Faculty of Mathematics and Information Science

WARSAW UNIVERSITY OF TECHNOLOGY

Praca Domowa 1

AutoML

Sabina Sidarovich

Spis treści

1 Cel

Celem eksperymentu jest analiza tunowalności hiperparametrów trzech algorytmów uczenia maszynowego (decision tree, random forest, K-nearest neighbors) na co najmniej czterech zbiorach danych, wzorując się na pracy naukowej Tunability: Importance of Hyperparameters of Machine Learning Algorithms [PBB18]. W ramach eksperymentu zostaną wykorzystane Random search i Bayes Optimization.
Eksperyment składa się z kilku kroków, obejmujących przygotowanie danych, ustalenie jednolitej siatki hiperparametrów, przeprowadzenie tuningu hiperparametrów dla każdej techniki losowania punktów na każdym zbiorze danych, oraz zapisanie historii tuningu dla analizy. W analizie wyników zostaną uwzględnione m.in. liczba iteracji potrzebna do uzyskania stabilnych wyników, ocena tunowalności algorytmów i hiperparametrów, a także porównanie wyników pomiędzy technikami losowania punktów.

2 Dane

Zbiory danych użyte podczas eksperymentu pochodzą z portalu OpenML [Van+13] oraz są zbiorami klasyfikacji binarnej.

- Zbior 1: Spambase (ID: 44)
- Zbior 2: MagicTelescope (ID: 1120)
- Zbior 3: Nomao (ID: 1486)
- Zbior 4: Phoneme (ID: 1489)

2.1 Przygotowanie danych

Zmienną docelową przemapowano na wartości binarne przed rozpoczęciem procesu treningu modeli. Do selekcji cech wykorzystano algorytm Boruta. Stworzono dedykowany pipeline, który zapewnia jednolite przetwarzanie dla każdego zestawu danych. W trakcie treningu modeli skorzystano z wcześnej wspomnianego pipeline'a, aby uniknąć biasu. Cechy numeryczne zostały poddane eliminacji wartości odstających i skalowaniu, natomiast cechy kategoryczne zostały zakodowane przy użyciu kodowania one-hot. W przypadku brakujących wartości zastosowano zastępowanie ich najczęściej występującą wartością.

3 Wybór domyślnych hiperparametrów

Hiperparametry oraz ich zakresy zostały wybrane zgodnie z propozycjami z artykułu [PBB18].

Hiperparametr	**Przestrzeń**
ccp_alpha	[0.0, 1.0]
max_depth	1 - 30
min_samples_leaf	1 - 60
min_samples_split	2 - 60

Tabela 1: Przestrzeń przeszukiwań dla algorytmu Decision Tree Classifier

Hiperparametr	**Przestrzeń**
n_neighbors	1 - 30

Tabela 2: Przestrzeń przeszukiwań dla algorytmu K-nearest Neighbors

Hiperparametr	Przestrzeń
n_estimators	1 - 2000
max_depth	1 - 15
min_samples_leaf	1 - 60
min_samples_split	2 - 60

Tabela 3: Przestrzeń przeszukiwań dla algorytmu Random Forest Classifier

3.1 Random search

W celu dokonania wyboru parametrów domyślnych, utworzono klasę *DefaultRandomOptimizer*. Klasy *DTCRandomOptimizer*, *KNNRandomOptimizer* i *RFRandomOptimizer* dziedziczą po klasie *DefaultRandomOptimizer*. Kluczową metodą jest funkcja *optimize*. Algorytm tej metody został zdefiniowany w następujący sposób:

1: **for** i **in** range(n) **do**
2: $hyperparameters \leftarrow$ random_search($space$)
3: **for** each *dataset* **in** *datasets* **do**
4: Split *dataset* using Cross-Validation
5: Preprocess data
6: Train model
7: Calculate accuracy, brier, roc_auc
8: **end for**
9: Calculate mean scores
10: **end for**
11: Generate results

gdzie n zależy od wybranego modelu, a funkcja *random_search* odpowiada za losowanie parametrów z przestrzeni wspomnianej wyżej. W trakcie eksperymentu optymalne parametry domyślne zostały wyselekcjonowane z wykorzystaniem metryki ROC AUC. Jako najlepszy uznano zestaw parametrów, który osiągnął najwyższą średnią wartość tej metryki.

3.2 Bayesian optimization

Na każdym z zestawów przeprowadzono optymalizację bayesowską przy użyciu pakietu scikit-optimize, wykorzystując kroswalidację. Dokonano obliczeń wartości metryk ROC AUC, accuracy oraz brier. Optymalizacja prowadzona była pod kątem metryki ROC AUC. W rezultacie uzyskano cztery odrębne zestawy optymalnych hiperparametrów.

4 Tunowalność (dostrajalność) hiperparametrów

Tunowalność hiperparametru zdefiniowano jako

$$tunability = \frac{1}{n} \sum_{i}^{n=n\_datasets} best\_score_i - default\_score$$

gdzie *best_score* jest wyznaczany dla każdego ze zbiorów danych osobno. Score odpowiada funkcji ryzyka zdefiniowanej w [PBB18]. Przy użyciu wcześniej zastosowanej metody *optimize* zbadano tunowalność hiperparametrów. W tym kontekście przed zastosowaniem *random_search* ustawiono domyślne wartości wcześniej uzyskane dla wszystkich hiperparametrów, z wyjątkiem tych, które były przedmiotem analizy. Dla tych konkretnych parametrów ponownie przeprowadzono losowanie wartości z wcześniej zdefiniowanego zakresu. Tunowalność została oceniona zarówno dla zestawu otrzymanego z przeszukiwania losowego, jak i dla tych uzyskanych poprzez optymalizację bayesowską. Rolę funkcji ryzyka peni ROC AUC.

5 Wyniki

Poniżej zamieszczone są rezultaty eksperymentów, obejmujące przebieg czasowy, uzyskane wskaźniki tunowalności oraz wykresy tunowalności hiperparametrów. Liczba iteracji dla każdego z modeli została dobrana z uwzględnieniem złożoności obliczeniowej algorytmu oraz przestrzeni hiperparametrów. Optymalizacja bayesowska przy użyciu pakietu scikit-optimize wymaga znacznej mocy obliczeniowej, dlatego liczba iteracji jest istotnie mniejsza niż w przypadku losowego przeszukiwania.

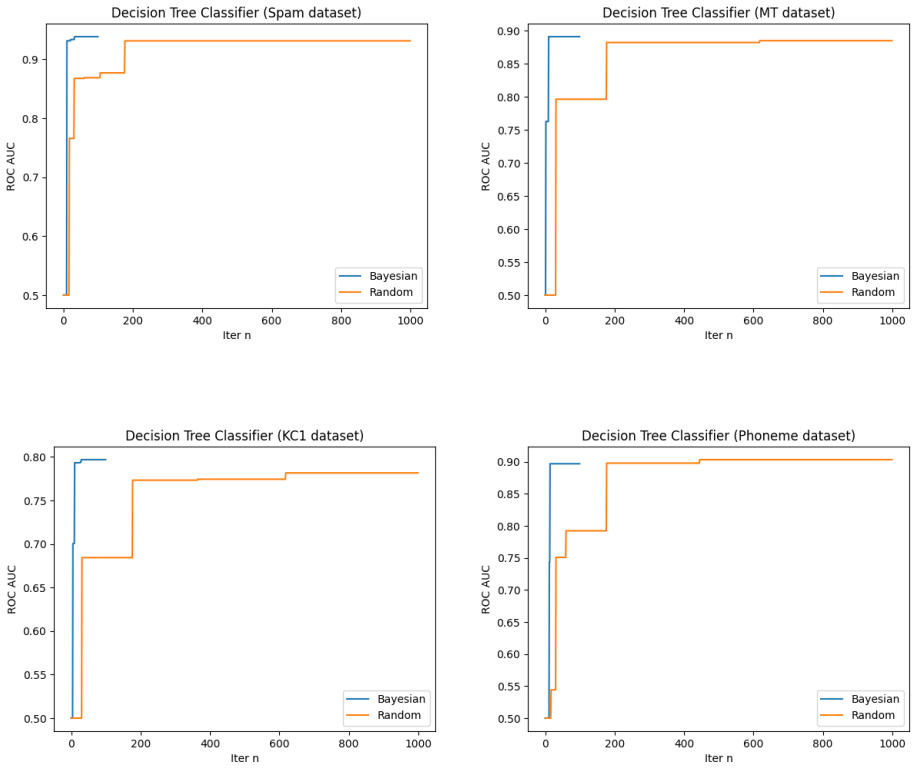
5.1 Decision Tree Classifier

Zbadano cztery parametry:

- cpp_alpha
- max_depth
- min_samples_leaf
- min_samples_split

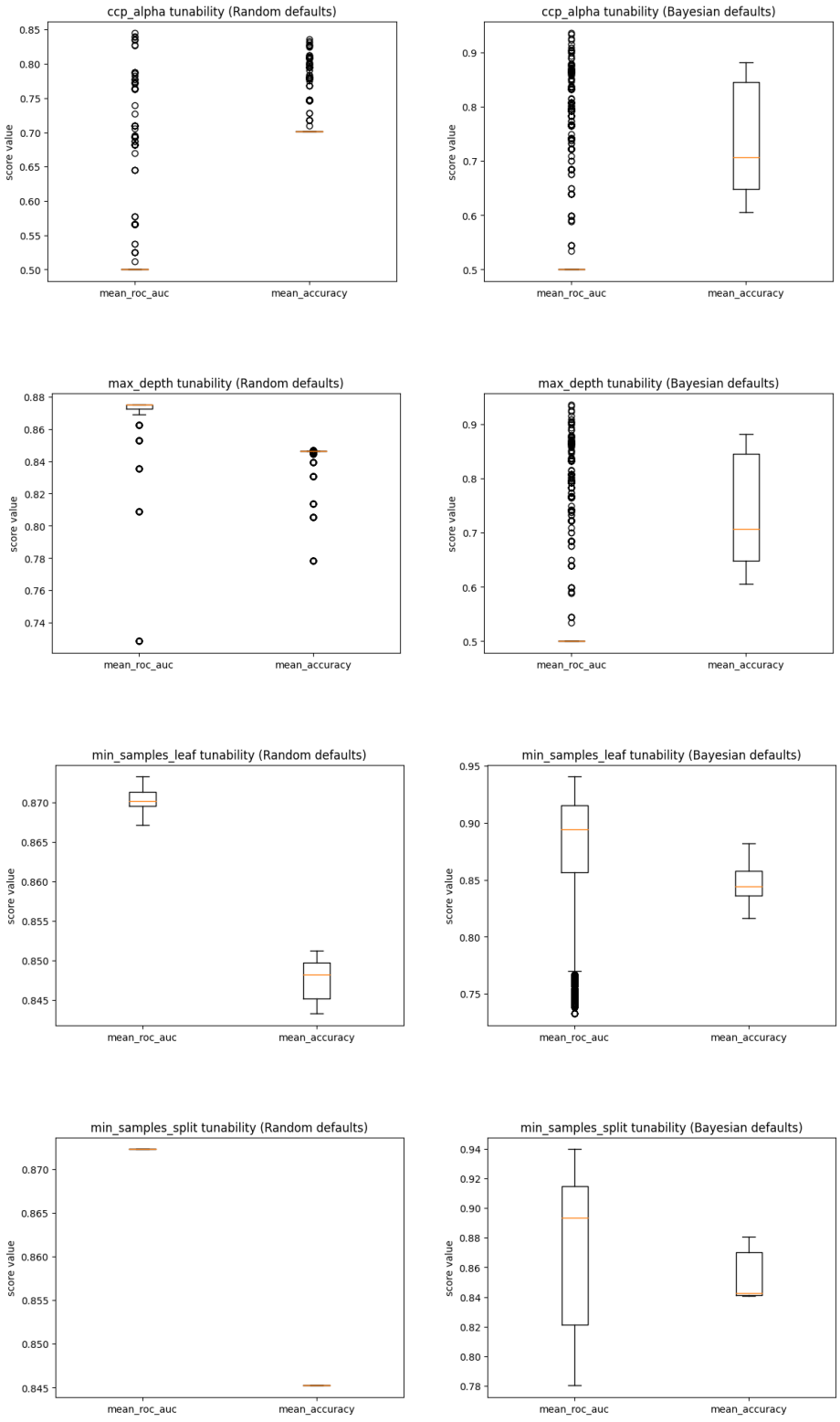
Przeprowadzono random search przy użyciu 1000 iteracji zarówno do ustalania wartości domyślnych, jak i do określenia tunowalności. W przypadku optymalizacji bayesowskiej obliczono 100 iteracji.

Zbadano zbieżność algorytmu, czyli tempo osiągania najlepszego wyniku, podczas ustalania wartości domyślnych. Prezentacja wyników została zawarta na rysunku 1.



Rysunek 1: Zbieżność algorytmu DTC

Wartości funkcji ryzyka są wyższe w przypadku optymalizacji bayesowskiej, która jednocześnie charakteryzuje się znacznie szybszą zbieżnością. Można również zwrócić uwagę na to, że w przypadku optymalizacji bayesowskiej skoki funkcji są większe.



Rysunek 2: Tunowalność hiperparametrów algorytmu DTC

Dla zestawu hiperparametrów wybranego za pomocą przeszukania losowego funkcja ryzyka podczas tunowania jednego hiperparametru jest bardziej stabilna, co dobrze widać na wykresach pudełkowych.

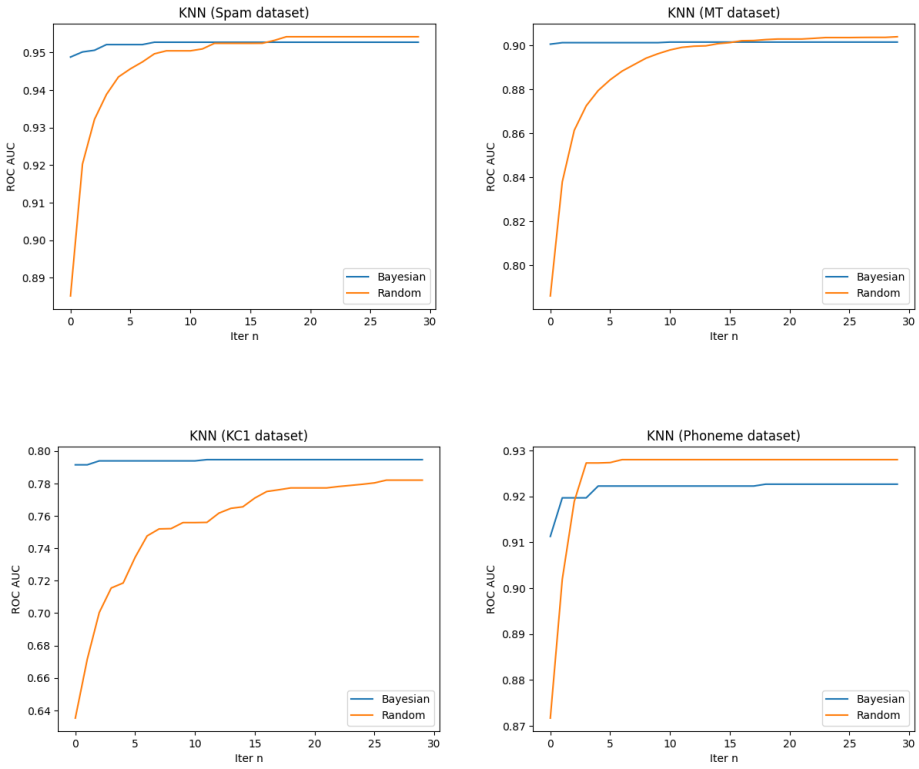
	ccp_alpha	max_depth	min_samples_leaf	min_samples_split
BO	0.0022	0.0084	0.0095	0.0062
RS	-0.0276	0.0046	0.0045	0.0000

Tabela 4: Tunowalność dla algorytmu DTC

5.2 K-nearest Neighbors Classifier

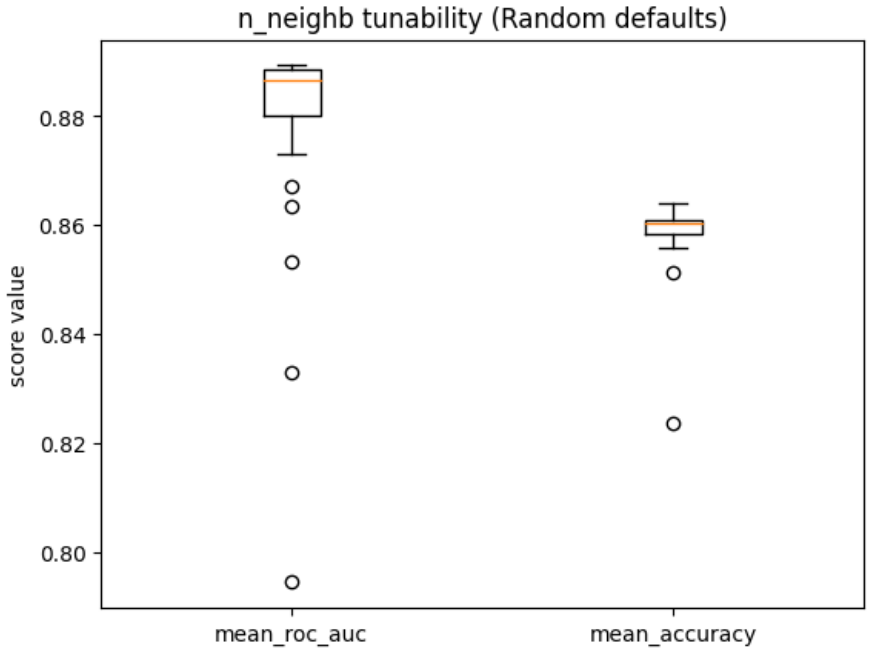
W niniejszym przypadku jedynym optymalizowanym hiperparametrem jest *n_neighbors*. Wartości tego parametru zostały wybierane w sposób pseudolosowy w przypadku metody random search, przy iteracji w zakresie od 1 do 30. W celu zachowania jednolitości eksperymentu optymalizację bayesowską również przeprowadzono z wykorzystaniem 30 iteracji.

Przebieg czasowy eksperymentu dla każdego z zestawów danych przedstawia się następująco:



Rysunek 3: Zbieżność algorytmu KNN

W tym przypadku optymalizacja bayesowska nie ma przewagi nad metodą random search.



Rysunek 4: Tunowalność hiperparametrów algorytmu KNN

	n_neighbors
BO	-0.001045
RS	0.002740

Tabela 5: Tunowalność dla algorytmu KNN

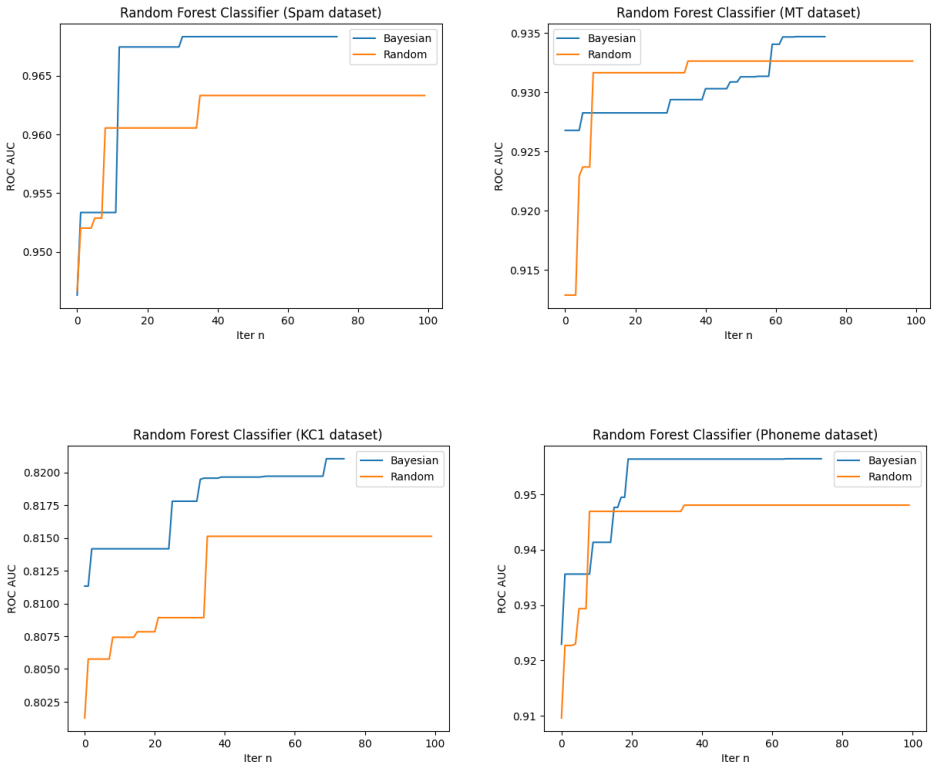
Zauważalne jest, że wartości domyślne dla algorytmu KNN wykazują lepszą skuteczność w porównaniu do tych, które zostały uzyskane w wyniku optymalizacji bayesowskiej. Hiperparametr n_neighbors ma duży wpływ na skuteczność algorytmu.

5.3 Random Forest Classifier

Ten model charakteryzował się największym zapotrzebowaniem obliczeniowym, dlatego liczba iteracji wyniosła 100 w przypadku poszukiwania wartości domyślnych przy użyciu random search, 75 dla optymalizacji bayesowskiej oraz 50 dla badania tunowalności. Zbadano następujące parametry:

- max_depth
- min_samples_leaf
- min_samples_split
- n_estimators

Zbieżność algorytmu przedstawiona jest na rysunku 5.



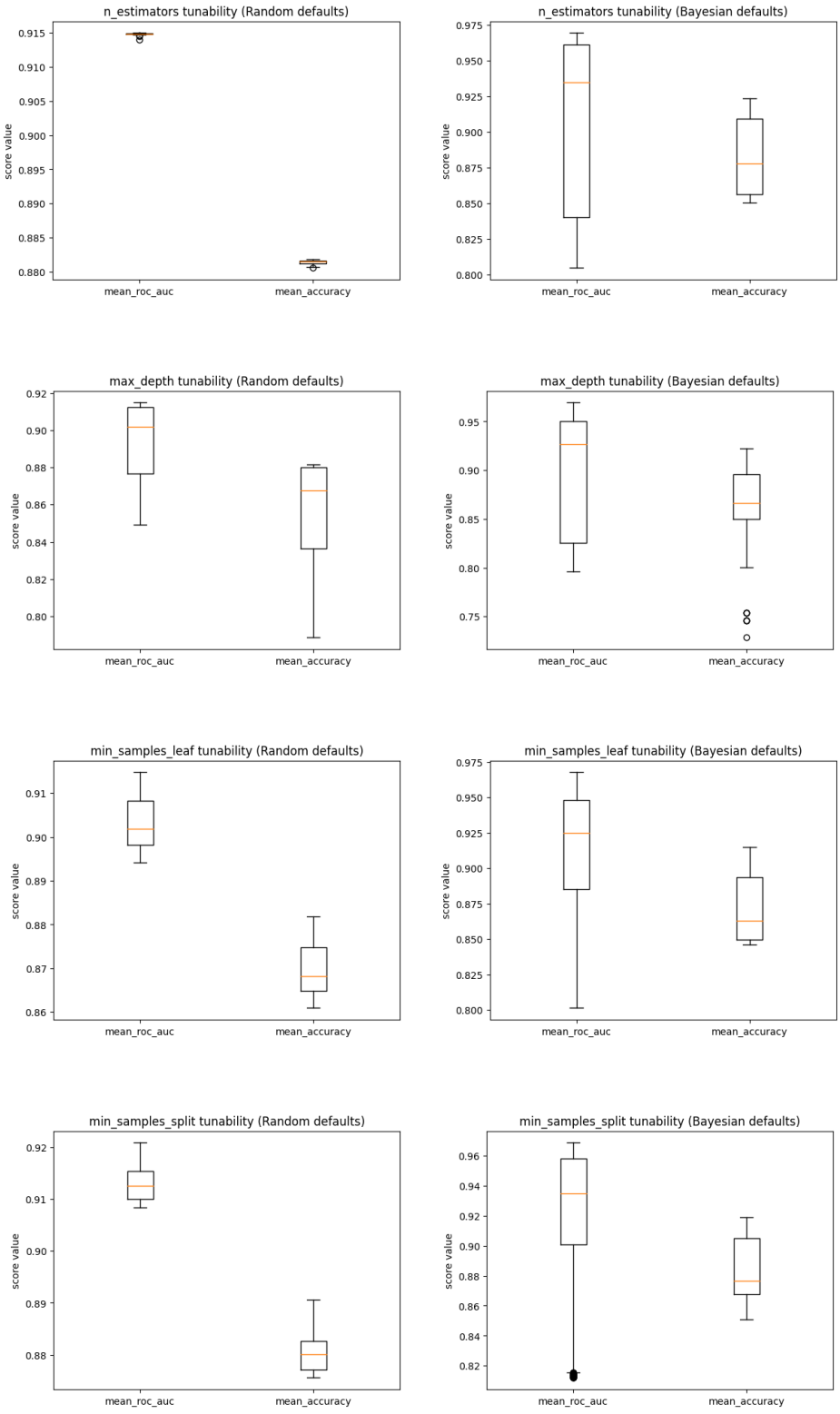
Rysunek 5: Zbieżność algorytmu RF

Optymalizacja bayesowska pozwala uzyskać nieznacznie lepsze wyniki, lecz czasem wymaga więcej iteracji.

	max_depth	**min_samples_leaf**	**min_samples_split**	**n_estimators**
BO	0.0032	0.0016	0.0032	0.0033
RS	0.0001	0.0000	0.0061	0.0003

Tabela 6: Tunowalność dla algorytmu RF

Ponownie możemy zaobserwować rożnice w zachowaniu funkcji ryzyka dla różnych metod wyboru wartości domyślnych. Tak jak wcześniej, jest ona bardziej stabilna dla przeszukiwania losowego.



Rysunek 6: Tunowalność hiperparametrów algorytmu RF

6 Wnioski

- Dla Decision Tree Classifier tunowanie min_samples_split nie daje lepszych wyników. Uzyskaną ujemną wartość tunability score dla ccp_alpha można wytłumaczyć małą liczbą iteracji.
- Dla KNN Random search okazał się być lepszy niż optymalizacja bayesowska.
- Liczba iteracji dla lasu losowego nie pozwala uzyskać satysfakcjonujących wyników, podobnych do tych przedstawionych w [PBB18].
- Optymalizacja bayesowska pozwala na szybsze uzyskanie wyników w przypadku modeli drzewiastych. Hiperparametry po optymalizacji są tunowalne.
- Problem tunowalności jest zależny od wybranej funkcji ryzyka oraz danych, co możemy zaobserwować na wykresach pudełkowych.
- Nie ma znacznych różnic w tunowalności pomiędzy przeszukiwaniem losowym a optymalizacją bayesowską.

Bibliografia

[Van+13] Joaquin Vanschoren i in. "OpenML: networked science in machine learning". W: *SIGKDD Explorations* 15.2 (2013), s. 49–60. DOI: 10.1145/2641190.2641198. URL: http://doi.acm.org/10.1145/2641190.264119.

[PBB18] Philipp Probst, Bernd Bischl i Anne-Laure Boulesteix. *Tunability: Importance of Hyperparameters of Machine Learning Algorithms*. 2018. arXiv: 1802.09596 [stat.ML].