Projekt 1

Automatyczne uczenie maszynowe

Pola Mościcka, Klaudia Kwoka, Maciej Wach

1 Opis

Celem projektu było przeanalizowanie tunowalności wybranych algorytmów uczenia maszynowego. Do badania wybraliśmy algorytmy: Gradient Boosting, KNN oraz SVM. Do optymalizacji hiperparametrów zastosowaliśmy metody Random Search oraz Grid Search. W przypadku metody Random Search na każdym ze zbiorów danych zastosowana została ta sama siatka parametrów. Dla wybranych algorytmów sprawdziliśmy czy występuje zjawisko *bias sampling* i jak wpływa to na tunowalność modelu. Zastosowaliśmy również optymalizację bayesowską dla każdego z algorytmów w celu porównania wybranych hiperparametrów. We wszystkich metodach optymalizacyjnych wykorzystaliśmy kroswalidację 3 krotną oraz metrykę AUC.

Do eksperymentu wybraliśmy cztery zbiory danych zawierające klasy binarne, o różnym stopniu zbalansowania. Dane nie zawierają wartości brakujących. Dwa z nich pobraliśmy ze strony *kaggle*, a pozostałe z biblioteki *openml*. Charakterystykę wybranych zbiorów obrazuje Tabela 1.

Zbiór danych	l. obserwacji	l. zmiennych	stosunek klas 1:0	źródło
banana	8000	7	50:50	kaggle
credit	1548	18	11:89	kaggle
wind	6574	14	53:47	openml
elevators	5000	18	69:31	openml

Tabela 1: Charakterystyka zbiorów danych

Do stworzenia wszystkich modeli oraz optymalizacji hiperparametrów wykorzystaliśmy implementacje zawarte w bibliotece *scikit-learn*. Do wstępnego przetworzenia zbiorów danych wykorzystany został Pipeline. Dane numeryczne zostały przeskalowane za pomocą MinMaxScaler do przedziału (0,1). Dla danych kategorycznych zastosowano kodowanie za pomocą OneHotEncoder z parametrem *handle_unknown = 'ignore'*. Przeprowadzona została również selekcja zmiennych za pomocą SelectFromModel z modelu RandomForestClassifier.

Dla poszczególnych algorytmów wybraliśmy kilka hiperparametrów, których wartości optymalizowaliśmy, aby efektywność modeli była jak największa. Zakresy parametrów zostały wybrane na podstawie artykułu *Tunability: Importance of Hyperparameters of Machine Learning Algorithms* i nieznacznie zmienione w wybranych przypadkach. Parametry oraz przyjęte zakresy przedstawia Tabela 2.

2 Gradient Boosting

W pierwszym modelu optymalizowaliśmy 4 hiperparametry: *n_estimators*, *max_depth*, *learning rate* oraz *subsample* (zob. tab. 2). Dwa parametry przyjmują wartości rzeczywiste, a dwa całkowite. Z tego względu oraz z powodu relatywnie dużej ilości hiperparametrów do optymalizacji zdecydowaliśmy się na zastosowanie techniki losowania punktów Random Search. W wyniku przeprowadzonej optymalizacji uzyskaliśmy kombinację prarametrów (w przypadku zmiennych ciągłych podane z dokładnością do dwóch zmiennych po przecinku) $n\_estimators = 477$, $max\_depth = 9$, $learning\ rate = 0.18$, $subsample = 0.46$. Najlepsze parametry uzyskane dla poszczególnych zbiorów danych przestawia Tabela 4.

Algorytm	Hiperparametr	Typ	Lower	Upper
Gradient Boosting	n_estimators	int	100	500
	learning_rate	float	0.001	1
	subsample	float	0.1	1
	max_depth	int	4	10
KNN	n_neighbors	int	1	45
	p	float	1	2
SVM	C	float	0.001	1000
	gamma	float	0.001	1000

Tabela 2: Zakresy hiperparametrów

2.1 Analiza wyników

1. **Tunowalność**
   Rysunek 1 przedstawia tunowalność każdego z analizowanych algorytmów na wybranych zbiorach danych. Model okazał się być najbardziej tunowalny na zbiorze *credit*, na pozostałych zbiorach danych tunowalność osiągnęła niskie wyniki, nie przekraczające 0.01. Porównując do innych algorytmów, Gradient Boosting okazał się być najmniej tunowalny ze wszystkich, osiągając najniższą tunowalność na 3 spośród 4 zbiorów danych.

2. **Wydajność modelu**
   Rysunek 2 przedstawia rozkład wyników uzyskanych podczas kroswalidacji, dla każdego ze zbiorów danych. Największe wyniki model osiąga na zbiorze *banana*, z medianą na poziomie 0.95. Najniższe wartości metryki AUC uzyskujemy dla zbioru danych *credit*, który ze wszystkich ma najmniej obserwacji oraz dodatkową trudnością jest jego niezbalansowanie klas. Zarówno szerokie przedziały międzykwartylowe, jak i rozpiętość wartości od minimum do maksimum obejmują dużą część zakresu, co wskazuje na znaczne zróżnicowanie wyników miary AUC w zależności od podziału danych uzyskanego w kroswalidacji. Świadczy to o dużej zmienności wydajności modelu w zależności od zbioru treningowego, co może wskazywać na wrażliwość modelu na zmiany w danych. Spośród wszystkich analizowanych modeli, model *Gradient Boosting* wydaje się najbardziej podatny na zjawisko *bias sampling* w wynikach miary AUC.

3 SVM

W modelu SVM używaliśmy domyślnego jądra *rbf*. Ze względu na to optymalizowane były dwa parametry: C oraz *gamma*. Parametry te uznaliśmy za najbardziej kluczowe, gdyż odpowiadają za regularyzację oraz wpływ pojedynczego punktu przy trenowaniu modelu.

Zakresy przyjęte dla obu parametrów przedstawia Tabela 2. Na tym modelu porównaliśmy czy technika losowania punktów ma wpływ na wyniki tunowalności algorytmów oraz na ogólną wydajność modelu. Zbudowaliśmy dwa modele SVM, jeden oparty na technice Grid Search, a drugi na technice Random Search. Parametry wyznaczone w wyniku optymalizacji dla obu technik przedstawia Tabela 3. Najlepsze parametry uzyskane dla poszczególnych zbiorów danych przestawiają Tabela 6 i Tabela 7.

	C	*gamma*
GridSearch	340.16	30.78
RandomSearch	828.94	4.7

Tabela 3: Optymalne wartości parametrów uzyskane dla metod GridSearch i RandomSearch

3.1 Analiza wyników

1. **Tunowalność** (por. 1)
   Tak jak dla pozostałych algorytmów, na zbiorze *credit* model ten jest najbardziej tunowalny. Najgniższy wynik odnotowany został na zbiorze *elevators* w obu przypadkach. Na zbiorach danych *credit* i *banana* zaobserwowano wyższą tunowalność dla modelu opartego na technice Random Search, natomiast na pozostałych zbiorach sytuacja była odwrotna. W związku z tym nie można uznać żadnej z metod optymalizacji hiperparametrów za wyraźnie bardziej skuteczną.

2. **Wydajność modelu i porównanie technik losowania punktów**
   Rysunek 3 przedstawia rozkład miary AUC na zbiorach danych w zależności od sposobu doboru punktów do siatki hiperparametrów. Można zauważyć, że dla każdego zbioru danych wykresy boxplot dla metod Grid Search i Random Search różnią się w niewielkim stopniu, co sugeruje, iż sposób losowania siatki hiperparametrów nie wpływa istotnie na wydajność algorytmu. Z tego powodu korzystniej jest wybrać metodę Random Search, ponieważ jest ona szybsza, w szczególności dla większej siatki parametrów. Dla zbiorów *banana* oraz *wind* przedziały międzykwartylowe są najmniejsze (jednak nieco szersze niż dla algorytmu KNN (por. rys. 5)), co wskazuje na niewielkie zróżnicowanie wyników miary AUC w zależności od podziału danych w kroswalidacji. Na tych zbiorach algorytm uzyskał najwyższe wyniki. Natomiast dla zbiorów *credit* oraz *elevators* wyniki są bardziej zróżnicowane.
   Rysunek 4 przedstawia rozkład wyników tunowalności na podziałach zbioru treningowego z kroswalidacji mierzonej metryką AUC. Na zbiorach *banana* oraz *credit* uzyskane zostały największe przedziały międzykwartylowe, co wskazuje na dużą zmienność osiąganych wyników w zależności od próbki treningowej. Zbiory *wind* oraz *elevators* osiągają tunowalność nie większą niż odpowienio 0.03 oraz 0.01. Stąd nie obserwujemy dla nich znacznego wpływu na wynik tunowalności w zależności od podziału zbioru treningowego. Zmienność tunowalności algorytmu w zależności od doboru danych treningowych może wskazywać na obecność zjawiska *bias sampling*, które w wpływa na miarę AUC i wydajność modelu.

4 K-nearest neighbours

4.1 Opis

W tym algorytmie parametry, które optymalizowaliśmy to *n_neighbors* oraz parametr p. Zakres został lekko zwiększony w porównaniu do artykułu, na którym bazowaliśmy ze względu na to, że zajmujemy się zbiorami danych o dość dużej liczbie obserwacji, stąd optymalna liczba punktów wykorzystywanych do klasyfikacji może okazać się duża. Metryka stosowana w algorytmie to domyślna dla tego algorytmu metryka Minkowskiego. Rozważyliśmy parametr p o wartościach $1, 1.25, 1.5$ oraz 2. Zbiory danych, na których przeprowadziliśmy eksperymenty mają stosunkowo dużą liczbę zmiennych, przez co może występować zjawisko klątwy wymiarowości. Wówczas niewskazane jest używanie metryk L_p, dla $p > 2$ (zob. [1]). Dla tego modelu użyliśmy techniki optymalizacji parametrów Grid Search. Optymalna kombinacja parametrów uzyskana w eksperymencie dla tego modelu to $n\_neighbors = 7$, $p = 1.25$. Najlepsze parametry uzyskane dla poszczególnych zbiorów danych przestawia Tabela 9.

4.2 Analiza wyników

1. **Tunowalność** (por. 1)
   Podobnie jak w przypadku innych algorytmów, na zbiorze *credit* model KNN wykazuje najwyższą tunowalność, jednak jego wynik nie odbiega znacząco od wyników uzyskanych na innych zbiorach. Ze wszystkich algorytmów, ten dawał najstabilniejsze wyniki tunowalności. Może to być spowodowane tym, że ciężko jest określić ogólnie 'dobrą' wartość hiperparametru liczby sąsiadów, która była by odpowiednia dla każdego zbioru. Metoda KNN uzyskała najlepszą tunowalność na zbiorach *banana*, *wind* i *elevators*, podczas gdy reszta algorytmów nie wyróżniała się wysoką tunowalnością dla tych zbiorów.

2. **Wydajność modelu**
   Rysunek 5 obrazuje, że na większości zbiorów algorytm dokonuje skutecznych podziałów, z wynikami

metryki AUC na poziomie bliskim 0.9. Jednak dla zbioru *credit* wyniki klasyfikacji wykazują przeciętną efektywność w porównaniu do wcześniej analizowanych metod. Powodem tak niskich wyników na tym zbiorze danych może być niezbalanowanie klas. Porównując efektywność modelu z wynikami tunowalności (por. rys. 1) zauważamy, że największa tunowalność uzyskana jest na zbiorze *credit*, dla którego uzyskane wyniki wydajności są najniższe. Dla zbiorów *banana*, *wind* oraz *elevators* rozstęp międzykwartylowy wynosi około 0.02 co wskazuje na niewielkie zróżnicowanie wyników miary AUC w zależności od dokonanego podziału danych w kroswalidacji. Sugeruje to stosunkowo małą zmienność wydajności modeli trenowanych na różnych zbiorach treningowych, co może świadczyć o braku zjawiska *sampling bias* w wynikach miary AUC.

5 Bayes Optimization

Do każdego z algorytmów dodatkowo zastosowaliśmy optymalizację bayesowską. Liczba iteracji dla każdego modelu na każdym zbiorze danych wynosiła 150. Najlepsze parametry dla poszczególnych zbiorów danych wraz z wynikami są przedstawione w Tabelach 5, 8 i 10.

Wykres 6 pokazuje średnią uzyskaną wartość AUC w zależności od iteracji dla poszczególnych zbiorów danych dla algorytmu KNN. Można zauważyć, że dla trzech zbiorów danych: *banana*, *wind* oraz *elevators*, średni wynik walidacji mierzony metryką AUC przekracza wartość 0.8, a po około 50 iteracjach rezultaty zaczynają się stabilizować. Pomimo że najlepsze wyniki osiągane są dla zbioru *banana*, średnia wartość AUC nawet po 100 iteracjach wykazuje pewne wahania. W przypadku zbiorów *wind* i *elevators* wyniki od pewnego momentu stabilizują się do tego stopnia, że przebieg krzywej zaczyna przypominać prostą linię. Najmniej efektywnym i wyraźnie niestabilnym zbiorem danych jest *credits*.

Rysunek 7 przedstawia jak zmienia się najlepszy wynik w zależności od iteracji dla optymalizacji bayesowskiej i metody Random Search dla modelu Gradient Boosting kna każdym ze zbiorów danych. Zauważalne jest, że wynik stabilizował się szybciej dla metody Random Search, jednak w każdym przypadku optymalizacja bayesowska uzyskiwała lepszy wynik końcowy. Przeprowadziliśmy również analogiczną analizę dla modelu SVM (zob. 8). W przypadku zbiorów *banana* oraz *credit*, zaobserwowano podobną tendencję wzrostu jakości oraz stabilizacji wyników dla obu metod. Natomiast dla dwóch pozostałych zbiorów danych, metoda optymalizacji bayesowskiej osiągała lepsze wyniki szybciej oraz szybciej się stabilizowała, choć różnica ta nie była znacząca. Na podstawie analizy czterech wykresów dla algorytmu SVM można stwierdzić, że metoda optymalizacji bayesowskiej daje lepsze rezultaty, co czyni ją bardziej efektywną w doborze parametrów. Niemniej jednak należy zauważyć, że czas obliczeń dla optymalizacji bayesowskiej był trzykrotnie dłuższy niż w przypadku Random Search przy tej samej liczbie iteracji, co stanowi istotną wadę tej metody.

6 Końcowe wnioski

Podsumowując, algorytm GradientBoosting okazał się najmniej tunowalny, zwłaszcza na zbiorach *banana* i *wind*. Z rozważanych modeli algorytm KNN wyróżniał się najbardziej jednolitą tunowalnością, osiągając wysokie wartości niezależnie od wybranego zbioru danych. SVM wykazał dużą zmienność, zwłaszcza na zbiorze *credit*. Najwyższą wydajność uzyskano na zbiorze *banana*, a najniższą na *credit*, gdzie niezbalansowanie klas miało wpływ na wyniki klasyfikacji na zbiorze treningowym. W porównaniu metod wyboru punktów w siatce hiperparametrów, Random Search okazał się równie efektywny jak Grid Search, ale zdecydowanie szybszy. Tunowalność algorytmu SVM zmienia się w zależności od wyboru zbioru testowego w procesie kroswalidacji, jednak nie obserwujemy tego zjawiska w równym stopniu na każdym ze zbiorów danych. Projekt wskazuje, że wybór algorytmu i metody optymalizacji ma kluczowy wpływ na tunowalność i wydajność modeli, a także zależy od charakterystyki danych.

Zbiór danych	Learning rate	Max_depth	n_estimator	Subsample	Test score	CV score
banana	0.065	6	327	0.879	0.972	0.986
credit	0.117	9	477	0.457	0.721	0.745
wind	0.017	4	220	0.461	0.823	0.919
elevators	0.072	5	215	0.431	0.825	0.919

Tabela 4: Wyniki optymalizacji Gradient Boostingu metodą Random Search

Zbiór danych	Learning rate	Max_depth	n_estimator	Subsample	Test score	CV score
banana	0.239	10	100	1.000	0.926	0.990
credit	0.204	10	500	0.489	0.736	0.752
wind	0.001	7	500	0.283	0.824	0.920
elevators	0.310	4	500	0.855	0.827	0.923

Tabela 5: Wyniki optymalizacji Gradient Boostingu metodą bayesowską

Zbiór danych	C	Gamma	Test score	CV score
banana	340.16	30.78	0.906	0.968
credit	582.95	344.57	0.683	0.717
wind	610.20	30.78	0.806	0.876
elevators	582.95	30.78	0.795	0.874

Tabela 6: Wyniki optymalizacji SVM metodą Random Search

Zbiór danych	C	Gamma	Test score	CV score
banana	729.99	171.63	0.974	0.975
credit	64.15	692.47	0.569	0.718
wind	828.94	4.70	0.822	0.904
elevators	828.94	4.70	0.774	0.901

Tabela 7: Wyniki optymalizacji SVM metodą Grid Search

Zbiór danych	C	Gamma	Test score	CV score
banana	2.94	3.57	0.973	0.988
credit	642.86	998.51	0.694	0.711
wind	700.74	0.001	0.825	0.921
elevators	541.35	0.027	0.839	0.939

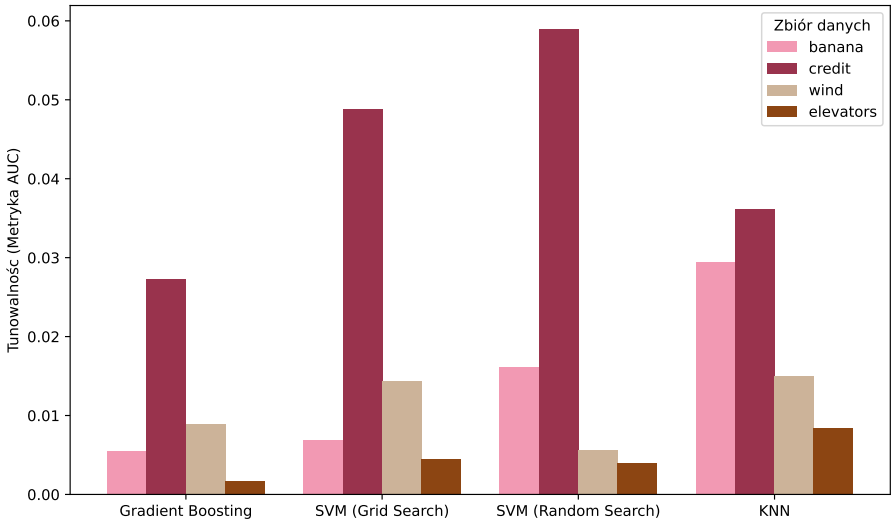
Tabela 8: Wyniki optymalizacji SVM metodą bayesowską

Zbiór danych	n_neighbors	p	Test score	CV score
banana	37	1.25	0.976	0.989
credit	2	1	0.569	0.678
wind	41	1.25	0.823	0.921
elevators	20	1	0.773	0.866

Tabela 9: Wyniki optymalizacji KNN metodą Grid Search

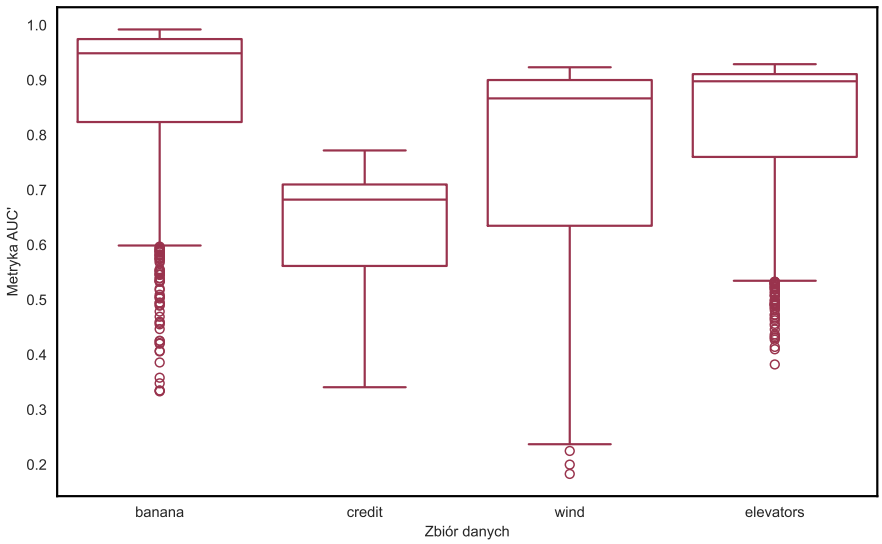
Zbiór danych	n_neighbors	p	Test score	CV score
banana	41	1.998	0.972	0.992
credit	1	1	0.655	0.694
wind	43	1.997	0.827	0.922
elevators	18	1.002	0.775	0.868

Tabela 10: Wyniki optymalizacji KNN metodą bayesowską

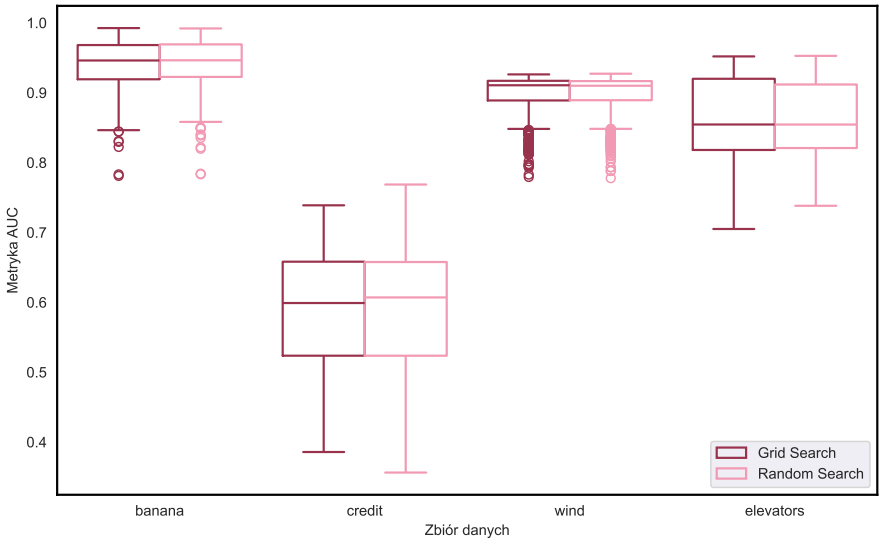


Rysunek 1: Porównanie tunowalności algorytmów na wybranych zbiorach danych z uwzględnieniem podziału na technikę optymalizacji hiperparametrów dla algorytmu SVM

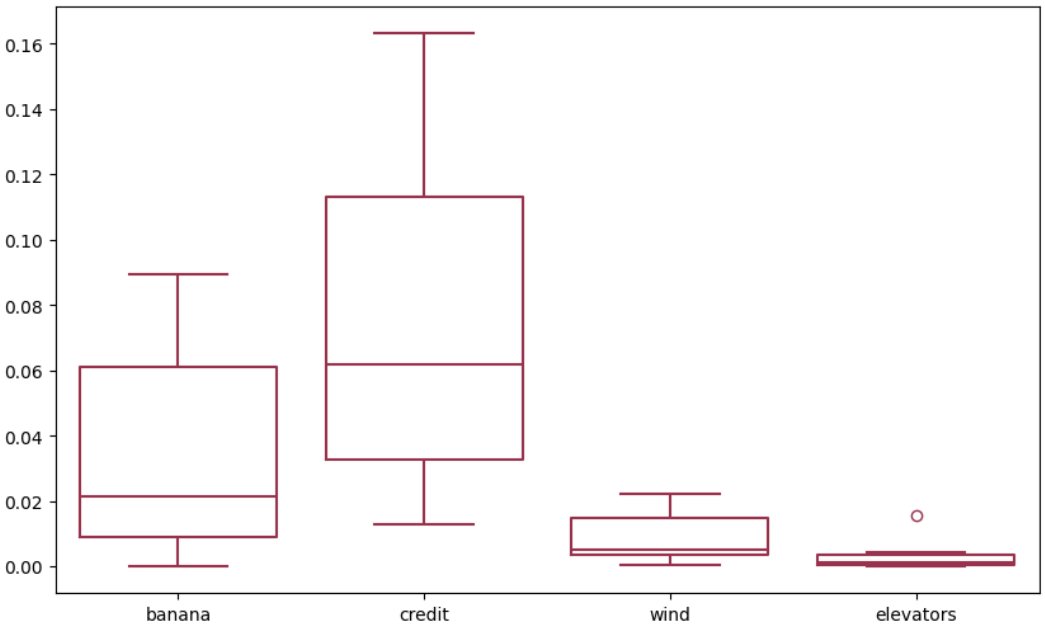
[1] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim, *On the Surprising Behavior of Distance Metrics in High Dimensional Space*, IBM T. J. Watson Research Center Yorktown Heights, NY 10598, USA. Institute of Computer Science, University of Halle Kurt-Mothes-Str.1, 06120 Halle (Saale), Germany



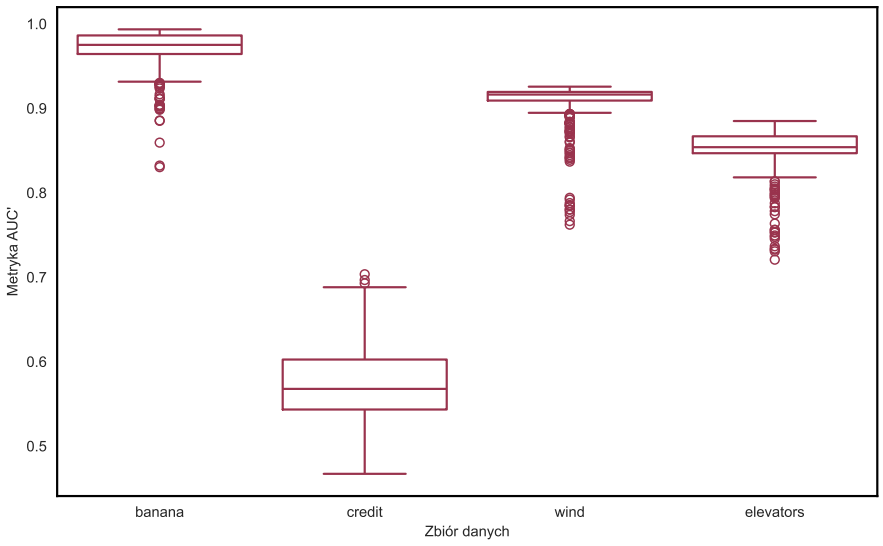
Rysunek 2: Rozkład wartości metryki AUC uzyskanej podczas kroswalidacji dla algorytmu Gradient boosting na wybranych zbiorach danych



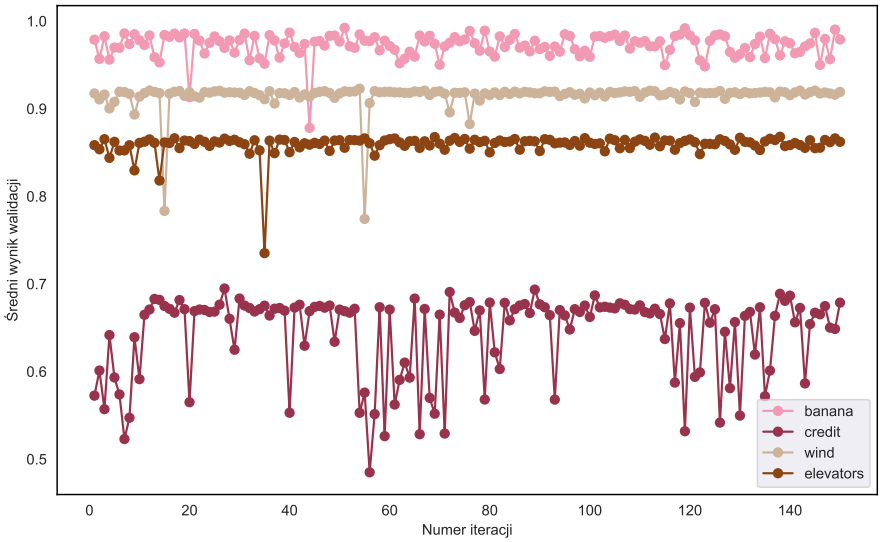
Rysunek 3: Porównanie rozkładu wartości metryki AUC uzyskanej podczas kroswalidacji dla algorytmu SVM na wybranych zbiorach danych w podziale na technikę optymalizacji hiperparametrów



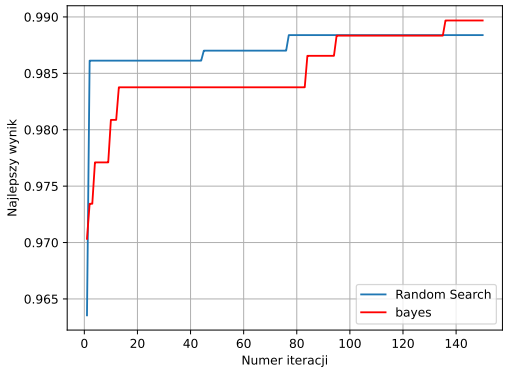
Rysunek 4: Tunowalność dla różnych podziałów zbioru treningowego podczas kroswalidacji dla modelu SVM (Grid Search)



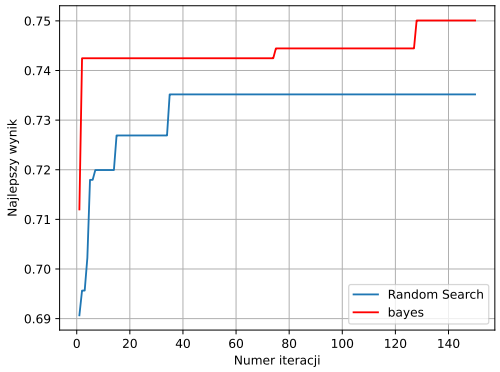
Rysunek 5: Rozkład wartości metryki AUC uzyskanej podczas kroswalidacji dla algorytmu KNN na wybranych zbiorach danych



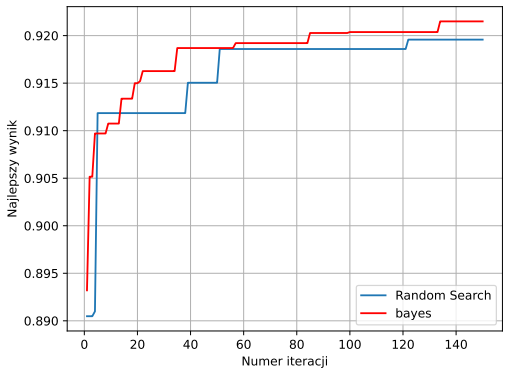
Rysunek 6: Średni wynik walidacji metryką AUC na wszystkich zbiorach danych w kolejnych iteracjach optymalizacji bayesowskiej dla algorytmu KNN



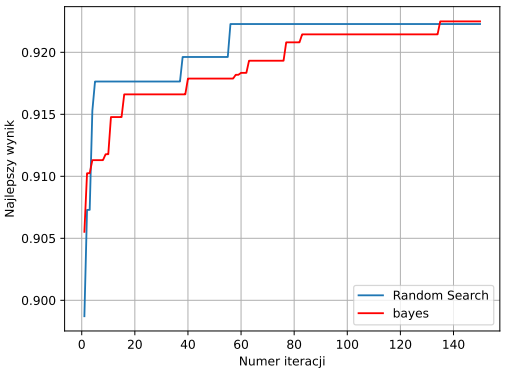
(a) zbiór 'banana'



(b) zbiór 'credit'

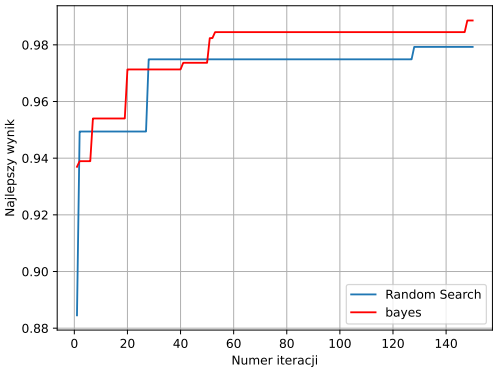


(c) zbiór 'wind'

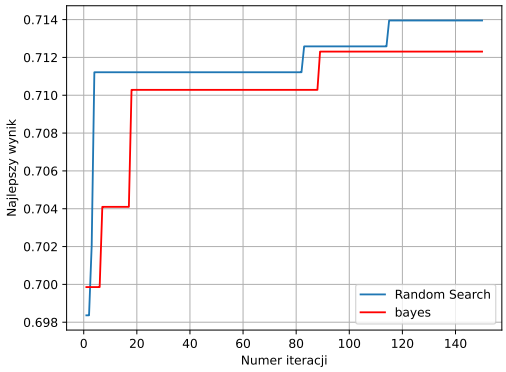


(d) zbiór 'elevators'

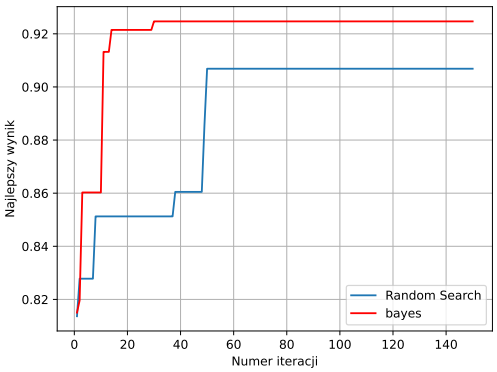
Rysunek 7: AUC w zależności od iteracji metody optymalizacji bayesowskiej oraz Random Search dla algorytmu Gradient Boosting



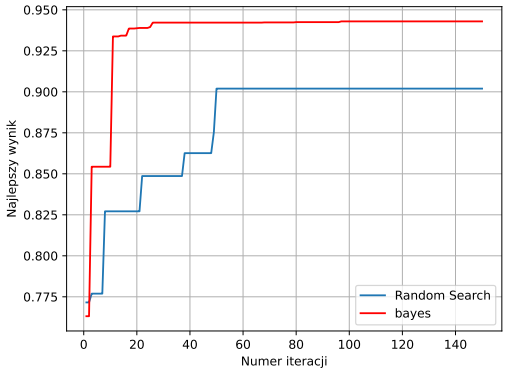
(a) zbiór 'banana'



(b) zbiór 'credit'



(c) zbiór 'wind'



(d) zbiór 'elevators'

Rysunek 8: AUC w zależności od iteracji metody optymalizacji bayesowskiej oraz Random Search dla algorytmu SVM