Analiza tunowalności hiperparametrów wybranych algorytmów uczenia maszynowego

Maksim Razantsau, Oleksii Vinichenko, Yahor Lahunovich

16 listopada 2025

1 Wstęp

Celem projektu jest analiza tunowalności hiperparametrów dla trzech wybranych algorytmów uczenia maszynowego: CatBoost, XGBoost i LightGBM. Zdecydowałyśmy się je wykorzystać, aby dodać do naszego projektu element eksperymentu i zbadać różnice między tymi trzema powszechnie używanymi algorytmami boostingowymi. Tunowalność wymienionych algorytmów została sprawdzona przy użyciu pięciu zbiorów danych pochodzących ze strony OpenML i przeznaczonych do klasyfikacji binarnej. Dodatkowo analizowałyśmy wyniki modeli w poszczególnych iteracjach oraz porównywałyśmy te wyniki z wynikami modeli o optymalnych hiperparametrach.

2 Opis danych

Wybraliśmy zróżnicowane zbiory danych różnych rozmiarów do klasyfikacji binarnej:

Adult (49k × 15) - zbiór danych spisowych, używany do przewidywania, czy roczny dochód osoby przekracza 50 tys. dolarów.

Numarai (96k × 22) - zanonimizowane dane finansowe z turnieju Numerai, służące do przewidywania trendów giełdowych.

MagicTelescope (19k × 12) - dane z teleskopu MAGIC, używane do klasyfikacji promieniowania gamma od hadronów.

kr-vs-kp (3.2k × 37) - zbiór opisujący pozycje szachowe w końcówce "Król i Wieża kontra Król i Pion". Służy do klasyfikacji, czy pozycja jest wygrana.

Mushroom (8.1k × 23) - zbiór zawierający cechy fizyczne grzybów. Służy do klasyfikacji, czy grzyb jest jadalny, czy trujący.

3 Transformacja zmiennych

Na początku przeprowadziłyśmy wstępną analizę danych w wybranych zbiorach. Pewne zbiory zawierały braki danych, a więc przez nas został zbudowany jeden pipeline, składający się z SimpleImputer oraz OneHotEncoder do trnsformacji zmiennych.

4 Zakresy hiperparametrów

Dla każdego algorytmu wybraliśmy 6 hiperparametrów do eksperymentu. W celach porównawczych, wybrałyśmy 5 wspólnych hiperparametrów dla każdego algorytmu: `learning_rate`, `n_estimators`, `max_depth`, `subsample`, `colsample_bylevel` oraz jeden specyficzny dla danego

algorytmu: `gamma` dla XGBoost, `num_leaves` dla LightGBM oraz `l2_leaf_reg` dla CatBoost. Te hiperparametry zostały wybrane bazując na artykule *„Tunability: Importance of Hyperparameters of Machine Learning Algorithms"* oraz na dokumentacji algorytmów. Dla każdego zestawu danych mieliśmy taką samą siatkę hiperparametrów.

- `learning_rate` **[loguniform(1e-3, 0.3)]**: zgodnie z literaturą, XGBoost jest wrażliwy na ten parametr, który silnie wpływa na tunowalność.
- `n_estimators` **[randint(100, 2000)]**: liczba drzew w modelu kontroluje jego złożoność. Więcej nie zawsze lepiej, bo powoduje przeuczenie się, a więc jest istotne do zbadania.
- `max_depth` **[randint(2, 12)]**: kontroluje stopień skomplikowania pojedynczego drzewa. Zbyt duża głębokość prowadzi do przeuczenia, podczas gdy zbyt mała może skutkować niedouczeniem.
- `subsample` **uniform(0.5, 0.5)**: określa procent próbek używanych do budowy każdego drzewa. Wartość mniejsza niż 1.0 wprowadza losowość i pomaga w redukcji wariancji oraz zapobieganiu przeuczeniu się.
- `colsample_bylevel` **[uniform(0.5, 0.5)]**: kontroluje frakcję cech używanych przy każdym podziale drzewa. Jest to kolejna technika regularyzacji, która zmniejsza ryzyko overfittingu i jest szczególnie użyteczna, gdy zbiór danych ma wiele cech.
- `gamma` (dla XGBoost) **[loguniform(1e-8, 10)]**: określa minimalną redukcję straty wymaganą do wykonania dalszego podziału węzła.
- `num_leaves` (dla LightGBM) **[randint(16, 256)]**: główny parametr kontrolujący złożoność modelu w LightGBM, który buduje drzewa metodą *leaf-wise*, która bezpośrednio ogranicza liczbę liści w drzewie.
- `l2_leaf_reg` (dla CatBoost) **[loguniform(0.1, 10)]**: współczynnik regularyzacji L2 dla wartości w liściach.

5 Metryki

Główną metryką jaką używaliśmy do badania optymalności otrzymanych wyników oraz prównywania wybranych technik optymalizacji jest ROC AUC.

6 Eksperyment

6.1 Wybrane metody losowania

W celu przeprowadzenia eksperymentu, który pozwolił nam zbadać tunowalność hiperparametrów zastosowałyśmy dwie techniki optymalizacji RandomSearch oraz framework Optuna do optymalizacji bayesowskiej.
Ustawiliśmy liczbę iteracji równą 50, kross walidację równą 3 oraz ziarno do otrzymania tych samych konfiguracji.

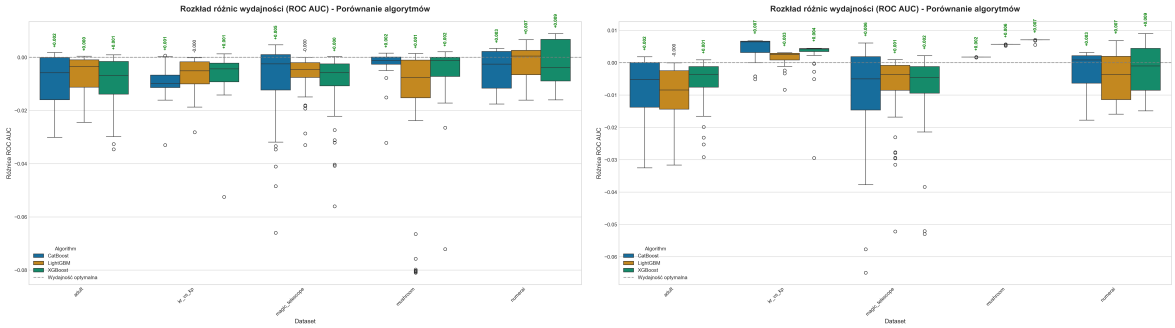
6.2 Tunowalność algorytmów

Tunowalność algorytmów zdefiniowano w oparciu o metodologię przedstawioną w artykule (źródło [1]). Rozumiana jest ona jako różnica w wydajności pomiędzy najlepszą znalezioną konfiguracją dla danego zbioru danych, a konfiguracją referencyjną (globalnie optymalną).

Spośród badanych metod, XGBoost wykazuje największą tunowalność. Oferuje on najwyższy potencjał poprawy wyników, co jest widoczne szczególnie na trudnym zbiorze danych *numerai*

(część rozkładu na wykresie znajdująca się powyżej linii referencyjnej zero), gdzie zysk względem konfiguracji globalnej sięga 0.009 AUC. Nieco mniejszym potencjałem charakteryzują się LightGBM oraz CatBoost, które pozwalają na maksymalny przyrost wydajności odpowiednio w okolicach 0.007 oraz 0.005 AUC na poszczególnych zbiorach danych.

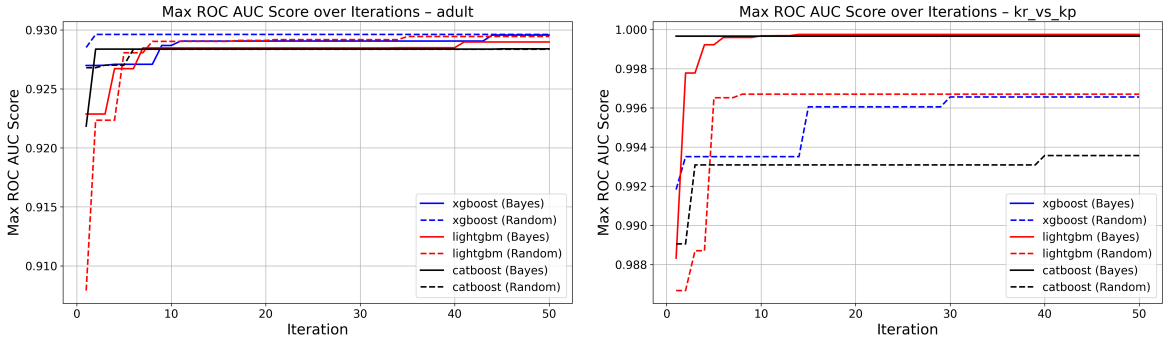
Podsumowując, badane algorytmy gradient boostingu nie wykazują tak wysokiej tunowalności, jak np. maszyny wektorów nośnych (SVM), dla których różnice mogą sięgać nawet 0.15 AUC (jak wskazano w (źródło [1])). Niemniej jednak, ich potencjał optymalizacyjny pozostaje niezerowy, co w specyficznych zastosowaniach uczenia maszynowego i przy trudnych problemach klasyfikacyjnych może mieć istotne znaczenie praktyczne.

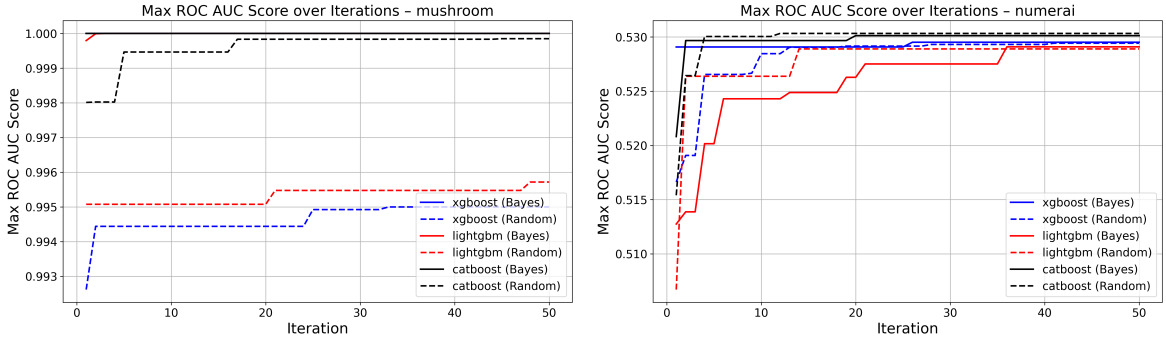


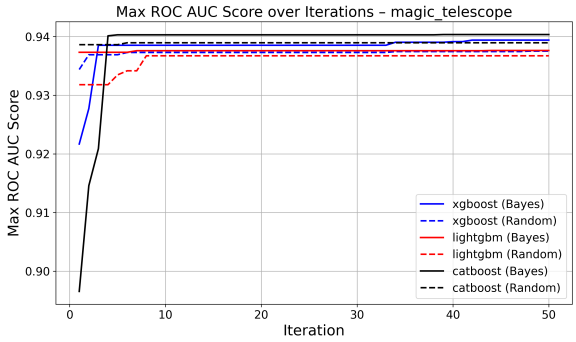
Rysunek 1: Rozkład różnic metryki ROC AUC uzyskanych dla poszczególnych konfiguracji hiperparametrów względem ustalonej konfiguracji optymalnej. Wartości liczbowe umieszczone nad słupkami wskazują maksymalną zanotowaną różnicę (maksymalny zysk) dla danego algorytmu. Po lewej: wyniki dla eksperymentu Random Search; po prawej: wyniki dla optymalizacji Bayesowskiej.

6.3 Zbieżność metod

Poniżej są przedstawione wykresy zbieżności. Każdy wykres odpowiada osobnemu zbioru danych i zawiera 6 krzywych z których każda wskazuje dotychczasowo najlepszy wynik ROC AUC score w zależności od iteracji. Kolor krzywej zależy od modelu którego wyniki reptezentuje. Wyniki uzyskane metodą optymizacji Bayesowskiej są naszkicowane krzywymi ciągłymi, natomiast metodzie Random Search odpowiadają linie przerwane.







Z powyższych wykresów zbieżności wynika że dobór hiperparametrów za pomocą algorytmu optymizacji Bayesowskiej niezależnie od wybranego modelu daje średnio trochę lepsze wyniki i potrzebuje mniej iteracji w porównaniu do RandomSearch.

6.4 Wpływ metody losowania na ocenę tunowalności

Można zaobserwować na Rysunku 1, że tunowalność jest wyższa dla metody Bayes (na mushroom, `kr_vs_kp`). Czyli metoda Bayesowska, skuteczniej identyfikuje konfiguracje lepsze od optymalnych. Podczas gdy RandomSearch niedoszacowuje tunowalności. Jest to związane z inteligentnym przeszukiwaniem metody Bayes. Czyli występuje bias sampling.

7 Podsumowanie

Z naszej analizy wynika, że XGBoost jest najbardziej tunowalny spośród badanych metod. Mniej tunowalne okazały się LightGBM oraz CatBoost. Również z naszych badań wynika, że występuje bias sampling oraz dla optymalizacji Bayesowskiej jest większy wynik tunowalności. Zatem stwierdzamy, że optymalizacja Bayesowska zapewnia lepsze wyniki i szybszą zbieżność niż Random Search.

Literatura

1. `https://jmlr.org/papers/volume20/18-444/18-444.pdf`.