AUTO-PYTORCH

Adam Frej, Łukasz Tomaszewski, Marcel Witas

Założenia ogólne

- Połączenie neural architecture search i optymalizacji hiperparametrów.
- Auto Deep Learning
- Dane tabelaryczne – klasyfikacja i regresja
- Klasyfikacja obrazów
- Popularność - 1.6k gwiazdek na GitHub, 55 cytowań artykułu

Opcje preprocessingu

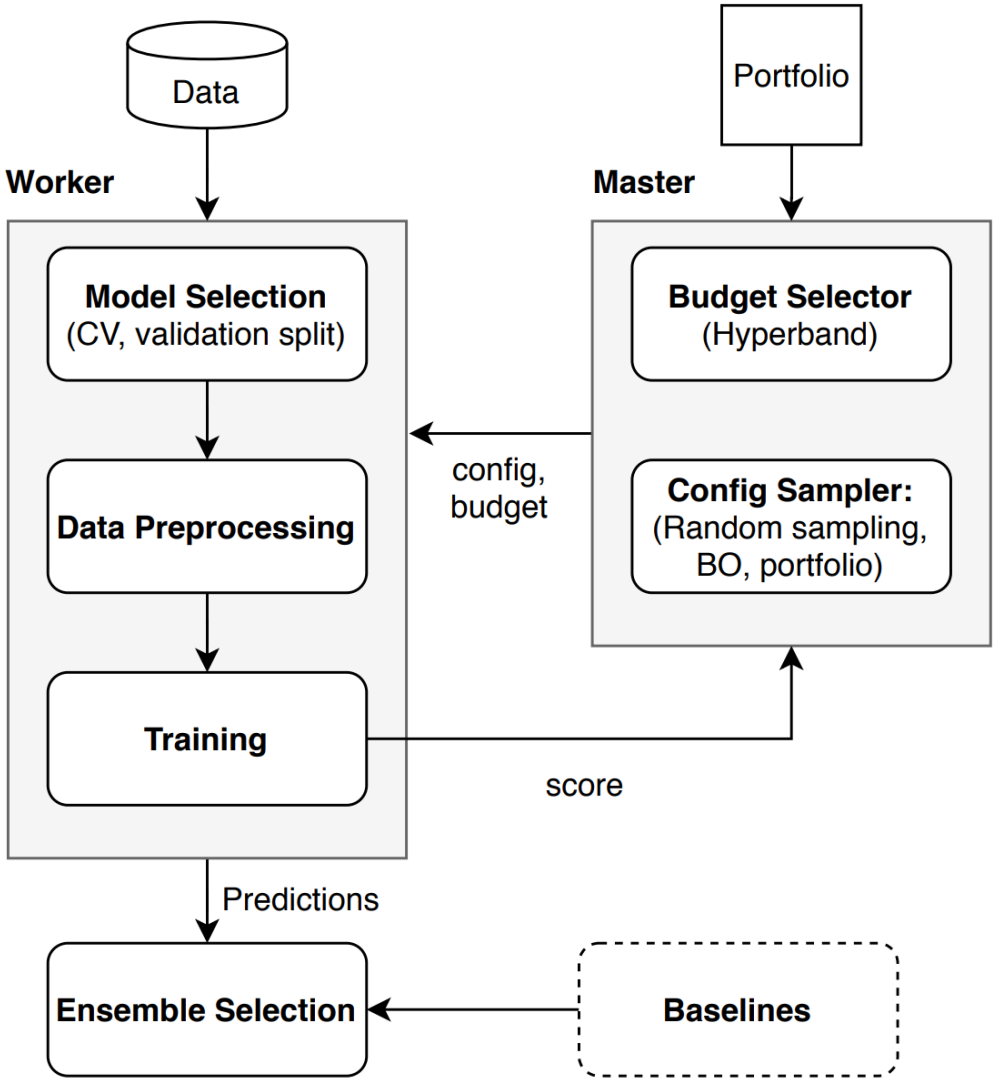
- **Imputation**
- **Encoding**
  - Choice of *OneHotEncoder* or no encoding.
- **Scaling**
  - Choice of *MinMaxScaler*, *Normalizer*, *StandardScaler* and no scaling.
- **Feature preprocessing**
  - Choice of *FastICA*, *KernelPCA*, *RandomKitchenSinks*, *Nystroem*, *PolynomialFeatures*, *PowerTransformer*, *TruncatedSVD*,

Dostępne modele

- Sieci neuronowe
- Tradycyjne modele ML

AutoPyTorch

- Oparty na PyTorchu oraz Deep Learningu
- Optymalizacja parametrów i hiperparametrów sieci neuronowej
- Implementacja i automatyczne strojenie pełnego pipeline'u Deep Learningowego
- Warmstart optymalizacji, dzięki próbkowaniu konfiguracji z portfolio
- Ensemble selection



AUTOPYTORCH

Multi-Fidelity Optimalization

- BOHB – połączenie optymalizacji bayesowskiej (BO) i hyperbandu (HB)
- BOHB lepszy niż BO i HB
- Nawet 55 razy szybszy niż Random Search
- Dwie pętle podobnie jak w HB (SuccessiveHalving)
- Identyfikowanie obiecujących obszarów w przestrzeni konfiguracyjnej

Optymalizacja równoległa

- Architektura master-worker
- Wykorzystanie dodatkowych zasobów

Wybór modelu

- Podziały:
  - Zdefiniowane przez użytkownika
  - Automatyczne podziały
  - Kroswalidacja (scikit-learn)
- Metryka:
  - Predefiniowana
  - Wybrana przez użytkownika
- Wcześniejsze zatrzymanie

Łączenie modeli

- Trenowanie wielu modeli, a następnie ich łączenie
- Łączenie modeli inspirowane auto-sklearnem
- Podawanie modeli do zbioru i wyznaczanie ich wag

Portfolia

- Warmstart optymalizacji
- Pierwsza iteracja BOHB z przygotowanym zestawem konfiguracji

LCBench

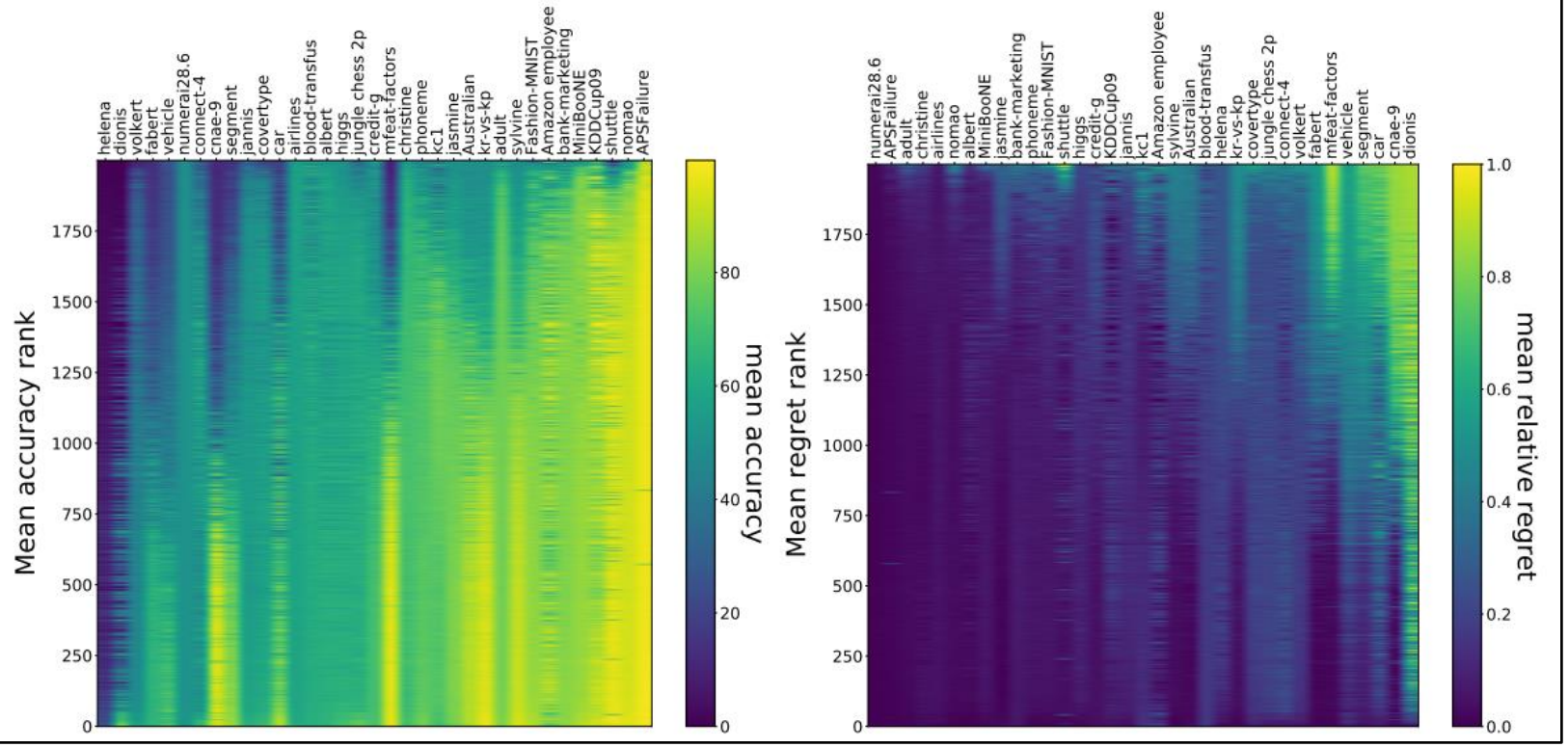
- Nowy benchmark
- 2000 konfiguracji na 35 zbiorach danych
- 7 hiperparametrów
- 12 / 25 / 50 epok
- Zbierano wiele różnych metryk

TABLE 1
Hyperparameters and ranges of our Configuration Space 1.

	Name	Range	log	Type
Architecture	number of layers	[1, 5]	-	int
	max. number of units	[64, 512]	✓	int
Hyper-parameters	batch size	[16, 512]	✓	int
	learning rate (SGD)	[1e-4, 1e-1]	✓	float
	L2 regularization	[1e-5, 1e-1]	-	float
	momentum	[0.1, 0.99]	-	float
	max. dropout rate	[0,1.0]	-	float

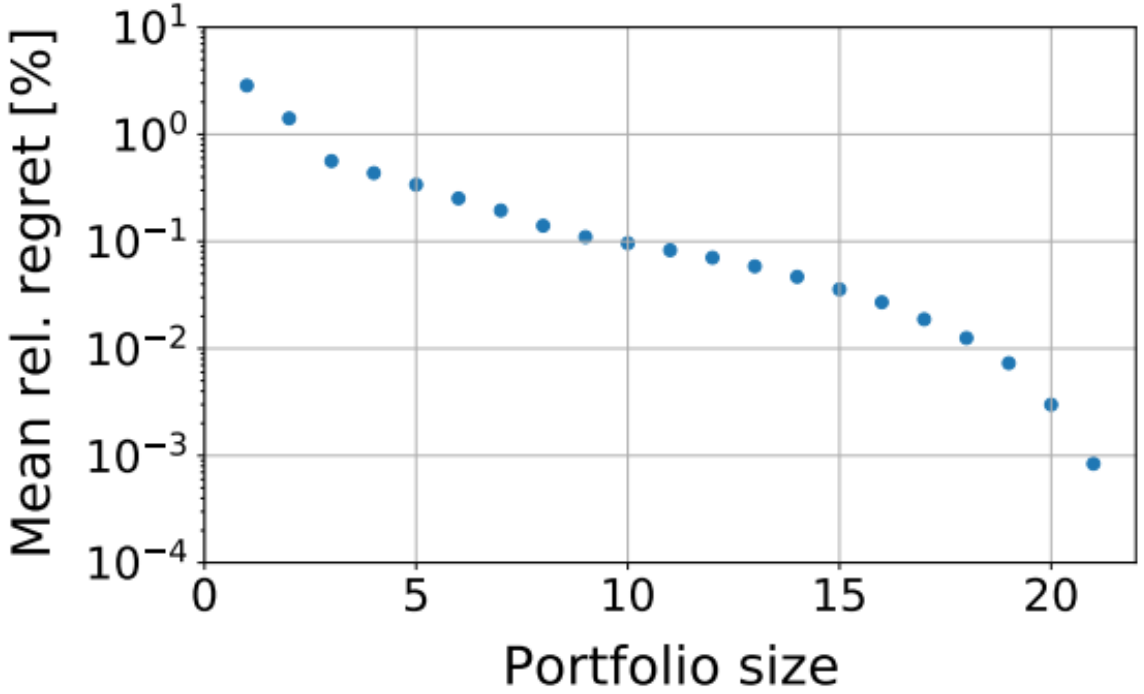
Wyniki pomiędzy zbiorami danych

- Te same konfiguracje na różnych zbiorach danych.
- Wyróżniono 22 dobre unikalne konfiguracje.
- Transfer konfiguracji ma obiecujące wyniki – portfolio.

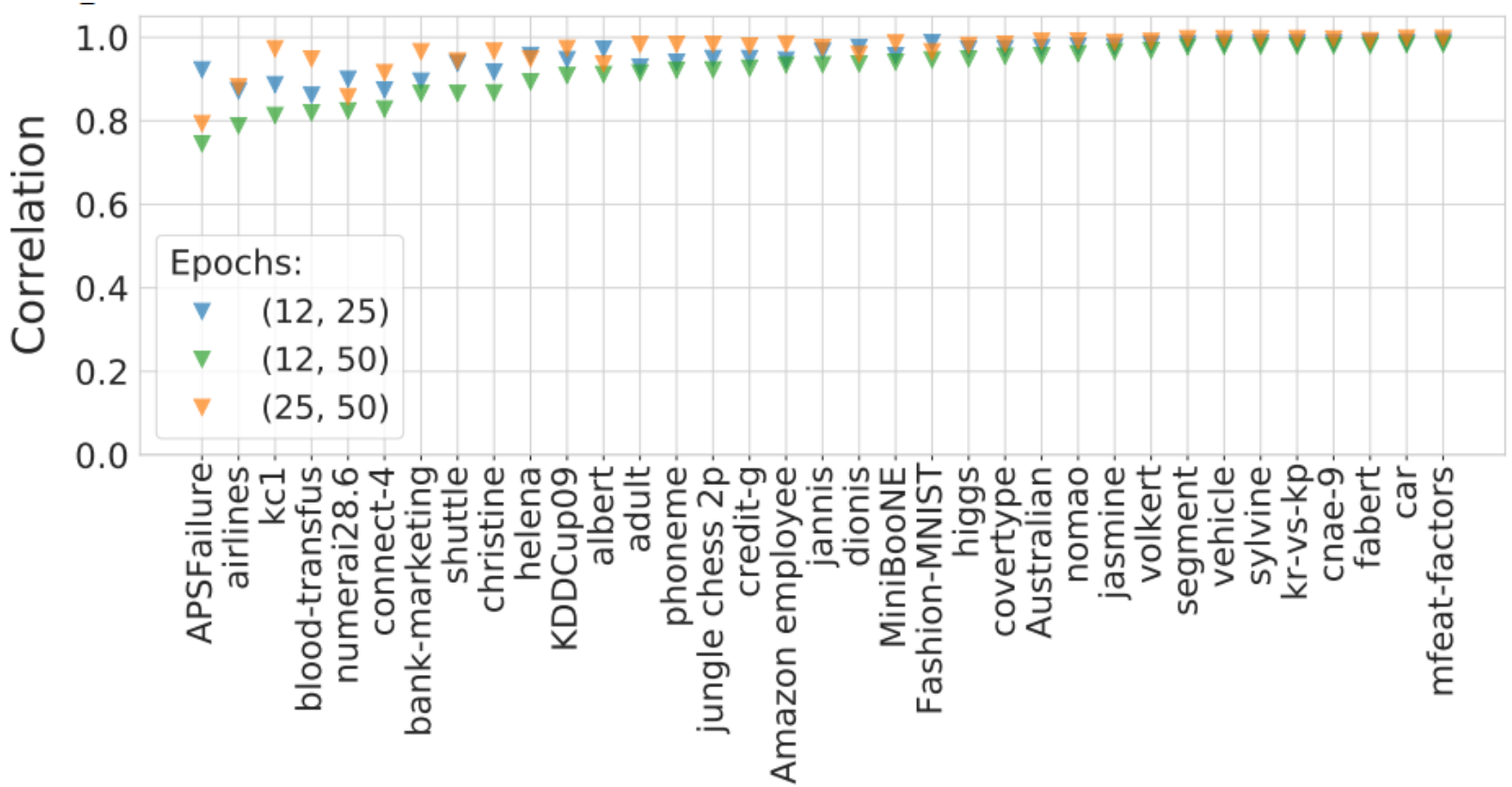


Wielkość portfolio

- 10 konfiguracji – mniej niż 1% average accuracy regret

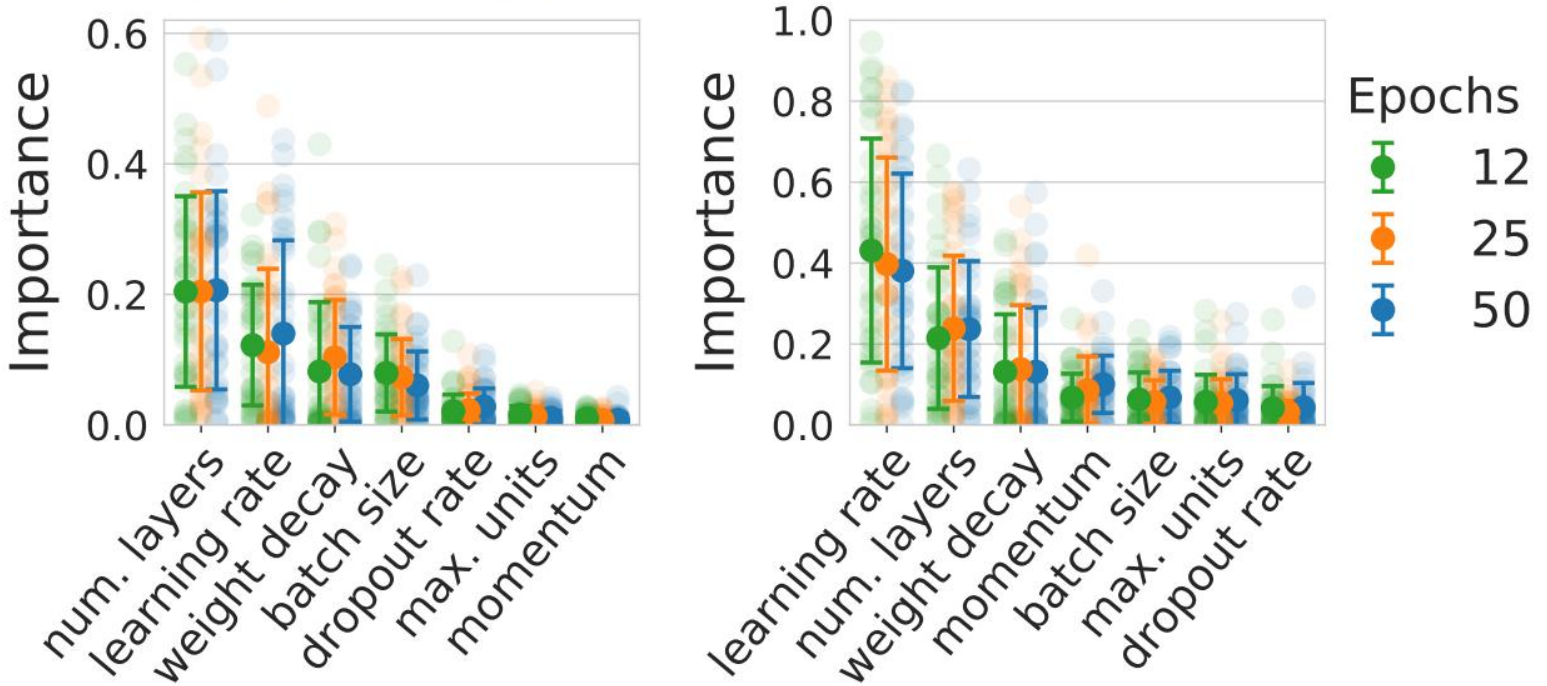


Korelacje budżetów



Znaczenie hiperparametrów

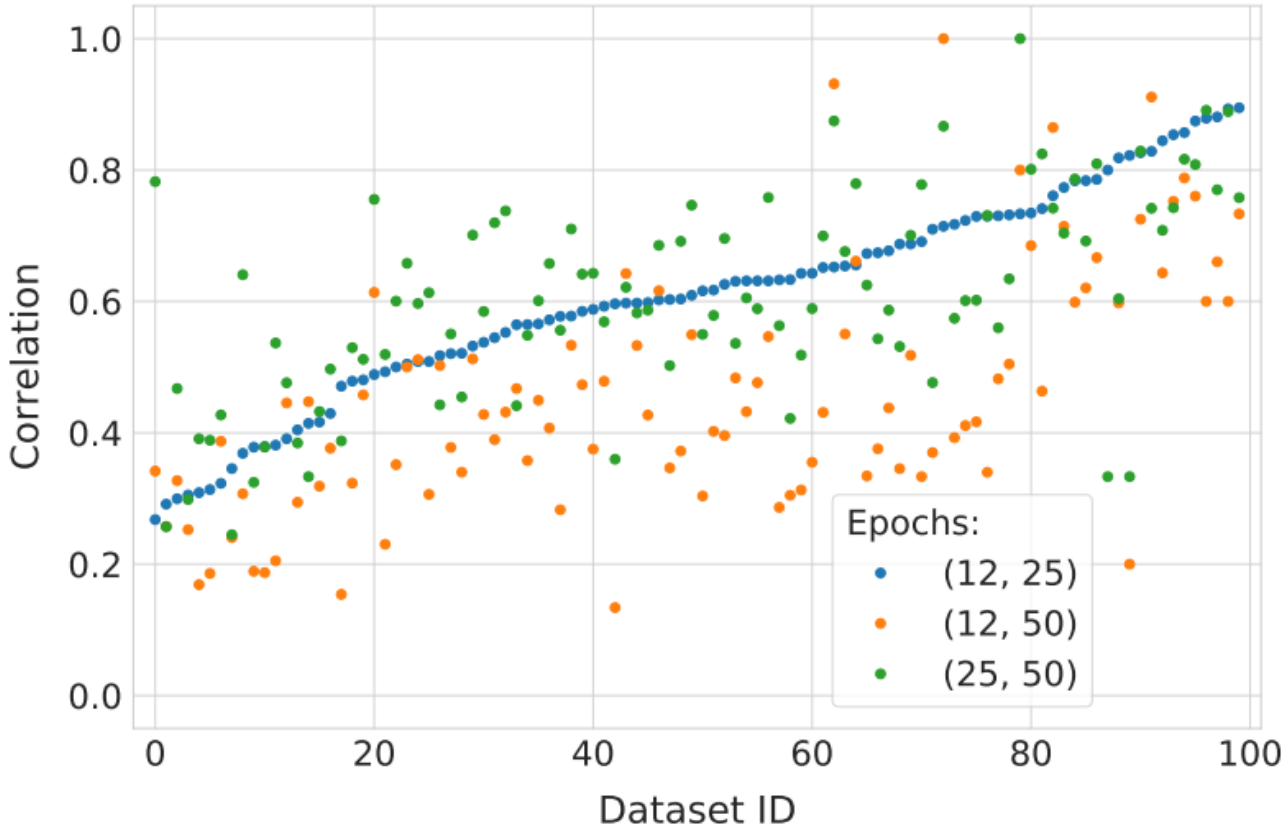
Fig. 6. Boxplots on the hyperparameter importance according to fANOVA (left) and LPI (right) on LCBench.



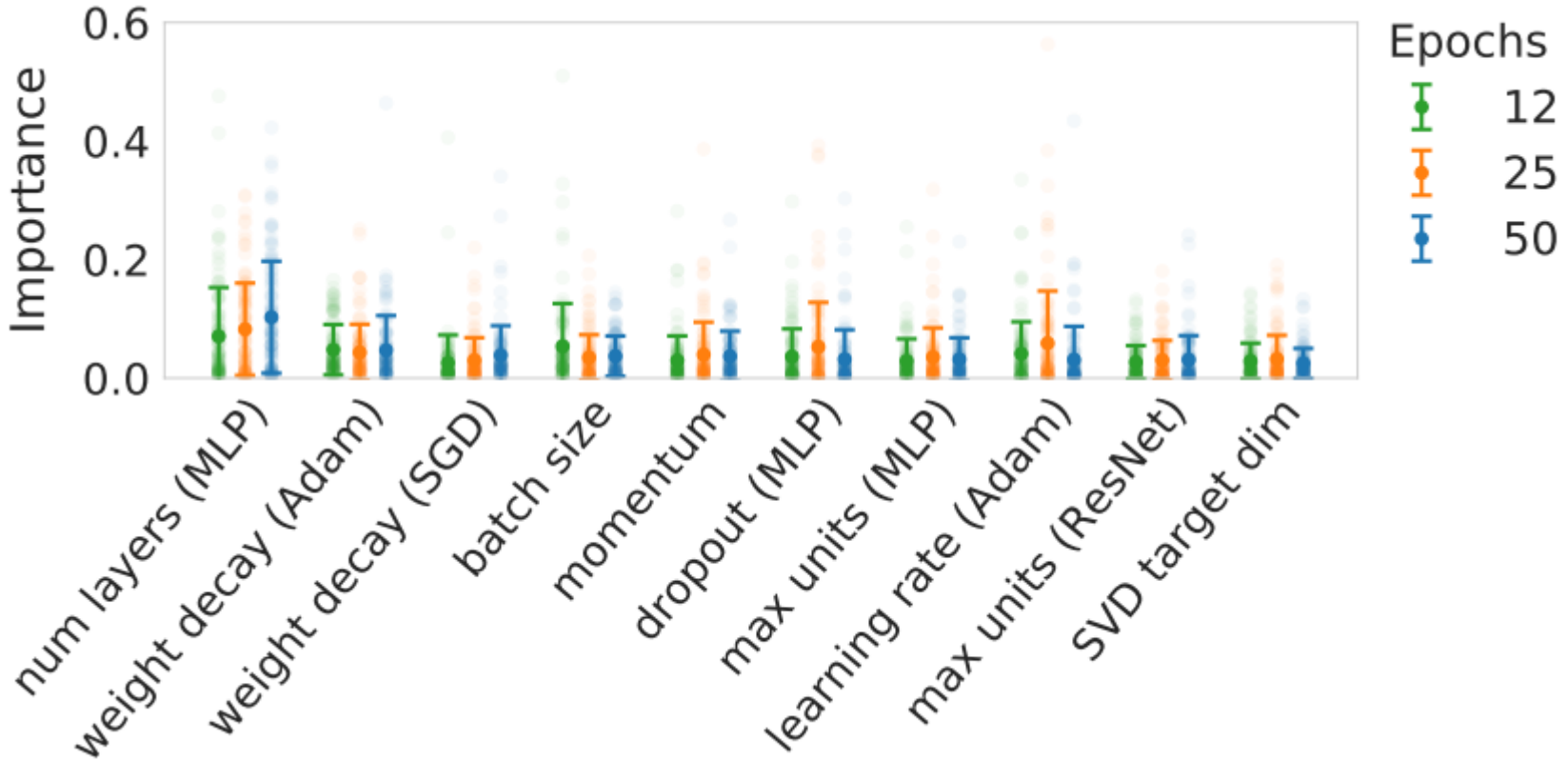
Ocena Auto-PyTorcha

- Duża przestrzeń konfiguracji i zbiorów danych.
- Rozszerza wyniki LCBench na więcej przypadków.
- Zarówno dobór architektury jak i hiperparametrów ma duże znaczenie.

Korelacje budżetów

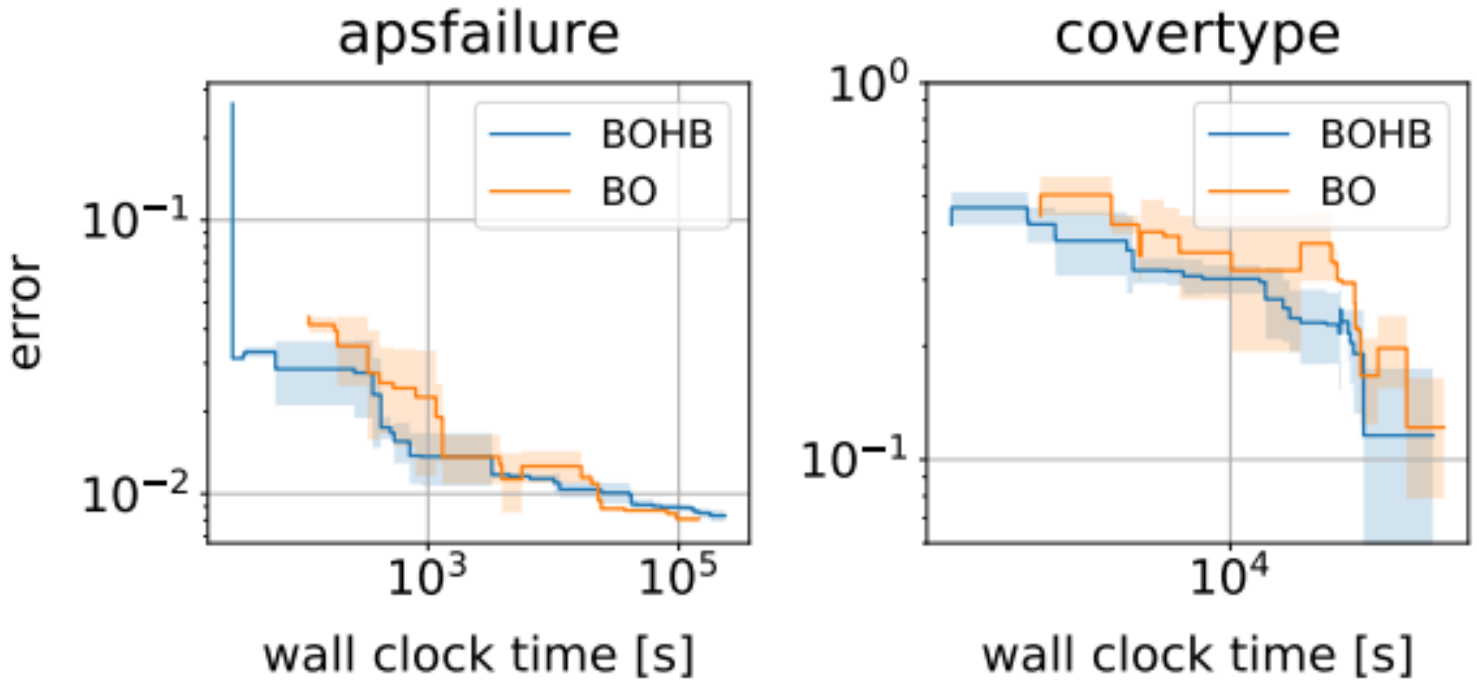


Znaczenie hiperparametrów

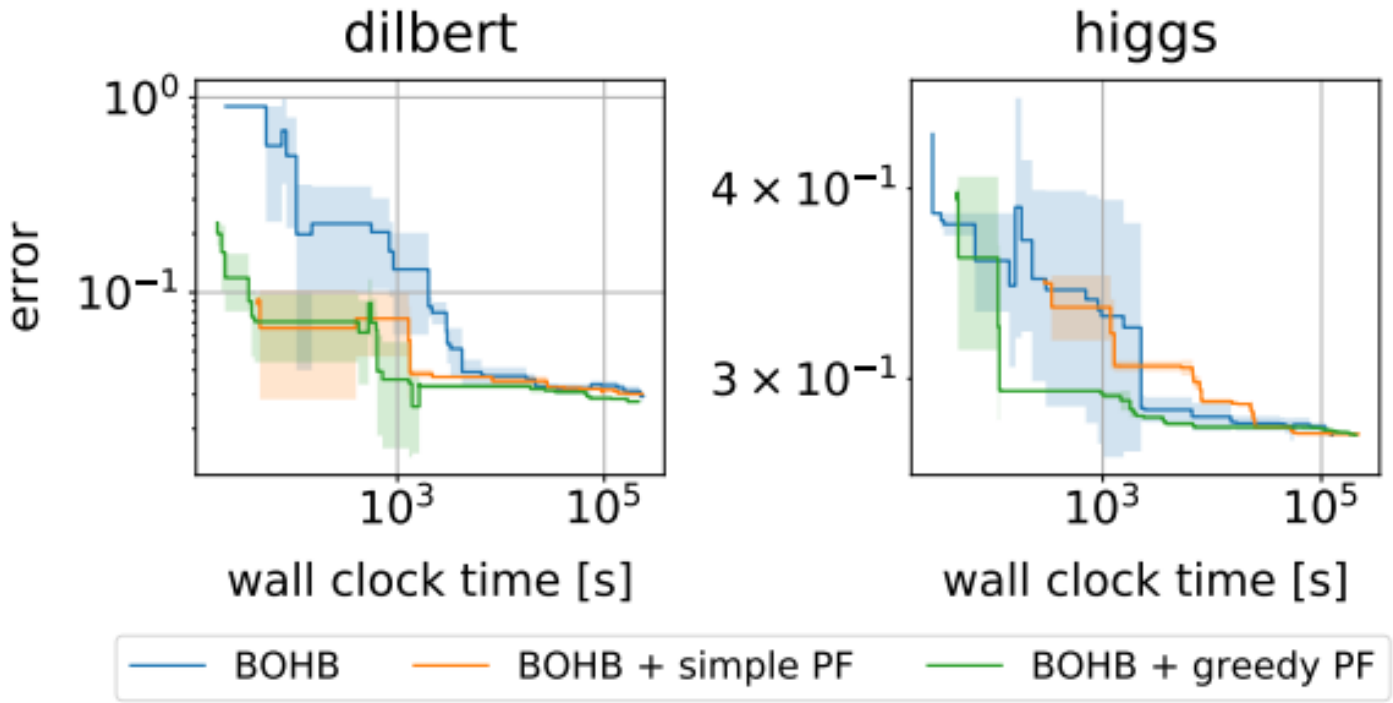


BO vs BOHB

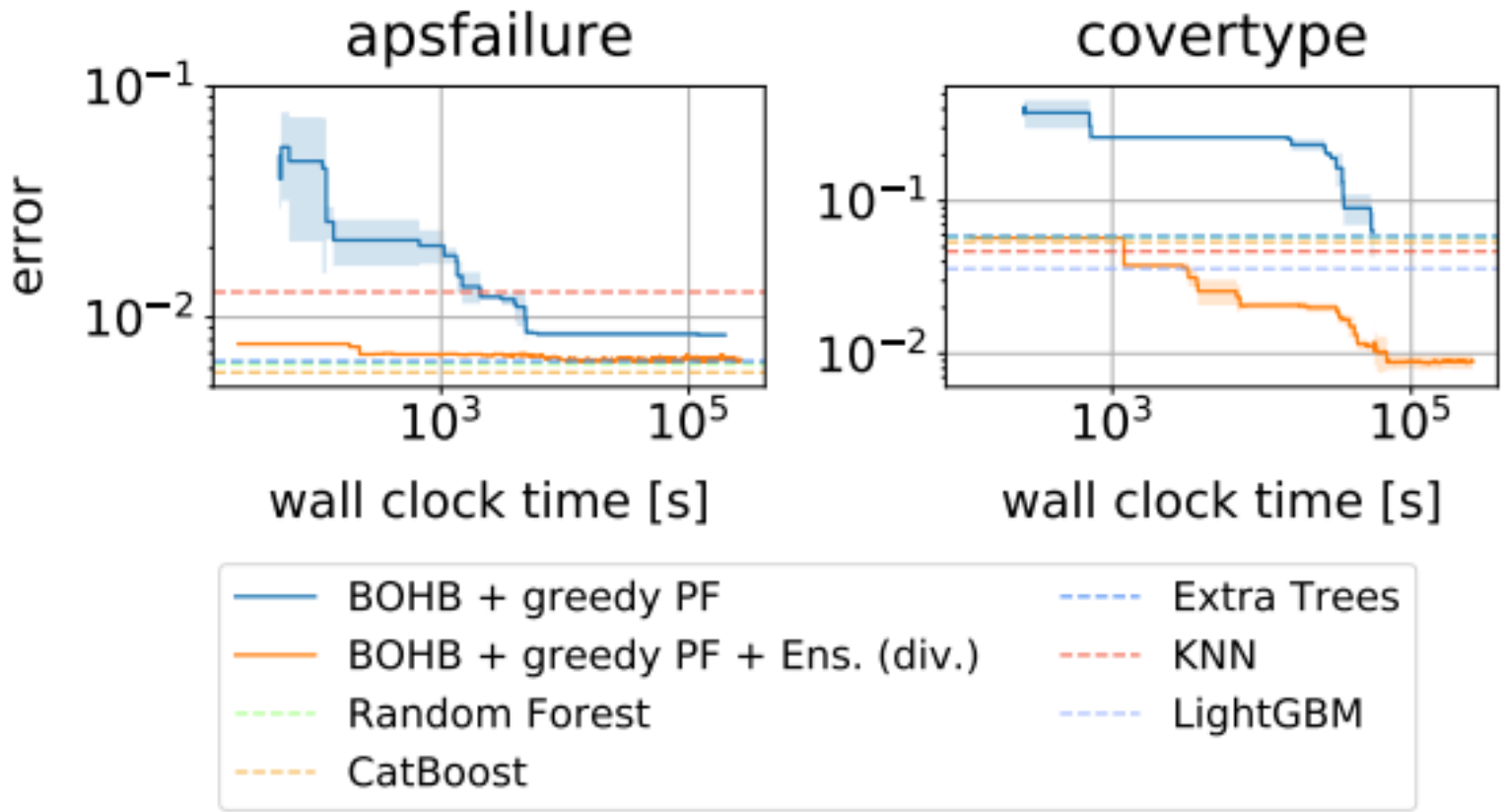
- Bayesian optimization vs Bayesian optimization with Hyperband



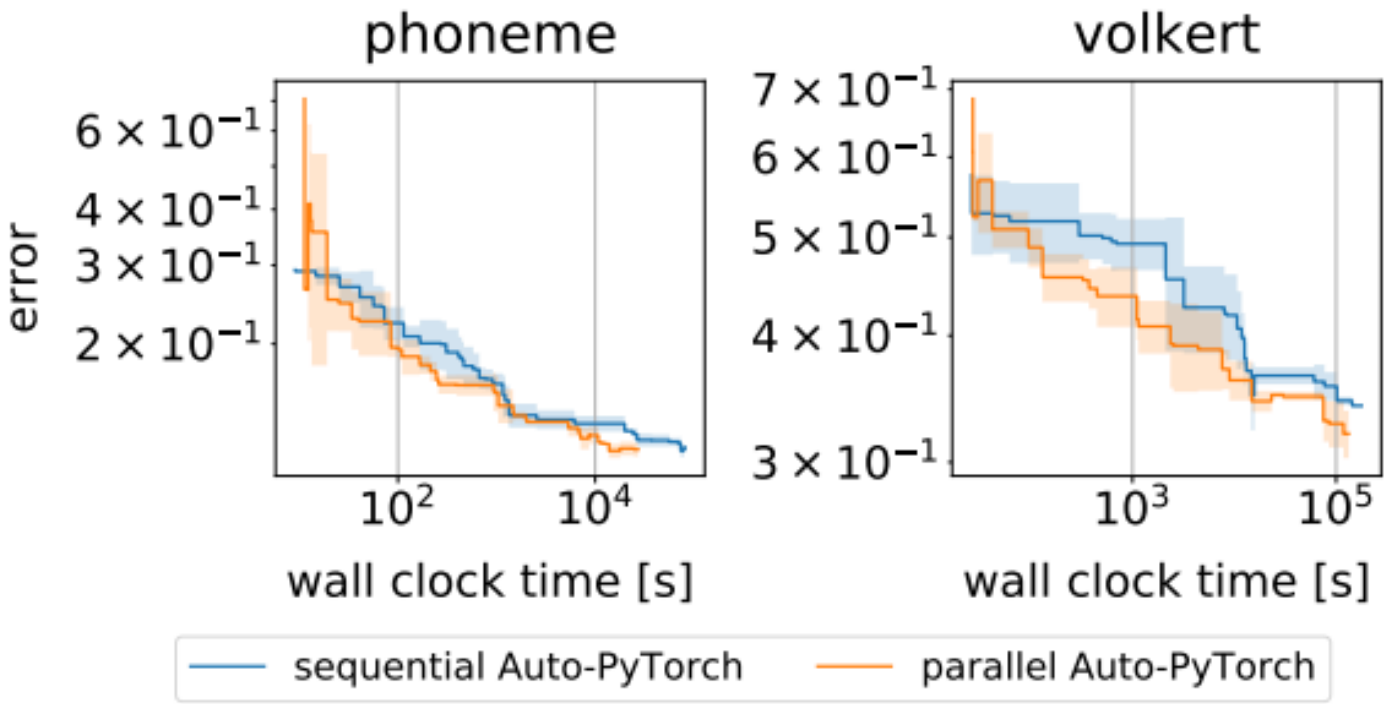
Portfolio


dilbert
higgs
error
10^0
10^{-1}
4×10^{-1}
3×10^{-1}
10^3
10^5
wall clock time [s]
BOHB
BOHB + simple PF
BOHB + greedy PF

Ensembling


apsfailure
covertype
error
10^{-1}
10^{-2}
10^{3}
10^{5}
wall clock time [s]
BOHB + greedy PF
BOHB + greedy PF + Ens. (div.)
Random Forest
CatBoost
Extra Trees
KNN
LightGBM

Wielowątkowość

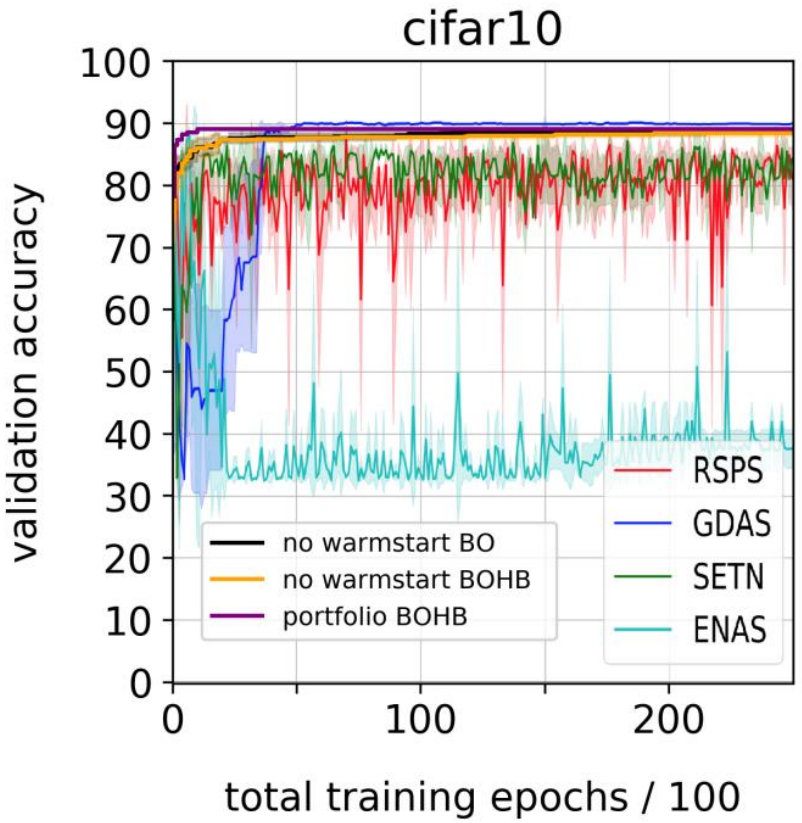


Porównanie do innych AutoML

- Accuracy w porównaniu do innych frameworków.
- 1h runtime
- '-': crash

	Auto-PyTorch Tabular	AutoNet2.0 [22]	AutoKeras [6]	Auto-Sklearn [5]	hyperopt. [3]	Auto-PyTorch w/ ens.	AutoGluon w/ stack.	Auto-PyTorch w/o ens.	AutoGluon w/o stack.
covertype	**96.86 ± 0.41**	68.22 ± 2.46	61.61 ± 3.52	-	-	**96.86 ± 0.41**	-	**93.35 ± 0.02**	-
volkert	**70.52 ± 0.51**	60.90 ± 3.89	44.25 ± 2.38	67.32 ± 0.46	-	70.52 ± 0.51	**72.16 ± 0.00**	**70.74 ± 0.01**	68.34 ± 0.10
higgs	**73.01 ± 0.09**	71.36 ± 0.55	71.25 ± 0.29	72.03 ± 0.33	-	73.01 ± 0.09	**73.26 ± 0.00**	**72.64 ± 0.02**	72.60 ± 0.00
car	96.41 ± 1.47	96.14 ± 0.35	93.39 ± 2.82	**98.42 ± 0.62**	**98.95 ± 0.96**	96.41 ± 1.47	**99.12 ± 0.50**	**98.59 ± 0.01**	97.16 ± 0.35
mfeat-fact.	**99.10 ± 0.18**	**98.97 ± 0.21**	97.73 ± 0.23	98.64 ± 0.39	97.88 ± 38.48	**99.10 ± 0.18**	98.79 ± 0.15	**98.78 ± 0.01**	98.03 ± 0.23
apsfailure	**99.41 ± 0.05**	98.83 ± 0.03	-	**99.43 ± 0.04**	-	99.41 ± 0.05	**99.57 ± 0.01**	99.38 ± 0.00	**99.50 ± 0.03**
phoneme	**90.55 ± 0.14**	86.61 ± 0.19	86.76 ± 0.12	89.26 ± 0.14	89.79 ± 4.54	**90.55 ± 0.14**	**90.63 ± 0.08**	**90.40 ± 0.01**	89.62 ± 0.06
dilbert	**98.70 ± 0.15**	97.43 ± 0.46	96.51 ± 0.62	98.14 ± 0.47	-	98.70 ± 0.15	**99.33 ± 0.00**	**98.54 ± 0.00**	98.17 ± 0.05

Image classification



Kierunki rozwoju

- Ensembling jest znany z overfittingu – generalizacja.
- Połączenie Auto-PytTorcha i Auto-Sklearna, aby stroić parametry pozostałych modeli w ensemblingu.
- Użycie wyników z analizy znaczenia hiperparametrów do polepszenia frameworka.
- Rozszerzenie danych tabelarycznych na dowolne zadania (image, text, time series).

Instalacja

System requirements

Auto-PyTorch has the following system requirements:

- Linux operating system (for example Ubuntu) (get Linux here),
- Python (>=3.6) (get Python here).
- C++ compiler (with C++11 supports) (get GCC here) and
- SWIG (version 3.0.* is required; >=4.0.0 is not supported) (get SWIG here).

Pierwsze doświadczenia z kodem

```
# initialise Auto-PyTorch api
api = TabularClassificationTask()
```

```
api.search(
    X_train=X_train,
    y_train=y_train,
    X_test=X_test,
    y_test=y_test,
    optimize_metric='accuracy',
    total_walltime_limit=300,
    func_eval_time_limit_secs=50
)
```

```
[WARNING] [2022-04-05 17:57:07,320:Client-Validation] AutoPyTorch previously received features of type <class 'numpy.ndarra
y'> yet the current features have type <class 'pandas.core.frame.DataFrame'>. Changing the dtype of inputs to an estimator m
ight cause problems
[WARNING] [2022-04-05 17:57:07,355:Client-Validation] AutoPyTorch previously received features of type <class 'numpy.ndarra
y'> yet the current features have type <class 'pandas.core.frame.DataFrame'>. Changing the dtype of inputs to an estimator m
ight cause problems
[ERROR] [2022-04-05 17:57:14,904:Client-AutoPyTorch:0a12ddf8-b4f9-11ec-8060-00155d61704c:1] Traditional prediction for lgb f
ailed with run state StatusType.CRASHED.
Additional info:
traceback: Traceback (most recent call last):
  File "/home/witasm/.local/lib/python3.8/site-packages/autoPyTorch/evaluation/tae.py", line 39, in fit_predict_try_except_d
ecorator
    ta(queue=queue, **kwargs)
  File "/home/witasm/.local/lib/python3.8/site-packages/autoPyTorch/evaluation/train_evaluator.py", line 485, in eval_functi
on
    evaluator.fit_predict_and_loss()
  File "/home/witasm/.local/lib/python3.8/site-packages/autoPyTorch/evaluation/train_evaluator.py", line 163, in fit_predict
_and_loss
    y_train_pred, y_opt_pred, y_valid_pred, y_test_pred = self._fit_and_predict(pipeline, split_id,
  File "/home/witasm/.local/lib/python3.8/site-packages/autoPyTorch/evaluation/train_evaluator.py", line 337, in _fit_and_pr
edict
    fit_and_suppress_warnings(self.logger, pipeline, X, y)
  File "/home/witasm/.local/lib/python3.8/site-packages/autoPyTorch/evaluation/abstract_evaluator.py", line 321, in fit_and_
suppress_warnings
    pipeline.fit(X, y)
  File "/home/witasm/.local/lib/python3.8/site-packages/autoPyTorch/evaluation/abstract_evaluator.py", line 94, in fit
    return self.pipeline.fit(X, y)
  File "/home/witasm/.local/lib/python3.8/site-packages/autoPyTorch/pipeline/base_pipeline.py", line 155, in fit
    self.fit_estimator(X, y, **fit_params)
  File "/home/witasm/.local/lib/python3.8/site-packages/autoPyTorch/pipeline/base_pipeline.py", line 174, in fit_estimator
    self._final_estimator.fit(X, y, **fit_params)
  File "/home/witasm/.local/lib/python3.8/site-packages/autoPyTorch/pipeline/components/base_choice.py", line 217, in fit
```

Statystyki

```
print(api.sprint_statistics())
```

```
autoPyTorch results:
        Dataset name: 0a12ddf8-b4f9-11ec-8060-00155d61704c
        Optimisation Metric: accuracy
        Best validation score: 0.9842696629213483
        Number of target algorithm runs: 23
        Number of successful target algorithm runs: 19
        Number of crashed target algorithm runs: 2
        Number of target algorithms that exceeded the time limit: 2
        Number of target algorithms that exceeded the memory limit: 0
```

Funkcja show_models()

	Preprocessing	Estimator	Weight
0	SimpleImputer,NoEncoder,Normalizer,KitchenSink	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0.14
1	SimpleImputer,NoEncoder,NoScaler,Nystroem	no embedding,ShapedResNetBackbone,FullyConnectedHead,nn.Sequential	0.14
2	SimpleImputer,NoEncoder,Normalizer,KernelPCA	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0.14
3	SimpleImputer,NoEncoder,Normalizer,KernelPCA	no embedding,MLPBackbone,FullyConnectedHead,nn.Sequential	0.1
4	SimpleImputer,NoEncoder,StandardScaler,NoFeaturePreprocessing	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0.1
5	None	ETLearner	0.08
6	SimpleImputer,NoEncoder,MinMaxScaler,KitchenSink	no embedding,MLPBackbone,FullyConnectedHead,nn.Sequential	0.06
7	SimpleImputer,NoEncoder,StandardScaler,NoFeaturePreprocessing	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0.06
8	SimpleImputer,NoEncoder,MinMaxScaler,PowerTransformer	no embedding,ResNetBackbone,FullyConnectedHead,nn.Sequential	0.04
9	SimpleImputer,NoEncoder,StandardScaler,NoFeaturePreprocessing	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0.04
10	SimpleImputer,NoEncoder,Normalizer,PowerTransformer	no embedding,ShapedResNetBackbone,FullyConnectedHead,nn.Sequential	0.02
11	SimpleImputer,NoEncoder,Normalizer,TruncSVD	no embedding,ShapedMLPBackbone,FullyConnectedHead,nn.Sequential	0.02
12	None	CBLearner	0.02
13	None	SVMLearner	0.02
14	None	KNNLearner	0.02

Inne parametry

```
api = TabularClassificationTask(
    search_space_updates=get_search_space_updates(),
    include_components={'network_backbone': ['MLPBackbone', 'ResNetBackbone'],
                        'encoder': ['OneHotEncoder']}
)
```

Bibliografia

- Lucas Zimmer, Marius Lindauer and Frank Hutter *Auto-Pytorch: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL*
- https://automl.github.io/Auto-PyTorch/development

Dziękujemy za uwagę