AutoML - raport nr 1

Aleksandra Buchowicz, Filip Pazio

1 Wstęp

Celem projektu było przeanalizowanie tunowalności hiperparametrów trzech algorytmów uczenia maszynowego - Random Forest, Xgboost oraz GradientBoosting. Modele zaimplementowano w języku Python przy pomocy bibliotek `scikit-learn` i `xgboost`. Do oceny jakości dopasowania modelu wykorzystano miarę AUC.
Wybór i siatkę hiperparametrów przedstawiono w Tabeli 7. W porównaniu do literatury [1, 5], z powodu ograniczeń sprzętowych, ograniczono ilość optymalizowanych hiperparametrów.
Do tunowania hiperparametrów wykorzystano dwie różne techniki losowania punktów, oparte kolejno na wyborze punktów z rozkładu jednostajnego (Randomized Search) oraz na technice bayesowskiej. Techniki zaimplementowano odpowiednio przy pomocy pakietów `sklearn.model_selection` i `scikit-optimize`. Dla obu metod wykonano 50 iteracji.

1.1 Zbiory danych

Do przeprowadzenia eksperymentów wykorzystano następujące zbiory danych do klasyfikacji binarnej z serwisu OpenML:

1. adult (ID: 45068)
2. blood-transfusion-service-center (ID: 1464)
3. diabetes (ID: 37)
4. phoneme (ID: 1489)

Wybrane zbiory danych nie zawierały braków danych i nie wymagały obróbki. W niektórych przypadkach wykorzystano `LabelEncoder` z biblioteki `sklearn.preprocessing` do kodowania etykiet.

2 Wyniki

W tej sekcji opisano wyniki eksperymentu dla poszczególnych algorytmów. Podsumowanie wyników w formie wykresów pudełkowych można znaleźć na Rysunku 2.

2.1 RandomForest

Model został zaimplementowany przy pomocy `sklearn.ensemble.RandomForestClassifier`.

	Zbiór 1	Zbiór 2	Zbiór 3	Zbiór 4
RandomizedSearchCV	0.914	0.767	0.821	0.9997
BayesSearchCV	0.916	0.765	0.823	0.9998

Tabela 1: Maksymalne wartości AUC dla `RandomForestClassifier` osiągnięte na poszczególnych zbiorach przy wybranej metodzie tunowania.

Estymator, który osiągnął najwyższe średnie AUC pomiędzy czterema zbiorami, podczas poszukiwań z użyciem metody RandomSearchCV przedstawiono w Tabeli 2.

bootsrap	max_features	min_sampes_split	n_estimators	mean AUC
True	0.934	0.08	706	0.835

Tabela 2: Hiperparametry modelu `RandomForestClassifier`, dla których pozyskano najlepsze średnie AUC obliczone na czterech zbiorach przy pomocy `RandomSearchCV` oraz średnia wartość tej miary.

2.2 XGBoost

Model został zaimpementowany przy pomocy `xgboost.XGBClassifier`.

	Zbiór 1	Zbiór 2	Zbiór 3	Zbiór 4
RandomizedSearchCV	0.928	0.711	0.793	0.949
BayesSearchCV	0.930	0.736	0.799	0.956

Tabela 3: Maksymalne wartości AUC dla `XGBClassifier` osiągnięte na poszczególnych zbiorach przy wybranej metodzie tunowania.

Estymator, który osiągnął najwyższe średnie AUC pomiędzy czterema zbiorami, podczas poszukiwań z użyciem metody RandomSearchCV przedstawiono w Tabeli 4.

n_estimators	learning_rate	max_depth	min_child_weight	mean AUC
427	0.387	5	0.601	0.830

Tabela 4: Hiperparametry modelu `XGBClassifier`, dla których pozyskano najlepsze średnie AUC obliczone na czterech zbiorach oraz średnia wartość tej miary.

2.3 Gradientboosting

Model został zaimpementowany przy pomocy `sklearn.ensemble.HistGradientBoostingClassifier`.

	Zbiór 1	Zbiór 2	Zbiór 3	Zbiór 4
RandomizedSearchCV	0.929	0.749	0.830	0.955
BayesSearchCV	0.928	0.753	0.840	0.955

Tabela 5: Maksymalne wartości AUC dla `HistGradientBoostingClassifier` osiągnięte na poszczególnych zbiorach przy wybranej metodzie tunowania.

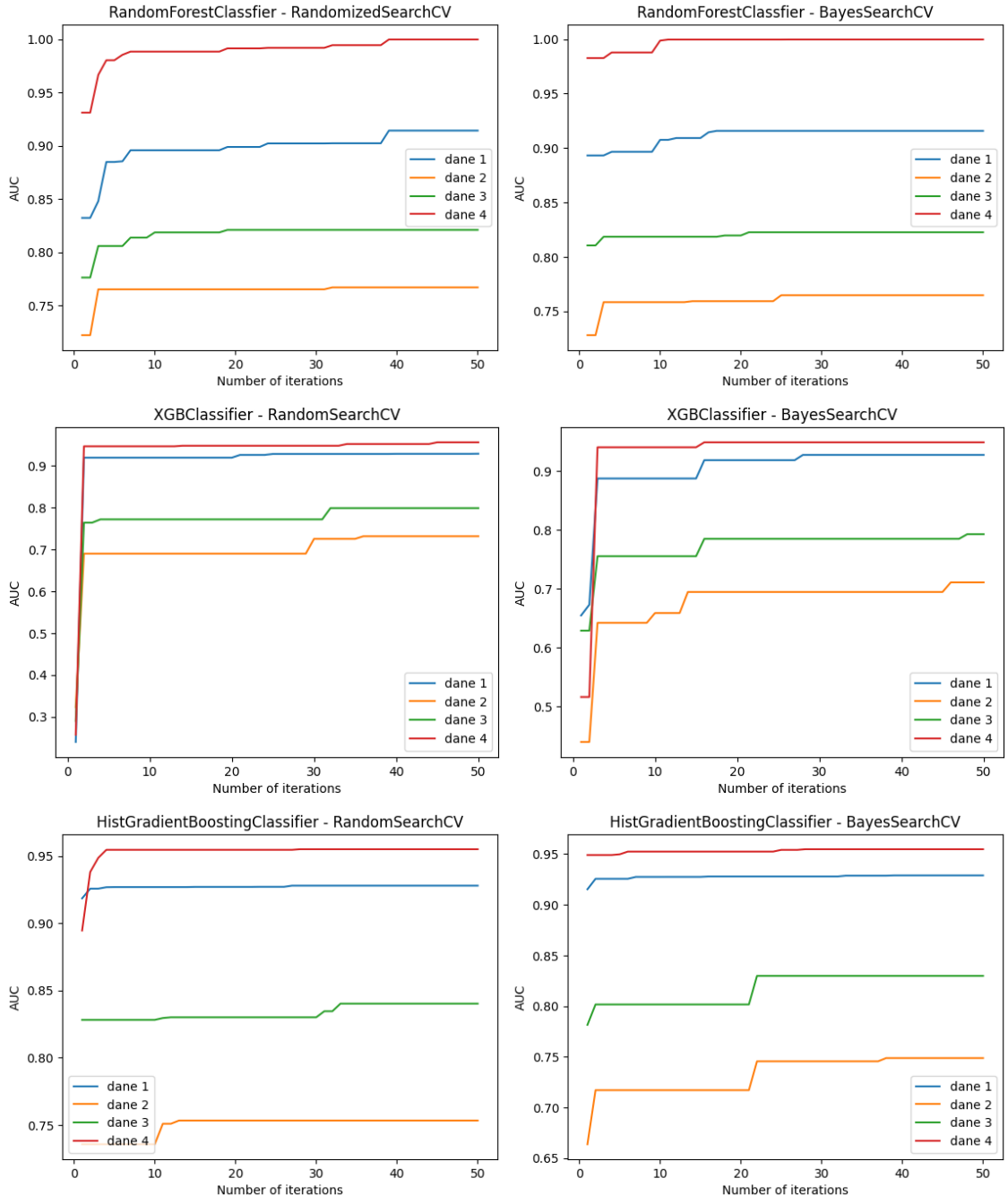
Estymator, który osiągnął najwyższe średnie AUC pomiędzy czterema zbiorami, podczas poszukiwań z użyciem metody RandomSearchCV przedstawiono w Tabeli 5.

max_iter	learning_rate	max_leaf_ndoes	min_samples_leaf	max_depth	mean AUC
42	0.081	22	30	5	0.856

Tabela 6: Hiperparametry modelu `HistGradientBoostingClassifier`, dla których pozyskano najlepsze średnie AUC obliczone na czterech zbiorach oraz średnia wartość tej miary.

3 Zbieżność metod tunowania parametrów

Na Rysunku 1 zwizualizowaliśmy proces znajdowania optymalnego rozwiązania przez rozważane przez nas techniki. Krzywe na wykresach przedstawiają najwyższe AUC spośród wszystkich znalezionych kombinacji hiperparametrów w zależności od liczby iteracji wykonanych przez technikę.



Rysunek 1: Liczba iteracji potrzebna na osiągnięcie optymalnego rozwiązania dla poszczególnych algorytmów i metod tunowania.

4 Wnioski

1. `XGBClassifier` charakteryzował się największą AUC dla poszczególnych zbiorów oraz szybkim ustabilizowaniem się zarówno metody `RandomizedSearchCV` jak i `BayesSearchCV`.
2. Metoda `BayesSearchCV` z reguły daje lepsze wyniki, jednak potrzebuje większej liczby iteracji na ustabilizowanie.
3. Istotny wpływ na czas wymagany do wytunowania parametrów ma wielkość zbioru danych, na którym trenowany jest model.

Literatura

[1] Probst, Philipp, Anne-Laure Boulesteix, and Bernd Bischl. "Tunability: Importance of hyperparameters of machine learning algorithms." The Journal of Machine Learning Research 20.1 (2019): 1934-1965.

[2] `https://xgboost.readthedocs.io/en/latest/parameter.html#general-parameters`

[3] `https://xgboost.readthedocs.io/en/stable/python/python_api.html`

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingClassifier.html#sklearn.ensemble.HistGradientBoostingClassifier`

[5] `https://inria.github.io/scikit-learn-mooc/python_scripts/ensemble_hyperparameters.html`

[6] `https://machinelearningmastery.com/histogram-based-gradient-boosting-ensembles/`

Tabele i wykresy

	parametr	typ	kres dolny	kres górny
random forest	`bootstrap`	logical	-	-
	`max_features`	numeric	0.01	1
	`min_samples_split`	numeric	0.01	1
	`n_estimators`	integer	1	2000
xgboost	`n_estimators`	integer	1	5000
	`learning_rate`	numeric	0	10
	`max_depth`	integer	1	15
	`min_child_weight`	numeric	0	7
gradient boosting	`max_iter`	integer	1	500
	`learning_rate`	numeric	0.01	1
	`max_leaf_nodes`	integer	2	50
	`min_samples_leaf`	integer	1	50
	`max_depth`	integer	1	15

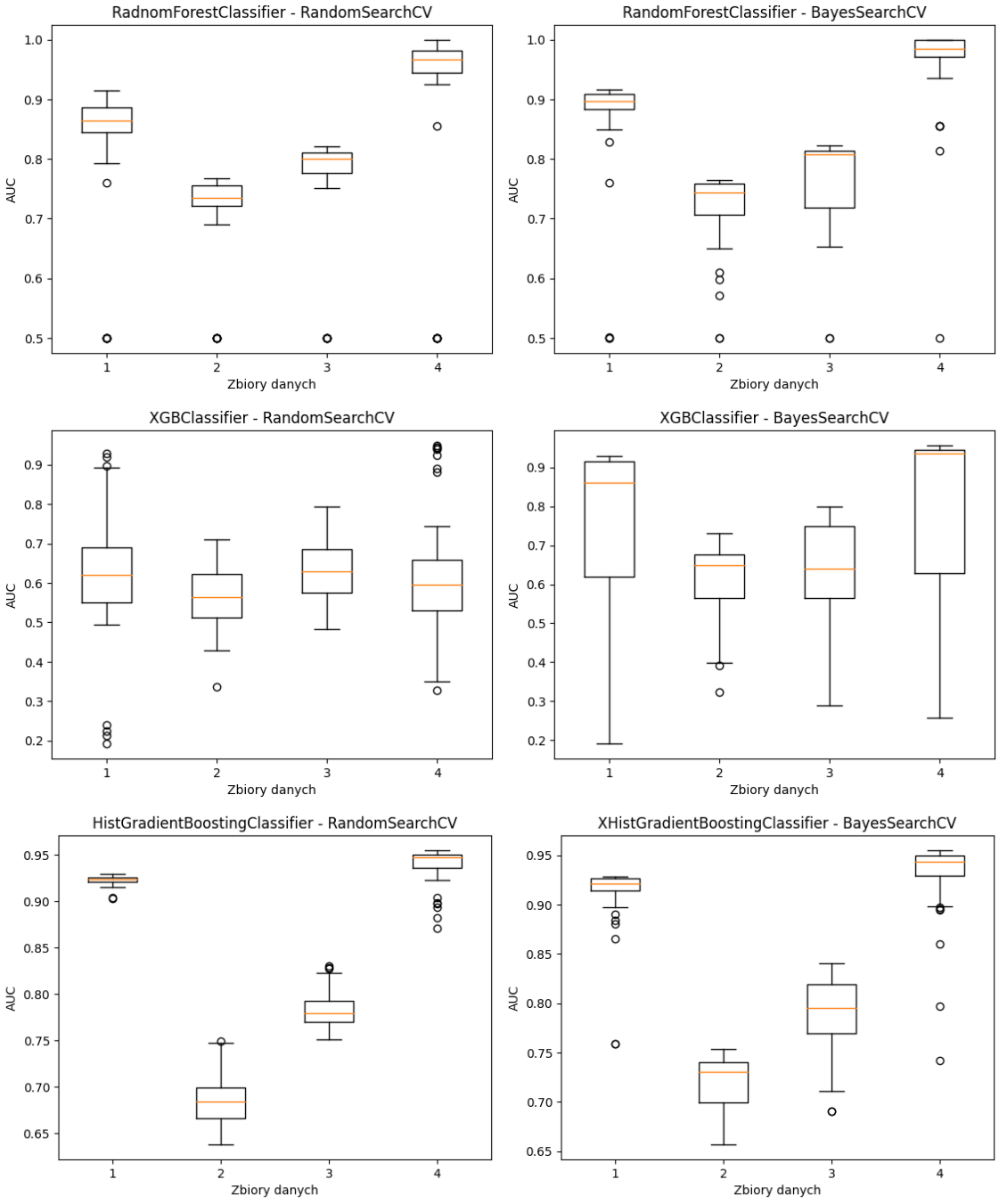
Tabela 7: Zakresy parametrów dla poszczególnych metod.

	parametr	Zbiór 1	Zbiór 2	Zbiór 3	Zbiór 4
RandomizedSearchCV	`bootstrap`	`True`	`True`	`False`	`True`
	`max_features`	0.743	0.934	0.411	0.743
	`min_samples_split`	0.012	0.08	0.241	0.012
	`n_estimators`	1894	706	1377	1894
BayesSearchCV	`bootstrap`	`True`	`True`	`True`	`False`
	`max_features`	0.47	0.823	0.258	0.117
	`min_samples_split`	0.01	0.458	0.041	0.01
	`n_estimators`	2000	2000	1046	694

Tabela 8: Optymalne parametry dla `RandomForestClassifier`.

	parametr	Zbiór 1	Zbiór 2	Zbiór 3	Zbiór 4
RandomizedSearchCV	`n_estimators`	537	1469	1005	427
	`learning_rate`	1.463	3.286	2.483	0.387
	`max_depth`	1	5	11	5
	`min_child_weight`	0.853	5.639	0.002	0.601
BayesSearchCV	`n_estimators`	3623	4	2329	4670
	`learning_rate`	0.806	1.132	0.873	1.009
	`max_depth`	1	1	15	14
	`min_child_weight`	0.186	7.000	0.000	0.000

Tabela 9: Optymalne parametry dla `XGBClassifier`.



Rysunek 2: Wykresy pudełkowe, przedstawiające zmienność AUC dla poszczególnych modeli, metod tunowania i zbiorów danych.

	parametr	Zbiór 1	Zbiór 2	Zbiór 3	Zbiór 4
RandomizedSearchCV	`max_iter`	438	191	42	386
	`learning_rate`	0.087	0.043	0.081	0.555
	`max_leaf_nodes`	16	21	22	45
	`min_samples_leaf`	18	36	30	6
	`max_depth`	5	1	5	12
BayesSearchCV	`max_iter`	377	500	183	179
	`learning_rate`	0.097	0.010	0.097	0.423
	`max_leaf_nodes`	34	50	2	39
	`min_samples_leaf`	50	19	2	23
	`max_depth`	9	1	1	15

Tabela 10: Optymalne parametry dla `HistGradientBoostingClassifier`.