Raport AutoML

Franciszek Filipek, Julia Girtler, Jan Skalski

November 2025

1 Cel projektu

Celem projektu było przeanalizowanie tunowalności wybranych algorytmów uczenia maszynowego (regresja logistyczna, lasy losowe, sieci neuronowe) wykorzytsując dwie techniki losowania punktów. Dodatkowo przeanalizowany został wpływ wielkości zbioru danych.

2 Dane

W analizie wykorzystano pięć zestawów danych z repozytorium OpenML służących do zadań klasyfikacji binarnej:

- Phoneme - dane akustyczne do rozróżniania dwóch typów fonemów; ok. 5,400 obserwacji, 5 cech.
- Bank Marketing - dane o kampanii marketingowej portugalskiego banku; ok. 45,000 obserwacji, 17 cech.
- Blood Transfusion - dane o dawcach krwi z centrum krwiodawstwa w Tajwanie; 748 obserwacji, 4 cechy.
- QSAR Biodegradation - dane chemiczne o biodegradowalności związków; 1,055 obserwacji, 41 cech.
- Phishing Websites - dane o stronach internetowych służące do wykrywania phishingu; ok. 11,000 obserwacji, 30 cech.

Przed przystąpieniem do badania tunowalności algorytmów utworzono dwa pipeline'y - osobny dla zmiennych numerycznych oraz dla zmiennych kategorycznych. Dla zmiennych numerycznych brakujące wartości uzupełniano medianą, a następnie dane standaryzowano przy użyciu metody `StandardScaler`. W przypadku zmiennych kategorycznych brakujące wartości zastępowano najczęściej występującą kategorią, po czym zastosowano `one-hot encoding`. Oba pipeline połączono za pomocą `ColumnTransformer`.

3 Tunowanie algorytmów

3.1 Tunowanie za pomocą Grid Search

Zdefiniowane zostały siatki hiperparametrów dla trzech wymienionych wyżej modeli.
Funkcja `grid_tuning_per_dataset_and_model` pobiera wskazany zbiór danych z OpenML i przygotowuje go do analizy poprzez wymieszanie obserwacji, binarną konwersję etykiet oraz podział na część treningową i testową. Dodatkowo umożliwia trenowanie modelu jedynie na zadanej frakcji zbioru treningowego.

Następnie stosowany jest opisany wcześniej pipeline. Strojenie hiperparametrów odbywa się za pomocą metody `Grid Search` z czterokrotną walidacją krzyżową, a kryterium oceny stanowi zbalansowana dokładność. Po wyznaczeniu najlepszej konfiguracji model jest oceniany na wcześniej odłożonym zbiorze testowym.

3.2 Tunowanie za pomocą Optymalizacji Bayesowskiej

Funkcja `bayes_tuning_per_dataset_and_model` jest zdefiniowana analogicznie do poprzedniej. Jednak w odróżnieniu od funkcji wykorzystującej Grid Search, która sprawdza wszystkie kombinacje hiperparametrów zdefiniowane w siatce, druga funkcja wykorzystuje optymalizację bayesowską dobierając kolejne konfiguracje w sposób adaptacyjny. Zamiast przeszukiwać całą przestrzeń parametrów, algorytm wykorzystuje wcześniejsze wyniki ewaluacji, aby wybierać kolejne punkty, które mają największą szansę poprawić wydajność modelu. Proces ten jest ograniczony maksymalnie do 100 iteracji. Optymalizacja Bayesowska została przeprowadzona za pomocą pakietu SMAC3.

3.3 Dalsza analiza

Wszystkie wyniki z powyższych funkcji zapisywane są do plików csv, które wykorzytsujemy w późniejszej analizie. Funkcje `read_tuning_data` i `read_tuning_data_bayes` wczytują wyniki z plików CSV, łączą je w jedną tabelę i konwertują parametry do odpowiednich typów, tak aby można było je porównywać między różnymi datasetami. Następnie funkcja `get_optimal_default_configurations_for_model` analizuje wszystkie odczytane wyniki, oblicza średnią wartość zbalansowanej dokładności dla każdej konfiguracji na wielu zbiorach danych i wybiera konfigurację o najwyższej średniej wydajności. Wybrany zestaw hiperparametrów jest zapisywany do plików CSV, a funkcja `train_and_evaluate_best_overall_config` dodatkowo trenuje model z tą konfiguracją na każdym zbiorze danych, oceniając go na zbiorze testowym i zapisując końcowe wyniki.

4 Wyniki i wizualizacje

4.1 Tabele

W tabelach przedstawiono najlepsze zestawy hiperparametrów wybrane przez Grid Search oraz optymalizację bayesowską dla różnych frakcji danych treningowych. Dla każdej frakcji pokazano wartości poszczególnych parametrów uzyskane obiema metodami oraz odpowiadające im średnie wyniki zbalansowanej dokładności (BA) w walidacji krzyżowej.

	max_depth		min_samples_leaf		min_samples_split		n_estimators		mean_cv_BA	
fraction	**Grid**	**Bayes**	**Grid**	**Bayes**	**Grid**	**Bayes**	**Grid**	**Bayes**	**Grid**	**Bayes**
0.25	25	15	0.01	0.022	2	3	200	147	0.716	0.698
0.50	10	7	0.01	0.012	17	24	100	71	0.742	0.736
0.75	10	None	0.01	0.011	17	36	150	151	0.739	0.729
1.0	15	None	0.01	0.011	7	36	50	151	0.734	0.726

Tabela 1: Porównanie parametrów uzyskanych z Grid Search i optymalizacji Bayesowskiej w zależności od frakcji danych dla **Random Forest**.

	activation		alpha		hidden_layer_sizes		mean_cv_BA	
fraction	**Grid**	**Bayes**	**Grid**	**Bayes**	**Grid**	**Bayes**	**Grid**	**Bayes**
0.25	relu	relu	0.001	8.017	(100, 100)	(100, 100)	0.783	0.661
0.50	relu	relu	0.016	8.017	(100, 50)	(100, 100)	0.814	0.683
0.75	relu	relu	0.001	8.017	(100,)	(100, 100)	0.799	0.700
1.00	relu	relu	0.125	8.017	(100, 50)	(100, 100)	0.813	0.693

Tabela 2: Porównanie wyników Grid Search i optymalizacji Bayesowskiej w zależności od frakcji danych dla **Sieci Neuronowej**.

	C		l1_ratio		penalty		mean_cv_BA	
fraction	**Grid**	**Bayes**	**Grid**	**Bayes**	**Grid**	**Bayes**	**Grid**	**Bayes**
0.25	4.0	273.284	0.9	0.597	elasticnet	l1	0.742	0.736
0.50	2.0	22.195	0.5	0.244	elasticnet	elasticnet	0.741	0.740
0.75	2.0	5.234	0.25	0.229	elasticnet	elasticnet	0.726	0.731
1.00	2.0	5.234	0.1	0.229	l2	elasticnet	0.731	0.728

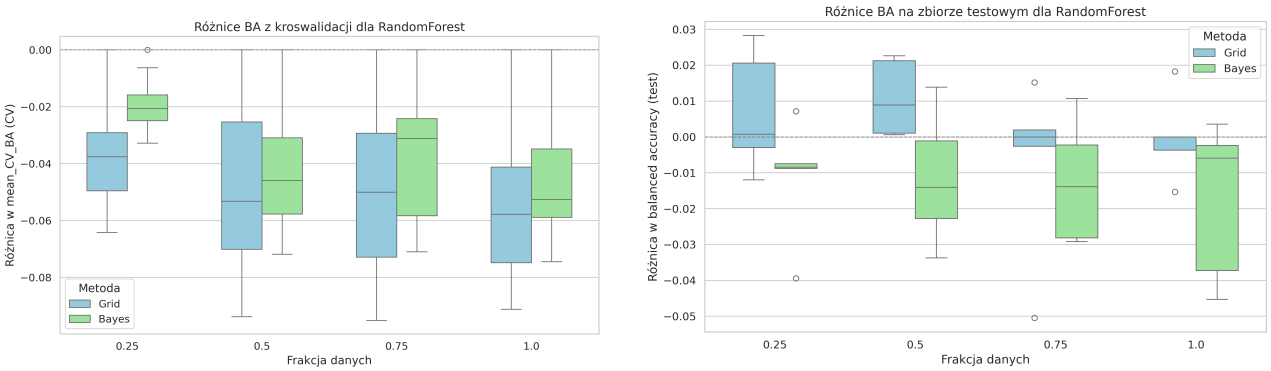
Tabela 3: Porównanie wyników Grid Search i optymalizacji Bayesowskiej w zależności od frakcji danych dla **Regresji Logistycznej**.

4.2 Wykresy

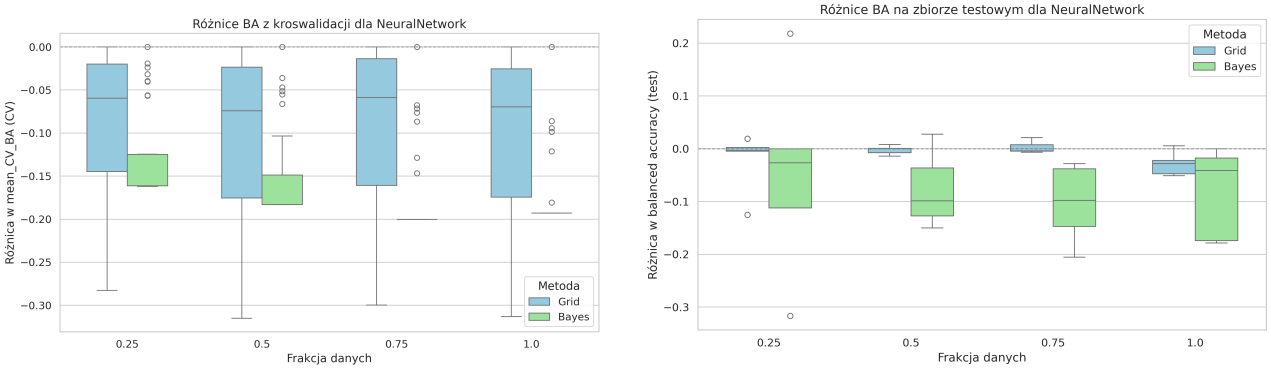
Na poniższych wykresach przedstawiono dwa różne rozkłady dla każdego algorytmu:

1. Boxploty po lewej przedstawiają rozkład różnic uśrednionych wyników z kroswalidacji z historii uczenia, a najlepszym uśrednionym wynikiem w podziale na frakcje i rodzaj metody tunowania (stąd wartości mniejsze równe 0).

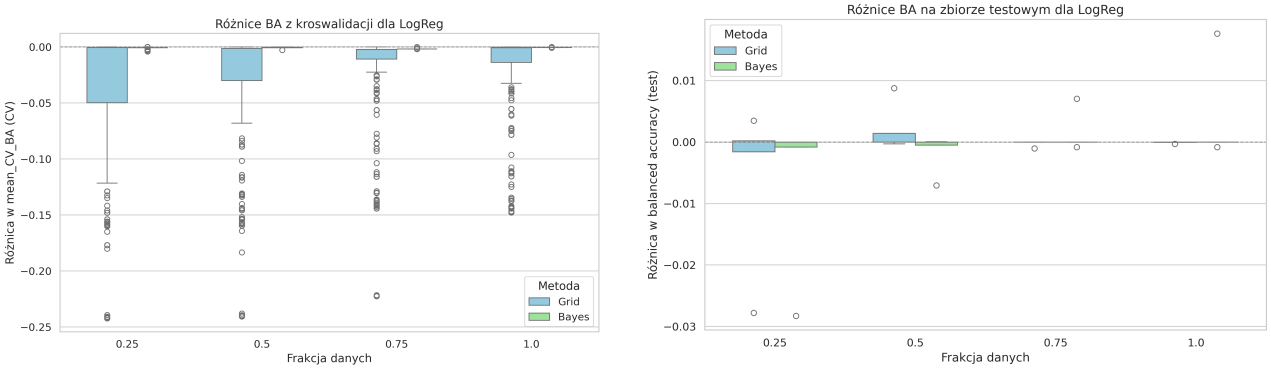
2. Boxploty po prawej przedstawiają rozkład różnic wyników dla najlepszych konfiguracji z kroswalidacji dla konkretnych zbiorów dancyh a wynikami uśrednionej najlepszej konfiguracji na tych zbiorach danych (każdy boxplot składa się jedynie z 5 punktów).



Rysunek 1: Porównanie dla **Random Forest**.



Rysunek 2: Porównanie wyników dla **Sieci Neuronowej**.



Rysunek 3: Porównanie wyników dla **Regresji Logistycznej**.

5 Wnioski

5.1 Wnioski na podstawie wyników modeli

1. Za pomocą Grid Searcha otrzymywane były wyższe średnie zbalansowane dokładności. Szczególną różnicę widać przede wszystkim na modelach Sieci Neuronowej, gdzie średnia zbalansowana dokładność w optymalizacji bayesowskiej dla każdej frakcji waha się w okolicach 0.7, gdzie dla Grid Searcha wynosi ona nawet 0.81.

2. Frakcja nie ma znaczącego wpływu na średnią zbalansowaną dokładność. Metody tunowania zachowują się podobnie niezależnie od części zbioru danych, na której modele są uczone.

3. Parametry wybierane przy optymalizację bayesowską widocznie różnią się od tych wybieranych przez Grid Search. Przykładowo w Sieci neuronowej Bayes wybiera alpha większą ok. 10 tysięcy razy.

4. Optymalizacja bayesowska wybiera modele Random Forest z mniejszą liczbą drzew, co zwiększa wariancję i zmniejsza stabilność predykcji. Mniejszy las jest bardziej podatny na przypadkowe fluktuacje danych, dlatego jego wyniki są mniej stabilne i zwykle nieco gorsze niż w przypadku modeli z większą liczbą drzew wybieranych przez Grid Search.

5. Optymalizacja bayesowska dla Sieci Neuronowej ma identyczne hiperparametry dla każdej frakcji. Może to sugerować, że przy pierwszych kilku próbach trafiono na zestaw, który wygląda wystarczająco dobrze i nie kontynuowano dalszych poszukiwań. Grid search okazał się lepszy, dzięki pzeszukaniu pełnej siatki danych.

6. Wyniki Regresji Logistycznej wskazują na to, że dobór hiperparametrów nie ma znaczenia dla średniej zbalansowanej dokładności. Metoda Grid Search, jak i optymalizacja bayesowska są równoważne dla tego modelu.

5.2 Wnioski z analizy wykresów

1. Z boxplotów prezentujących różnice BA z kroswalidacji punkty w Grid Search mają zdecydowanie większy rozrzut. Oznacza to, że optymalizacja bayesowska szybciej znajduję optymalne według niej parametry, podczas gdy Grid Search przeszukuje całą siatkę.

2. Dla Random Forest rozrzut różnic BA z kroswalidacji jest spory dla obu metod losowania. Oznacza to, że kolejne iteracje poprawiają wyniki. W obu optymalizacjach najmniejsza frakcja wiązała się z mniejszymi różnicami BA

3. Dla Sieci Neuronowej można zauważyć, że optymalizacjia bayesowska szybko znajduję zestaw hiperparametrów, a kolejne iteracje nie zwiększają BA. Przeciwnie jest dla Grid Searcha, gdzie rozrzut wyników jest zauważalny.

4. Dla Regresji logistycznej przy optymalizacji bayesowskiej dobór parametrów nie ma za dużego znaczenia, ponieważ różnice są bardzo bliskie 0. Jest to spójne z wcześniejszym wnioskiem z tabeli. Rozrzut punktów przy Grid Search jest mniejszy niż w dwóch poprzednich algorytmach. Można przypuszczać, że Regresja Logistyczna nie jest podatna na tunowanie, a wyniki dla tego modelu nie zależą od doboru hiperparametrów bez kontekstu konkretnych dancyh.

5. Skala w wynikach Random Forest BA na zbiorze testowym sugeruje, że najlepsze hiperparametry działają dobrze dla każdego z pięciu zbiorów. Jeszcze mocniejszy efekt jest dla Regresji Logistycznej, jednak tam może wynikać on z niewielkiego wpływ tunowania hiperparametrów na wyniki.

6. W przypadku Sieci Neuronowej różnice BA na zbiorze testowym są zauważalne. Można przypuszczać, że dobrane hiperparametry nie adaptują się dobrze dla różnych zbiorów danych.

6 Podsumowanie

Celem projektu było porównanie skuteczności różnych metod tunowania hiperparametrów (Grid Search i optymalizacja bayesowska) w trzech modelach: Sieci Neuronowej, Random Forest i Regresji Logistycznej. Analiza wykazała, że Grid Search zazwyczaj daje wyższe i bardziej stabilne wyniki, zwłaszcza w modelach złożonych, jak Sieci Neuronowe, natomiast optymalizacja bayesowska szybciej znajduje parametry, co jest szczególnie widoczne w mniej wrażliwych modelach, np. Regresji Logistycznej. Frakcja danych nie miała istotnego wpływu na wyniki. Skuteczność tunowania hiperparametrów jest uzależniona od rodzaju modelu i jego podatności na zmiany parametrów.