Tunowalność Hiperparametrów w wybranych algorytmach uczenia maszynowego

Katsiaryna Bokhan, Monika Jarosińska

15 listopada 2024

1 Wprowadzenie

Celem projektu jest analiza tunowalności hiperparametrów dla trzech wybranych algorytmów uczenia maszynowego, w projekcie wykorzystałyśmy algorytmy **SupportVectorMachine**, **CatBoostClassifier** oraz **ExtraTreesClassifier**. Tunowalność wymienionych algorytmów została sprawdzona przy użyciu czterech zbiorów danych pochodzących ze strony OpenML i przeznaczonych do klasyfikacji binarnej. Dodatkowo analizowałyśmy wyniki modeli w poszczególnych iteracjach oraz porównywałyśmy te wyniki z wynikami modeli o domyślnych hiperparametrach.

2 Dane

2.1 Zbiory danych

1. **Zbiór danych 1 (ID = 833)** - zbiór danych został syntetycznie wygenerowany w celu modelowania wyboru banków przez klientów oraz procesu obsługi w bankach. Celem zadania było przewidywanie odsetka klientów, którzy opuszczają bank z powodu pełnych kolejek. Zbiór danych pochodzi z symulacji, w której klienci wybierają banki na podstawie odległości, poziomu cierpliwości oraz innych czynników. Wartość docelowa przyjmuje wartość 'P', gdy klient został obsłużony i 'N', gdy nie został obsłużony.

2. **Zbiór danych 2 (ID=44157)** - zbiór danych dotyczy zadania klasyfikacji opartego na ruchach oczu i jest wykorzystywany w badaniach do przewidywania trafności tekstu. Dane zawierają informacje o ruchach oczu uczestników czytających zdania, aby określić, na ile słowa są istotne dla odpowiedzi na zadane pytanie. Każdy wiersz reprezentuje pojedyncze słowo i zawiera 22 cechy, mierzące różne charakterystyki ruchów oczu, takie jak liczba fiksacji, czas trwania fiksacji, pozycja wzroku. Wartość docelowa przyjmuje wartość 1, gdy słowo jest istotne i 0 gdy nie jest istotne.

3. **Zbiór danych 3 (ID=1120)** - zbiór danych z projektu MAGIC (Major Atmospheric Gamma Imaging Cherenkov Telescope) służy do klasyfikacji wysokoenergetycznych

cząstek gamma i cząstek tła pochodzących z promieni kosmicznych na podstawie obrazów promieniowania Czerenkowa. Każdy rekord reprezentuje zdarzenie opisane parametrami, takimi jak długość i szerokość elipsy, asymetria i kąt nachylenia, które pozwalają odróżnić "sygnał" (gamma) od "tła" (hadron).

4. **Zbiór danych 4 (ID = 45553)** - zbiór danych dotyczy oceny zdolności kredytowej pożyczkobiorców na podstawie ich profilu finansowego. Zestaw danych pochodzi z "Explainable Machine Learning Challenge", zorganizowanego przez FICO i obejmuje cechy, które mogą pomóc w przewidywaniu ryzyka kredytowego. Wartość docelowa określa, czy klient jest niskiego ('Good') czy wysokiego ryzyka ('Bad')

Zbiór danych (ID)	**Liczba rekordów**	**Rozkład zmiennej objaśnianej**	**Liczba kolumn**
Zbiór danych 1 (833)	8192	31% vs 69%	32
Zbiór danych 2 (44157)	7608	50% vs 50%	23
Zbiór danych 3 (1120)	19020	35% vs 65%	10
Zbiór danych 4 (45553)	9871	48% vs 52%	23

Tabela 1: Informacje na temat zbiorów danych.

2.2 Transformacje zmiennych

Na początku przeprowadziłyśmy wstępną analizę danych w wybranych zbiorach. Żaden zbiór nie zawierał braków danych. Następnie zadecydowałyśmy o usunięciu skorelowanych kolumn we wszystkich zbiorach, z tego powodu została przez nas stworzona klasa **RemoveCorrelatedFeatures** (próg jest wybierany przez użytkownika), jednak później zdecydowano, że chcemy dokonać tylko niezbędnych zmian w danych. Więc ostatecznie, został zbudowany jeden pipeline, składający się z **OneHotEncoder** do transformacji zmiennych kategorycznych oraz **MinMaxScaler** do transformacji zmiennych numerycznych.

3 Aspekty techniczne

3.1 Wybrane modele do analizy i ustalone siatki hiperparametrów

Projekt został wykonany przy użyciu języka **Python** w wersji 3.10.7. Jako modele do analizy zostały wybrane algorytmy: **Support Vector Machine**, **ExtraTreesClassifier** oraz **CatBoostClassifier**. Informacje o rozważanych hiperparametrach oraz ich zakresach dla poszczególnych modeli są przedstawione w poniższych tabelach 2, 3, 4. Dla każdego zestawu danych przgotowaną taką samą siatkę hiperparametrów. Warto zauważyć, że przy ustaleniu siatki hiperparametrów, inspirowałyśmy się głównie artykułem [1] oraz dokumentacją wykorzystanych modeli.

Hiperparametr	Możliwe wartości	Liczba wartości
kernel	'linear', 'poly', 'rbf', 'sigmoid'	4
C	$2^{-10}, 2^{-9}, \ldots, 2^{9}$	20
gamma	$2^{-10}, 2^{-9}, \ldots, 2^{9}$	20
degree	2, 3, 4, 5	4
Liczba możliwych kombinacji		6400

Tabela 2: Siatka hiperparametrów dla **SVM**.

Hiperparametr	Możliwe wartości	Liczba wartości
iterations	2, 3, 4, 5, 6, 7, 8, 9, 11, 13, 15, 18, 21, 25, 29, 34, 40, 47, 56, 65, 77, 90, 105, 124, 145, 170, 200	27 (log space used)
learning_rate	0.01, 0.1325, 0.255, 0.3775, 0.5	5
depth	1, 3, 5, 7, 9, 11, 13, 16	8
l2_leaf_reg	1.0, 3.0, 5.0, 7.0, 9.0, 11.0, 13.0, 15.0	8
colsample_bylevel	0.1, 0.325, 0.55, 0.775, 1.0	5
Liczba możliwych kombinacji		43200

Tabela 3: Siatka hiperparametrów dla **CatBoostClassifier**.

Hiperparametr	Możliwe wartości	Liczba wartości
n_estimators	2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 19, 21, 25, 29, 33, 38, 44, 51, 58, 67, 78, 89, 103, 119, 137, 158, 182, 209, 241, 277, 319, 368, 424, 488, 562, 647, 745, 858, 988, 1137, 1310, 1508, 1737, 2000	46 (log space used)
max_depth	None, 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 30	16
min_samples_split	2, 3, 4, 5, 6, ... 58, 59, 60	59
min_samples_leaf	1, 2, 3, 4, 5, ..., 58, 59, 60	60
min_weight_fraction_leaf	20 values from 0 to 0.5 uniformly distributed	20
max_leaf_nodes	None, 2, 3, 4, ... , 56, 57, 58, 60	59
max_features	None, "sqrt", "log2", 20 values from 0.1 to 1 uniformly distributed	23
criterion	'gini', 'entropy'	2
Liczba możliwych kombinacji		141,423,283,200

Tabela 4: Siatka hiperparametrów dla **ExtraTreesClassifier**.

3.2 Wybrana metryka do optymalizacji

Główną metryką jaką używaliśmy do badania optymalności otrzymanych wyników oraz prównywania wybranych technik optymalizacji jest *ROC AUC*. Natomiast przy badaniu tunowalności dla metody **RandomSearch** wykorzystałyśmy dodatkowo metryki *BrierScore* oraz *Accuracy*.

4 Eksperyment

4.1 Wybrane metody losowania punktów hiperparametrów

W celu przeprowadzenia eksperymentu, który pozwolił nam zbadać tunowalność hiperparametrów zastosowałyśmy dwie techniki optymalizacji **BayesOptimization** oraz **RandomSearch**. Tabela 5 przedstawia liczbę wykonanych iteracji dla każdego zbioru danych oraz algorytmu.

Zbiór danych	**RandomSearch**			**BayesOptimization**		
	SVM	**CatBoost**	**ExtraTrees**	**SVM**	**CatBoost**	**ExtraTrees**
Zbiór danych 1	449 (2)	449 (1)	449 (5)	100 (1)	100 (1)	100 (1)
Zbiór danych 2	449 (2)	449 (1)	449 (5)	200 (1)	200 (1)	200 (1)
Zbiór danych 3	449 (2)	449 (1)	449 (5)	100 (1)	100 (1)	100 (1)
Zbiór danych 4	449 (2)	449 (1)	449 (5)	100 (1)	100 (1)	100 (1)

Tabela 5: Liczba iteracji dla metod **BayesOptimization** oraz **RandomSearch**. W nawiasach znajduję się liczba powtórzeń pięciokrotnej kroswalidacji.

1. **RandomSearch** - w celu sprawdzenia tej metody została przez nas stworzona klasa **RandomSearchWithMetrics**, która w każdej iteracji generowała hiperparametry z podanej siatki i liczyła wartości kilku metryk: *ROC AUC*, *BrierScore*, *Accuracy* i *f1*.

2. **BayesOptimization** - w tym przypadku wykorzystałyśmy funkcję **skopt.BayesSearchCV** z biblioteki *scikit-optimize*.

4.2 Analiza otrzymanych wyników

Wzorując się artykułem [1], zdefiniowałyśmy dla każdego zbioru danych $\theta^{(j)*}$ - optymalną konfigurację hiperparametrów modelu dla danego zbioru danych $j \in \{1, 2, 3, 4\}$

$$\theta^{(j)*} := \arg\min_{\theta \in \Theta} R^{(j)}(\theta).$$

Przy czym wykorzystałyśmy trzy różne metryki:

1. $R^{(j)}(\theta) = -ROC\_AUC(\theta, j)$;

2. $R^{(j)}(\theta) = -Accuracy(\theta, j)$;

3. $R^{(j)}(\theta) = BrierScore(\theta, j)$.

Po dokonaniu powyższych obliczeń, stworzyłyśmy odpowiednie wizualizacje (Rozdział 7), które pozwoliły nam wysnuć następujące wnioski.

W przypadku **SVM** hiperparametry wybrane za pomocą metod **RandomSearch** i **BayesOptimization** osiągały lepsze wyniki niż domyślne wartości tych hiperparametrów dla każdego ze zbiorów danych. Natomiast, porównując liczbę iteracji obu metod w stosunku do najwyższej osiągniętej wartości *ROC AUC*, możemy stwierdzić, że technika **BayesOptimization** jest lepsza od **RandomSearch**. Ponadto, zbadałyśmy stabilność otrzymywanych wartości *ROC AUC* w stosunku do wykonanych iteracji. Okazuję się, że dla **SVM** obie metody wahają się poziomem stabilności dla zbiorów 1,3 oraz 4. Jednak, jeśli chodzi o zbiór 2, to metoda **RandomSearch** jest bardziej stabilna, niezależnie od liczby iteracji.

Jeśli chodzi o algorytm **CatBoostClassifier**, to wyniki dla obu metod były porównywalne. W przypadku zbiorów 1, 3 i 4 minimalnie lepsza okazywała się technika **BayesOptimization**, odwrotna sytuacja dla zbioru 2. Jednak dla każdego ze zbiorów danych najwyższe wartości osiągane są dla domyślnych wartości hiperparametrów. Stabilność osiąganych wartości *ROC AUC* względem liczby iteracji również okazuję się lepsza dla techniki **BayesOptimization** w przypadku zbioru 1. Dla zbiorów 3 oraz 4 wahania wartości *ROC AUC* w stosunku do wykonanych iteracji są dość podobne. Natomiast dla zbioru 2 metoda **RandomSearch** jest dużo bardziej stabilna niż metoda **BayesOptimization**.

Dla algorytmu **ExtraTreesClassifier** lepiej korzystać z metody **BayesOptimization** niż **RandomSearch**, jednak warto zauważyć, że dla każdego ze zbiorów danych najlepsze wyniki otrzymamy korzystając z domyślnych wartości hiperparametrów. Co więcej, metoda **BayesOptimization** osiąga bardziej stabilne wyniki dla *ROC AUC* w stosunku do liczby iteracji niz **RandomSearch** dla wszystkich zbiorów oprócz zbioru 2, gdzie **RandomSearch** wykazuję większą stabilność.

4.3 Analiza tunowalności modeli

W celu zbadania tunowalności wybranych przez nas modeli, zgodnie z artykułem [1], zdefiniowałyśmy dla każdego modelu θ^* - najbardziej optymalną konfigurację hiperparametrów średnio dla 4 zbiorów danych

$$\theta^* := \arg\min_{\theta \in \Theta} \left(\frac{R^{(1)}(\theta) + R^{(2)}(\theta) + R^{(3)}(\theta) + R^{(4)}(\theta)}{4} \right).$$

Jako $d^{(j)}$ oznaczamy całkowitą tunowalność algorytmu uczenia maszynowego na zbiorze danych $j \in \{1, 2, 3, 4\}$, czyli

$$d^{(j)} := R^{(j)}(\theta^*) - R(\theta^{(j)*}).$$

Tunowalność modeli badałyśmy przy użyciu techniki **RandomSearch**. Z przeprowadzonego przez nas eksperymentu wynika, że najbardziej tunowalnym algorytmem jest **SVM**,

osiąga on najwyższe wyniki niezależnie od wybranej metryki. Jeśli chodzi o metody **CatBoostClassifier** oraz **ExtraTreesClassifier** wykazują one znacznie niższą tunowalność dla każdej z wybranych metryk. Wyniki zostały przedstawione na wizualizacjach 1, 2, 3.

5 Testy statystyczne

W celu zbadania istotności różnic pomiędzy metodami **BayesOptimization** oraz **RandomSearch** wykorzystałyśmy test Manna-Whitneya dla każdego z algorytmów oraz zbioru danych. Przyjęłyśmy poziom istotności $\alpha = 0.05$ i testowałyśmy hipotezę zerową, że rozkłady metryki $ROC\ AUC$ metody **BayesOptimization** i **RandomSearch** są takie same, względem hipotezy alternatywnej, że rozkład $ROC\ AUC$ metody **BayesOptimization** jest stochastycznie większy od rozkładu $ROC\ AUC$ metody **RandomSearch**. Niech F_X oznacza rozkład $ROC\ AUC$ dla metody **RandomSearch**, a F_Y rozkład $ROC\ AUC$ dla metody **BayesOptimization**, wówczas test ma postać:

$$\begin{cases} H_0 : F_X = F_Y, \\ H_1 : X \overset{st}{<} Y. \end{cases}$$

Tabele 6, 7, 8 przedstwiają wyniki testów dla każdego z algorytmów. Zauważmy, że dla algorytmu **SVM** metoda **BayesOptimization** osiąga lepsze wyniki dla każdego ze zbiorów. Natomiast w przypadku algorytmów **CatBoost** technika **BayesOptimization** radzi sobie lepiej tylko w przypadku zbiorów 1 oraz 3. Jeśli chodzi o algorytm **ExtraTreesClassfier**, to również meto **BayesOptimization** radzi sobie lepiej dla wszystkich zbiorów poza zboriem 2. Powodem, dla którego otrzymałyśmy takie wyniki może być fakt, że algorytm **SVM** wykazuje najlepszą tunowalność spośród wszystkich algorytmów, oraz specyfika zbioru 2.

Zbiór danych	**p-wartość**	**Wniosek**
Zbiór danych 1	9.725e-14	$X \overset{st}{<} Y$
Zbiór danych 2	9.725e-14	$X \overset{st}{<} Y$
Zbiór danych 3	1.02e-12	$X \overset{st}{<} Y$
Zbiór danych 4	2.238e-13	$X \overset{st}{<} Y$

Tabela 6: Wyniki testu Manna-Whitneya dla algorytmu SVM.

Zbiór danych	**p-wartość**	**Wniosek**
Zbiór danych 1	1.106e-02	$X \overset{st}{<} Y$
Zbiór danych 2	1.000	$F_X = F_Y$
Zbiór danych 3	1.121e-05	$X \overset{st}{<} Y$
Zbiór danych 4	5.301e-01	$F_X = F_Y$

Tabela 7: Wyniki testu Manna-Whitneya dla algorytmu CatBoost.

Zbiór danych	p-wartość	Wniosek
Zbiór danych 1	4.324e-34	$X \overset{st}{<} Y$
Zbiór danych 2	1.000	$F_X = F_Y$
Zbiór danych 3	1.057e-44	$X \overset{st}{<} Y$
Zbiór danych 4	9.256e-42	$X \overset{st}{<} Y$

Tabela 8: Wyniki testu Manna-Whitneya dla algorytmu ExtraTreesClassifier.

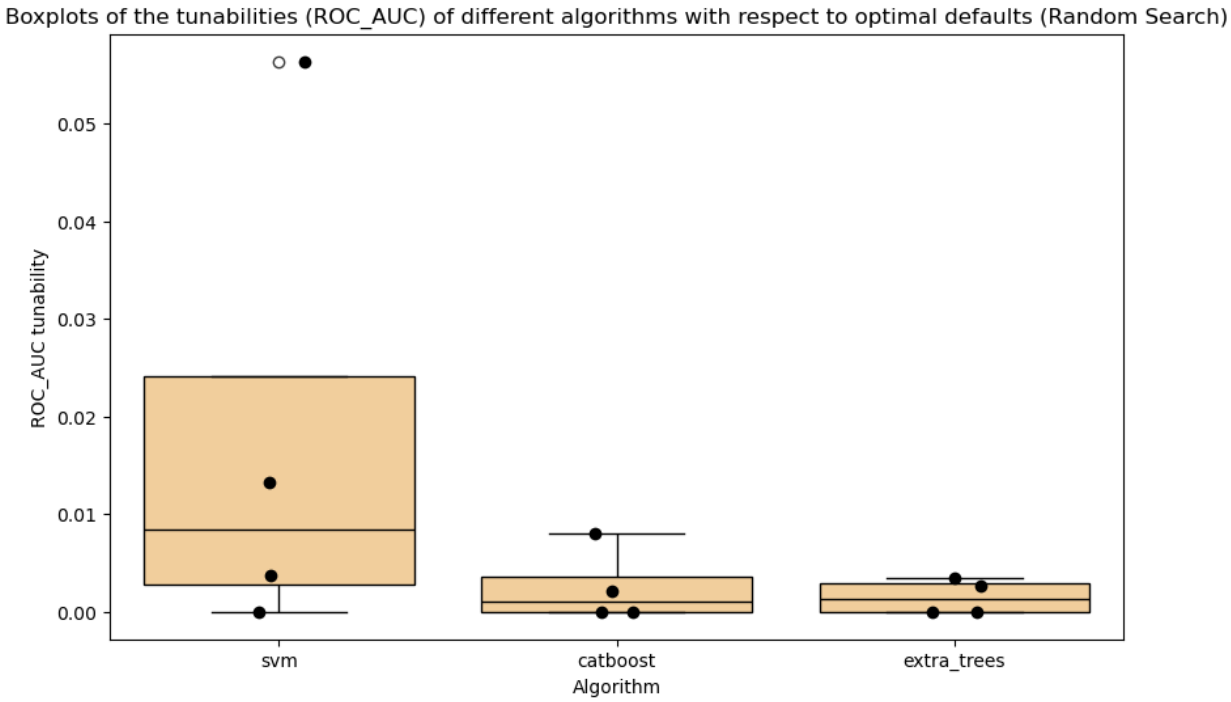
6 Podsumowanie

Podsumowując powyższe rozważania, stwierdzamy, że najlepszą tunowalnością cechuję się algorytm **SVM**. Najmniej tunowalny okazał się być algorytm **ExtraTreesClassifier**. Ponadto, metoda optymalizacji **BayesOptimization** w większości wykazywała lepsze wyniki niż **RandomSearch** dla wszystkich zbiorów danych, za wyjątkiem zbioru 2, dla którego ze względu na swoją specyfikę najlepiej używać domyślnych wartości hiperparametrów. W kwestii stabilności otrzymanych wartości $ROC\,AUC$ względem liczby iteracji, metody **BayesOptimization** oraz **RandomSearch** wykazywały podobną stabilność lub minimalnie bardziej stabilny okazywał się **BayesOptimization** w zależności od badanego algorytmu, bądź zbioru danych. Jedynym wyjątkiem jest zbiór 2, dla którego zawsze bardziej stabilna była metoda **RandomSearch**. Pondato, wyniki testu Manna-Whitneya wykazały, że metoda optymalizacji **BayesOptimization** w większości przypadków radzi sobie lepiej niż **RandomSearch**. Jednak dla specyficznych zbiorów danych, takich jak zbiór 2, różnice między obiema metodami nie były istotne.

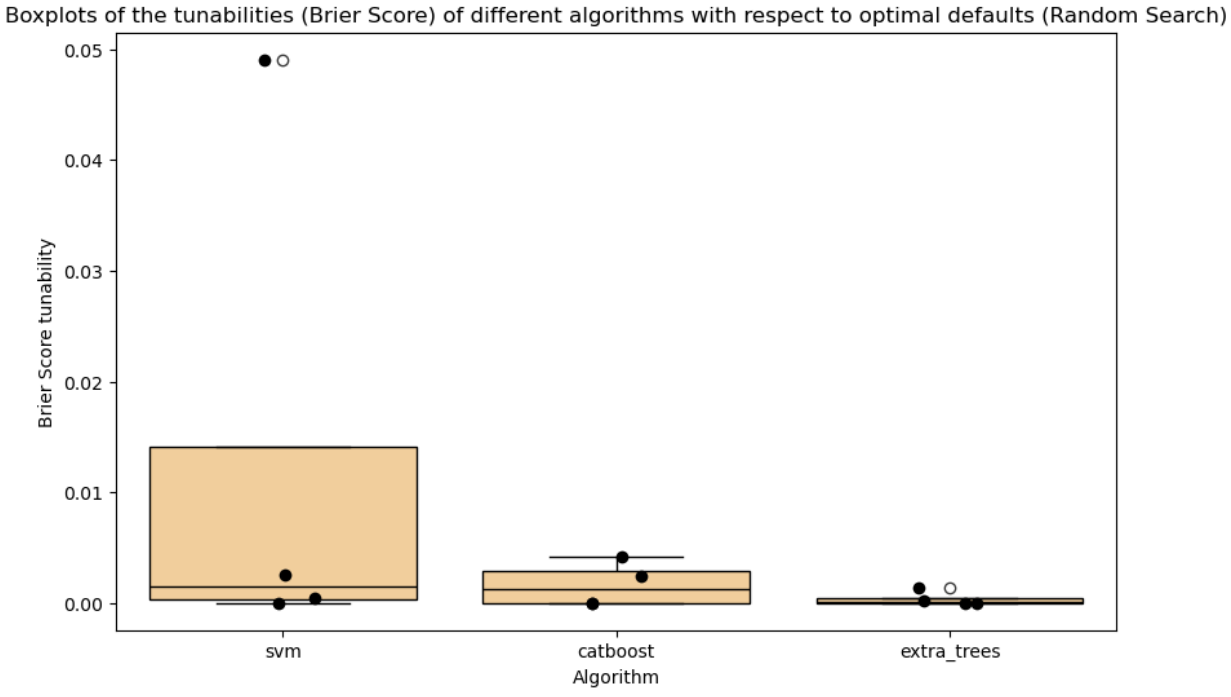
7 Wizualizacja otrzymanych wyników

Literatura

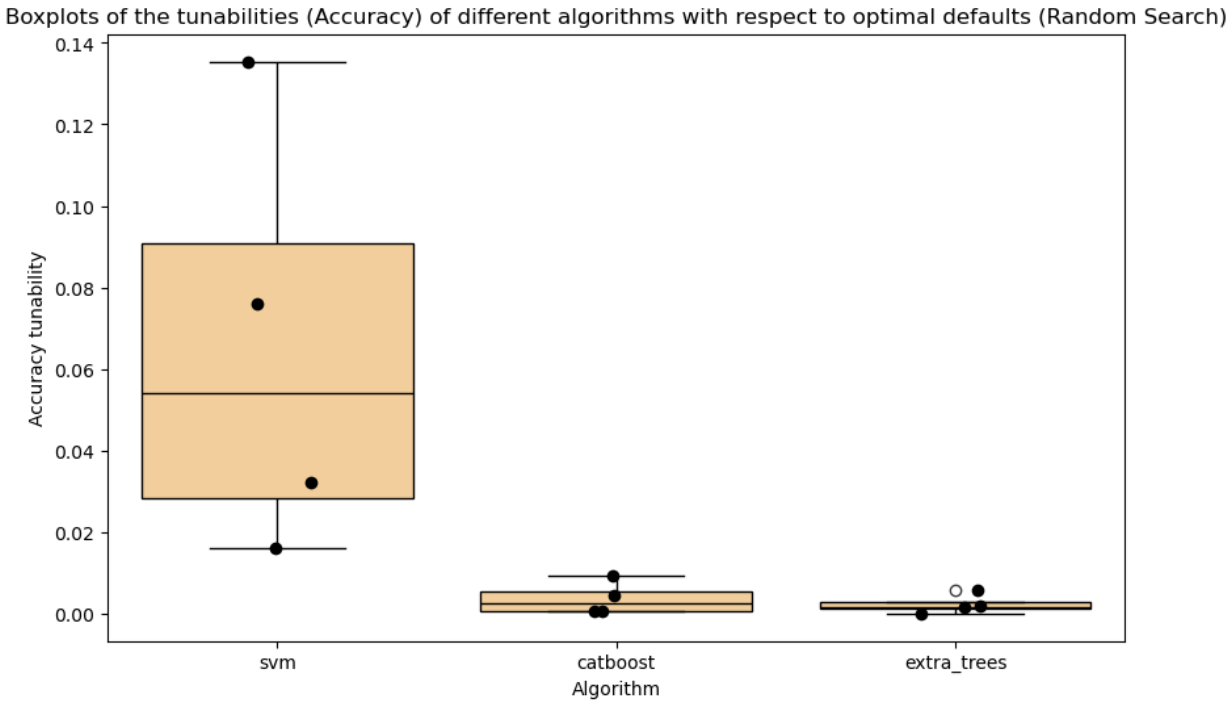
[1] P.Probst, A.Boulesteix, , Tunability: Importance of Hyperparameters of Machine Learning Algorithms, 2019.



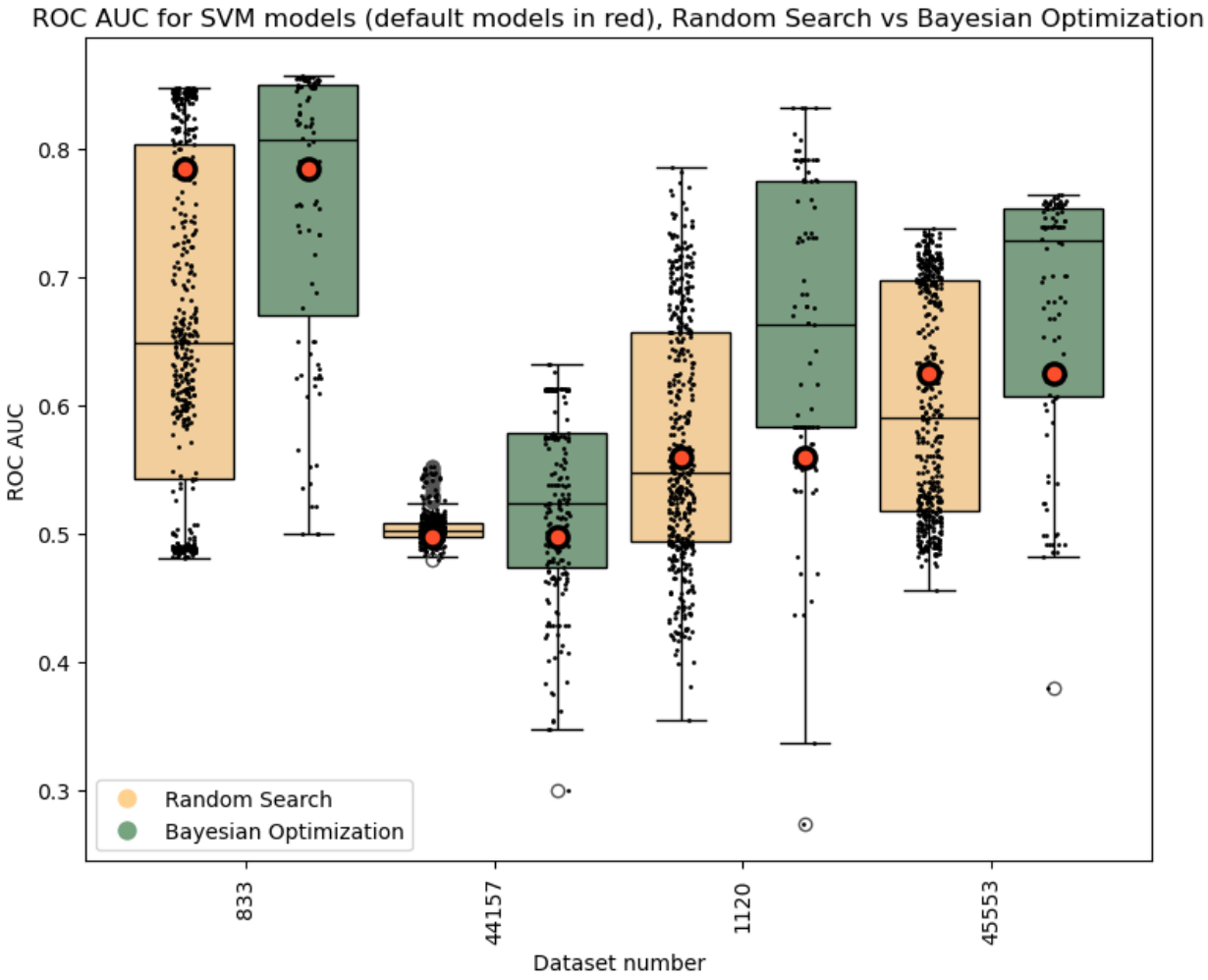
Rysunek 1: Tunowalność modelu **SVM** przy użyciu metryki *ROC AUC*.



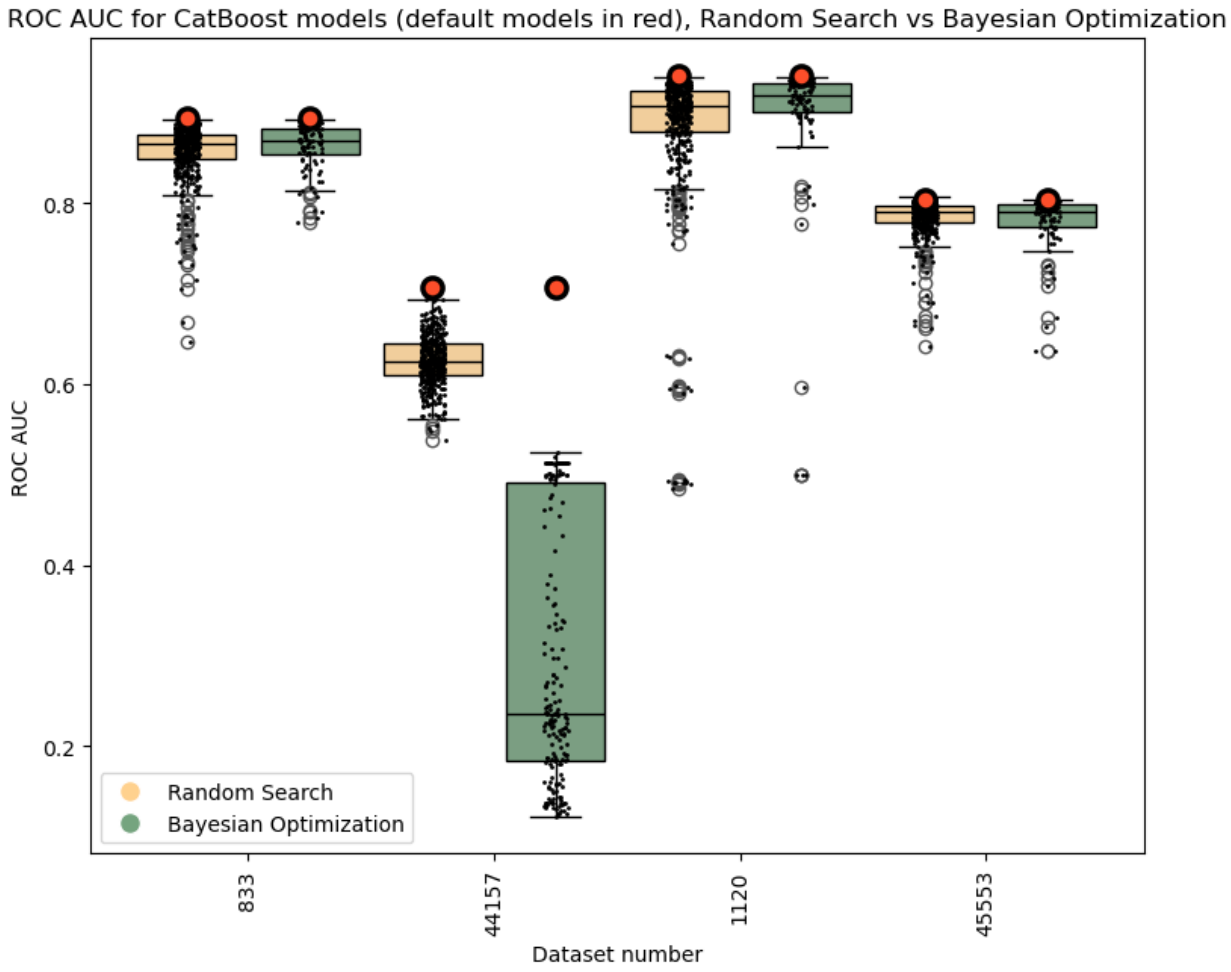
Rysunek 2: Tunowalność modelu **CatBoostClassifier** przy użyciu metryki *Brier Score*.



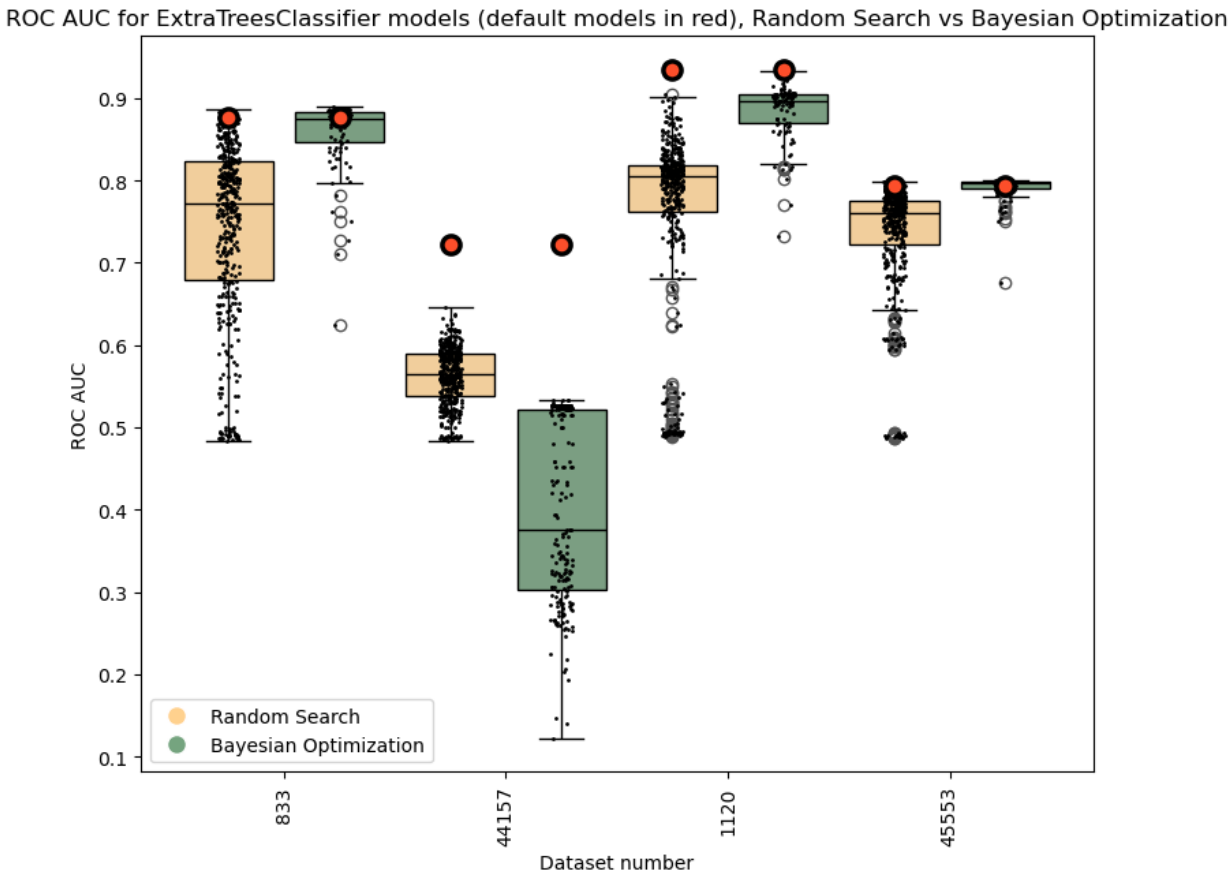
Rysunek 3: Tunowalność modelu **ExtraTreesClassifier** przy użyciu metryki *Accuracy*.



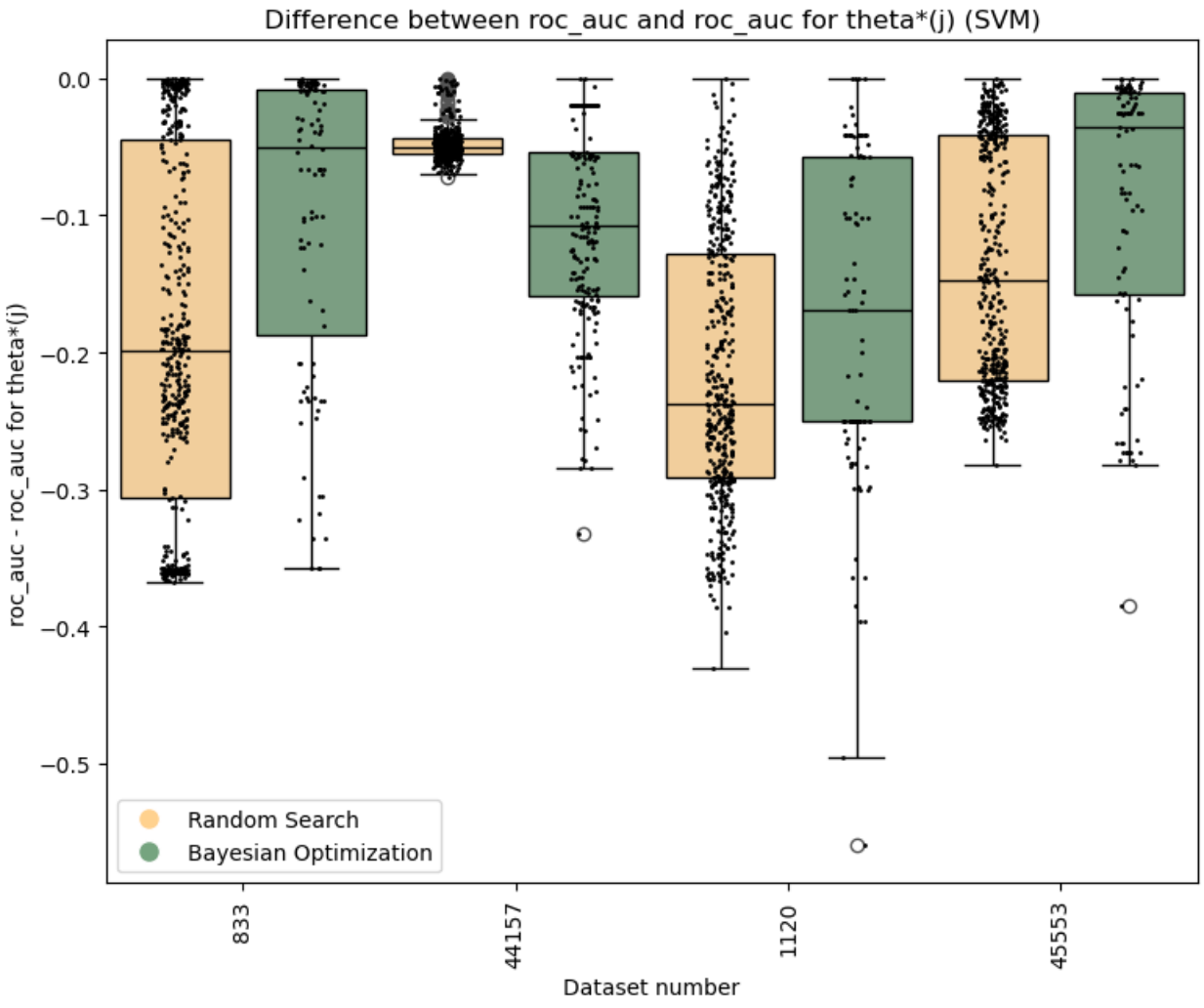
Rysunek 4: Porównanie wyników metryki *ROC AUC* dla metod **BayesOptimization** oraz **RandomSearch** z domyślnymi wartościami parametrów dla algorytmu **SVM** (czerwone punkty).



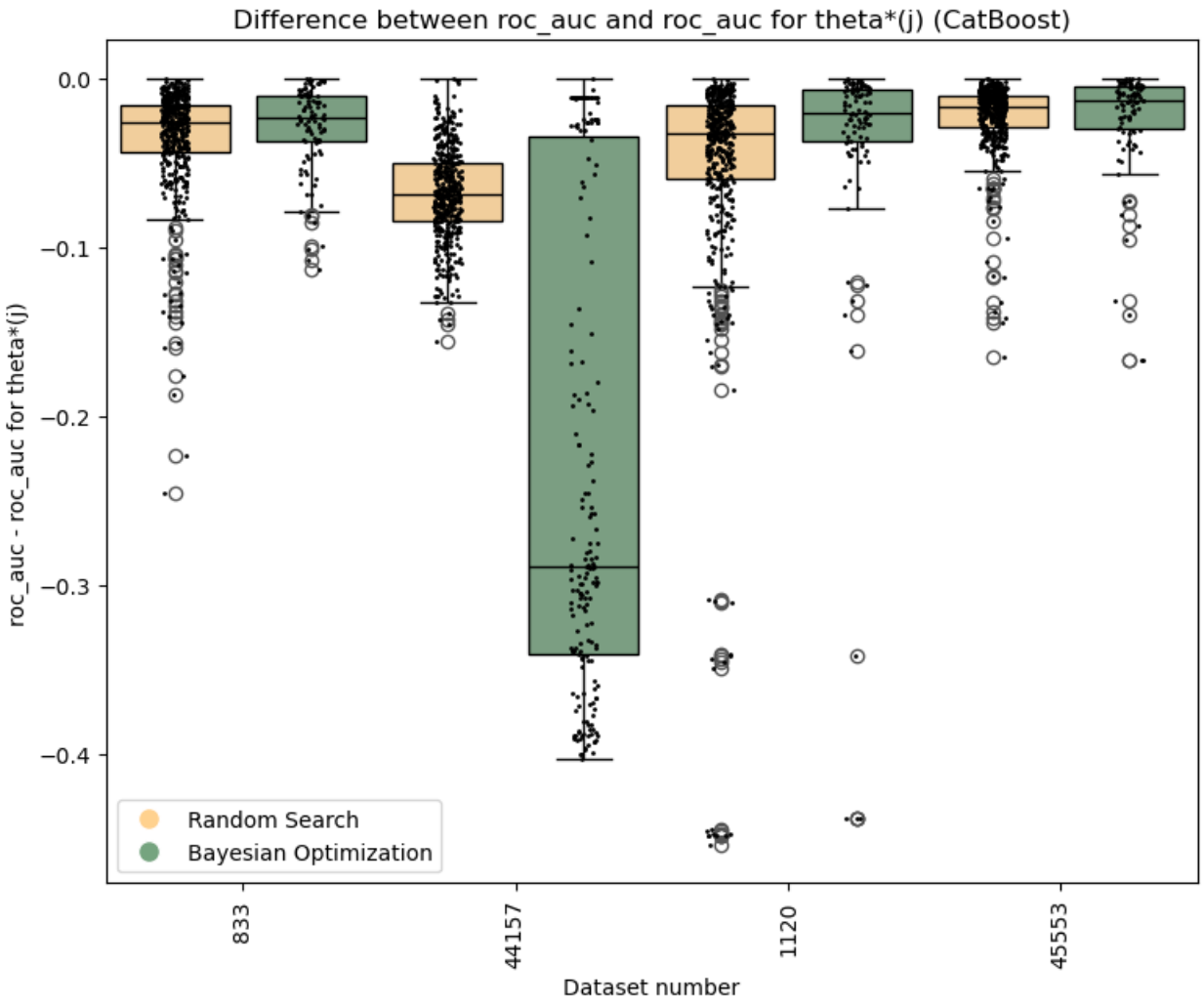
Rysunek 5: Porównanie wyników metryki *ROC AUC* dla metod **BayesOptimization** oraz **RandomSearch** z domyślnymi wartościami parametrów dla algorytmu **CatBoostClassifier** (czerwone punkty).



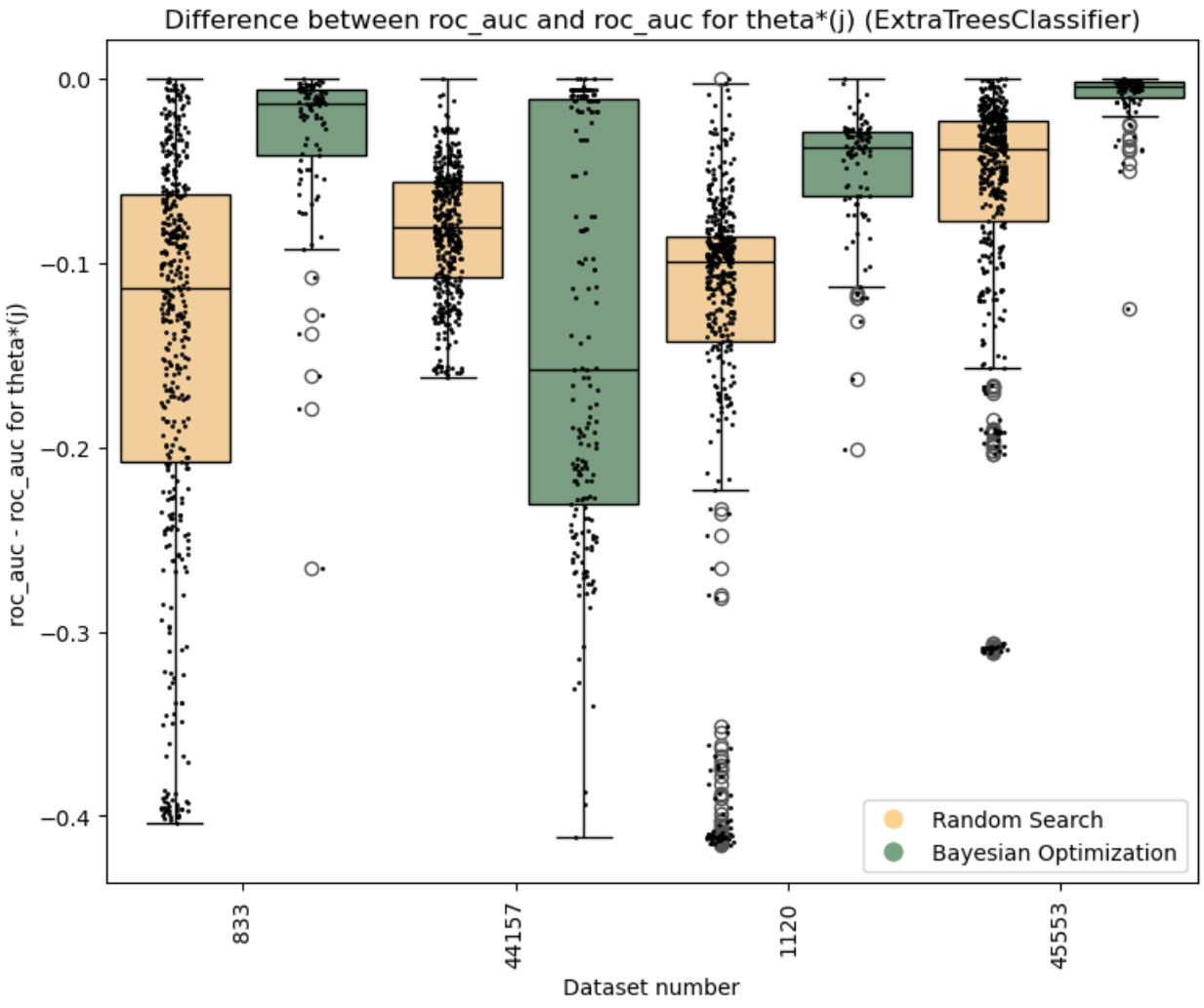
Rysunek 6: Porównanie wyników metryki *ROC AUC* dla metod **BayesOptimization** oraz **RandomSearch** z domyślnymi wartościami parametrów dla algorytmu **ExtraTreesClassifier** (czerwone punkty).



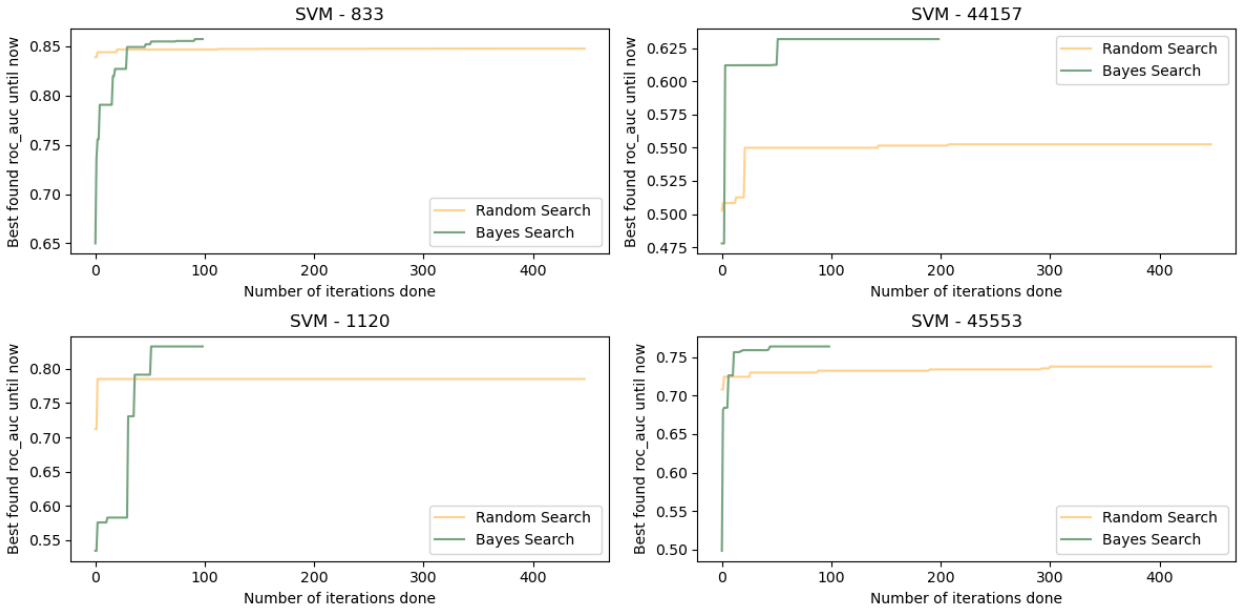
Rysunek 7: Różnica między *ROC AUC* metod **BayesOptimization** i **RandomSearch** dla algorytmu **SVM**.



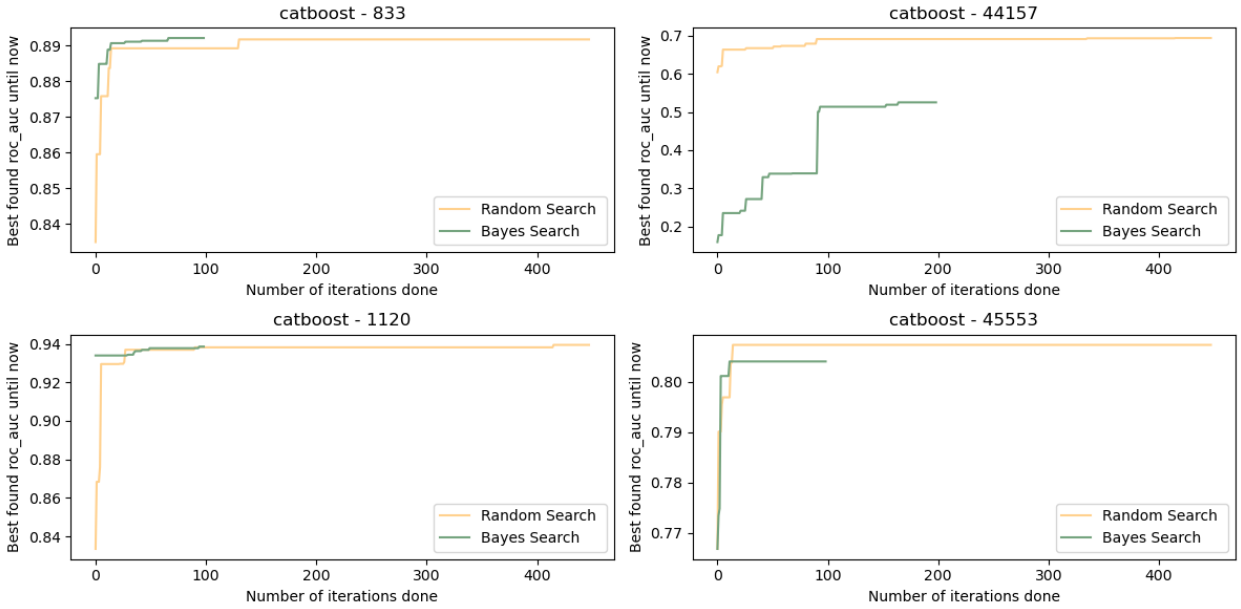
Rysunek 8: Różnica między *ROC AUC* metod **BayesOptimization** i **RandomSearch** dla algorytmu **CatBoostClassifier**.



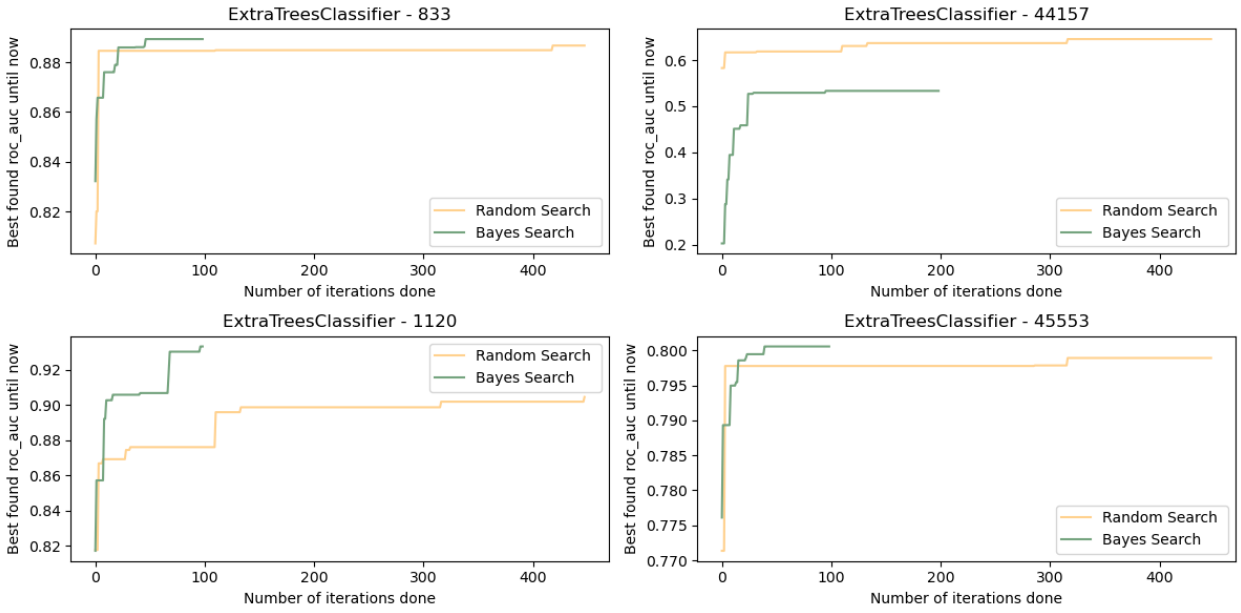
Rysunek 9: Różnica między *ROC AUC* metod **BayesOptimization** i **RandomSearch** dla algorytmu **ExtraTreesClassifier**.



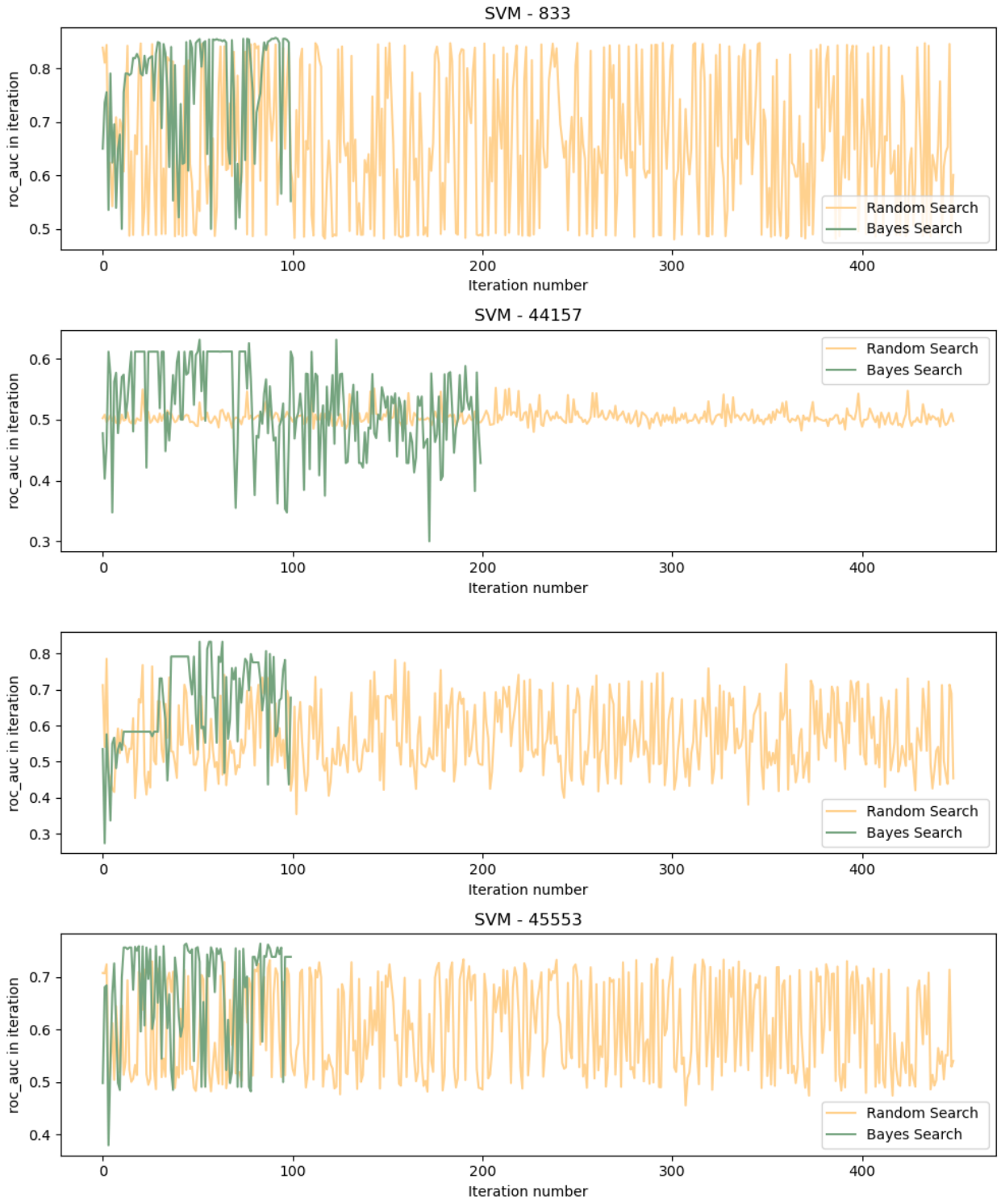
Rysunek 10: Liczba iteracji względem metryki *ROC AUC* metod **BayesOptimization** i **RandomSearch** dla algorytmu **SVM**.



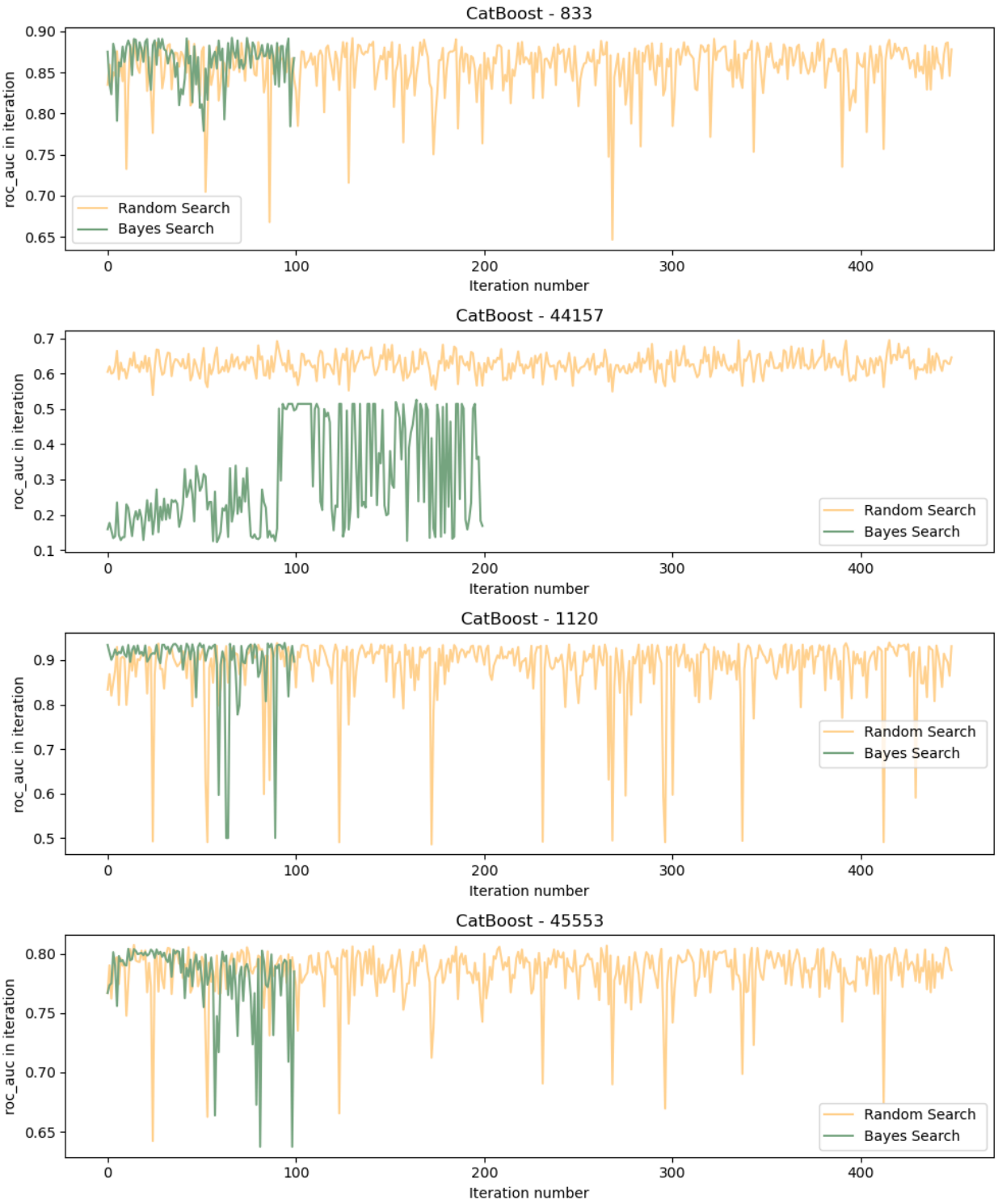
Rysunek 11: Liczba iteracji względem metryki *ROC AUC* metod **BayesOptimization** i **RandomSearch** dla algorytmu **CatBoostClassifier**.



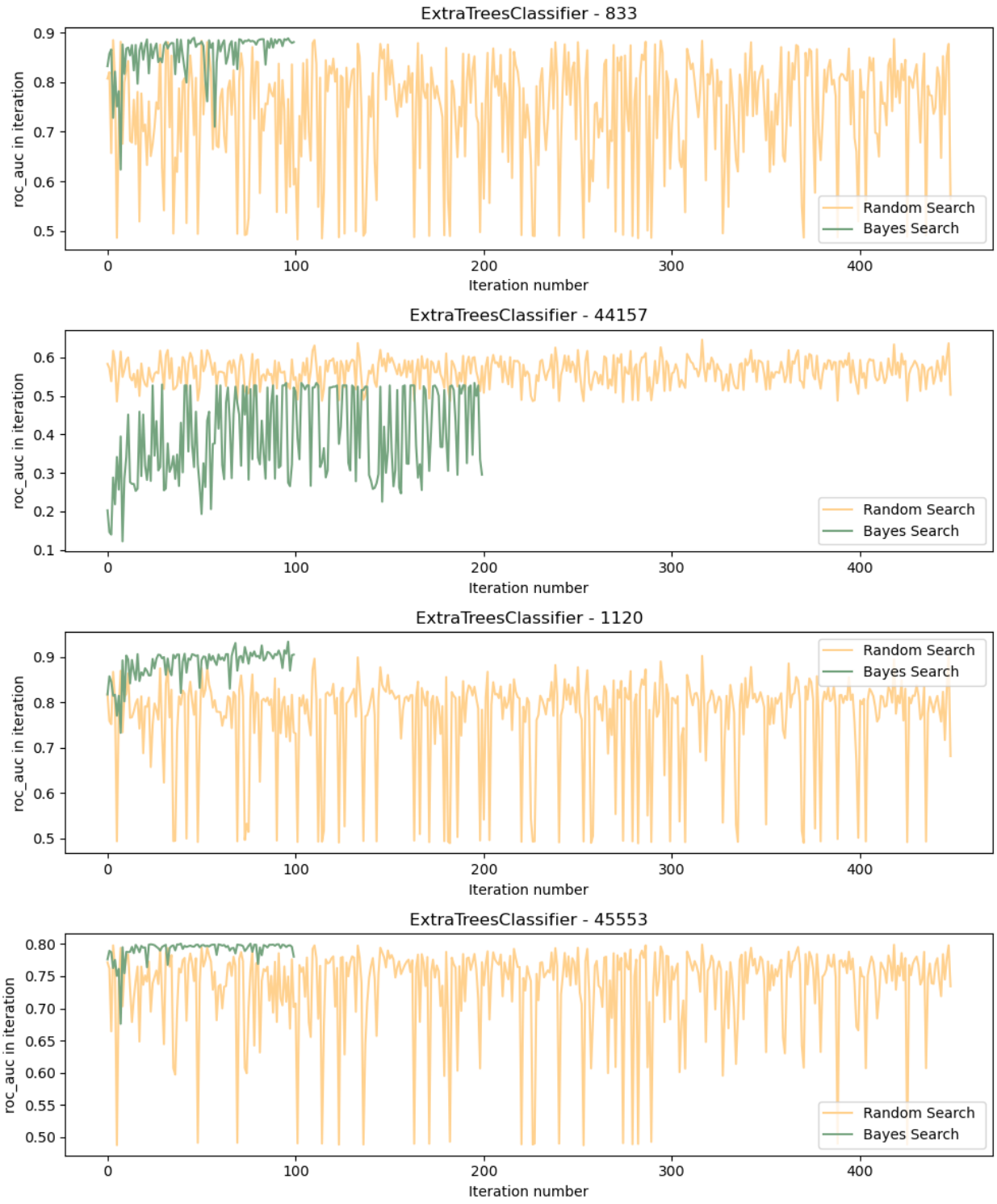
Rysunek 12: Liczba iteracji względem metryki *ROC AUC* metod **BayesOptimization** i **RandomSearch** dla algorytmu **ExtraTreesClassifier**.



Rysunek 13: Zmienność metryki *ROC AUC* względem iteracji metod **BayesOptimization** i **RandomSearch** dla algorytmu **SVM**.



Rysunek 14: Zmienność metryki metryki *ROC AUC* względem iteracji metod **BayesOptimization** i **RandomSearch** dla algorytmu **CatBoostClassifier**.



Rysunek 15: Zmienność metryki *ROC AUC* względem iteracji metod **BayesOptimization** i **RandomSearch** dla algorytmu **ExtraTreesClassifier**.