СБЕР

Tabular AutoML

Дмитрий Симаков

05/2021

Что такое AutoML?

Классификация AutoML, I

6 уровней AutoML от Bojan Tunguz*:

1. No automation. You code your own ML algorithms. From scratch. In C++.
2. Use of high-level algorithm APIs. Sklearn, Keras, Pandas, H2O, XGBoost, etc.
3. Automatic hyperparameter tuning and ensembling. Basic model selection.
4. Automatic (technical) feature engineering and feature selection, technical data augmentation, GUI.

Мы здесь (если хватает ресурсов, обычно нет)

5. Automatic domain and problem specific feature engineering, data augmentation, and data integration.
6. Full ML Automation. Ability to come up with super-human strategies for solving hard ML problems without any input or guidance. Fully conversational interaction with the human user.

Из статьи AutoSklearn…

- No single machine learning method performs best on all datasets
- Some machine learning methods (e.g., non-linear SVMs) crucially rely on hyperparameter optimization

Сюда же относятся:

- Neural Architecture Search (NAS)
- Простой Meta Learning
- Стратегии обучения + управление бюджетом

Перспективные направления:

- Продвинутое Meta Learning (больше мета датасетов)
- Domain Specific Language **
- Базы знаний (Графы?)

https://medium.com/@tunguz/six-levels-of-auto-ml-a277aa1f0f38

** some fun examples (Kaggle ARC)

Классификация AutoML, II

	Проприетарные	Открытые
Industrial	TAZI Google H2O Driverless AI DotData DataRobot	H2O AI Mindsdb LightAutoML MLJar AutoGluon
Research		TPOT Oboe AutoSklearn

По задаче:

- tabular
- DL
- CV
- NLP

По части пайплайна:

- Гиперпараметры
- Генерация признаков
- Выбор модели
- General

AutoML Survey:

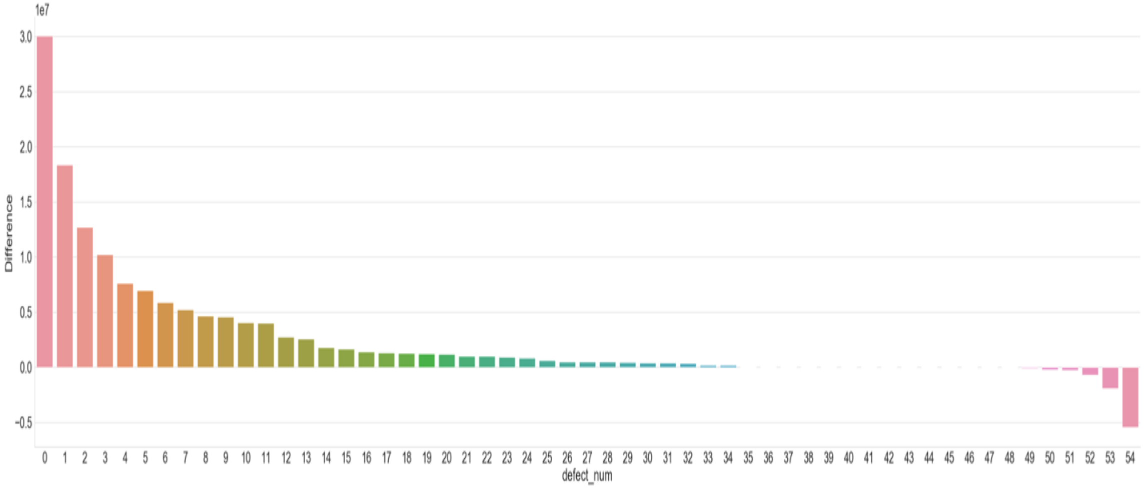
- https://arxiv.org/abs/1908.00709
- https://github.com/hibayesian/awesome-automl-papers

AutoML Benchmark (tabular):

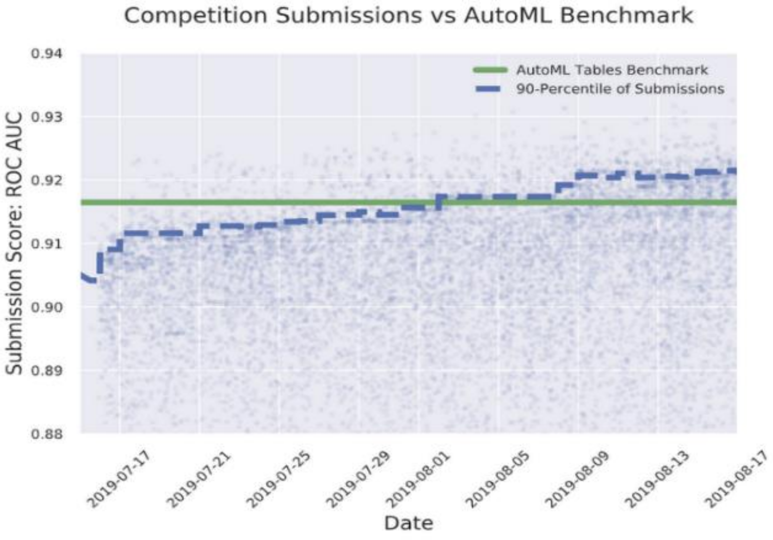
- https://openml.github.io/automlbenchmark/

Зачем нужен AutoML?

Использование AutoML для решения бизнес задач



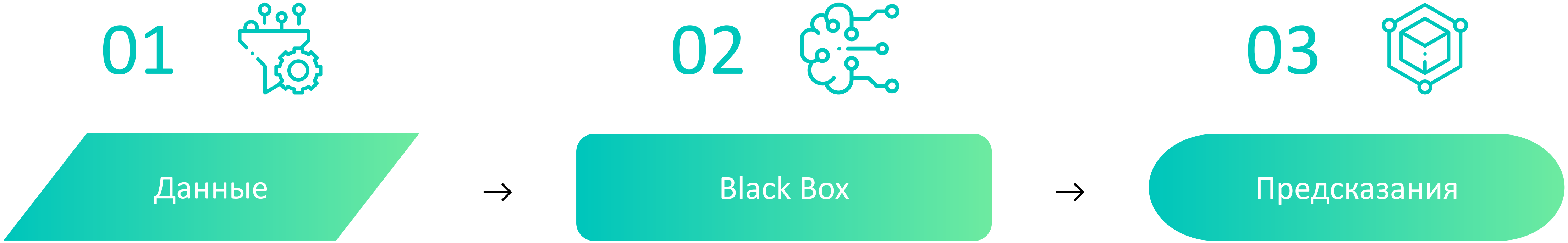
ML теперь выгоден там, где ранее не окупался



Автоматическое решение обходят только ТОП специалисты или для этого требуется время

Элементы AutoML.

Разработка моделей в парадигме AutoML, I



01 Данные → 02 Black Box → 03 Предсказания

Основной принцип as is:

- Нет предобработки
- Разные источники и форматы
- Структурированные и неструктурированные

Магия

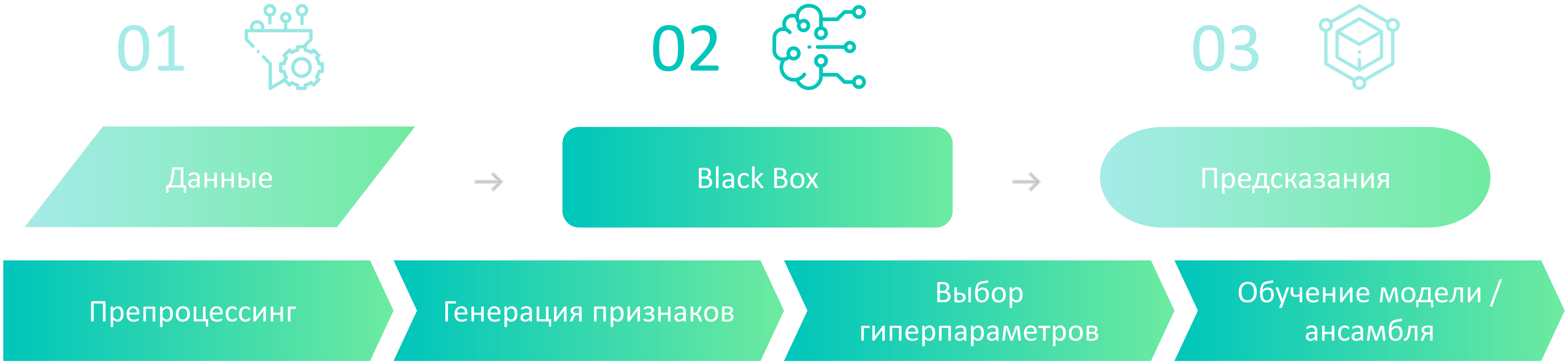
Дополнительно сюда передается задача и параметры (опционально)

Любого формата, в зависимости от задачи:

- Что и целевая переменная
- Набор распределений
- Кластеры
- Сгенерированные данные

Отчет, интерпретация, сохраненная модель

Разработка моделей в парадигме AutoML, II



01

Данные

→

02

Black Box

→

03

Предсказания

Препроцессинг

Генерация признаков

Выбор гиперпараметров

Обучение модели / ансамбля

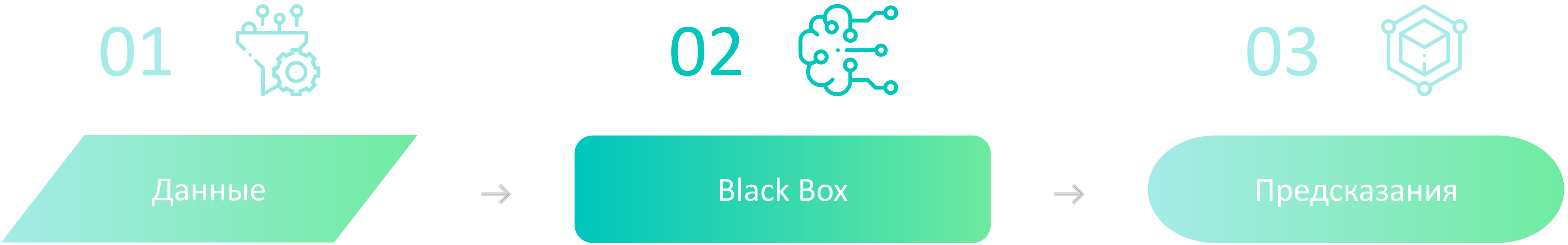
- Поиск взаимосвязей входов
- Определение типов, их препроцессинг
- Детекция проблем в данных

- Типичные для всех задач
- Типичные для домена
- Под конкретные данные
- Поиск мусорных и отбор

- Для генерации признаков
- Подбор или выбор модели
- Гиперпараметры модели

- Оптимизация целевой метрики – не только loss

Разработка моделей в парадигме AutoML, III



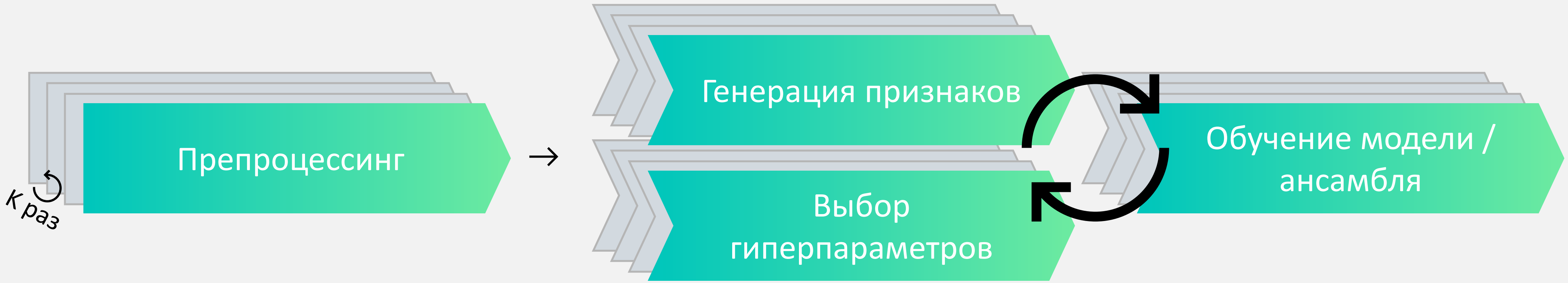
01 Данные →

02 Black Box →

03 Предсказания

Экспертная система:

- Менеджмент ресурсов
- Адаптация под домен
- Метамодели / правила



Препроцессинг →

Генерация признаков

Выбор гиперпараметров

Обучение модели / ансамбля

Элементы AutoML: Препроцессинг

- Мердж датасетов.
- Типизация ролей.
- Обработка категорий, дат, текстов, картинок.

https://arxiv.org/abs/2009.03358 - поиск взаимосвязей в таблице.
https://arxiv.org/abs/2006.14806 - семантическая типизация ролей при помощи Transformer.
https://adalabucsd.github.io/papers/TR_2019_SortingHat.pdf - определение типа переменной (категория / число).
https://arxiv.org/abs/1809.01604 - автоматическое объединение датасетов.

Элементы AutoML: Генерация признаков

https://github.com/alteryx/featuretools – автоматическая генерация признаков из нескольких датасетах.
https://github.com/ealcobaca/pymfe - автоматическая генерация мета-признаков.
https://arxiv.org/abs/1904.12857 - генерация пересечений категориальных признаков.
https://tsfresh.com – признаки TS.

Элементы AutoML: Подбор гиперпараметров, I

Назначение:
Подбор гиперпараметров модели
Оптимизация пайплайна

Как?
TPE, GP, …, любой алгоритм дискретной оптимизации

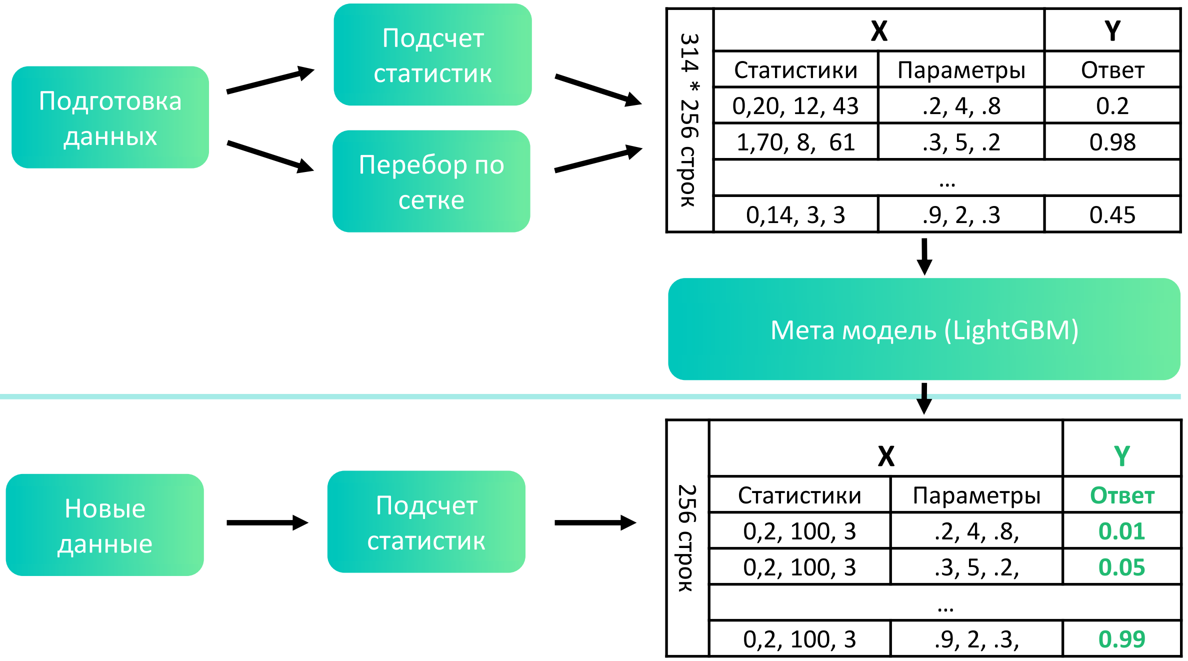
https://bbochallenge.com – соревнование к NIPS 2020
Virtual room page – решения.
AutoSklearn authors – 3rd place, Optuna – 6th.

https://github.com/quark0/darts - дифференцируемый NAS.
https://github.com/mit-han-lab/proxylessnas - быстрый NAS.
https://github.com/microsoft/FLAML - поиск лучшей модели.
https://arxiv.org/abs/2012.14905 - метаRNN для обучения NN.
https://github.com/kakaobrain/fast-autoaugment - автоматический подбор аугментаций.

Элементы AutoML: Подбор гиперпараметров, II

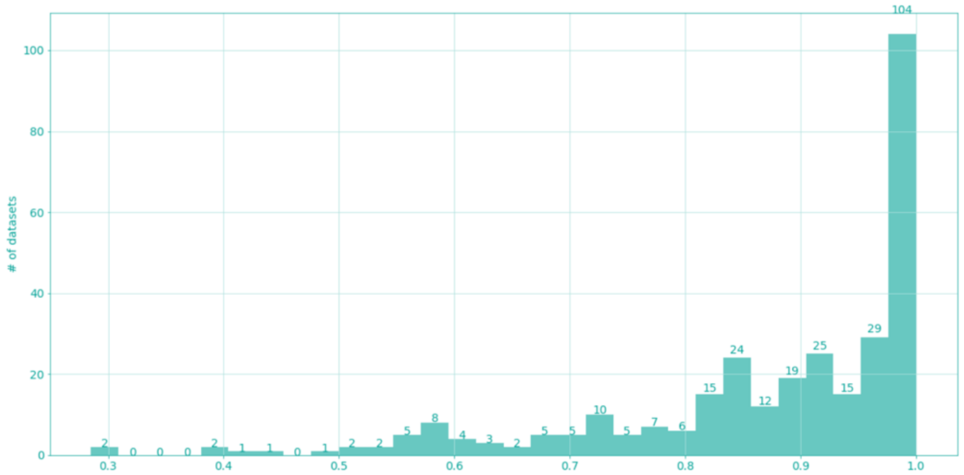
Метамодель предсказания оптимальных гиперпараметров

В трети случаев удается быстро получить качество модели на уровне 98% лучших результатов. В половине случаев – на уровне 90%.



X		Y
Статистики	Параметры	Ответ
0,20, 12, 43	.2, 4, .8	0.2
1,70, 8, 61	.3, 5, .2	0.98
…		
0,14, 3, 3	.9, 2, .3	0.45

X		Y
Статистики	Параметры	**Ответ**
0,2, 100, 3	.2, 4, .8,	**0.01**
0,2, 100, 3	.3, 5, .2,	**0.05**
…		
0,2, 100, 3	.9, 2, .3,	**0.99**



Подробнее https://www.youtube.com/watch?v=Q7SdMpKBass

Примеры существующих opensource решений.

AutoSklearn, 2015

AutoSklearn automatically considers past performance on similar datasets and constructs ensembles from the models evaluated during the optimization.

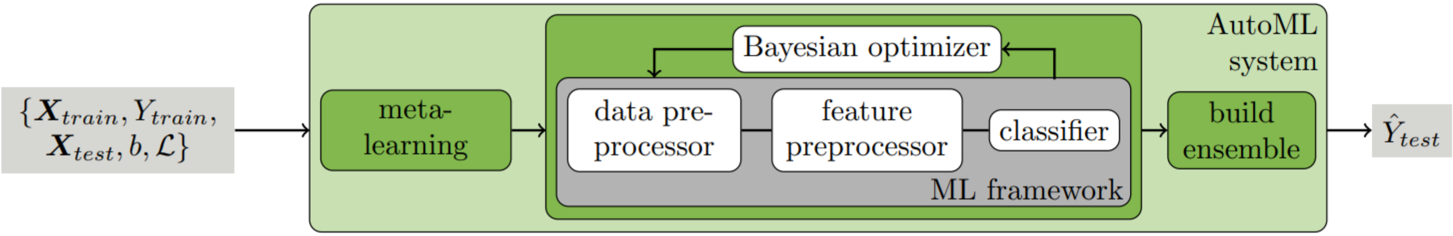


Figure 1: Our improved AutoML approach. We add two components to Bayesian hyperparameter optimization of an ML framework: meta-learning for initializing the Bayesian optimizer and automated ensemble construction from configurations evaluated during optimization.

Ансамбль: все итерации оптимизатора сохраняют модели и строят взвешенную комбинацию итеративным алгоритмом на основе холдаута.

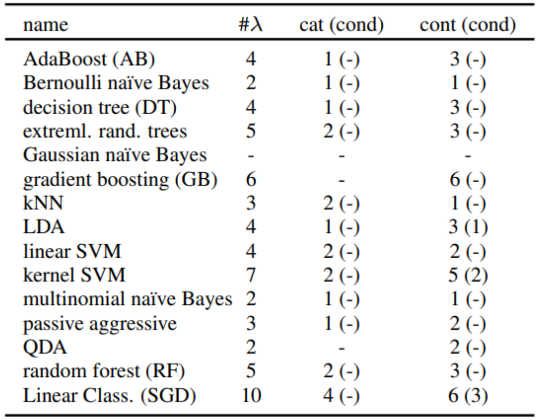
Git: https://github.com/automl/auto-sklearn, Site: https://www.automl.org/automl/auto-sklearn/; AS: https://proceedings.neurips.cc/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf

AutoSklearn, 2015

Задача **классификации**.

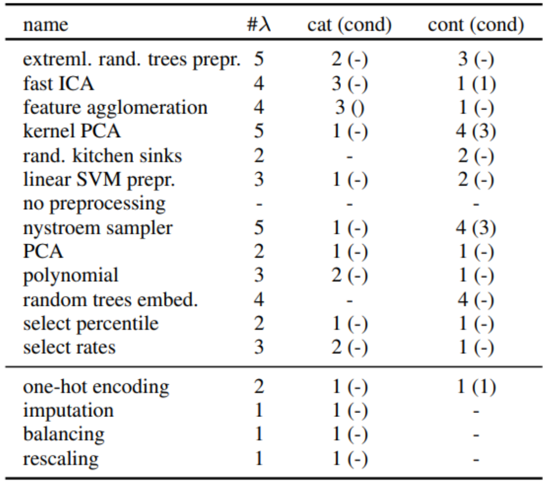
Датасеты: OpenML (140)

1. Meta model – инициализация гиперпараметров – пайплайн, модель, параметры.
2. На каждом датасете ищется оптимальный пайплайн (hyperopt).
3. Для новых данных – выбирается топ 25 кандидатов по L1 расстоянию по мета-признакам для инициализации запусков оптимизатора.



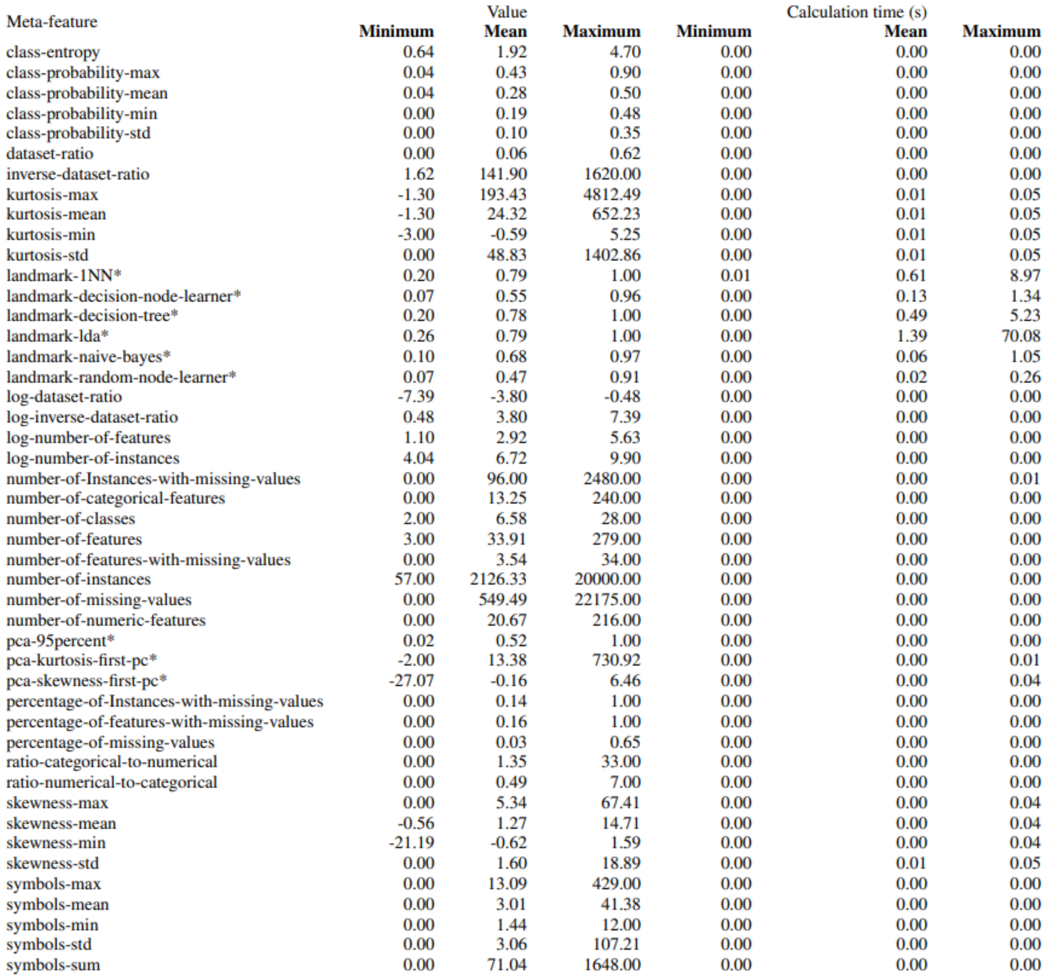
name	#λ	cat (cond)	cont (cond)
AdaBoost (AB)	4	1 (-)	3 (-)
Bernoulli naïve Bayes	2	1 (-)	1 (-)
decision tree (DT)	4	1 (-)	3 (-)
extreml. rand. trees	5	2 (-)	3 (-)
Gaussian naïve Bayes	-	-	-
gradient boosting (GB)	6	-	6 (-)
kNN	3	2 (-)	1 (-)
LDA	4	1 (-)	3 (1)
linear SVM	4	2 (-)	2 (-)
kernel SVM	7	2 (-)	5 (2)
multinomial naïve Bayes	2	1 (-)	1 (-)
passive aggressive	3	1 (-)	2 (-)
QDA	2	-	2 (-)
random forest (RF)	5	2 (-)	3 (-)
Linear Class. (SGD)	10	4 (-)	6 (3)

(a) classification algorithms



name	#λ	cat (cond)	cont (cond)
extreml. rand. trees prepr.	5	2 (-)	3 (-)
fast ICA	4	3 (-)	1 (1)
feature agglomeration	4	3 ()	1 (-)
kernel PCA	5	1 (-)	4 (3)
rand. kitchen sinks	2	-	2 (-)
linear SVM prepr.	3	1 (-)	2 (-)
no preprocessing	-	-	-
nystroem sampler	5	1 (-)	4 (3)
PCA	2	1 (-)	1 (-)
polynomial	3	2 (-)	1 (-)
random trees embed.	4	-	4 (-)
select percentile	2	1 (-)	1 (-)
select rates	3	2 (-)	1 (-)
one-hot encoding	2	1 (-)	1 (1)
imputation	1	1 (-)	-
balancing	1	1 (-)	-
rescaling	1	1 (-)	-

(b) preprocessing methods



Meta-feature		Value			Calculation time (s)	
	Minimum	**Mean**	**Maximum**	**Minimum**	**Mean**	**Maximum**
class-entropy	0.64	1.92	4.70	0.00	0.00	0.00
class-probability-max	0.04	0.43	0.90	0.00	0.00	0.00
class-probability-mean	0.04	0.28	0.50	0.00	0.00	0.00
class-probability-min	0.00	0.19	0.48	0.00	0.00	0.00
class-probability-std	0.00	0.10	0.35	0.00	0.00	0.00
dataset-ratio	0.00	0.06	0.62	0.00	0.00	0.00
inverse-dataset-ratio	1.62	141.90	1620.00	0.00	0.00	0.00
kurtosis-max	-1.30	193.43	4812.49	0.00	0.01	0.05
kurtosis-mean	-1.30	24.32	652.23	0.00	0.01	0.05
kurtosis-min	-3.00	-0.59	5.25	0.00	0.01	0.05
kurtosis-std	0.00	48.83	1402.86	0.00	0.01	0.05
landmark-1NN*	0.20	0.79	1.00	0.01	0.61	8.97
landmark-decision-node-learner*	0.07	0.55	0.96	0.00	0.13	1.34
landmark-decision-tree*	0.20	0.78	1.00	0.00	0.49	5.23
landmark-lda*	0.26	0.79	1.00	0.00	1.39	70.08
landmark-naive-bayes*	0.10	0.68	0.97	0.00	0.06	1.05
landmark-random-node-learner*	0.07	0.47	0.91	0.00	0.02	0.26
log-dataset-ratio	-7.39	-3.80	-0.48	0.00	0.00	0.00
log-inverse-dataset-ratio	0.48	3.80	7.39	0.00	0.00	0.00
log-number-of-features	1.10	2.92	5.63	0.00	0.00	0.00
log-number-of-instances	4.04	6.72	9.90	0.00	0.00	0.00
number-of-Instances-with-missing-values	0.00	96.00	2480.00	0.00	0.00	0.01
number-of-categorical-features	0.00	13.25	240.00	0.00	0.00	0.00
number-of-classes	2.00	6.58	28.00	0.00	0.00	0.00
number-of-features	3.00	33.91	279.00	0.00	0.00	0.00
number-of-features-with-missing-values	0.00	3.54	34.00	0.00	0.00	0.00
number-of-instances	57.00	2126.33	20000.00	0.00	0.00	0.00
number-of-missing-values	0.00	549.49	22175.00	0.00	0.00	0.00
number-of-numeric-features	0.00	20.67	216.00	0.00	0.00	0.00
pca-95percent*	0.02	0.52	1.00	0.00	0.00	0.00
pca-kurtosis-first-pc*	-2.00	13.38	730.92	0.00	0.00	0.01
pca-skewness-first-pc*	-27.07	-0.16	6.46	0.00	0.00	0.04
percentage-of-Instances-with-missing-values	0.00	0.14	1.00	0.00	0.00	0.00
percentage-of-features-with-missing-values	0.00	0.16	1.00	0.00	0.00	0.00
percentage-of-missing-values	0.00	0.03	0.65	0.00	0.00	0.00
ratio-categorical-to-numerical	0.00	1.35	33.00	0.00	0.00	0.00
ratio-numerical-to-categorical	0.00	0.49	7.00	0.00	0.00	0.00
skewness-max	0.00	5.34	67.41	0.00	0.00	0.04
skewness-mean	-0.56	1.27	14.71	0.00	0.00	0.04
skewness-min	-21.19	-0.62	1.59	0.00	0.00	0.04
skewness-std	0.00	1.60	18.89	0.00	0.01	0.05
symbols-max	0.00	13.09	429.00	0.00	0.00	0.00
symbols-mean	0.00	3.01	41.38	0.00	0.00	0.00
symbols-min	0.00	1.44	12.00	0.00	0.00	0.00
symbols-std	0.00	3.06	107.21	0.00	0.00	0.00
symbols-sum	0.00	71.04	1648.00	0.00	0.00	0.00

Table 1: List of implemented meta-features. Meta-features marked with an asterisks were only used to do the dataset clustering in Section 6

Git: https://github.com/automl/auto-sklearn, Site: https://www.automl.org/automl/auto-sklearn/; AS: https://proceedings.neurips.cc/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf

AutoSklearn 2.0, 2020

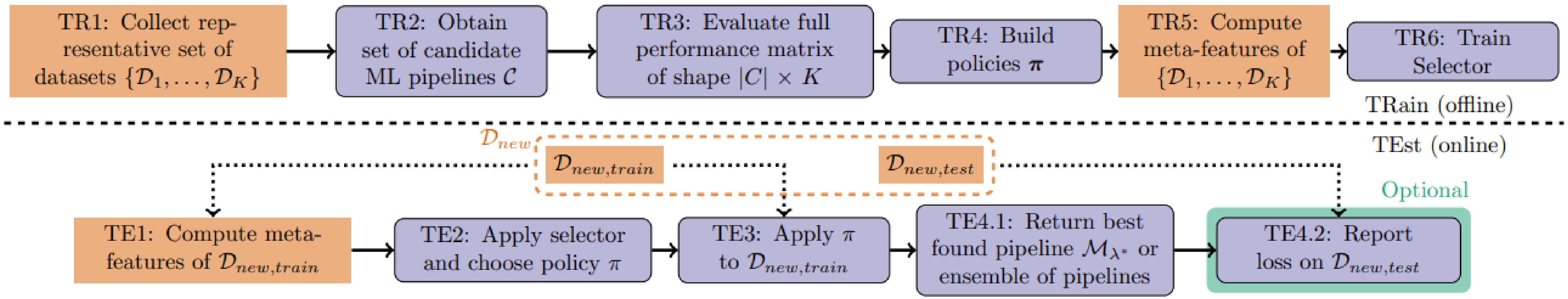
Задача **классификации**.

Датасеты: обучение OpenML (209), тест OpenML AutoML benchmark (39)

- Tuning: Portfolio and Successive Halving strategy: assigning higher budgets to more promising pipelines
- Reduce space to 42 HP with only iterative models.

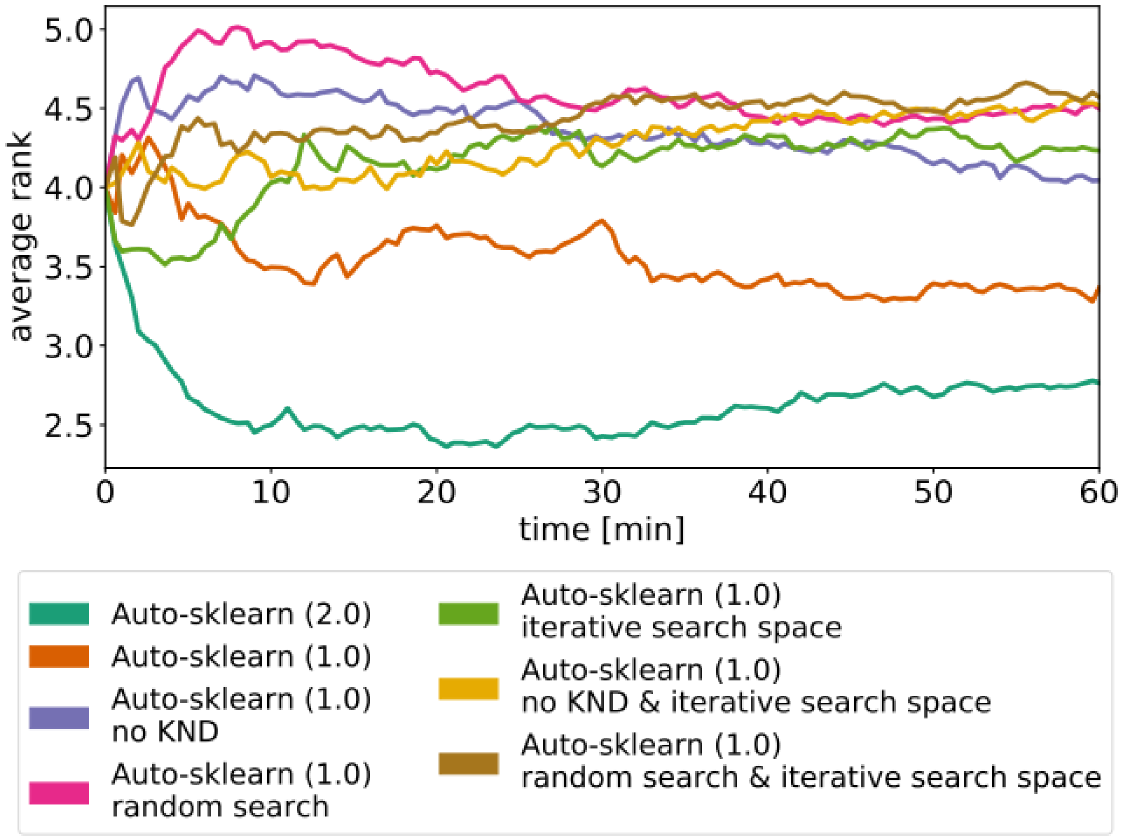
Portfolio: set of complementary pipelines that covers as many diverse datasets as possible and minimizes the risk of failure when facing a new task.

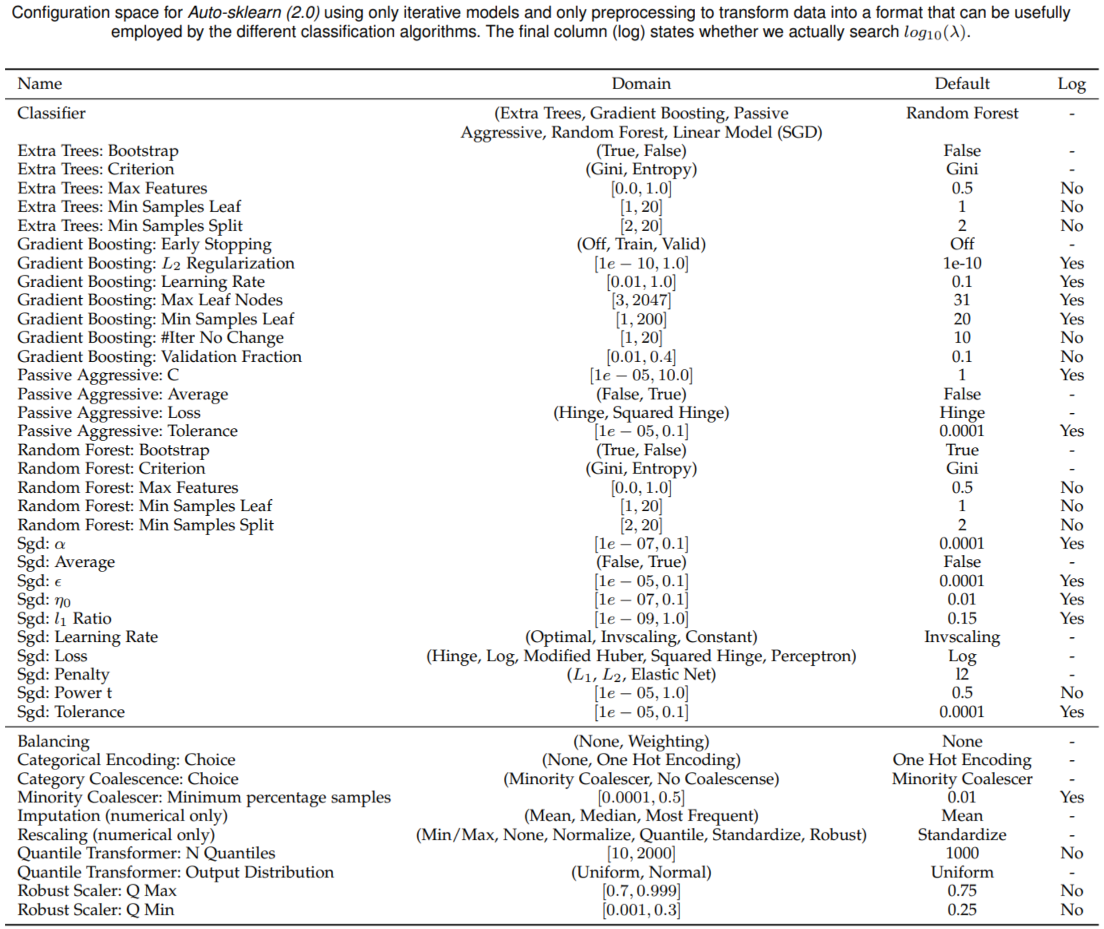
- Candidates – best pipeline for dataset from training metadataset
- Random Forest for pairwise candidates **ranking**


TR1: Collect representative set of datasets $\{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$
TR2: Obtain set of candidate ML pipelines $\mathcal{C}$
TR3: Evaluate full performance matrix of shape $|C| \times K$
TR4: Build policies π
TR5: Compute meta-features of $\{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$
TR6: Train Selector
TRain (offline)
TEst (online)
$\mathcal{D}_{new}$
$\mathcal{D}_{new,train}$
$\mathcal{D}_{new,test}$
Optional
TE1: Compute meta-features of $\mathcal{D}_{new,train}$
TE2: Apply selector and choose policy π
TE3: Apply π to $\mathcal{D}_{new,train}$
TE4.1: Return best found pipeline $\mathcal{M}_{\lambda^*}$ or ensemble of pipelines
TE4.2: Report loss on $\mathcal{D}_{new,test}$

AS2: https://arxiv.org/abs/2007.04074

AutoSklearn 2.0, 2020





Configuration space for *Auto-sklearn (2.0)* using only iterative models and only preprocessing to transform data into a format that can be usefully employed by the different classification algorithms. The final column (log) states whether we actually search $log_{10}(\lambda)$.

Name	Domain	Default	Log
Classifier	(Extra Trees, Gradient Boosting, Passive Aggressive, Random Forest, Linear Model (SGD)	Random Forest	-
Extra Trees: Bootstrap	(True, False)	False	-
Extra Trees: Criterion	(Gini, Entropy)	Gini	-
Extra Trees: Max Features	$[0.0, 1.0]$	0.5	No
Extra Trees: Min Samples Leaf	$[1, 20]$	1	No
Extra Trees: Min Samples Split	$[2, 20]$	2	No
Gradient Boosting: Early Stopping	(Off, Train, Valid)	Off	-
Gradient Boosting: L_2 Regularization	$[1e-10, 1.0]$	1e-10	Yes
Gradient Boosting: Learning Rate	$[0.01, 1.0]$	0.1	Yes
Gradient Boosting: Max Leaf Nodes	$[3, 2047]$	31	Yes
Gradient Boosting: Min Samples Leaf	$[1, 200]$	20	Yes
Gradient Boosting: #Iter No Change	$[1, 20]$	10	No
Gradient Boosting: Validation Fraction	$[0.01, 0.4]$	0.1	No
Passive Aggressive: C	$[1e-05, 10.0]$	1	Yes
Passive Aggressive: Average	(False, True)	False	-
Passive Aggressive: Loss	(Hinge, Squared Hinge)	Hinge	-
Passive Aggressive: Tolerance	$[1e-05, 0.1]$	0.0001	Yes
Random Forest: Bootstrap	(True, False)	True	-
Random Forest: Criterion	(Gini, Entropy)	Gini	-
Random Forest: Max Features	$[0.0, 1.0]$	0.5	No
Random Forest: Min Samples Leaf	$[1, 20]$	1	No
Random Forest: Min Samples Split	$[2, 20]$	2	No
Sgd: α	$[1e-07, 0.1]$	0.0001	Yes
Sgd: Average	(False, True)	False	-
Sgd: ϵ	$[1e-05, 0.1]$	0.0001	Yes
Sgd: η_0	$[1e-07, 0.1]$	0.01	Yes
Sgd: l_1 Ratio	$[1e-09, 1.0]$	0.15	Yes
Sgd: Learning Rate	(Optimal, Invscaling, Constant)	Invscaling	-
Sgd: Loss	(Hinge, Log, Modified Huber, Squared Hinge, Perceptron)	Log	-
Sgd: Penalty	(L_1, L_2, Elastic Net)	l2	-
Sgd: Power t	$[1e-05, 1.0]$	0.5	No
Sgd: Tolerance	$[1e-05, 0.1]$	0.0001	Yes
Balancing	(None, Weighting)	None	-
Categorical Encoding: Choice	(None, One Hot Encoding)	One Hot Encoding	-
Category Coalescence: Choice	(Minority Coalescer, No Coalescense)	Minority Coalescer	-
Minority Coalescer: Minimum percentage samples	$[0.0001, 0.5]$	0.01	Yes
Imputation (numerical only)	(Mean, Median, Most Frequent)	Mean	-
Rescaling (numerical only)	(Min/Max, None, Normalize, Quantile, Standardize, Robust)	Standardize	-
Quantile Transformer: N Quantiles	$[10, 2000]$	1000	No
Quantile Transformer: Output Distribution	(Uniform, Normal)	Uniform	-
Robust Scaler: Q Max	$[0.7, 0.999]$	0.75	No
Robust Scaler: Q Min	$[0.001, 0.3]$	0.25	No

AS2: https://arxiv.org/abs/2007.04074

Oboe, 2019

I. **Oboe is** a collaborative filtering method for time-constrained model selection and hyperparameter tuning using **meta learning.**

Classification

Datasets: UCI + OpenML, no missing values, 150 < n < 10k.

II.

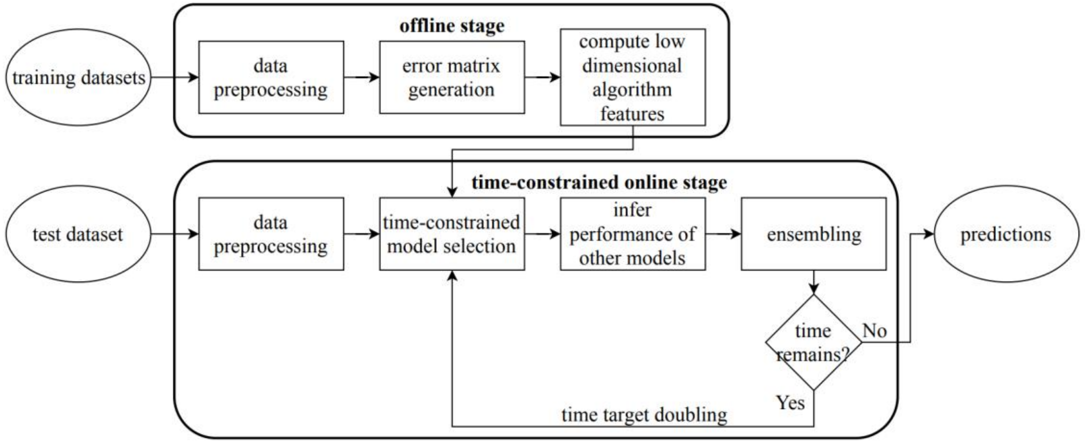
Feature pipeline:

- OHE categories
- No missings
- Standard Scaler

Meta models:

- Runtime: polynomial regression (up to 3)
- Choose models: QR algorithm, time constraints
- Model performance CF: PCA + OLS

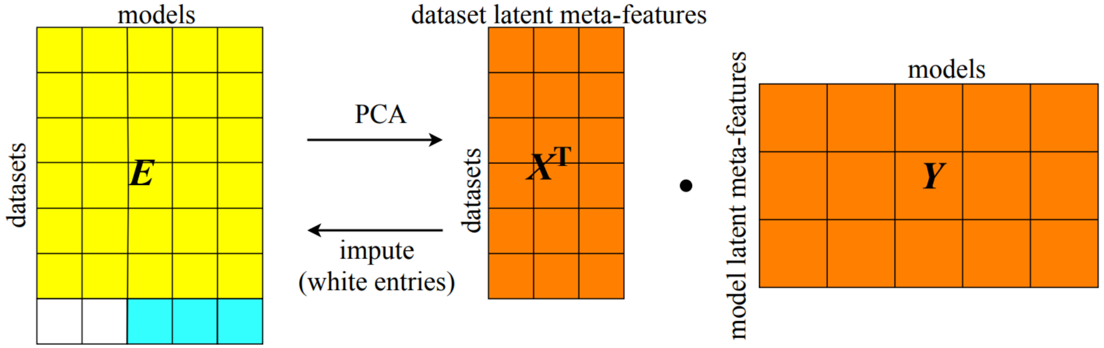
III. Pipeline:



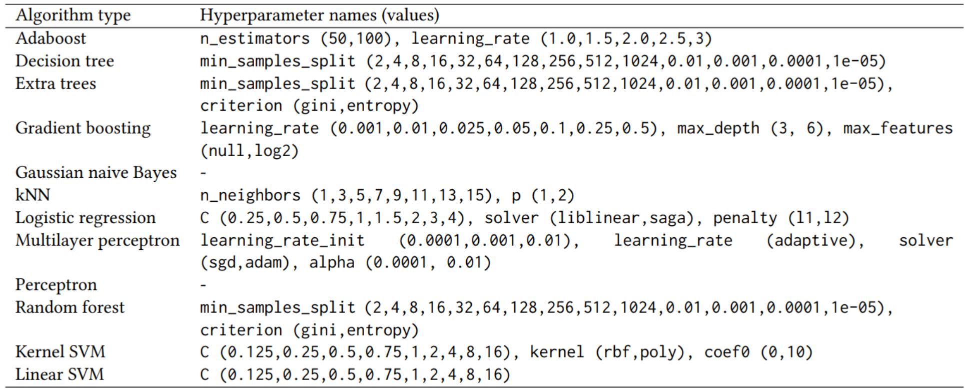
IV. Collaborate filtering part:

Error matrix (empirical) – Datasets, Models x Params

Balanced error rate – macro (FP + FN) / 2, KFold.



V. Models and hyperparameters:



Algorithm type	Hyperparameter names (values)
Adaboost	n_estimators (50,100), learning_rate (1.0,1.5,2.0,2.5,3)
Decision tree	min_samples_split (2,4,8,16,32,64,128,256,512,1024,0.01,0.001,0.0001,1e-05)
Extra trees	min_samples_split (2,4,8,16,32,64,128,256,512,1024,0.01,0.001,0.0001,1e-05), criterion (gini,entropy)
Gradient boosting	learning_rate (0.001,0.01,0.025,0.05,0.1,0.25,0.5), max_depth (3, 6), max_features (null,log2)
Gaussian naive Bayes	-
kNN	n_neighbors (1,3,5,7,9,11,13,15), p (1,2)
Logistic regression	C (0.25,0.5,0.75,1,1.5,2,3,4), solver (liblinear,saga), penalty (l1,l2)
Multilayer perceptron	learning_rate_init (0.0001,0.001,0.01), learning_rate (adaptive), solver (sgd,adam), alpha (0.0001, 0.01)
Perceptron	-
Random forest	min_samples_split (2,4,8,16,32,64,128,256,512,1024,0.01,0.001,0.0001,1e-05), criterion (gini,entropy)
Kernel SVM	C (0.125,0.25,0.5,0.75,1,2,4,8,16), kernel (rbf,poly), coef0 (0,10)
Linear SVM	C (0.125,0.25,0.5,0.75,1,2,4,8,16)

Git: https://github.com/udellgroup/oboe , Oboe: https://arxiv.org/abs/1808.03233

TensorOboe, 2020

TensorOboe is automated system to design a supervised learning pipeline. It uses low rank tensor decomposition as a surrogate model for efficient pipeline search.

Classification

Datasets: UCI (50) + OpenML (215)

Meta models:

- Runtime: polynomial regression (up to 3)
- Choose models: Greedy algorithm
- Model performance CF: EM-Tucker algorithm

Single ML Pipeline and CF:

Error matrix (empirical) – Dataset, Imputer, Encoder, Standartizer, dim. Reducer, Estimater – 6 dimensions.

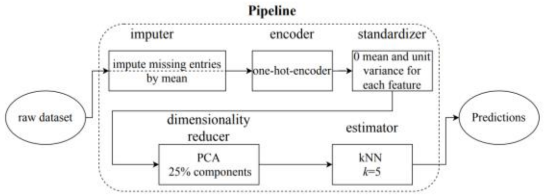


Figure 1: An example pipeline.

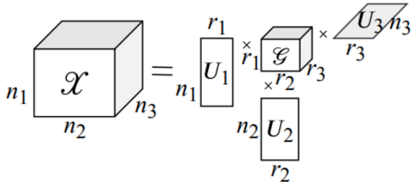
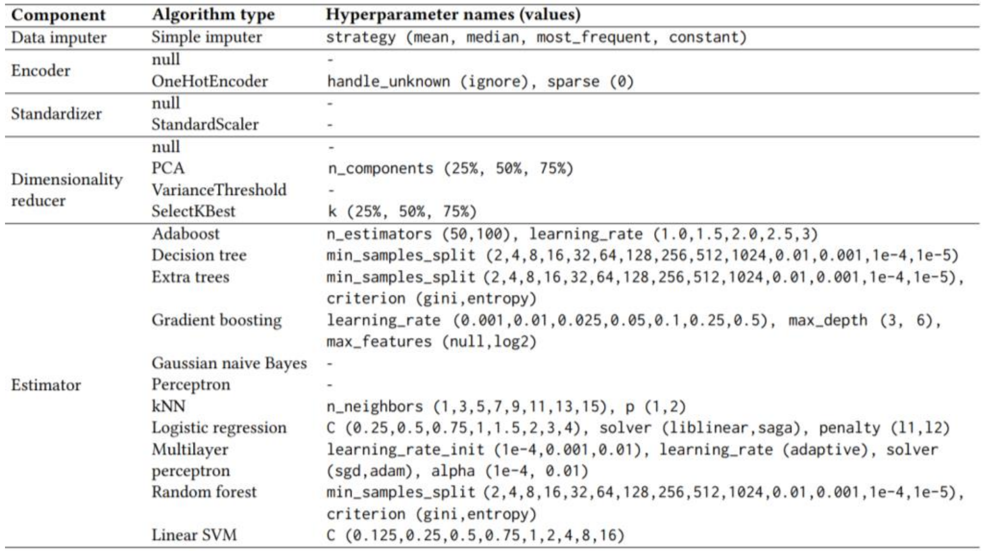


Figure 3: Tucker decomposition on an order-3 tensor.

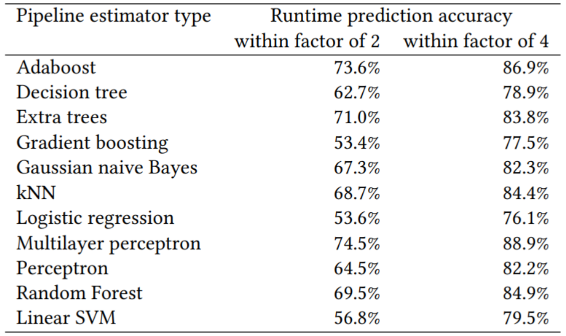
IV. Pipelines elements:



Component	Algorithm type	Hyperparameter names (values)
Data imputer	Simple imputer	`strategy (mean, median, most_frequent, constant)`
Encoder	null	-
	OneHotEncoder	`handle_unknown (ignore), sparse (0)`
Standardizer	null	-
	StandardScaler	-
Dimensionality reducer	null	-
	PCA	`n_components (25%, 50%, 75%)`
	VarianceThreshold	-
	SelectKBest	`k (25%, 50%, 75%)`
Estimator	Adaboost	`n_estimators (50,100), learning_rate (1.0,1.5,2.0,2.5,3)`
	Decision tree	`min_samples_split (2,4,8,16,32,64,128,256,512,1024,0.01,0.001,1e-4,1e-5)`
	Extra trees	`min_samples_split (2,4,8,16,32,64,128,256,512,1024,0.01,0.001,1e-4,1e-5), criterion (gini,entropy)`
	Gradient boosting	`learning_rate (0.001,0.01,0.025,0.05,0.1,0.25,0.5), max_depth (3, 6), max_features (null,log2)`
	Gaussian naive Bayes	-
	Perceptron	-
	kNN	`n_neighbors (1,3,5,7,9,11,13,15), p (1,2)`
	Logistic regression	`C (0.25,0.5,0.75,1,1.5,2,3,4), solver (liblinear,saga), penalty (l1,l2)`
	Multilayer perceptron	`learning_rate_init (1e-4,0.001,0.01), learning_rate (adaptive), solver (sgd,adam), alpha (1e-4, 0.01)`
	Random forest	`min_samples_split (2,4,8,16,32,64,128,256,512,1024,0.01,0.001,1e-4,1e-5), criterion (gini,entropy)`
	Linear SVM	`C (0.125,0.25,0.5,0.75,1,2,4,8,16)`

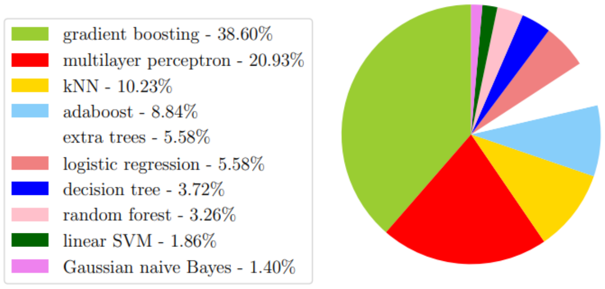
V. Some results:

Table 1: Runtime prediction accuracy on OpenML datasets



Pipeline estimator type	Runtime prediction accuracy	
	within factor of 2	within factor of 4
Adaboost	73.6%	86.9%
Decision tree	62.7%	78.9%
Extra trees	71.0%	83.8%
Gradient boosting	53.4%	77.5%
Gaussian naive Bayes	67.3%	82.3%
kNN	68.7%	84.4%
Logistic regression	53.6%	76.1%
Multilayer perceptron	74.5%	88.9%
Perceptron	64.5%	82.2%
Random Forest	69.5%	84.9%
Linear SVM	56.8%	79.5%

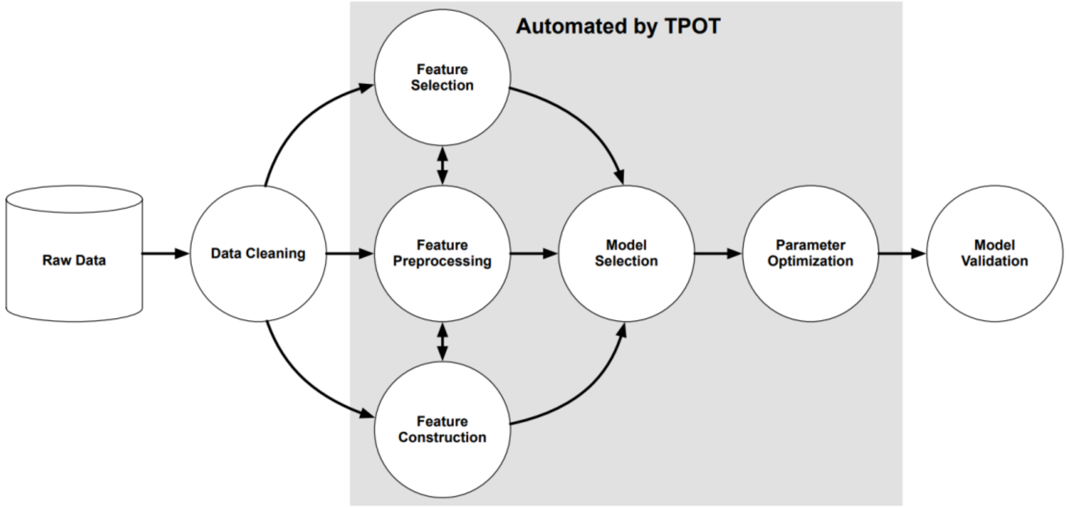
Best model:

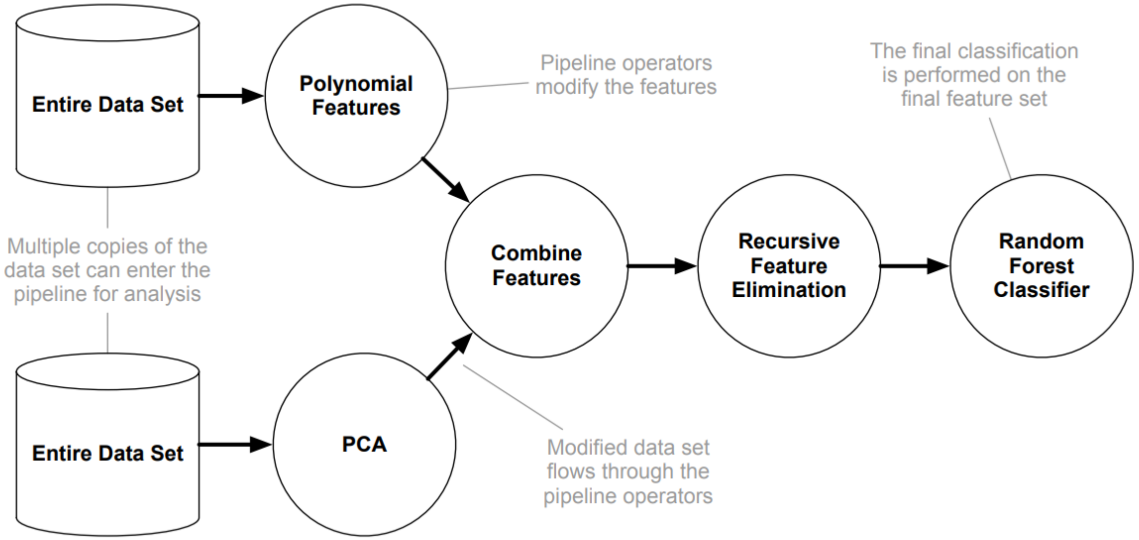


Git: https://github.com/udellgroup/oboe, TensorOboe: https://people.ece.cornell.edu/cy/_papers/tensor_oboe.pdf

TPOT, 2016, 2020

I. **TPOT (T**ree-based **P**ipeline **O**ptimization **T**ool**)**




Entire Data Set
Polynomial Features
Pipeline operators modify the features
The final classification is performed on the final feature set
Multiple copies of the data set can enter the pipeline for analysis
Combine Features
Recursive Feature Elimination
Random Forest Classifier
Entire Data Set
PCA
Modified data set flows through the pipeline operators

II. Pipeline optimization – genetic programming

- sequence of pipeline operators
- hyperparameters

Git: https://github.com/EpistasisLab/tpot, TPOT: https://dl.acm.org/doi/10.1145/2908812.2908918

AutoGluon, 2020

I. **AutoGluon** automates machine learning tasks enabling you to easily achieve strong predictive performance in your applications

II. **Key features**:

- Boostings + NN
- Stacking
- Repeated k-fold Bagging

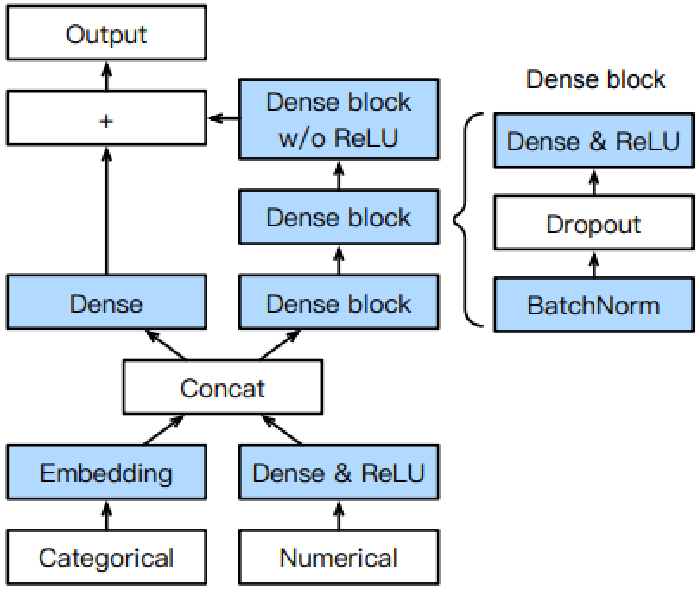


Figure 1. Architecture of AutoGluon's neural network for tabular data composed of numerical and categorical features. Layers with learnable parameters are marked as blue.

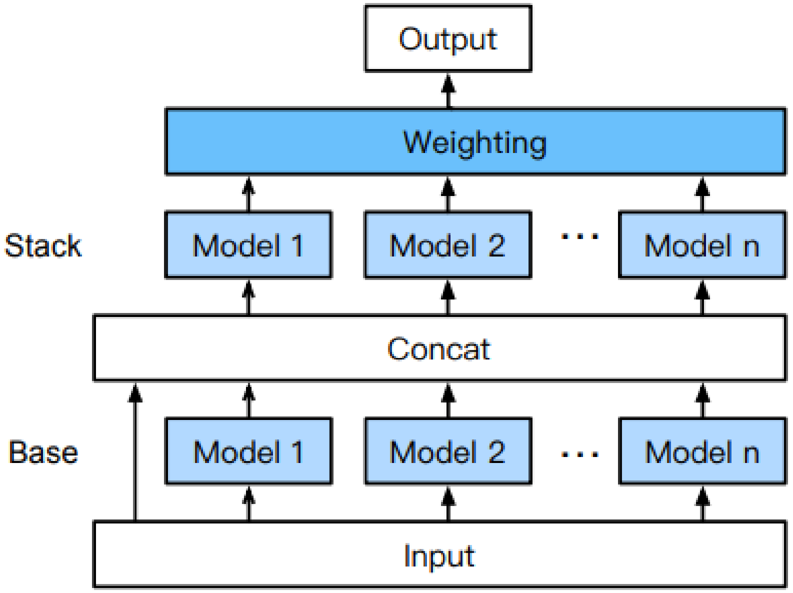


Figure 2. AutoGluon's multi-layer stacking strategy, shown here using two stacking layers and n types of base learners.

Git: https://github.com/awslabs/autogluon , AutoGluon: https://arxiv.org/abs/2003.06505

Анализ и выводы

Слабые пайплайны:
- Простые / неэффективные модели.
- Наивный препроцессинг и генерация признаков.

Мета-алгоритмы:
- Относительно маленький набор датасетов.
- Странные (ROC AUC 0.99; flattened ~MNIST)…
- …синтетические (make_classification – accuracy 100% с unsupervised GMM признаками)…
- …игрушечные (Boston) наборы данных.
- Широкая сетка параметров с (потенциально) бесполезными параметрами.

…или
- Вычислительно дорогая оптимизация параметров.

Модели выбираются индивидуально, а не по их вкладу в ансамбль.

Недо~ или Пере~ оцененная важность отдельных элементов пайплайна.

LightAutoML, 2020

LAMA – кросс-платформенный модульный фреймворк, включающий в себя подготовленные пайплайны данных для end-to-end решения ML проблем.

- Сильный бейзлайн: хорошо работает на большинстве датасетов.
- Быстрый: нет оптимизации пайплайнов.
- Production ready (с нюансами)

Имплементированные пайплайны:

✓ BlackBox preset

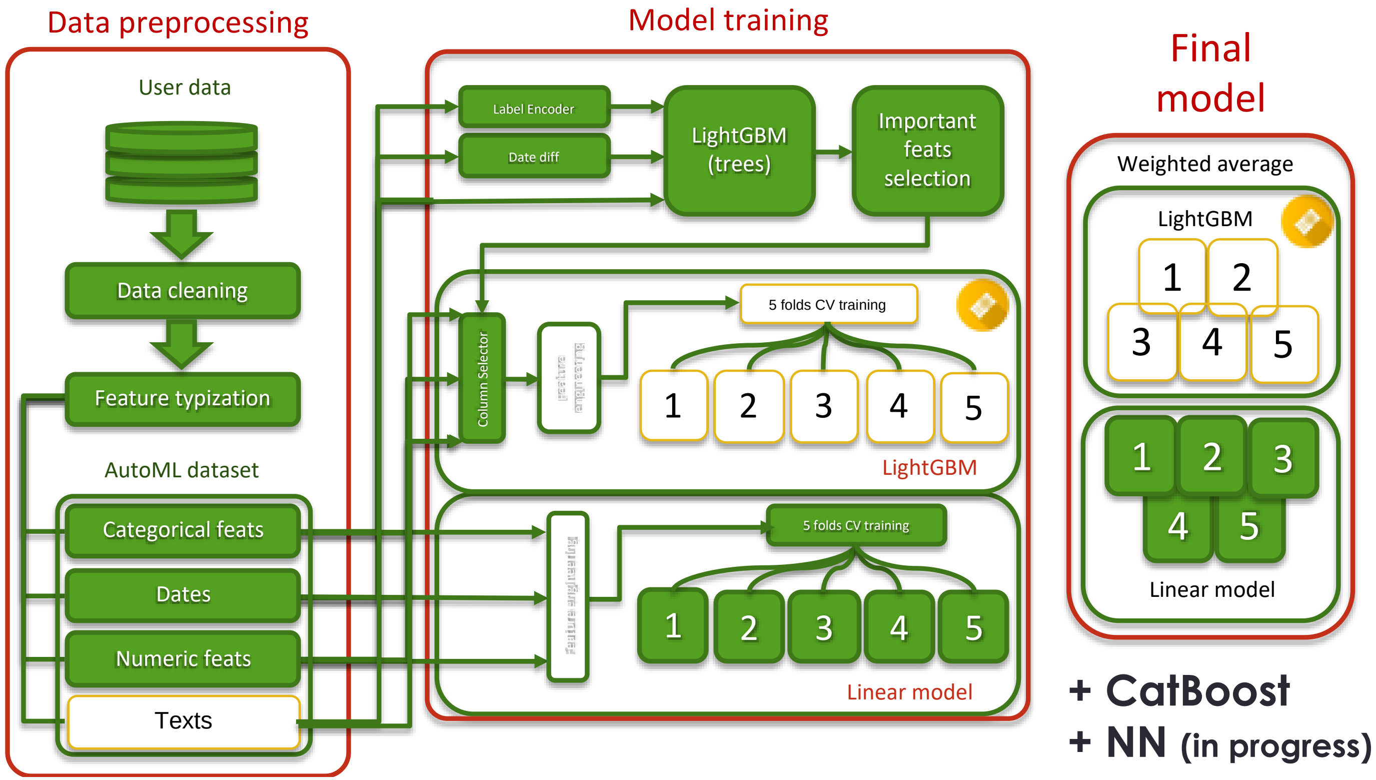
✓ WhiteBox preset

✓ NLP preset

✓ CV preset

Не только AutoML, но и конструктор пайплайнов.

+ отчеты из коробки



Data preprocessing

User data

Data cleaning

Feature typization

AutoML dataset

Categorical feats

Dates

Numeric feats

Texts

Model training

Label Encoder

Date diff

LightGBM (trees)

Important feats selection

Column Selector

5 folds CV training

1 2 3 4 5

LightGBM

5 folds CV training

1 2 3 4 5

Linear model

Final model

Weighted average

LightGBM

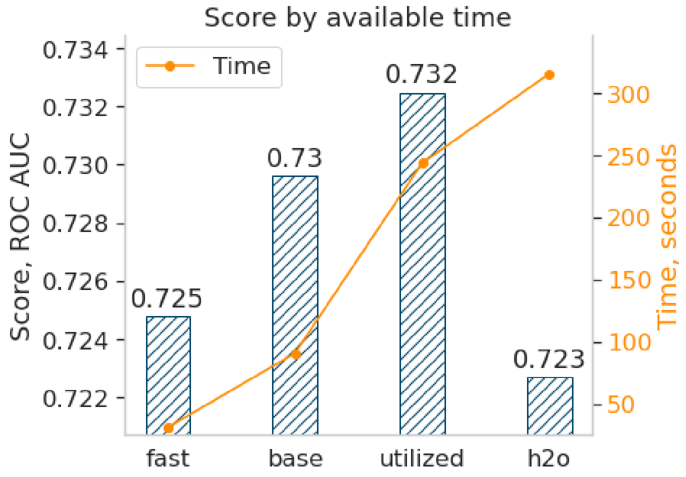
1 2 3 4 5

1 2 3 4 5

Linear model

+ CatBoost

+ NN (in progress)

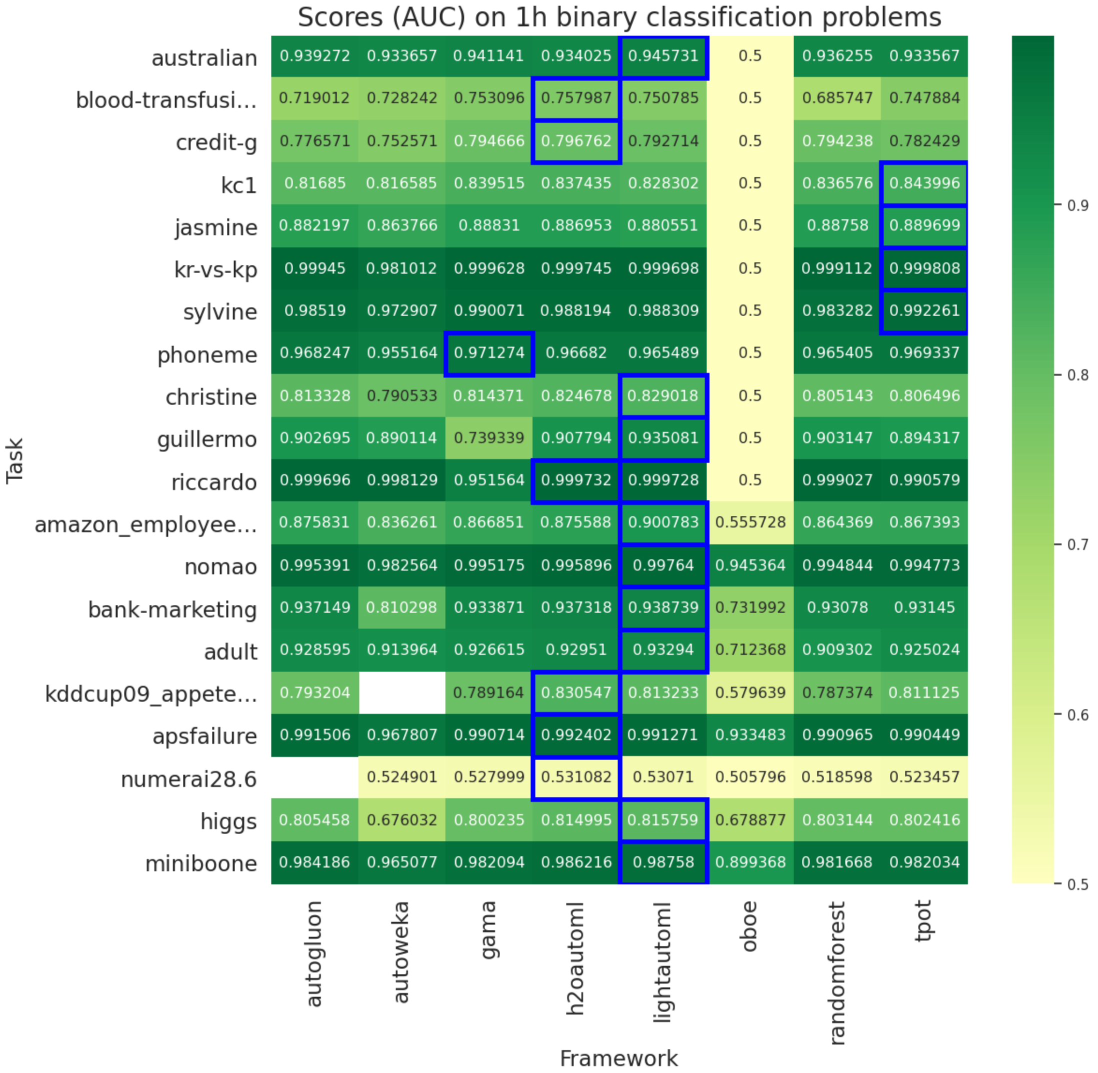


Score by available time

Git: https://github.com/sberbank-ai-lab/LightAutoML

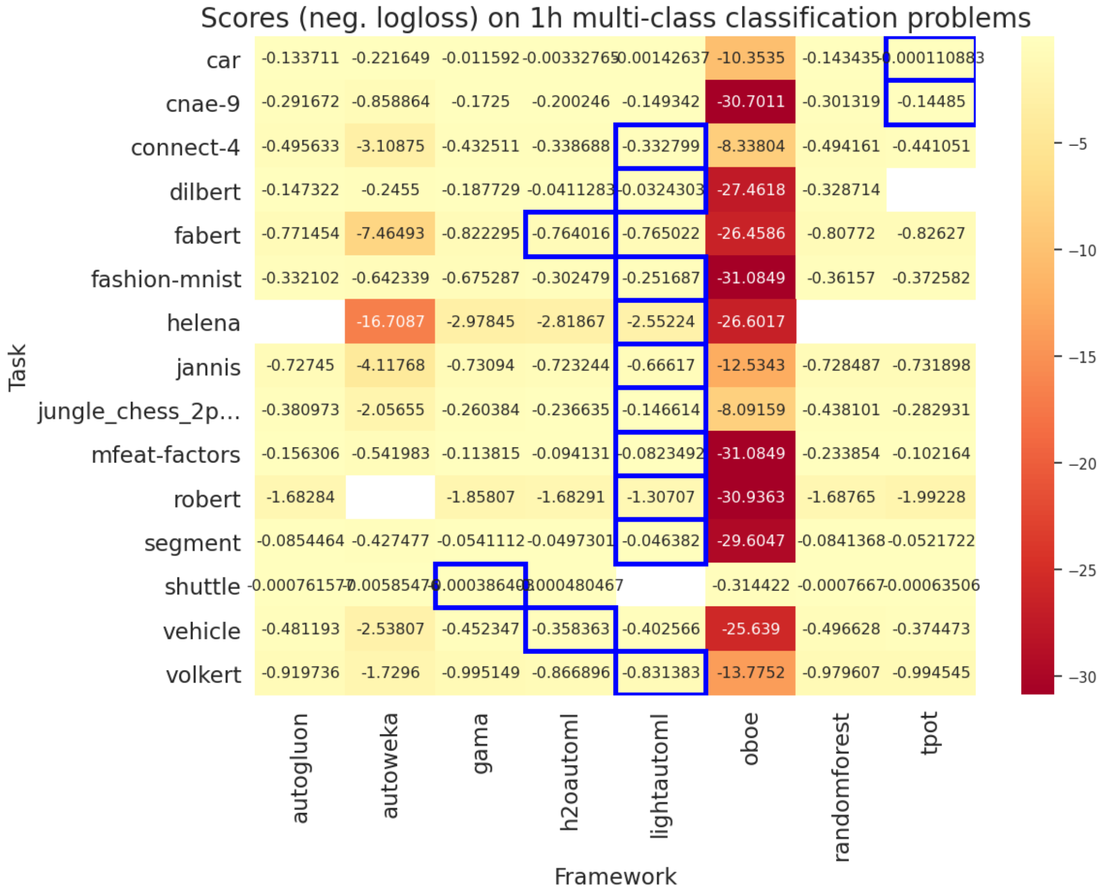
Бенчмарки.

Результаты OpenML, 1h8c



Scores (AUC) on 1h binary classification problems

Task	autogluon	autoweka	gama	h2oautoml	lightautoml	oboe	randomforest	tpot
australian	0.939272	0.933657	0.941141	0.934025	0.945731	0.5	0.936255	0.933567
blood-transfusi...	0.719012	0.728242	0.753096	0.757987	0.750785	0.5	0.685747	0.747884
credit-g	0.776571	0.752571	0.794666	0.796762	0.792714	0.5	0.794238	0.782429
kc1	0.81685	0.816585	0.839515	0.837435	0.828302	0.5	0.836576	0.843996
jasmine	0.882197	0.863766	0.88831	0.886953	0.880551	0.5	0.88758	0.889699
kr-vs-kp	0.99945	0.981012	0.999628	0.999745	0.999698	0.5	0.999112	0.999808
sylvine	0.98519	0.972907	0.990071	0.988194	0.988309	0.5	0.983282	0.992261
phoneme	0.968247	0.955164	0.971274	0.96682	0.965489	0.5	0.965405	0.969337
christine	0.813328	0.790533	0.814371	0.824678	0.829018	0.5	0.805143	0.806496
guillermo	0.902695	0.890114	0.739339	0.907794	0.935081	0.5	0.903147	0.894317
riccardo	0.999696	0.998129	0.951564	0.999732	0.999728	0.5	0.999027	0.990579
amazon_employee...	0.875831	0.836261	0.866851	0.875588	0.900783	0.555728	0.864369	0.867393
nomao	0.995391	0.982564	0.995175	0.995896	0.99764	0.945364	0.994844	0.994773
bank-marketing	0.937149	0.810298	0.933871	0.937318	0.938739	0.731992	0.93078	0.93145
adult	0.928595	0.913964	0.926615	0.92951	0.93294	0.712368	0.909302	0.925024
kddcup09_appete...	0.793204		0.789164	0.830547	0.813233	0.579639	0.787374	0.811125
apsfailure	0.991506	0.967807	0.990714	0.992402	0.991271	0.933483	0.990965	0.990449
numerai28.6		0.524901	0.527999	0.531082	0.53071	0.505796	0.518598	0.523457
higgs	0.805458	0.676032	0.800235	0.814995	0.815759	0.678877	0.803144	0.802416
miniboone	0.984186	0.965077	0.982094	0.986216	0.98758	0.899368	0.981668	0.982034



Scores (neg. logloss) on 1h multi-class classification problems

Task	autogluon	autoweka	gama	h2oautoml	lightautoml	oboe	randomforest	tpot
car	-0.133711	-0.221649	-0.011592	-0.00332765	-0.00142637	-10.3535	-0.143435	-0.000110883
cnae-9	-0.291672	-0.858864	-0.1725	-0.200246	-0.149342	-30.7011	-0.301319	-0.14485
connect-4	-0.495633	-3.10875	-0.432511	-0.338688	-0.332799	-8.33804	-0.494161	-0.441051
dilbert	-0.147322	-0.2455	-0.187729	-0.0411283	-0.0324303	-27.4618	-0.328714	
fabert	-0.771454	-7.46493	-0.822295	-0.764016	-0.765022	-26.4586	-0.80772	-0.82627
fashion-mnist	-0.332102	-0.642339	-0.675287	-0.302479	-0.251687	-31.0849	-0.36157	-0.372582
helena		-16.7087	-2.97845	-2.81867	-2.55224	-26.6017		
jannis	-0.72745	-4.11768	-0.73094	-0.723244	-0.66617	-12.5343	-0.728487	-0.731898
jungle_chess_2p...	-0.380973	-2.05655	-0.260384	-0.236635	-0.146614	-8.09159	-0.438101	-0.282931
mfeat-factors	-0.156306	-0.541983	-0.113815	-0.094131	-0.0823492	-31.0849	-0.233854	-0.102164
robert	-1.68284		-1.85807	-1.68291	-1.30707	-30.9363	-1.68765	-1.99228
segment	-0.0854464	-0.427477	-0.0541112	-0.0497301	-0.046382	-29.6047	-0.0841368	-0.0521722
shuttle	-0.000761577	-0.00585476	-0.00038640	-0.000480467		-0.314422	-0.0007667	-0.00063506
vehicle	-0.481193	-2.53807	-0.452347	-0.358363	-0.402566	-25.639	-0.496628	-0.374473
volkert	-0.919736	-1.7296	-0.995149	-0.866896	-0.831383	-13.7752	-0.979607	-0.994545

Соревнования по AutoDL

AutoCV, AutoNLP, AutoTS, AutoSignal… AutoDL

- **В основном про менеджмент времени и ресурсов**
- http://autodl.chalearn.org
- https://www.automl.org/wp-content/uploads/2020/07/AutoML_2020_paper_51.pdf

Прочее.

Интересные материалы

HungaBunga: https://github.com/ypeleg/HungaBunga

Забавный репозиторий: https://github.com/autogoal/autogoal

Материалы по Мета-обучению: https://sites.google.com/mit.edu/aaai2021metalearningtutorial

AutoML case studies:

https://research.aimultiple.com/automl-case-studies/

https://github.com/microsoft/forecasting AutoTS

https://github.com/THUMNLab/AutoGL AutoML on Graphs

https://github.com/AutoViML/AutoViz AutoEDA???

https://github.com/KartikChugh/Otto AutoML + чатбот

Материалы по LAMA

LightAutoML video guides:

- (Russian) LightAutoML hands-on tutorial in Kaggle Kernels (Alexander Ryzhkov)
- (English) LightAutoML framework general overview, benchmarks and advantages for business (Alexander Ryzhkov)
- (English) LightAutoML practical guide - ML pipeline presets overview (Dmitry Simakov)

Articles about LightAutoML:

- (English) LightAutoML vs Titanic: 80% accuracy in several lines of code (Medium)
- (English) Hands-On Python Guide to LightAutoML – An Automatic ML Model Creation Framework (Analytic Indian Mag)

Kaggle kernel examples of LightAutoML usage:

- Tabular Playground Series April 2021 competition solution
- Titanic competition solution (80% accuracy)
- Titanic **12-code-lines** competition solution (78% accuracy)
- House prices competition solution
- Natural Language Processing with Disaster Tweets solution
- Tabular Playground Series March 2021 competition solution
- Tabular Playground Series February 2021 competition solution
- Interpretable WhiteBox solution
- Custom ML pipeline elements inside existing ones

СБЕР

✉ Simakov.D.E@sberbank.ru

@DmitrySimakov

SBER AI LAB

Лаборатория Искусственного Интеллекта