PAKDD 2019: the 4th AutoML Challenge
AutoML for Lifelong Machine Learning

Wei-Wei Tu, Hugo Jair Escalante, Ling Yue, Xiawei Guo, Isabelle Guyon,
Daniel L. Silver, Evelyne Viegas, Yuqiang Chen, Wenyuan Dai, Qiang Yang

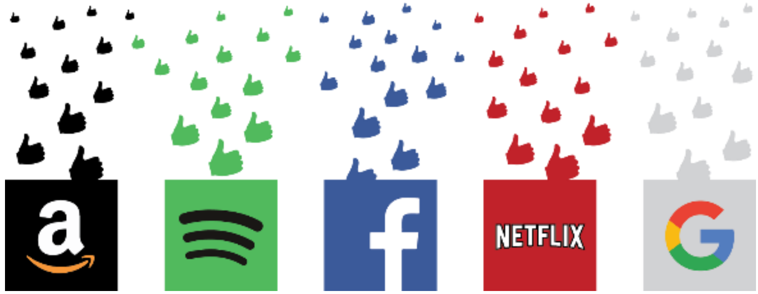
Schedule

14:10 ~ 14:25	Overview and Award Ceremony
14:25 ~ 14:35	*4th Place Solution Presentation*
14:35 ~ 14:45	*3rd Place Solution Presentation*
14:45 ~ 14:55	*2nd Place Solution Presentation*
14:55 ~ 15:05	*1st Place Solution Presentation*
15:05 ~ 15:10	Photo Time

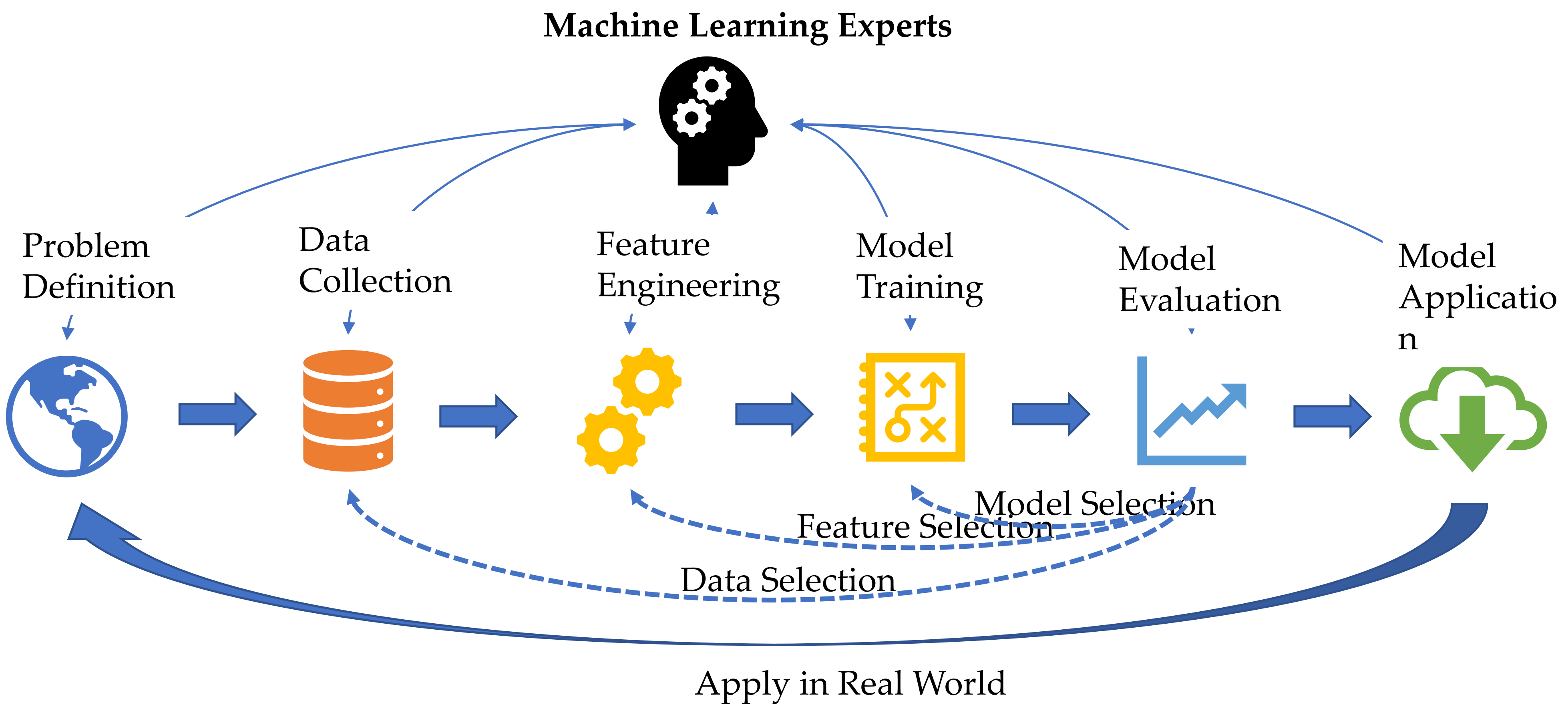
Machine Learning Applications

AlphaGo

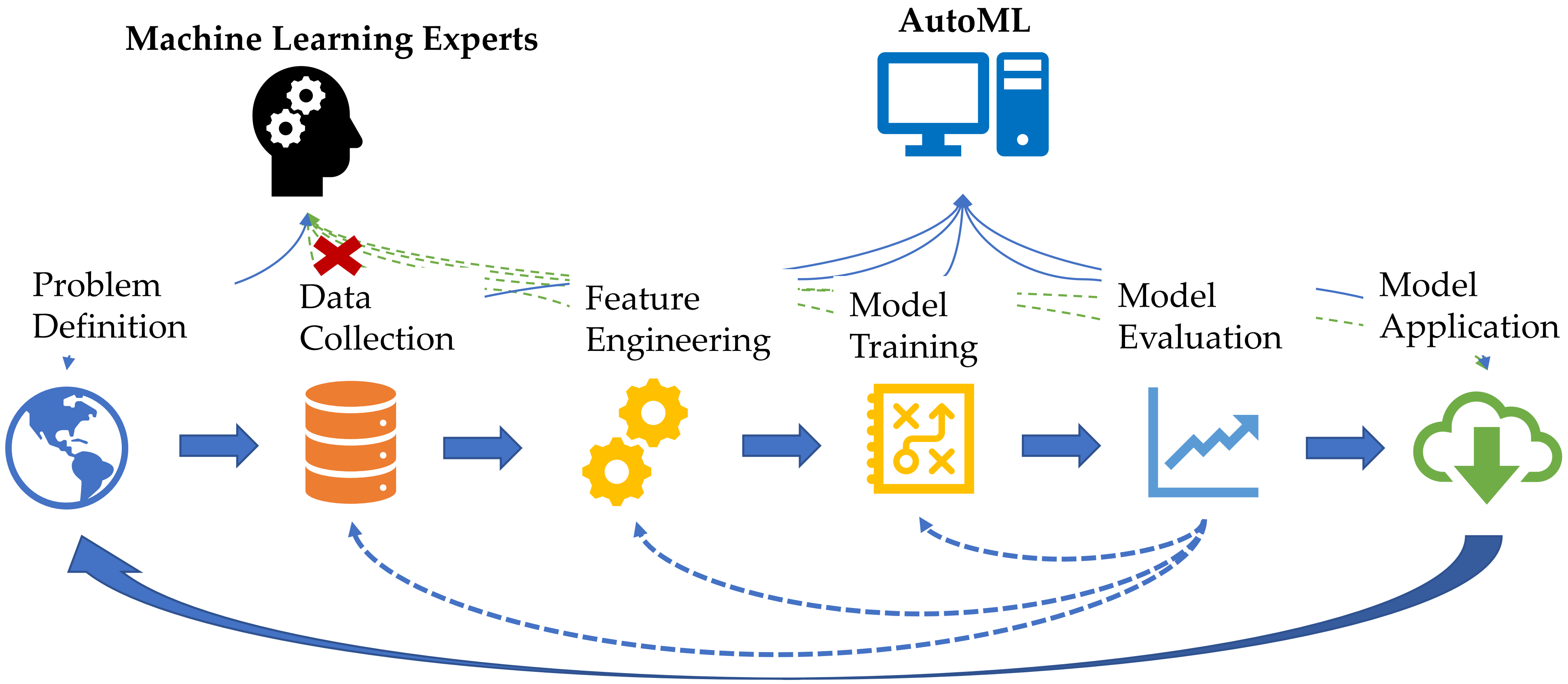
Google
AdWords



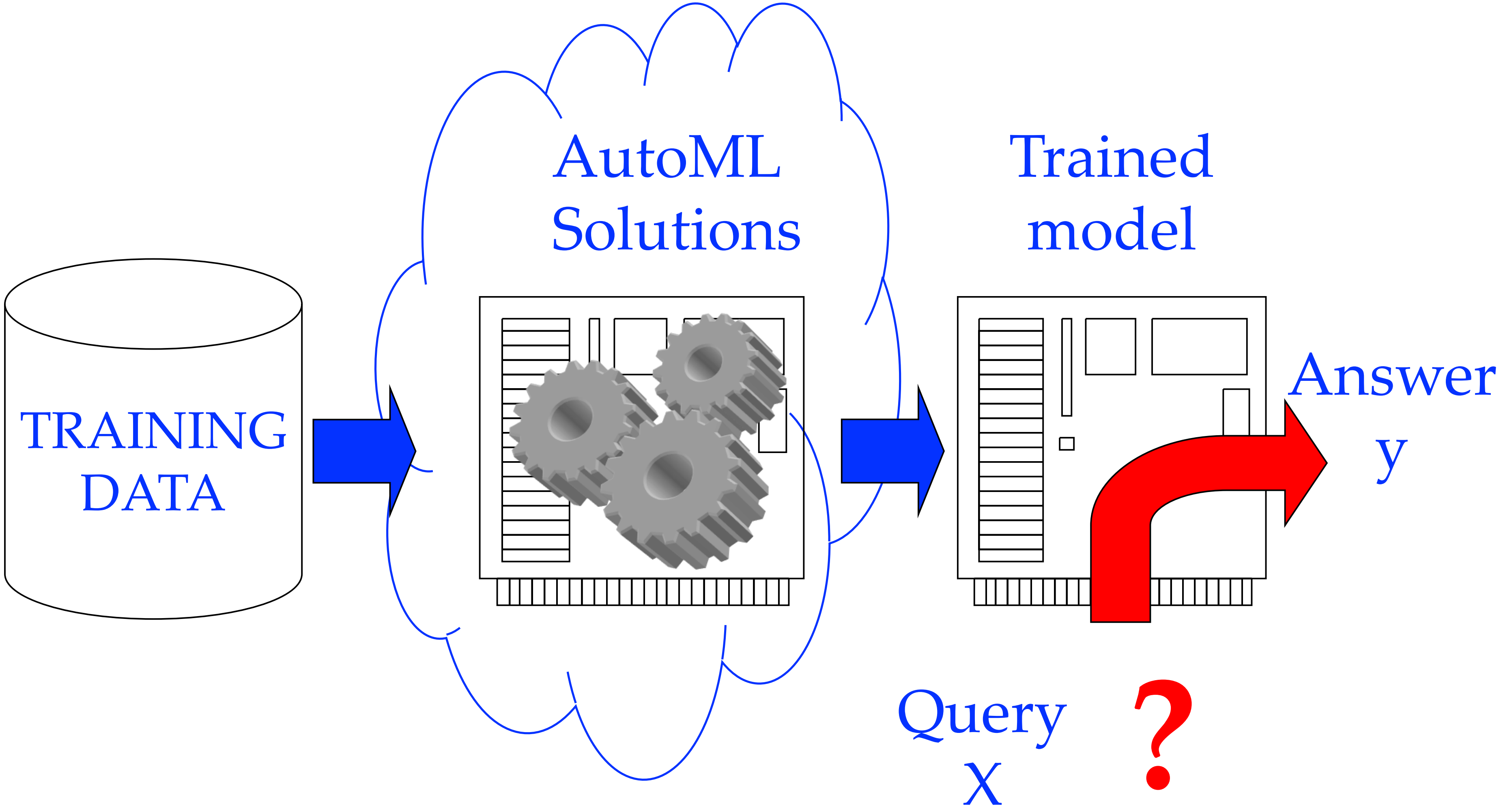
Challenges in Real World Applications



Solution: AutoML



The AutoML Challenges

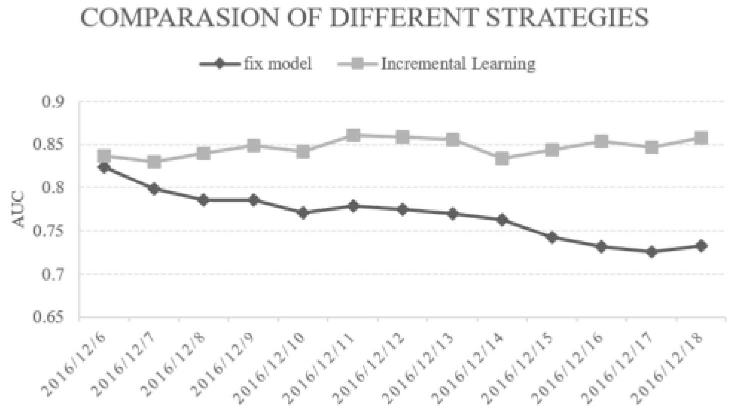


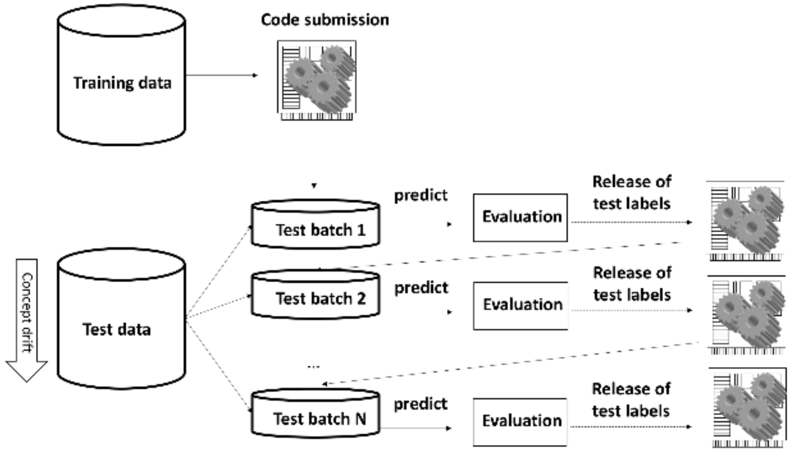
Brief History of AutoML Challenges

Competition	Year	Lasts	Collocated Events	#Participants	Prizes	Providers & Sponsors
AutoML1	2015-2016	2 Years	NIPS, ICML, IJCNN	600+	30,000 USD	Microsoft and ChaLearn
AutoML2	2018	4 Months	PAKDD 2018	250+	10,000 USD	4Paradigm and ChaLearn
AutoML3	2018	3 Months	NeurIPS 2018	300+	15,000 USD	4Paradigm, Microsoft and ChaLearn
AutoML4 (AutoML3+)	**2019**	**3 Months**	**PAKDD 2019**	**130+ Teams**	**6,500 USD**	**4Paradigm, ChaLearn, Microsoft and Amazon**
AutoML5	**2019**	**3 Months Ongoing!**	**KDD Cup 2019**	**~500 Teams (growing)**	**33,500 USD**	**4Paradigm, ChaLearn, Microsoft and Amazon**
AutoML6 (AutoCV)	2019	Coming Soon			10,000 USD	ChaLearn, 4Paradigm and Google
AutoML7 (AutoDL)	2019	Coming Soon	NeurIPS 2019		15,000 USD	ChaLearn, 4Paradigm and Google

AutoML4 at PAKDD 2019

- A **rematch** of AutoML3 @ NeurIPS 2018
- AutoML challenged by:
  - **Scalability**. Data sets larger than ever will be considered (Up to 10M instances)
  - **Concept drift**. Dependency between instances, concept changing through time.
  - **LifeLong setting**. Evaluation of the lifelong capabilities of learning machines.

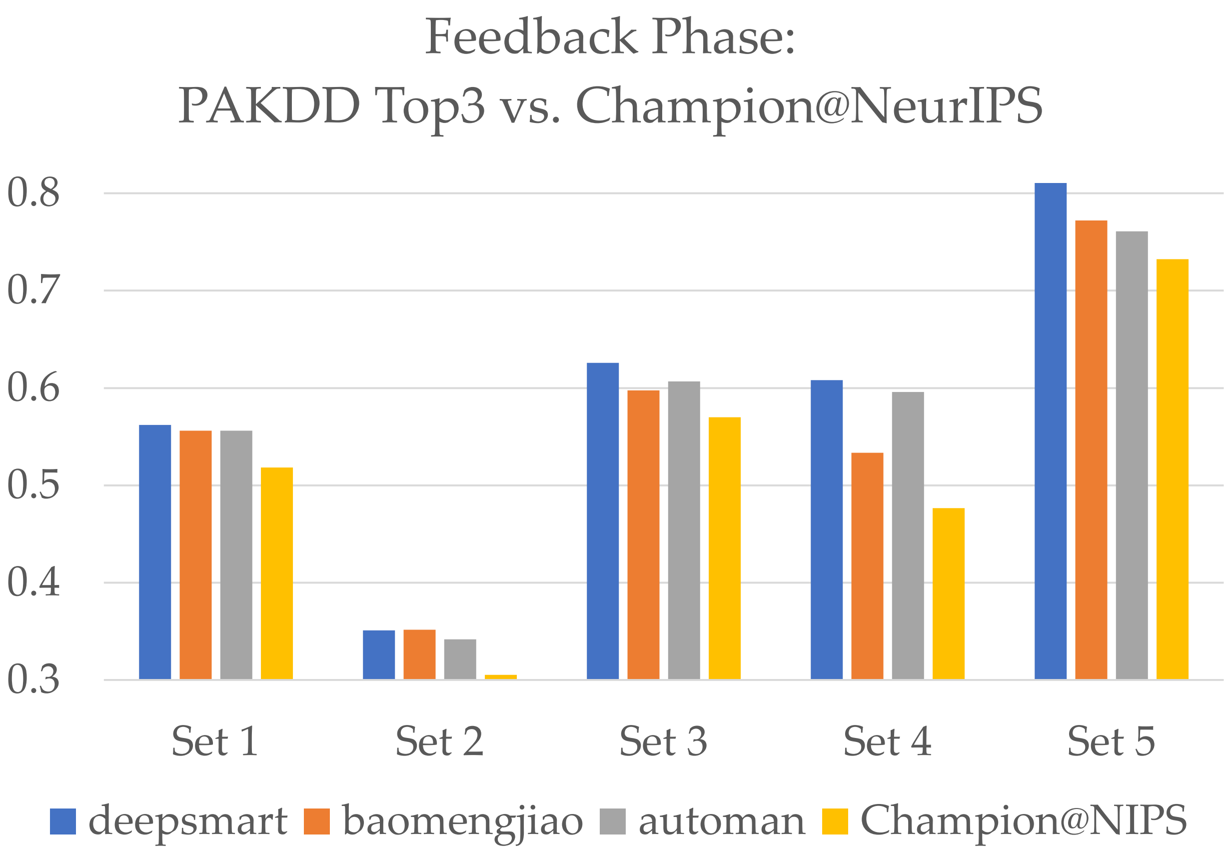




An Overview

- Lasts 3 months
- About 130 teams, 27 teams qualified for the final ranking
- 550+ submissions!
- +50% more Time Budget than NeurIPS 2018
- **Much Better** Performance than Champion@NeurIPS

Great Improvement vs. Champion@NeurIPS!



Winners!

- **Champion: DeepBlueAI**
  - *Zhipeng Luo, Jianqiang Huang, Mingjian Chen*
  - DeepBlue Technology (Shanghai) Co., Ltd

- **Second Place: ML Intelligence**
  - *Mengjiao Bao, Hui Xue, Yihuan Mao, Yujing Wang*
  - Microsoft Research Asia, Beihang University

- **Third Place: Meta_Learners**
  - *Zheng Xiong, Jiyan Jiang, Wenpeng Zhang*
  - Tsinghua University

Following Activities

- KDD Cup 2019 AutoML Competition

KDD Cup 2019
AutoML
for Temporal Relational Data
4Paradigm ChaLearn Microsoft

- AutoML Workshop @ ICML 2019

- AutoML Special Issue at TPAMI

Call for papers - Special Issue
Automation in AI and Machine Learning
IEEE TRANSACTIONS ON
PATTERN ANALYSIS AND
MACHINE INTELLIGENCE

- AutoDL Competition at NeurIPS 2019

ChaLearn Google 4Paradigm
Help Automating Deep Learning

We’re Hiring!

第四范式公众号

Let’s begin!

PAKDD 2019 Challenge
The 4th AutoML Challenge (AutoML3+): AutoML for Lifelong Machine Learning

LightGBM for Lifelong Machine Learning

Yikang Zheng[1], Chunhui Bao[2]

[1]Beijing Institute of Technology, [2]Sichuan University

Agenda

- Problem Overview
- Existing Approaches
- Data Preprocessing
- LightGBM Algorithm
- Experiments & Results
- Conclusion

Problem Overview

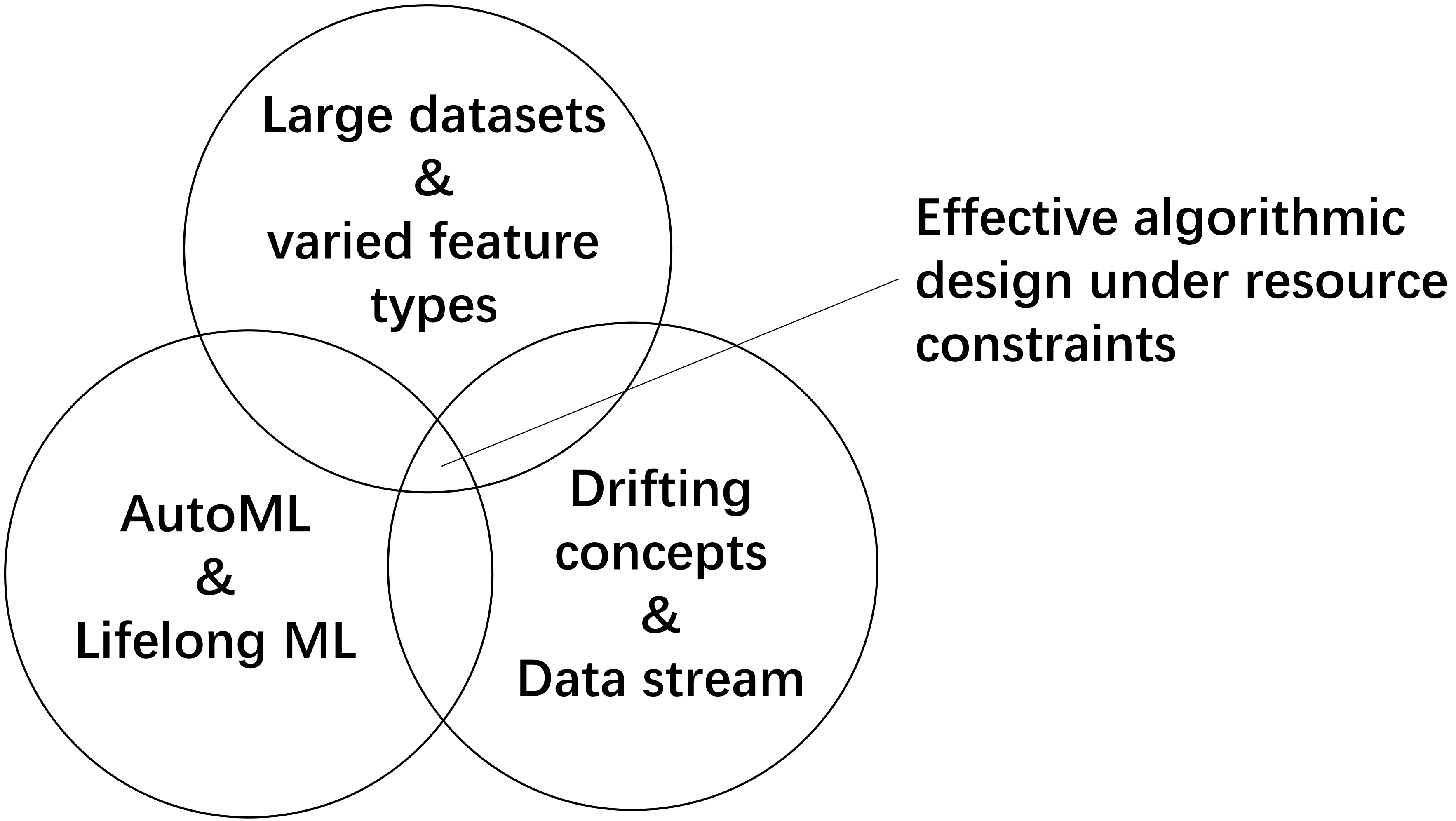
- **Large datasets and varied feature types**
- **Drifting concepts [1]**

getting away from the simpler i.i.d. cases, data distributions are changing relatively slowly over time.

- **Data stream**

Real-world data arrives as streams/batches ordered in time.

- **AutoML [2] system combined with Lifelong Machine Learning.**
- **Resource constraints**


Large datasets
&
varied feature
types
Effective algorithmic
design under resource
constraints
AutoML
&
Lifelong ML
Drifting
concepts
&
Data stream

Existing Approaches

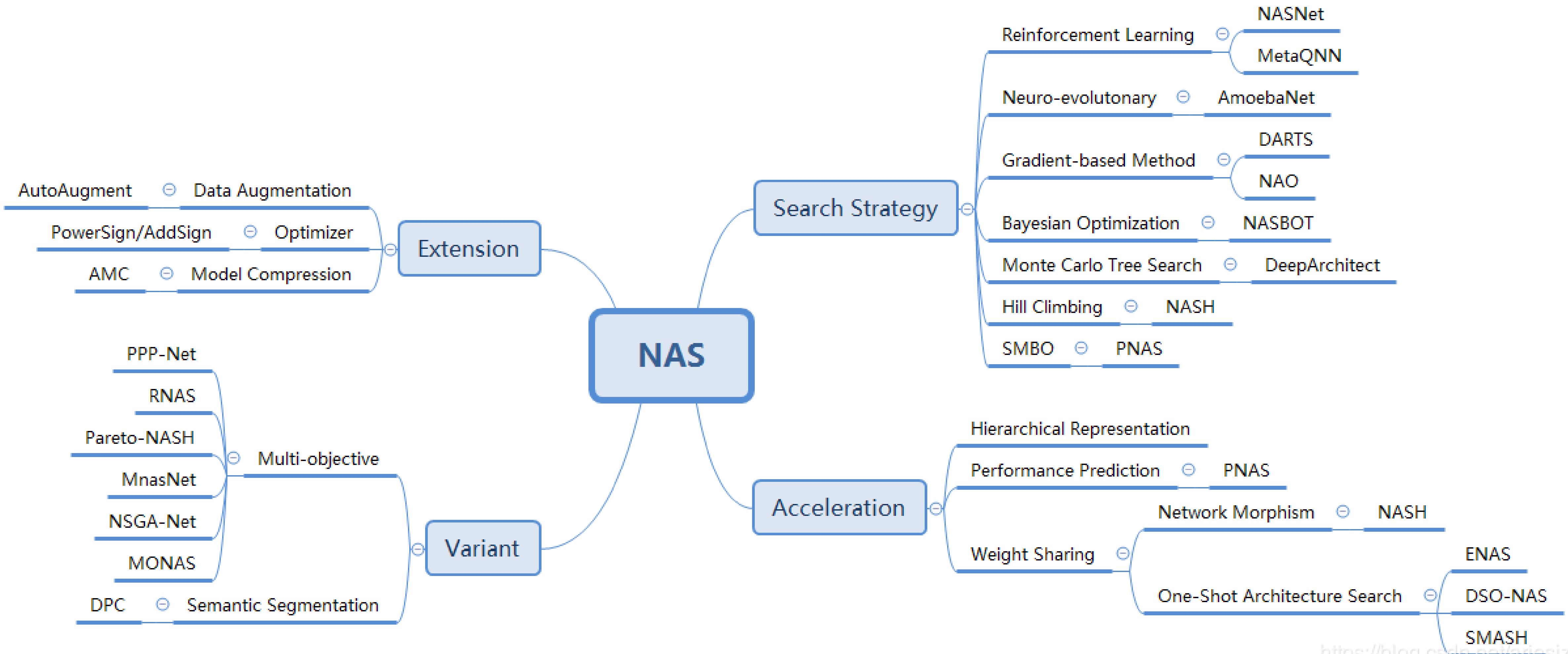
Neural Architecture Search（NAS）[3]

Merit: Automatically search and optimize the structures of neural networks.

Defect: Slow convergence speed and huge resource consumption.

Auto sklearn [4]

Automatically search and optimize the hyperparameters of models.


NAS
Search Strategy
Reinforcement Learning
NASNet
MetaQNN
Neuro-evolutonary
AmoebaNet
Gradient-based Method
DARTS
NAO
Bayesian Optimization
NASBOT
Monte Carlo Tree Search
DeepArchitect
Hill Climbing
NASH
SMBO
PNAS
Acceleration
Hierarchical Representation
Performance Prediction
PNAS
Weight Sharing
Network Morphism
NASH
One-Shot Architecture Search
ENAS
DSO-NAS
SMASH
Extension
Data Augmentation
AutoAugment
Optimizer
PowerSign/AddSign
Model Compression
AMC
Variant
Multi-objective
PPP-Net
RNAS
Pareto-NASH
MnasNet
NSGA-Net
MONAS
Semantic Segmentation
DPC

Data Preprocessing

There are four types of data. For different data types, we adopt different preprocessing methods.

1. ***Categorical* Feature:** an integer describing which category the instance belongs to.

   preprocessing methods: Hash coding and frequency coding

	Feature1	Feature2	…	Feature1	Feature2
1	Male	Alabama		0.6	0.2
2	Female	Missouri	Frequency Encoding →	0.4	0.2
3	Male	Texas		0.6	0.4
4	Male	Hawaii		0.6	0.2
5	Female	Texas		0.4	0.4

2. ***Numerical*** **Feature:** a real value.

preprocessing methods: standardization

For a random variable X, standardization means converting X to its standardized random variable X^*

$$X^* = \frac{X - E(X)}{\sqrt{D(X)}}$$

Where $E(X)$ denotes the mean value of X and $D(X)$ denotes the variance of X.

3. *Multi-value Categorical* **Feature:** a set of integers, split by the comma.

preprocessing methods: Hash coding and frequency coding

4. *Time* **Feature:** an integer describing time information.

preprocessing methods: Using second-order features

LightGBM Algorithm

LightGBM [5]

Gradient boosting decision tree (GBDT) combined with Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) is called LightGBM.

$$\text{LightGBM} = \text{GBDT} + \text{GOSS} + \text{EFB}$$

Merits:

High computational speed

Small memory consumption

High rate of Accuracy

Algorithm 1: Histogram-based Algorithm

```
Input: I: training data, d: max depth
Input: m: feature dimension
nodeSet ← {0} ▷ tree nodes in current level
rowSet ← {{0, 1, 2, ...}} ▷ data indices in tree nodes
for i = 1 to d do
    for node in nodeSet do
        usedRows ← rowSet[node]
        for k = 1 to m do
            H ← new  Histogram()
            ▷ Build histogram
            for j in usedRows do
                bin ← I.f[k][j].bin
                H[bin].y ← H[bin].y + I.y[j]
                H[bin].n ← H[bin].n + 1
            Find the best split on histogram H.
            ...
    Update rowSet and nodeSet according to the best
    split points.
    ...
```

Algorithm 2: Gradient-based One-Side Sampling

```
Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: loss: loss function, L: weak learner
models ← {}, fact ← (1−a)/b
topN ← a × len(I) , randN ← b × len(I)
for i = 1 to d do
    preds ← models.predict(I)
    g ← loss(I, preds), w ← {1,1,...}
    sorted ← GetSortedIndices(abs(g))
    topSet ← sorted[1:topN]
    randSet ← RandomPick(sorted[topN:len(I)],
    randN)
    usedSet ← topSet + randSet
    w[randSet] × = fact ▷ Assign weight fact to the
    small gradient data.
    newModel ← L(I[usedSet], − g[usedSet],
    w[usedSet])
    models.append(newModel)
```

Algorithm 3: Greedy Bundling

```
Input: F: features, K: max conflict count
Construct graph G
searchOrder ← G.sortByDegree()
bundles ← {}, bundlesConflict ← {}
for i in searchOrder do
    needNew ← True
    for j = 1 to len(bundles) do
        cnt ← ConflictCnt(bundles[j],F[i])
        if cnt + bundlesConflict[i] ≤ K then
            bundles[j].add(F[i]), needNew ← False
            break
    if needNew then
        Add F[i] as a new bundle to bundles
Output: bundles
```

Algorithm 4: Merge Exclusive Features

```
Input: numData: number of data
Input: F: One bundle of exclusive features
binRanges ← {0}, totalBin ← 0
for f in F do
    totalBin += f.numBin
    binRanges.append(totalBin)
newBin ← new   Bin(numData)
for i = 1 to numData do
    newBin[i] ← 0
    for j = 1 to len(F) do
        if F[j].bin[i] ≠ 0 then
            newBin[i] ← F[j].bin[i] + binRanges[j]
Output: newBin, binRanges
```

Experiments & Results

Method	A	B	C	D	E	Score
LightGBM	0.5262	0.3103	0.5115	0.4761	0.7357	0.5262

Dataset	Hyperparameters
A	{'bagging_fraction': 0.6429102033075881, 'bagging_freq': 1, 'boosting_type': 'gbdt', 'feature_fraction': 0.5751024007960667, 'learning_rate': 0.01776646216240752, 'metric': 'auc', 'n_estimators': 650, 'num_leaves': 120, 'objective': 'binary', 'verbose': -1}
B	{'bagging_fraction': 0.6801121812229166, 'bagging_freq': 2, 'boosting_type': 'gbdt', 'feature_fraction': 0.5651806829109673, 'learning_rate': 0.016275184165815373, 'metric': 'auc', 'n_estimators': 600, 'num_leaves': 80, 'objective': 'binary', 'verbose': -1}
C	{'bagging_fraction': 0.5677632211211332, 'bagging_freq': 1, 'boosting_type': 'gbdt', 'feature_fraction': 0.6674443293907164, 'learning_rate': 0.014287195835157099, 'metric': 'auc', 'n_estimators': 650, 'num_leaves': 90, 'objective': 'binary', 'verbose': -1}
D	{'bagging_fraction': 0.6920539291952178, 'bagging_freq': 2, 'boosting_type': 'gbdt', 'feature_fraction': 0.526846643677197, 'learning_rate': 0.016886243904186683, 'metric': 'auc', 'n_estimators': 600, 'num_leaves': 110, 'objective': 'binary', 'verbose': -1}
E	'bagging_fraction': 0.6558659725219756, 'bagging_freq': 3, 'boosting_type': 'gbdt', 'feature_fraction': 0.6058620454195228, 'learning_rate': 0.01436197593104331, 'metric': 'auc', 'n_estimators': 650, 'num_leaves': 110, 'objective': 'binary', 'verbose': -1

Conclusion

- LightGBM can effectively solve this problem.
- Second-order feature processing for time feature is effective for this problem.

References

- [1] P. B. Dongre and L. G. Malik, "A review on real time data stream classification and adapting to various concept drift scenarios," in *2014 IEEE International Advance Computing Conference (IACC)*, 2014, pp. 533-537.
- [2] J. Madrid, H. J. Escalante, E. Morales, W.-W. Tu, Y. Yu, L. Sun-Hosoya, et al., "Towards AutoML in the presence of Drift: first results," in Workshop AutoML 2018@ ICML/IJCAI-ECAI, 2018
- [3] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," arXiv preprint arXiv:1611.01578, 2016.
- [4] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in Advances in neural information processing systems, 2015, pp. 2962-2970.
- [5] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, et al., "Lightgbm: A highly efficient gradient boosting decision tree," in Advances in Neural Information Processing Systems, 2017, pp. 3146-3154.

Thank you

A Boosting Tree Based AutoML System with Concept Drift Adaptation

Meta_Learners:

Zheng Xiong, Jiyan Jiang, Wenpeng Zhang

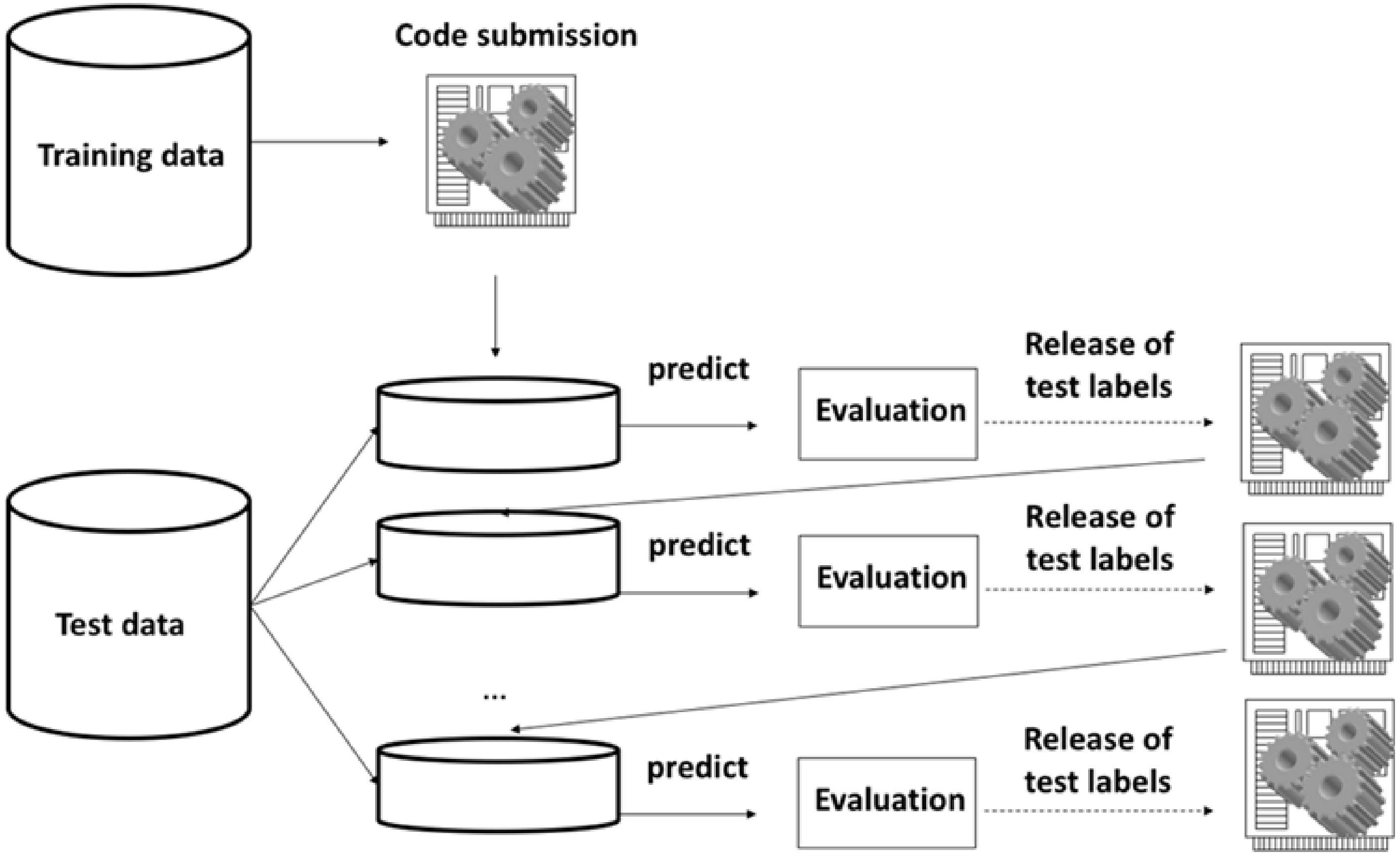
Advisor: Prof. Wenwu Zhu

Department of Computer Science, Tsinghua University, Beijing

Outline

- Problem Statement
- System Framework
- Automated Feature Engineering
- Concept Drift Adaptation
- Conclusion

Problem Statement



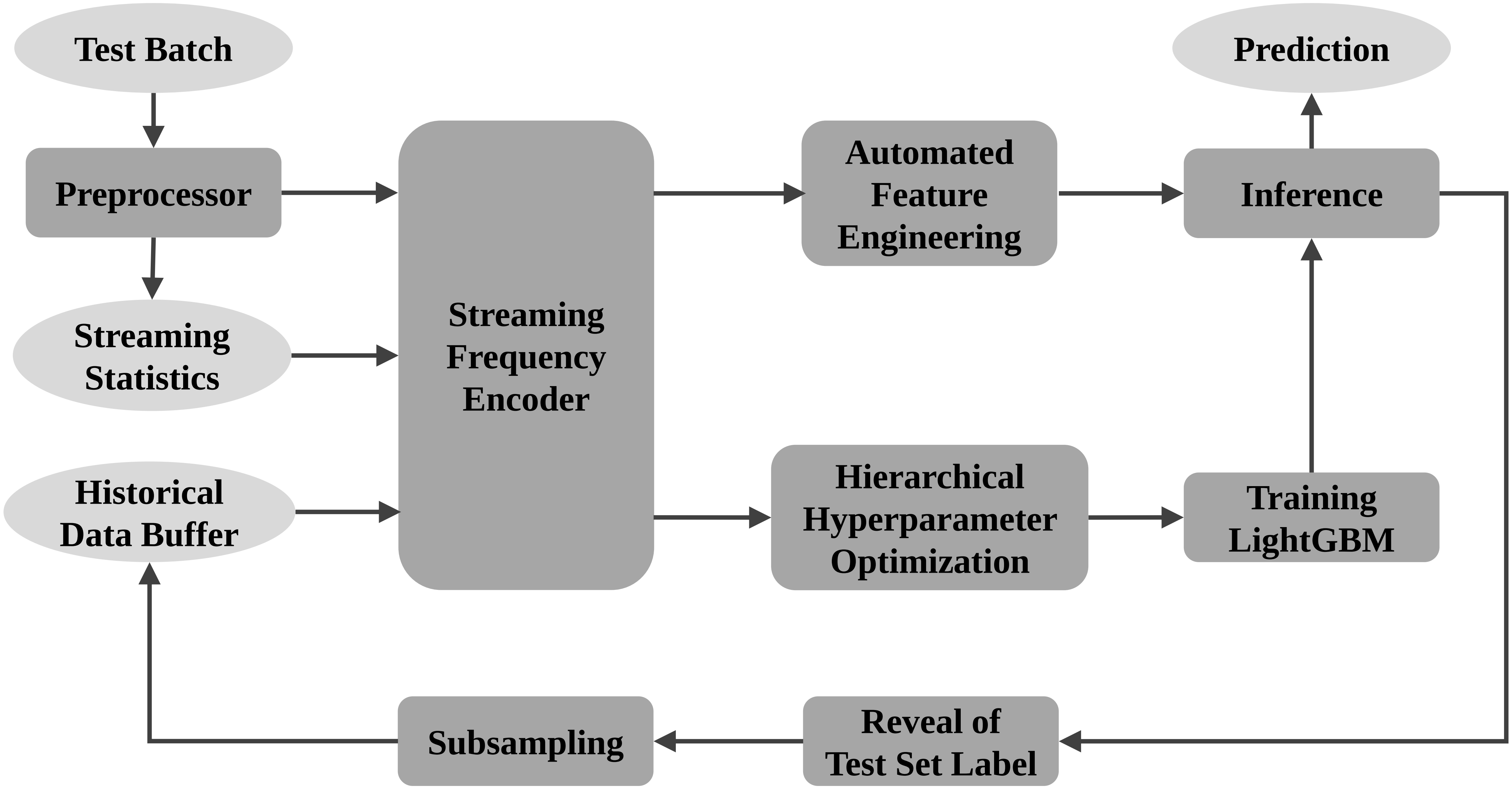
AutoML: the final submission of the feedback phase is blindly tested on 5 unseen new datasets without human intervention

Concept drift: data comes in stream with data distribution changing between batches

System Framework

- Use gradient boosting tree (GBT) as the classifier across different datasets
- Perform automated feature engineering to improve model performance
- Tackle concept drift by retraining and streaming co-encoding
- Optimize hyperparameters hierarchically to improve the efficiency and robustness of the system

System Framework



Automated Feature Engineering

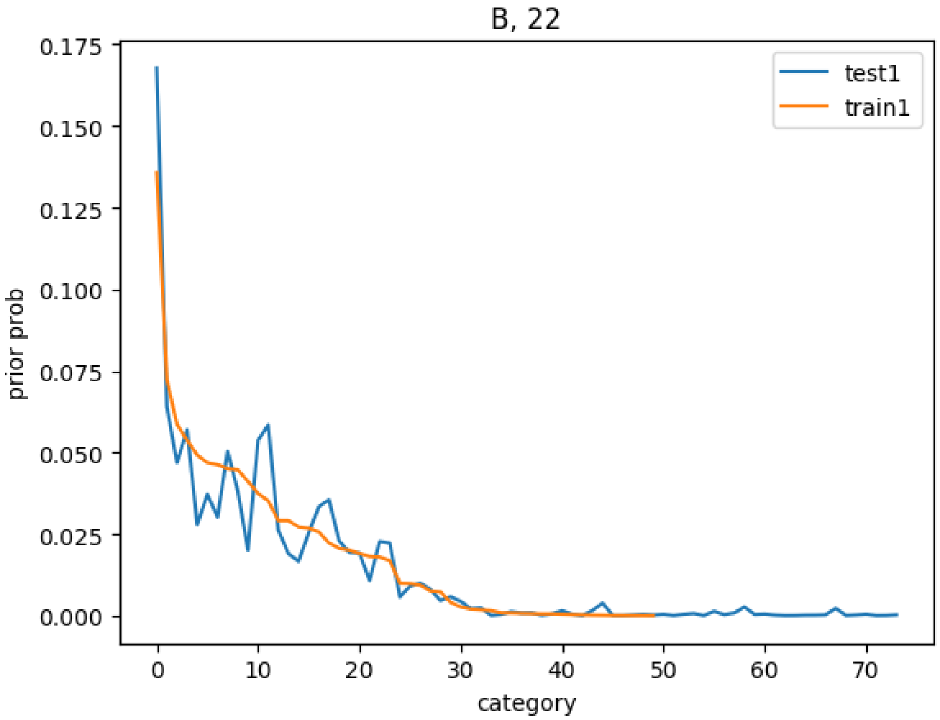
- **First-order feature engineering**
  - Frequency encoding of categorical features

- **High-order feature engineering**
  - Predefine a set of binary transformations based on prior knowledge
  - Apply each type of transformation on the original feature set to generate new features in an expansion-reduction fashion

Automated Feature Engineering

- **High-order feature engineering**
  - Predefined binary transformations:
    - Numerical-numerical: $+, -, \times, \div$
    - Categorical-numerical: num_mean_groupby_cat
    - Categorical-categorical: cat_cat_combine, cat_nunique_groupby_cat
    - Categorical-temporal: time_difference_groupby_cat
  - Key steps in the expansion-reduction strategy:
    - **Pre-selection**: select features used for feature generation based on prior knowledge
    - **Feature generation**: generate new feature with all feasible pairs of the pre-selected features
    - **Post-selection**: select generated features based on the performance and feature importance of a coarsely trained GBT model

Concept Drift Adaptation



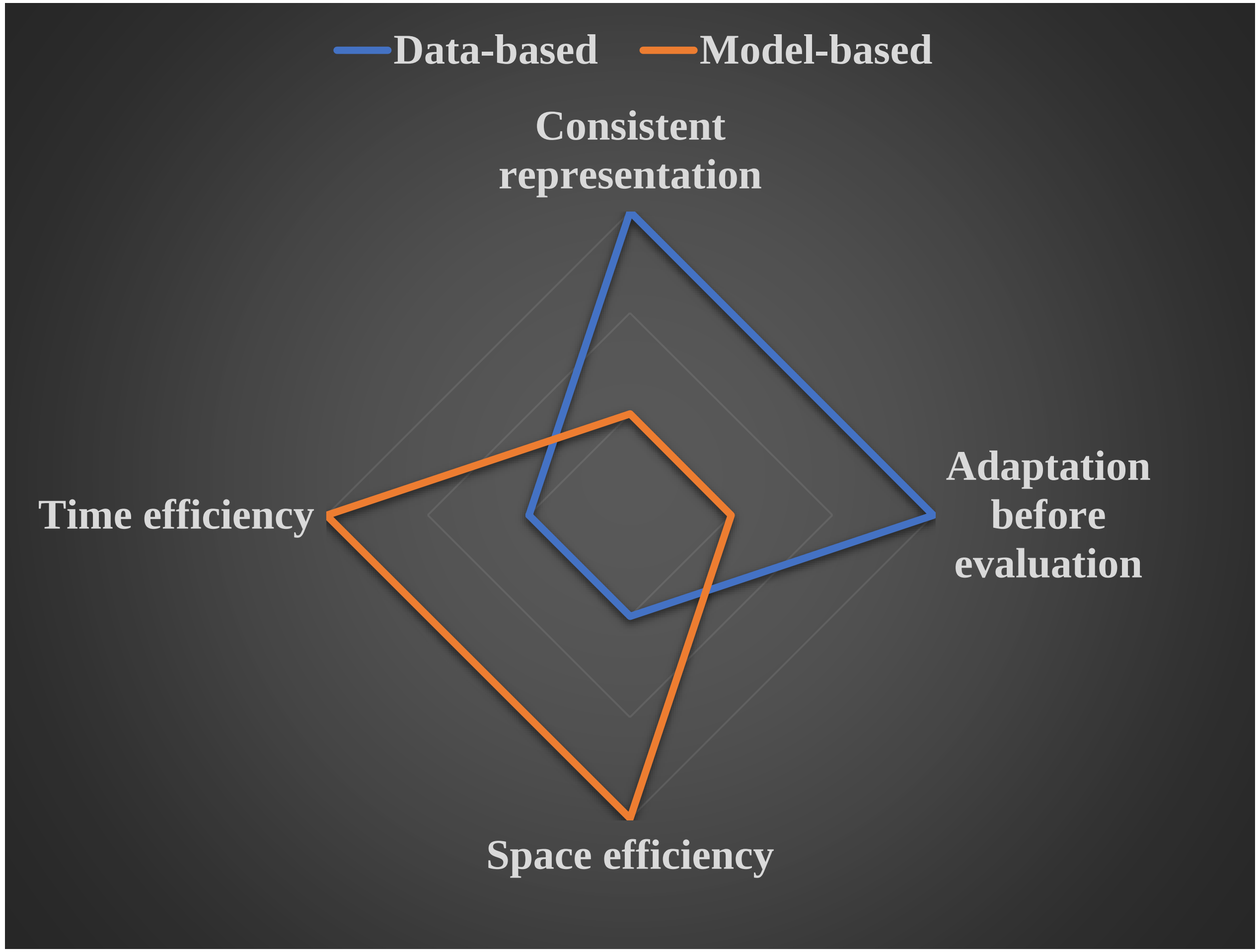
Concept drift in categorical features

- ➢ Many unseen new categories may appear in the test batch
- ➢ The frequency of existing categories may change significantly between batches

Concept Drift Adaptation

Different strategies for concept drift adaptation

- Data-based: retrain a new model with historical data
- Model-based: create an ensemble with models trained on previous batches



Concept Drift Adaptation

Our solution:

- Retrain a GBT model with the last N batches for each test batch
- Apply streaming co-encoding to achieve a consistent representation between training set and test set
- Strategies to improve space and time efficiency:
    - Data subsampling
    - Streaming statistics

Conclusion

- Propose a boosting tree based AutoML system with concept drift adaptation
- Design an efficient automated feature engineering strategy which significantly improves model performance
- Adapt to concept drift by retraining and streaming co-encoding
- Future work:
  - Explicit concept drift detection and adaptation
  - Generalization and scalability of automated feature engineering

Thank You!

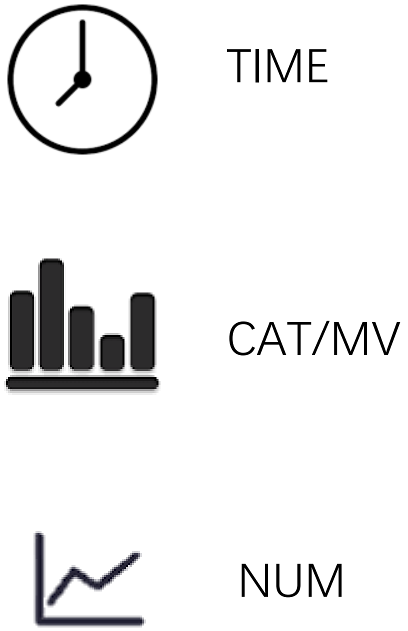
AutoML Challenge @ PAKDD 2019
Team: ML Intelligence

Mengjiao Bao[1,2], Huixue[1], Yihuan Mao[1], Yujing Wang[1]
Microsoft Research Asia[1], Beihang University[2]

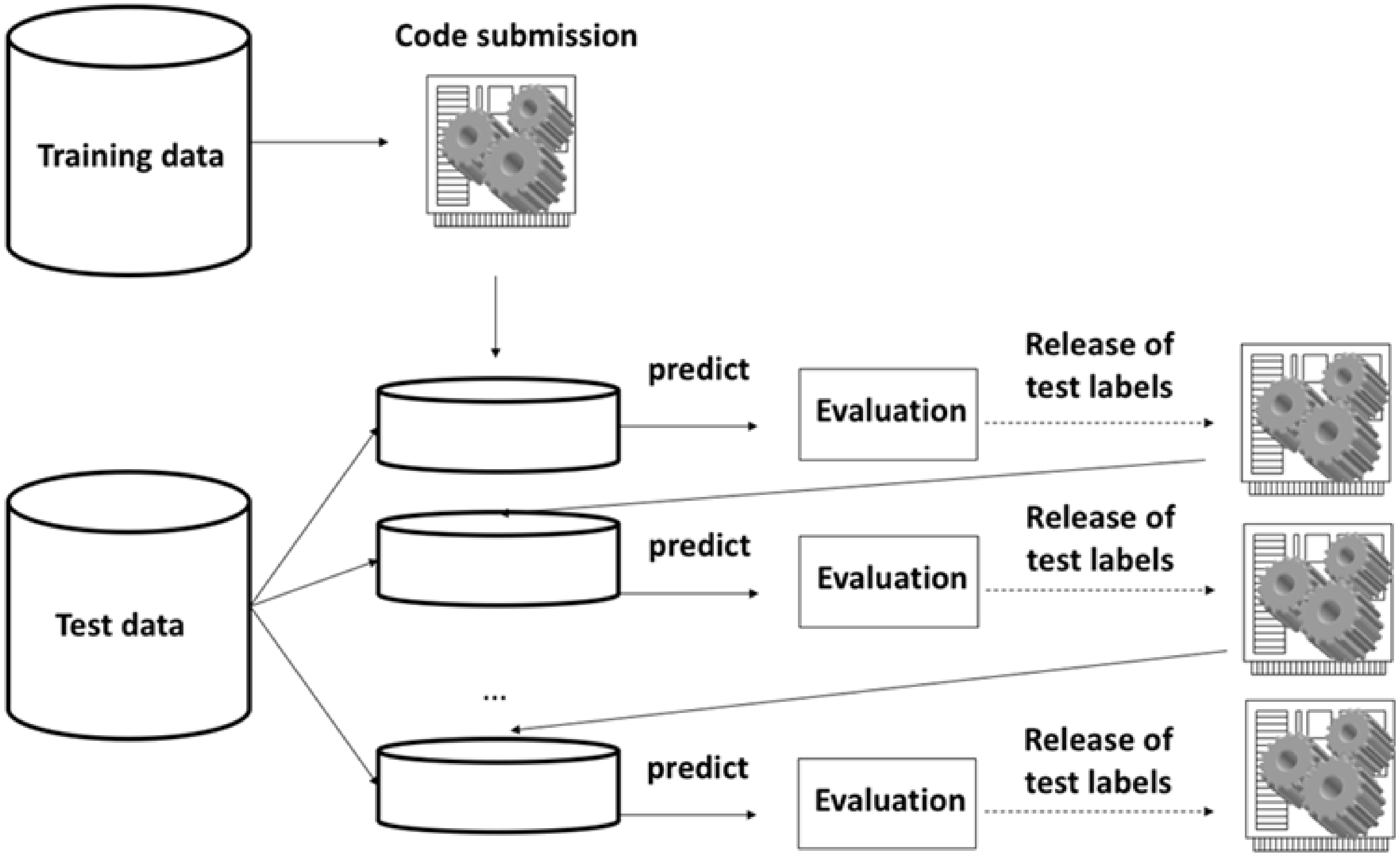
Content

- Overview
- Key Challenge
- Pipeline
- Concept Drift
- Auto Part

Overview



TIME
CAT/MV
NUM

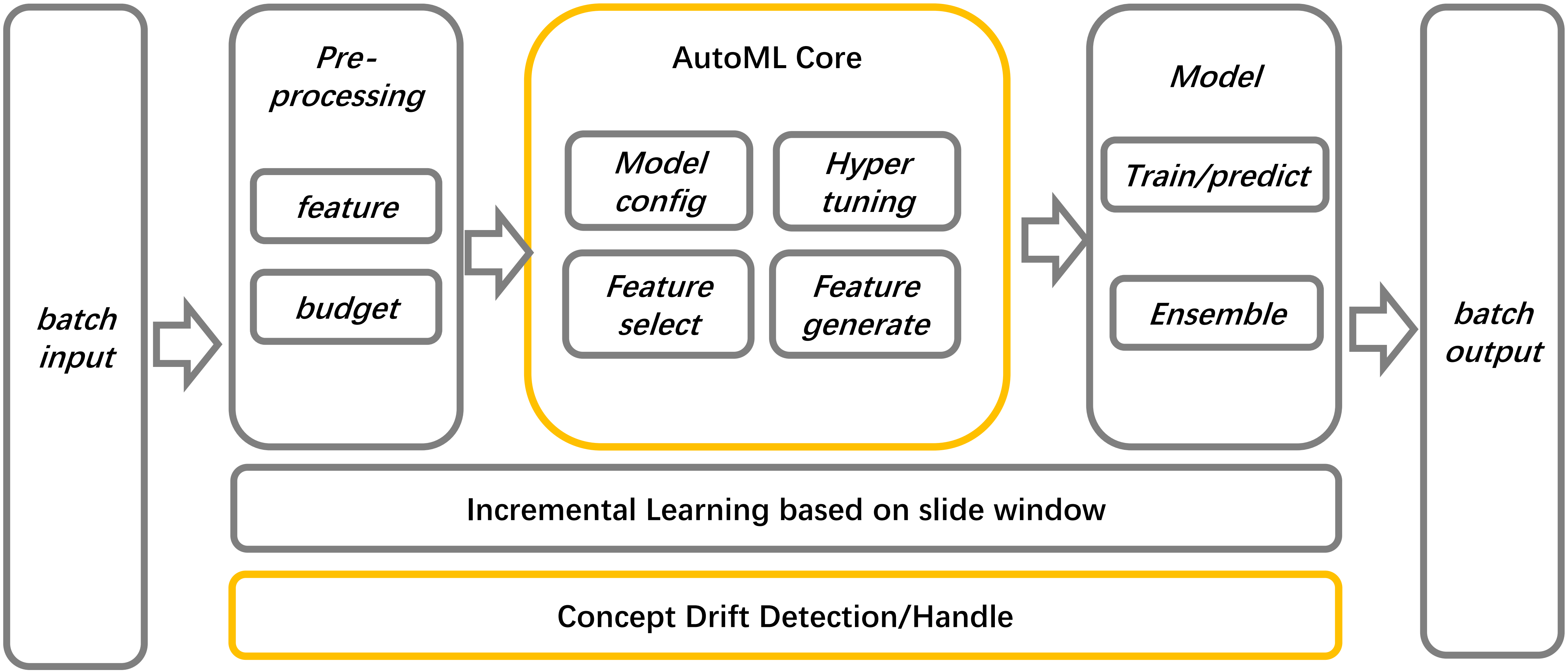


Training data
Code submission
Test data
predict
Evaluation
Release of test labels
predict
Evaluation
Release of test labels
...
predict
Evaluation
Release of test labels

Difficulties

- **Algorithm scalability:** large datasets(10^7)
- **Varied feature types:** continuous, binary, ordinal, categorical, multi-value categorical, temporal
- **Concept drift:** Distribution change over time
- **Lifelong setting:** Datasets chronologically -> 10 batches, test -> labels reveal.

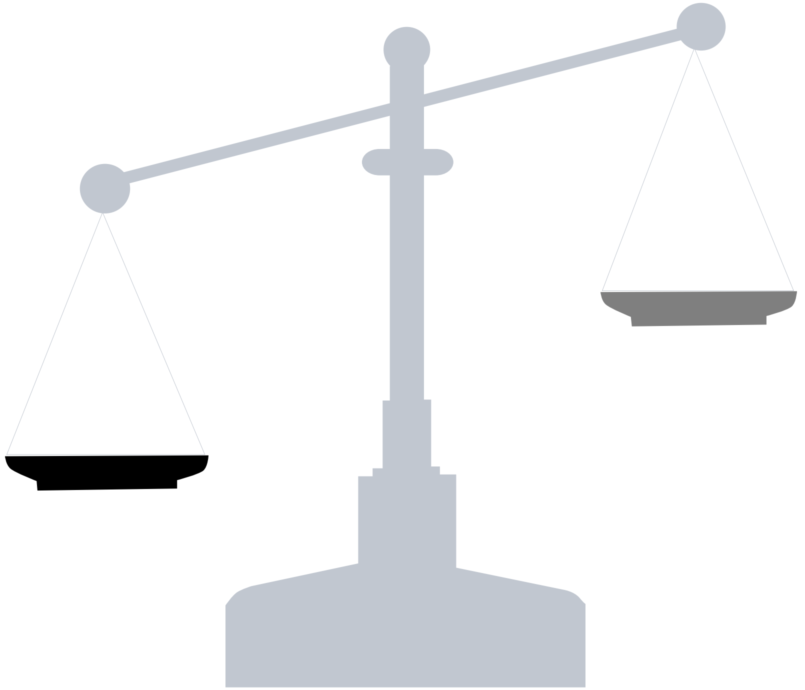
Pipline


batch input
Pre-processing
feature
budget
AutoML Core
Model config
Hyper tuning
Feature select
Feature generate
Model
Train/predict
Ensemble
batch output
Incremental Learning based on slide window
Concept Drift Detection/Handle

Auto Model Config

Performance

1.More Feature
2.More Data
3.More Parameter



Budget

1.Time Limit
2.RAM Limit

$$Estimate = len(Data) * (\alpha + \beta + \gamma) * len(Cat_{feature})$$

Auto Feature Selection

- Rank methods(for selection and beam search)
  - Naive feature importance generated by LightGBM
  - Bagging methods: 5-kflods Feature importance average / Bagging with different LightGBM models
  - Feature selection with null importance

- Drop methods
  - time accumulation LightGBM importance

Feature Encoding Methods for HD cat

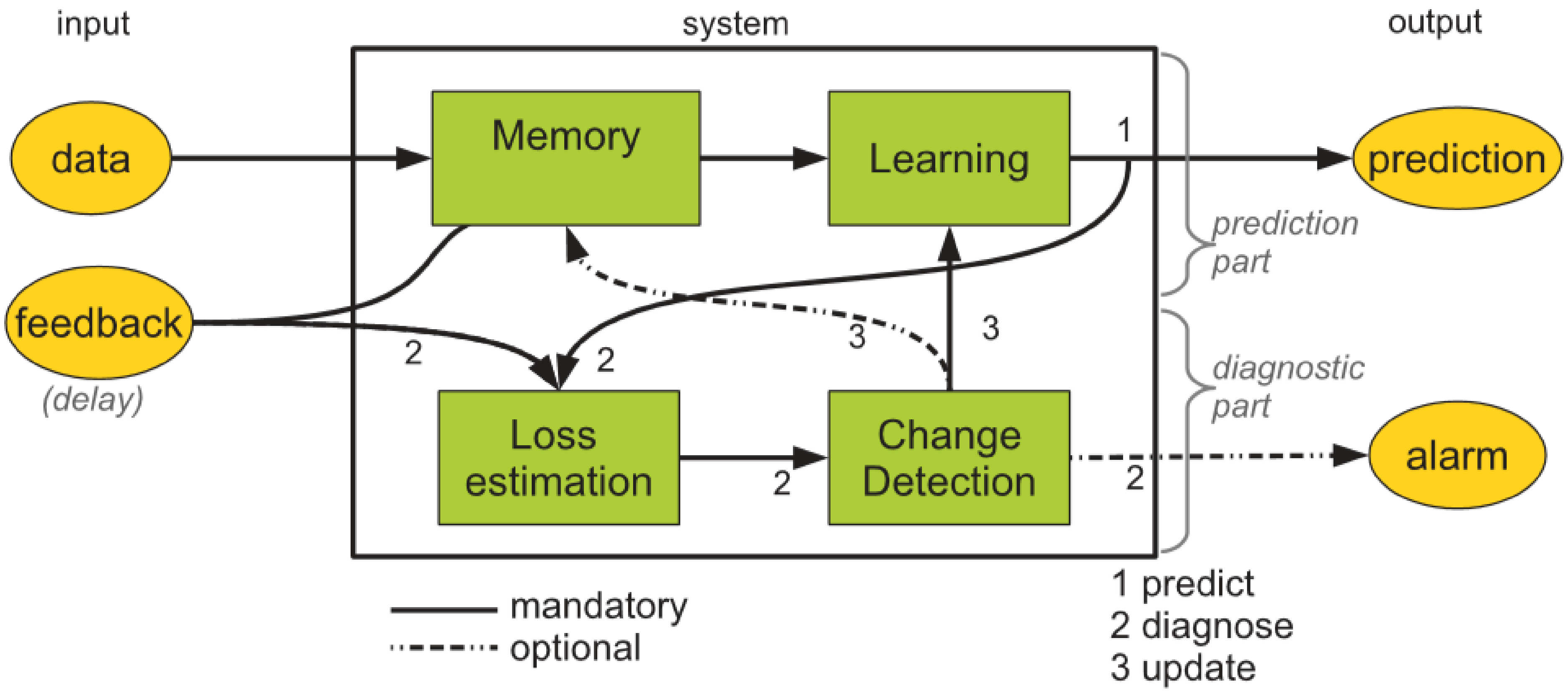
AutoGBT	Hash encode	0.46	0.18	0.45	0.33	0.68
	Incremental encode	0.503	0.278	0.535	0.451	0.732
	Hash + incremental	0.506	0.277	0.505	0.450	0.729
Our method	Hash encode	0.491	0.281	0.480	0.451	0.730
	Count encode	0.513	0.328	0.484	0.569	0.737
	Encode based on first batch	0.533	0.333	0.490	0.571	0.740
	Incremental (slide window)	**0.540**	**0.338**	**0.555**	**0.575**	**0.775**

Auto hyper parameter tuning（fix lr）

- a. Hyperband
- b. Tree of Parzen Estimators(implements by hyperopt)
- c. metis

	A	B	C	D	E
metis	0.812	0.666	**0.848**	0.780	0.885
TPE	0.808	0.669	**0.884**	0.784	0.883
Hand min	0.812	0.661	**0.822**	0.774	0.885
Had max	0.814	0.672	**0.826**	0.782	0.886

Handle Concept Drift Overview

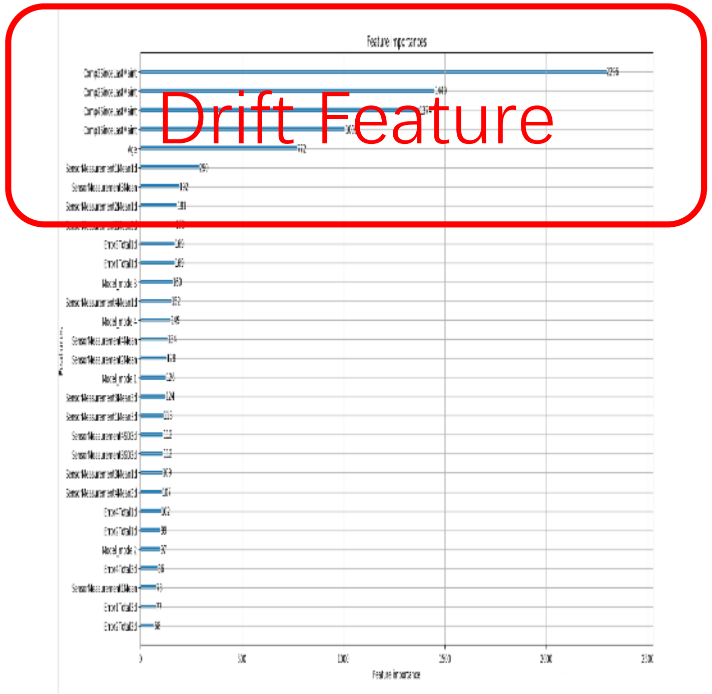


Detection and Handle

TRAIN POSITIVE

TEST NEGATIVE

Simple GBDT split train and test



Decision Tree importance

Basic important feature

High-order feature

Detection

handle

Detection and Handle

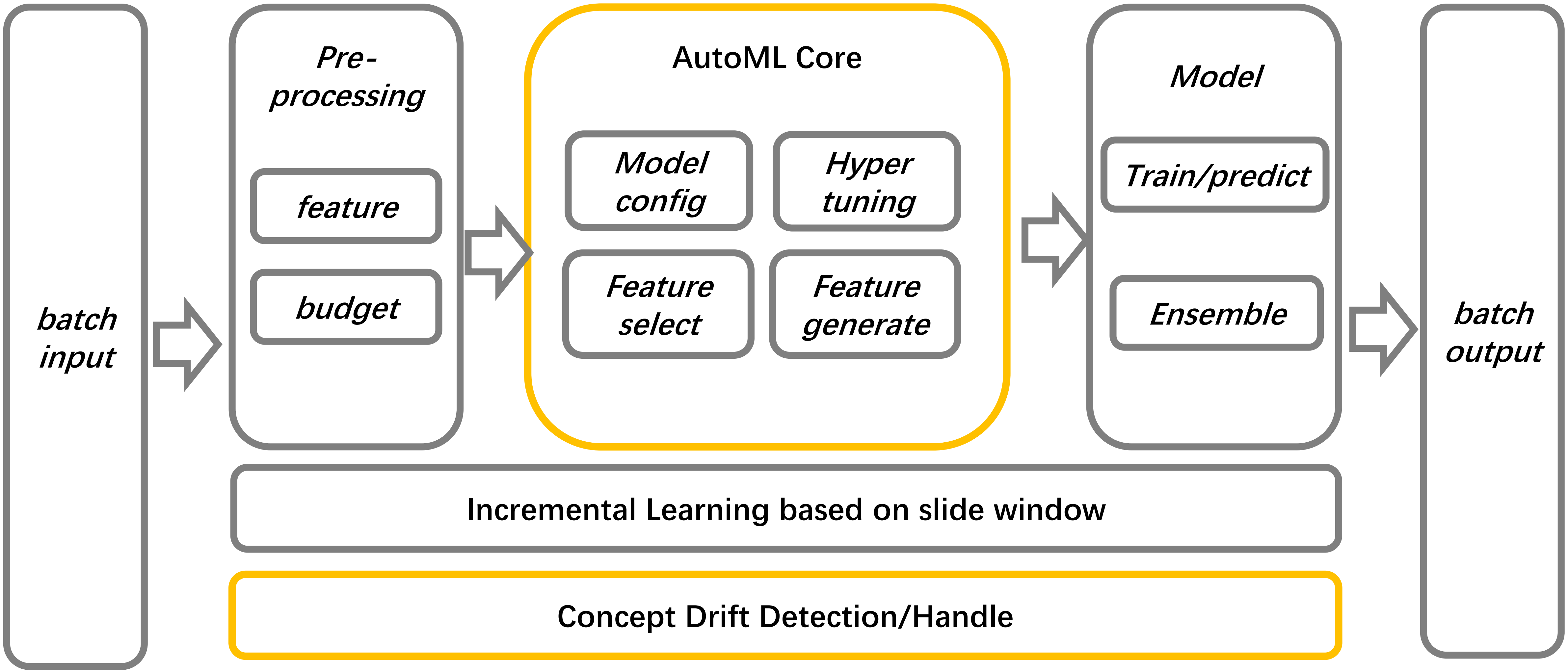
Algorithm 2 Concept Detection and Handle

Input: TrainSet D_{train}, TestSet D_{test}

Set TrainSet D_{train} as positive, TestSet D_{test} as negative;
Train a binary classifier by GBDT to split train and test.
get the *auc* score in training procedure.
Calculate the feature importance and train set *auc* score.
if *auc* > 0.65, get the most *n* largest gain feature as F_{drift}.
get concept drift feature set F_{drift}
for $f \in F_{drift}$: **do**
    if $f \notin F_{highorder}$: **then**
        delete f from F_{drift}.
    end if
end for

return Out put F'_{drift}

Conclusion



batch input
Pre-processing
feature
budget
AutoML Core
Model config
Hyper tuning
Feature select
Feature generate
Model
Train/predict
Ensemble
batch output
Incremental Learning based on slide window
Concept Drift Detection/Handle

Q&A

Neural Network Intelligence

https://github.com/Microsoft/nni

Solution to PAKDD Cup 2019

The 4th AutoML Challenge (3+)

DeepBlue Technology (Shanghai) Co., Ltd
深兰科技

01
Team members

Zhipeng Luo

DeepBlue Technology(Shanghai) Co., Ltd

Peking University

Jianqiang Huang

Peking University

Mingjian Chen

DeepBlue Technology(Shanghai) Co., Ltd

Peking University

DeepBlue Technology (Shanghai) Co., Ltd

Peking University

02

Awards

PAKDD 2019 AutoML	1st place
NeurIPS 2018 AutoML (Phase 1)	1st place
CIKM Cup 2018	1st place
KDD Cup2018(Second 24- Hour Prediction Track)	1st place
KDD Cup 2018(Last 10-Day Prediction Track)	1st place
WSDM Cup2019(Task 3,LB)	1st place
Shanghai BOT Big Data Application Competition	1st place
Daguan text Classification	1st place

CONTENTS

PART 01

Introduction

01

Challenge of this track

Direction 1:
Class im-balance

The rate of positive against negative examples is 1 to 100

Direction 2:
Various feature types

continuous、binary、categorical、multi-value categorical、temporal.

Direction 3:
Lifelong setting

successive test batches chronologically ordered.

Direction 4:
Concept drift

the data distribution is slowly changing over time.

PART 02

Solution

1: Framework

2: Data Processing

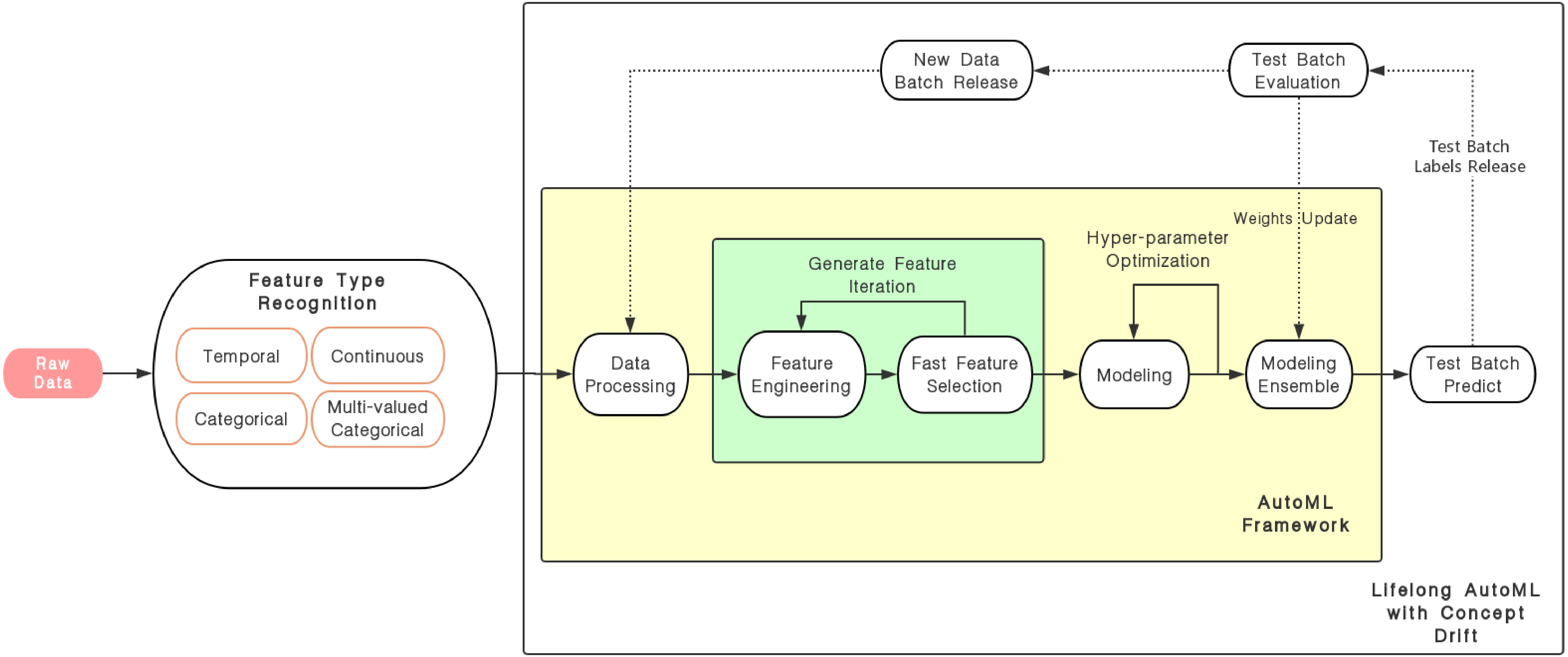
3: Feature Engineering

4: Feature Selection

5: Modeling

6: Model Ensemble

01 Framework



02 Data Processing

Traditional data cleaning

Duplicate data and constant data

Massive data analysis

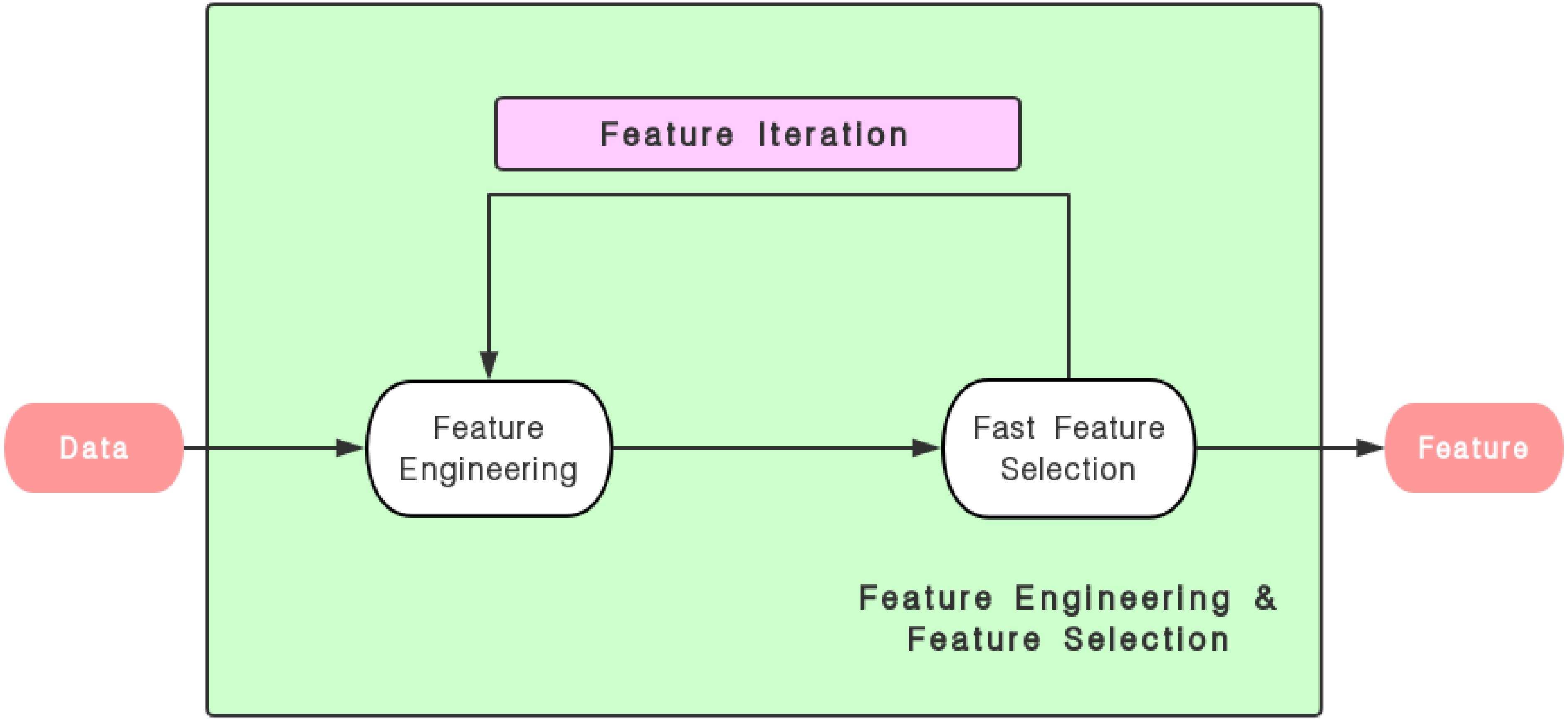
leverage prior knowledge

Other strategies

such as random sampling for computational efficiency

03

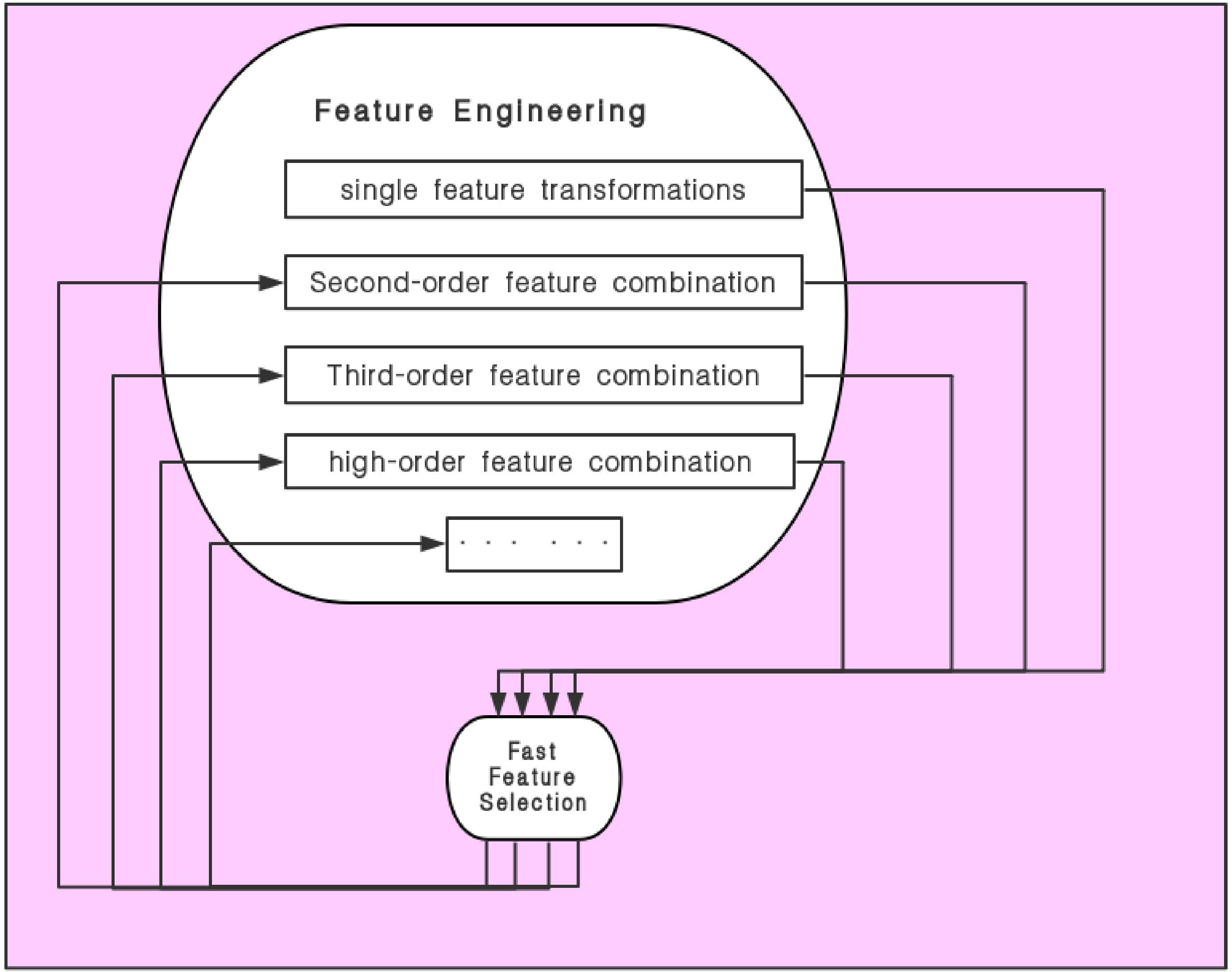
Feature Engineering& Feature Selection



04

Feature Engineering & Selection

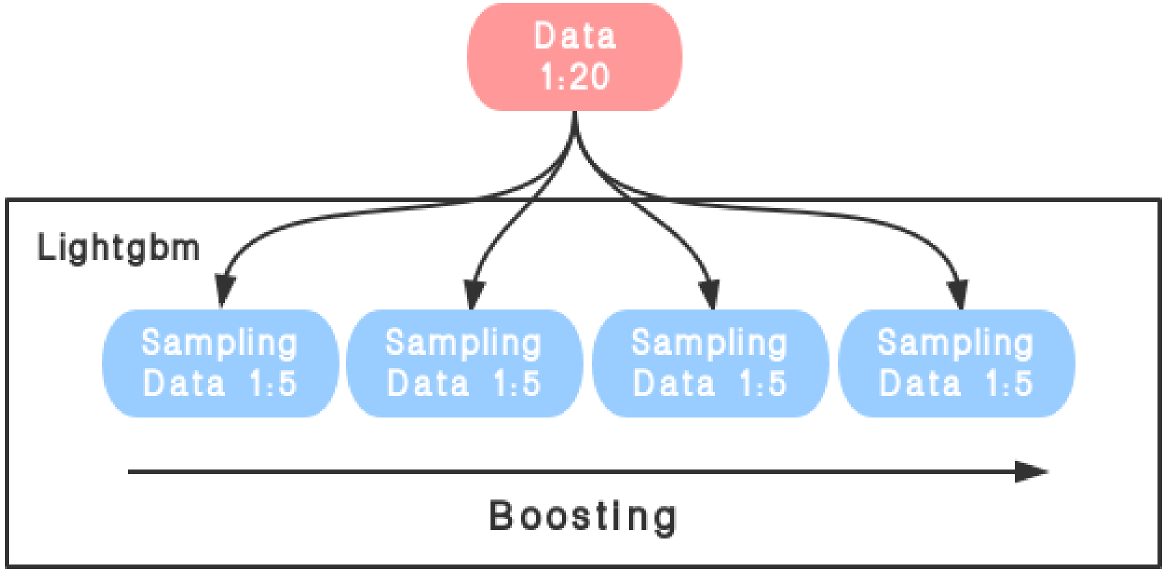
Generate Feature Iteration



05
Modeling

1:Base Model : LightGBM

2:Address class-imbalance issue

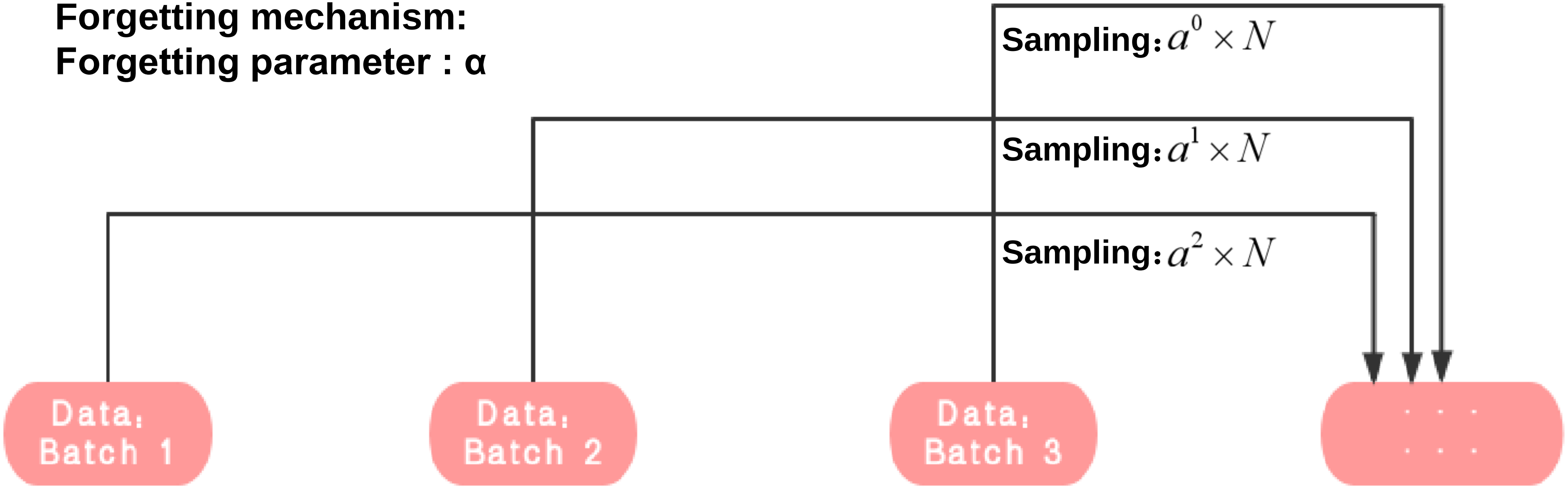


3:Address concept drift issue

06 Address concept drift

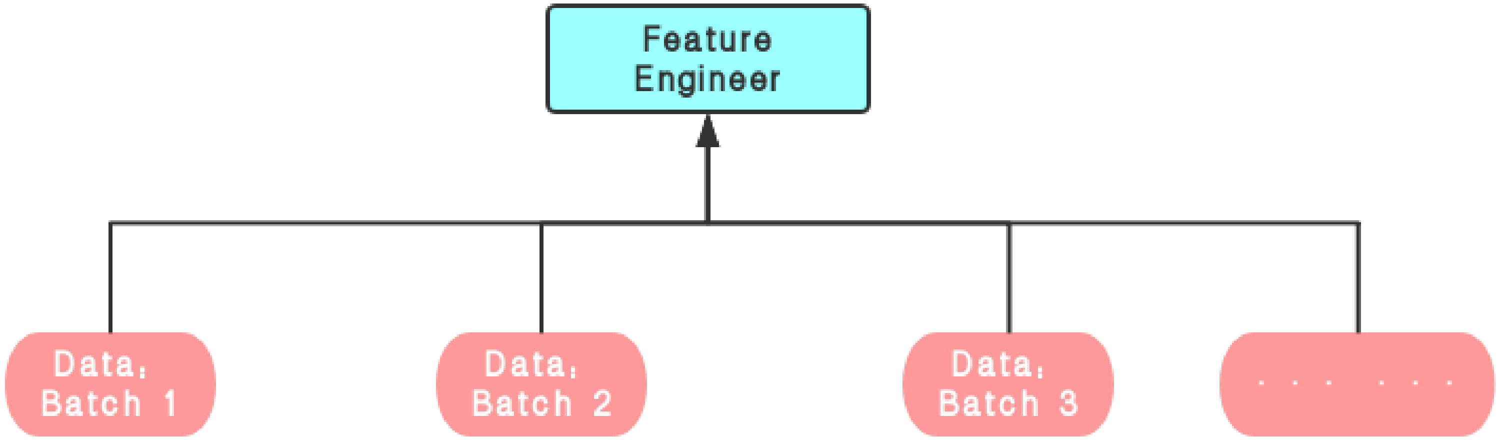
1:Data merge

Forgetting mechanism:
Forgetting parameter : α



07
Address concept drift

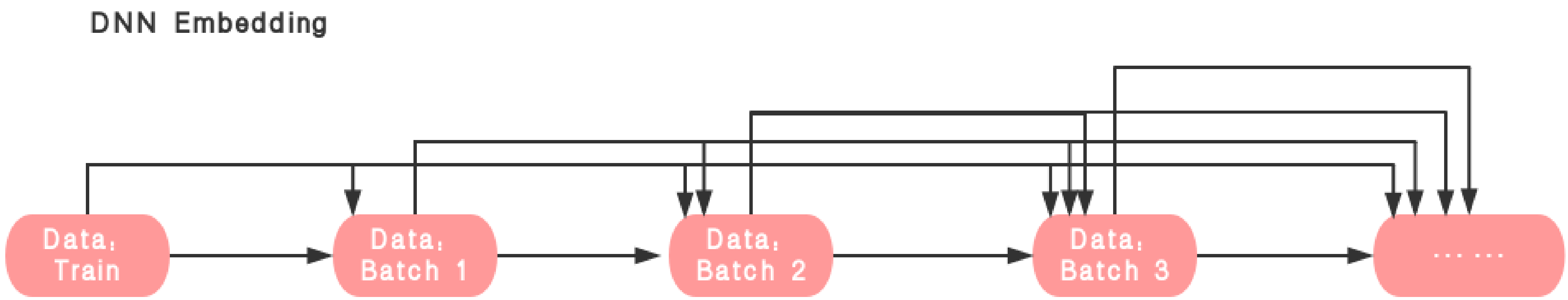
2:Anti concept drift feature



E.g. batch id, batch numerical drift, etc.

08 Address concept drift

3:Feature Embedding



Model Hyper-parameter Search

- Leverge some prior knowledge to reduce the search range of the parameters.

- Compromise between grid search and random search.

10

Modeling Ensemble

Model parameter diversity

Data diversity

Feature diversity

PART 03

Summary

01

Summary

- extend to various feature types.
- data merge method
- pre-training feature embedding
- multiple down-sampling for model boosting
- time and space optimization

THANKS