Warszawa, listopad 2023 r.

Automatyczne uczenie maszynowe

Praca domowa nr 1 - tunowalność algorytmów

Filip Skrzeczkowski
Filip Suchorab

Prowadzące: Anna Kozak, Katarzyna Woźnica

Streszczenie

Niniejszy dokument stanowi sprawozdanie z przeprowadzenia eksperymentów dotyczących tunowalności trzech algorytmów uczenia maszynowego wykorzystywanych do klasyfikacji binarnej: *random forest*, *svm*, *k neighbors* na czterech zbiorach danych. Wykorzystujemy do tego 2 różne techniki losowania punktów: *random search* i *optymalizację bayesowską*. Otrzymane wyniki poddaliśmy analizie i wyznaczyliśmy tunowalność poszczególnych algorytmów dla danych zbiorów danych.

1 Wstep

Celem pracy domowej było przeanalizowanie tunowalności 3 wybranych algorytmów uczenia maszynowego na 4 zbiorach danych. Zdecydowaliśmy, że zajmiemy się problemem klasyfikacji binarnej. W tym celu utworzyliśmy notes Jupiter, pozwalający na wyznaczenie historii tunowania dla poszczególnych algorytmów, co umożliwiło nam dalszą ich analizę.

1.1 Wybrane algorytmy

Do przetestowania tunowalności wybraliśmy 3 algorytmy: Las losowy, K najbliższych sąsiadów i SVM. Dla niektórych z nich wybraliśmy tylko część z możliwych hiperparametrów do dostrajania w sytuacji, gdy pozostałe mogłyby sprawić trudność dla jednej z metod losowania (np. hiperparametry boolowskie).

1.2 Zakresy hiperparametrów

Zakresy hiperparametrów lasu losowego i SVM ustaliliśmy na podstawie dostępnej literatury [1] [2]. Zakres hiperparametrów algorytmu K najbliższych sąsiadów ustaliliśmy na podstawie wielkości testowych zbiorów danych.

1.2.1 Las losowy

Do testów jako reprezentację klasyfikatora las losowy użyliśmy klasyfikator RandomForestClassifier z pakietu sklearn. Hiperparametry, które testowaliśmy pod kątem tunowalności to

- n_estimators zakres $(1, 100)$
- max_depth zakres $(1, 50)$
- min_samples_spit zakres $(2, 50)$
- min_samples_leaf zakres $(1, 50)$

1.2.2 K najbliższych sąsiadów

Jako reprezentację klasyfikatora k najbliższych sąsiadów klasyfikator KNeighborsClassifier z pakietu sklearn. Hiperparametry, które testowaliśmy pod kątem tunowalności to

- n_neighbors zakres $(1, 500)$

1.2.3 SVM

Jako reprezentację klasyfikatora k najbliższych sąsiadów klasyfikator SVC z pakietu sklearn. Hiperparametry, które testowaliśmy pod kątem tunowalności to

- C zakres $(2^{-10}, 2^{10})$
- gamma zakres $(2^{-10}, 1)$

1.3 Zbiory danych

Rozważania prowadzimy na czterach zbiorach danych zdobytych z serwisu OpenML.

- credit-g zakres
- blood-transfusion-service-center
- diabetes
- hill-valley

W wyborze zbiorów danych kierowaliśmy się tym, by nie były one zbyt duże, aby nie tracić zbyt wiele czasu na trenowanie modeli, a zamiast tego skupić się na analizie wyników. Wczytujemy je poprzez bezpośrednie pobranie z OpenMl w samym kodzie.

1.4 Przebieg eksperymentów

Aby obliczyć tunowalność danego algorytmu w pierwszej kolejności wyznaczono za pomocą metody *random search* najlepszy globalnie (dla wszystkich zbiorów danych) wektor hiperparametrów (ozn. $\theta*$. Następnie z użyciem dwóch metod losowania punktów przeprowadzono optymalizację dla poszczególnych zbiorów danych. Jako tunowalność przyjęto różnicę między oceną jakości modelu wykorzystującego hiperparametry najlepsze dla poszczególnego zbioru oraz modelu wykorzystującego θ^*.

Za miarę jakości uznaliśmy AUC krzywej ROC.

2 Random Search

2.1 Opis metody

W celu zdobycia informacji o najlepszych hiperparametrach wybranych algorytmów użyto metody Random Search. Metoda ta polega na wylosowaniu pewnej liczby zestawów hiperparametrów z danej przestrzeni możliwych wartości hiperparametrów i wytrenowaniu modelu z wylosowanymi hiperparametrami. Metodę random search zrealizowano przy pomocy funkcji RandomizedSearchCV dostępnej w pakiecie skleran.

Dla każdego algorytmu wylosowano 500 konfiguracji hiperparametrów i następnie dla każdej konfiguracji wytranowano model na każdym ze zbioru danych. Podczas trenowania użyto 3-krotnej kroswalidacji aby otrzymać średni wynik AUC dla danej konfiguracji.

2.2 Wyniki

Dla każdego algorytmu wyliczono θ^* licząc średnią z AUC na czterech zbiorach danych dla danego θ. θ^* oraz średnią AUC dla danego θ^* dla każdego algorytmu przedstawiono w tabeli 1. Tunowalność każdego algorytmu na każdym zbiorze danych przedstawiono w tabeli 2.

W celu przeanalizowania jak ilość iteracji metody Random Search wpływa na jakość modelu wyznaczono zależność najlepszego wytrenowanego modelu do momentu danej iteracji od liczby iteracji. Zależność tę dla każdego zbioru danych przedstawiono na wykresach: 1, 2, 3, 4. Z wykresów można wywnioskować, że metoda może dawać w kolejnych iteracjach lepsze wyniki nawet do 400 iteracji. Największa poprawa w jakości modeli występowała w pierwszych 200 iteracjach. Po 200 pierwszych iteracjach w $\frac{3}{4}$ przypadków wyniki optymalizacji się nie polepszały. Po 400 iteracjach wyniki w każdym z eksperymentów były stabilne.

3 Optymalizacja bayesowska

Optymalizacja bayesowska umożliwiła nam inne podejście do tematu i ponowne wyznaczenie tunowalności algorytmów. Do jest przeprowadzenia wykorzystano funkcję `gp_minimize` z pakietu `scikit-optimize`.

Historię tunowania dla poszczególnych zbiorów danych prezentują wykresy 5, 6, 7 i 8. Sugerują one, że dla testowanych danych zmiany w poszczególnych iteracjach zdają się być dość chaotyczne. Czasami można jednak zauważyć w zmianach AUC pewne trendy, które sugerują poprawę jakości w czasie w przypadku niektórych zbiorów danych i algorytmów. Choć patrząc na wykresy można dojść do wniosku, że "stabilizacja" to nieco zbyt mocne słowo, to można zauważyć, że ok. **20.** iteracji zakres osiąganych wyników przestaje się już znacząco zmieniać.

Zgodnie ze wcześniej opisanymi założeniami obliczono tunability dla każdego algorytmu, co opisuje tabela 3. Jak widać, metoda ta w tunowaniu radzi sobie zauważalnie gorzej, o czym świadczą niższe wartości tunability (dla pierwszych dwóch zbiorów danych są one nawet ujemne), a także średnio mniejsze wartości miary AUC.

4 Wnioski

Przeprowadzane eksperymenty sugerują że w ogólności tunowalność wszystkich testowanych algorytmów dla naszych zbiorów danych jest dość niska, przy czym należy mimo wszystko zauważyć, że metoda random search radzi sobie istotnie lepiej. Możemy zatem wysnuć wniosek, że mamy do czynienia ze

zjawiskiem bias samplingu, które sprawia, że różny wybór punktów może doprowadzić nas do różnych rezultatów.

Przyczyny takiego stanu rzeczy upatrujemy w dość niewielkiej liczbie próbek we wszystkich zbiorach, snując przy tym hipotezę, że dla zbioru bardziej licznego być może udałoby się uzyskać bardziej satysfakcjonujące efekty.

Literatura

Literatura

[1] Koen Smets, Brigitte Verdonk, Elsa M. Jordaan, *Evaluation of Performance Measures for SVR Hyperparameter Selection*

[2] Philipp Probst, Marvin Wright, Anne-Laure Boulesteix, *Hyperparameters and Tuning Strategies for Random Forest*, Published on February 27, 2019.

Tunability: Importance of Hyperparameters of Machine Learning Algorithms

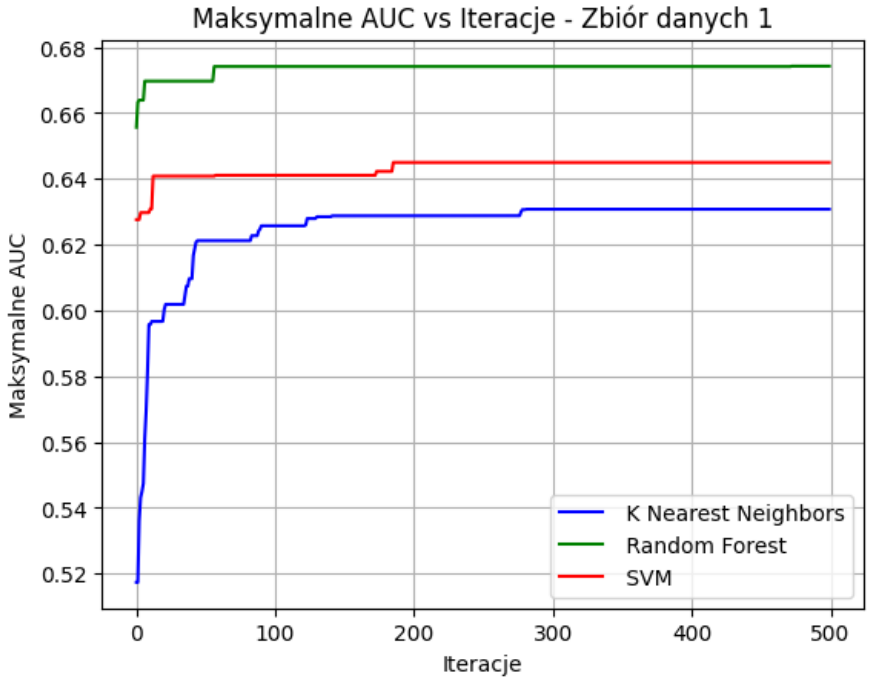
A Random Search

Tabela 1: θ^* rozważanych algorytmów

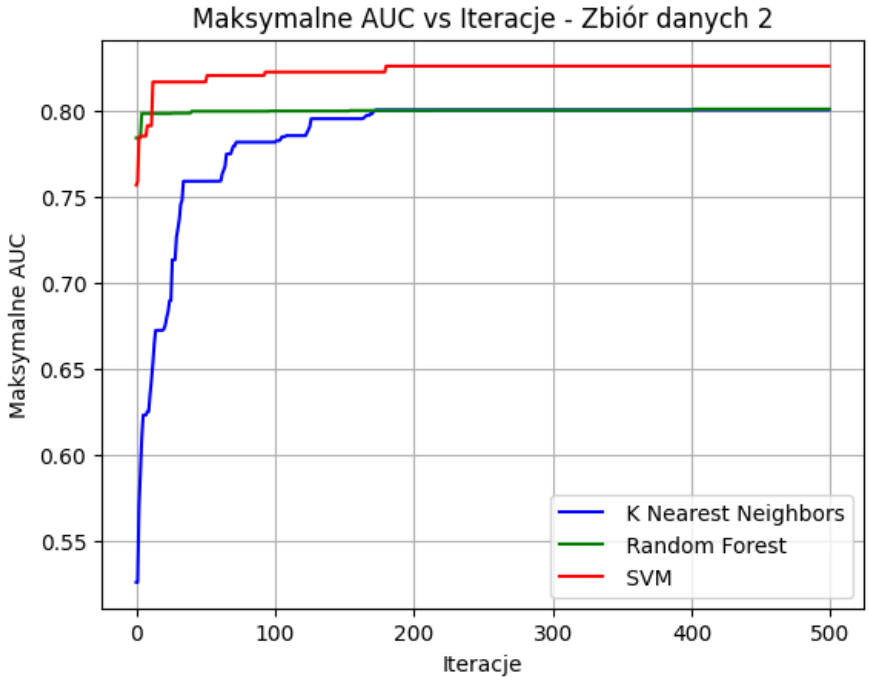
Algorytm	Średnie AUC	θ^*
k_neighbors	0.689	[143]
random_forest	0.702	$[100, 3, 32, 17]$
SVM	0.747	$[0.057, 891.320]$

Tabela 2: Tunowalność algorytmów na każdym zbiorze danych

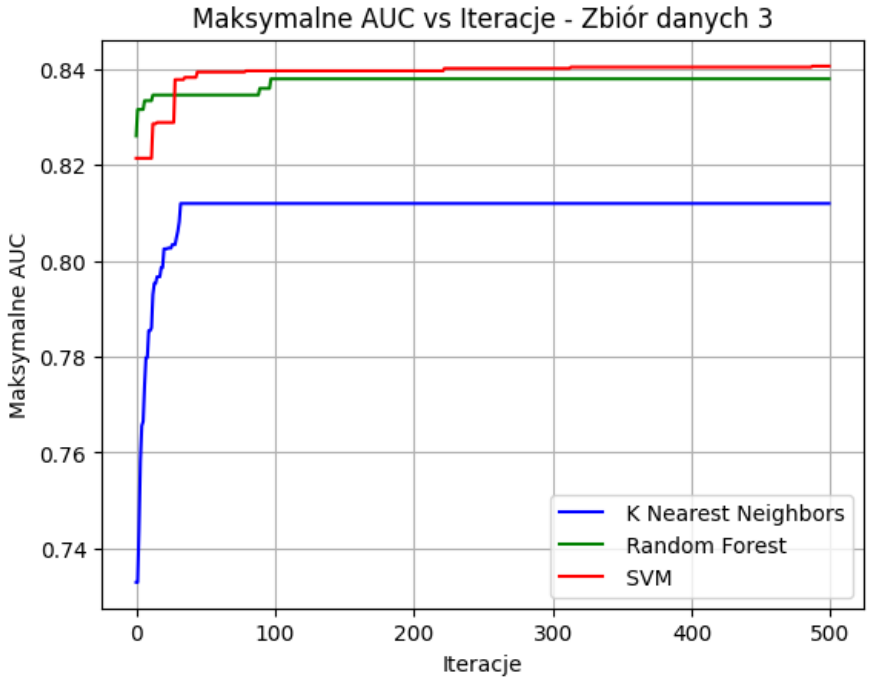
Algorytm	Zbiór danych	tunowalność	AUC	**Najlepsze θ dla zbioru**
k_neighbors	0	0.003	0.631	[282]
k_neighbors	1	0.011	0.801	[174]
k_neighbors	2	0.015	0.812	[33]
k_neighbors	3	0.013	0.554	[2]
random_forest	0	0.016	0.674	$[87, 4, 3, 6]$
random_forest	1	0.011	0.801	$[75, 42, 48, 8]$
random_forest	2	0.009	0.838	$[13, 23, 4, 23]$
random_forest	3	0.031	0.563	$[96, 8, 4, 30]$
SVM	0	0.004	0.645	$[0.135, 328.183]$
SVM	1	0.008	0.826	$[0.017, 361.662]$
SVM	2	0.008	0.841	$[0.615, 2.015]$
SVM	3	0.032	0.727	$[0.469, 1024]$



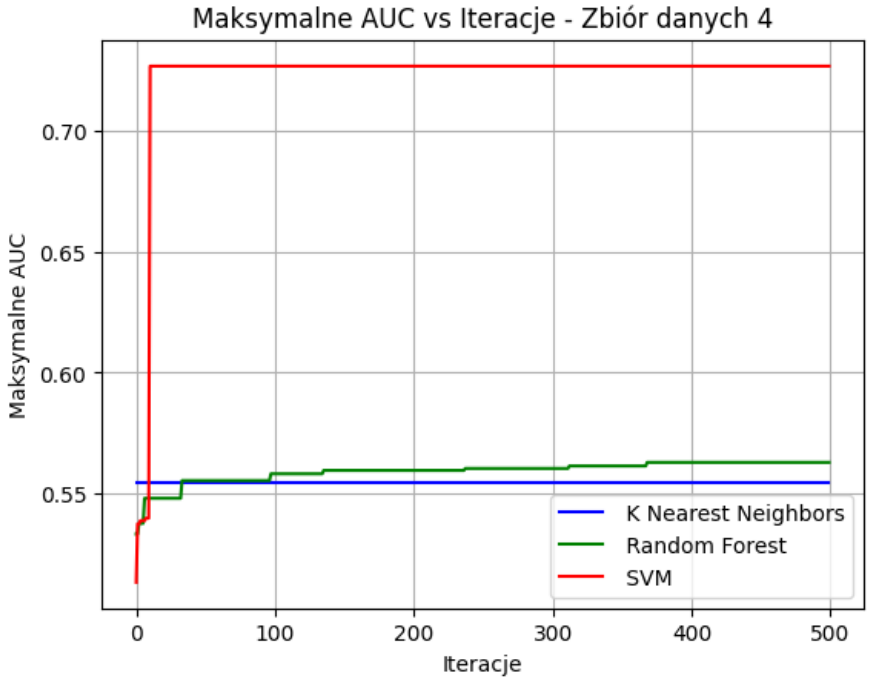
Rysunek 1: Zależność najlepszego znalezionego modelu od liczby iteracji na zbiorze danych 1



Rysunek 2: Zależność najlepszego znalezionego modelu od liczby iteracji na zbiorze danych 2

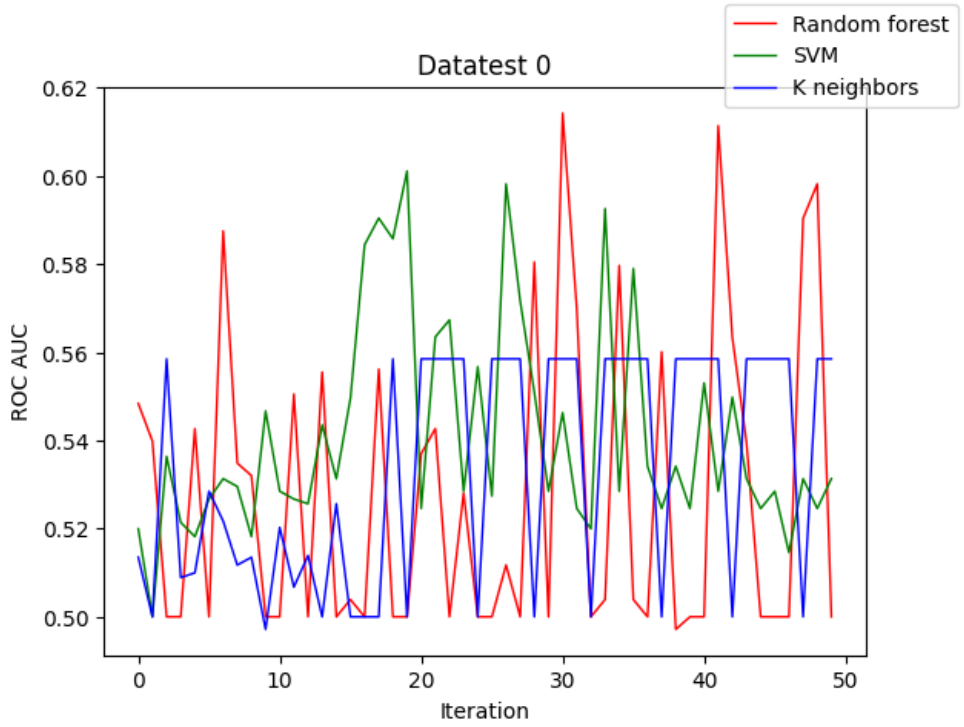


Rysunek 3: Zależność najlepszego znalezionego modelu od liczby iteracji na zbiorze danych 3

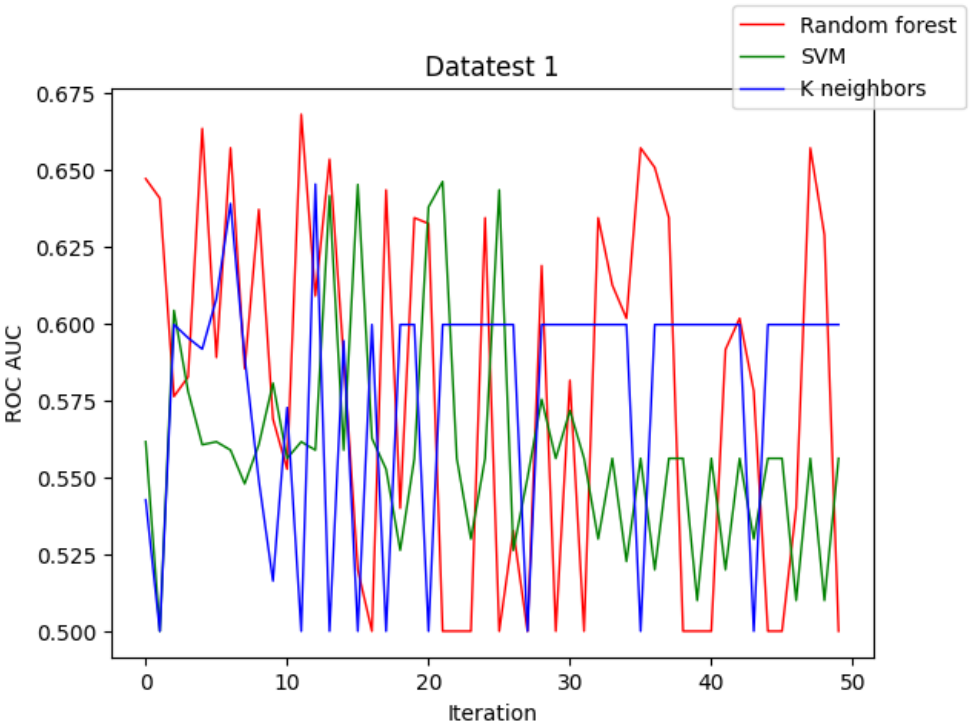


Rysunek 4: Zależność najlepszego znalezionego modelu od liczby iteracji na zbiorze danych 4

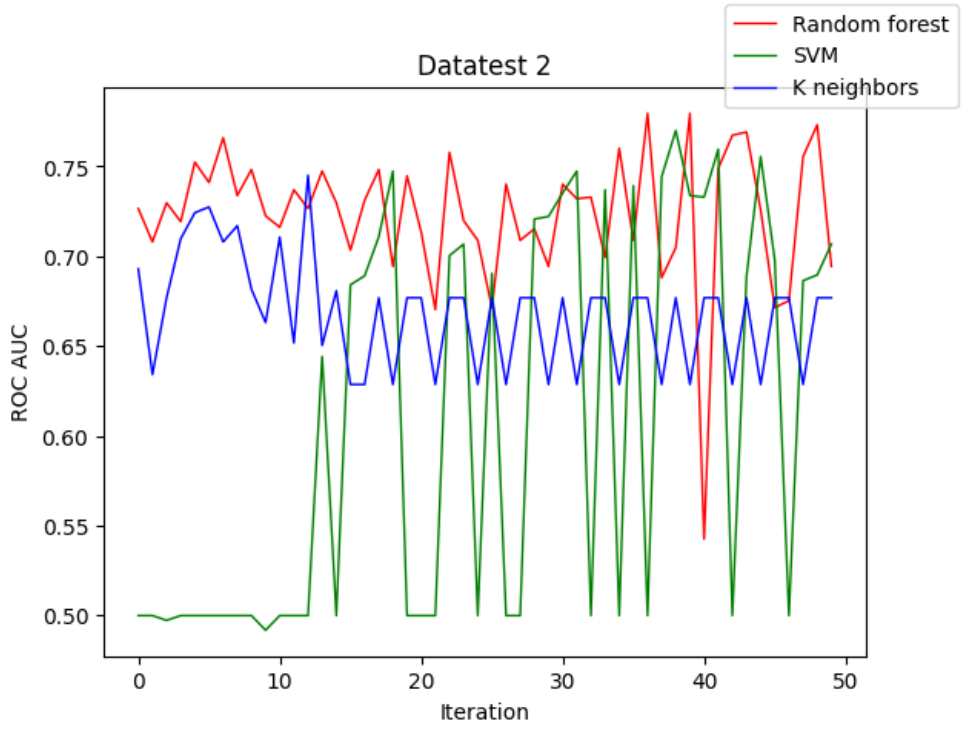
B Optymalizacja bayesowska



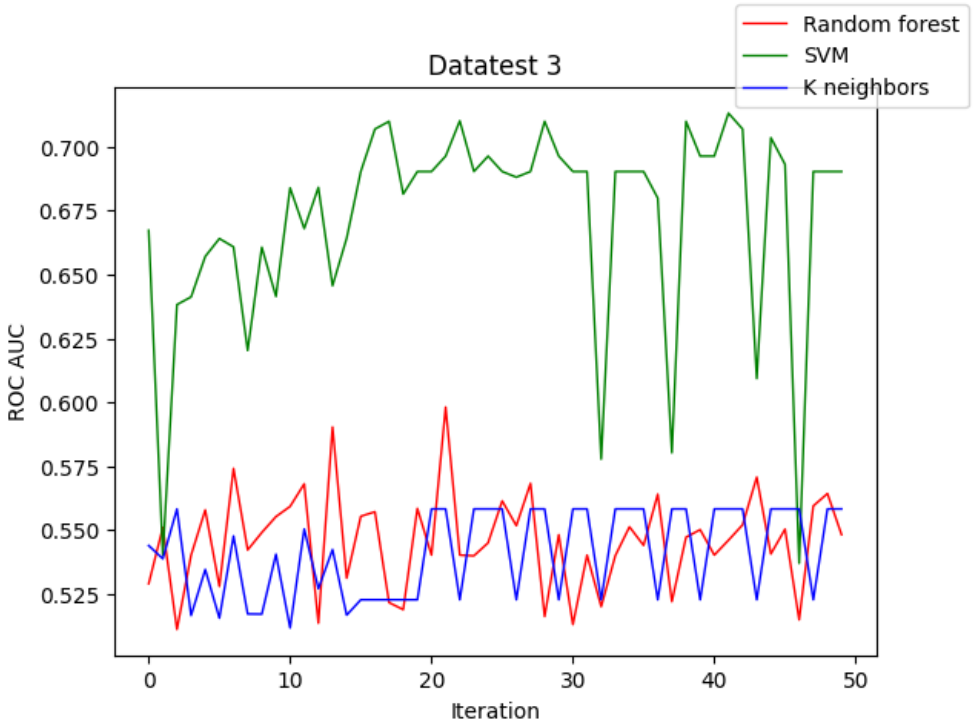
Rysunek 5: Historia tunowania algorytmów z użyciem optymalizacji bayesowskiej dla zbioru danych 0



Rysunek 6: Historia tunowania algorytmów z użyciem optymalizacji bayesowskiej dla zbioru danych 1



Rysunek 7: Historia tunowania algorytmów z użyciem optymalizacji bayesowskiej dla zbioru danych 2



Rysunek 8: Historia tunowania algorytmów z użyciem optymalizacji bayesowskiej dla zbioru danych 3

Tabela 3: Tunability dla optymalizacji bayesowskiej

Algorytm	**Dataset**	**Tunability**	**AUC najlepszego θ dla zbioru**	**Najlepsze θ dla zbioru**
k_neighbors	0	-0.143491	0.558507	$[1]$
k_neighbors	1	-0.141993	0.645401	$[21]$
k_neighbors	2	0.021181	0.745140	$[21]$
k_neighbors	3	0.029469	0.558346	$[1]$
random_forest	0	-0.091743	0.614251	$[8, 45, 3, 1]$
random_forest	1	-0.120664	0.668102	$[11, 23, 46, 15]$
random_forest	2	0.011375	0.779604	$[7, 49, 49, 4]$
random_forest	3	0.052802	0.598182	$[11, 48, 50, 1]$
SVM	0	-0.108864	0.601121	$[998.202, 2.214]$
SVM	1	-0.130529	0.646204	$[993.532, 6.373]$
SVM	2	0.003133	0.770060	$[1.177, 5.0816]$
SVM	3	0.075385	0.713174	$[990.696, 3.301]$