Faculty of Mathematics and Information Science

WARSAW UNIVERSITY OF TECHNOLOGY

Automatyczne uczenie maszynowe

Analiza tunowalności hiperparametrów

Autorzy: Damian Lubaszka, Mikołaj Nowak

Kierunek: Informatyka i Systemy Informacyjne

Prowadząca laboratorium: dr mgr Katarzyna Woźnica

Warszawa, 17.11.2023 r.

Spis treści

1 Streszczenie

W niniejszym dokumencie zawarto analizę wyników uzyskanych podczas realizacji zadania. Ponadto w sposób skrupulatny został opisany proces przeprowadzonego doświadczenia oraz zostały opisane osiągnięte wnioski. Instrukcja uruchomienia kodu do wygenerowania historii tuningu oraz jego analizy wraz z wszelkimi uwagami dotyczącymi szczegółów implementacji algorytmów został zawarty w pliku **README.md**. Ponadto w kodzie zawarto komentarze, w miejscach, gdzie uznano, że są niezbędne. W pliku **Analysis.ipynb** również zawarto szczegółową interpretację wyników.

2 Zbiory danych

Zbiory danych użyte podczas realizacji zadania wybrano ze strony **open.ml**. Wybrano takie zbiory, które były dedykowane problemowi klasyfikacji binarnej, a klasa będąca targetem miała rozkład jednostajny. Wybrane zbiory danych to:

- MagicTelescope [1],
- online-shoppers-intention [2],
- bank-marketing [3],
- credit [4].

Podczas wyboru zbioru kierowano się tym, by liczba features była nie większa niż 20 oraz nie mniejsza niż 8, oraz by liczność zbioru danych mieściła się w zakresie od 10000 do 20000, a w zbiorze nie występowały brakujące dane.

Przy realizacji ćwiczenia pobrano dane ze strony **open.ml** bezpośrednio w kodzie notatnika jupyter notebook. Następnie podzielono każdy dataset na zbiór treningowy (80%) oraz testowy (20%).

3 Algorytmy uczenia maszynowego

Do realizacji zadania zdecydowano się wybrać następujące algorytmy uczenia maszynowego:

- Random Forest Classifier [5],
- Gradient Boosting Classifier [6].
- SVM Classifier [7],

Wszystkie algorytmy zostały użyte w implementacji dostarczonej przez bibliotekę **sklearn**. W tabeli 1 zawarto zoptymalizowane hiperparametry dla każdego modelu. Podczas definicji zakresów brane były pod uwagę dane z literatury: [8], [9], [10], [11], a także dokumentacja oraz czynnik długości obliczeń. Ponadto przetestowano wiele różnych zakresów i doświadczalnie wybrano najlepsze zakresy w oparciu o dostępne wyniki. Nie zachowano jednak historii historii tuningu, a jedynie historię tuningu dla finalnych zakresów. Dla SVM Classifier, ze względu na zależność hyperparametru **degree** od hyperparametru **kernel** postanowiono go pominąć, co pociągnęło ze sobą pominięcie wartości **poly** w hiperparametrze **kernel**. Dla Random Forest Classifier oraz Gradient Boosting Classifier posłużono się standardowo wybieranymi zbiorami parametrów przy problemie tunowalności. Zdecydowano się pominąć wartość **None** w hiperparametrze **max_features**, ze względu na chęć odnalezienia optymalnych hiperparametrów z uwzględnieniem ścisłego maksimum dotyczącym liczby cech branych pod uwagę podczas podziału węzłów drzew decyzyjnych.

Tabela 1: Tabela zawierająca użyte zakresy hiperparametrów.

Algorytm uczenia maszynowego	**Hyperparametr**	**dolny zakres**	**górny zakres**
Random Forest Classifier	n_estimators	10	230
Random Forest Classifier	max_depth	1	15
Random Forest Classifier	min_samples_split	1	15
Random Forest Classifier	min_samples_leaf	1	10
Random Forest Classifier	max_features	['sqrt', 'log2']	
Gradient Boosting Classifier	n_estimators	10	150
Gradient Boosting Classifier	max_depth	1	8
Gradient Boosting Classifier	min_samples_split	1	10
Gradient Boosting Classifier	min_samples_leaf	1	5
Gradient Boosting Classifier	learning_rate	0.05	0.5
Gradient Boosting Classifier	subsample	0.75	1.0
SVM Classifier	C	0.01	10.0
SVM Classifier	kernel	['linear', 'rbf', 'sigmoid']	
SVM Classifier	gamma	0.001	6.5

4 Metody Samplingu

Zdecydowano się wybrać następujące metody samplingu:

- Random Search - jako przykład metody opierającej się na wyborze punktów z rozkładu jednostajnego,
- Bayes Optimization - jako przykład metody opierającej się na technice bayesowskiej.

Użyto pakietu **RandomizedSearchCV** z biblioteki **sklearn**, ze względu na możliwość wykorzystania wszystkich wątków CPU do przeprowadzania obliczeń. Pakiet **RandomizedSearchCV** pozwala na użycie cross-walidacji do oceny poprawności modelu, bazując na dostarczonym zbiorze treningowym. Dzięki dostarczeniu stałego **random_state** uzyskano stałą siatkę parametrów dla każdego rozpatrywanego zbioru danych.

Użyto pakietu **HyperParameterOptimizationFacade** z biblioteki **smac** jako implementację metody bayesowskiej. Zdecydowano się na testowanie tylko tej jednej Facade'y, ze względu na powszechne opinie uważające ją za najlepszą implementacje metody bayesowskiej z biblioteki **smac**. Użycie tego pakietu pozwala na samodzielne zdefiniowanie **target_function** będącą oceną poprawności szukanych hiperparametrów. W tej funkcji zdefiniowano analogiczną logikę oceny używaną przez pakiet **RandomizedSearchCV** w celu zwiększenia wiarygodności przeprowadzonego doświadczenia i umożliwienie analizy porównawczej obu metod samplingu.

W obu pakietach użyto **roc_auc** jako funkcji **scoring**, a ponadto współczynnikowi cross-walidacji nadano wartość 5. W tabeli 2 zaprezentowano informację o przyjętych wartościach iteracji dla poszczególnych metod oraz czasu ich wykonania. Bayes Optimization nie przyjmuje initial point.

Tabela 2: Tabela zawierająca informację o parametrach wywołania na CPU Intel i5-10300H 2.5GHz napisanego kodu niezbędnego do określenia optymalnych hiperparametrów.

Algorytm uczenia maszynowego	**Ilość iteracji dla Random Search**	**Ilość iteracji dla Bayesian Optimization**	**Czas obliczeń dla Random Search**	**Czas obliczeń dla Bayesian Optimization**
Random Forest Classifier	450	150	57min	102min
Gradient Boosting Classifier	300	150	67min	110min
SVM Classifier	300	150	87min	149min

5 Analiza wyników

Poniżej przedstawiono wyniki analizy uzyskanych wyników optymalizacji hiperparametrów. W komentarzach w pliku **Analysis.ipynb**, zawarto więcej szczegółów interpretacji wyników. W tabelach 3, 4 oraz 5 zaprezentowano znalezione hiperparametry. Średnie wyniki poszczególnych modeli na znalezionych średnio najlepszych hiperparametach sprawdzonych na zbiorze testowym można zobaczyć w tabeli 6. Zauważono średnią poprawę modelów przy użyciu znalezionych hiperparametrów. Większa poprawa następuje dla metody samplingu Bayes Optimization, co nasuwa wniosek, że jest ona lepsza.

Tabela 3: Tabela zawierająca znalezione hiperparametry dla Random Forest Classifier.

Metoda samplingu	**n_estimators**	**max_depth**	**min_samples_split**
Random Search	153	11	2
Bayes Optimization	227	12	4
Metoda samplingu	**min_samples_leaf**	**max_futures**	
Random Search	4	log2	
Bayes Optimization	3	log2	

Tabela 4: Tabela zawierająca znalezione hiperparametry dla Gradient Boosting Classifier.

Metoda samplingu	**n_estimators**	**max_depth**	**min_samples_split**
Random Search	116	4	6
Bayes Optimization	141	5	5
Metoda samplingu	**min_samples_leaf**	**learning_rate**	**subsample**
Random Search	3	0.105102	0.775510
Bayes Optimization	5	0.051132	0.751077

Tabela 5: Tabela zawierająca znalezione hiperparametry dla SVM Classifier.

Metoda samplingu	**C**	**kernel**	**gamma**
Random Search	9.388367	rbf	3.316816
Bayes Optimization	9.08872	rbf	3.708863

5.1 Stabilność wyników

Na wykresach 1 oraz 2 zawarto informację o stabilnościach metod sampligu.

Podczas przeprowadzania eksperymentu ustalono większą liczbę iteracji dla Random Search, by zaprezentować, że liczba iteracji nie ma bezpośredniego wpływu na moment znalezienia najlepszego punktu w hiperprzestrzeni. Dla wykresu **rf_rd_accumulate** widać, że prawie najlepszy punkt znaleziono już na samym

Tabela 6: Tabela zawierająca średnie wyniki poszczególnych modeli na znalezionych hiperparametach sprawdzonych na zbiorze testowym.

Algorytm	Random Search	Bayes Optimization	Default
Random Forest	0.8912434977360744	0.8921311162696635	0.8869180357772917
Gradient Boosting	0.8981076691283966	0.8986140892075709	0.8956905999452571
SVM	0.8667340916601738	0.8665752806450856	0.8599154568972678

początku, a dla wykresu **gb_rd_accumulate** dopiero około 200 iteracji. Można natomiast zauważyć, że dla metody samplingu Bayes Optimization wyniki uzyskują pewną stabilność. Np. na wykresie **sv_bo** widać dokładnie granicę stabilności. Z wykresów łatwo odczytać od jakich wartości Bayes Optimization uzyskuje stabilność. Dla Random Search takiej stabilności nigdy nie osiągniemy ze względu losowy wybór punktów z rozkładu jednostajnego.

5.2 Tunowalność algorytmów

Tunowalność algorytmu została policzona wzorując się na artykule [8]. Użyto dwóch następujących równań do każdej metody samplingu i do każdego modelu uczenia maszynowego:

$$d^{(i)} := E^{(i)}(\theta^*) - E^{(i)}(\theta^{(i)*}), \quad \text{dla każdego } i = 1, \ldots, m. \tag{1}$$

$$d^{(i)}_{(j)} := E^{(i)}(\theta^*) - E^{(i)}(\theta^{(i)}_{(j)}), \quad \text{dla każdego } i = 1, \ldots, m \quad \text{dla każdego } j = 1, \ldots, n, \tag{2}$$

gdzie $E^{(i)}$ to funkcja, która oblicza poprawność modelu zgodnie ze skalą **roc_auc** dla $i - tego$ zbioru danych, θ^* - średnio najlepszy zestaw hiperparametrów dla wszystkich zbiorów danych, $\theta^{(i)*}$ - najlepszy zestaw hiperparametrów dla $i - tego$ zbioru danych, $\theta^{(i)}_{(j)}$ - to zestaw hiperparametrów dla $i - tego$ zbioru danych dla $j - tej$ iteracji.

Wartości z równania 2 zostały umieszczone na rysunku 3, aby móc lepiej zobaczyć empiryczną dystrybucje. Ponadto zieloną linią zostały zaznaczone wartości z równania 1 na rysunku 3. Ponadto czerwoną linią zaznaczono te wartości z równania 2, dla których wybrano średnio najlepszy punkt w hiperprzestrzeni.

Wynik ujemny dla równania 1 oznacza, że istnieje taki punkt w przestrzeni hiperprzestrzeni dla danego datasetu, który przy wyborze jego i założeniu, że inne wyniki dla innych datasetów nie zmienią się wpłynąłby na poprawę średnio najlepszego wyniku. Oznacza to, że jeżeli byłoby spełnione założenie to jesteśmy w stanie znaleźć jeszcze lepszy hiperparametr w przestrzeni hiperparametrów generalizujący model jeszcze bardziej. Ciekawsza wydaje nam się być suma po wszystkich i zbiorach dla równania 1, która mówi o tym jak bardzo daleko jesteśmy od optymalnego rozwiązania mając obecnie dostępną wiedzę o najlepszych hiperparametrach dla danego datasetu przy założeniu, że jesteśmy znaleźć taki hiperparametr, który będzie średnia tych najlepszych wyników. Aby zobaczyć średnią możliwą poprawę dla danego datasetu powinno się podzielić tą sumę przez liczbę datasetów (więcej o tym w **Analisis.ipynb**). Na podstawie rysunku 3 można wyciągnąć następujące wnioski:
1. Dla metody Random Search zawsze występuje więcej próbek odstających niż dla Bayes Optimization.
2. Boxy są mniejsze dla Bayes Optimization, co ma związek ze zbieżnością algorytmu.
3. Dla Bayes Optimization czerwona kreska prawie zawsze pokrywa się z pomarańczową co wskazuję na stabilność i ma związek ze znalezionym ekstremum lokalnym.
4. Oczekiwanym rezultatem jest zbliżanie się kreski czerwonej do kreski zielonej.
5. Wszystkie wartości dodatnie można interpretować o ile model został poprawiony w skali **roc_auc** podczas sampligu.

Można również zobaczyć, że wyniki dla RandomForestClassifier oraz dla GradientBoostingClassifier są dość podobne niezależnie od metody samplingu, a wyniki dla SVM Classifier bardzo się różnią w stosunku do Bayes Optimization. Jest to związane z tym, że jest to algorytm, który jest najbardziej zależny od danych wejściowych, co jest pewnego rodzaju przesłanką, że znaczenie optymalizacji hiperparametrów dla SVM jest

bardzo duże, co wiąże się z największym rezultatem średniej poprawy algorytmu dla danych testowych niezależnie od metody samplingu. W ogólności można stwierdzić, że najlepiej tunowalnym algorytmem jest SVM, a najgorzej tunowalnym Gradient Boosting, ale jest dokładnie odwrotnie jeżeli chodzi o średnią skuteczność modelu na zbiorze testowym, więc można stwierdzić, że im większa skuteczność modelu podstawowego tym mniejszy wpływ ma optymalizacja hiperparametrów.

W tabeli 7 widoczna jest policzona uśredniona po wszystkich datasetach średnia tunowalność dla każdego zbioru danych (z uwzględnieniem metody samplingu). Zrobiono to dla wartości widocznych na rysunku 3. Szerszą analizę i interpretacje wyników tunowalności można znaleźć w pliku **Analysis.ipynb**.

Tabela 7: Tabela zawierająca policzoną uśrednioną po wszystkich datasetach średnią tunowalność dla każdego zbioru danych z uwzględnieniem metody samplingu.

Algorytm	**Random Search**	**Bayes Optimization**
Random Forest	0.012505219974044877	0.009512046580359325
Gradient Boosting	0.012457472044636137	0.007652257342948278
SVM	0.10194249009495136	0.032331579039159924

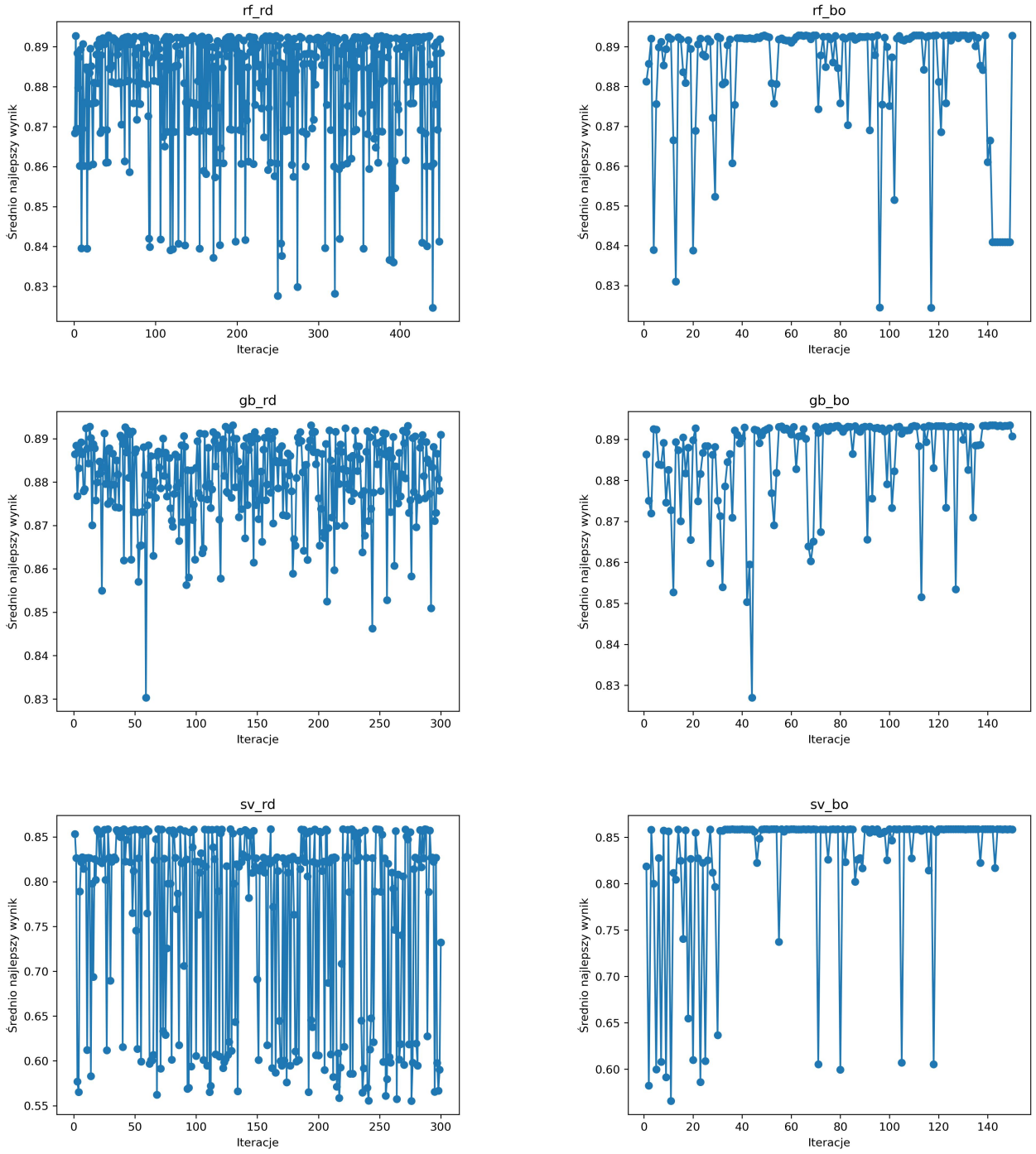
5.3 Czy występuję Bias Sampling?

Technika losowania punktów wpływa na różnice we wnioskach w rozdziale 5.2.

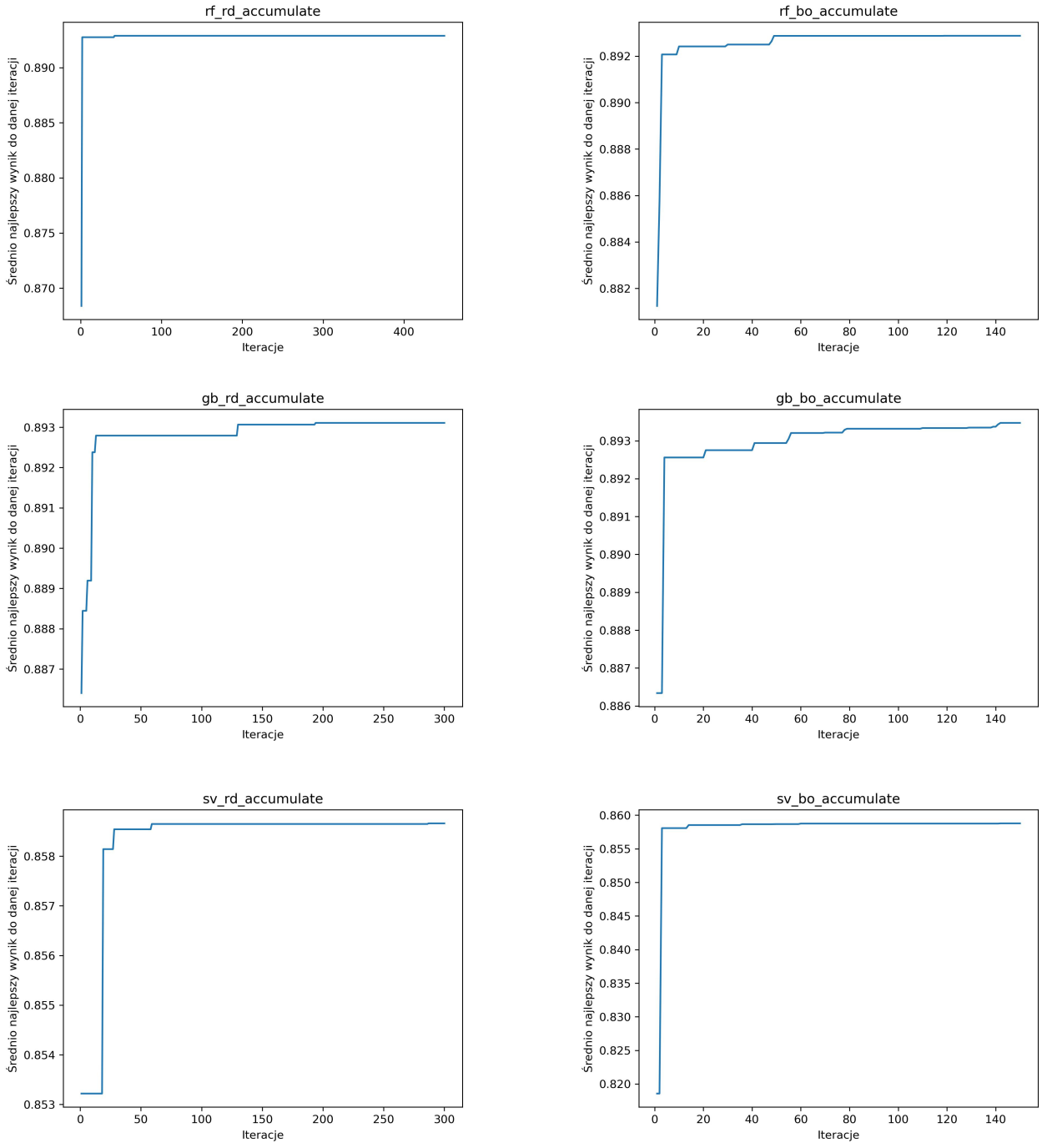
Dla Random Search może istnieć pewne obciążenie, ponieważ algorytm może przegapić obszary przestrzeni hiperparametrów, które są obiecujące, ale wymagają pewnej kontynuacji lub bardziej systematycznego badania. Widoczne jest to na wykresach 3, gdzie bardzo duża cześć obserwacji odbiega od najlepszych wyników dla danego zbioru danych i dla danego algorytmu uczenia maszynowego. Jest to widoczne w tabeli 7 przy uśrednieniu wyników tunowalności dla każdego zbioru danych dla danej metody samplingu i dla danego algorytmu uczenia maszynowego, gdzie średnia tunowalność jest zazwyczaj większa, ale nie przenosi się to na bezpośrednio większą poprawę modelu, co wskazuje na mniejszą reprezentatywność próbek.

Dla Bayes Optimization **bias sampling** może występować gdy źle zdefiniuje się **target_function** lub gdy model jest źle zdefiniowany lub gdy dane są za małe lub nieoczyszczone. O wszystkie te rzeczy zadbano w trakcie realizowania zadania, co ma potwierdzenie na wykresie 3, gdzie Bayes Optimization faworyzuje prawidłowe obszary przestrzeni hiperparametrów (wniosek 3.), co wpływa na generalizacje modelów. Ponadto wcześniej wpominana średnia tunowalność w tabeli 7 pokazuje wyniki będące bliżej prawdy, biorąc pod uwagę tabelę 6.

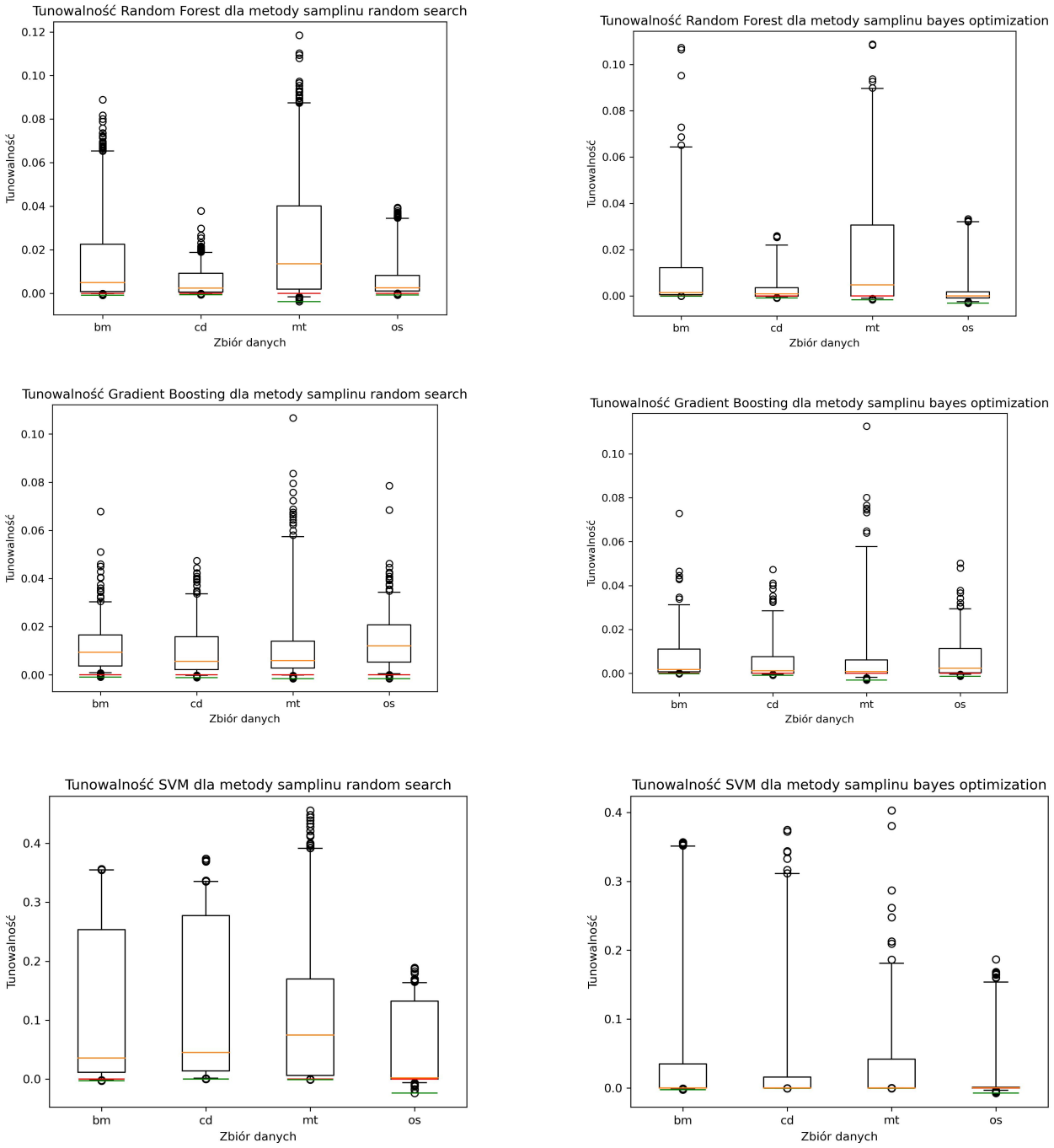
6 Dodatek



Rysunek 1: Wykresy poszczególnych wyników średnich zależnych od iteracji dla danego modelu oraz dla danej metody samplingu



Rysunek 2: Wykresy najlepszych do tej pory znalezionych wyników średnich zależnych od iteracji dla danego modelu oraz dla danej metody samplingu



Rysunek 3: Wykresy tunowalności dla danego modelu oraz dla danej metody samplingu dla ustalonych zbiorów danych. Zieloną linią zaznaczone są wartości tunowalności dla najlepszego punktu w hiperprzestrzeni dla danego zbioru danych. Czerwoną linią zaznaczone są wartości tunowalności dla średnio najlepszego punktu w hiperprzestrzeni biorąc pod uwagę wszystkie zbiory danych. Pomarańczową linią zaznaczona jest mediana z wartości. Box: 25 - 75, Whis: 5 - 95.

Literatura

[1] Pierwszy zbiór danych

[2] Drugi zbiór danych

[3] Trzeci zbiór danych

[4] Czwarty zbiór danych

[5] Dokumentacja sklearn Random Forest Classifier

[6] Dokumentacja sklearn Gradient Boosting Classifier

[7] Dokumentacja sklearn SVM Classifier

[8] Tunability: Importance of Hyperparameters of Machine Learning Algorithms

[9] Random Forest Classifier - wstęp do optymalizacja hiperparametrów

[10] Gradient Boosting Classifier- wstęp do optymalizacja hiperparametrów

[11] SVM Classifier- wstęp do optymalizacja hiperparametrów