AutoML - Praca domowa 1

Mieszko Mirgos, Anna Rutkiewicz

Listopad 2023

1 Wstęp

Celem tej pracy domowej było zbadanie tunowalności trzech wybranych modeli na czterech wybranych zbiorach danych, przy wykorzystaniu różnych technik samplowania zestawów hiperparametrów.

2 Eksperment

Do przeprowadzenia eksperymentu wybraliśmy problem klasyfikacji binarnej.

2.1 Szczegóły techniczne

W celu przeprowadzenia eksperymentu zdecydowaliśmy się na wybór popularnych zbiorów danyc z platformy openml, są zbiory *credit-g*, *blood-transfusion-service-center*, *steel-plates-fault*, *diabetes*.

Jako modele wybrane zostały *KNeighborsClassifier*, *LogisticRegression*, *ExtraTreesClassifier* z pakietu *scikit-learn*. Modele zostały wytrenowane na każdym zbiorze danych przy użyciu mechanizmu pipeline'ów z tego samego pakietu.

Do samplowania wybraliśmy algorytmy *RandomizedSearchCV* z pakietu *scikit-learn*, dla rozkładu jednostajnego, a jako bayesowski wybrano *HyperparameterOptimizationFacade* z pakietu *smac*.

2.2 Konstrukcja eksperymentu

W opisie eksperymentu pomijamy proces ściągania danych i instalacji paczek.

Podstawą trenowania i predykcji w eksperymencie były pipeline'y, pojedynczy pipeline obejmował pojedynczy dataset oraz jeden algorytm klasyfikacyjny. Preprocessing danych w pipelinie obejmował, dla danych numerycznych imputowanie wartościami średnimi oraz skalowanie do przedziału [0, 1], dla danych kategorycznych do imputowania została wykorzystana strategia most frequent, a także zostało przeprowadzone kodowanie *OneHotEncoding*.

Tak skonstruowane pipeline'y były wykorzystywane przez *RandomizedSearchCV* oraz *HyperparameterOptimizationFacade*. Dla wszystkich zbiorów danych wykorzystano tę samą siatkę hiperparametrów. Na jej podstawie dla każdego z modeli $i \in \{KNN, LogisticRegression, ExtraTrees\}$ doświadczalnie wyznaczono średnio najlepszy (gdzie miarą dla danego zestawu hiperparametrów była średnia z wyników zwróconych przez *Pipeline.score* dla każdego ze zbiorów danych) zestaw defaultowych hiperparametrów $\theta^{(i)*}$. Wyznaczono również miarę SN jako średnią z wyników uzyskanych przez wszystkie trzy modele z ustawionymi hiperparametrami zgodnie z $\theta^{(i)*}$.

3 Analiza wyników i odpowiedzi na pytania

Wyniki w tabeli 1 są wyliczone jako dokładność na zbiorze testowym dla modelu.

3.1 Liczba iteracji niezbędna do stabilnych wyników optymalizacji

Na podstawie tabeli 2 widać, że algorytmem najszybciej osiągającym najlepszy wynik była regresja logistyczna.

Model	credit-g	blood-transfusion-service-center	steel-plates-fault	diabetes	mean test score
KNN	0.71	0.99	0.79	0.72	0.80
LogisticRegression	0.78	0.81	1	0.71	0.69
ExtraTrees	0.75	0.67	0.99	0.66	0.7
KNN - Bayes	0.85	0.76	1	0.67	0.8
ExtraTrees - Bayes	0.7	0.76	0.65	0.64	0.69
LogisticRegression - Bayes	0.85	0.77	1	0.67	0.79

Tabela 1: Tabela przedstawia wyniki osiągnięte przez model dla danego zbioru danych

Model	credit-g	blood-transfusion-service-center	steel-plates-fault	diabetes
KNN	40	6	1	40
LogisticRegression	2	7	1	6
ExtraTrees	37	1	37	2

Tabela 2: Tabela przedstawia numer pierwszej iteracji, w której model po raz pierwszy osiągnął wynik równy najlepszemu.

W przypadku optymalizacji bayesowskiej oszacowano ilość iteracji metody w następujący sposób: optymalizacja ta korzystała ze 100 konfiguracji startowych, przy czym każda kolejna konfiguracja startowa była oparta o wyniki z poprzedniej. Dla każdej z tych konfiguracji startowych były przeprowadzane obliczenia, a także zmiany parametrów - jest to zatem pewien mnożnik liczby 100 wspomnianej wcześniej. Szacuje się zatem, że średnia liczba iteracji potrzebna optymalizacji bayesowskiej do wyznaczenia najlepszego możliwego modelu była znacznie większa niż w przypadku *RandomizedSearchCV*.

3.2 Określenie zakresów hiperparametrów dla poszczególnych modeli

Przy wyznaczaniu zakresów hiperparametrów dla poszczególnych modeli, kierowano się wartościami defaultowych hiperparametrów przyjmowanymi przez poszczególne modele (na podstawie oficjalnej dokumentacji - odpowiednio: źródła [2], [3], [4]). Po zapoznaniu się z dostępną literaturą wspomnianą poniżej, a także z oficjalną dokumentacją, wybrano na ich bazie zestawy hiperparametrów "najciekawszych" i potencjalnie mogących zauważalnie wpłynąć na wyniki. Starano się przy tym dla wartości kategorycznych uwzględnić jak największą ilość kategorii, a dla wartości numerycznych możliwie szeroki zakres wartości:

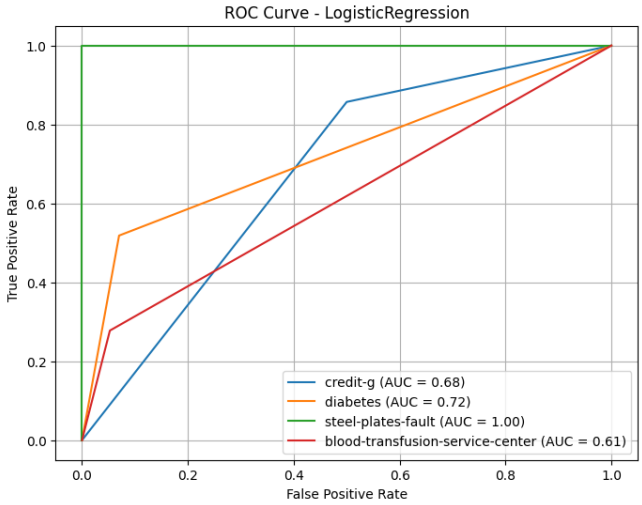
- Dla *LogisticRegression* sugerowano się tuningami hiperparametrów ze źródeł [5] oraz [7], oraz oficjalną dokumentacją.
- Dla *KNeighborsClassifier* sugerowano się wyliczonymi optymalnymi hiperparametrami w źródle [6] oraz oficjalną dokumentacją.
- Dla *ExtraTreesClassifier* sugerowano się tuningiem hiperparametrów ze źródła [7] oraz oficjalną dokumentacją.

Na Rysunku 1 przedstawiono krzywe ROC uzyskane dla rozpatrywanych modeli po zastosowaniu metody *RandomizedSearchCV*.

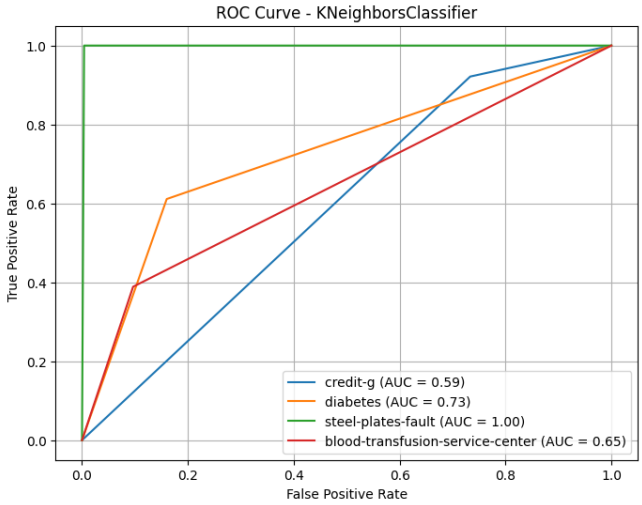
3.3 Tunowalność poszczególnych algorytmów. Bias sampling

W celu wyznaczenia tunowalności, zgodnie z artykułem [1], dla każdego z zastosowanych algorytmów i, obliczono dla każdej konfiguracji j następującą wartość: $d_j^i = mean\_test\_score_j^i - SN$, gdzie SN to stała opisana w sekcji 2.2, a $mean\_test\_score_j^i$ to średnia po zbiorach danych z wyników zwróconych dla danej konfiguracji j, dla danego modelu i, przez *Pipeline.score* - czyli miara dla zestawu hiperparametrów opisana w sekcji 2.2. Tak uzyskane d_j^i przedstawiono na wykresach na Rysunku 2.

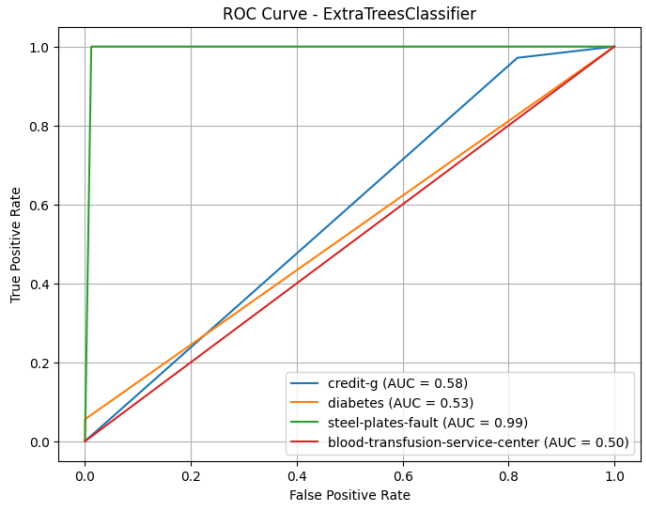
Na podstawie wykresu widzimy, że najbardziej tunowalny jest model *LogisticRegression*. Model *ExtraTrees* wykazuje bardzo małą podatność na zmiany hiperparametrów. Można również zauważyć, że optymalizacja za pomocą *RandomizedSearchCV* zapewniła największy przyrost (dla *LogisticRegression*), a optymalizacja bayesowska największy spadek (dla *ExtraTreesClassifier*). W związku z tym,



(a)

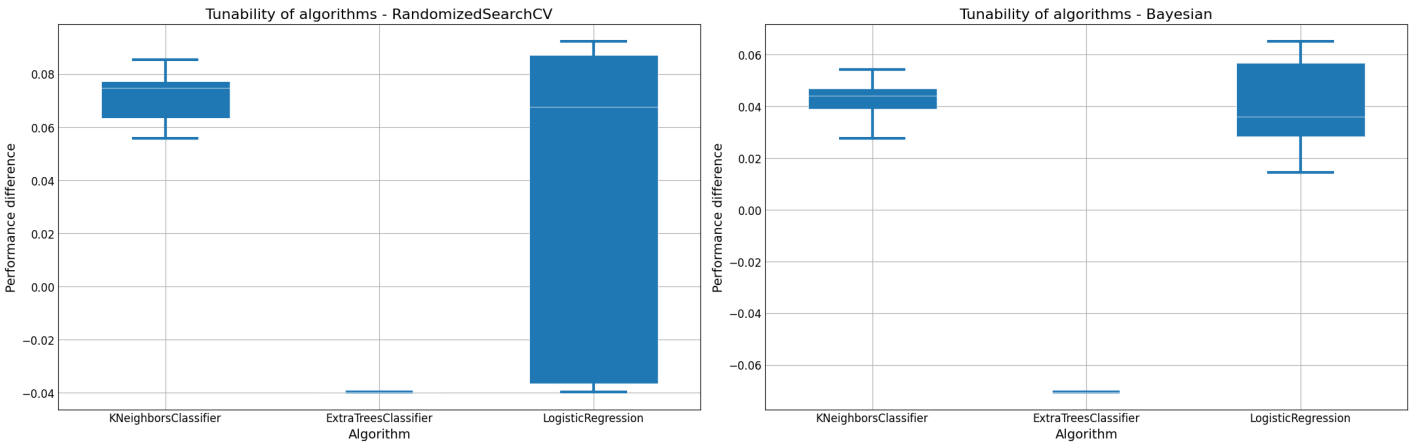


(b)



(c)

Rysunek 1: Rysunek przedstawia krzywe ROC dla każdego z trzech rozpatrywanych modeli - *LogisticRegression* - (a), *KNeighborsClassifier* - (b) oraz *ExtraTreesClassifier* - (c)
, uzyskane dla *RandomizedSearchCV*.



(a) (b)

Rysunek 2: Boxploty przedstawiające wyliczone tunowalności poszczególnych algorytmów dla *RandomizedSearchCV* - (a) oraz *HyperparameterOptimizationFacade* - (b).

biorąc pod uwagę koszty obu tych metod, zdecydowanie bardziej opłacalnym i efektywnym podejściem jest korzystanie z *RandomizedSearchCV* w celu tunowania hiperparametrów.

Sampling bias nie występuje dla zestawów hiperparametrów, ponieważ wszystkie ich elementy są wybierane albo z rozkładów równomiernych, albo są zmiennymi kategorycznymi o równym prawdopodobieństwie wylosowania. Więc wylosowanie pewnych wartości nie jest bardziej prawdopodobne od innych.

4 Bibliografia

1. *Tunability: Importance of Hyperparameters of Machine Learning Algorithms* - Philipp Probst, Anne-Laure Boulesteix, Bernd Bischl. Journal of Machine Learning Research 20 (2019) 1-32. Submitted 7/18; Revised 2/19; Published 3/19.

2. LogisticRegression official documentation

3. KNeighborsClassifier official documentation

4. ExtraTreesClassifier official documentation

5. *Hyperparameter Tuning for Machine Learning Algorithms Used for Arabic Sentiment Analysis.* Enas Elgeldawi, Awny Sayed, Ahmed R. Galal, Alaa M. Zaki. Special Issue Multimodal Data Processing and Semantic Analysis.

6. *Prediction and Analysis of Heart Disease Using Machine Learning.* 2021 IEEE International Conference on Robotics, Automation and Artificial Intelligence (RAAI).

7. *Multiple Heart Diseases Prediction using Logistic Regression with Ensemble and Hyper Parameter tuning Techniques.* Sateesh Ambesange, A. Vijayalaxmi, S. Sridevi, Venkateswaran, B. S. Yashoda. 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)