Analiza tunowalności wybranych algorytmów uczenia maszynowego

Piotr Tyrakowski, Dorota Rzewnicka

8 grudnia 2024

1 Wstęp

Współczesne algorytmy uczenia maszynowego charakteryzują się obecnością wielu hiperparametrów, których odpowiedni dobór może znacząco wpłynąć na jakość modelu. Tunowalność algorytmów, czyli wrażliwość modelu na zmianę wartości hiperparametrów, jest zatem istotnym aspektem w praktyce modelowania.

Celem niniejszej pracy jest analiza tunowalności trzech wybranych algorytmów: XGBoost, Random Forest oraz Extra Trees. Analiza została przeprowadzona na czterech różnych zbiorach danych z wykorzystaniem dwóch technik losowania punktów hiperparametrów: Random Search oraz optymalizacji bayesowskiej. Na podstawie uzyskanych wyników oceniono wpływ tuningu na jakość modeli oraz zbadano wpływ wybranych metod samplingu na tunowalność algorytmów i hiperparametrów.

2 Dane i algorytmy

2.1 Zbiory danych

Do analizy wykorzystano następujące zbiory danych:

1. **Heart Disease** (Heart Disease Dataset): Zbiór danych dotyczących chorób serca zawierający 303 obserwacje i 14 cech.
2. **Diabetes** (OpenML): Zbiór danych dotyczący cukrzycy, zawierający 768 obserwacji i 8 cech.
3. **Spambase** (OpenML): Zbiór danych dotyczący klasyfikacji wiadomości jako spam lub nie-spam, zawierający 4601 obserwacji i 57 cech.
4. **Phishing Websites** (OpenML): Zbiór danych dotyczący klasyfikacji stron internetowych jako phishing lub legalne, zawierający 2456 obserwacji i 30 cech.

2.2 Algorytmy i hiperparametry

W analizie wykorzystano trzy algorytmy uczenia maszynowego:

- **Random Forest**: Algorytm ensemble wykorzystujący wiele drzew decyzyjnych.

- **XGBoost**: Implementacja gradientowego boostingu drzew decyzyjnych.
- **Extra Trees**: Algorytm podobny do Random Forest, ale z losowym podziałem punktów podziału.

Zakresy hiperparametrów dla każdego algorytmu zostały ustalone na podstawie literatury i dokumentacji biblioteki `scikit-learn`:

- **Random Forest**:
  - `n_estimators`: [50, 100, 150, 200, 250, 300, 350, 400]
  - `criterion`: ['gini', 'entropy', 'log_loss']
  - `max_depth`: [5, 10, 15, ..., 50]
  - `min_samples_leaf`: [1, 2, ..., 14]
  - `min_samples_split`: [2, 3, ..., 14]
  - `max_features`: ['sqrt', 'log2', None]
- **XGBoost**:
  - `n_estimators`: [50, 100, 150, 200, 250, 300, 350, 400]
  - `learning_rate`: [0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3]
  - `max_depth`: [3, 4, ..., 14]
  - `subsample`: [0.5, 0.6, ..., 1.0]
- **Extra Trees**:
  - `n_estimators`: [50, 100, 150, 200, 250, 300, 350, 400]
  - `criterion`: ['gini', 'entropy', 'log_loss']
  - `max_depth`: [5, 10, 15, ..., 50]
  - `min_samples_leaf`: [1, 2, ..., 14]
  - `min_samples_split`: [2, 3, ..., 14]
  - `max_features`: ['sqrt', 'log2', None]

3 Metody

3.1 Techniki losowania punktów

W celu tuningu algorytmów zastosowano dwie techniki losowania punktów:

- **Random Search (RS)**: Metoda polegająca na losowym przeszukiwaniu przestrzeni hiperparametrów.
- **Optymalizacja bayesowska (BO)**: Metoda wykorzystująca modele probabilistyczne do kierowania przeszukiwaniem przestrzeni hiperparametrów. Zastosowano funkcję `BayesSearchCV` z pakietu `scikit-optimize`.

3.2 Procedura eksperymentalna

Dla każdego algorytmu i zbioru danych przeprowadzono następujące kroki:

1. Podział danych na zbiór treningowy (70%) i testowy (30%).
2. Przeprowadzenie tuningu za pomocą Random Search z 100 iteracjami.
3. Przeprowadzenie tuningu za pomocą optymalizacji bayesowskiej z 50 iteracjami (liczba iteracji została ograniczona ze względu na czas wykonania kodu).
4. Ocena modeli na zbiorze testowym za pomocą miary *balanced accuracy*.
5. Analiza tunowalności algorytmów zgodnie z definicjami z artykułu (1).

4 Wyniki

Ze względu na ograniczenia objętości raportu, przedstawiono skrótowo wyniki dla każdego algorytmu i zbioru danych.

4.1 Defaultowe parametry

Defaultowe parametry zostały wyznaczone poprzez wyznaczenie takich parametrów, dla których Random Search uzyskał średnio najlepsze wyniki. Były to:

- dla lasu losowego: `'n_estimators': 350, 'min_samples_split': 9, 'min_samples_leaf': 1, 'max_features': 'sqrt', 'max_depth': 25, 'criterion': 'entropy'`
- dla XGBoost: `'subsample': 0.5, 'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.01`
- dla Extra Trees: `''n_estimators': 50, 'min_samples_split': 9, 'min_samples_leaf': 1, 'max_features': 'log2', 'max_depth': 25, 'criterion': 'entropy'`.

4.2 Tunowalność

Tunowalność została wyliczona jako różnica średniej wartości miary w kroswalidacji pomiędzy defaultowymi parametrami a najlepszymi parametrami określonymi dla danego zbioru.

Tabela 1: Tunowalność dla różnych algorytmów i metod losowania punktów

	Random Forest		XGBoost		Extra Trees	
	RS	BO	RS	BO	RS	BO
Dane 1	−0.016	−0.012	0.000	0.0002	−0.008	−0.019
Dane 2	−0.010	−0.006	−0.013	−0.016	−0.021	−0.020
Dane 3	−0.024	−0.025	−0.005	−0.006	−0.001	−0.004
Dane 4	0.0	0.0	−0.014	−0.010	−0.015	−0.015

Ujemne wyniki świadczą o uzyskaniu wyników lepszych niż przy defaultowych parametrach. Im większa wartość bezwzględna, tym różnica jest bardziej znacząca.

4.3 Analiza liczby iteracji

Zazwyczaj optymalizacja bayesowska osiąga nieco wyższe wyniki w mniejszej liczbie iteracji, która jest zależna również od zbioru danych.

- Random Search wymaga około 70 iteracji, aby osiągnąć stabilne wyniki.
- Optymalizacja bayesowska osiąga stabilne wyniki już po około 30-40 iteracjach.

Ilustrujące to wykresy znalazły się w sekcji 8.1.

5 Bias sampling

Różnice we wnioskach dotyczących tunowalności mogą wynikać z zastosowanej metody samplingu. Random Search losuje punkty z rozkładu jednostajnego, podczas gdy optymalizacja bayesowska kieruje poszukiwania w obszary bardziej obiecujące. Sprawia to, że zazwyczaj przy stosowaniu BO różnice pomiędzy wynikami defaultowymi a uzyskanymi w danej iteracji są mniej rozproszone w porównaniu do losowania z siatki hiperparametrów. Przedstawiające to zjawisko wykresy znajdują się w sekcji 8.2.

6 Wnioski

Przeprowadzone eksperymenty wskazują, że tuning hiperparametrów jest istotny dla poprawy jakości modeli uczenia maszynowego. Metoda optymalizacji bayesowskiej okazała się bardziej efektywna niż Random Search, osiągając lepsze wyniki w mniejszej liczbie iteracji.

Wyniki dotyczące tunowalności algorytmów pokazały, że zależy ona od wybranego algorytmu, jak i w dużym stopniu od zbioru danych.

7 Bibliografia

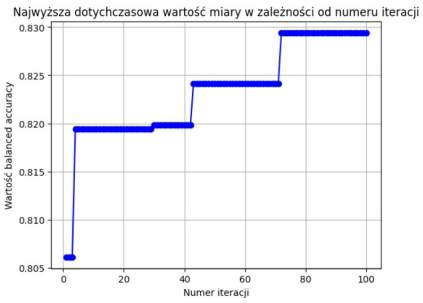
[1] Probst, P., Wright, M. N., & Boulesteix, A. L. (2019). Tunability: Importance of Hyperparameters of Machine Learning Algorithms. *Journal of Machine Learning Research*, 20(53), 1-32.

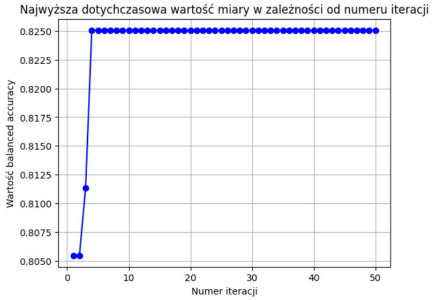
[Heart Disease Dataset] UCI Machine Learning Repository: Heart Disease Data Set. `https://archive.ics.uci.edu/ml/datasets/heart+disease`

[OpenML] OpenML: Machine Learning Datasets. `https://www.openml.org/`

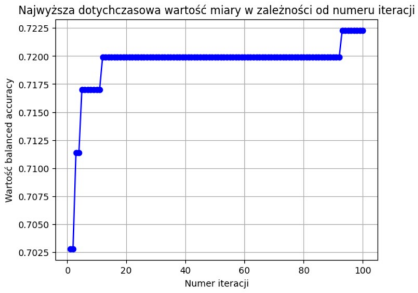
8 Wykresy

8.1 Analiza liczby iteracji

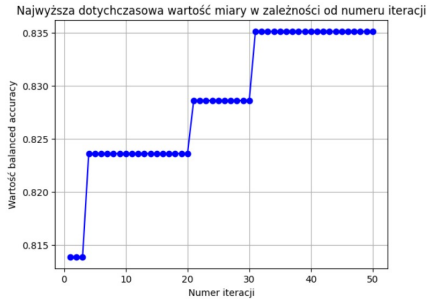




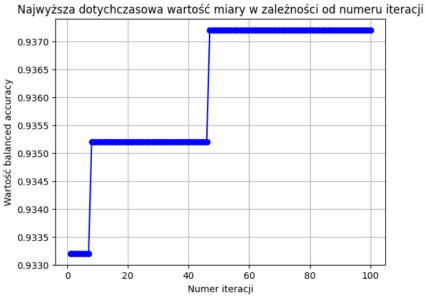
Rysunek 1: Random forest na zbiorze 1, po lewej RS, po prawej BO.

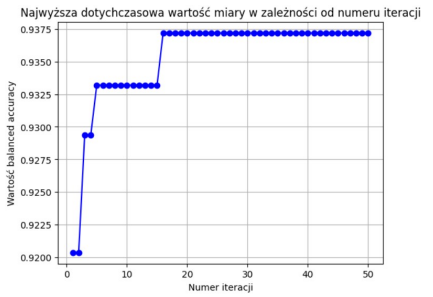


Rysunek 2: Random forest na zbiorze 2, po lewej RS, po prawej BO.

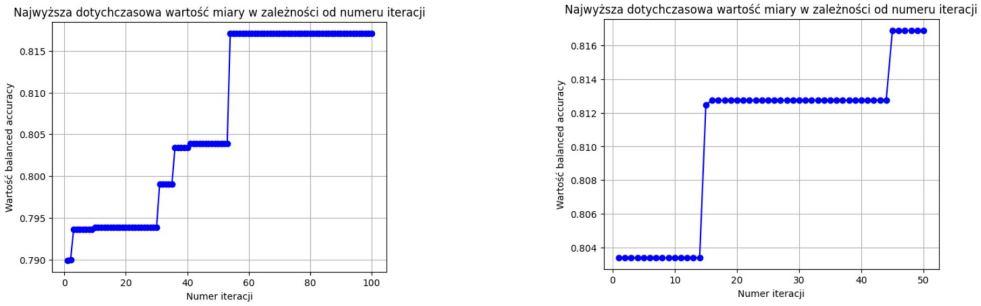


Rysunek 3: Random forest na zbiorze 3, po lewej RS, po prawej BO.

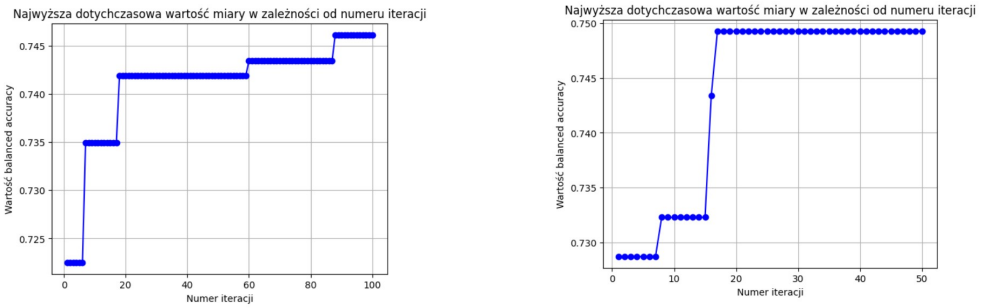




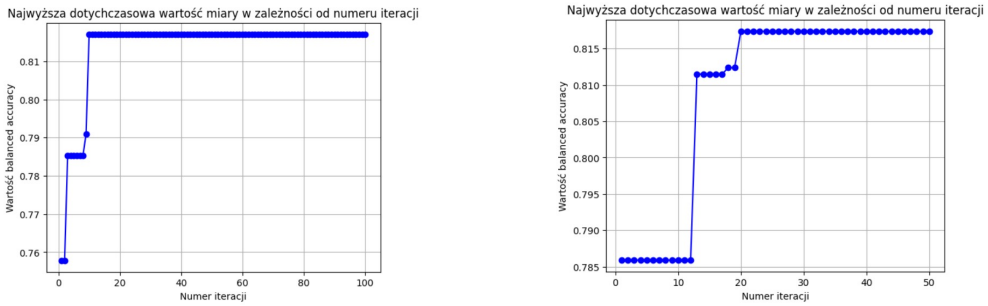
Rysunek 4: Random forest na zbiorze 4, po lewej RS, po prawej BO.



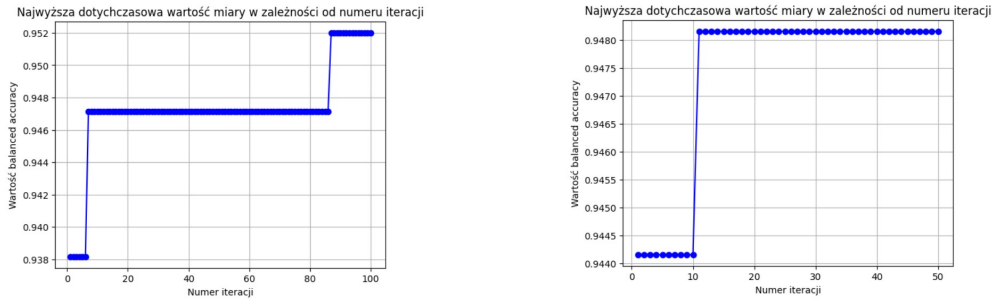
Rysunek 5: XGBoost na zbiorze 1, po lewej RS, po prawej BO.



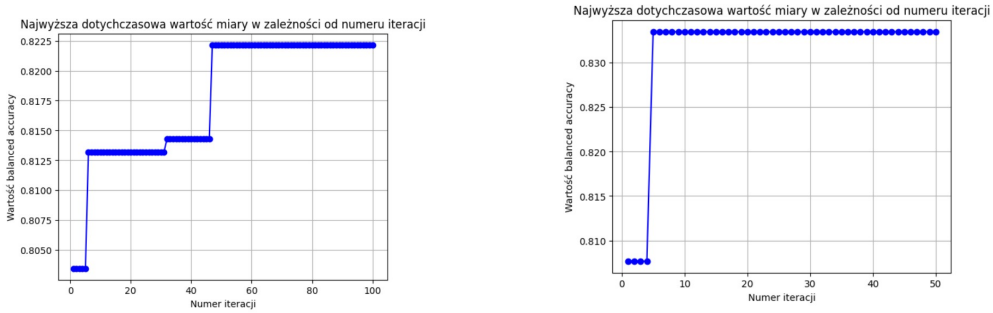
Rysunek 6: XGBoost na zbiorze 2, po lewej RS, po prawej BO.



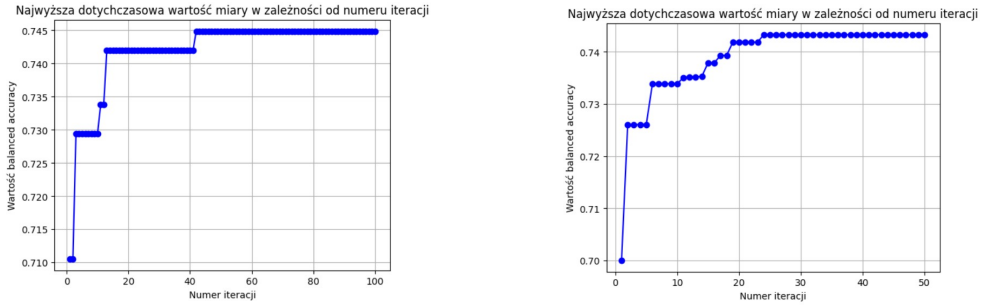
Rysunek 7: XGBoost na zbiorze 3, po lewej RS, po prawej BO.



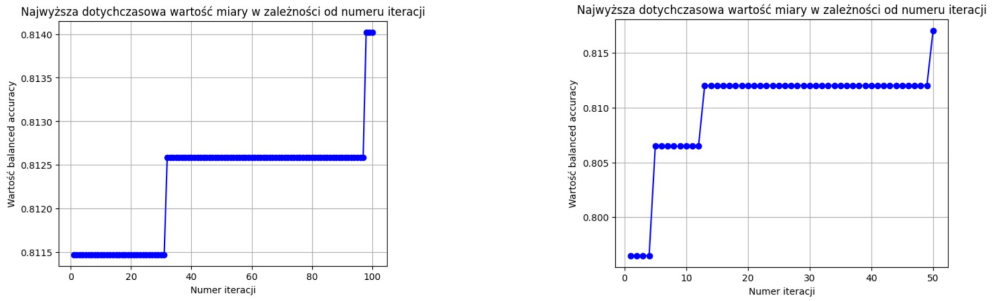
Rysunek 8: XGBoost na zbiorze 4, po lewej RS, po prawej BO.



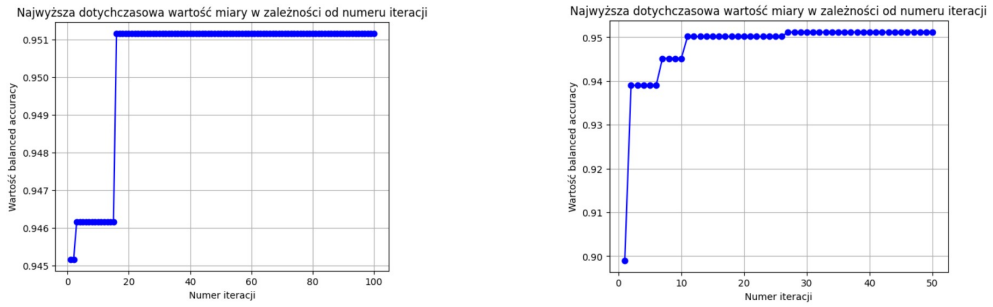
Rysunek 9: Extra trees na zbiorze 1, po lewej RS, po prawej BO.



Rysunek 10: Extra trees na zbiorze 2, po lewej RS, po prawej BO.



Rysunek 11: Extra trees na zbiorze 3, po lewej RS, po prawej BO.

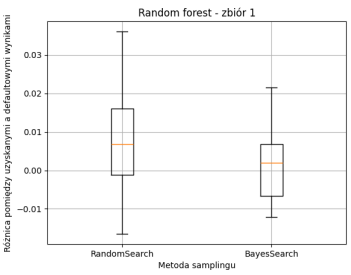


Rysunek 12: Extra trees na zbiorze 4, po lewej RS, po prawej BO.

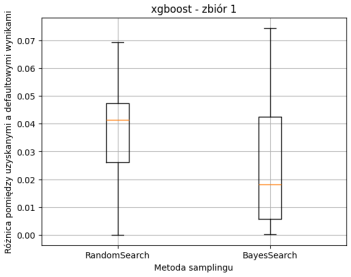
8.2 Boxploty

Poniżej przedstawiono boxploty ilustrujące różnicę pomiędzy wynikami uzyskanymi za pomocą tuningu a wynikami przy domyślnych parametrach. Każdy wykres pokazuje dane dla jednej metody i jednego zbioru danych.

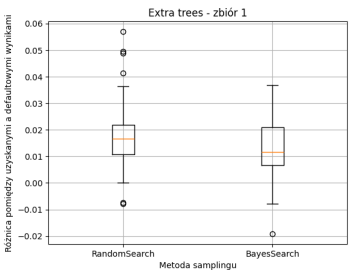
8.2.1 Dataset 1



(a) Random Forest - Dataset 1



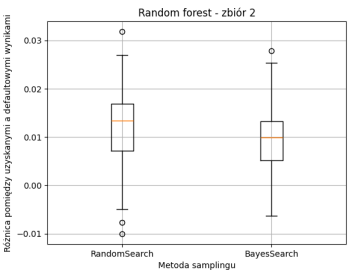
(b) XGBoost - Dataset 1



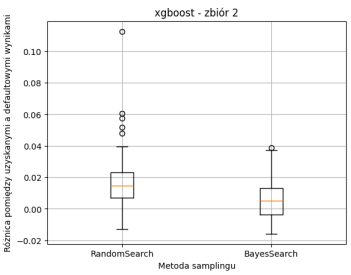
(c) Extra Trees - Dataset 1

Rysunek 13: Boxploty dla Dataset 1

8.2.2 Dataset 2



(a) Random Forest - Dataset 2

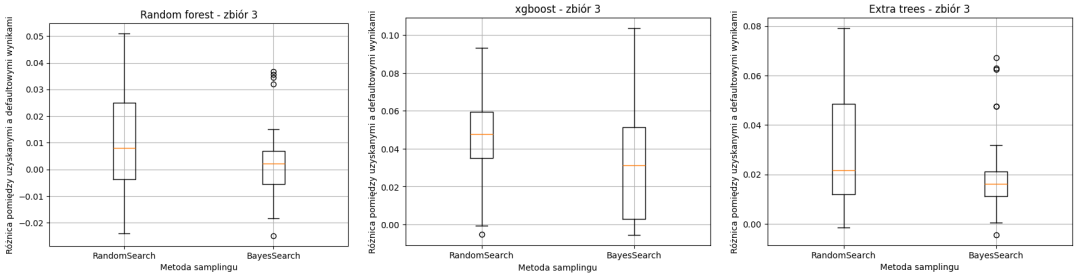


(b) XGBoost - Dataset 2

(c) Extra Trees - Dataset 2

Rysunek 14: Boxploty dla Dataset 2

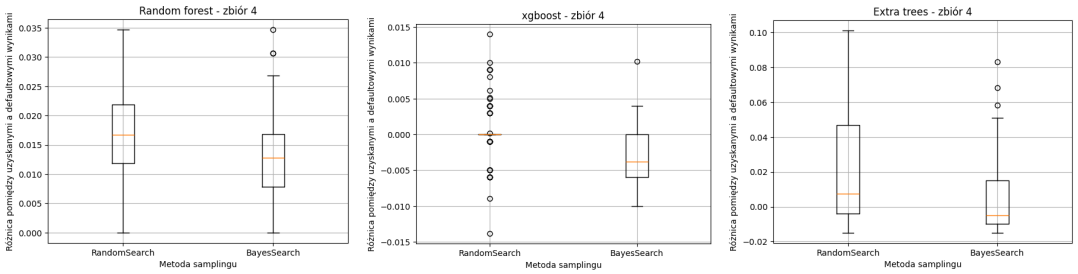
8.2.3 Dataset 3



(a) Random Forest - Dataset 3 (b) XGBoost - Dataset 3 (c) Extra Trees - Dataset 3

Rysunek 15: Boxploty dla Dataset 3

8.2.4 Dataset 4



(a) Random Forest - Dataset 4 (b) XGBoost - Dataset 4 (c) Extra Trees - Dataset 4

Rysunek 16: Boxploty dla Dataset 4