Politechnika Warszawska
Wydział Matematyki i Nauk Informacyjnych

Automatyczne uczenie maszynowe

Analiza tunowalności hiperparametrów wybranych algorytmów uczenia maszynowego

Autorzy:

Siwak Paweł

Wujkowski Daniel

19.11.2023

Spis treści

1 Wstęp

Poniższy dokument jest sprawozdaniem z realizacji pierwszego zadania domowego w ramach przedmiotu **Automatyczne uczenie maszynowe** na wydziale **MiNI PW**. Przeprowadzony został eksperyment polegający na próbie optymalizacji hiperparametrów wybranych modeli uczenia maszynowego przy użyciu dwóch metod samplingu - wybór punktów z rozkładu jednostajnego oraz technika bayesowska.

2 Szczegółowy opis eksperymentu

W ramach zadania konieczne było podjęcie kilku kluczowych decyzji, tj.:

1. Wybór zbiorów danych
2. Wybór modeli uczenia maszynowego
3. Wybór tunowanych hiperparametrów oraz określenie ich zakresów
4. Wybór metod samplingu

W dalszej części tego rozdziału przedstawione oraz uzasadnione zostaną podjęte decyzje.

2.1 Wybór zbiorów danych

Pierwsza decyzja została podjęta w kontekście rozwiązywanego problemu. Postanowiono wybrać klasyfikację binarną z powodu intuicyjności metod oceniania jakości rozwiązania oraz jako problem prostszy w analizie niż klasyfikacja wieloetykietowa.

Zbiory danych na potrzeby eksperymentu zostały wybrane spośród tych dostępnych na platformie **OpenML**. Kryteriami decydującymi o wyborze były:

- Wielkość - celowano w zbioru o rozmiarze od 10 do 20 tys. wierszy tak, aby zapewnić odpowiednią reprezentatywność
- Brak missing values - docelowo starano się, aby w wybranych zbiorach nie było żadnych missing values i aby tym samym móc ograniczyć preprocessing danych do minimum w celu skupienia się na właściwym zadaniu
- Reprezentatywne klasy - ważnym kryterium była również reprezentatywność przewidywanych klas, tj. starano się uniknąć sytuacji, gdzie jedna z klas byłaby wynikiem np. w 95% przypadków

Tym samym wybrane zostały następujące cztery zbiory danych: *elevators* (id: 846), *MagicTelescope* (id: 44125), *mozilla4* (id: 1046) oraz *PhishingWebsites* (id: 4534).

2.2 Wybór modeli uczenia maszynowego oraz tunowanych hiperparametrów

W ramach eksperymentu przebadane zostały trzy algorytmy uczenia maszynowego, skorzystano z implementacji w Pythonowej bibliotece **scikit-learn**:

1. Drzewo decyzyjne (DecisionTreeClassifier)
2. Las lasowy (RandomForestClassifier)
3. Regresja logistyczna z penalty ElasticNet (LogisticRegression)

Doboru tunowanych hiperparametrów oraz ich zakresów dokonano głównie w oparciu o (2), a następnie nieco dostosowano w oparciu o zdrowy rozsądek oraz analizę procesu uczenia (przykładowo maks. liczbę drzew w lesie losowym zmniejszono z 2000 do 500 z powodu bardzo długiego czasu uczenia i braku widocznej poprawy jakościowej dla dużych wartości tego parametru).

Tym samym eksperyment przeprowadzano przy następujących zakresach badanych hiperparametrów:

- DecisionTreeClassifier
  - max_depth (1, 30)
  - min_samples_leaf (1, 60)
  - min_samples_split (2, 60)
- RandomForestClassifier
  - n_estimators (1, 500)
  - max_samples - rozkład jednostajny na przedziale (0.1, 1)
  - max_features - rozkład jednostajny na przedziale (0.1, 1)
- LogisticRegression
  - C - rozkład loguniform na przedziale (10^{-3}, 10^2)
  - l1_ratio - rozkład jednostajny na przedziale (0, 1)

2.3 Wybór metod samplingu

Z uwagi na potencjalnie ciekawsze wyniki, w przypadku wyboru punktów z rozkładu jednostajnego zdecydowano się wykorzystać metodę Random Search, a dokładniej implementację **RandomizedSearchCV** z ww. biblioteki. Jeśli chodzi o technikę bayesowską, to postawiono na metodę Bayesian Optimization zaimplementowaną jako **BayesSearchCV** w bibliotece **scikit-optimize**.

3 Analiza wyników

W poniższym rozdziale przedstawiono analizę wyników uzyskanych w ramach przeprowadzenia wyżej opisanego eksperymentu.

3.1 Stabilizacja metod optymalizacji hiperparametrów

W pierwszej kolejności zdecydowano się przebadać tempo stabilizacji używanych metod, tj. określić ile iteracji potrzebne jest, aby uzyskać stabilne wyniki optymalizacji. W obydwu przypadkach maksymalna liczba iteracji ustawiona została na 50.

W celu określenia momentu stabilizacji zdecydowano się przygotować wykresy typu *best so far* przedstawiające najlepszy uzyskany wynik (w tym przypadku AUC) w danej iteracji i wszystkich poprzednich. Jako **moment stabilizacji** określa się zatem numer iteracji, od której przedstawiona krzywa zupełnie się wypłaszczyła. Wszystkie wykresy zostały umieszczone w dodatku na końcu raportu.

Można zauważyć, że w każdym przypadku krzywa wypłaszczyła się i z dość dużą pewnością da się stwierdzić, że algorytmy się ustabilizowały, co dodatkowo zgadzałoby się z informacjami znalezionymi w dyskusji użytkowników forum StackExchange (1) (przynajmniej w przypadku Random Searcha).

Dla każdej z metod samplingu obliczono również medianę liczby iteracji (zarówno osobno dla każdego algorytmu, jak i jedną dla zagregowanych danych). Wybór padł na medianę z powodu odstających obserwacji, które mogłyby znacząco zaburzać średnią. Wyniki prezentują się następująco:

RandomizedSearchCV

- DecisionTreeClassifier - 20.5
- LogisticRegression - 25
- RandomForestClassifier - 41
- Zagregowane - 25.5

BayesSearchCV

- DecisionTreeClassifier - 30
- LogisticRegression - 12
- RandomForestClassifier - 25.5
- Zagregowane - 20.5

Wyniki wydają się być zgodne z intuicją. Po pierwsze, optymalizacja Bayesowska zbiega średnio szybciej niż losowe przeszukiwanie możliwych przestrzeni parametrów. Po drugie, w przypadku Random Searcha wyniki nie wydają się być mocno zależne od typu optymalizowanego modelu i liczby hiperparametrów. Z kolei dla Bayes Searcha można zauważyć sporą różnicę między modelami drzewa decyzyjnego oraz lasu losowego, gdzie konieczna była optymalizacja 3 hiperparametrów, a regresją logistyczną, gdzie tunowane były jedynie 2 hiperparametry, co również jest zgodne z informacjami znalezionymi w (1).

3.2 Tunowalność poszczególnych algorytmów

Tunowalność algorytmów została zbadana w oparciu o (2), jednakże z powodu znacznie mniejszej liczby zbiorów danych, na których został przeprowadzony eksperyment, metody opisywane w dokumencie zostały nieco zmienione. Z tego względu zdecydowano się obliczyć tunowalność w nastepujący sposób:

$$d_i^{(j)} := R^{(j)}(\theta^{(j)*}) - R^{(j)}(\theta_i^{(j)}) \tag{1}$$

gdzie:
$d_i^{(j)}$ - tunowalność danego algorytmu na j-tym zbiorze danych w i-tej iteracji,
$R^{(j)}(\theta^{(j)*})$ - AUC dla konfiguracji wybranej jako optymalnej (defaultowej) dla j-tego zbioru danych,
$R^{(j)}(\theta_i^{(j)})$ - AUC dla i-tej iteracji (konfiguracji) dla j-tego zbioru danych

Tak wykonane obliczenia zostały zebrane i przedstawione w postaci wykresów typu box plot umieszczonych w dodatku. Analizę tych wykresów podzielić można przeprowadzić na dwa sposoby - dzieląc wyniki ze względu na zbiór danych i algorytm ML albo agregując wyniki dla wszystkich zbiorów danych i dzieląc jedynie ze względu na algorytm ML.

Analiza pozwoliła zaobserwować następujące fakty:

- W przypadku drzew decyzyjnych obydwie metody samplingu punktów pozwoliły w większości przypadków uzyskać pewną siłę predykcyjną defaultowej konfiguracji. Jedynie w przypadku zbioru *PhishingWebsites* większość uzyskanych wartości jest ujemna, co oznacza, że wyznaczony default mógłby zostać poprawiony.

- Dla regresji logistycznej zauważalną moc predykcyjną udało się uzyskać jedynie dla datasetu elevators - średnio blisko 0.1 AUC. Dla pozostałych zbiorów danych w przypadku Random Searcha udało się również uzyskać pewną moc predykcyjną, lecz była ona zauważalnie niższa. Z kolei w przypadku Bayes Searcha ciężko mówić o uzyskaniu jakiejkolwiek mocy predykcyjnej. Dla większości zbiorów danych wybrana konfiguracja dawała zbliżone rezultaty do dowolnej innej, a w przypadku zbioru mozilla4 były one średnio gorsze, co oznacza, że z pewnością dałoby się ją poprawić.

- W przypadku lasu losowego otrzymano zbliżone wyniki dla obydwu metod samplingu, we wszystkich przypadkach ustalony default okazał się mieć pewną moc predykcyjną.

- Analiza drugiego typu pozwoliła stwierdzić, że uzyskane defaulty są zauważalnie mocniejsze dla drzewa decyzyjnego i regresji logistycznej w przypadku użycia optymalizacji bayesowskiej. Dla lasu losowego tunowalność w obydwu przypadkach jest na bardzo zbliżonym poziomie.

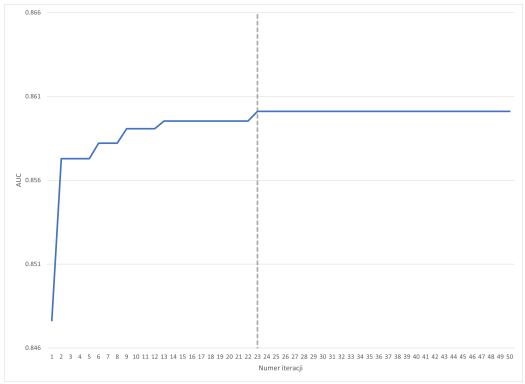
3.3 Wpływ metody samplingu na wyniki

Podsumowując wnioski z poprzednich podrozdziałów, można zauważyć, że wybór metody samplingu potrafi zauważalnie wpływać na tunowalność hiperparametrów dla niektórych modeli uczenia maszynowego. Ponadto metody te stabilizują się w nieco innym tempie, które w przypadku BO może być powiązane z liczbą optymalizowanych hiperparametrów.

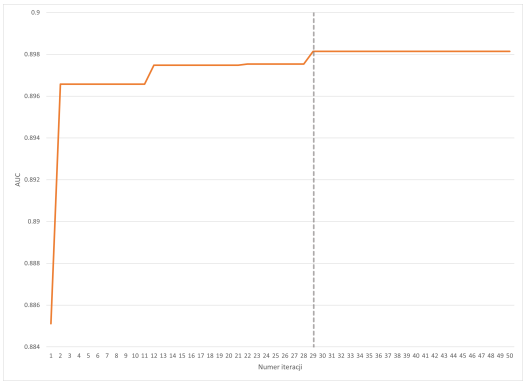
Literatura

[1] Optimization when cost function slow to evaluate. `https://stats.stackexchange.com/questions/193306/optimization-when-cost-function-slow-to-evaluate#193310`.

[2] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. *Journal of Machine Learning Research*, 20:53:1–53:32, 2019.

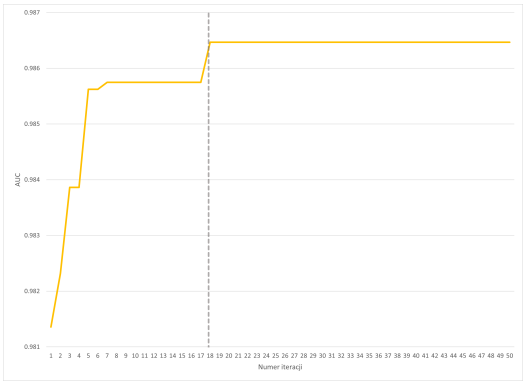


elevators



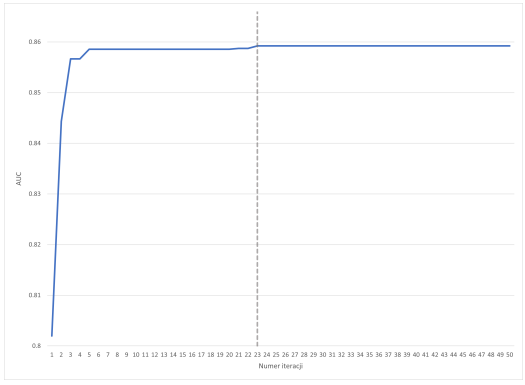
MagicTelescope

mozilla4

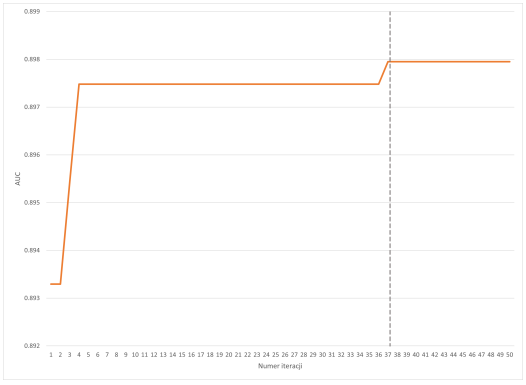


PhishingWebsites

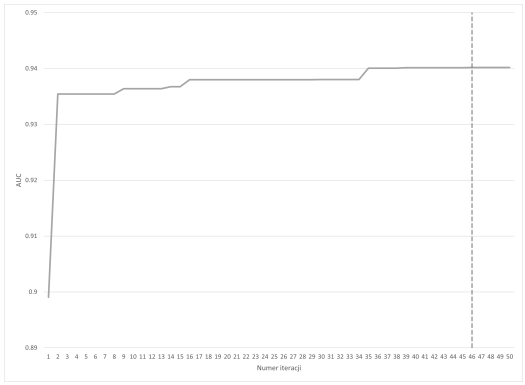
Rysunek 1: *Best so far* dla DecisionTreeClassifier i RandomizedSearchCV



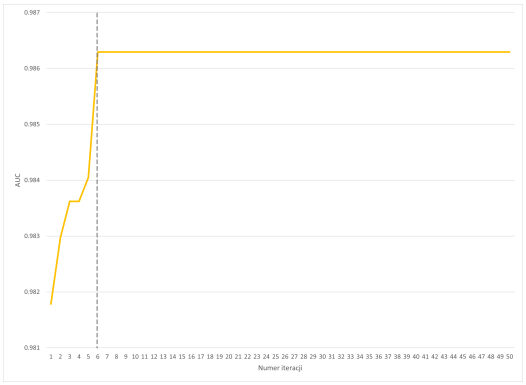
elevators



MagicTelescope

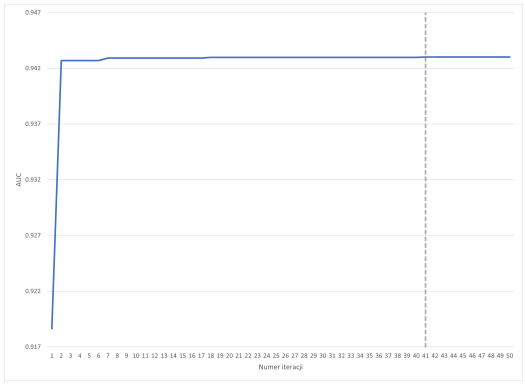


mozilla4

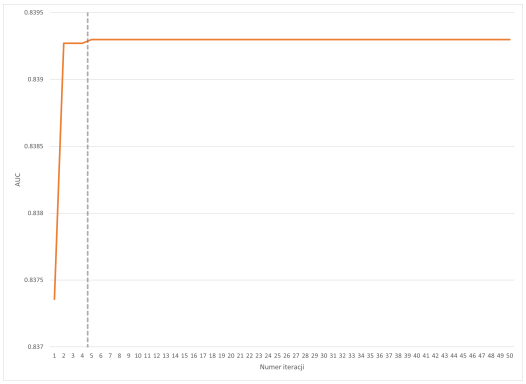


PhishingWebsites

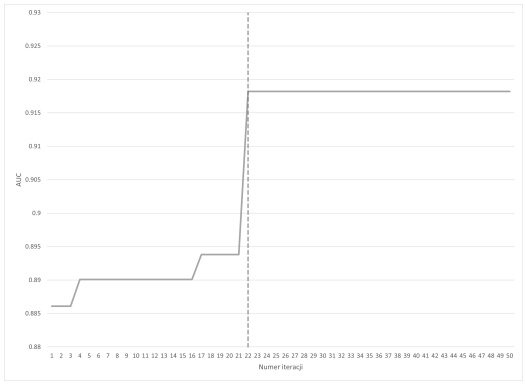
Rysunek 2: *Best so far* dla DecisionTreeClassifier i BayesSearchCV



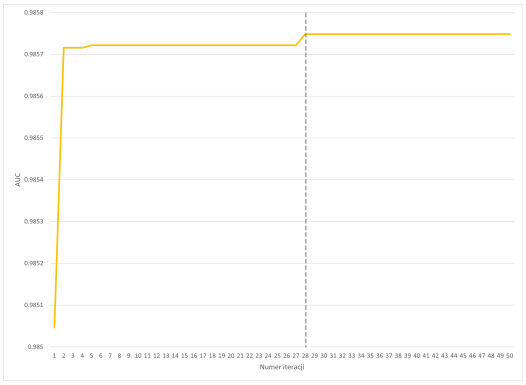
elevators



MagicTelescope

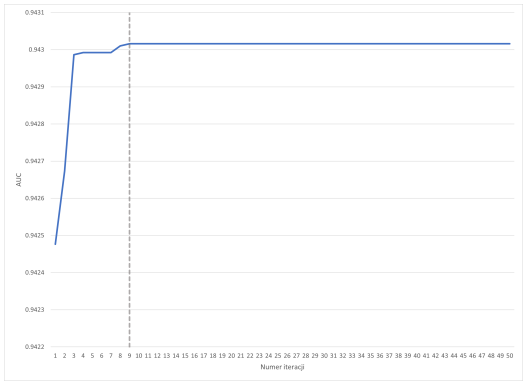


mozilla4

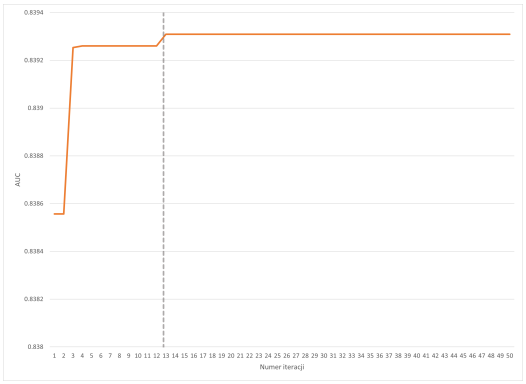


PhishingWebsites

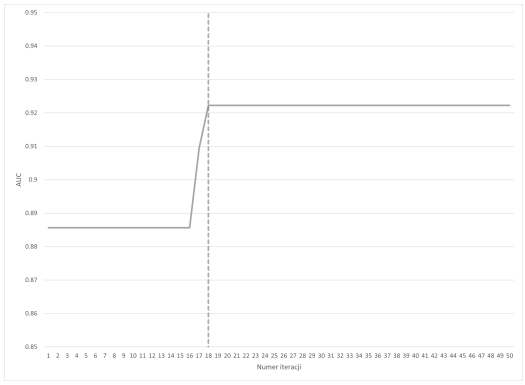
Rysunek 3: *Best so far* dla LogisticRegression i RandomizedSearchCV



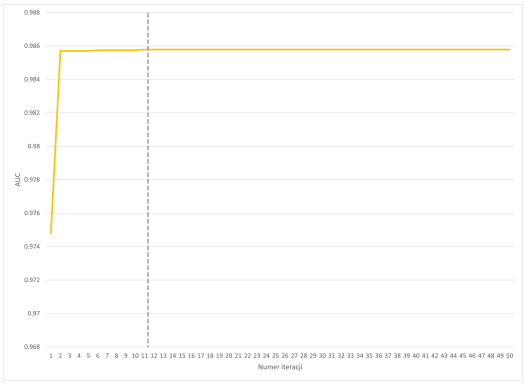
elevators



MagicTelescope

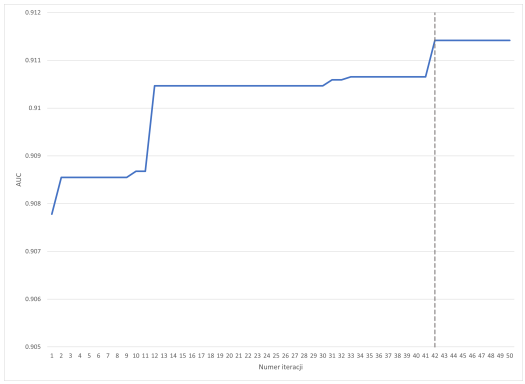


mozilla4

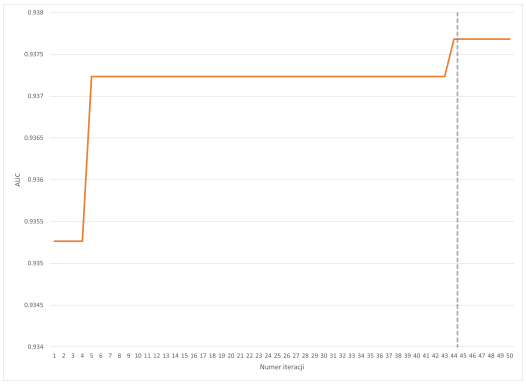


PhishingWebsites

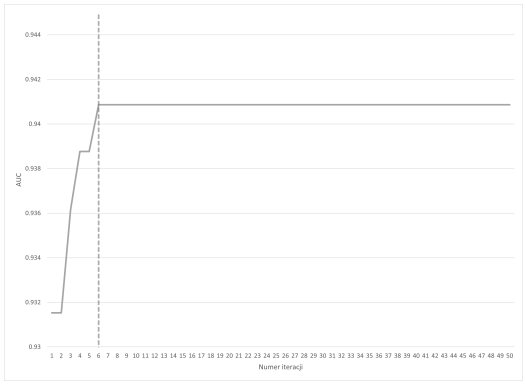
Rysunek 4: *Best so far* dla LogisticRegression i BayesSearchCV



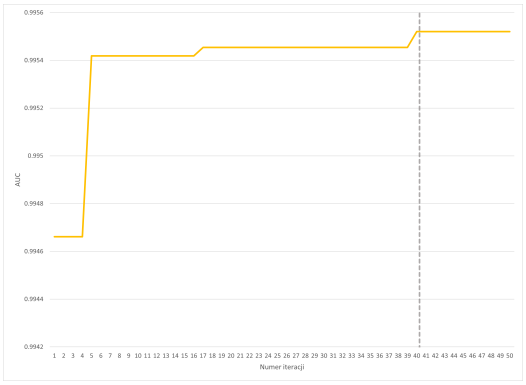
elevators



MagicTelescope

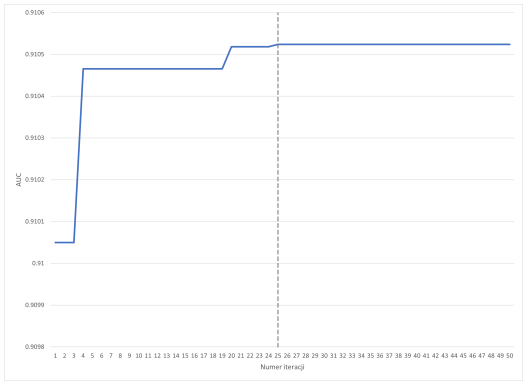


mozilla4

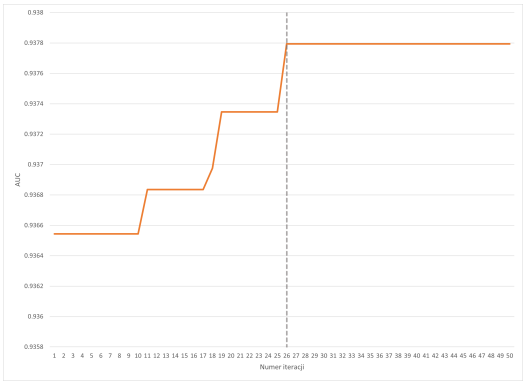


PhishingWebsites

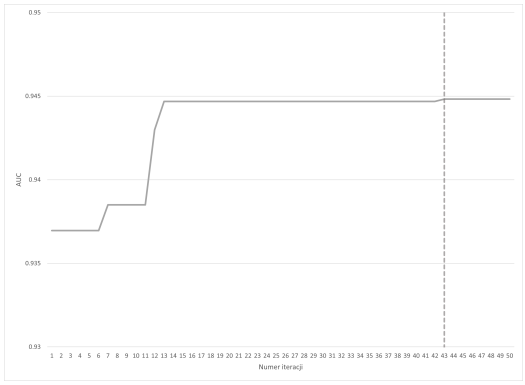
Rysunek 5: *Best so far* dla RandomForestClassifier i RandomizedSearchCV



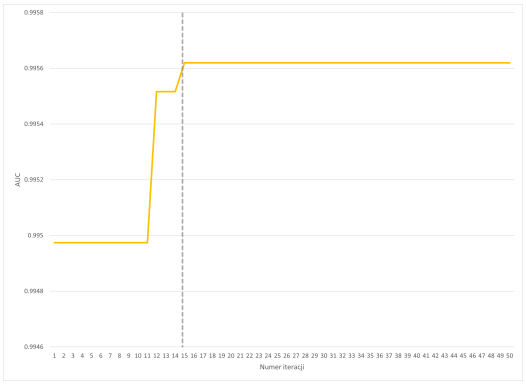
elevators



MagicTelescope

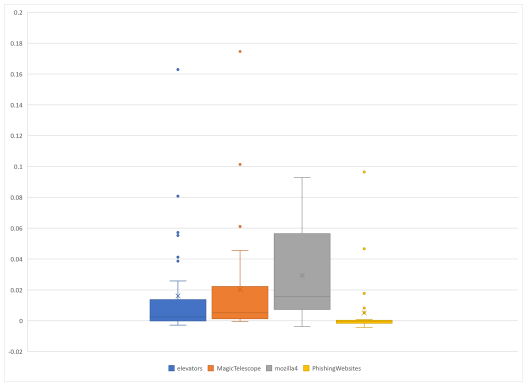


mozilla4

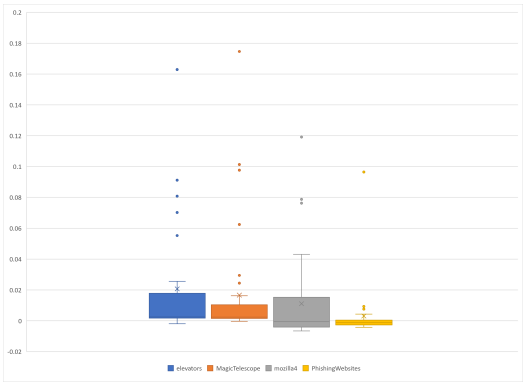


PhishingWebsites

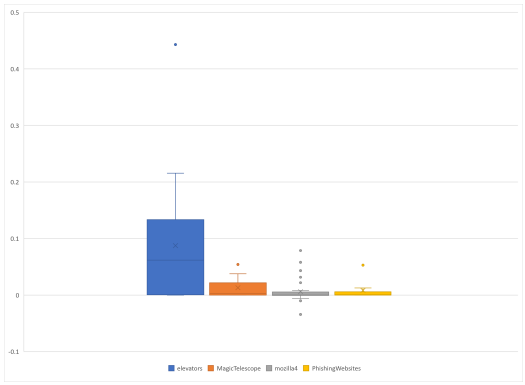
Rysunek 6: *Best so far* dla RandomForestClassifier i BayesSearchCV



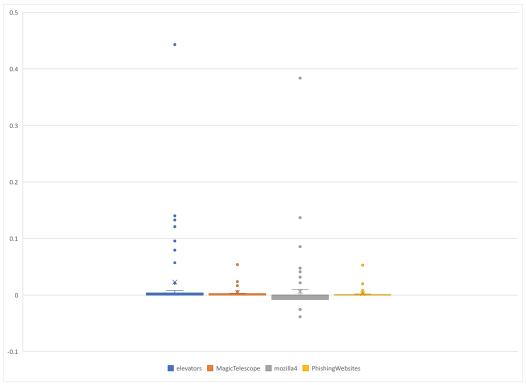
Decision tree, random search



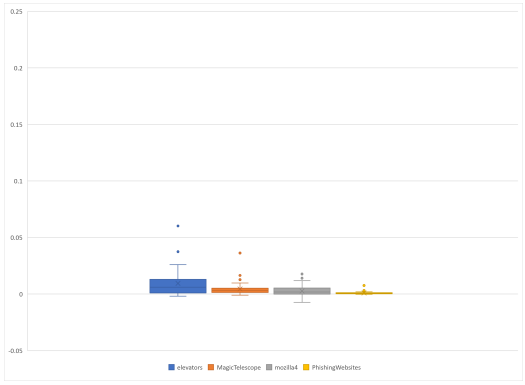
Decision tree, Bayes search



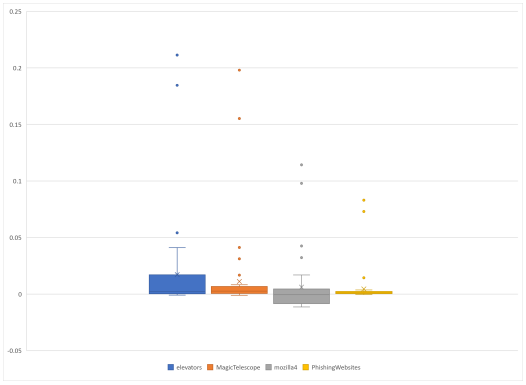
Logistic regression, random search



Logistic regression, Bayes search

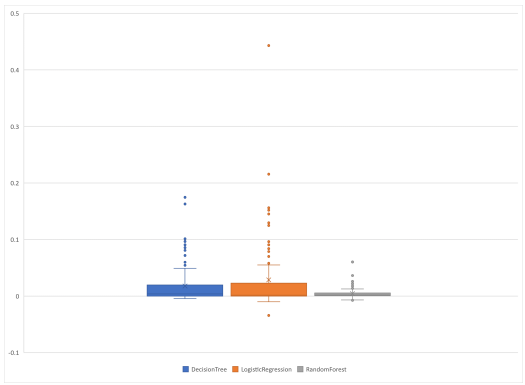


Random forest, random search

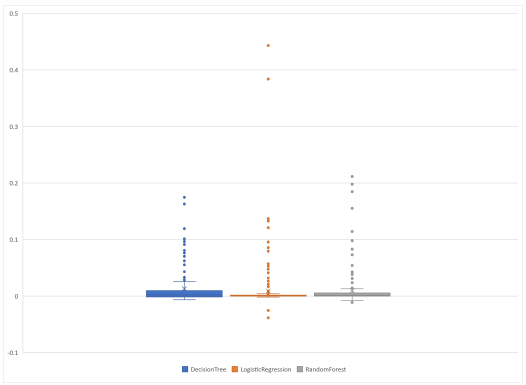


Random forest, Bayes search

Rysunek 7: Tunowalność AUC dla każdego datasetu, algorytmu i metody samplingu



Random search



Bayes search

Rysunek 8: Zbiorcza tunowalność AUC dla algorytmu i metody samplingu