AUTOMATYCZNE UCZENIE MASZYNOWE

HW1 — sprawozdanie

Zuzanna Glinka
Jakub Kasprzak

Spis treści

1 Wstep

Poniższy dokument stanowi sprawozdanie z pracy domowej nr 1 z przedmiotu Automatyczne Uczenie Maszynowe. Jej przedmiotem było zbadanie tunowalności trzech algorytmów uczenia maszynowego na czterech zbiorach danych przy użyciu dwóch metod optymalizacji hiperparametrów. Wybranymi przez nas modelami sa **drzewo decyzyjne (Decision tree)**, **k najbliższych sasiadów (K-nearest neighbors)** oraz **las losowy (Random forest)**. Wykorzystaliśmy implementacje modeli zawarte w biliotece sklearn[Pedregosa et al., 2011]. Do optymalizacji bayesowskiej wykorzystaliśmy biblioteke scikit-optimize[Head et al., 2021]. Wybrane przez nas zbiory danych pobraliśmy ze zbiorów danych zawartych na stronie projektu OpenML[Vanschoren et al., 2013]. Sa one problemami klasyfikacji binarnej.

2 Schemat eksperymentu

Przeprowadzany przez nas eksperyment był kilkuetapowy i niezależny dla każdego z modeli:

1. Przygotowanie zbioru danych. Na wszystkich zbiorach stosowaliśmy ten sam pipeline do preprocessingu danych. Kolumny numeryczne imputowaliśmy średnia oraz skalowaliśmy do przedziału (0,1), natomiast kolumny kategoryczne imputowaliśmy moda oraz stosowaliśmy OneHotEncoding.

2. Wybór hiperparametrów do badania w eksperymencie oraz ich zakresów. Decyzje podejmowaliśmy zgodnie z zaproponowanymi zakresami z artykułu[Probst et al., 2019] dotyczacego analogicznego problemu.

3. Utworzenie przestrzeni wektorów hiperparametrów poprzez jednostajne losowanie z określonych zakresów. Wykorzystywaliśmy je zgodnie z zasada algorytmu random search, ale potrzebowaliśmy takiej samej przestrzeni parametrów dla każdego zbioru, dlatego losowaliśmy je recznie.

4. Wytrenowanie na każdym z czterech zbiorów danych modelu o parametrach zgodnych z kolejnymi wektorami z przestrzeni. Metryka, która mierzyliśmy trafność działania modelu, było accuracy.

5. Obliczenie średniej arytmetycznej wartości accuracy dla każdego z modeli oraz wybranie zestawu hiperparametrów, dla którego była ona maksymalna. W ten sposób wybieraliśmy domyślny zestaw parametrów dla danego modelu, nazywany później θ^*.

6. Przeprowadzenie na wszystkich czterech zbiorach danych optymalizacji bayesowskiej. Wykorzystywaliśmy pieciokrotna kroswalidacje.

7. Wyznaczenie optymalnej wartości każdego z hiperparametrów przy pozostawieniu pozostałych zgodnie z ustalonym θ^* poprzez jednostajne losowanie jego wartości z uprzednio zdefiniowanego zakresu oraz trenowanie kolejnych modeli. Takie działanie wykonywaliśmy zarówno dla wyników otrzymanych z przeszukiwania losowego, jak i optymalizacji bayesowskiej.

8. Obliczenie tunowalności hiperparametrów dla obu metod optymalizacji zgodnie ze wzorem z artykułu [Probst et al., 2019].

9. Stworzenie wizualizacji przebiegu czasowego obu metod w celu porównania wyników.

3 Wyniki eksperymentu

W poniższej sekcji zaprezentujemy wyniki przeprowadzonych eksperymentów, tj. wartości funkcji ryzyka dla poszczególnych parametrów modeli oraz wykresy przebiegu czasowego algorytmów optymalizujacych.

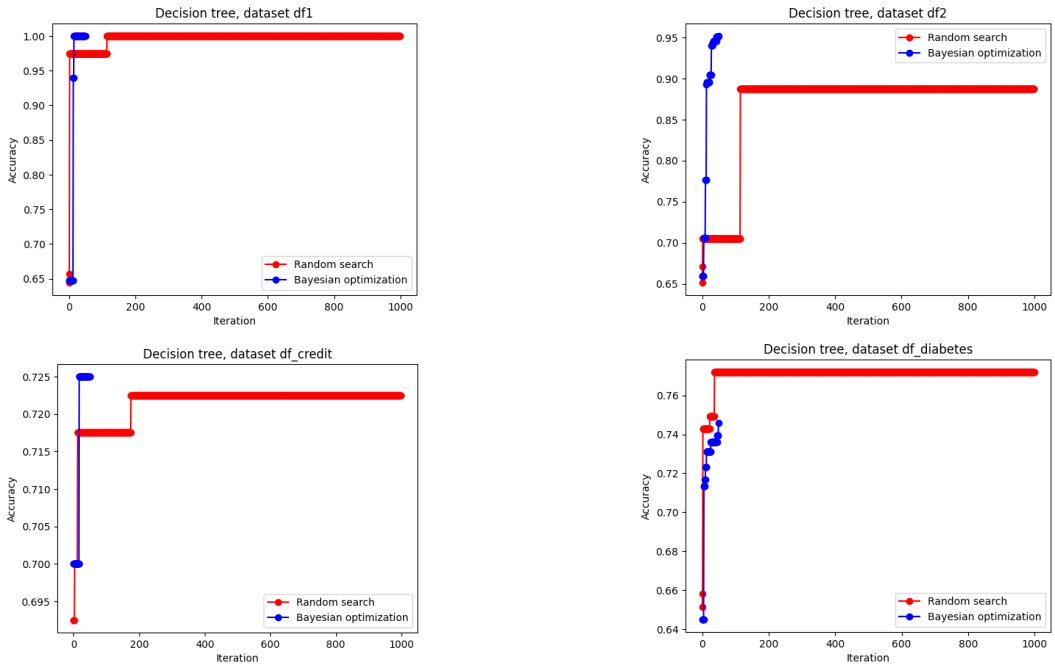
3.1 Drzewo decyzyjne

W tabeli 1 umieszczone zostały wartości funkcji ryzyka dla drzewa decyzyjnego. Analizowaliśmy dla niego cztery parametry: *ccp_alpha*, *max_depth*, *min_samples_leaf* oraz *min_samples_split*. Na rysunku 1 można zobaczyć przebiegi czasowe algorytmów dla poszczególnych zbiorów danych. Jak widać, nawet przy znacznie mniejszej liczbie iteracji, algorytm bayesowski znajdował zazwyczaj lepsze wyniki. Był w stanie osiagać duże przyrosty, podczas gdy random search dość szybko osiagał stabilność na niższym poziomie.

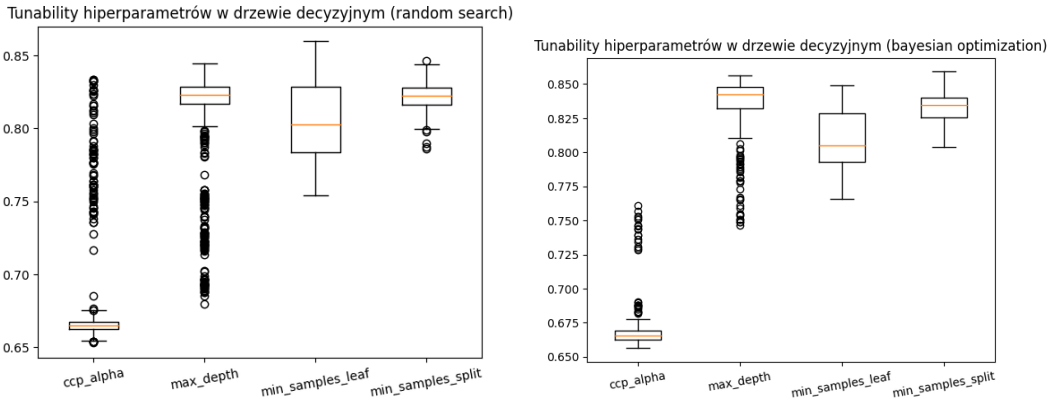
Wartości ryzyka sa nieznacznie wyższe dla metody random search. Patrzac jednak na rysunek 4 nie widać znaczacych różnic pomiedzy algorytmami.

	ccp_alpha	**max_depth**	**min_sample_leaf**	**min_sample_split**
Random search	0.0178	0.0323	0.0452	0.0282
Bayesian optimization	-0.0137	0.0105	0.0063	0.0068

Tabela 1: Wartości ryzyka dla drzewa decyzyjnego



Rysunek 1: Wykresy przedstawiajace przebiegi czasowe algorytmu random search (czerwony) oraz bayesian optimization (niebieski) dla drzewa decyzyjnego. W zwiazku ze znaczacym kosztem obliczeniowym optymalizacji bayesowskiej, obliczaliśmy dla niej znacznie mniej iteracji, jednak widać, że w wiekszości przypadków już w tak krótkim czasie osiaga ona lepsze wyniki.



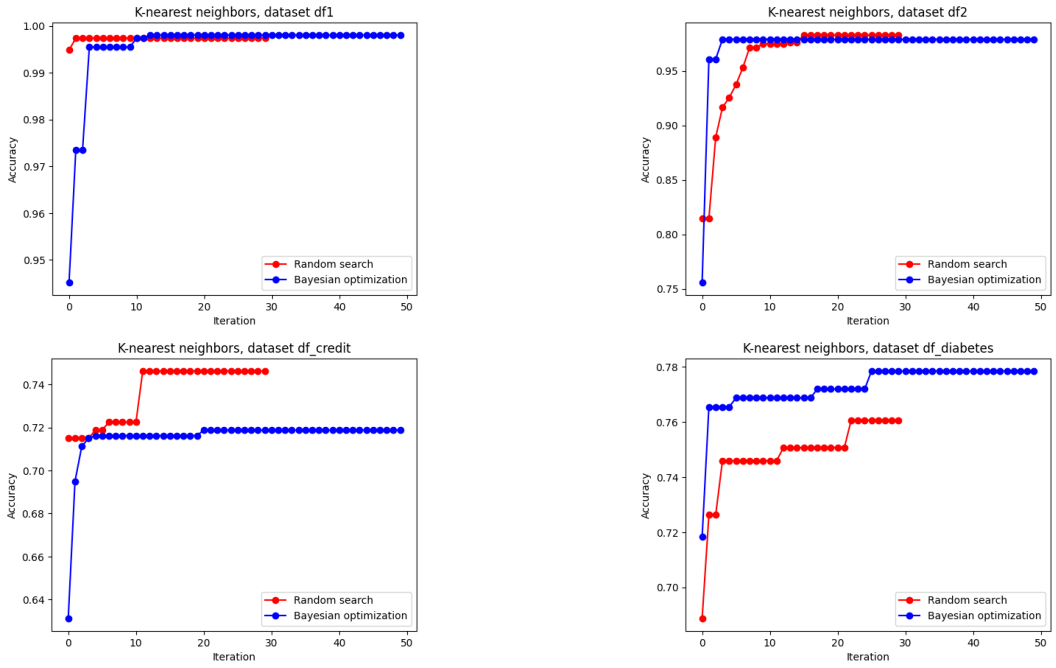
Rysunek 2: Wykresy tunability dla poszczególnych parametrów w drzewie decyzyjnym dla metody random search (po lewej) i optymalizacji bayesowskiej (po prawej). Jak widać, wykresy sa bardzo podobne, wiec w tym przypadku wybrana metoda nie wpływa znaczaco na tunowalność.

	k
Random search	0.0104
Bayesian optimization	0.0032

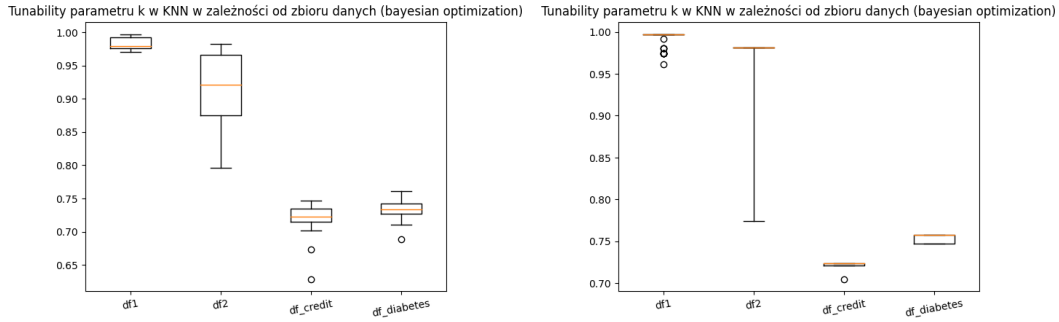
Tabela 2: Tabela przedstawiajaca wartości ryzyka dla modelu KNN. Nie odbiegaja one od siebie znaczaco, co wskazuje na brak wyraźnych różnic miedzy algorytmami dla tego modelu.

3.2 K najbliższych sasiadów

Ten algorytm był najprostszym z wybranych przez nas, bo optymalizowany jest w nim jedynie jeden hiperparametr — k. Jest on parametrem dyskretnym, a artykuł, z którego czerpaliśmy zakresy, wskazuje przedział (1, 30), wiec w tym przypadku zamiast random search wykorzystaliśmy swego rodzaju grid search, zwyczajnie sprawdzajac wszystkie 30 wartości parametru. Jako że algorytm nie był wymagajacy obliczeniowo, dla optymalizacji bayesowskiej testowaliśmy wartości od 0 do 50. Na wykresach 3 można zobaczyć przebieg czasowy działania algorytmu, natomiast w tabeli **??** znajduja sie wartości ryzyka dla poszczególnych metod.



Rysunek 3: Wykresy przedstawiajace przebiegi czasowe algorytmu random search (czerwony) oraz bayesian optimization (niebieski) dla modelu KNN. Jak widać, algorytmy uzyskiwały podobne wyniki na dwóch pierwszych zbiorach danych, a na dwóch pozostałych raz lepsze wyniki osiagnał random search, a raz optymalizacja bayesowska.



Rysunek 4: Wykresy tunability dla poszczególnych parametrów w modelu KNN dla metody random search (po lewej) i optymalizacji bayesowskiej (po prawej) w zależności od zbioru danych. Wykresy sa bardzo zbliżone i skupione blisko średniej.

3.3 Las losowy

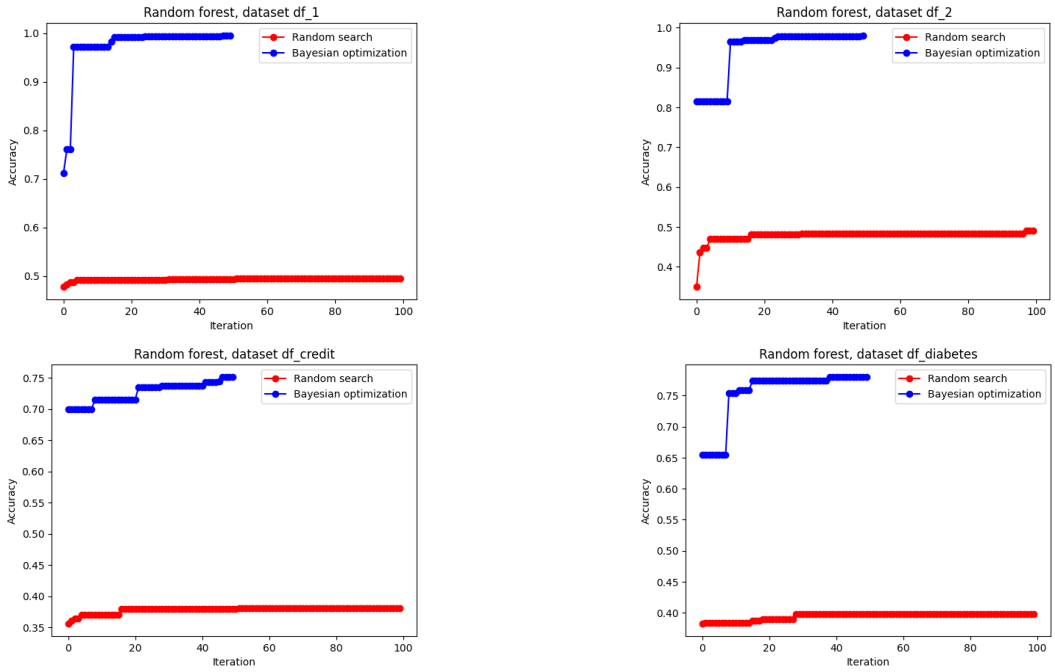
Dla lasu losowego optymalizowaliśmy parametry $n\_estimators$, min_sample_leaf i min_sample_split. Zgodnie z sugestia artykułu, zastosowaliśmy pewne przekształcenia drugiego i trzeciego z nich. Z jakiegoś powodu (być może błednie zaimplementowanych przekształceń, ale nie udało nam sie dojść do sedna problemu) metoda random search osiagała bardzo słabe wyniki, ich porównanie z dużo lepszya optymalizacja bayesowska można zobaczyć na wykresach 5.

W zwiazku ze specyfika zadania, dodatkowe modele służace do wyliczenia tunowalności modelu były liczone poprzez pipeline powiazany z random searchem, czyli nawet tunability fragmentu powiazanego z optymalizacja bayesowska było liczone w ten sposób. Możemy zatem przynajmniej zaobserwować zachowanie tunability w sytuacji, gdy wartość domyślna (wyliczona z bayesian optimization) jest znacznie lepsza niż wartości teoretycznie optymalne lokalnie. W tabeli 3 możemy zobaczyć poszczególne wartości. Widać, że o ile dla metody random search tunability jest śladowe, dla optymalizacji bayesowskiej jest ujemne o wysokiej wartości bezwzglednej, co wynika właśnie z tego, że rozwiazanie domyślne okazało sie znacznie lepsze niż rozwiazanie teoretycznie lokalnie optymalne.

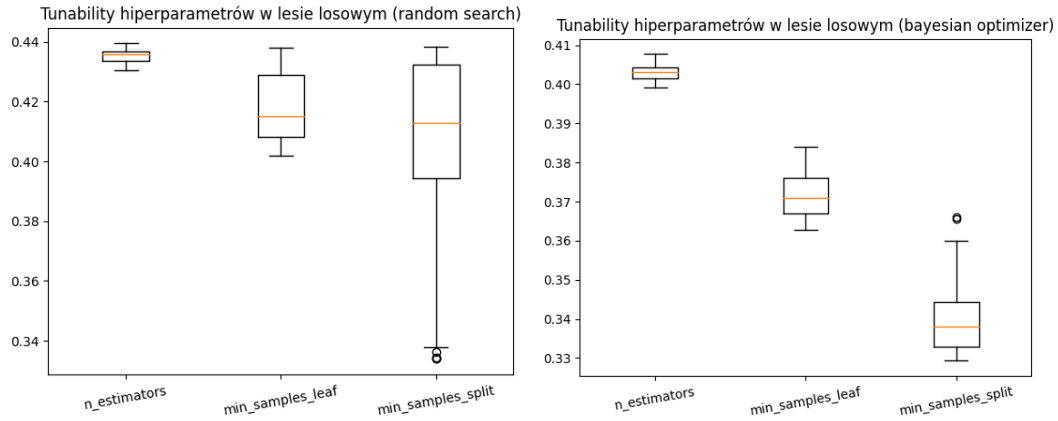
Ciekawie wygladaja również wykresy pudełkowe 6. Widać, że w tym przypadku wystepuje duża wariancja wyników, co oznacza, że w tym modelu, w tej implementacji, zmiana wartości któregokolwiek parametru (szczególnie $min\_samples\_split$), może spowodować duża zmiane wartości funkcji celu.

	n_estimators	**min_samples_leaf**	**min_samples_split**
Random search	0.0093	0.0068	0.0072
Bayesian optimization	-0.4660	-0.4827	-0.4868

Tabela 3: Wartości ryzyka dla modelu lasu losowego



Rysunek 5: Wykresy przedstawiajace przebiegi czasowe algorytmu random search (czerwony) oraz bayesian optimization (niebieski) dla modelu KNN. Jak widać, algorytmy uzyskiwały podobne wyniki na dwóch pierwszych zbiorach danych, a na dwóch pozostałych raz lepsze wyniki osiagnał random search, a raz optymalizacja bayesowska.



Rysunek 6: Wykresy tunability dla poszczególnych parametrów w modelu KNN dla metody random search (po lewej) i optymalizacji bayesowskiej (po prawej) w zależności od zbioru danych. Wykresy sa bardzo zbliżone i skupione blisko średniej.

4 Wnioski

Z przeprowadzonych przez nas eksperymentów wynika, że tunowalność parametrów jest cecha ściśle powiazana z modelem oraz zbiorem danych, na którym go trenujemy. Jak można zobaczyć na przykład na wykresie pudełkowym powiazanym z modelem drzewa decyzyjnego 4, dwa pierwsze zbiory danych indukowały dużo wieksza zmienność niż dwa ostatnie, podobnie dla KNN. W zwiazku z wyraźnym problemem z modelem lasu losowego, postanowiliśmy nie bazować na nim, jeśli chodzi o ostateczne wnioski, jest jednakże ciekawym obiektem do obserwacji zachowania funkcji ryzyka w przypadku dużej rozbieżności w wynikach pomiedzy modelem z parametrami domyślnymi, a modelem dotunowanym. Również czas potrzebny na stabilizacje algorytmu różni sie zależnie od samego algorytmu optymalizacji, wybranego modelu i zbiorów danych.

Na podstawie naszego eksperymentu nie można stwierdzić znaczacych różnic w tunowalności pomiedzy dwoma metodami tuningu hiperparametrów.

Literatura

[Head et al., 2021] Head, T., Kumar, M., Nahrstaedt, H., Louppe, G., and Shcherbatyi, I. (2021). scikit-optimize/scikit-optimize.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Probst et al., 2019] Probst, P., Boulesteix, A.-L., and Bischl, B. (2019). Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1):1934–1965.

[Vanschoren et al., 2013] Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. (2013). Openml: networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60.