Automatyczne uczenie maszynowe – projekt 1

Magdalena Bartczak, Aleksandra Mach

Spis treści

1 Analizowane zbiory danych

Do naszego projektu wykorzystałyśmy cztery zbiory danych pobranych z *openml*, które opisują problem klasyfikacji binarnej.

1. **Zbiór 1: *cylinder-bands*** – zbiór rozmiaru 540×38, zawiera 18 zmiennych numerycznych i 19 zmiennych kategorycznych; zawiera braki danych w 263 obserwacjach; zmienna odpowiedzi *band_type* przyjmuje dwie wartości: 'band' oraz 'noband'.

2. **Zbiór 2: *adult*** – zbiór rozmiaru 48842×15, zawiera 6 zmiennych numerycznych i 8 zmiennych kategorycznych; zmienna odpowiedzi *class* przyjmuje dwie wartości: '$<= 50K$' oraz '$> 50K$'.

3. **Zbiór 3: *labor*** – zbiór rozmiaru 57×17; zawiera 8 zmiennych numerycznych i 8 zmiennych kategorycznych; zwiera braki danych w 56 obserwacjach; zmienna odpowiedzi *class* przyjmuje dwie wartości: 'good' oraz 'bad'.

4. **Zbiór 4: *sick*** – zbiór rozmiaru 3772×30; zwiera 6 zmiennych numerycznych i 23 zmienne kategoryczne; zawiera braki danych w 3772 obserwacjach; zmienna odpowiedzi *Class* przyjmuje dwie wartości: 'sick' oraz 'negative'.

Każdy z wymienionych zbiorów został poddany preprocessingowi składającemu się z następujących kroków zawartych w odpowiednim pipelinie:

- transformacje kolumn kategorycznych do kolumn numerycznych przy użyciu funkcji *OrdinalEncoder()* oraz *OneHotEncoder()*,
- wypełnianie braków danych przy użyciu funkcji *SimpleImputer()* z argumentem strategy = 'median' dla zmiennych numerycznych oraz strategy = 'most_frequent' dla zmiennych kategorycznych,
- skalowanie zmiennych numerycznych przy użyciu funkcji *MinMaxScaler()*.

Pipeline został dopasowany (metoda fit) na odpowiednich zbiorach, a następnie zbiory te zostały odpowiednio przekształcone (metoda transform). Nie było podziału na zbiory treningowy i testowy ze względu na to, że w obu metodach samplingu kroswalidacja w nich zawarta sama dokonuje tych podziałów.

2 Metody samplingu

Zgodnie z treścią polecenia próbowałyśmy zdefiniować siatki hiperparametrów dla każdego z trzech algorytmów na podstawie artykułu. Ostatecznie, ze względu na ograniczenia obliczeniowe Google colab, w którym pracowałyśmy, postanowiłyśmy ograniczyć te siatki do następujących:

- Las losowy:
  - *n_estimators*: np.arange(50, 501)
  - *max_depth*: np.arange(10,21)
  - *min_samples_leaf*: np.arange(1,11)
- XGBoost:

  - *learning_rate*: uniform($2^{-10}, 1$)
  - *max_depth*: np.arange(5,13)
  - *num_boost_round*: np.arange(50,501)
  - *min_child_weight*: uniform(1, 2^7)

- SVM:
  - C: np.arange(1,12)
  - *gamma*: np.arange(1,12)

Do tuningu hiperparametrów wybranych algorytmów zastosowałyśmy dwie metody losowania punktów:

1. ***Random Search*** – z wykorzystaniem funkcji *RandomizedSearchCV* z pakietu *scikit-learn*, przeszukuje losowo zdefiniowaną przestrzeń hiperparametrów modelu, oceniając skuteczność każdej kombinacji za pomocą kroswalidacji, aby znaleźć optymalne parametry, przyspieszając proces przeszukiwania w porównaniu do pełnego przeglądu siatki. Algorytm losuje określoną liczbę kombinacji parametrów, ocenia ich wydajność i zwraca najlepsze parametry dla ostatecznego dopasowania modelu.

2. ***Bayes Optimization*** – z wykorzystaniem pakietu *SMAC*, wykorzystuje model statystyczny, zwany procesem Gaussa, do adaptacyjnego przeszukiwania przestrzeni hiperparametrów, selekcji nowych punktów na podstawie aktualnych ocen, minimalizując liczbę prób potrzebnych do znalezienia optymalnych parametrów modelu. To iteracyjny proces, gdzie każda iteracja dostosowuje model na podstawie wyników poprzednich prób, skoncentrowanych na obszarach przestrzeni parametrów, które mają największe szanse zawierania optymalnych ustawień.

Napisałyśmy dwie funkcje *random_search* oraz *bayesian_opt* odpowiedzialne za przeszukiwanie siatki hiperparametrów oraz wybranie najlepszego zestawu przy użyciu odpowiedniej metody samplingu.

1. ***random_search*** – przyjmuje argumenty *model*, gdzie model ∈ ['rf_classifier', 'xgb_classifier', 'SVM_classifier'], *X_train*, *y_train* oraz *liczba_iteracji*; kolejne kroki w funkcji:
   (i) w zależności od wybranego modelu jest definiowana odpowiednia siatka hiperparametrów, z której będą losowane kombinacje,
   (ii) użycie funkcji *RandomizedSearchCV* z pakietu *scikit-laern* do przeprowadzenia 3-krotnej kroswalidacji ze scoringiem *roc_auc*,
   (iii) dopasowanie danch *X_train* i *y_train*,
   (iv) funkcja zwraca wyniki przeprowadzonej kroswalidacji dla każdej iteracji algorytmu oraz zestaw wybranych optymalnych hiperparametrów.

2. ***bayesian_opt*** – przyjmuje argumenty *model*, gdzie model ∈ ['rf_classifier', 'xgb_classifier', 'SVM_classifier'], *X*, *y*, *liczba_prob*, *seed*; kolejne kroki w funkcji:
   (i) w zależności od wybranego modelu jest definiowana odpowiednia siatka hiperparametrów, z której będą losowane kombinacje,

(ii) definiowana jest funkcja celu *train*, w której przeprowadzana jest 3-krotna kroswalidacja przy użyciu wbudowanej funkcji *cross_val_score* przy użyciu scoringu *roc_auc_score*, a zwracany jest średni wynik dla każdej wylosowanej kombinacji parametrów,

(iii) funkcja celu jest optymalizowana i zapisywane są wyniki z każdej wylosowanej kombinacji parametrów; pakiet *smac* pozwala nam na dodanie maksymalnej liczby prób (tj. losowanych kombinacji parametrów), ale może zakończyć działanie wcześniej, gdy wyniki zaczną zbiegać,

(iv) funkcja zwraca wszystkie zapisywane wyniki oraz zestaw parametrów, który maksymalizuje AUC.

3 Wyniki optymalizacji

Każdy z wymienionych algorytmów został zastosowany do każdego z czterech wspomnianych zbiorów danych. Do każdego z nich metody samplingu dobierały optymalne parametry przeszukując tą samą siatkę zdefiniowaną wcześniej. W tym rozdziale prezentujemy najlepsze parametry wybrane dla każdego z algorytmów zarówno dla każdego z 4 zbiorów indywidualnie jak i dla wszystkich zbiorów razem (defaultowe) znalezione przy pomocy dwóch wspomnianych metod samplingu.

3.1 Las losowy

Tabela 1: Optymalne parametry wybrane przez las losowy

	RandomSearch					Bayes				
	db 1	db 2	db 3	db 4	all	db 1	db 2	db 3	db 4	all
min_samples_leaf	6	1	2	1	10	4	2	3	1	2
max_depth	17	18	15	18	20	16	19	23	19	19
n_estimators	84	394	241	394	237	328	155	299	260	480

3.2 XGBoost

Tabela 2: Optymalne parametry wybrane przez XGBoost

	RandomSearch					Bayes				
	db 1	db 2	db 3	db 4	all	db 1	db 2	db 3	db 4	all
learning_rate	0.002	0.001	0.01	0.001	0.002	0.11	0.12	0.70	0.27	0.7
max_depth	6	10	9	5	6	12	6	7	11	10
num_boost_round	447	257	121	489	447	369	65	126	94	106
min_child_weight	9.89	21.2	37.87	36.74	9.89	19.55	48.65	3.58	34.25	24.81

3.3 SVM

Tabela 3: Optymalne parametry wybrane przez SVM

	RandomSearch					Bayes				
	db 1	db 2	db 3	db 4	all	db 1	db 2	db 3	db 4	all
C	9	6	5	5	7	11	14	18	17	15
gamma	2	4	1	1	3	1	2	2	3	5

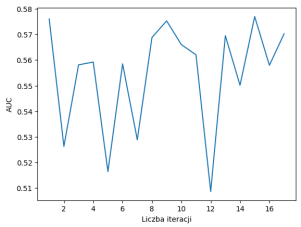
4 Wnioski

Podsumowując przedstawione wyniki można stwierdzić, że proces doboru optymalnych parametrów jest bardzo istotnym krokiem w procedurach uczenia maszynowego. Chcąc przeszukać odpowiednią dużą siatkę hiperparametrów, aby udało się nam znaleźć jak najlepszą kombinację, musimy liczyć się ze sporym czasem obliczeń. Podobnie, ograniczeniem jest też liczba iteracji, czyli liczba losowanych kombinacji parametrów. Zwiększając ją, zwiększamy szansę na wylosowanie jak najlepszej kombinacji. Mimo to zauważyłyśmy, że dla optymalizacji bayesowskiej stabilne wyniki otrzymujemy już po kilkunastu iteracjach dla każdego zbioru. Generalnie optymalne parametry wybrane dla wszystkich zbiorów wspólnie różnią się od tych wybieranych dla każdego zbioru indywidualnie, a wyniki dla parametrów defaultowych są zazwyczaj nieco mniejsze niż najlepsze wyniki dla dl każdego ze zbiorów z osobna. Jednak różnice te są na tyle małe, że uważamy, że dobór parametrów defaultowych może być dobrym i uniwersalnym rozwiązaniem.

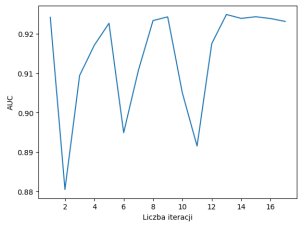
Dodatkowo przedstawiamy wybrane wizualizacje procesów uczenia oraz tunowalności algorytmów.

A Proces uczenia

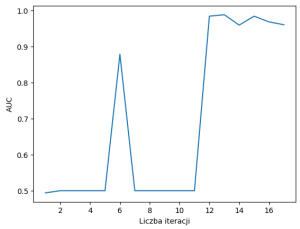
1. XGBoost – optymalizacja bayesowska



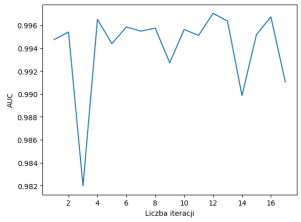
(a) Zbiór 1



(b) Zbiór 2

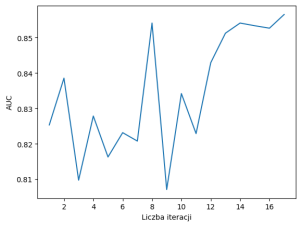


(c) Zbiór 3

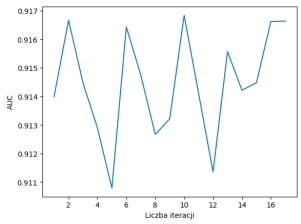


(d) Zbiór 4

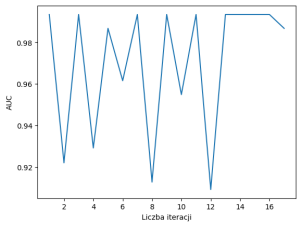
2. Random Forest – optymalizacja bayesowska



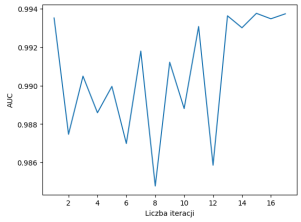
(a) Zbiór 1



(b) Zbiór 2

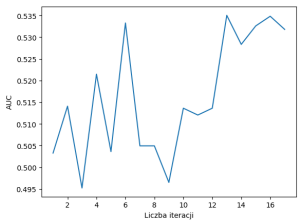


(c) Zbiór 3

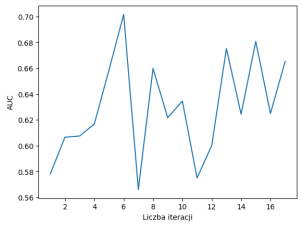


(d) Zbiór 4

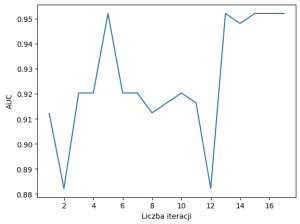
3. SVM – optymalizacja bayesowska



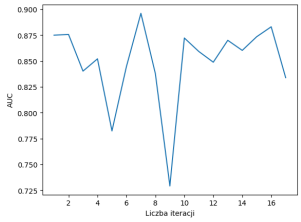
(a) Zbiór 1



(b) Zbiór 2

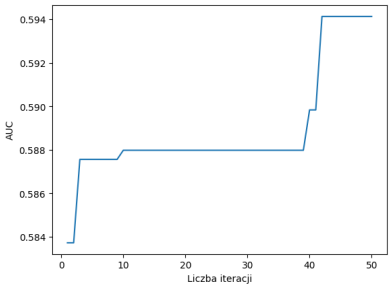


(c) Zbiór 3

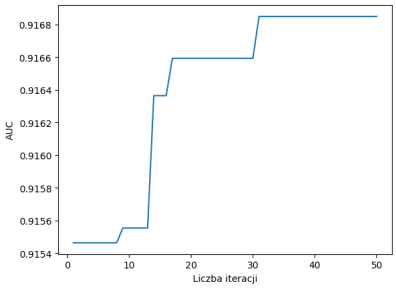


(d) Zbiór 4

4. Random Forest – RandomSearch



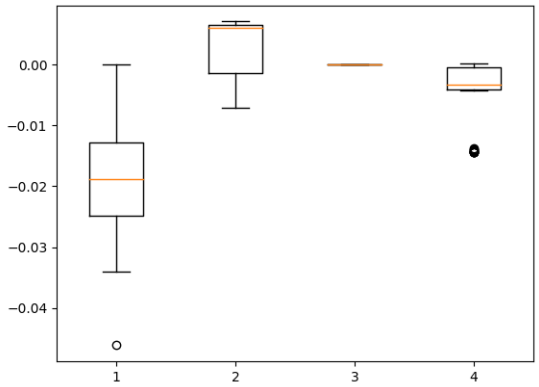
(a) Zbiór 1



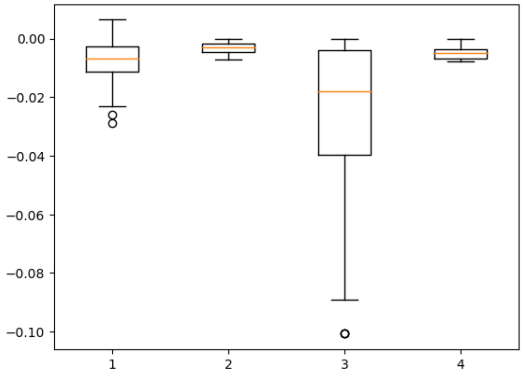
(b) Zbiór 2

B Tuning algorytmów

1. Tuning dla 4 zbiorów – XGBoost, RandomSearch



2. Tuning dla 4 zbiorów – Random Forest, RandomSearch



3. Tuning dla 4 zbiorów – SVM, RandomSearch