Wydział Matematyki
i Nauk Informacyjnych

POLITECHNIKA WARSZAWSKA

AutoML, Praca Domowa 1

Raport

Adam Majczyk 313420, Szymon Matuszewski 313435

Wersja 1.0

22.11.2023

Spis treści

1 Cel

Celem eksperymentu jest zbadanie tunowalności algorytmów uczenia maszynowego wzorując się na pracy naukowej ***Tunability: Importance of Hyperparameters of Machine Learning Algorithms***[1].

2 Zbiory danych

Wszystkie wykorzystane zbiory pochodzą z portalu OpenML. Wszystkie zbiory danych są zbiorami klasyfikacji binarnej.

2.1 Zbiór 1: credit-g_reproduced_1 (ID: 44098)

2.1.1 Opis zbioru

Zbiór posiada 21 kolumn (w tym 1 to kolumna celu - klasa binarna **good/bad** oznaczająca czy aplikant okazał się dobrym lub złym kredytobiorcą). 7 kolumn jest numerycznych, pozostałe nominalne. Zbiór posiada 1000 obserwacji z czego 700 zakwalifikowano jako dobrych aplikantów, 300 jako złych. Zbiór nie posiada brakujących wartości.

2.1.2 Przygotowanie danych

Klasę **bad** zakodowano jako 1, klasę **good** jako 0. Zmienne nominalne poddano procesowi **One-Hot Encoding**.

2.1.3 Selekcja zmiennych

Dla przygotowanego zbioru danych policzono korelację (Pearsona) zmiennych objaśniających z objaśnianą. Spośród 20 zmiennych o najwyższej co do modułu korelacji wylosowano 7. Finalne zmienne na których trenowano model to:

checking_status_no_checking, *purpose_new_car*, *employment_1*, *checking_status_0*, *credit_history_all_paid*, *credit_amount*, *checking_status_0_X_200*

Wszystkie z tych zmiennych oprócz zmiennej *credit_amount* to zmienne binarne (*credit_amount* jest ciągła - numeryczna).

2.1.4 Podział na zbiory treningowe i testowe

Dokonano podziału na zbiór treningowy i testowy. Oba zbiory posiadają 500 obserwacji. Uwzględniono próbkowanie warstwowe po kolumnie celu - aby zapobiec niezrównoważeniu klasy celu.

2.2 Zbiór 2: bank-marketing dataset (ID: 1461)

2.2.1 Opis zbioru

Zbiór posiada 17 kolumn (w tym 1 to kolumna celu - klasa binarna **yes/no** oznaczająca czy klient zasubskrybował depozyt okresowy). 7 kolumn jest numerycznych, pozostałe nominalne. Zbiór posiada 1000 obserwacji z czego 39922 zakwalifikowano jako klasę 1, 5289 jako klasę 2 (w zaimportowanych danych kolumna celu nie była nominalna, lecz posiadała wartości 1 oraz 2 - nie odnaleziono interpretacji która oznacza klasę **yes** i klasę **no**; można podejrzewać, że mniejsza z klas to klasa **no**). Zbiór nie posiada brakujących wartości.

2.2.2 Przygotowanie danych

Kolumnę celu z wartości 1 oraz 2 zamieniono na 0 oraz 1 (odjęto 1). Kolumny nominalne poddano procesowi **One-Hot Encoding**.

2.2.3 Selekcja zmiennych

Dla przygotowanego zbioru danych policzono korelację (Pearsona) zmiennych objaśniających z objaśnianą. Spośród 20 zmiennych o najwyższej co do modułu korelacji wylosowano 7. Finalne zmienne na których trenowano model to:

month_dec, *month_mar*, *loan_no*, *month_may*, *contact_cellular*, *campaign*, *housing_yes*.

Wszystkie z tych zmiennych to zmienne binarne.

2.2.4 Podział na zbiory treningowe i testowe

Dokonano podziału na zbiór treningowy i testowy. Zbiór treningowy posiada 500 obserwacji, a testowy 1000 obserwacji. Uwzględniono próbkowanie warstwowe po kolumnie celu - aby zapobiec niezrównoważeniu klasy celu.

2.3 Zbiór 3: MagicTelescope (ID: 1120)

2.3.1 Opis zbioru

Zbiór posiada 11 kolumn (w tym 1 to kolumna celu - klasa binarna **g/h** oznaczająca czy obserwacja to cząstka typu **gamma** albo typu **hadron**). 10 kolumn jest numerycznych, pozostała (celu) nominalna. Zbiór posiada 19020 obserwacji z czego 12332 zakwalifikowano jako klasę **gamma**, 6688 jako klasę **hadron**. Zbiór nie posiada brakujących wartości.

2.3.2 Przygotowanie danych

Zbiór nie wymagał obszernego przygotowania danych. Klasę **bad** zakodowano jako 1, klasę **good** jako 0. Zmienne nominalne poddano procesowi **One-Hot Encoding**.

2.3.3 Selekcja zmiennych

Dla przygotowanego zbioru danych policzono korelację (Pearsona) zmiennych objaśniających z objaśnianą. Spośród 20 zmiennych o najwyższej co do modułu korelacji wylosowano 7. Finalne zmienne na których trenowano model to:

fAsym, *fDist*, *fM3Trans*, *fM3Long*, *fAlpha*, *fLength*, *fWidth*

Wszystkie z tych zmiennych to zmienne ciągłe (numeryczne).

2.3.4 Podział na zbiory treningowe i testowe

Dokonano podziału na zbiór treningowy i testowy. Zbiór treningowy posiada 500 obserwacji, a testowy 1000 obserwacji. Uwzględniono próbkowanie warstwowe po kolumnie celu - aby zapobiec niezrównoważeniu klasy celu.

2.4 Zbiór 4: Titanic (40945)

2.4.1 Opis zbioru

Zbiór posiada 14 kolumn (w tym 1 to kolumna celu - klasa binarna **0/1** oznaczająca czy pasażer przeżył katastrofę). 6 kolumn jest numerycznych, 3 nominalne, 5 tekstowych. Zbiór posiada 1309 obserwacji z czego 809 jest klasy **0** (nie przeżyło) klasę, 500 jako klasy **1** (przeżyło). Zbiór posiada brakujące wartości.

2.4.2 Przygotowanie danych

Odrzucono kolumny tekstowe - mianowicie: *cabin*, *boat*, body, *home.dest*, *name*, *ticket*. Następnie usunięto wiersze z brakami danych. Zmienne nominalne poddano procesowi **One-Hot Encoding**.

2.4.3 Selekcja zmiennych

Dla przygotowanego zbioru danych policzono korelację (Pearsona) zmiennych objaśniających z objaśnianą. Spośród 20 zmiennych o najwyższej co do modułu korelacji wylosowano 7. Finalne zmienne na których trenowano model to: *embarked_Q*, *sex_male*, *embarked_C*, *parch*, *age*, *sex_female*, *fare*.

embarked_Q, *sex_male*, *embarked_C*, *sex_female* to zmienne binarne, pozostałe są numeryczne.

2.4.4 Podział na zbiory treningowe i testowe

Dokonano podziału na zbiór treningowy i testowy. Zbiór treningowy posiada 500 obserwacji, a testowy 809 obserwacji (wszystkie pozostałe obserwacje). Uwzględniono próbkowanie warstwowe po kolumnie celu - aby zapobiec niezrównoważeniu klasy celu.

3 Wybór parametrów default

Modele predykcyjne oraz rozważane różne kombinacje hiperparametrów z zakresami:

- LBGM Classifier - 'n_estimators' - int 10-1000, 'max_depth' - int 2-10, 'learning_rate' - float 0.01-1.0, 'subsample' - float 0.5-1.0, 'min_child_weight' - int 1-10, 'num_leaves' - int 2-50, 'reg_alpha' - float 0.01-10.0, 'reg_lambda' - float 0.01-10.0
- XGBoost Classifier - 'n_estimators' - int 10-1000, 'max_depth' - int 2-10, 'learning_rate' - float 0.01-1.0, 'subsample' - float 0.5-1.0, 'min_child_weight' - int 1-10, 'max_leaves' - int 2-50, 'gamma' - float 0.01-10.0, 'reg_alpha' - float 0.01-10.0, 'reg_lambda' - float 0.01-10.0
- Random Forest Classifier - 'n_estimators' - int 10-1000, 'max_depth' - int 2-10, 'min_samples_split' - int 2-10,'min_samples_leaf' - int 1-10, 'max_features' - float 0.1-1.0

Dokonano wyboru parametrów uznanych za najlepsze domyślne parametry dla każdego algorytmu przy pomocy algorytmu poszukiwań *Random Search*. Schemat poszukiwań przedstawia poniższy pseudokod:

```
for model_type in ['lgbm', 'rfc', 'xgb']:
    for i in range(200):
        hyperparameters = random_search(model_type)
        for dataset in datasets:
            train model
            calculate auc, accuracy, brier
            add to results
calculate best defaults for each model_type based on chosen metric: here AUC
```

Wyznaczenie najlepszych defaultów per model oznaczało dla każdej z 200 iteracji uśrednienie (średnia arytm.) AUC dla 4 zbiorów danych. Potem z tych 200 średnich wybrano hiperparametry o najwyżej średniej.

W ten sposób zostały wyznaczone najlepsze defaultowe hiperparametry per algorytm ML. Zapisano wartości metryk dla tych hiperparametrów z uwzględnieniem zbioru danych.

4 Optymalizacja Bayesowska

Kolejnym krokiem było wykorzystanie optymalizacji bayesowskiej do wyznaczania hiperparametrów pozwalających zbadanie tunowalności danego algorytmu. Siatka poszukiwań hiperparametrów była identyczna jak w 3. Poniższy pseudokod przedstawia logikę wyznaczania tych hiperparametrów:

```
for model_type in ['rfc', 'xgb', 'lgbm']:
    for dataset in datasets:
        hyperparameter_space, best_defaults_ = get_hyper_space_and_defaults
        for seed in range(50):
            model, bayes_hyperparameters = train_hyperparam_bayes
            calculate auc, accuracy, brier
            add to results
```

5 Wyznaczenie tunowalności modeli

Pierwszym krokiem w wyznaczeniu tunowalności modeli było pogrupowanie wyników optymalizacji bayesowskiej po modelu, metryce oraz zbiorze danych wraz z agregacją średniej, odchylenia standardowego, minimum, maksimum oraz zliczenia.

Następnie dla każdej iteracji (od 1 do 50) policzono różnicę między daną metryką przy maksimum (minimum dla Brier) dla najlepszych hiperparametrów uzyskanych dla danego algorytmu i zbioru danych za pomocą optymalizacji bayesowskiej oraz defaultowych hiperparametrach dla tego samego modelu i zbioru danych ($bayes_i - default_i$; dla **Brier** odwrócno znaki, ze względu na fakt, iż niższa wartość tej metryki oznacza lepszy model).

W ten sposób otrzymano słownik odpowiednich różnic. Jest on następującej postaci (tunability_stats.json):

```
{"model_type1":{"metric1":[difference1, difference2, difference3, difference4], "metric2":...},
"model_type2":...}
```

gdzie:

- $difference\_i$ - tunowalność modelu dla i-tego zbioru danych względem danej metryki.

Tak skompletowane wyniki pozwoliły na przedstawienie wyników w postaci wykresów pudełkowych.

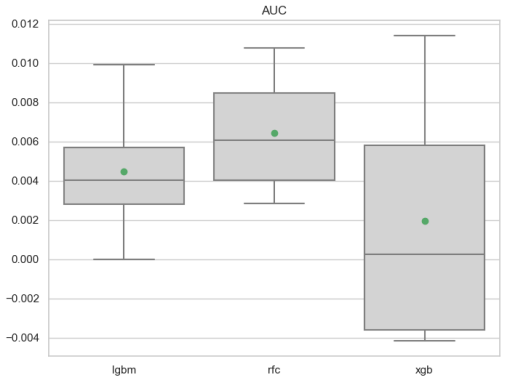
6 Wyniki

Jako ostateczną tunowalność hiperparametrów modelu uznano **średnią AUC** z różnic pomiędzy wynikami przy defaultowych hiperparametrach a najlepszymi hiperparemetrami z iteracji przy optymalizacji bayesowskiej (grupując względem algorytmu ML, metryki i zbioru danych, czyli patrząc na strukturę słownika w Sekcji 5, dla wybranego modelu i metryki, średnią z $difference\_i$).

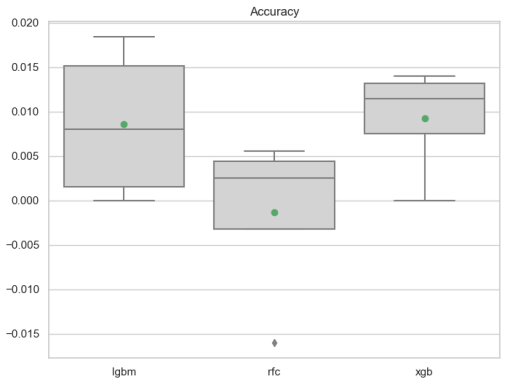
Ostatecznie tunowalność (tutaj AUC) dla poszczególnych typów modeli wynosi (uśredniając dla algorytmu):

- 'lgbm': 0.00448
- 'rfc': 0.00643
- 'xgb': 0.00193

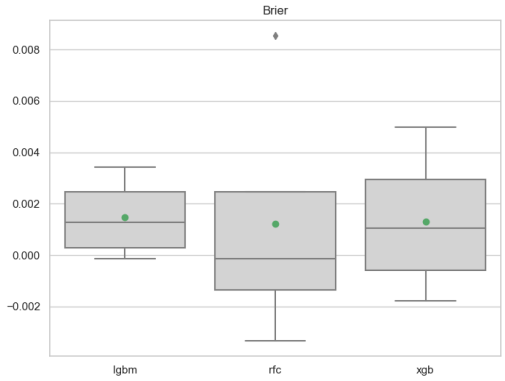
Poniżej, tj. na Rys. 1, 2, 3 przedstawiono wszystkie rozkłady ww. różnic dla różnych metryk. Zieloną kropką oznaczoną średnią.



Rysunek 1: Rozkład tunowalności dla metryki AUC.



Rysunek 2: Rozkład tunowalności dla metryki Accuracy.



Rysunek 3: Rozkład tunowalności dla metryki Brier.

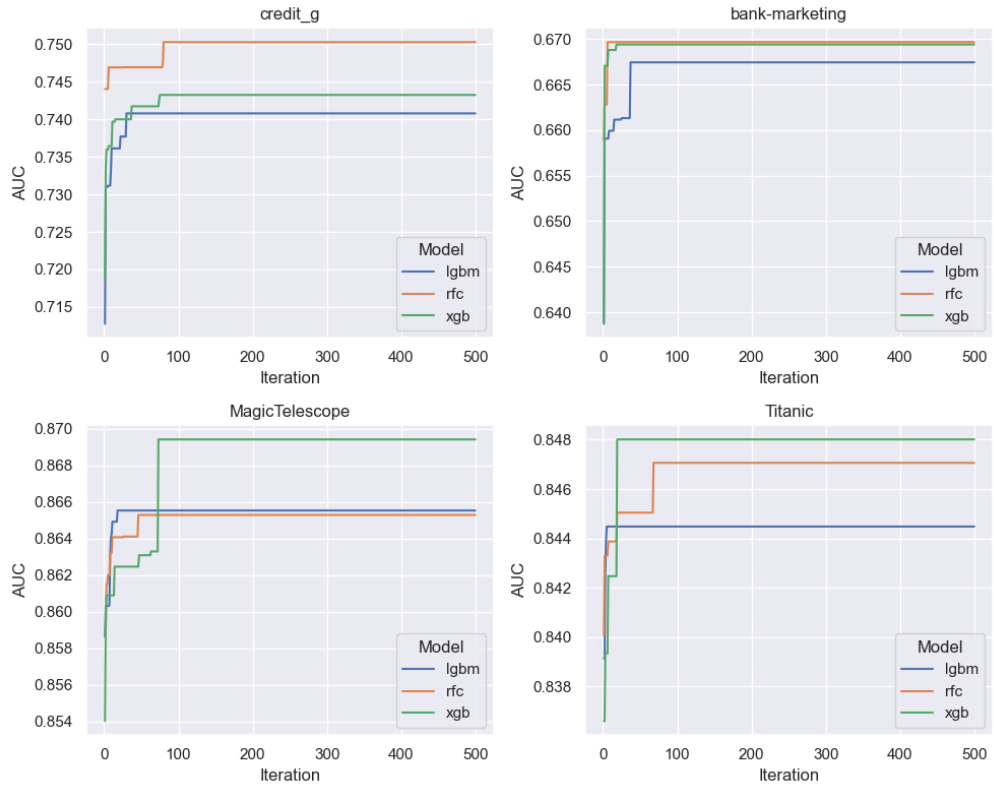
7 Bias Sampling

Ze względu na stosowane techniki, tj. podział na train i test funkcją pochodzącą z pakietu **sklearn** oraz stosowaniu próbkowania warstwowego, można założyć, że w naszym eksperymencie nie występuje Bias Sampling.

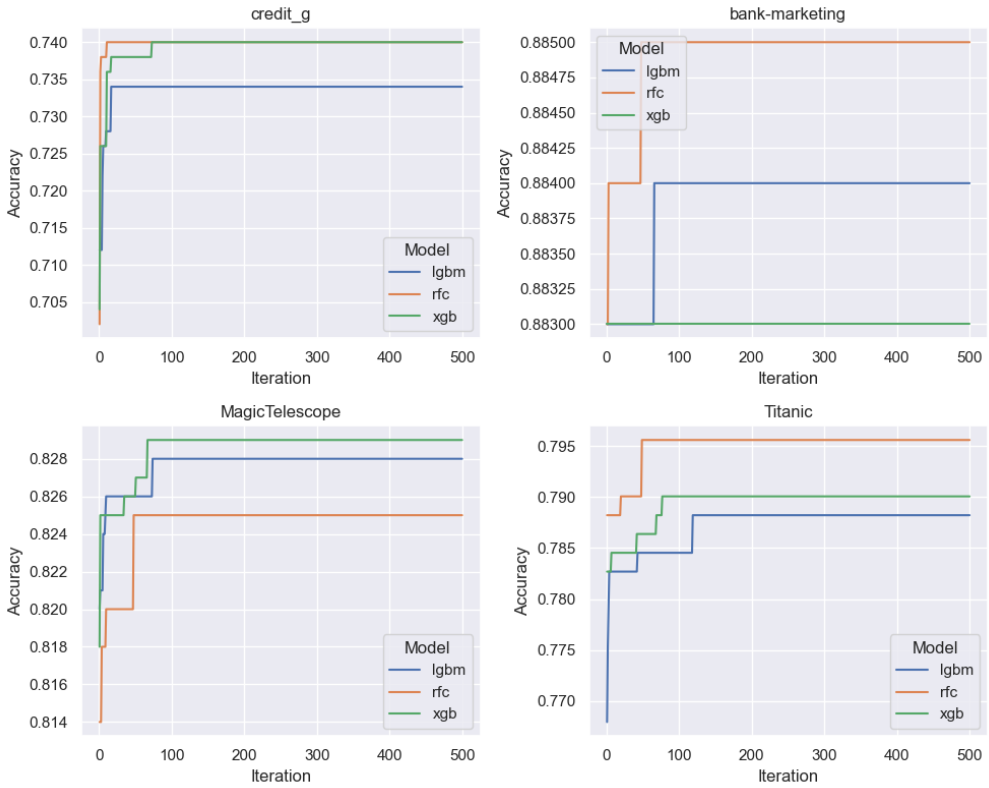
8 Zbieżność algorytmów optymalizacji

8.1 Random Search

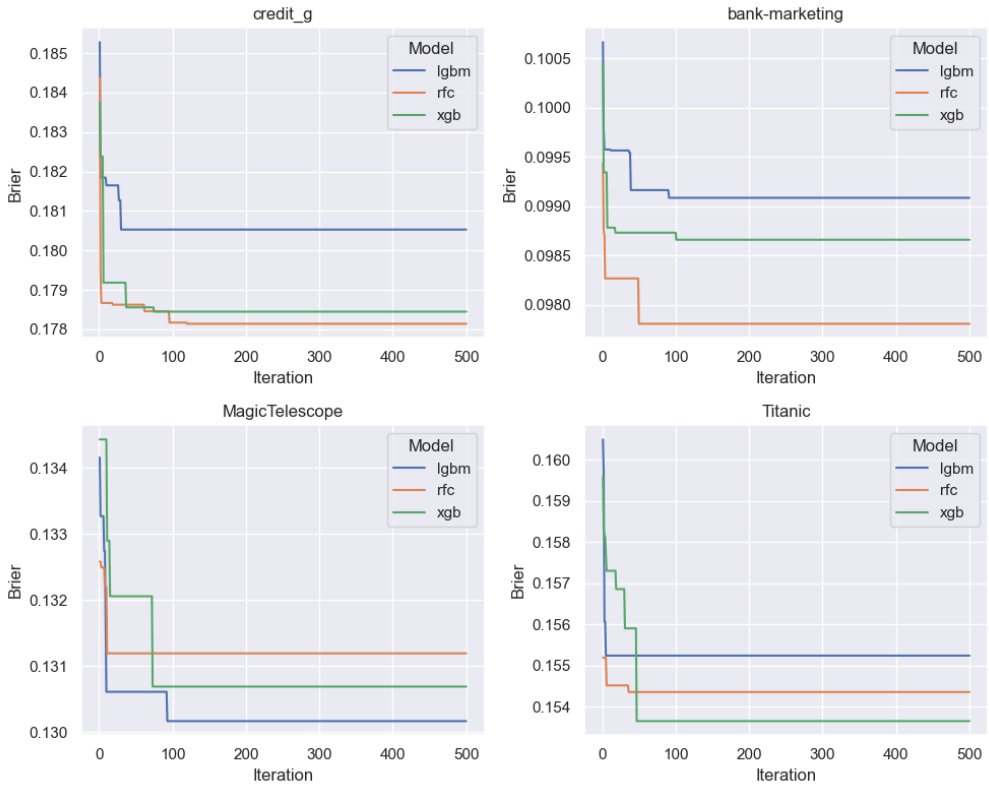
W celu zbadania zbieżności zapisywano przez 500 iteracji wartości metryk z podziałem na zbiór danych oraz stosowany algorytm uczenia maszynowego. Zapisywano dotychczas najlepszą wartość metryki. Wyniki przedstawiono na Rys. 4, 5, 6.



Rysunek 4: Zbieżność RS dla AUC



Rysunek 5: Zbieżność RS dla Accuracy

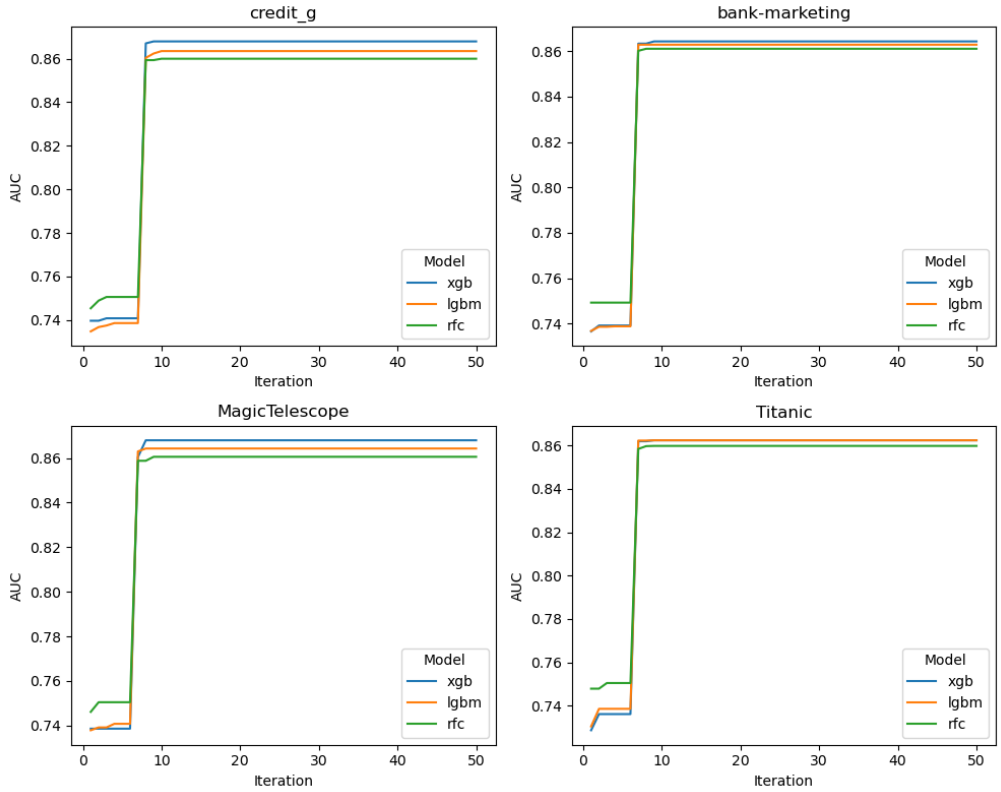


Rysunek 6: Zbieżność RS dla Brier

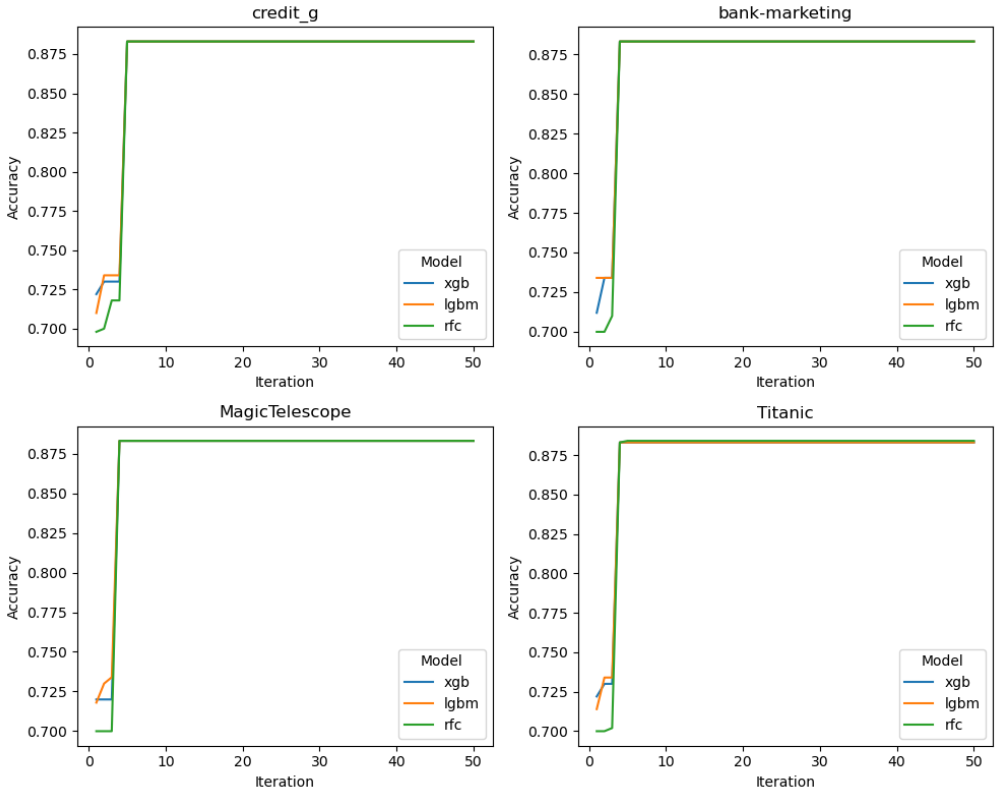
Jak widać, po około 150 (a czasami poniżej 100) iteracjach nie odnaleziono w żadnym z przypadków lepszych hiperparametrów.

8.2 BayesSearchCV

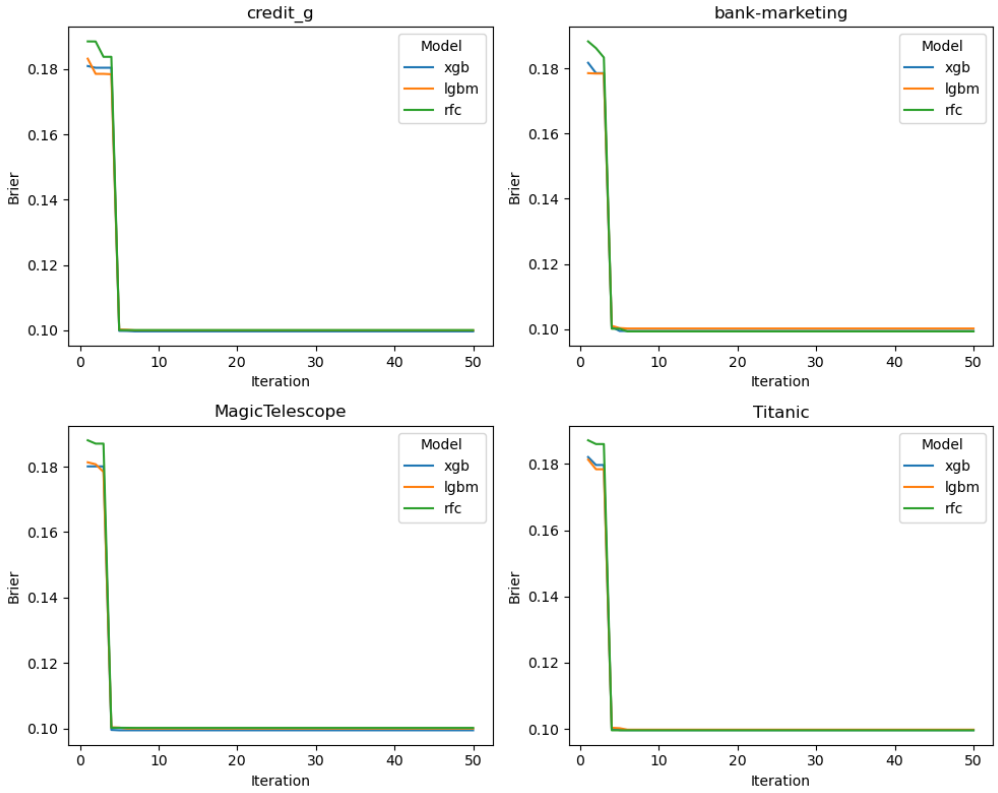
Przeprowadzono analogiczny eksperyment, jedynie ze względu na długi czas potrzebny do wyznaczenia hiperparametrów metodą Bayesowską, wykonano 50 iteracji. Wyniki przedstawiono na Rys. 7, 8, 9.



Rysunek 7: Zbieżność BS dla AUC



Rysunek 8: Zbieżność BS dla Accuracy



Rysunek 9: Zbieżność BS dla Brier

Jak widać, po około 10 (a czasami nawet 5) iteracjach nie odnaleziono w żadnym z przypadków lepszych hiperparametrów.

9 Wnioski

- Najbardziej podatnym na zmianę hiperparametrów algorytmem spośród trzech analizowanych jest Random Forest.
- Biorąc pod uwagę metrykę **Accuracy** przy podawaniu końcowej tunowalności to **XGBoost** byłby najbardziej podatny na zmianę hiperparametrów. Dla **Brier** algorytmy wypadły prawie identycznie (różnica dopiero na 4 miejscu po przecinku), lecz faworytem byłby **LightBGM**.
- Problem tunowalności hiperparametrów wydaje się być problemem zależnym od wyboru końcowej metryki porównawczej.
- Czasami dla konkretnej metryki, algorytmu oraz datasetu otrzyma się 'dostatecznie dobre' wyniki bez tunowania hiperparametrów.
- Tunowalność może być wyznaczana na wiele sposobów (u nas ustaliliśmy, że będzie to średnia odpowiednich różnic względem metryki **AUC**).

10 Bibliografia

[1] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: Importance of Hyperparameters of Machine Learning Algorithms. 2019.