AutoML - Praca Domowa 1

Damian Sarna

21 listopada 2023

1 Wstęp

Celem eksperymentu jest przetestowanie dostrajalności (ang. *tunability*) algorytmów uczenia maszynowego. Dla wybranych algorytmów i danych zostanie zbadana zmienność średniej jakości modeli wyrażonej miarą AUC w zależności od zmienności wektora hiperparametrów modelu (dostrajalność łączna) oraz od zmienności pojedynczych hiperparametrów (dostrajalność hiperparametru).

2 Opis eksperymentu

2.1 Testowane algorytmy

Dla celów eksperymentu zostały wybrane trzy różne algorytmy uczenia maszynowego rozwiązujące problem klasyfikacji binarnej: klasyczne drzewo decyzyjne oraz dwa algorytmy oparte na komitetach, czyli random forest oraz gradient boosting. W przypadku dwóch pierwszych wykorzystano implementację z pakietu scikit-learn, a ostatni pochodzi z pakietu xgboost.

2.2 Źródła danych

Do eksperymentu zostały wykorzystane 4 zbiory danych z repozytorium OpenML.

1. credit-g (ID: 31, 1000 obserwacji, 61 kolumn)
2. mozilla4 (ID: 1046, 15545 obserwacji, 5 kolumn)
3. kc1 (ID: 1067, 2109 obserwacji, 21 kolumn)
4. phoneme (ID: 1489, 5404 obserwacji, 5 kolumn)

Kryterium wyboru danych był niewielki rozmiar danych (rzędu tysięcy obserwacji) w celu zapewnienia racjonalnie krótkiego czasu wykonania eksperymentu. Należy odnotować, że już na etapie doboru danych niejawnie faworyzujemy niektóre z testowanych hiperparametrów (np. przy takiej proporcji kolumn do wierszy regularyzacja / selekcja zmiennych nie powinny być istotnym problemem). W opinii autora jest to jednak nieuniknione, a zważywszy na niewielki zakres eksperymentu, wybrane zbiory są dla jego celu wystarczająco zróżnicowane zarówno pod kątem liczby wierszy, jak i kolumn.

2.3 Dobór Hiperparametrów

Dla każdego algorytmu zostało wybranych 6-7 najważniejszych parametrów, które w największym stopniu wpływają na elastyczność struktury modelu. Kolejne tabele prezentują rozważane hiperparametry oraz ich zakresy. Dopisek "log"w kolumnie "Uwagi"oznacza, że wartości liczbowe rozpatrujemy na skali logarytmicznej.

Parametr	Od	Do	Kategorie	Uwagi
criterion	-	-	gini, entropy	-
splitter	-	-	best, random	-
max_depth	2	32	None	-
min_samples_split	-4	-1	-	log
min_impurity_decrease	-5	-1	-	log
ccp_alpha	-5	-1	-	log

Tabela 1: Siatka parametrów dla algorytmu DecisionTree

Warto odnotować, że domyślną wartością parametru min_samples_split w pakiecie scikit-learn jest wartość całkowitoliczbowa 2. W tym eksperymencie przyjęto natomiast podejście bazujące na proporcji, tzn. minimalną liczność poddrzewa musi stanowić nie mniej niż pewien ułamek liczności całych danych, i waha się on w przedziale od 0.1% do 10%. Ta sama uwaga dotyczy parametrów stosowanych w algorytmie RandomForest, które w większości pokrywają się z parametrami dla pojedynczego drzewa. Dodatkowym parametrem jest oczywiście n_estimators, który kontroluje wielkość lasu.

Parametr	Od	Do	Kategorie	Uwagi
criterion	-	-	gini, entropy, log_loss	-
n_estimators	2	500	-	-
max_depth	2	32	None	-
min_samples_split	-4	-1	-	log
max_features	-	-	sqrt, log2, None	-
min_impurity_decrease	-10	-1	-	log

Tabela 2: Siatka parametrów dla algorytmu RandomForest

W przypadku ostatniego algorytmu (XGBoost) zdecydowano się na badanie wpływu liczby estymatorów przy ustalonej stałej uczenia (learning_rate). Decyzja ta została podjęta bazując na ogólnie znanym fakcie, że im mniejsze tempo uczenia, tym więcej potrzeba drzew aby osiągnąć podobny wynik. Aby zapewnić odpowiednią jakość algorytmu, parametr learning_rate został przyjęty na 0.05, czyli niższy niż wartość domyślna w pakiecie xgboost (domyśle wartości learning_rate/n_estimators to odpowiednio 0.3/100).

Parametr	Od	Do	Uwagi
n_estimators	10	1000	-
max_depth	2	32	-
subsample	0.5	1	-
colsample_bytree	0.5	1	-
gamma	-10	0	log
reg_alpha	-10	2	log
reg_lambda	-10	2	log

Tabela 3: Siatka parametrów dla algorytmu XGBoost

Optymalizacja parametrów przebiegała na trzy sposoby. Najpierw za pomocą funkcji przeszukiwania losowego (RandomizedSearchCV) została przetestowana próbka 250 kombinacji (wektorów) hiperparametrów, na podstawie której dla każdego modelu został wyznaczony θ^* - empiryczny optymalny hiperparametr maksymalizujący średnie AUC. Następnie dla każdego modelu i hiperparametru z osobna ponownie została przeprowadzona jego jednowymiarowa optymalizacja przy pozostałych parametrach niezmienionych w stosunku do θ^*. Niezależnie przestrzeń parametrów została również przeszukana metodą bayesowską (funkcja BayesSearchCV, również 250 iteracji).

3 Wyniki

3.1 Kalibracja hiperparametrów

Optymalne wektory hiperparametrów uzyskane za pomocą obu metod prezentują poniższe tabele.

Metoda	criterion	splitter	max_depth	min_samples_split	min_imp_decrease	ccp_alpha	AUC
Random	entropy	best	13	0.0464	0.0006	0.0046	0.8202
Bayes	gini	best	9	0.1	0.0001	0.0037	0.8363

Tabela 4: Optymalne hiperparametry algorytmu DecisionTree

Dla algorytmu DecisionTree największa różnica występuje na parametrze min_samples_split – optymalizacja bayesowska preferuje znacznie mniejsze drzewa, w których do podziału potrzeba, aby w poddrzewach znalazło się przynajmiej 10% całego zbioru, co w znacznym stopniu zapobiega overfittingowi. Dzięki temu uzyskiwane AUC jest o ok. 0.015 wyższe.

Metoda	criterion	n_est	max_depth	min_samples_split	max_features	min_imp_dec	AUC
Random	log_loss	232	15	0.001	log2	0.0	0.8708
Bayes	entropy	500	24	0.0004	sqrt	0.0	0.8783

Tabela 5: Optymalne parametry algorytmu RandomForest

W przypadku RandomForest optymalizacja bayesowska tym razem preferuje większe i głębsze modele i ponownie skutkuje to lepszym AUC, choć różnica jest mniej znacząca (poniżej 0.01).

Metoda	n_est	max_depth	subsample	colsample_bytree	gamma	reg_alpha	reg_lambda	AUC
Random	520	13	0.94	0.67	1.0	1.0	0	0.8683
Bayes	1000	17	0.83	0.5	0	5.42	0.0	0.8752

Tabela 6: Optymalne parametry algorytmu XGBoost

Dla algorytmu XGBoost obserwujemy podobny trend jak dla RandomForest, jeśli chodzi o wielkość modelu. Różnice w pozostałych parametrach pokazują, że optymalne parametry znalezione metodą bayesowską skutkują bardziej odpornymi modelami, mniej skłonnymi do overfittingu (szczególnie chodzi o mniejsze wartości parametrów subsample, colsample_bytree oraz większą wartość reg_alpha). Rysunek. 1 potwierdza, że przy podobnej liczbie iteracji funkcja BayesSearchCV wyszukiwała kombinacje dające średni uzysk powyżej 0.01 AUC dla drzew decyzyjnych oraz poniżej 0.01 dla pozostałych modeli. Jeżeli chodzi o dynamikę zachowania obu algorytmów trudno stwierdzić istotne różnice i wydaje się że niezależnie od metody, parametry bliskie optymalnych, zwykle uzyskiwano już po kilkudziesięciu iteracjach.

3.2 Dostrajalność parametrów

Dostrajalność hiperarametrów danego algorytmu dla zbioru D^j (tutaj $j = 1, 2, 3, 4$) zdefiniowana jest jako $d^j = AUC(\theta^{(j)*}) - AUC(\theta^*)$, gdzie θ^* to empiryczny optymalny wektor hiperparametrów uzyskany metodą przeszukiwanaia losowego (minimalizujący średnie AUC po wszystkich zbiorach), a $\theta^{(j)*}$ to optymalny wektor hiperparametrów dla zbioru D_j. Łączna dostrajalnośc d określona jest jako średnia arytmetyczna po d_i. Poniższa tabela prezentuje dostrajalność dla poszczególnych algorytmów.

Algorytm	Dane 1	Dane 2	Dane 3	Dane 4	Średnia dostrajalność
DecisionTree	0.0055	0.0122	0.0284	0.0285	0.0187
RandomForest	0.0031	0.0015	0.0291	0.0032	0.0092
XGBoost	0.0036	0.0058	0.0194	0.0106	0.0098

Tabela 7: Dostrajalność testowanych algorytmów

Przetestowana została również dostrajalnośc każdego z hiperparametrów modelu, którą definiujemy jako (dla ustalonych danych D_i) $d_i^{(j)} = AUC(\theta_i^{(j)*}) - AUC(\theta^*)$, gdzie $\theta_i^{(j)*}$ to optymalny wektor hiperparametrów dla zbioru D_j, który różni się od θ^* co najwyżej na i-tym miejscu. Wielkość ta reprezentuje maksymalną poprawę AUC przy manipulowaniu i-tym parametrem, gdy wszystkie pozostałe są niezmienione. Łączna dostrajalność dla hiperparametru i to ponownie średnia po zbiorach. Poniższe tabele prezentują dostrajalność hiperparametrów dla każdego algorytmu.

criterion	splitter	max_depth	min_samples_split	min_impurity_decrease	ccp_alpha
0.0010	0.0030	0.0027	0.0020	0.0014	0.0037

Tabela 8: Dostrajalność hiperparametrów algorytmu DecisionTree

criterion	n_estimators	max_depth	min_samples_split	max_features	min_impurity_decrease
0	0.0007	0.0075	0.0063	0.0	0.0076

Tabela 9: Dostrajalność hiperparametrów algorytmu RandomForest

n_estimators	max_depth	subsample	colsample_bytree	gamma	reg_alpha	reg_lambda
0.0006	0.0037	0.0013	0.0030	0.0019	0.0073	0.0059

Tabela 10: Dostrajalność hiperparametrów algorytmu XGBoost

Przetestowana została również dostrajalność przyjmując za θ^* optymalny parametr z metody BayesSearchCV. Wyniki kształtują się wówczas następująco:

Algorytm	Dostrajalność (Random)	Dostrajalność (Bayes)
DecisionTree	0.0187	0.0086
RandomForest	0.0092	0.0018
XGBoost	0.0098	0.0038

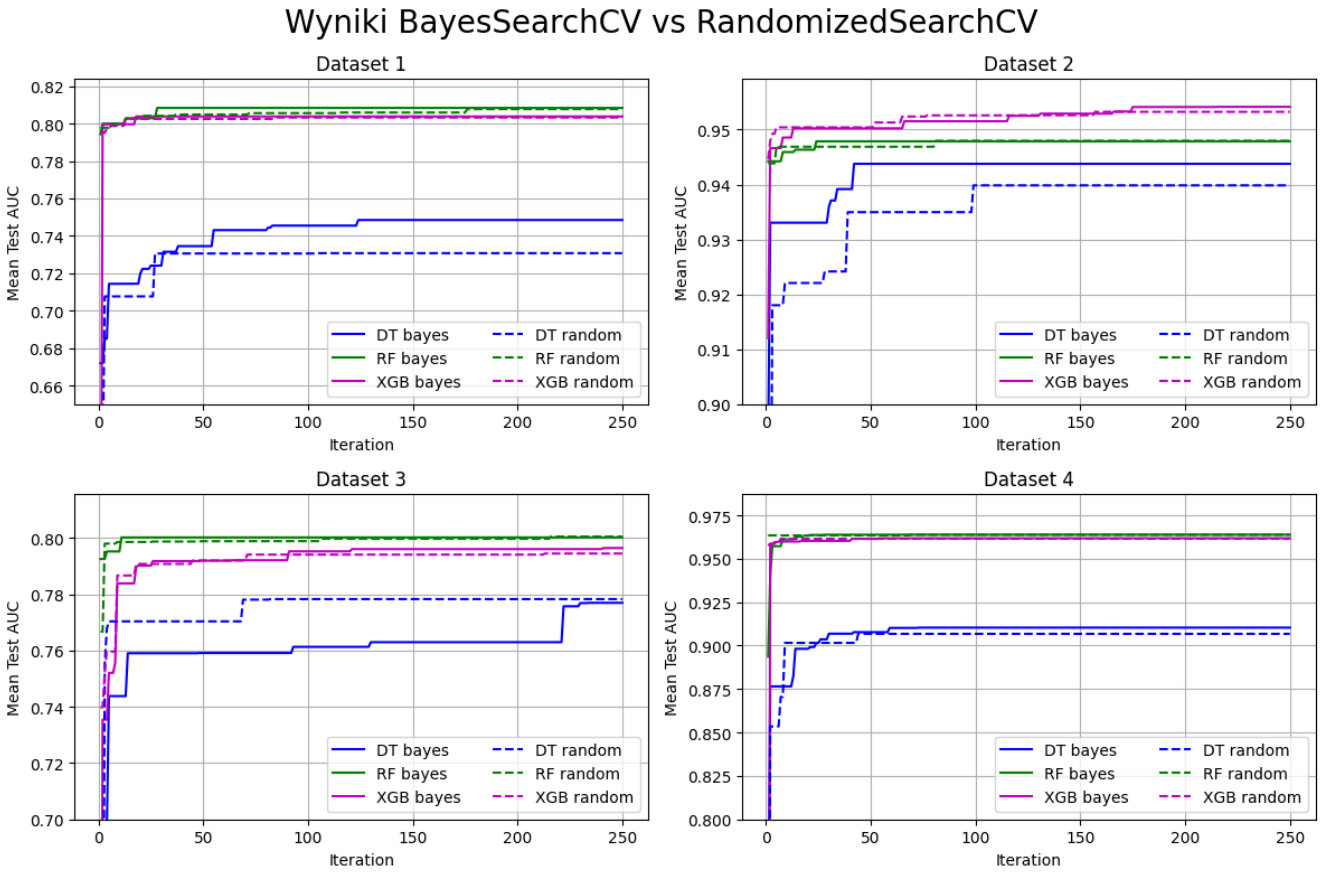
Tabela 11: Dostrajalność wg różnych metod próbkowania

Alternatywna metoda nie zmienia istotnie relacji pomiędzy wynikami poszczególnych modeli (choć są one znacznie niższe), ale można zauważyć, że znacznie obniżyła ona szacowaną dostrajalność algorytmu RandomForest.

4 Podsumowanie

W wyniku porównania modeli można stwierdzić, że algorytm DecisionTree cechuje się największą dostrajalnością, a z analizy dostrajalności poszczególnych parametrów widać, że wiele z nich w podobnym stopniu może wpływać na jakość modelu. Bardziej złożone algorytmy RandomForest i XGBoost okazują się bardziej stabilne i działają porównywalnie dobrze dla szerokiego zakresu wielu parametrów. W przypadku RandomForest największą zależność wyniku od parametru zaobserwowano dla parametrów regularyzujących drzewa (min_samples_split, max_depth, min_impurity_decrease i reg_alpha, reg_lambda, odpowiednio). Z kolei, co może najbardziej zaskakiwać, liczba drzew (n_estimators) okazała się jednym z czynników mających najmniejszy wpływ na jakość tych modeli.

Metodą prókowania zwracającą lepsze wyniki okazała się metoda BayesSearchCV (średni uzyska na AUC powyżej 0.01); obie metody zwracały parametry bliskie optymalnych już po kilkudziesięciu iteracjach. Dostrajalność modeli okazała się zależna od przyjętej metody próbkowania (metoda BayesSearch zmniejszała tę miarę ok. dwukrotnie), jednak relatywna tunowalność między modelami okazała się podobna (z wyjątkiem algorytmu RandomForest, dla którego istotnie spadła).



Rysunek 1: Porównanie jakości przeszukiwania przestrzeni hiperparametrów obiema metodami