Automatyczne uczenie maszynowe - projekt nr 1

Iza Danielewska, Patrycja Żak

Spis treści

1 Wprowadzenie

1.1 Cel projektu

Celem projektu jest analiza tunowalności hiperparametrów dla trzech wybranych algorytmów uczenia maszynowego na co najmniej czterech zbiorach danych.

1.2 Zbiory danych

Do projektu wybrano cztery zestawy danych do klasyfikacji binarnej ze strony `https://www.openml.org/`, która jest otwartą platformą do udostępniania zbiorów danych. Poza doborem danych w podobnej tematyce, skupiono się również aby liczba rekordów oraz zmiennych była w mieszanych konfiguracjach.
Wyselekcjonowano następujące zbiory danych:

- **christine** (ID: 41142) - 30 zestawów danych z różnorodnymi typami danych i dystrybucjami, stworzony przez autorów do problemu automatycznego uczenia maszynowego;
- **KDDCup09_upselling** (ID: 44035) - baza danych marketingowych firmy Orange, mających na celu przewidywanie skłonności klientów do zmiany dostawcy, zakupu nowych produktów lub usług, oraz wykrywania potrzebnych ulepszeń, co stanowi kluczowy element współczesnych strategii CRM;
- **Click_prediction_small** (ID: 1220) - dane dotyczą reklam wyświetlanych obok wyników wyszukiwania w wyszukiwarce oraz tego, czy ludzie kliknęli w te reklamy;
- **SantanderCustomerSatisfaction** (ID: 45566) - zestaw danych z portugalskiej kampanii marketingowej, którego zmienną docelową jest informacja, czy klient zasubskrybował lokatę terminową.

Poniżej przedstawiono tabelę z ilością rekordów w danych źrodłowych oraz w projekcie, a także liczbę zmiennych:

Tabela 1: Tabela z liczbą rekordów oraz zmiennych w danych

Nazwa zbioru danych	Liczba rekordów	Liczba zmiennych	Użyta liczba rekordów
christine	5418	1637	5418
KDDCup09_upselling	5032	46	5032
Click_prediction_small	39948	10	39948
SantanderCustomerSatisfaction	200000	201	10000

W przypadku zbioru **SantanderCustomerSatisfaction** zdecydowałyśmy się na zmniejszenie liczby wierszy poprzez wylosowanie 10000 spośród dostępnych. Miało to na celu zmniejszenie czasu potrzebnego do budowania modeli.

1.3 Wybrane algorytmy

Projekt został stworzony przy użyciu języka Python w wersji 3.10.
Wykorzystano trzy algorytmy klasyfikacji binarnej oraz zastosowano po dwie techniki losowania punktów do optymalizacji hiperparametrów, co przedstawia tabela poniżej.

Tabela 2: Tabela z użytymi strategiami optymalizacji hiperparametrów do modeli uczenia maszynowego

Algorytm	Technika optymalizacji 1	Technika optymalizacji 2
Random Forest Classifier	Random Search	Bayes Optimization
KNN	Random Search	Bayes Optimization
CatBoost	Random Search	Bayes Optimization

Dla każdego zestawu danych przygotowano taką samą siatkę parametrów. Poniżej w Tabeli 3 znajduje się wykaz użytych parametrów razem z zakresem wykorzystanych wartości.
Parametry te zostały wyznaczone na podstawie artykułu *Tunability: Importance of Hyperparameters of Machine Learning Algorithms*. Poza nimi dobrano kilka ciekawych (według autorek) parametrów, które mogłyby wpłynąć na dokładność dopasowania modelu.

Tabela 3: Tabela z hiperparametrami użytymi do badania ich wyboru do modeli uczenia maszynowego

Algorytm	Nazwa parametru	Typ	Min.	Max.	Liczba wartości	Lista wartości
Random Forest Classifier	n_estimators	int	1	2000	20	-
	max_depth	int	1	30	15	-
	min_samples_split	int	2	60	20	-
	min_samples_leaf	int	1	60	20	-
	max_samples	float	0	1	10	-
	ccp_alpha	float	0	1	15	-
KNN	n_neighbors	int	1	30	30	-
	weights	string	-	-	2	'uniform', 'distance'
	P	float	1	2	2	-
CatBoost	iterations	int	100	300	3	-
	learning_rate	float	-	-	3	0.01, 0.1, 0.5
	bagging_temperature	float	-	-	4	0.1, 0.5, 1.0, 1.5
	depth	int	4	10	4	-
	l2_leaf_reg	int	1	9	5	-
	colsample_bylevel	float	-	-	3	0.1, 0.5, 1.0

2 Eksperyment

Każda z metod została przygotowana w 50 iteracjach, poza zbiorem *christine* przy użyciu *CatBoost* oraz *Bayes Optimazation*, do którego użyto tylko 30 iteracji. Każda metoda miała także różną liczbę pętli kroswalidacyjnych (od 3 do 7). Wynika to z charakterystyki zbioru oraz wielogodzinnej generacji przy podanych parametrach.

2.1 Badanie stabilności wyników

Początkowym etapem było badanie stabilności wyników optymalizacji. Zbadano oraz porównano średnią wartość wyników testowych przy użyciu metryki *ACC*.
W pierwszym kroku dokonano analizy algorytmicznej, która zakładała stabilizację wyników w różnicy nie większej niż $1e-6$ oraz przy minimalnej liczbie trzech takich przypadków. Niestety, mimo zmniejszaniu ustanowionego progu, nie otrzymano na koniec żadnego pozytywnego rezultatu.
W drugim kroku sprawdzono zmianę omawianych wartości poprzez wizualizację dla optymalizacji *Bayes Optimazation*.
Jak wynika z Rysunków 4, 8, 12, 16, optymalizacje wykazują niestabilność. Na podstawie wykonanego eksperymentu, nie da się określić numeru iteracji dla każdego z algorytmów, kiedy wyniki zaczną być w wybranym przybliżeniu niezmienne. Rezultatem jest brak odpowiedzi na problem wskazania odcięcia, po którym wyniki będą nieznacznie lepsze. Powodem mogą być dobrane hiperparametry, ale przede wszystkim własności wybranych danych. Mimo reprezentowalnych próbek, zbiory wymagają większej precyzji oraz liczniejszych badań. Budzącym wątpliwości obiektem jest wynik dla zbioru *Santander* z wykrozystaniem *Random Forest Classifier*. Niezależnie od stosowanych hiperparametrów, rezultat jest stały. Zbyt mała ilość testowanych wartości może sprawiać, że nie dokonano poprawy jakości wyników, które na starcie były na bardzo dobrym poziomie.

2.2 Analiza tunowalności modeli

Kolejnym celem prowadzonego badania było sprawdzenie tunowalności używanych modeli. Celem optymalizacji hiperparametrów jest również stworzenie zestawu uniwersalnych parametrów. Podnosi to właściwości adaptacyjne modelu, co sprawia, że dla różnych danych można otrzymywać wciąż zadowalające wyniki. Przykładem prowadzenia takich analiz jest organizacja *ChaLearn Automatic Machine Learning Challenge (AutoML)*, która za pomocą stworzonego zbioru *christine*, starała się zbudować skrupulatny zestaw hiperparametrów pod względem ich uniwersalności. Ważną uwagą jest fakt, że charakterystyka danych oraz wybrany model wpływają na łatwość owej optymalizacji.
Dla każdego wykorzystanego modelu oraz metody optymalizacji w tej pracy, stworzono wykresy typu *boxplot* (Rysunek 20, 24), które wskazują na jakość tunowalności poprzez różnicę między najlepszym wynikiem, a pozostałymi rezultatami, stworzonymi poprzez średnią dla każdej iteracji.
Jak przedstawiają wykresy, tunowalność algorytmów zależy w dużej mierze od wyboru metody optymalizacji. Można

zauważyć, że wybrane algorytmu wykazują tendencję do tunowalności poza algorytmem *Random Forest Classifier*. W tym szczególnym przypadku mamy bardzo dużą rozbieżność w wartości dokładności dopasowania modelu.
W przypadku metody optymalizacji bayesowskiej algorytmu przyjmują wartości bliższe 0 (z wyjątkiem *Random Forest Classifier*), co świadczy o tym, że metoda ta osiąga lepsze wyniki. Nie można jednak pomijać faktu, że metoda *Random Search* też wskazuje na tendencję do tunowalności, jednak rozkłady te są bardziej rozproszone.

2.3 Miara dopasowania

Jednym z kluczowych kroków analizy przeprowadzonych badań, było stworzenie wykresów miar dopasowania dla każdej kombinacji danych, algorytmu oraz metody optymalizacji. Zostały one przedstawione na Rysunkach 28, 32. Jak przedstawiają wyniki, najlepszą dokładnością we wszystkich zaprezentowanych przypadkach wykazał się zbiór *Santander*. Zbiór *Click Prediction* mimo niższych wyników, również prezentuje zadowalające wartości. Jak można zauważyć, najmniej skuteczną metodą w przypadku tego zbioru jest *CatBoost*. Pozostałe dwa zestawy danych mogą budzić niepokój w poszczególnych przypadkach.

2.4 Różnica w dokładności algorytmów

W pracy zbadano również rozkład różnicy dokładności dla każdego zbioru danych oraz algorytmu uczenia maszynowego w przypadku użytych metod optymalizacji. Na podstawie uzyskanego maksymalnego wyniku, obliczono różnicę względem średnich dla każdej z iteracji, co przedstawiono na Rysunkach 36, 40.

3 Testy statystyczne

W celu zbadania różnic wyników pomiędzy technikami losowania hiperparametrów, użyto testu *Wilcoxona* dla dwóch próbek. Dla wybranych zbiorów danych oraz dla każdego algorytmu uczenia maszynowego, porównano i zbadano istotność różnicy w wynikach dla obu metod optymalizacji za pomocą testu rangowego. Ustalono *poziom istotności* równy 0,05. Wyniki przedstawiono w Tabeli 4.

Algorytm	**Zbiór danych**	**p-value**	**Wniosek**
KNN	Santander	1.105e-03	Istnieją istotne różnice między próbkami
KNN	Click	2.179e-02	Istnieją istotne różnice między próbkami
Random Forest	Santander	1.863e-09	Istnieją istotne różnice między próbkami
Random Forest	Click	1.192e-09	Istnieją istotne różnice między próbkami
CatBoost	Santander	2.594e-05	Istnieją istotne różnice między próbkami
CatBoost	Click	1.101e-01	Brak istotnych różnic między próbkami

Tabela 4: Tabela z przykładowymi wynikami testu Wilcoxona dla zbiorów danych *Santander* oraz *Click*.

4 Podsumowanie

Wybrane przez nas algorytmy potrafią dobrze dopasowywać się do modelu, a optymalizacja hiperparmetrów umożliwia otrzymanie jeszcze lepszych wyników.
W celu optymalizacji lepiej wykorzystywać metodę bayesowską (*Bayes Search*), gdyż sprawniej i trafniej dopasowuje odpowiednie kombinację hiperparametrów do zbioru. Wymaga ona jednak dużej mocy obliczeniowej oraz odpowiedniej ilości czasu. Metoda *Random Search* umożliwia uzyskanie mniej zadowalających wyników, ale robi to szybciej oraz nie jest tak mocno zasobożerna jak *Bayes*.
W naszym przypadku trudno jest dokładnie określić zarówno tunowalność algorytmów, jak i stabilność optymalizacji. Powodem jest zbyt słabo przystosowany do tego sprzęt dla tak wybranych zbiorów. Zakładając, że różnorodne zbiory będą dobrze odzwierciedlać zachowania algorytmów przy optymalizacji, nie zdawano sobie sprawy o wielkości postawionego wyzwania.
Otrzymane wyniki jednak są dobrą podstawą do dalszych eksperymentów i ukazania pewnych różnic przy omawianych zagadnieniach, jak również inspiracją do tworzenia kolejnych udoskonaleń w uczeniu maszynowym.

Literatura

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

5 Wizualizacja wyników

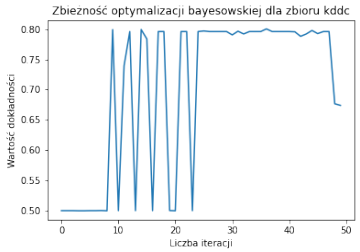
Rysunek 1: Wykres dla algorytmu KNN.

Rysunek 2: Wykres dla algorytmu Random Forest Classifier.

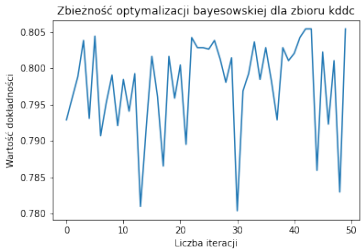
Rysunek 3: Wykres dla algorytmu CatBoost.

Rysunek 4: Wykresy dla zbioru *Christine* do badania stabilności.

Rysunek 5: Wykres dla algorytmu KNN.

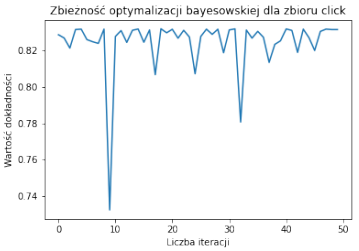


Rysunek 6: Wykres dla algorytmu Random Forest Classifier.

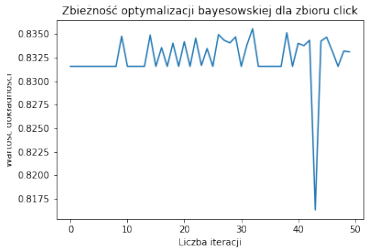


Rysunek 7: Wykres dla algorytmu CatBoost.

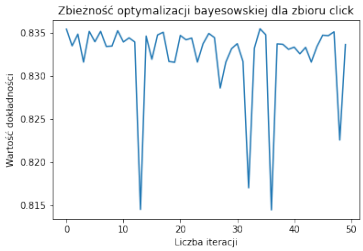
Rysunek 8: Wykresy dla zbioru *KDDC* do badania stabilności.



Rysunek 9: Wykres dla algorytmu KNN.



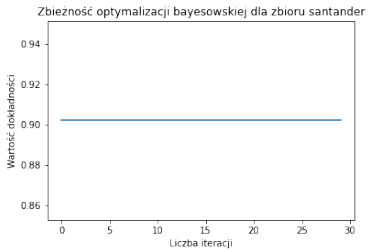
Rysunek 10: Wykres dla algorytmu *Random Forest Classifier*.



Rysunek 11: Wykres dla algorytmu CatBoost.

Rysunek 12: Wykresy dla zbioru *Click Prediction* do badania stabilności.

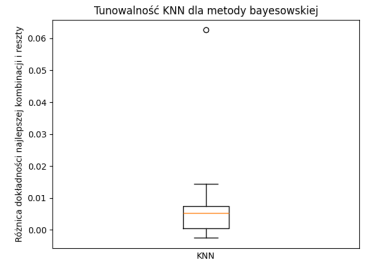
Rysunek 13: Wykres dla algorytmu KNN.



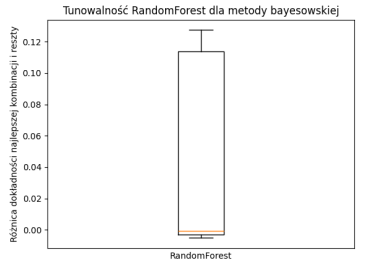
Rysunek 14: Wykres dla algorytmu Random Forest Classifier.

Rysunek 15: Wykres dla algorytmu CatBoost.

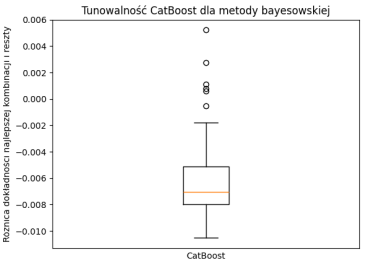
Rysunek 16: Wykresy dla zbioru *Santander* do badania stabilności.



Rysunek 17: Wykres dla algorytmu KNN.

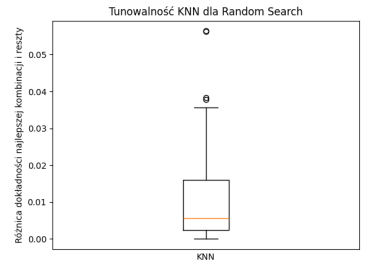


Rysunek 18: Wykres dla algorytmu Random Forest Classifier.

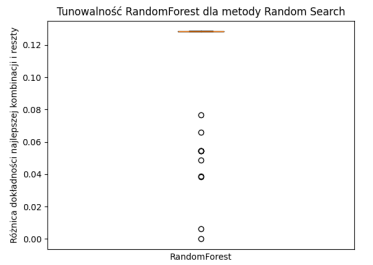


Rysunek 19: Wykres dla algorytmu CatBoost.

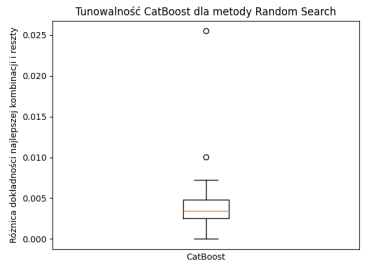
Rysunek 20: Tunowalność z wykorzystaniem *Bayes Optimazation*.



Rysunek 21: Wykres dla algorytmu KNN.

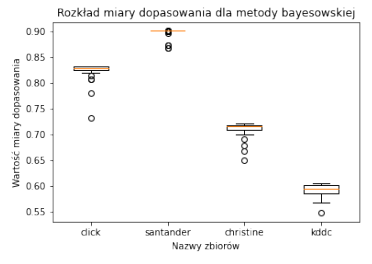


Rysunek 22: Wykres dla algorytmu Random Forest Classifier.

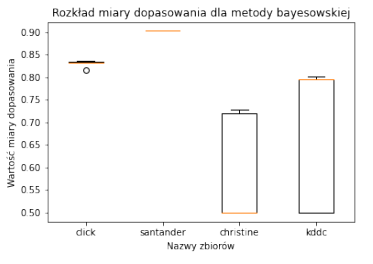


Rysunek 23: Wykres dla algorytmu CatBoost.

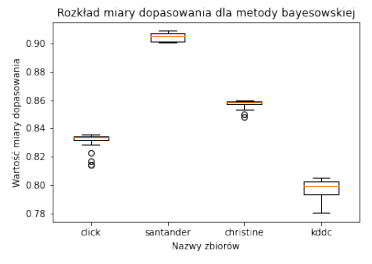
Rysunek 24: Tunowalność z wykorzystaniem *Random Search*.



Rysunek 25: Wykres dla algorytmu KNN.

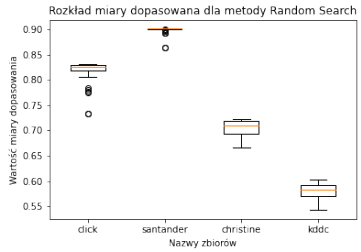


Rysunek 26: Wykres dla algorytmu Random Forest Classifier.

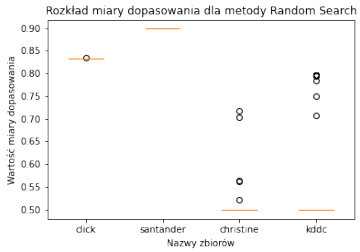


Rysunek 27: Wykres dla algorytmu CatBoost.

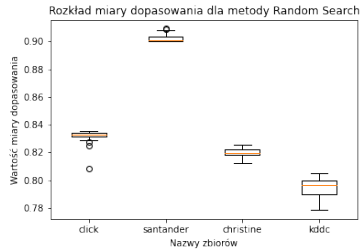
Rysunek 28: Wykresy dla metody optymalizacji *Bayes Optimazation* miary dopasowania.



Rysunek 29: Wykres dla algorytmu KNN.

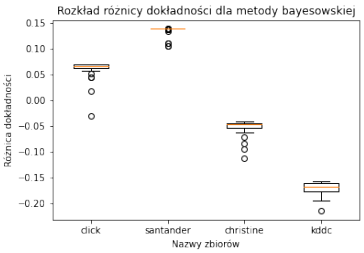


Rysunek 30: Wykres dla algorytmu Random Forest Classifier.

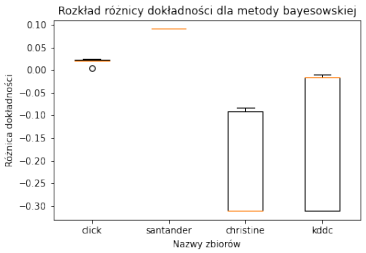


Rysunek 31: Wykres dla algorytmu CatBoost.

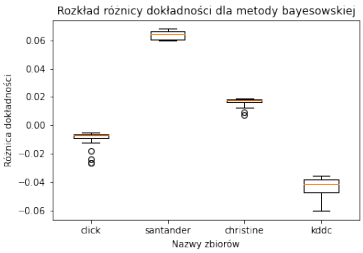
Rysunek 32: Wykresy dla metody optymalizacji *Random Search* miary dopasowania.



Rysunek 33: Wykres dla algorytmu KNN.

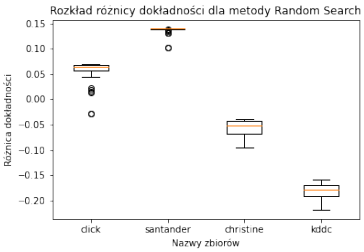


Rysunek 34: Wykres dla algorytmu Random Forest Classifier.

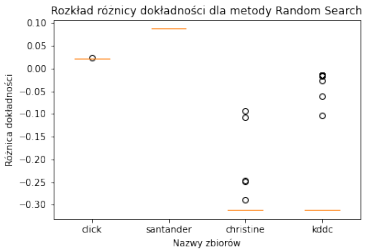


Rysunek 35: Wykres dla algorytmu CatBoost.

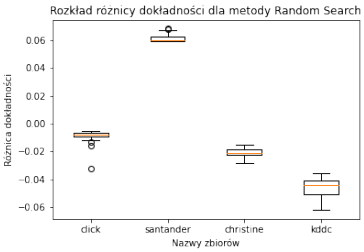
Rysunek 36: Wykresy dla metody optymalizacji *Bayes Optimazation* dla rozkładu różnicy dokładności.



Rysunek 37: Wykres dla algorytmu KNN.



Rysunek 38: Wykres dla algorytmu Random Forest Classifier.



Rysunek 39: Wykres dla algorytmu CatBoost.

Rysunek 40: Wykresy dla metody optymalizacji *Random Search* dla rozkładu różnicy dokładności.