Raport
Automatyczne uczenie maszynowe – Projekt 2

Krzysztof Adamczyk
Katarzyna Bownik
Aliaksei Shaukunou

26 stycznia 2026

1 Wstęp

2 Etap 1: Wybór Modeli do Portfolio

W ramach realizacji pierwszego etapu projektu przyjęto podejście związane z Lokalnym Screeningiem Modeli. Przeprowadzono eksperymenty mające na celu wyłonienie najbardziej uniwersalnych konfiguracji modeli, które będą cechowały się skutecznością i stabilnością na zróżnicowanych zbiorach danych.

2.1 Metodologia i Dane

Do eksperymentów wykorzystano zbiór 20 zróżnicowanych datasetów pochodzących z repozytorium OpenML (suite OpenML-CC18), obejmujących wiele problemów. Aby zapewnić uniwersalność rozwiązania, każda konfiguracja modelu została zamknięta w Pipeline, realizujący automatyczny preprocessing obejmujący:

- **Zmienne numeryczne:** SimpleImputer (uzupełnianie braków średnią bąć medianą) oraz MinMaxScaler (skalowanie do przedziału $[0, 1]$).
- **Zmienne kategoryczne:** Uzupełnianie braków najczęstszą wartością oraz kodowanie OneHotEncoder.

2.2 Przestrzeń Poszukiwań

Zdefiniowano przestrzeń poszukiwań obejmującą 6 rodzin algorytmów: modele liniowe (Logistic Regression, Linear SVM), oparte na sąsiedztwie (kNN) oraz zespoły drzew (Random Forest, Extra Trees, LightGBM). Zakres testowanych modeli i hiperparametrów został dobrany na podstawie wyników zewnętrznych z TabArena oraz zapewniając, że w finalnym zbiorze znajdą się algorytmy dobrze dopasowane do różnych charakterystyk danych.

Algorytm	**Hiperparametr**	**Typ**	**Zakres**
Logistic Regression	C	numeric	10^{-4} – 10^{2}
	penalty	discrete	l1, l2
Linear SVM	C	numeric	10^{-4} – 10^{2}
k-Nearest Neighbors	n_neighbors	integer	3 – 50
	weights	discrete	uniform, distance
	p	discrete	1, 2
Random Forest	n_estimators	integer	100 – 800
	max_depth	integer	3 – 19 or None
	min_samples_leaf	integer	1 – 20
	max_features	discrete	sqrt, log2, None
Extra Trees	n_estimators	integer	200 – 800
	max_depth	integer	3 – 19 or None
	min_samples_leaf	integer	1 – 20
	max_features	discrete	sqrt, log2, None
LightGBM	n_estimators	integer	100 – 800
	num_leaves	integer	16 – 256
	learning_rate	numeric	10^{-3} – 0.3
	min_child_samples	integer	5 – 100
	subsample	numeric	0.6 – 1.0
	colsample_bytree	numeric	0.6 – 1.0

Tabela 1: Zakresy hiperparametrów dla badanych modeli

2.3 Eksperymenty i Wyniki

Dla każdego z 6 typów algorytmów wylosowano metodą RandomSearch po 20 konfiguracji modeli, co dało łącznie 120 modeli kandydujących. Ewaluację przeprowadzono przy użyciu 5-krotnej walidacji krzyżowej.

Jako kryterium oceny przyjęto metrykę Balanced Accuracy, uśrednioną dla każdego modelu po wszystkich 20 zbiorach danych.

W rezultacie do finalnego portfolio wybrano 50 najlepszych konfiguracji, które uzyskały średnio najwyższą wartość miary. Pozwoliło nam to na znalezienie modeli zapewniających najwyższą stabilność i skuteczność na danych zróżnicowanych. Portfolio jest dostępne w załączonym pliku $\texttt{best}_m odels.json$

3 Klasa MiniAutoML

Została stworzona klasa MiniAutoML implementująca następujące metody publiczne.

- `__init__(self, models_config)`
  Tworzy obiekt MiniAutoML na podstawie listę modeli models_config
- `fit(self, X_train, y_train)` Metoda zwraca wytrenowany Soft Voting Ensemble składający się z wybranych 5 modeli (lub mniej w przypadku, gdy lista modeli przekazana podczas inicjalizacji obiektu zawierała mniej niż 5 elementów). Działanie metody zostało podzielone na 10 kroków, przedstawionych poniżej.
  1. **Walidacja danych wejściowych**
     Sprawdzane jest, czy X_train ma typ *pandas.DataFrame*, a w razie potrzeby następuje jego konwersja. Analogicznie weryfikowany jest typ zmiennej docelowej y_train.
  2. **Obsługa wartości brakujących**
     Usuwane są obserwacje zawierające brakujące wartości w zmiennej docelowej. Następnie dopasowywany jest mechanizm imputacji braków danych w cechach wejściowych oparty na lasach losowych.
  3. **Detekcja i usuwanie obserwacji odstających**
     W celu identyfikacji anomalii stosowany jest algorytm Isolation Forest. Obserwacje zaklasyfikowane jako odstające są eliminowane wraz z odpowiadającymi im etykietami.
  4. **Selekcja cech**
     Eliminowane są cechy o zerowej wariancji przy użyciu funkcji sklearn.feature_selection.Variance (to znaczy usuwane są cechy zawierające tylko 1 wartość unikatową). Następnie usuwane są silnie skorelowane zmienne wejściowe na podstawie macierzy korelacji.
  5. **Skalowanie cech**
     Cechy są normalizowane do przedziału $[0, 1]$ przy użyciu funkcji sklearn.preprocessing.MinMaxScale Równolegle zachowywana jest wersja nieskalowana danych dla modeli drzewiastych.
  6. **Kodowanie zmiennej docelowej**
     Identyfikowane są unikalne klasy decyzyjne oraz przypisywane są etykiety klas pozytywnej i negatywnej, które zostają zapisane w atrybutach obiektu.
  7. **Uczenie modeli z walidacją krzyżową**
     Dla każdego modelu z przekazanej podczas inicjalizacji listy wykonywana jest 3-krotna walidacja krzyżowa typu *Stratified K-Fold* . Modele oceniane są przy użyciu miary *balanced accuracy*, a uzyskane wyniki są zapamiętywane.
  8. **Selekcja najlepszych modeli**
     Modele są sortowane według uzyskanej jakości, a następnie wybieranych jest maksymalnie pięć najlepszych konfiguracji. W przypadku braku poprawnie wytrenowanych modeli zgłaszany jest wyjątek wykonania.
  9. **Ponowne uczenie najlepszych modeli na pełnych danych**
     Wybrane modele są ponownie dopasowywane na całym zbiorze treningowym.
  10. **Podsumowanie procesu uczenia**
      Obliczana jest średnia jakość modeli.
- `predict_proba(self, X_test)`
  Zwraca wektor prawdopodobieństw przynależności do klasy pozytywnej (za klasę pozytywną uznaje się wartość atrybutu positive_class, którego inicjalizacja następuje podczas działania metody fit). Prawdopodobieństwa te są obliczane z wykorzystaniem klasyfikatora *Soft Voting Ensemble* składającego się z 5 (albo mniej) wybranych modeli.

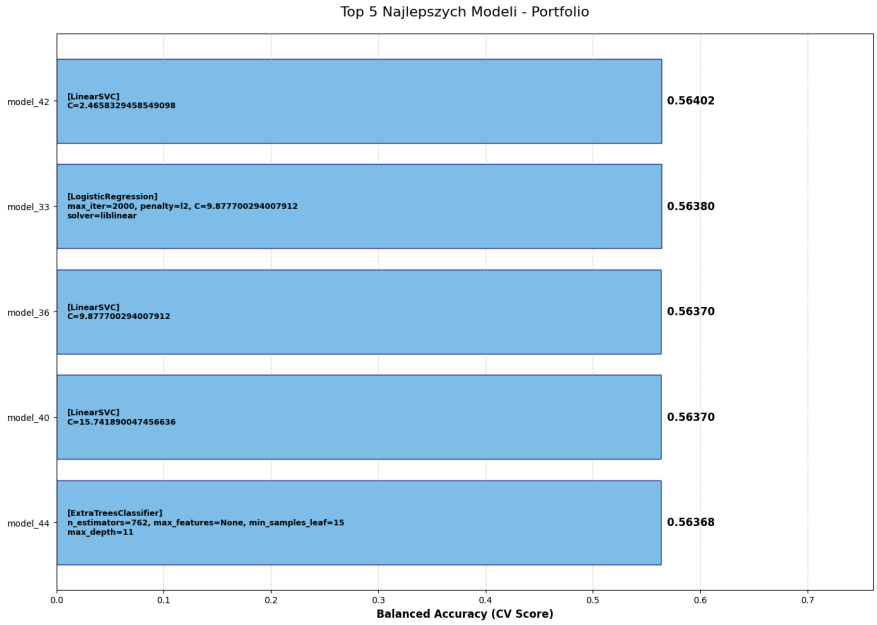
- `predict(self, X_test)`
  Zwara wektor predykcji klas.

4 Ewaluacja Rozwiązania

Zaimplementowany system MiniAutoML został przetestowany na przykładowym zbiorze danych podanym na zajęciach, który nie był wykorzystywany na etapie tworzenia portfolio. Eksperyment miał na celu weryfikację poprawności działania całego potoku przetwarzania oraz skuteczności mechanizmu selekcji modeli.

4.1 Przebieg eksperymentu i wyniki

Zbiór danych został podzielony na część treningową i testową w proporcji 80:20 (z zachowaniem stratyfikacji). Na zbiorze treningowym uruchomiono metodę fit, która przeprowadziła Preprocessing danych, 5 krotną walidację krzyżową wszystkich modeli z portfolio oraz wybór 5 najlepszych modeli na podstawie metryki Balanced Accuracy. W wyniku działania procedury selekcji, system automatycznie odrzucił modele słabsze. Wyniki dla przykładowego zbioru danych zostały przedstawione na wykresie 1.



Rysunek 1: Top 5 najlepszych modeli z portfolio dla przykładowego zbioru danych

5 Wnioski

Zrealizowany projekt systemu MiniAutoML spełnia wszystkie postawione wymagania funkcjonalne. Utworzone portfolio modeli stanowi uniwersalną bazę zawierającą różne typy zoptymalizowanych algorytmów, co pozwala na rozwiązywanie różnorodnych problemów klasyfikacyjnych. System skutecznie automatyzuje procesy przetwarzania danych, minimalizując ingerencję człowieka. Ponadto system poprawnie zdiagnozował specyfikę zbioru testowego, wybierając prostsze i stabilniejsze modele liniowe, co świadczy o skuteczności walidacji w klasie MiniAutoML. Umiarkowany poziom uzyskanych wyników wynikał ze specyfiki danych, a nie z błędów w działaniu systemu.