Wydział Matematyki
i Nauk Informacyjnych

POLITECHNIKA WARSZAWSKA

AutoML raport 1

Tunowalność hiperparametrów

Devid Khodak 317253
Kamil Kubiak 313293

21 listopada 2023

Spis treści

1 Algorytmy uczenia maszynowego

1.1 Random Forest

Random forest polega na zastosowaniu pewnej liczby niezależnie tworzonych drzew decyzyjnych. Predykcja wynikowa jest średnią z predykcji wszystkich drzew.

1.2 Gradient Boosting

Gradient boosting jest algorytmem podobnym do Random Forest z tą różnicą, że drzewa decyzyjne nie są tworzone niezależnie. Wybrana została implementacja XGBoost.

1.3 Support Vector Machine

SVM próbuje znaleźć płaszczyznę najlepiej rozdzielającą obserwacje należące do różnych klas.

2 Zbiory danych

Wszystkie zbiory danych pochodzą z bazy zbiorów danych OpenML i przeznaczone są do klasyfikacji binarnej.

2.1 Phishing Websites

Zbiór danych zawiera informacje o stronach internetowych. Features tutaj to parametry opisujące strony internetowe, takie jak długość URL, port itp. Natomiast target określa czy dana strona jest stroną uprawiającą phishing.

2.2 eeg-eye-state

Zbiór danych opisuje przebieg 117 sekundowego pomiaru podczas testu EEG (test ten wykrywa zaburzenia w aktywności mózgu). Features w tym przypadku to wyniki 14 konkretnych miar tego testu w kolejnych momentach w czasie. Natomiast target określa czy oko było w danym momencie otwarte czy zamknięte (1 - oko zamknięte, 0 - oko otwarte).

2.3 jm1

Zbiór danych zawiera informacje o kodzie narzędzia JM1. W features znajdują się kolumny opisujące długość i skomplikowanie kodu według różnych miar. Natomiast target to defects, które przyjmuje False, w przypadku, gdy wszystko działa prawidłowo, natomiast True w przeciwnym przypadku.

2.4 mozilla4

Zbiór danych zawiera informacje opisujące rozwój kodu produktu Mozilli w czasie. W features mamy kolumny, które opisują czas życia C++ klasy (powstanie, naprawianie błędu, usunięcie) oraz jej rozmiar. Natomiast target to state, który wynosi domyślnie 0, ale w przypadku gdy dana klasa miała poprawiany jakiś defekt wynosi 1.

3 Metoda Random Search

3.1 Opis metody

Metoda Random Search polega na przejrzeniu losowo wybranych zestawów hiperparametrów w celu znalezienia najlepszego.

Testy liczby iteracji do uzyskania stabilnych wyników optymalizacji przeprowadzano dla [1, 10, 20, 30, 40, 45, 50] iteracji.

3.2 Optymalizacja Random Forest

Rozważana była następująca siatka hiperparametrów:

- n_estimators: [50, 75, 100, 125, 150, 175, 200]
- max_features: ['log2', 'sqrt', None]
- max_depth: [5, 10, 15, 20, 25, None]
- min_samples_split: [2, 5, 10]
- min_samples_leaf: [1, 2, 4]
- bootstrap: [True, False]

Metoda Random Search dla algorytmu Random Forest daje stabilne wyniki po 40 iteracjach. Wyniki testów dla poszczególnych datasetów znajdują się w pliku , , $random\_search.ipynb''$ w sekcji Evaluate algorytmu Random Forest.

Maksymalna średnia wartość accuracy po wszystkich zbiorach danych wyniosła:

$$\theta_1^* = 0.9165127017545728,$$

dla zestawu hiperparametrów:

$$n\_estimators = 75,$$
$$min\_samples\_split = 5,$$
$$min\_samples\_leaf = 1,$$
$$max\_features = log2,$$
$$max\_depth = None,$$
$$bootstrap = False.$$

3.3 Optymalizacja SVM

Rozważana była następująca siatka hiperparametrów:

- C: [0.1, 1, 10, 100, 1000]
- gamma: [1, 0.1, 0.01, 0.001, 0.0001]
- kernel: ['rbf', 'linear', 'sigmoid']

Metoda Random Search dla algorytmu SVC daje stabilne wyniki po 30 iteracjach. Wyniki testów dla poszczególnych datasetów znajdują się w pliku , , $random\_search.ipynb''$ w sekcji Evaluate algorytmu SVC.

Maksymalna średnia wartość accuracy po wszystkich zbiorach danych wyniosła:

$$\theta_2^* = 0.8133374006292702,$$

dla zestawu hiperparametrów:

$$C = 100,$$
$$gamma = 1,$$
$$kernel = rbf.$$

3.4 Optymalizacja XGBoost

Rozważana była następująca siatka hiperparametrów:

- gamma: [0, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 5]
- learning_rate: [0.01, 0.03, 0.06, 0.1, 0.15, 0.2, 0.25, 0.3]
- max_depth: [3, 5, 6, 7, 8, 9, 10]
- n_estimators: [50, 75, 100, 125, 150, 175, 200]
- reg_alpha: [0, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 12.8, 25.6, 51.2, 102.4, 200]
- reg_lambda: [0, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, 12.8, 25.6, 51.2, 102.4, 200]

Metoda Random Search dla algorytmu XGBoost daje stabilne wyniki po 40 iteracjach. Wyniki testów dla poszczególnych datasetów znajdują się w pliku , , $random\_search.ipynb''$ w sekcji Evaluate algorytmu XGBoost.

Maksymalna średnia wartość accuracy po wszystkich zbiorach danych wyniosła:

$$\theta_3^* = 0.9177681913688276,$$

dla zestawu hiperparametrów:

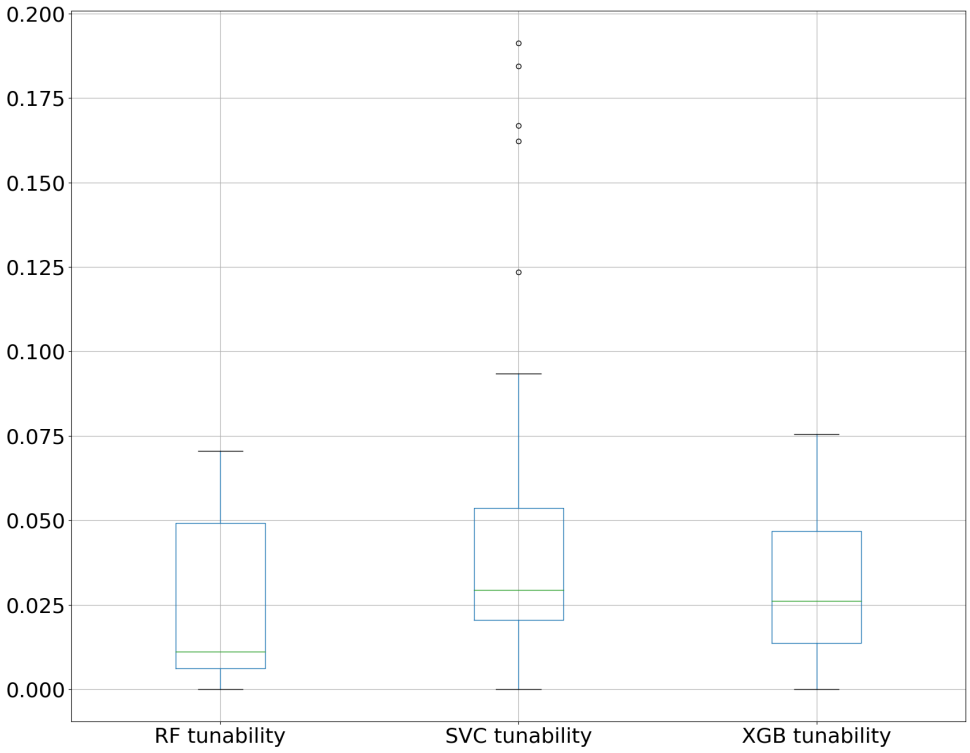
$$gamma = 0.1,$$
$$learning\_rate = 0.3,$$
$$max\_depth = 8,$$
$$n\_estimators = 200,$$
$$reg\_alpha = 0.1,$$
$$reg\_lambda = 6.4.$$

3.5 Wyniki tunowalności algorytmów

Do wyznaczenia tunowalności poszczególnych algorytmów korzystano z wartości θ_1^*, θ_2^* i θ_3^*. Została wybrana liczba iteracji Random Search równa 50. Jest to liczba iteracji dla której wszystkie algorytmy dają stabilne wyniki. Tunowalność algorytmu o najlepszym zestawie hiperparametrów θ_k^*, dla zestawu parametrów θ_i^k, wyznaczana była w sposób następujący:

$$d_i^k = \theta_k^* - \theta_i^k,$$

gdzie k = 1, 2, 3 oraz i = 1, 2, ..., 50. Wszystkie dane potrzebne do wyliczenia tunowalności można znaleźć w folderze , , $Wyniki''$. Wykresy pudełkowe tunowalności poszczególnych algorytmów zostały przedstawione na rysunku 1.



Rysunek 1: Wykresy pudełkowe tunowalności poszczególnych algorytmów

Z wykresu 1 wynika, że najbardziej tunowalny jest algorytm SVC.

4 Metoda Bayes Search

W przypadku metody Bayes Search siatki hiperparametrów dla każdego z algorytmów miały takie same górne i dolne ograniczenia, ale zawierały też wszystkie wartości pomiędzy. Przykładowo jak siatka hiperparametru P w Random Search wyglądała tak - [a,c,f,g,j,l], gdzie a najmniejszą wartością, a l największą to przy Bayes Search siatka hiperparametru P to (a, l).

4.1 Optymalizacja Random Forest

Hiperparametry	**Phishing Websites**	**eeg-eye-state**	**jm1**	**mozilla4**
n_estimators	185	188	50	191
min_samples_split	4	10	2	4
min_samples_leaf	2	4	1	4
max_features	log2	None	sqrt	log2
max_depth	25	6	5	5
bootstrap	False	True	True	True
Accuracy	0.9682496607869	0.522029372496	0.8091911764705	0.918173045995

Tabela 1: Zestawy hiperparametrów, dla których osiagane były najwyższe wyniki dla algorytmu Random Forest przy Bayes Search

4.2 Optymalizacja SVM

Hiperparametry	**Phishing Websites**	**eeg-eye-state**	**jm1**	**mozilla4**
C	157.83879853890	0.1	223.64202820542	170.0853453339
gamma	0.04247908159416	100.0	19.9476465446	2.685839247256
kernel	rbf	sigmoid	rbf	rbf
Accuracy	0.9620985979194	0.522029372	0.8073529411764	0.8692827275651

Tabela 2: Zestawy hiperparametrów, dla których osiagane były najwyższe wyniki dla algorytmu SVM przy Bayes Search

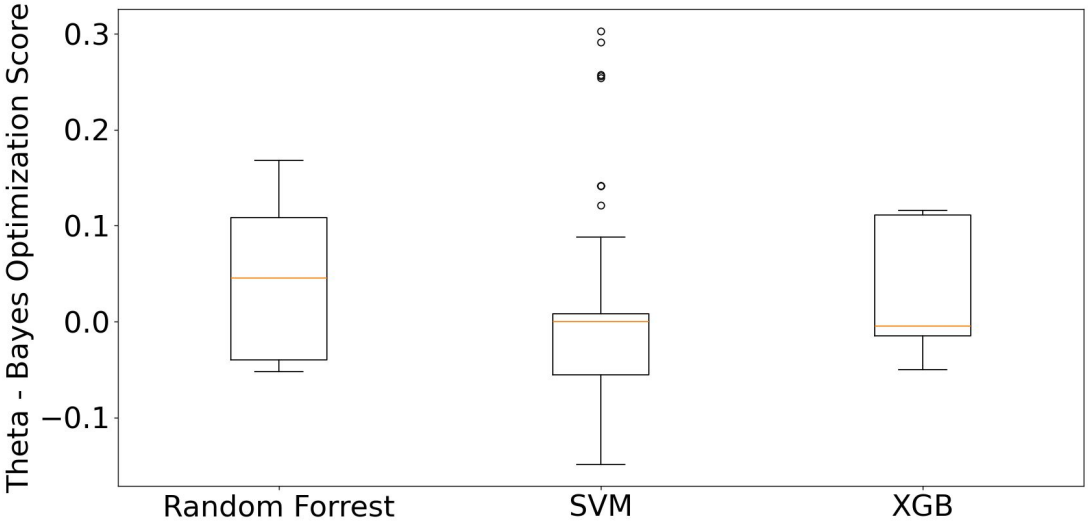
4.3 Optymalizacja XGBoost

Hiperparametry	Phishing Websites	eeg-eye-state	jm1	mozilla4
gamma	0	4	3	2
learning_rate	0.3	0.3	0.0149967377512	0.1267551045561
max_depth	10	8	7	6
n_estimators	50	50	117	101
reg_alpha	0	123	39	21
reg_lambda	0	176	197	83
Accuracy	0.967526006331	0.510413885180	0.806709558823	0.933676423287

Tabela 3: Zestawy hiperparametrów, dla których osiagane były najwyższe wyniki dla algorytmu XGBoost przy Bayes Search

5 Porównanie wyników

Po głębszej analizie wyników działania metod BayesSearch i RandomSearch dochodzimy do wniosku, że z reguły dają bardzo porównywalne wyniki. Potencjalnie przemawiający na korzyść BayesSearch jest fakt, że wymagał tylko 20 iteracji w porównaniu do 50 iteracji Random Search. Warto wspomnieć, że BayesSearch spisał się znacząco gorzej w przypadku jednego datasetu (eeg-eye-state) dla algorytmów Random Forrest i XGBoost.



Rysunek 2: Wykresy pudełkowe różnicy pomiędzy θ^*, a wynikami accuracy dla wszystkich zestawów hiperparametrów każdego z algorytmów ($\theta^* - accuracy$)