Analiza tunowalności hiperparametrów wybranych algorytmów uczenia maszynowego

Autorzy:

Mateusz Deptuch, Zofia Kamińska, Karolina Dunal

Na potrzeby:

Projekt z przedmiotu AutoML

Folder projektu:

AutoML-Projekt1

1. Wybór zbiorów danych i algorytmów

1.1. Zbiory danych i ich przygotowanie

Podczas eksperymentów będziemy korzystać z 3 zbiorów danych pochodzących z platformy *Kaggle*:

- ***Car Purchase Dataset*** *(1000 wierszy, 5 kolumn),*
- ***Hiring Dataset*** *(1500 wierszy, 11 kolumn),*
- ***Diabetes Dataset*** *(768 wierszy, 10 kolumn),*

oraz 1 zbioru pochodzącego z platformy *OpenML*:

- ***Banknote Authentication Dataset*** *(1372 wierszy, 6 kolumn).*

Wszystkie z wymienionych zbiorów służą do zadania klasyfikacji binarnej. W żadnym z nich nie stwierdzono braków danych. W zbiorze ***diabetes*** dokonano imputacji wartości, wykorzystując medianę, aby zastąpić wartości zerowe pojawiające się w kolumnach, gdzie były one niepoprawne. Zmienne kategoryczne zostały zakodowane za pomocą `LabelEncoder`, a zmienne numeryczne poddano standaryzacji przy użyciu `StandardScaler`.

1.2. Wybrane algorytmy

Do analizy wybraliśmy 3 różne modele: **Random Forest** oraz **Logistic Regression** pochodzące z pakietu **scikit-learn** (sklearn) a także model **XGBoost,** który pobrany z pakietu **XGBoost**.

2. Optymalizacja - eksperymenty

2.1. Wybór zakresów parametrów

Zakresy wartości parametrów dla wybranych modeli dobraliśmy w celu umożliwienia równowagi między wydajnością a dokładnością modelu, uwzględniając różne poziomy szczegółowości i generalizacji. *(patrz: Tabela 1)*

2.2. Tuning parametrów przy użyciu Random Search

Zakresy przyjęte dla poszczególnych hiperparametrów zostały określone powyżej. Kluczowym kryterium optymalizacji była miara **AUC**, ponieważ dobrze sprawdza się w klasyfikacji binarnej oraz jest dość odporna na problem niezbalansowanych klas.

W celu zapewnienia powtarzalności wyników, parametr `random_state` został ustawiony na wartość **42**. Pozwoliło to na uzyskanie spójnych wyników przy kolejnych uruchomieniach tego samego kodu, co jest istotne przy ocenie i porównywaniu modeli. Przeszukiwanie parametrów przeprowadzono dla **300 losowych kombinacji** z ustalonego zakresu.

W celu wyłonienia najlepszego zestawu hiperparametrów (default*), obliczono średni wynik AUC uzyskany przez każdą kombinację hiperparametrów na wszystkich czterech zbiorach danych. Najlepszy zestaw hiperparametrów został wybrany jako ten, który osiągnął najwyższą średnią wartość AUC, wskazując na stabilną skuteczność na różnych zbiorach danych. Poniżej prezentujemy konfigurację wybranych hiperparametrów, która uzyskała najwyższą średnią miarę AUC na wszystkich czterech zbiorach dla poszczególnych modeli - będzie to nasz nowy default*. *(patrz: Tabela 2)*

Aby dokonać analizy metody *RandomSearch* oraz stwierdzić, na który z modeli, tego typu optymalizacja ma największy wpływ, przedstawiliśmy na poniższych wykresach zależność między liczbą iteracji a najlepszym wynikiem AUC dla każdego datasetu. *(patrz: Wykres 1)*

W pierwszych dziesięciu iteracjach *RandomSearch* zauważamy największy przyrost wyników, choć różni się on w zależności od modelu. Dla *RandomForest* poprawa jest minimalna, a dla *LogisticRegression* prawie niezauważalna. W przeciwieństwie do nich, *XGBoost* odnotowuje znaczący wzrost - średnia wartość AUC w pierwszych dziesięciu iteracjach zwiększa się aż o **0.270 (± 0.132)**. Dla *RandomForest* jest to **0.00356 (± 0.00388)** , a dla *LogisticRegression* zaledwie **0.000820 (± 0.000743)**. Możemy zatem wywnioskować, że tunowalność *XGBoost* będzie prawdopodobnie znacznie większa niż w przypadku pozostałych modeli.

Po około 50 iteracjach, *RandomSearch* osiąga niemal pełną stabilizację wyników dla każdego z modeli. Użycie *RandomSearch* jest więc całkiem optymalne, ponieważ pozwala na szybkie i efektywne przeszukiwanie przestrzeni hiperparametrów, osiągając wysoką jakość modelu bez konieczności przeprowadzania kosztownych i czasochłonnych obliczeń wymaganych przez inne metody, takie jak np. *GridSearch*.

W celu oceny rozkładu wyników AUC dla różnych kombinacji hiperparametrów, stworzyliśmy wykresy skrzypcowe, na których wyróżniliśmy również wynik osiągany dla wyznaczonego zestawu parametrów defaultowych (default*). *(patrz: Wykres 2)*

Jak możemy zaobserwować, dla *RandomForest* wahania w osiąganych wynikach są znacznie mniejsze w porównaniu do *LogisticRegression* oraz *XGBoost*, a defaultowe parametry dają wyniki zbliżone do tych osiąganych przez większość rozważanych kombinacji. Dla *LogisticRegression* pomimo większych odchyleń w wynikach defaultowe parametry osiągają skuteczność porównywalną ze skutecznością dla większości rozważanych kombinacji. Najciekawsze natomiast jest zachowanie modelu *XGBoost*, dla którego znaczna większość zestawów hiperparametrów osiąga zauważalnie niższe wyniki niż dla zestawu defaultowego. Gorsze wyniki *XGBoost* przy większości kombinacji hiperparametrów mogą wynikać z dobrze dostrojonych parametrów domyślnych, nieoptymalnego zakresu siatki, ryzyka przeuczenia oraz dopasowania domyślnych ustawień do specyfiki danych. Nie jesteśmy w stanie jednoznacznie określić co powoduje występowanie tego zjawiska.

2.3. Tuning parametrów przy użyciu Bayes Optimization

Do algorytmu przekazano te same zakresy hiperparametrów, co w przypadku Random Search. Proces optymalizacji był uruchamiany w pętli **50 razy**, **z różnymi punktami startowymi**, co osiągnięto przy ustawieniu `random_state` na wartość ***42 + i*** (gdzie *i* to iteracja pętli). Taki zabieg pozwolił na zbadanie zbieżności algorytmu, umożliwiając ocenę stabilności i skuteczności znalezionych rozwiązań w kontekście różnych inicjalizacji.

Poniżej przedstawione są wykresy pokazujące zależność wyniku od losowo wybranych punktów startowych. *(patrz: Wykres 3)*

Modele *LogisticRegression*, *RandomForest* i *XGBoost* reagowały różnie na wybór punktów startowych. *LogisticRegression* był najmniej podatny na zmiany, z niemal stałymi wartościami AUC w każdej iteracji. *RandomForest* wykazał umiarkowaną zależność, z niewielkimi fluktuacjami w AUC. Natomiast *XGBoost*, zwłaszcza w przypadku bardziej złożonych zbiorów danych, wykazał największe wahania wyników, i możemy zaobserwować dużą zależność od początkowych punktów startowych. W pętli Bayesowskiej każda iteracja zewnętrzna obejmowała 30 kroków wewnętrznej optymalizacji (n_iter = 30). Ponieważ jednak zapisane przez nas dane dotyczą tylko iteracji zewnętrznych, trudno określić, ile iteracji w wewnętrznej pętli jest faktycznie potrzebnych do uzyskania stabilnego wyniku.

3. Analiza wyników *(patrz: Tabela 3)*

Eksperyment pokazał, że *Bayesian Optimization* daje minimalnie wyższe wyniki AUC niż *Random Search*, jednak różnice te są często niewielkie, zwłaszcza dla prostszych modeli takich jak *LogisticRegression.*

1. **Logistic Regression – minimalna tunowalność**
   W przypadku tego modelu tuning przyniósł jedynie znikome przyrosty efektywności, co może wynikać z ograniczonej liczby parametrów, które mają wpływ na jego wyniki.
2. **Random Forest – umiarkowana tunowalność**
   Random Forest wykazał lepszą responsywność na tuning, zwłaszcza przy Bayesian Optimization, ale wzrosty efektywności nadal nie były znaczące poza bardziej wymagającymi zbiorami.
3. **XGBoost – największa tunowalność**
   Spośród testowanych modeli, *XGBoost* najlepiej wykorzystał optymalizację Bayesian, co przełożyło się na większy wzrost AUC, szczególnie na bardziej wymagających danych.

Występowanie zjawiska bias samplingu
W metodzie Random Search punkty losowane są z określonych rozkładów, co oznacza, że przy użyciu rozkładów innych niż jednostajne, wyniki będą obarczone biasem wynikającym z charakterystyki danego rozkładu. W przypadku podejścia bayesowskiego występuje bias związany z tym, że proces optymalizacji zbiega w kierunku najbardziej obiecujących obszarów

przestrzeni parametrów. W miarę postępu iteracji próbkowanie zaczyna koncentrować się w okolicach punktów, które według modelu są optymalne, co prowadzi do eksploracji podobnych obszarów.

Podsumowanie – efektywność optymalizacji

Bayesian Optimization warto stosować dla bardziej złożonych modeli, takich jak *XGBoost*, gdzie może przynieść realne korzyści. W prostszych przypadkach *Random Search* pozostaje bardziej efektywnym wyborem, jako że daje zbliżone rezultaty, przy niższych kosztach obliczeniowych oraz znacznie szybszym czasie wykonania algorytmu przeszukiwania. Warto jednak zauważyć, że *RandomForest* z default parametrami osiąga lepsze wyniki niż najlepiej dopasowany *XGBoost*. Zatem, zamiast koncentrować się na intensywnym strojeniu hiperparametrów, czasem warto rozważyć bardziej strategiczny dobór modelu.

4. Dodatek: czy to wszystko ma sens?

Postanowiliśmy również porównać, jak modele radzą sobie z domyślnymi hiperparametrami - ustawionymi przez twórców poszczególnych pakietów.
W poniższej tabeli zamieszczone są średnie wyniki dla 3-krotnej kroswalidacji dla modeli z domyślnymi hieperparametrami porównane z wynikami wcześniejszych eksperymentów.
(patrz: Tabela 4)

Jak widać, jedynie w 4 przypadkach na 12, obliczone wcześniej hiperparametry default* osiągały lepsze wyniki niż domyślne hiperparametry modelu a i tak poprawa wyniku jest relatywnie mała. Choć parametry domyślne mogły zostać osiągnięte za pomocą metod *RandomSearch* i *BayesSearch* (były zawarte w przedziałach hiperparametrów podanych do modeli), liczba wykonanych iteracji była prawdopodobnie zbyt mała, aby uzyskać kombinację równą bądź zbliżoną do domyślnej.

Gdy optymalizacja została przeprowadzona dla konkretnego zbioru danych, zaobserwowano niewielki wzrost liczby przypadków, w których zoptymalizowane parametry przewyższały wyniki osiągane przy użyciu parametrów domyślnych; jednakże różnice te były marginalne. W związku z tym, jeśli celem jest uzyskanie jak najlepszych efektów, bardziej uzasadnione może być profilowanie modeli pod kątem konkretnych zbiorów danych, niż dążenie do znalezienia nowych, globalnie optymalnych parametrów. Domyślne parametry oferowane przez twórców pakietów okazują się być dobrze dostrojone, co prawdopodobnie wynika z ich opracowania na szerokiej próbie różnorodnych zbiorów danych oraz przetestowania w ramach szerokich zakresów hiperparametrów. W efekcie można mieć duże zaufanie do ich skuteczności.

Tabela 1 - wybrane zakresy parametrów

Regresja logistyczna

Parametr	Zakres wartości
solver	[‘saga’]
tol	loguniform(1e-6, 1e-4)
penalty	[‘elasticnet’, ‘l1’, ‘l2’, None]
C	loguniform(0.001, 100)
max_iter	randint(10, 501)
class_weight	[‘balanced’, None]
l1_ratio	uniform(0,1)

Random Forest

Parametr	Zakres wartości
n_estimators	[32, 64, 128, 256, 512]
max_depth	[1, 2, ..., 15]
min_samples_split	[2, 4, 8, 16]
min_smaples_leaf	[1, 2, 4, 8]
max_features	[0.1, 0.2, ..., 1.0]
bootstrap	[True, False]
criterion	['gini', 'entropy', log-loss]

XGBoost

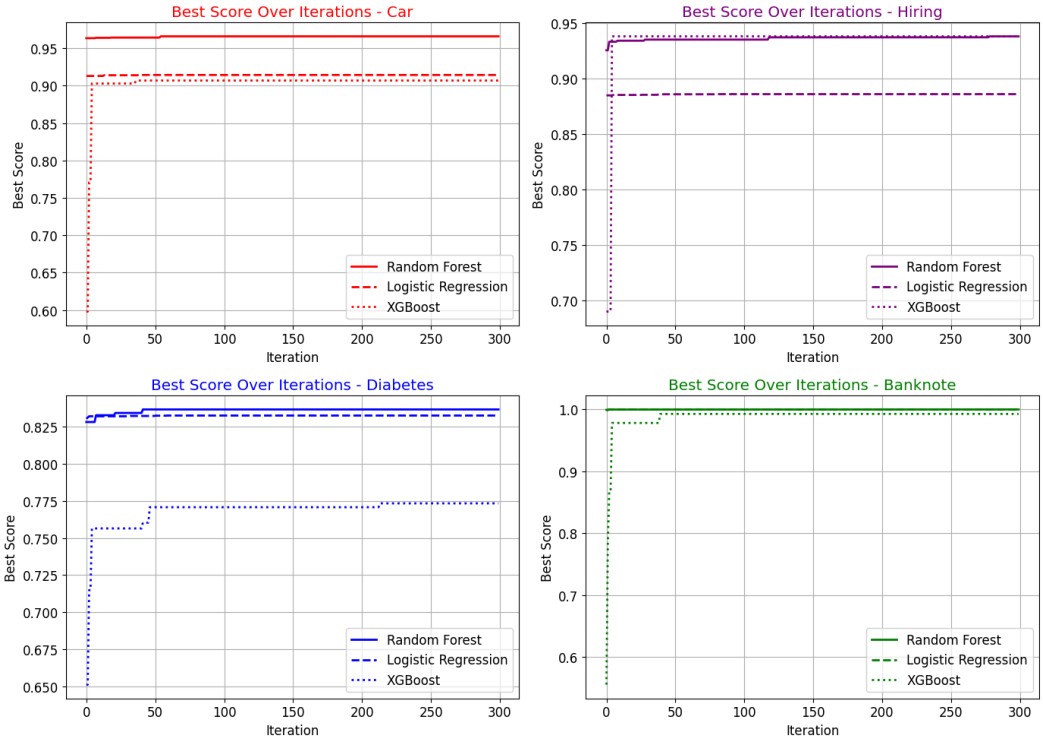
Parametr	Zakres wartości
booster	['gbtree', 'gblinear', 'dart']
eta	loguniform(1e-6, 1)
max_depth	[1, 2, …, 15]
min_child_weight	[1, 2,...,20]
subsample	uniform(0,1)
colsample_bytree	uniform(0,1)
sampling_method	[‘uniform’]
gamma	loguniform(1e-6, 1e3)
lamba	loguniform(1e-6, 1e3)
alpha	loguniform(1e-6, 1e3)

*Tabela 2 - parametry default**

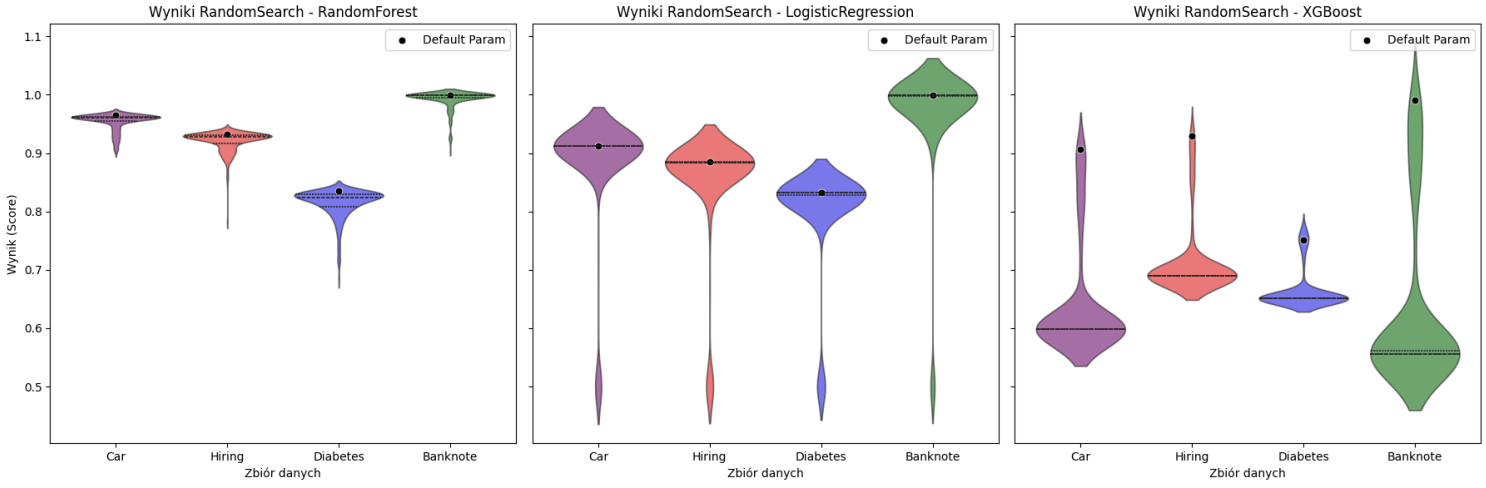
Regresja logistyczna	**XGBoost**	**Random Forest**
solver: saga	*booster*: gbtree	*n_estimators*: 256
penalty: elasticnet	*eta*: 0.07906	*min_samples_split:* 8
C: 0.271463	*max_depth*: 10	*min_samples_leaf*: 4
max_iter: 235	*min_child_weight*: 7	*max_features*: 0.2
tol: 0.00001	*subsample*: 0.897959	*max_depth*: 8
class_weight: None	*colsample_bytree*: 0.897959	*criterion*: log_loss
l1_ratio: 0.631139	*gamma*: 0.000029	*bootstrap*: False
	lambda: 0.323746	
	alpha: 0.000069	
Najlepszy średni* wynik AUC		
0.9075 ± 0.0697	0.8945 ± 0.0865	0.9327 ± 0.0701

**średnia wyników dla 4 różnych datasetów*

Wykres 1 - Najlepszy wynik AUC w kolejnych iteracjach RandomSearch



Wykres 2 - Rozkład wyników AUC (Score) w RandomSearch dla poszczególnych modeli



Wykres 3 - Zależność wyników AUC od wyboru punktów startowych

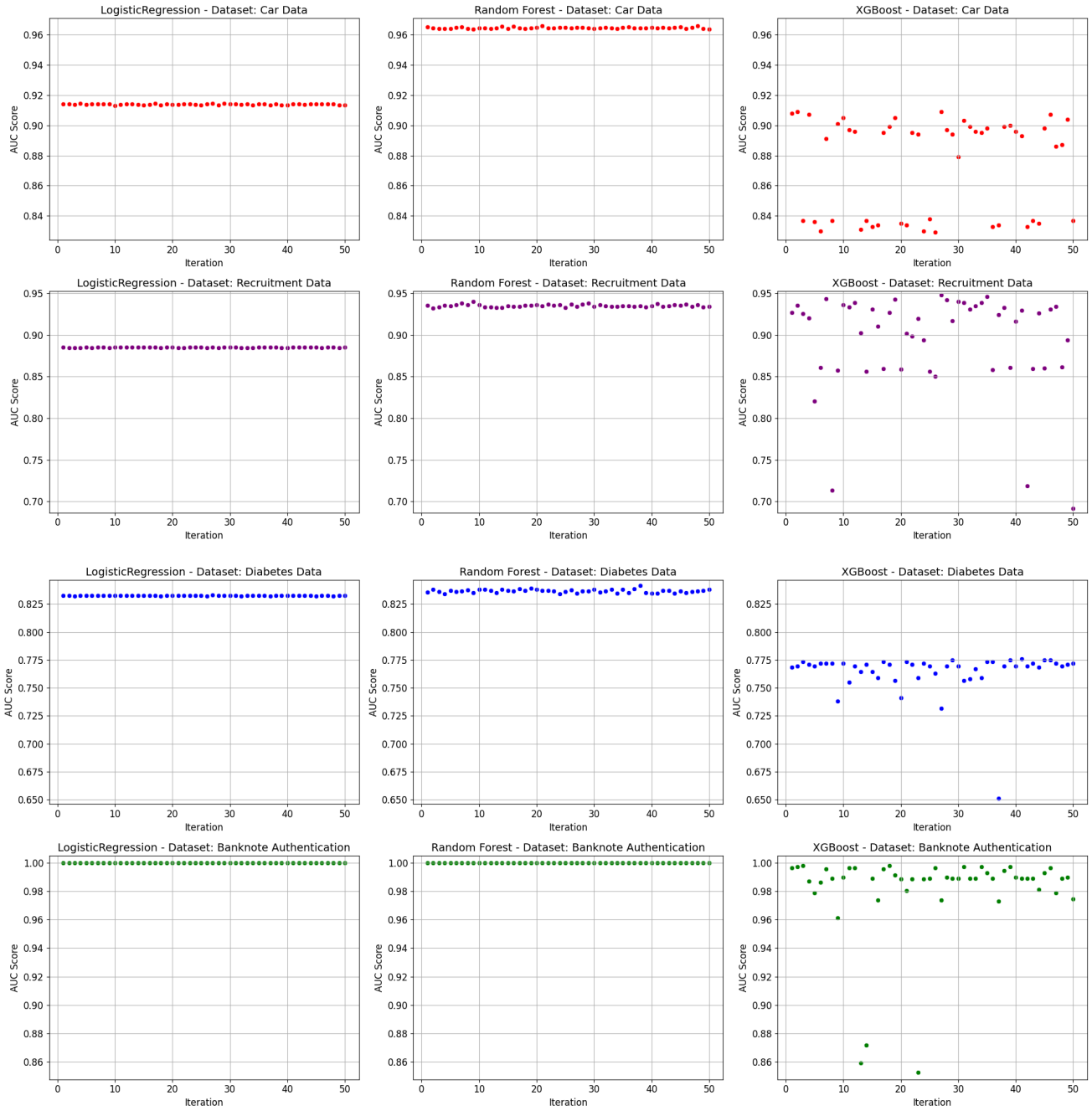


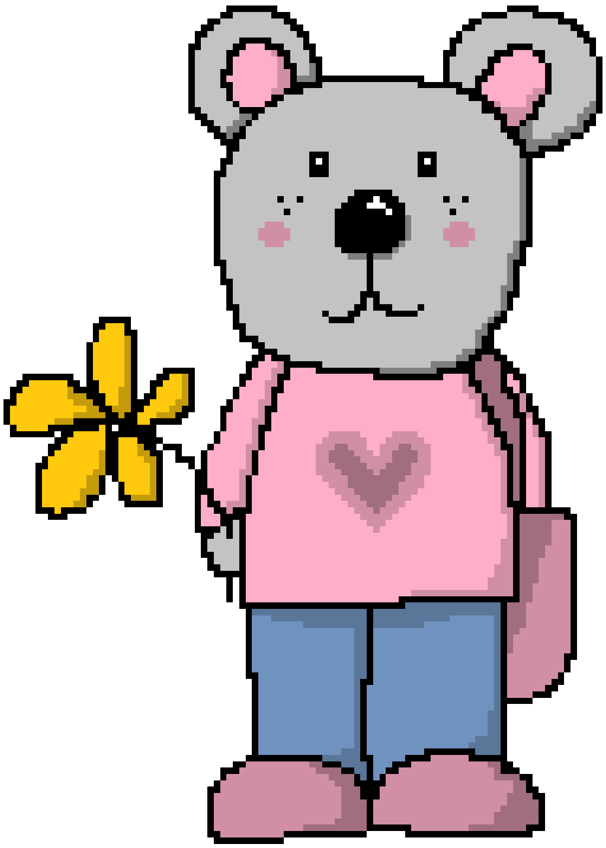
Tabela 3

TUNOWALNOŚC MODELI

dataset	default* params (AUC)	best params random search (AUC)	best params bayes opt. (AUC)	tunability random search	tunability bayes opt.
Logistic Regression					
car	0.9126	0.9145	0.9145	0.0019	0.0019
hiring	0.8853	0.8859	0.8857	0.0006	0.0004
diabetes	0.8326	0.8327	0.8328	0.0001	0.0002
banknote	0.9995	0.9998	0.9998	0.0003	0.0003
Random Forest					
car	0.9644	0.9661	0.9658	0.0017	0.0014
hiring	0.9322	0.9380	0.9406	0.0058	0.0084
diabetes	0.8345	0.8368	0.8413	0.0023	0.0068
banknote	0.9997	0.9999	0.9999	0.0002	0.0002
XGBoost					
car	0.9070	0.9070	0.9090	0.0000	0.0020
hiring	0.9293	0.9380	0.9480	0.0087	0.0187
diabetes	0.7513	0.7734	0.7760	0.0221	0.0247
banknote	0.9905	0.9927	0.9978	0.0022	0.0073

Tabela 4

dataset	default* params (AUC)	best params random search (AUC)	best params bayes opt. (AUC)	default params from package authors	default* - default	best random - default	best bayes - default
Logistic Regression							
car	0.9126	0.9145	0.9145	0.9133	-0.0007	**0.0012**	**0.0012**
hiring	0.8853	0.8859	0.8857	0.8872	-0.0019	-0.0013	-0.0015
diabetes	0.8326	0.8327	0.8328	0.8310	**0.0016**	**0.0017**	**0.0018**
banknote	0.9995	0.9998	0.9998	0.9996	-0.0001	**0.0002**	**0.0002**
Random Forest							
car	0.9644	0.9661	0.9658	0.9583	**0.0061**	**0.0078**	**0.0075**
hiring	0.9322	0.9380	0.9406	0.9336	**0.0014**	**0.0044**	**0.0070**
diabetes	0.8345	0.8368	0.8413	0.8227	**0.0118**	**0.0141**	**0.0186**
banknote	0.9997	0.9999	0.9999	0.9999	-0.0002	0.0000	0.0000
XGBoost							
car	0.9070	0.9070	0.9090	0.9599	-0.0529	-0.0529	-0.0509
hiring	0.9293	0.9380	0.9480	0.9402	-0.0109	-0.0022	**0.0078**
diabetes	0.7513	0.7734	0.7760	0.7881	-0.0368	-0.0147	-0.0121
banknote	0.9905	0.9927	0.9978	0.9999	-0.0094	-0.0072	-0.0021



z dedykacją dla Anny Kozak