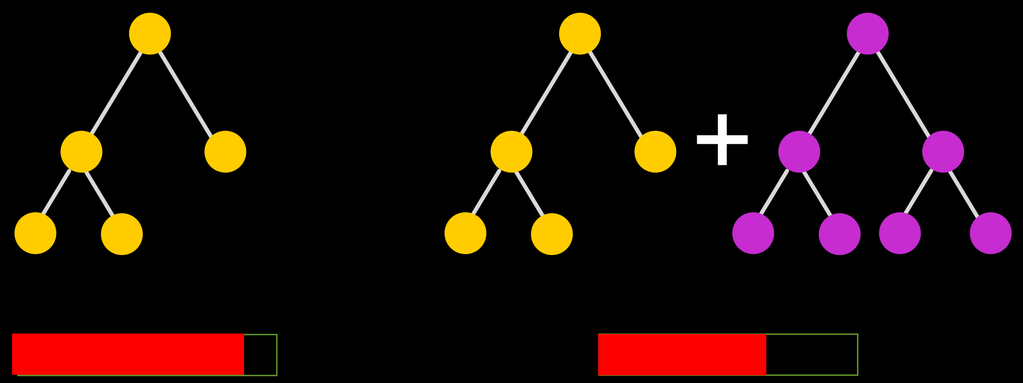
- State-of-the-art quality on tabular data
- Easy to use, no sophisticated parameter tuning
- Works well with small data and scales for big data problems



Big error

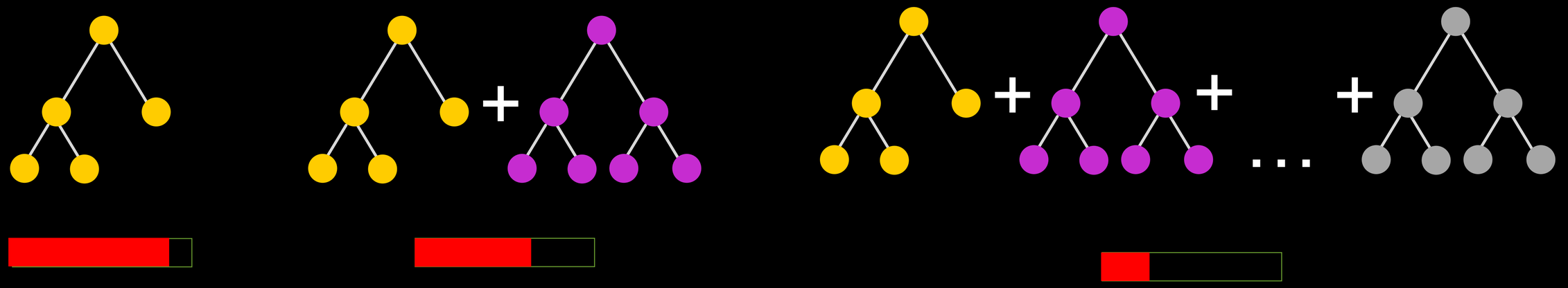
Gradient boosted decision trees

- 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



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 advantages

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

Yandex.Search

Task?

- Search document order prediction

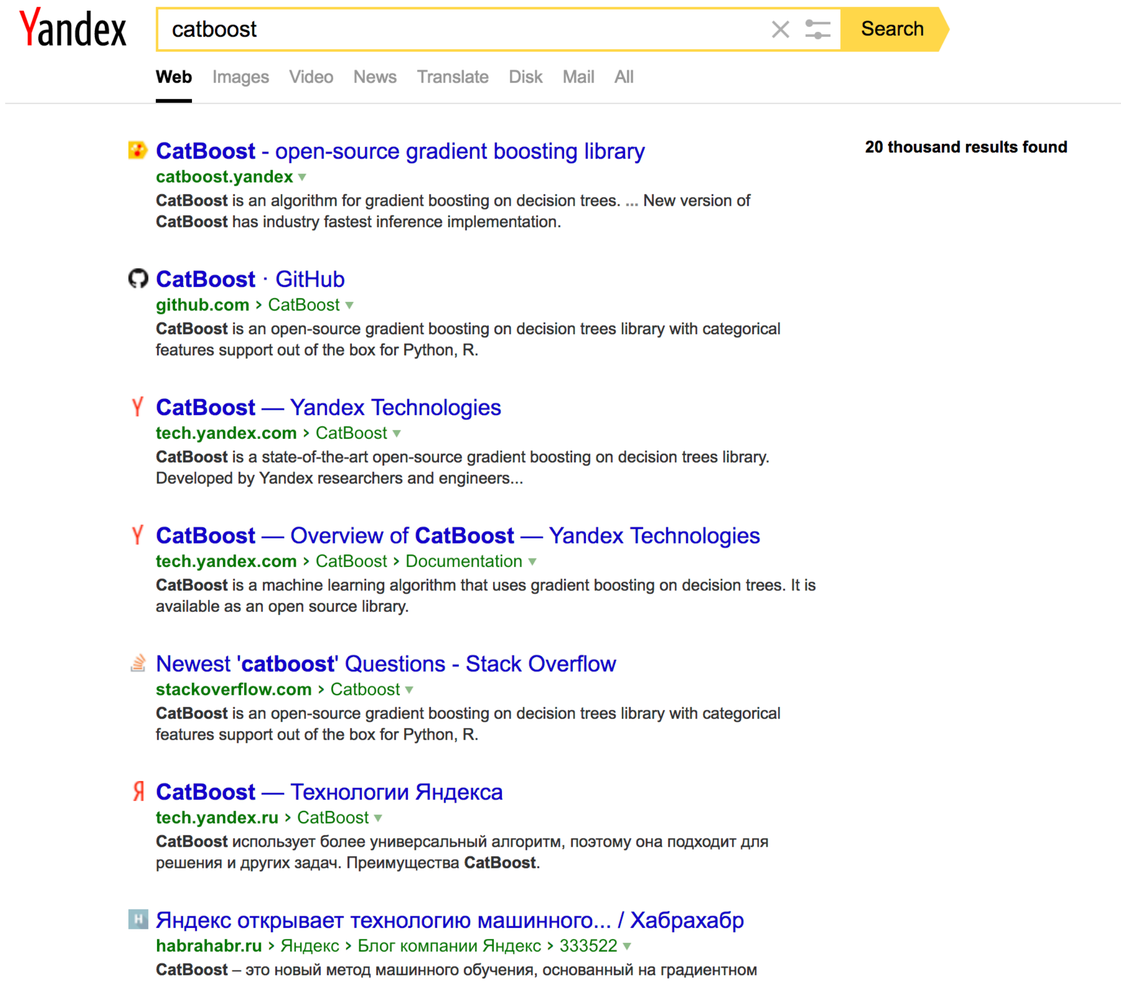
Task type: ranking

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

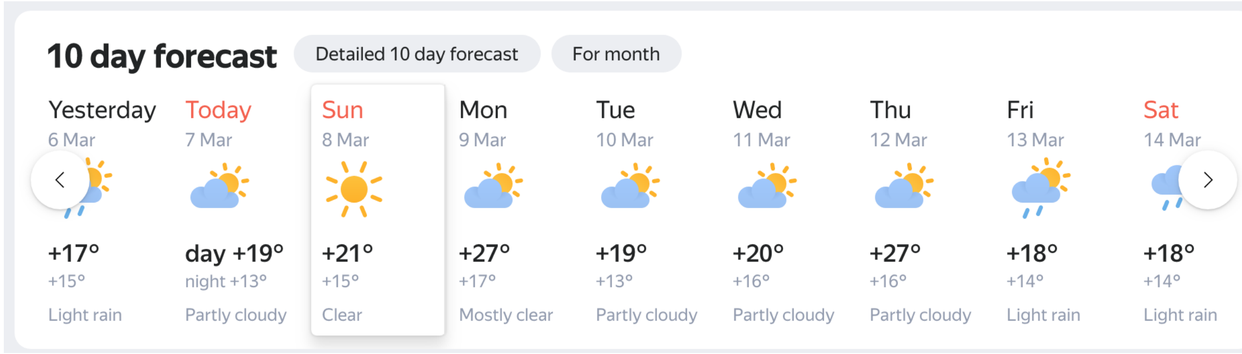
Task type: multiclassification and regression

Dataset features

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

CatBoost features used:

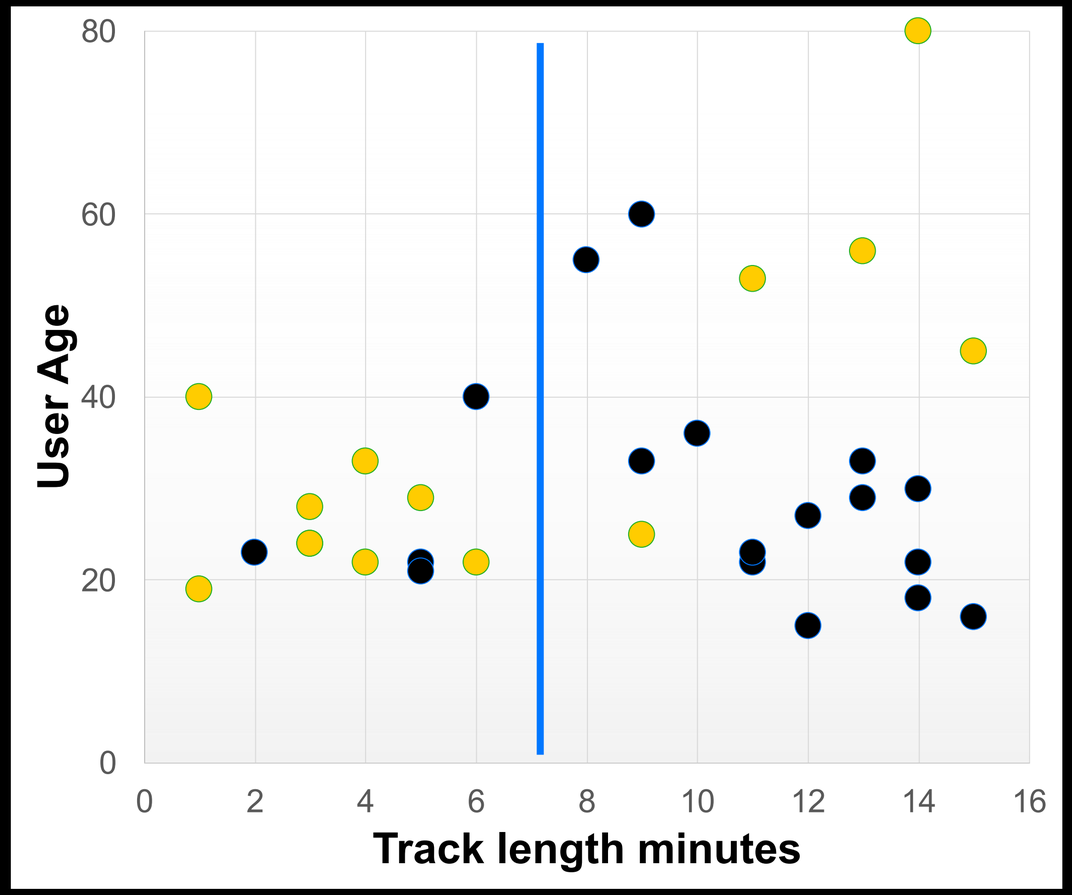
- Multiclassification target and RMSE (for temperature)
- GPU training
- Feature importance analysis
- Training process visualization

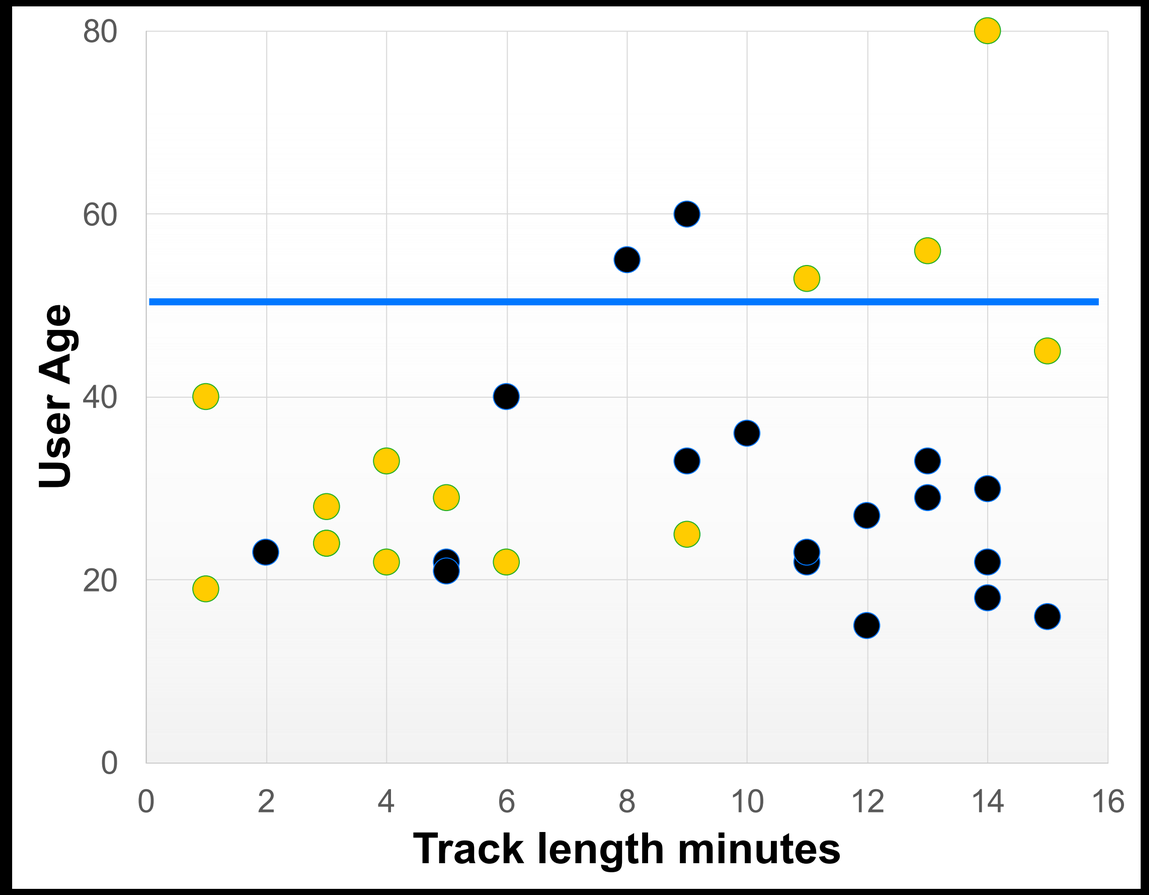


CatBoost in the Wild

- 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

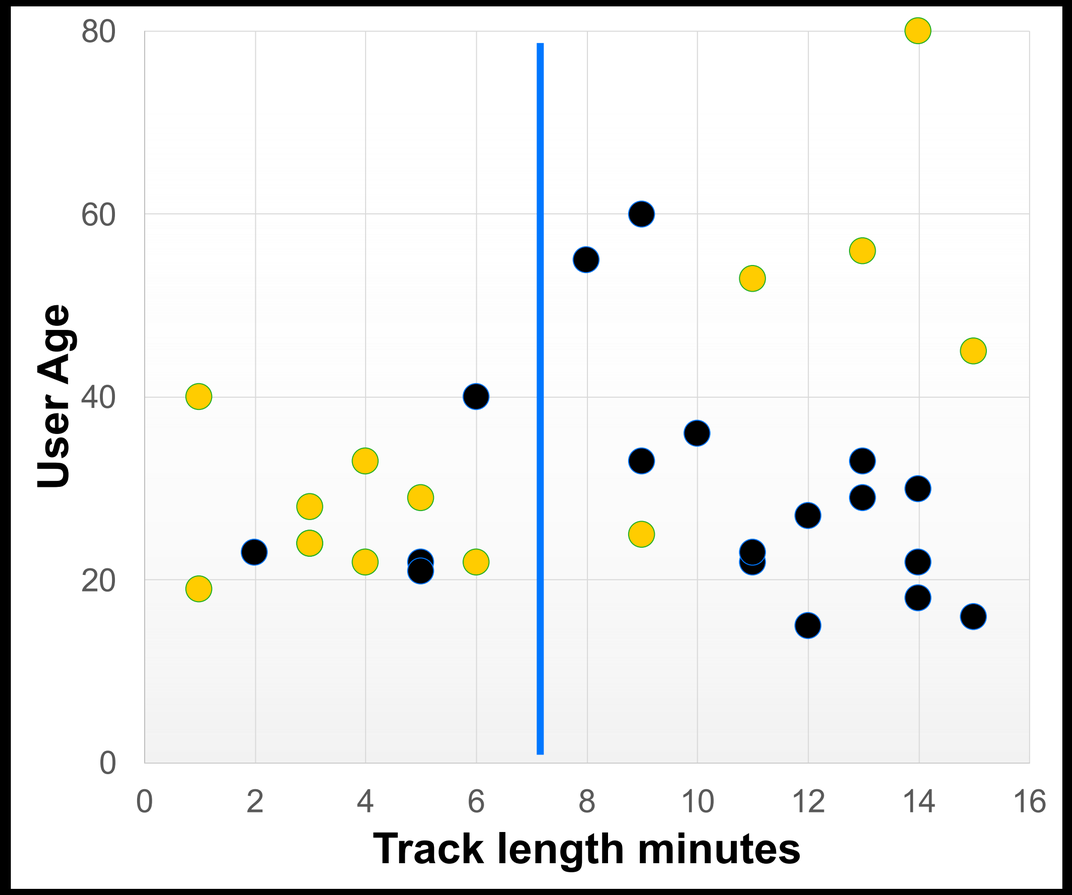
Decision tree: classification

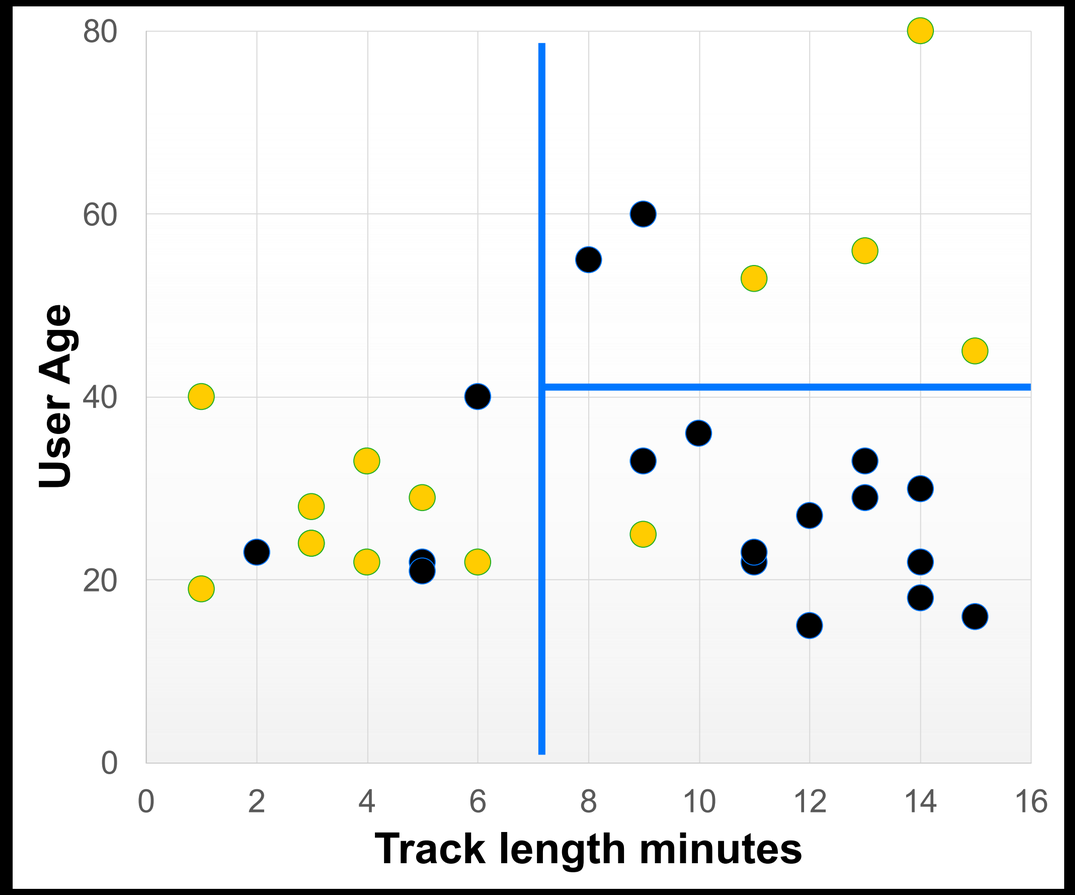




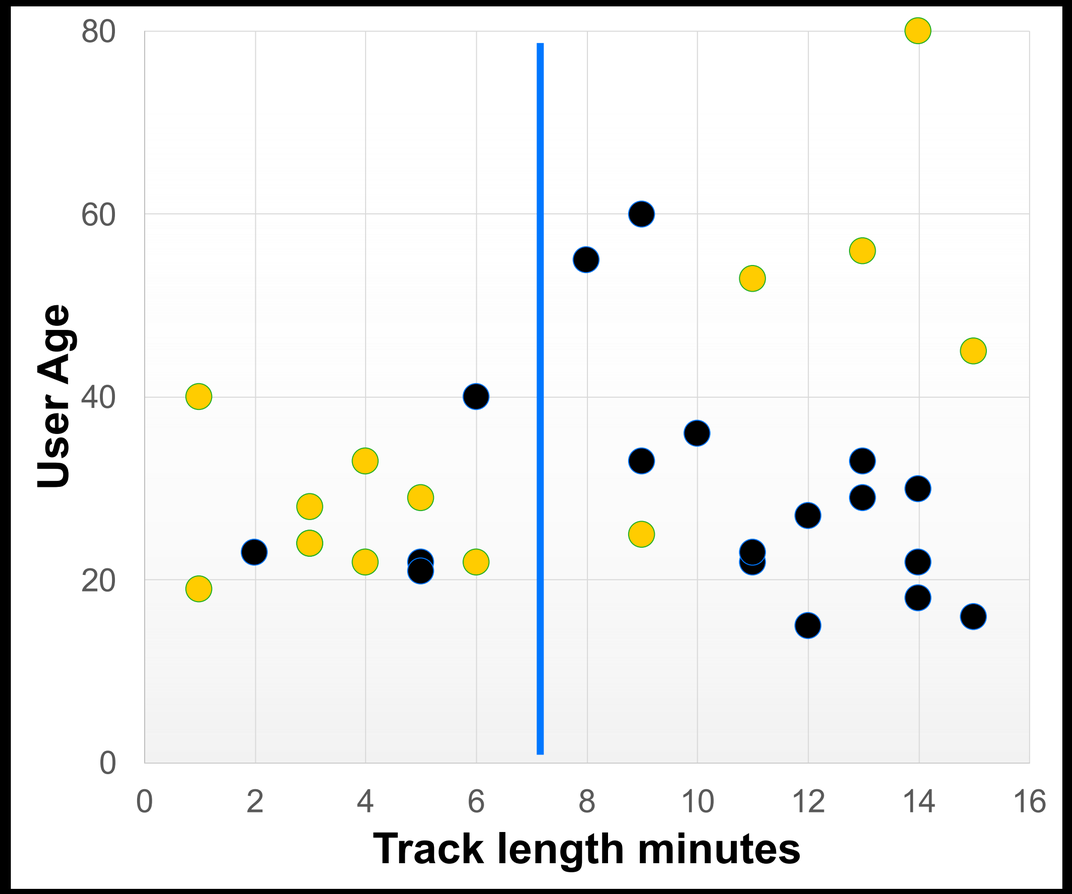
Best split: Track length minutes > 7

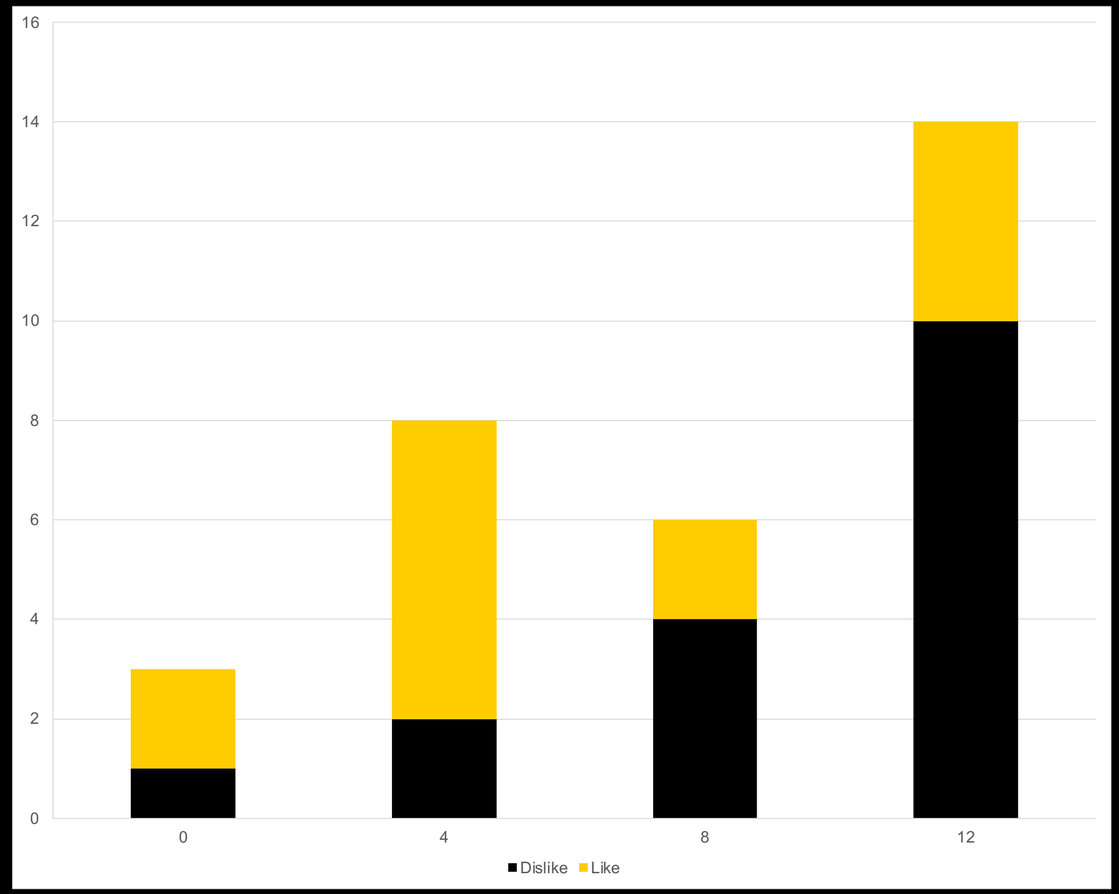
Decision tree: classification





Decision tree: histograms



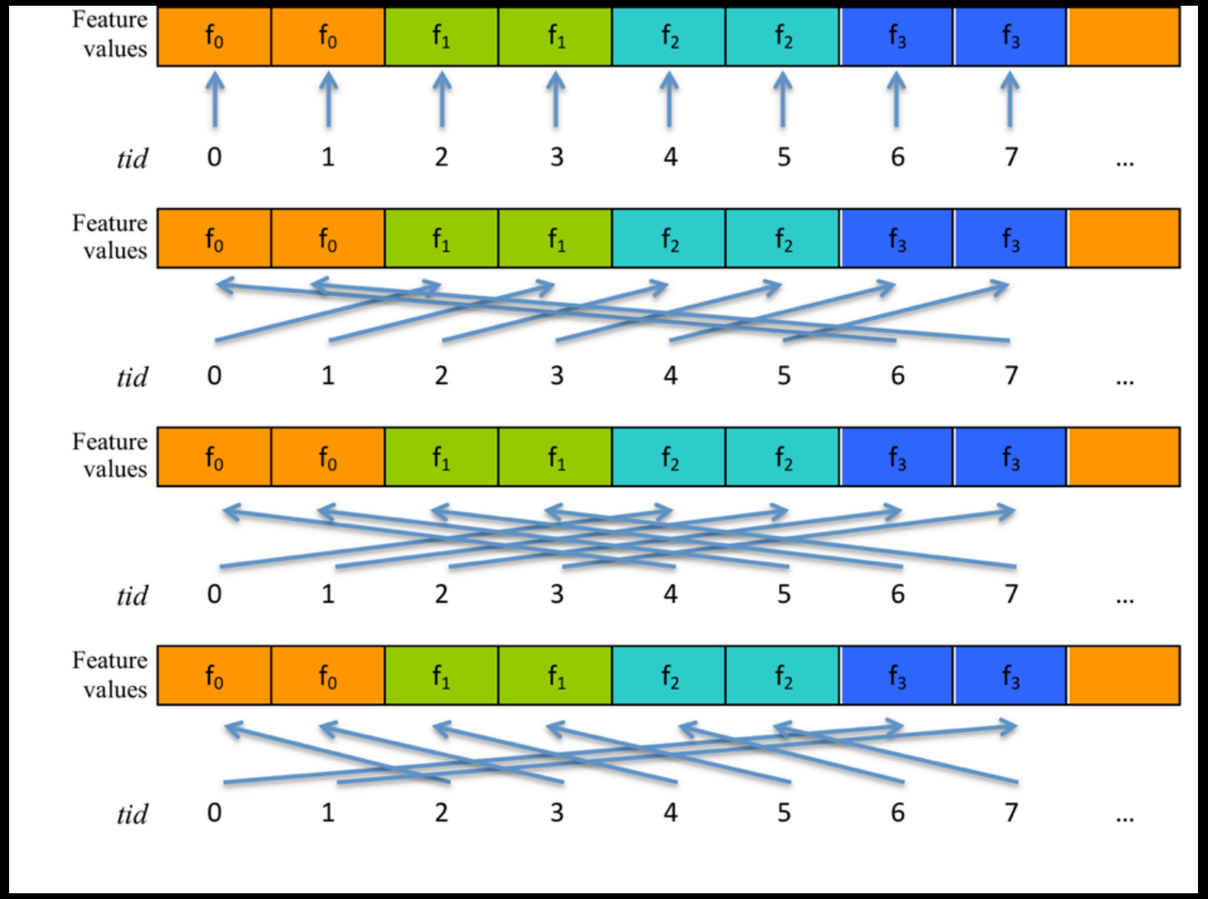


Histograms on GPU

- Aggregation in fast shared memory
- Layout to avoid bank conflicts
- No atomics: no need to hardware support
- Trade-off: Occupancy vs Atomics

384 threads

48KB shared memory

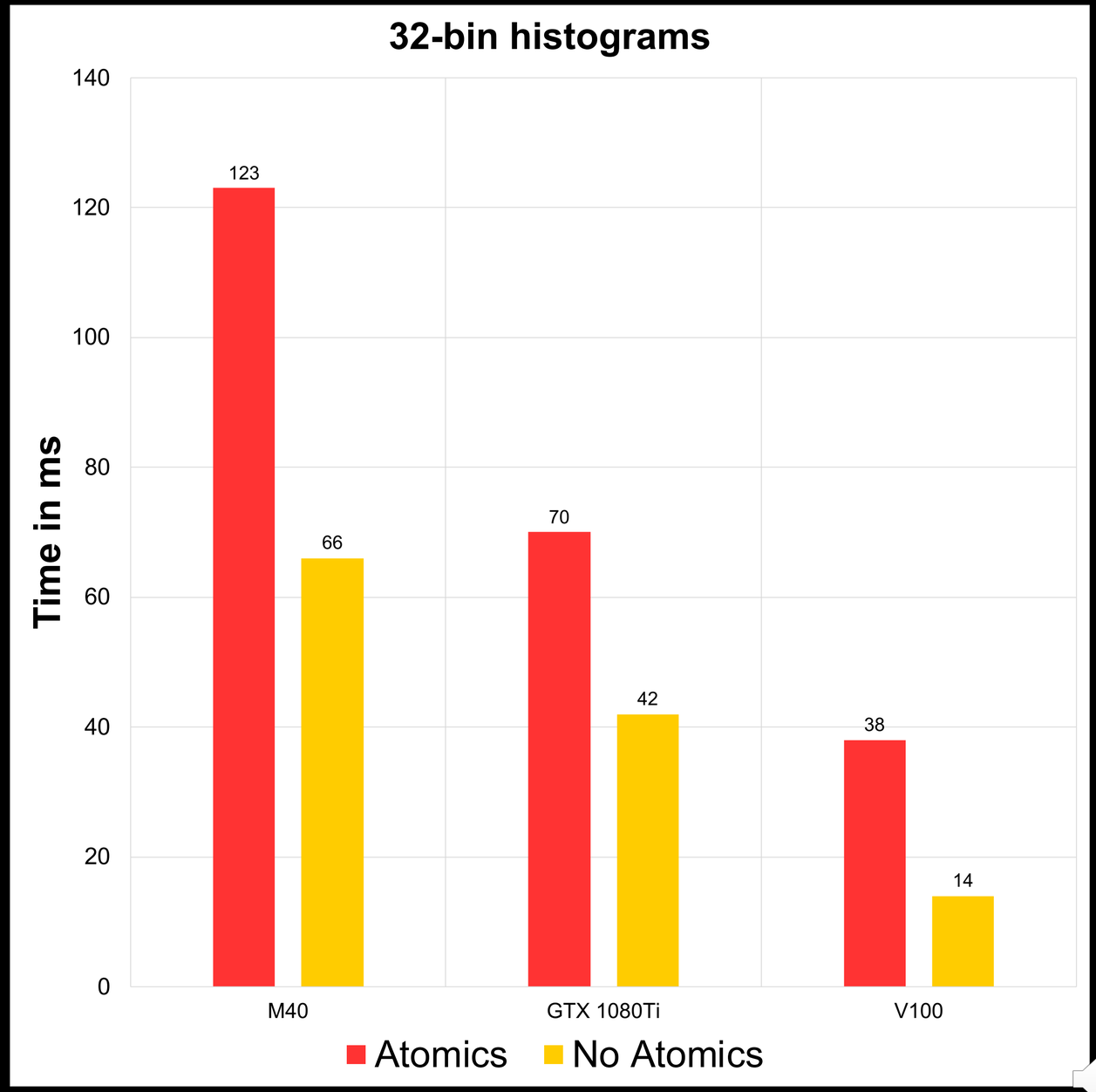


Avoid atomics

Experiment: increase occupancy in exchange for atomic operations

M40, 1080Ti, V100: 38% => 75%

Result (Maxwell and later):
x1.5-x3 performance for first level histograms
x1.25-x2 faster training time

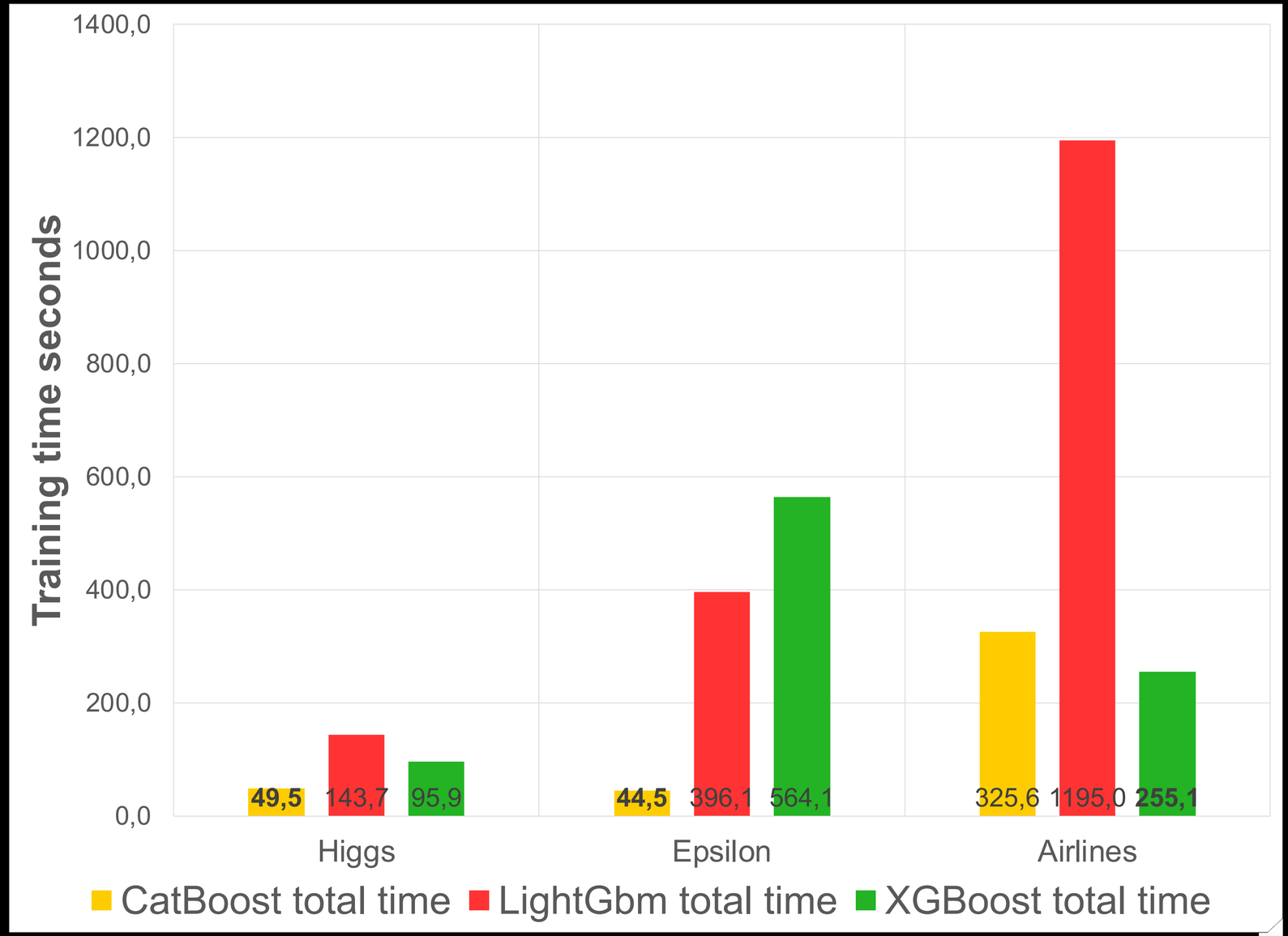


V100 Training performance: 1000 trees

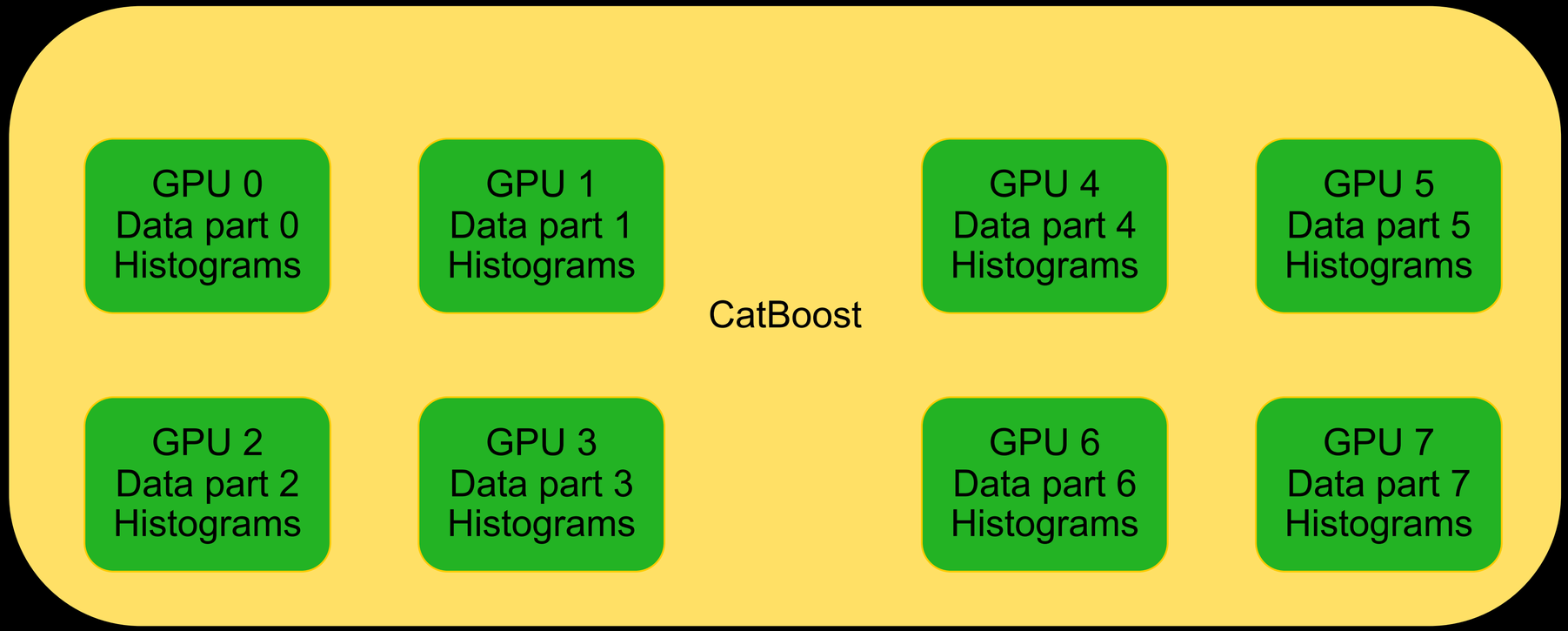
GPU training time on single NVIDIA Tesla V100

- Higgs
10.5mln objects
28 features
- Epsilon
400k objects
2k features
- Airlines
23mln objects
13 features

CatBoost v0.24.1, Lightgbm 2.3.1, XGBoost 1.0.2



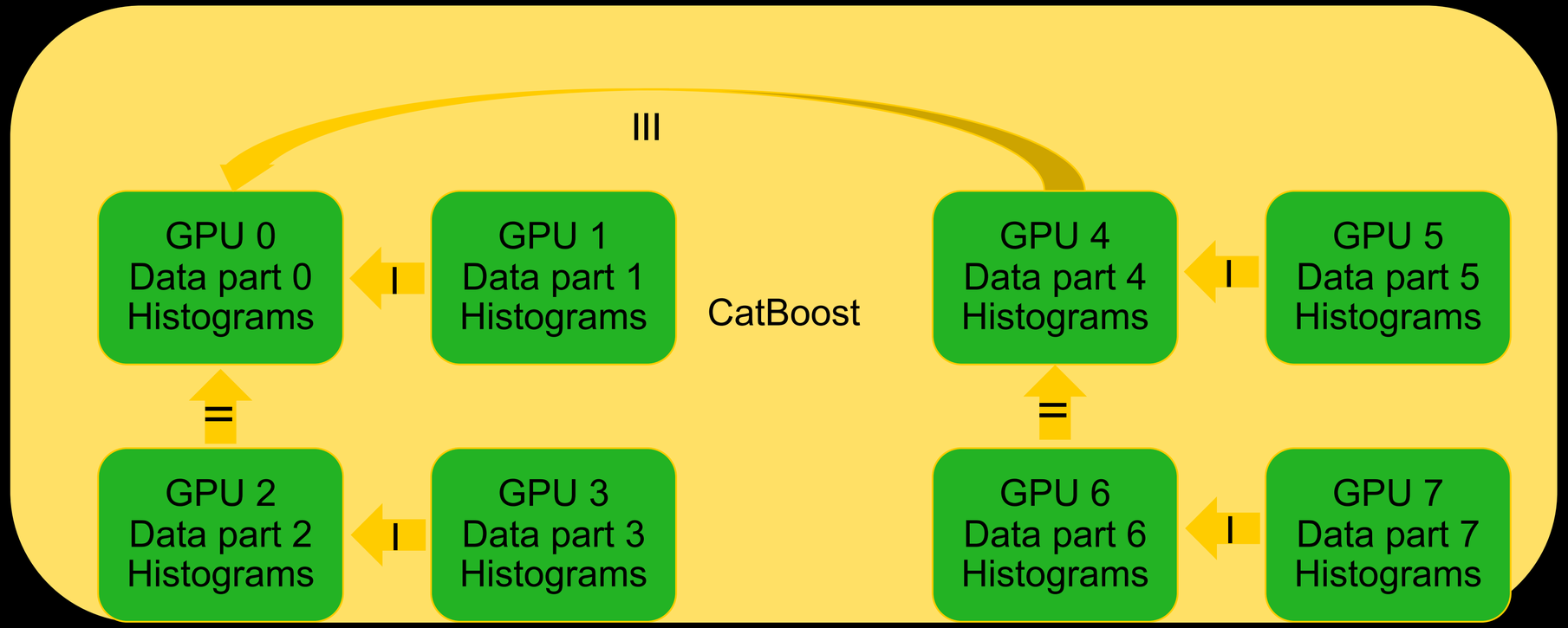
Distributed training: single-host multi-GPU



CatBoost uses 2 different data-layouts:

- Document parallel – dataset distributed between GPU by rows
- Feature parallel – dataset distributed between GPU by feature columns

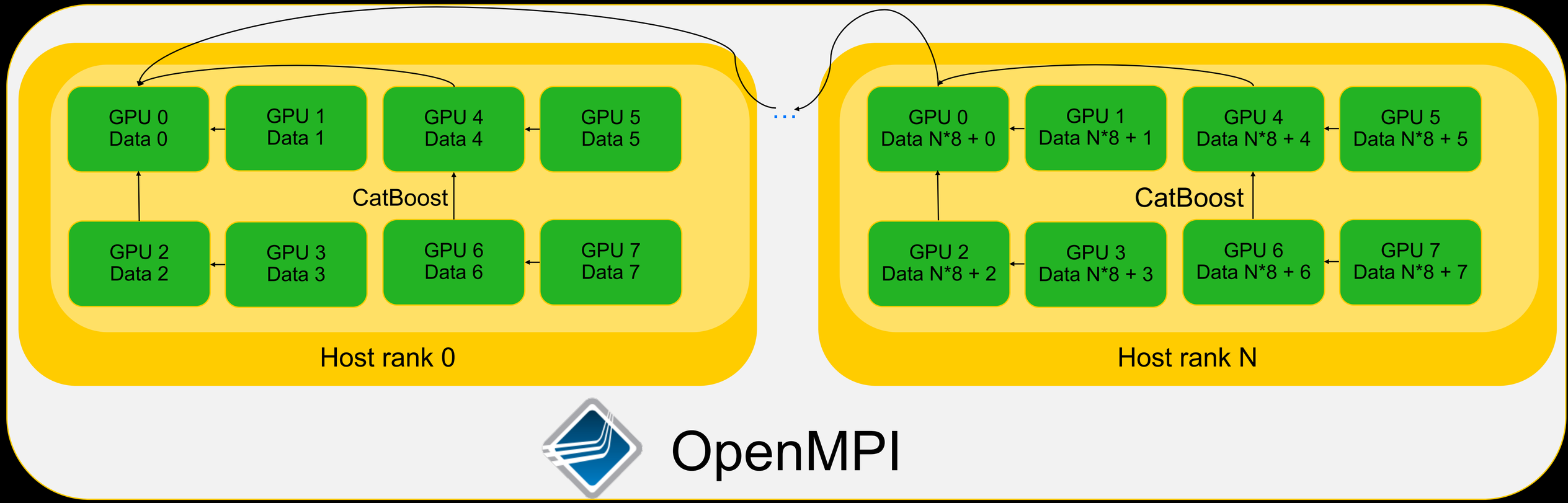
Distributed training: single host tree-reductions



Example: tree reductions for feature histograms

If two cards on current reduction step have p2p access (nvlink) reduction kernel uses peer GPU memory read.

Distributed training: multi-host multi-GPU



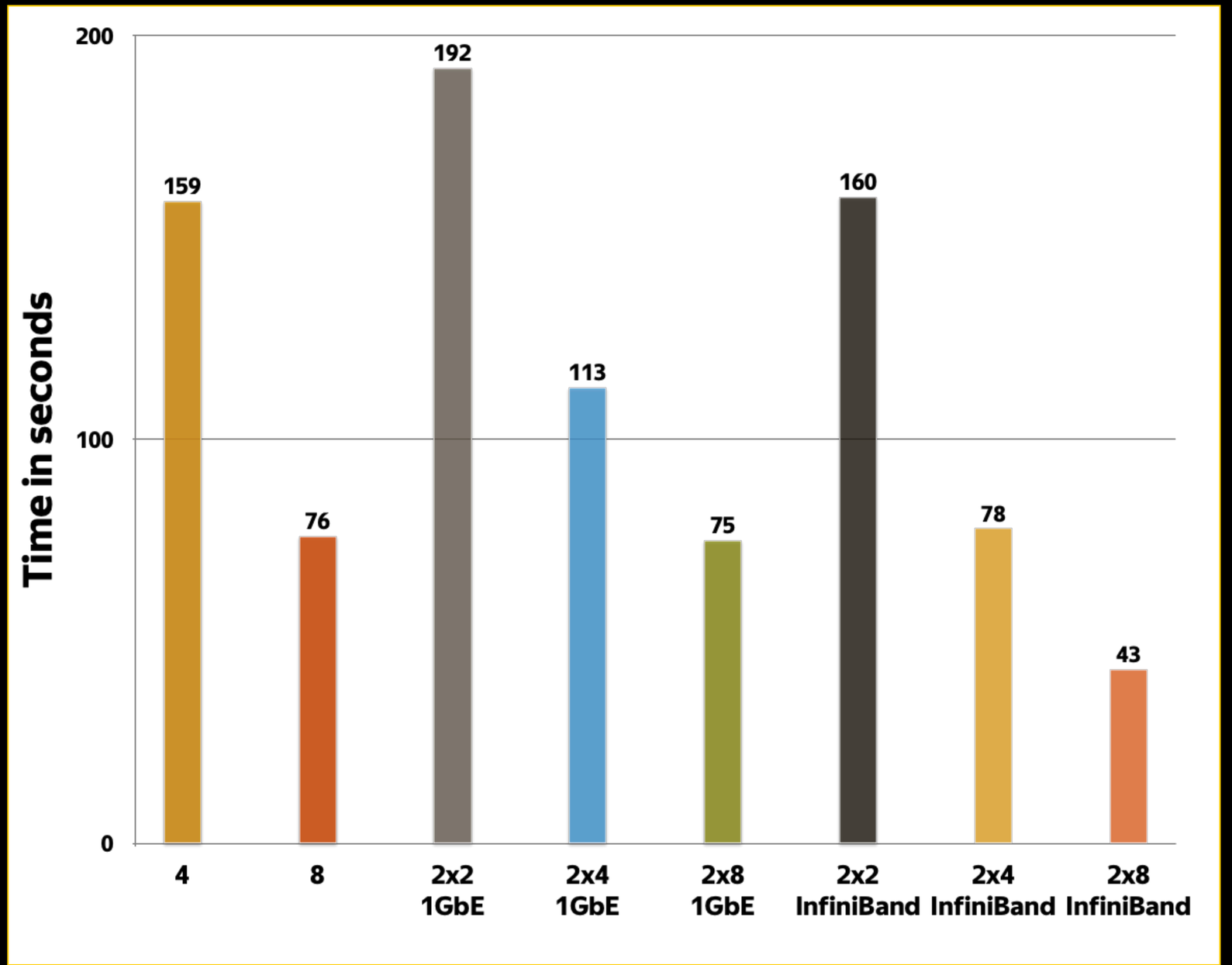
Distributed training: multi-host multi-GPU

CatBoost uses MPI for multi-host communications.
Easy to run:

```
mpirun -np $HOSTS_COUNT -bind-to none -map-by slot
/path/to/catboost fit --learn-set
quantized://path_to_quantized_train --loss-function
Logloss
```

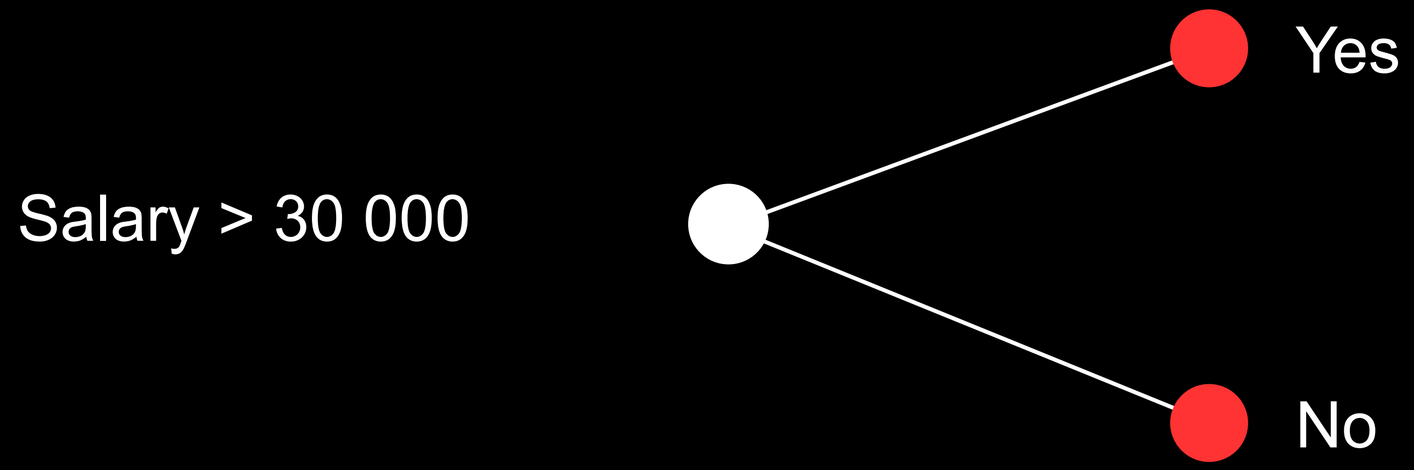
Currenlty, distributed training reads data from `Rank0` host and distributes it across all nodes evenly.

Distributed training: scaling benchmark



Text features and embeddings

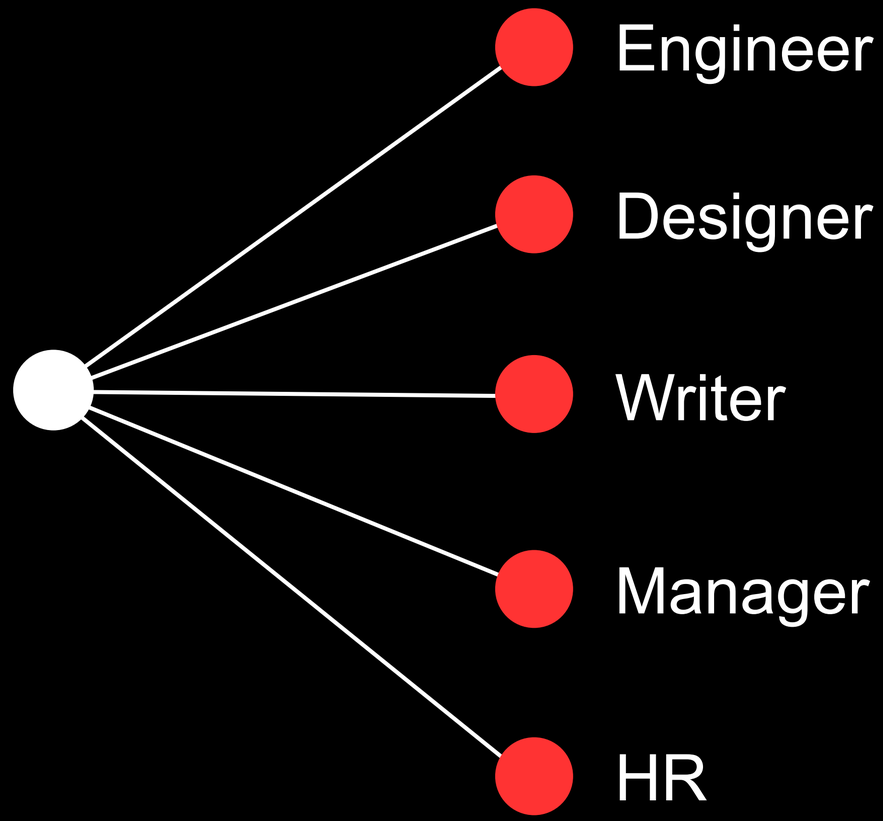
Numerical features



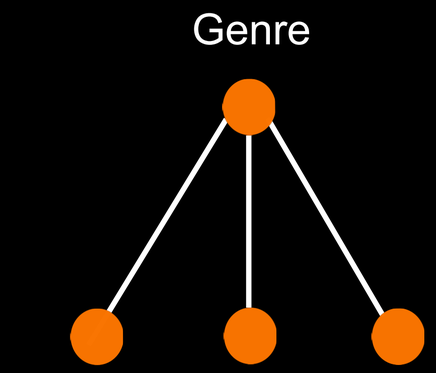
Categorical features

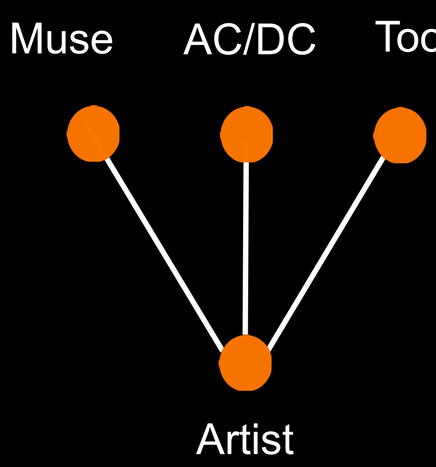
Categorical data

Occupation



Categorical features handling





One-hot encoding

Statistics based on category

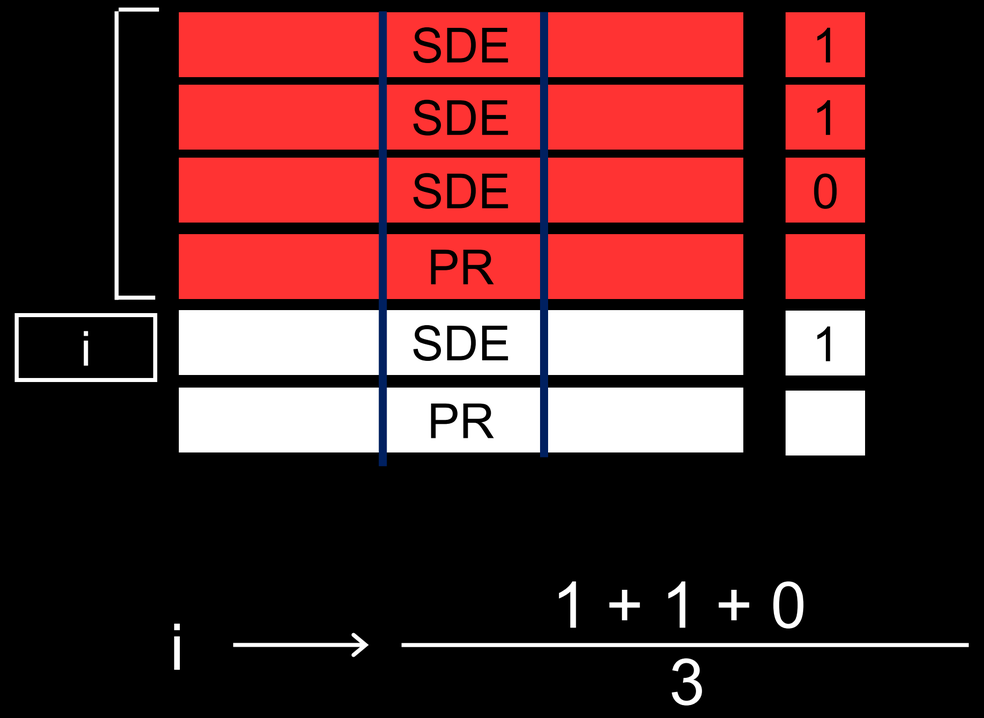
Category-based

Label based:
calculated “online”

Greedy search for
combinations

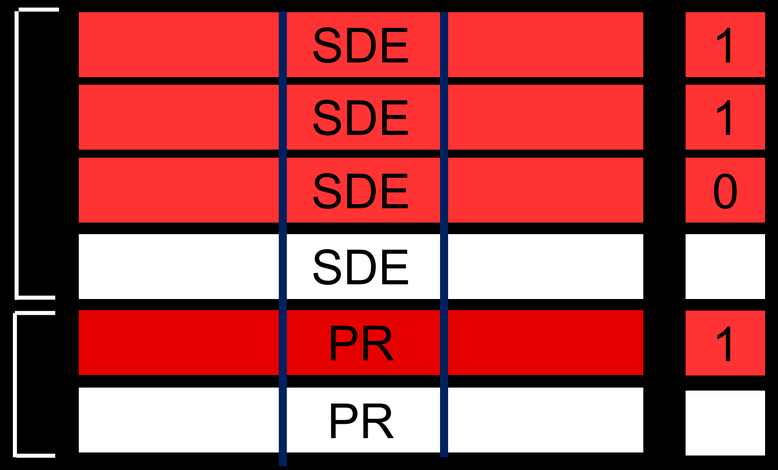
Online features for categories

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



Online features for categories: GPU implementation

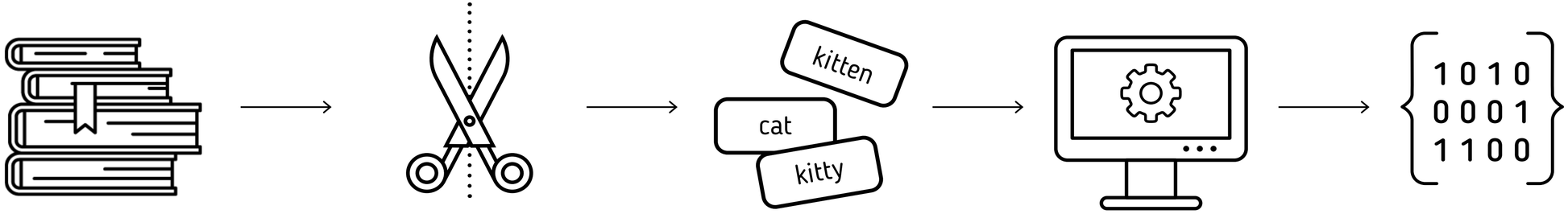
- Reindex categorical feature values from 0 to N-1
- Use CUB radix-sort for reordering data by categorical feature value
- Compute in parallel segments



Quality: categorical feature performance

	CatBoost		LightGBM		XGBoost		H2O	
	Tuned	Default	Tuned	Default	Tuned	Default	Tuned	Default
Adult	**0.26974**	0.27298 +1.21%	0.27602 +2.33%	0.28716 +6.46%	0.27542 +2.11%	0.28009 +3.84%	0.27510 +1.99%	0.27607 +2.35%
Amazon	**0.13772**	0.13811 +0.29%	0.16360 +18.80%	0.16716 +21.38%	0.16327 +18.56%	0.16536 +20.07%	0.16264 +18.10%	0.16950 +23.08%
Click prediction	**0.39090**	0.39112 +0.06%	0.39633 +1.39%	0.39749 +1.69%	0.39624 +1.37%	0.39764 +1.73%	0.39759 +1.72%	0.39785 +1.78%
KDD appetency	0.07151	**0.07138** –0.19%	0.07179 +0.40%	0.07482 +4.63%	0.07176 +0.35%	0.07466 +4.41%	0.07246 +1.33%	0.07355 +2.86%
KDD churn	**0.23129**	0.23193 +0.28%	0.23205 +0.33%	0.23565 +1.89%	0.23312 +0.80%	0.23369 +1.04%	0.23275 +0.64%	0.23287 +0.69%
KDD internet	**0.20875**	0.22021 +5.49%	0.22315 +6.90%	0.23627 +13.19%	0.22532 +7.94%	0.23468 +12.43%	0.22209 +6.40%	0.24023 +15.09%
KDD upselling	**0.16613**	0.16674 +0.37%	0.16682 +0.42%	0.17107 +2.98%	0.16632 +0.12%	0.16873 +1.57%	0.16824 +1.28%	0.16981 +2.22%
KDD 98	**0.19467**	0.19479 +0.07%	0.19576 +0.56%	0.19837 +1.91%	0.19568 +0.52%	0.19795 +1.69%	0.19539 +0.37%	0.19607 +0.72%
Kick prediction	**0.28479**	0.28491 +0.05%	0.29566 +3.82%	0.29877 +4.91%	0.29465 +3.47%	0.29816 +4.70%	0.29481 +3.52%	0.29635 +4.06%

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

…

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

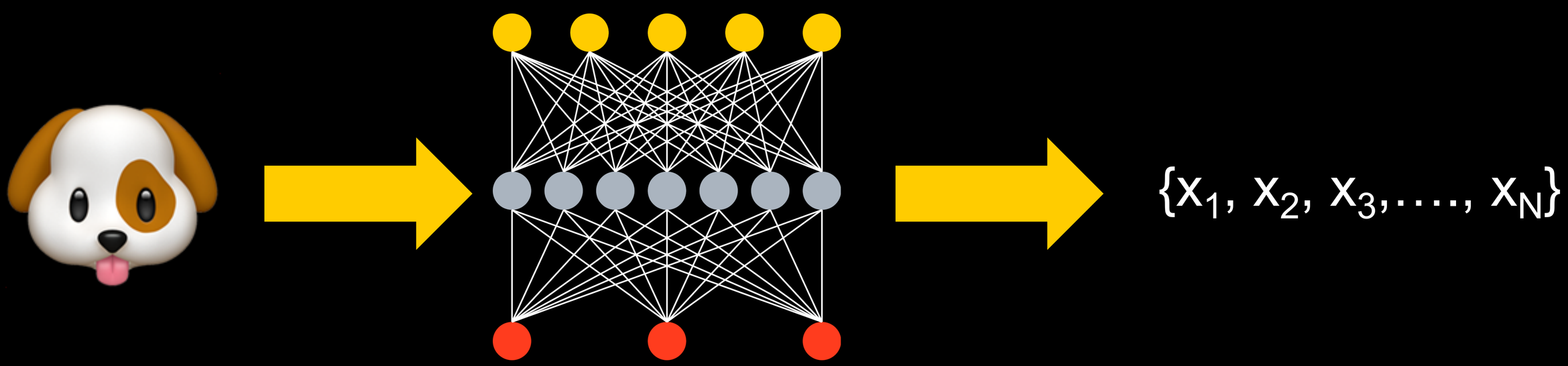
Naïve Bayes

- 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”

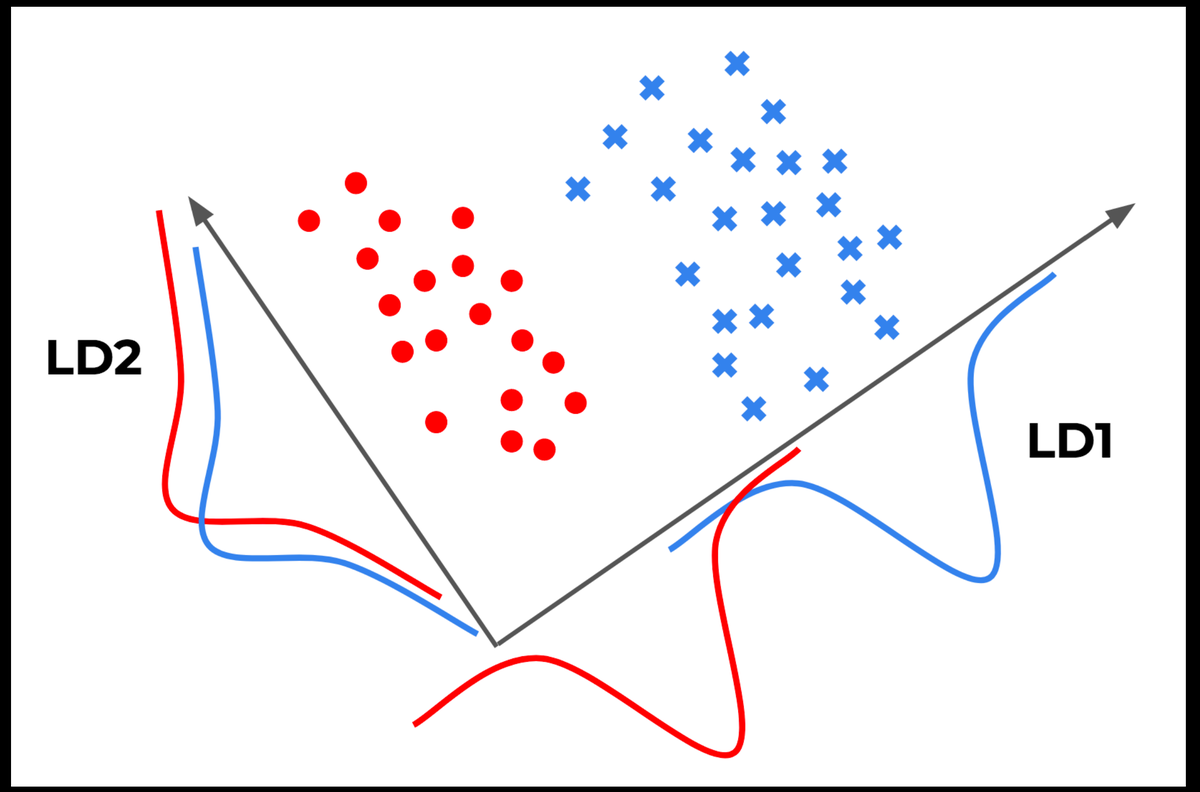
Object embeddings



For example well known text embeddings: Word2Vec, DSSM, BERT, GPT and another language models.

Linear discriminant analysis

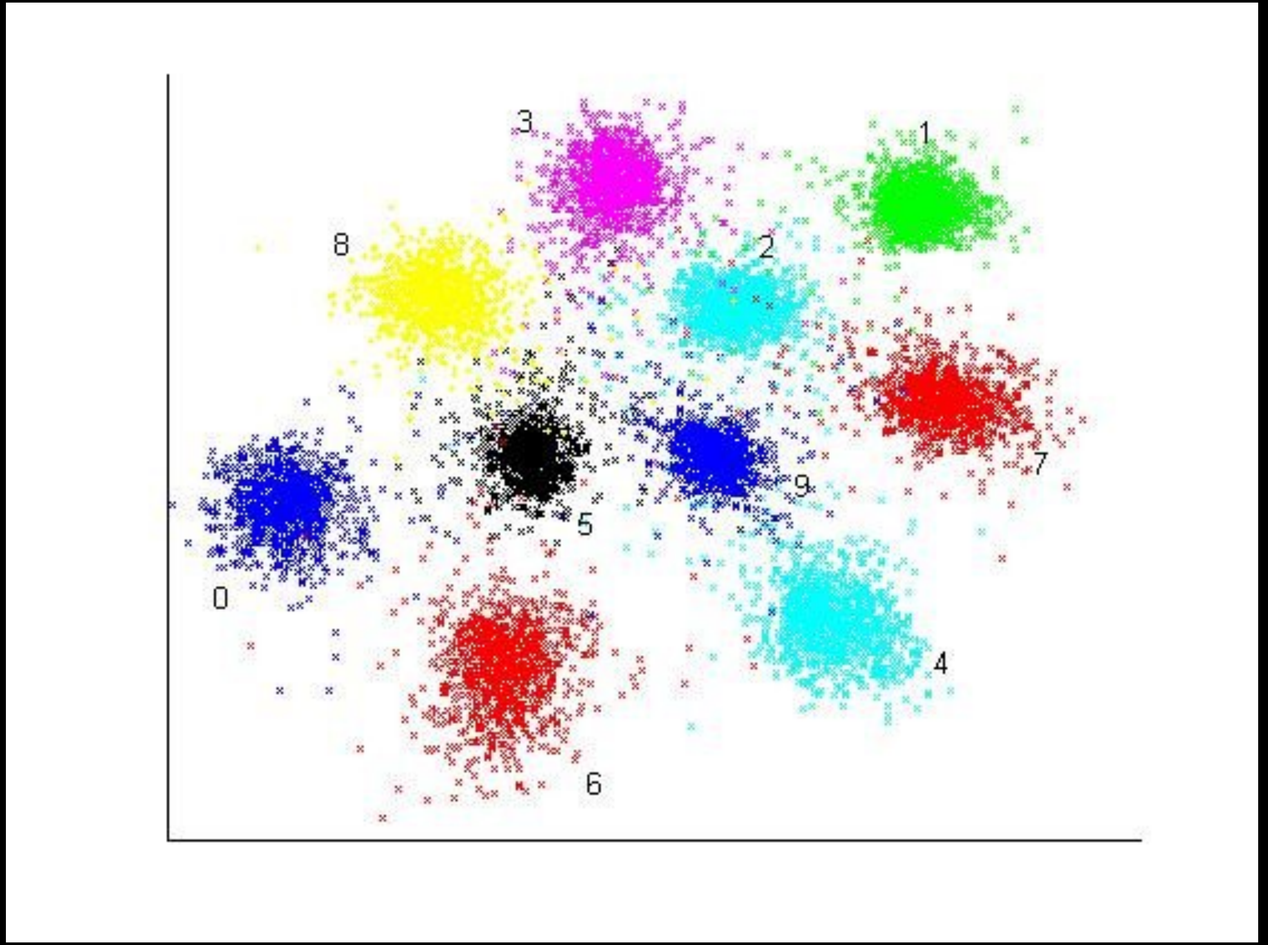
$$\log P(y = k|x) = \omega_k^t x + \omega_{k0} + Cst.$$



https://raw.githubusercontent.com/eigenfoo/eigenfoo.xyz/master/assets/images/lda-pic.png

kNN

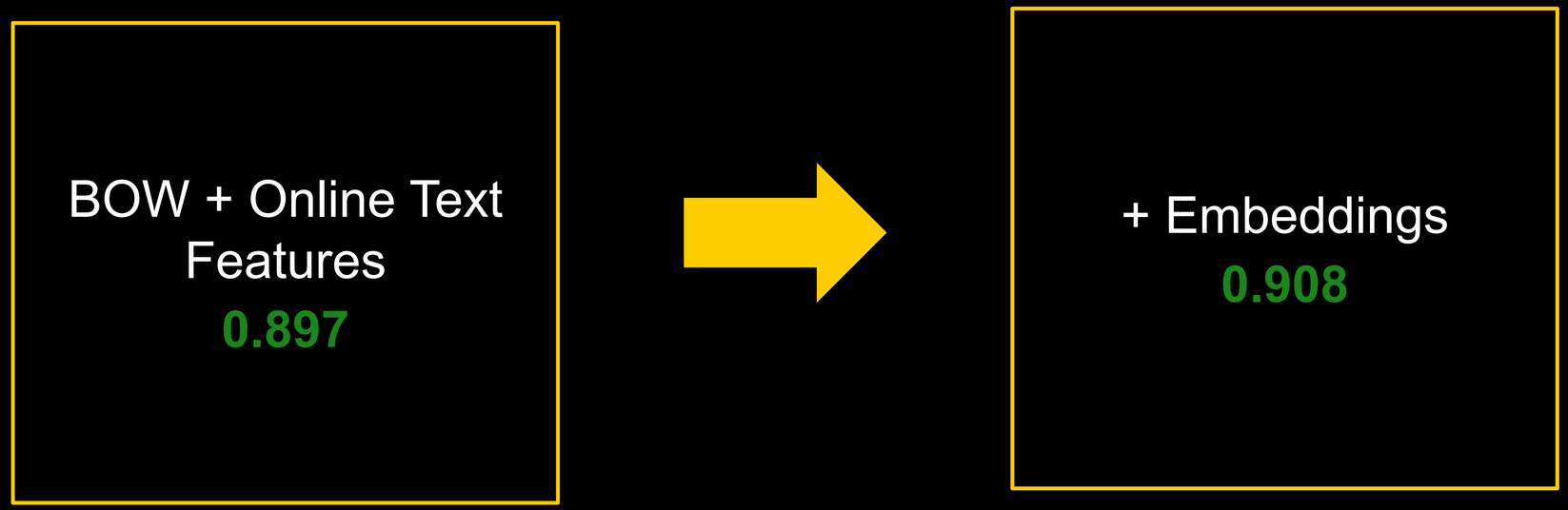
Online feature estimation.
For each object in permutation:
1. Query k-nearest neighbours
2. Get classes for found neighbours
3. Treat class-frequencies as features for CatBoost
4. Update HNSW index



A Deep Non-Linear Feature Mapping for Large-Margin kNN Classification - Scientific Figure on ResearchGate.
Available from: https://www.researchgate.net/figure/Two-dimensional-embedding-of-10-000-MNIST-test-data-using-the-Deep-Neural-Network-kNN_fig9_22076648

Profit from text features and embeddings

Accuracy on IMDB dataset with default parameters



Mixed dataset 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

Text embeddings

- BERT, GPT, Word2Vec or any other embeddings

review

One very long, dark ride.

genres

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

Profit from text features and embeddings

Accuracy on Rotten Tomatoes multiclass accuracy

How to try embeddings in CatBoost?

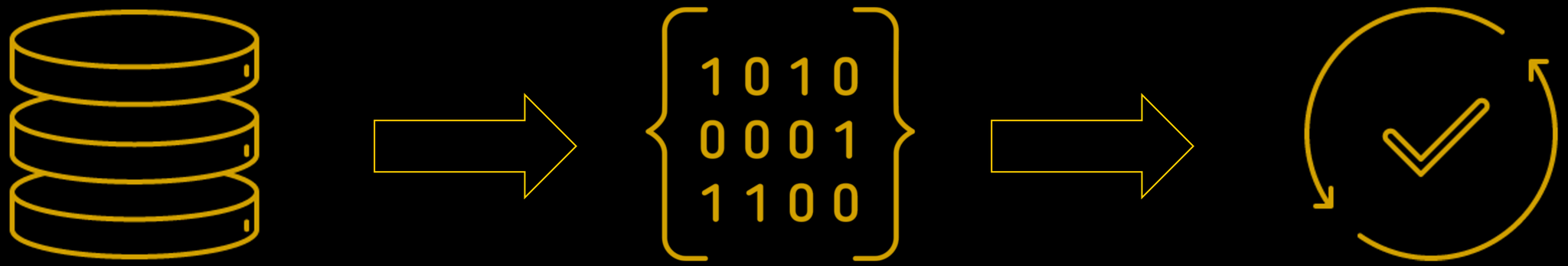
```
train_pool_embeddings = Pool(
    X_train_with_embeds, y_train,
    cat_features=cat_features,
    text_features=['synopsis', 'genre', 'director', 'writer', 'review’],
    embedding_features=['synopsis_embeds', 'review_embeds']
)

test_pool_with_embeddings = Pool(
    X_test_with_embeds, y_test,
    cat_features=cat_features,
    text_features=['synopsis', 'genre', 'director', 'writer', 'review’],
    embedding_features=['synopsis_embeds', 'review_embeds']
)
model = catboost.CatBoostClassifier(task_type='GPU')
model.fit(train_pool_embeddings, test_pool_with_embeddings)
```

CatBoost GPU 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
- MVS sampling: GOSS successor
- Various ranking objectives: YetiRank, StochasticRank, QueryCrossEntropy

Training on Large Data



Load raw

1TB
(tsv data)

Quantize

~160GB
quantized
representation

Train

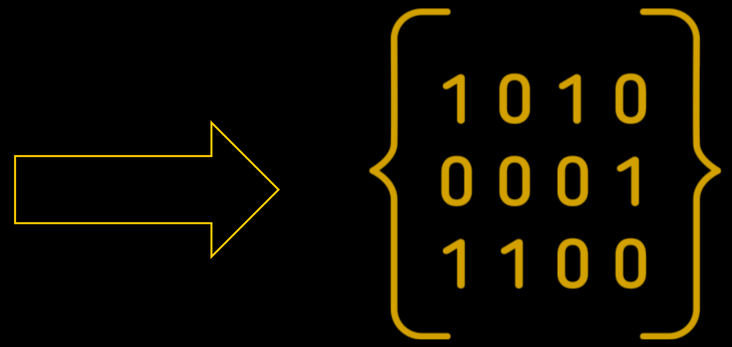
~160GB
in RAM or even less on GPU

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)



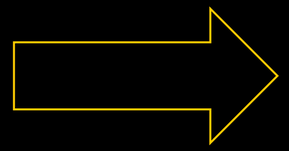
Quantize

Save quantized

~160GB
quantized
representation

Quantization before training

Load quantized

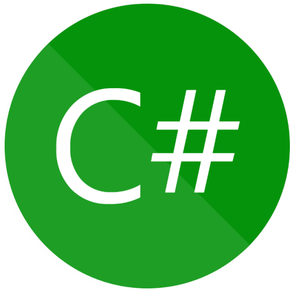


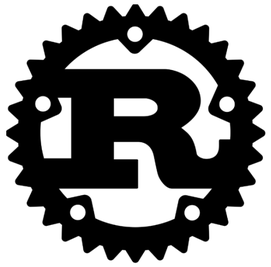
Train

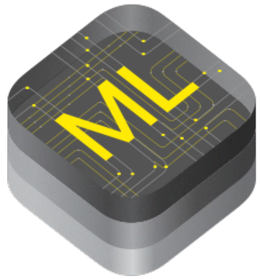
~160GB
quantized
representation

~160GB
in RAM or even less on GPU

Integration in production







Questions?

Kirillov Stanislav

Head of ML systems group & CatBoost team

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