[HW1] - Analiza tunowalności algorytmów

Dominik Kędzierski
Grzegorz Zakrzewski

Listopad 2023

1 Opis wykorzystanych danych

Do przeprowadzenia eksperymentów wykorzystaliśmy cztery zbiory danych pochodzące ze strony openml.org:

- KC1 software defect prediction - id 1067
- Amazon employee access - id 4135
- QSAR biodegradation - id 1494
- Satellite - id 40900

We wszystkich zbiorach danych zmienna celu jest zmienną binarną. Zbiory danych cechują się tym, że nie są bardzo duże oraz nie zawierają braków danych. Do oceny jakości modeli używamy miary *accuracy*.

2 Wybrane algorytmy i zakresy hiperparametrów

Wybrane algorytmy uczenia maszynowego to: *Random Forest*, *K-Nearest Neighbors*, *AdaBoost*. Zgodnie z poleceniem, zakresy hiperparametrów dla tych modeli miały być określone na podstawie literatury. Wbrew pozorom, nie było to bardzo proste zadanie. Zakresy hiperparametrów, określone na podstawie artykułów [1] i [2], znajdują się w Tabeli 3.

algorytm	hiperparametr	zakres wartości
Random Forest	`n_estimators`	{ 50, . . . , 300 }
	`criterion`	{ gini, entropy, log_loss }
	`max_depth`	{ 1, . . . , 50 }
	`min_samples_split`	{ 2, . . . , 20 }
	`min_samples_leaf`	{ 1, . . . , 20 }
	`max_features`	{ sqrt, log2, None }
K-Nearest Neighbors	`n_neighbors`	{ 1, . . . , 30 }
	`p`	[-1.0, 2.0]
	`weights`	{ uniform, distance }
AdaBoost	`n_estimators`	{ 50, . . . , 500 }
	`learning_rate`	[0.01, 2.0] (log-scale)
	`algorithm`	{ SAMME.R, SAMME }

Tabela 1: Zakresy hiperparametrów wybranych algorytmów uczenia maszynowego

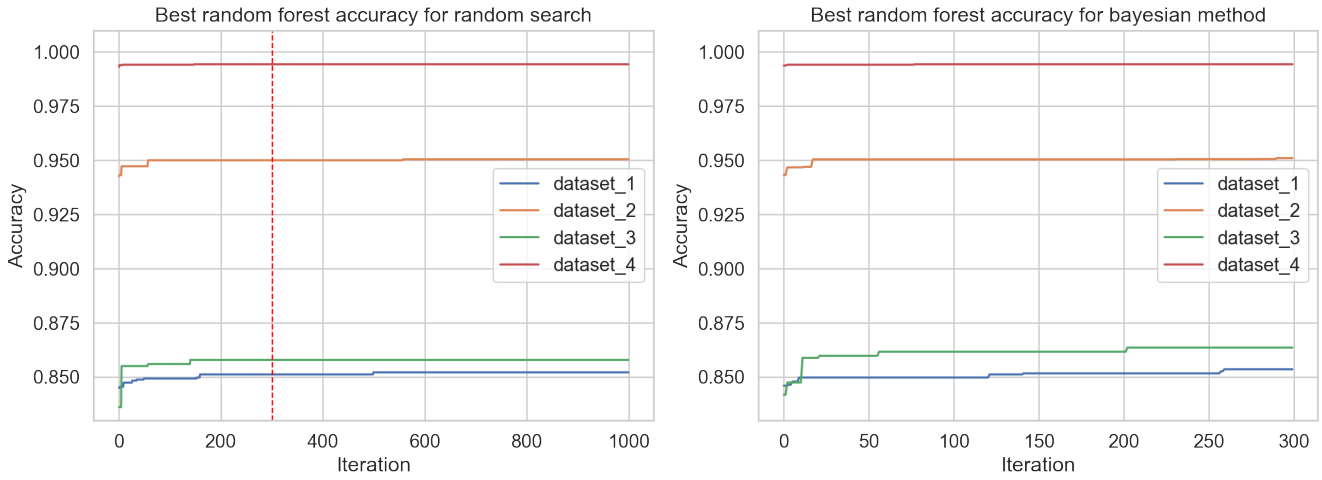
3 Techniki losowania punktów i liczba iteracji

Wykorzystujemy dwie techniki losowania punktów: *Random Search* (własna implementacja) oraz *Bayes Optimization* (biblioteka *skopt*). Niestety, ze względu na różną szybkość działania wybranych algorytmów uczenia maszynowego, liczba przeprowadzonych iteracji jest różna. Szczegóły przedstawione są w Tabeli 2.

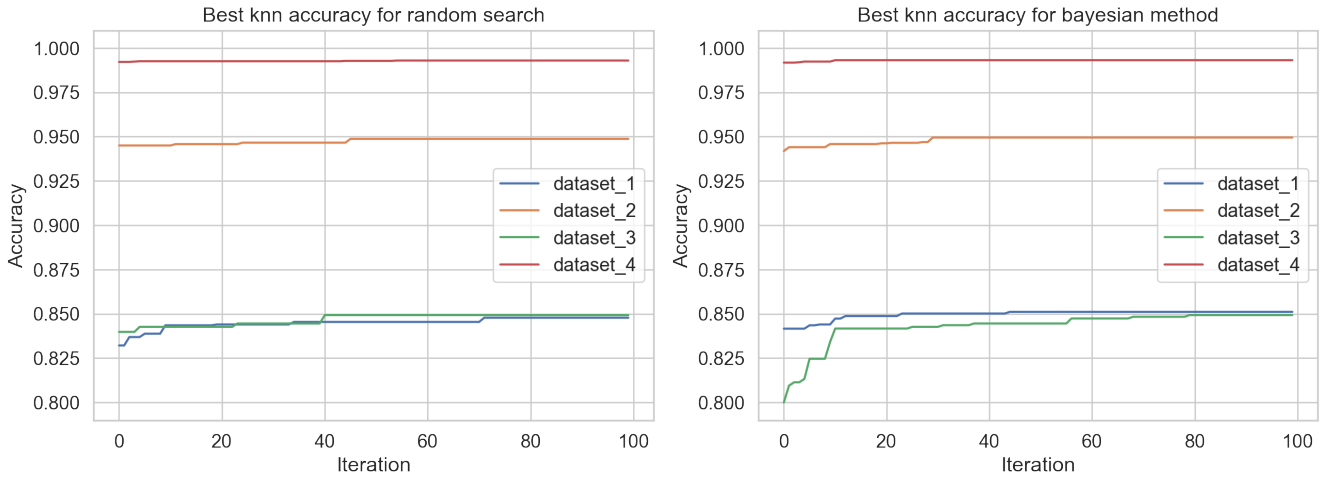
algorytm	liczba iteracji - random search	liczba iteracji - bayes optimization
Random Forest	1000	300
K-Nearest Neighbors	100	100
AdaBoost	120	300

Tabela 2: Liczba iteracji w dla każdego algorytmu i metody losowania punktów

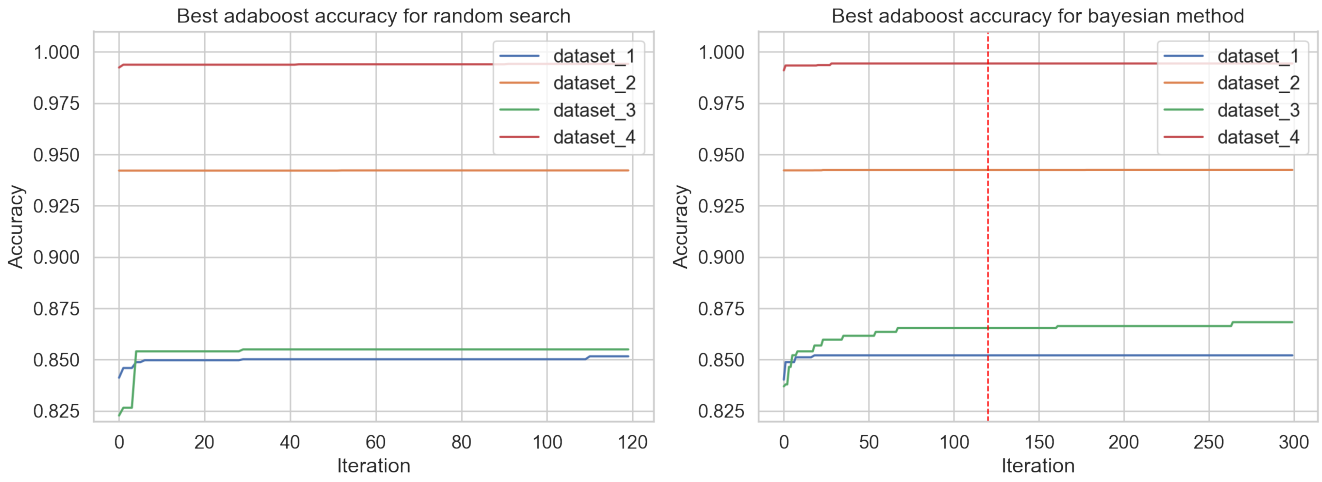
4 Porównanie zbieżności metod optymalizacji



Rysunek 1: Porównanie zbieżności dla *Random Forest*



Rysunek 2: Porównanie zbieżności dla *K-Nearest Neighbors*



Rysunek 3: Porównanie zbieżności dla *Adaboost*

Powyższe rysunki pokazują najlepszą wartość miary *accuracy* dla danej liczby przeprowadzonych iteracji, w porównaniu dla wszystkich zbiorów danych, algorytmów i metod losowania punktów. Pionowe, czerwone linie na rysunkach 1a i 3b podkreślają miejsca, gdzie liczba iteracji jest różna.
Na podstawie przeprowadzonych eksperymentów widać, że bayessowska metoda optymalizacji osiąga szybciej wskazany poziom *accuracy*. Istotnym faktem jest także to, że metoda random search bardzo szybko zatrzymuje polepszanie metryki. Nie jest już w stanie wylosować lepszych hiperparamtetrów. Natomiast metoda bayessowska nawet po długiej liczbie iteracji dalej jest w stanie znaleźć nowe parametry które nieznacznie polepszą działanie modelu.

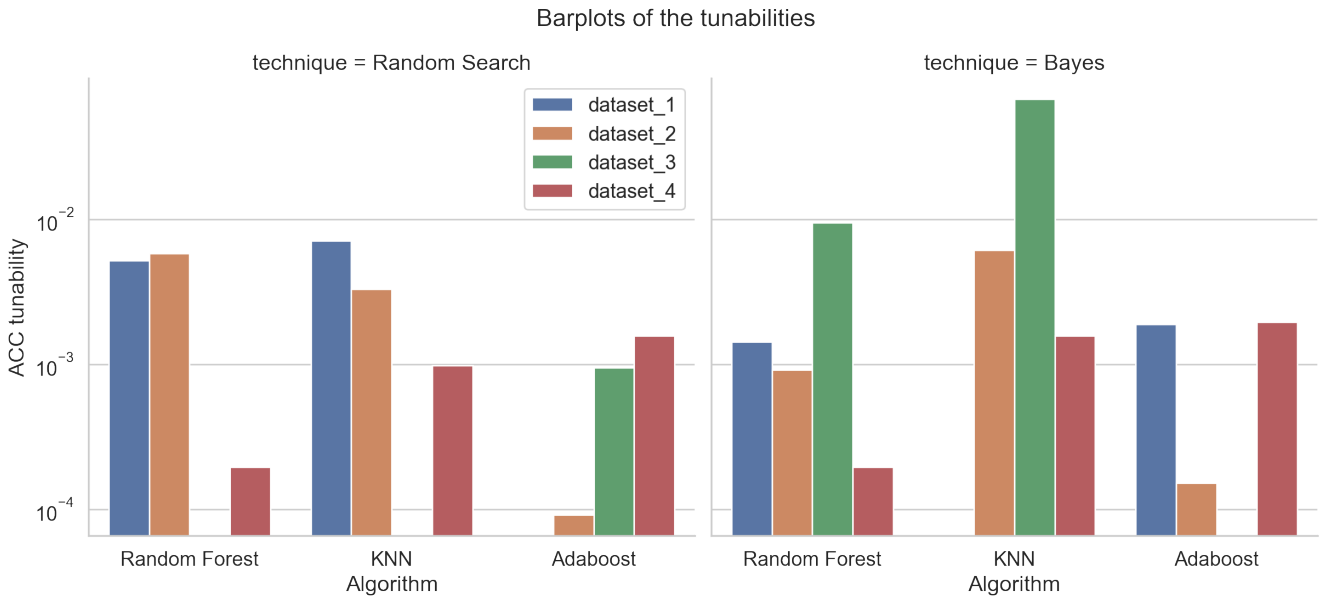
5 Porównanie tunowalności algorytmów

algorytm	hiperparametr	random search θ^*	bayes θ^*
Random Forest	`n_estimators`	92	300
	`criterion`	log_loss	entropy
	`max_depth`	11	9
	`min_samples_split`	12	9
	`min_samples_leaf`	4	5
	`max_features`	None	sqrt
K-Nearest Neighbors	`n_neighbors`	10	18
	`p`	0.31	0.001
	`weights`	uniform	uniform
AdaBoost	`n_estimators`	423	50
	`learning_rate`	0.10	0.45
	`algorithm`	SAMME.R	SAMME.R

Tabela 3: Wyznaczone deafulty konfiguracji hiperparametrów w zależności od metody losowania punktów

Zgodnie z poleceniem, badaliśmy tunowalność poszczególnych algorytmów. Najpierw wyznaczyliśmy najlepsze zestawy hiperparametrów dla każdego zbioru (nie umieszczone w raporcie z powodu objętości) oraz najlepsze zestawy hiperparametrów dla ogólnie, czyli takie, które pozwoliły osiągnąć "najlepszy średni wynik". Te zestawy hiperparametrów znajdują się w Tabeli 3. Co ciekawe, otrzymane zestawy hiperparametrów mocno różnię się dla różnych technik losowania punktów.
Ponieważ liczba zbiorów danych na których przeprowadzaliśmy eksperymenty jest znacznie mniejsza niż liczba zbiorów użytych w artykule omawianym na zajęciach, box plot do wizualizacji tunowalności nie sprawdzi się. Z tego też powodu użyliśmy wykresów słupkowych, dzięki czemu łatwo można wywnioskować jak wysoka wyszła tunowalność na danych zbiorach oraz ocenić jaka jest tendencja tunowalnosci dla danego algorytmu. Wykres

przedstawiony jest na rysunku 4. Tunowalność poszczególnych algorytmów na różnych zbiorach danych osiąga czasami znacząco różne wartości, dlatego wykorzystana jest skala logarytmiczna.



Rysunek 4: Porównanie tunowalności algorytmów na różnych zbiorach danych

6 Podsumowanie

Tunowalność algorytmów dla zbiorów danych na których przeprowadzone zostały eksperymenty wyszła bardzo mała, a na niektórych z nich nawet znikoma. Nie musi to jednak oznaczać, że algorytmy w ogólności cechują się małą tunowalnością. Zaobserwowane wnioski mogą wynikać z kilku przyczyn, jedną z nich może być zbyt mała ilość zbiorów danych. Same zbiory danych nie były też bardzo skomplikowane. Na wykresach *accuracy* od liczby iteracji widać, że modele od samego początku cechowały się wysoką wartością tej metryki. Taki fakt źle wpływa na dalsze możliwości tunowania hiperparametrów.

Bibliografia

[1] Eva Bartz i in. *Experimental Investigation and Evaluation of Model-based Hyperparameter Optimization*. 2021. arXiv: 2107.08761 [cs.LG].

[2] Jan N. van Rijn i Frank Hutter. "Hyperparameter Importance Across Datasets". W: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining*. KDD '18. ACM, lip. 2018. DOI: 10.1145/3219819.3220058. URL: http://dx.doi.org/10.1145/3219819.3220058.