Mini-AutoML

Daria Bartkowiak, Alicja Przeździecka, Oliwia Wójcicka

19 stycznia 2026

1 Wstęp

Celem projektu było stworzenie uproszczonego systemu Mini-AutoML do automatycznej klasyfikacji binarnej, opartego na selekcji i wykorzystaniu gotowej listy modeli.

2 Dobór modeli i hiperparametrów

Zestaw modeli został dobrany w taki sposób, aby obejmował różne rodziny algorytmów uczenia maszynowego. Uwzględniono modele liniowe, algorytmy oparte na drzewach decyzyjnych, metody zespołowe typu boosting, modele oparte na instancjach oraz algorytmy wykorzystujące metodę wektorów nośnych oraz modele probabilistyczne. Taki dobór zapewnia wysoką różnorodność algorytmiczną i pozwala wychwytywać różne charakterystyki danych, takie jak liniowość, nieliniowość, wysoka wymiarowość, obecność szumu czy niezbalansowany rozkład klas.

Dla każdego z rozważanych modeli przeprowadziliśmy szczegółową analizę dostępnych hiperparametrów, koncentrując się na ich wpływie na złożoność modelu, stabilność procesu uczenia oraz zdolność generalizacji.

Na podstawie tej analizy wytypowaliśmy najbardziej interesujące i reprezentatywne zestawy hiperparametrów. Wszystkie obserwacje, wnioski oraz rozważane konfiguracje zostały szczegółowo udokumentowane w pliku **config_library.ipynb**. Wyselekcjonowane zestawy parametrów (modele wraz z hiperparametrami), zostały następnie przeniesione do pliku **models_for_tests.json**.

3 Selekcja modeli na zbiorach OpenML

Aby wybrać finalne zestawy parametrów, przeprowadziliśmy testy na różnych zbiorach danych pochodzących z platformy OpenML. Zbiory te zostały poddane preprocessingowi, a następnie wykorzystane do oceny konfiguracji modeli. Na podstawie uzyskanych wyników przeprowadziliśmy selekcję według przyjętej strategii. Pozwoliło to wyłonić najbardziej obiecujące zestawy parametrów, które zapisano w pliku **models.json**. Cały proces został zrealizowany w pliku **selection.ipynb**, a wyniki zapisane w **results.csv**.

3.1 Strategia selekcji

Strategia selekcji modeli przebiega według następujących kroków:

- **Scoring** Każda konfiguracja oceniana jest na podstawie średniej wartości AUC, skorygowanej o zmienność wyników pomiędzy zbiorami danych (kara za niestabilność).

$$Score = Mean\_AUC - \text{PENALTY\_FACTOR} \cdot Volatility$$

- **Etap 1 - Diversity:** W pierwszym kroku wybierany jest po jednym najlepiej działającym modelu z każdej rodziny algorytmów (Linear, Geometry/Bayes, RandomForest, Boosting).
- **Etap 2 - Performance:** Lista konfiguracji jest następnie uzupełniana do 30 modeli na podstawie najwyższego wyniku oceny, niezależnie od przynależności do rodziny.
- **Optymalizacja kolejności:** Końcowa lista modeli jest uporządkowana z myślą o ograniczeniach czasowych. Na szczycie listy priorytetów umieściliśmy liderów poszczególnych rodzin algorytmicznych. Taka strategia gwarantuje, że w przypadku konieczności przedwczesnego przerwania treningu, system będzie dysponował zróżnicowanym i stabilnym zestawem modeli bazowych, niezbędnym do skonstruowania skutecznego Ensemblu, unikając sytuacji, w której przetestowane zostałyby tylko modele jednego typu.

3.2 Wyniki eksperymentu

Wyniki - wybrane przez naszą selekcję top 30 modeli:

Tabela 1: Wybrane modele do portfolio (skrócony opis konfiguracji).

#	Nazwa	Algorytm (klasa + 1–2 parametry)	Rodzina
1	lda1	LDA (solver=lsqr, shrinkage=auto, ...)	linear / LDA
2	svm2	SVC (kernel=linear, C=0.1, ...)	SVM
3	random_forest3	RF (n_estimators=500, max_depth=15, ...)	Random Forest
4	xgboost5	XGB (n_estimators=200, max_depth=10, ...)	XGBoost
5	ridge_clf	RidgeClassifier (alpha=1.0, class_weight=balanced, ...)	linear
6	lda2	LDA (solver=svd, ...)	linear / LDA
7	log_reg2	LogReg (solver=lbfgs, max_iter=2000, ...)	linear
8	log_reg1	LogReg (solver=saga, class_weight=balanced, ...)	linear
9	xgboost4	XGB (n_estimators=1000, max_depth=6, ...)	XGBoost
10	log_reg3	LogReg (penalty=l1, C=0.5, ...)	linear
11	xgboost1	XGB (n_estimators=100, max_depth=4, ...)	XGBoost
12	xgboost2	XGB (n_estimators=400, max_depth=7, ...)	XGBoost
13	xgboost6	XGB (n_estimators=300, max_depth=5, ...)	XGBoost
14	xgboost3	XGB (n_estimators=150, max_depth=6, ...)	XGBoost
15	hist_gradient_boosting	HistGB (max_iter=200, max_depth=10, ...)	boosting (sklearn)
16	random_forest4	RF (n_estimators=200, max_depth=25, ...)	Random Forest
17	random_forest2	RF (n_estimators=300, max_depth=None, ...)	Random Forest
18	gradient_boosting	GB (n_estimators=200, learning_rate=0.1, ...)	boosting (sklearn)
19	extra_trees1	ExtraTrees (n_estimators=200, max_depth=12, ...)	Extra Trees
20	extra_trees2	ExtraTrees (n_estimators=400, criterion=entropy, ...)	Extra Trees
21	random_forest1	RF (max_depth=10, class_weight=balanced, ...)	Random Forest
22	svm1	SVC (class_weight=balanced, probability=true, ...)	SVM
23	log_reg4	LogReg (penalty=elasticnet, C=0.1, ...)	linear
24	ada_boost1	AdaBoost (n_estimators=100, learning_rate=1.0, ...)	boosting (AdaBoost)
25	ada_boost2	AdaBoost (n_estimators=200, learning_rate=0.1, ...)	boosting (AdaBoost)
26	log_reg5	LogReg (solver=liblinear, C=0.01, ...)	linear
27	svm4	SVC (C=10, probability=true, ...)	SVM
28	svm3	SVC (kernel=poly, degree=3, ...)	SVM
29	knn4	kNN (n_neighbors=50, weights=distance, ...)	kNN
30	knn2	kNN (n_neighbors=25, weights=distance, ...)	kNN

4 Pakiet MiniAutoML

Po zakończeniu selekcji wybrane 30 konfiguracji przekazywane jest do właściwego modułu MiniAutoML, zaimplementowanego w pliku `automl.py`. Moduł ten odpowiada za automatyczny trening modeli, optymalizację progu decyzyjnego (w zakresie od 0.2 do 0.8) oraz budowę końcowego rankingu.

4.1 Selekcja modelu z portfolio dla nowego zbioru danych

Kluczowym etapem pakietu jest wykonanie wyboru modelu **w czasie wywołania metody** `fit()` na danych treningowych. U nas selekcja realizowana jest poprzez szybki *turniej modeli* (ranking na walidacji) oraz opcjonalny wybór ensemblu, z zachowaniem ograniczeń czasowych i pamięciowych. Selekcja w `fit(X_train, y_train)` przebiega następująco:

1. **Preprocessing danych.** Budowany jest wspólny pipeline przetwarzania: imputacja medianą i transformacja Yeo-Johnsona dla cech numerycznych oraz imputacja stałą wartością `missing` i One-Hot Encoding dla cech kategorycznych.

2. **Ograniczenie kosztu selekcji (subsampling).** Jeżeli liczba obserwacji przekracza limit (`max_rows_limit`), system losuje podzbiór danych do etapu selekcji. Celem jest przyspieszenie porównania konfiguracji modeli bez utraty ogólnego obrazu jakości.

3. **Walidacja hold-out i ranking modeli.** Na danych selekcyjnych wykonywany jest podział stratyfikowany 80/20 na zbiór treningowy i walidacyjny. Następnie każda konfiguracja z portfolio modeli jest trenowana na części treningowej i oceniana na walidacyjnej metryką **Balanced Accuracy**. Na tej podstawie budowany jest ranking (leaderboard).

4. **Dobór progu decyzyjnego.** Dla modeli zwracających prawdopodobieństwa (`predict_proba`) system przeszukuje siatkę progów $[0.2, 0.8]$ (krok 0.05) i wybiera próg maksymalizujący Balanced Accuracy na walidacji. Dla modeli bez prawdopodobieństw stosowany jest próg domyślny 0.5.

5. **Ensemble.** Szczegóły konstrukcji ensemblu opisano w sekcji 4.3.

6. **Trening finalny.** Wybrane rozwiązanie jest trenowane ponownie na całych danych treningowych (po preprocessingu).

4.2 Uzasadnienie wyboru metody

Zastosowana procedura jest kompromisem pomiędzy jakością a czasem działania:

- ranking na walidacji hold-out jest znacznie szybszy niż pełna k-krotna walidacja krzyżowa,
- Balanced Accuracy jest odporna na niezbalansowanie klas,
- tuning progu poprawia jakość klasyfikacji bez zmiany modelu,
- preferencja różnorodności rodzin zwiększa sens budowy ensemblu.

4.3 Ensembling

Po zakończeniu turnieju modeli system rozważa nie tylko najlepszy model pojedynczy, ale również możliwość zbudowania **Ensemblu typu soft-voting** (`VotingClassifier`). Do głosowania wybierane jest **`top_k`** (domyślnie 5) najlepszych modeli z turnieju. Algorytm priorytetyzuje różnorodność, starając się w pierwszej kolejności dobrać liderów z różnych rodzin (np. Liniowe, Drzewa, Boosting), aby zredukować korelację błędów.

W ensemblu zastosowano głosowanie miękkie (soft voting), czyli uśrednianie prawdopodobieństw klas zwracanych przez poszczególne modele. Dla tak skonstruowanego ensemblu system analogicznie przeprowadza dobór **optymalnego progu decyzyjnego**, aby zmaksymalizować Balanced Accuracy na zbiorze walidacyjnym.

Na końcu porównujemy wynik najlepszego modelu pojedynczego z wynikiem ensemblu i wybiera rozwiązanie o wyższej skuteczności. Jeśli ensemble wygrywa, jest ponownie trenowany na pełnym zbiorze danych, co pozwala zachować jakość końcowego modelu przy jednoczesnym ograniczeniu ryzyka przeuczenia jednego algorytmu.

4.4 Wyniki – przykład na zbiorze *diabetes*

Eksperyment wykonano na zbiorze *diabetes*, przy niezbalansowaniu klas: 0 – 65.1%, 1 – 34.9%.

Tabela 2: Leaderboard (TOP 10) dla zbioru *diabetes*.

#	**Model**	**Rodzina**	**BalAcc**	**Threshold**
1	knn4	knn	0.8151	0.30
2	knn2	knn	0.8052	0.30
3	extra_trees1	other	0.7936	0.45
4	svm1	svm	0.7919	0.25
5	extra_trees2	other	0.7863	0.50
6	log_reg2	linear	0.7820	0.35
7	log_reg1	linear	0.7820	0.50
8	log_reg3	linear	0.7820	0.50
9	lda1	linear	0.7810	0.50
10	lda2	linear	0.7810	0.50

Tabela 3: Porównanie: najlepszy model pojedynczy vs ensemble

Element	**Wartość**
Najlepszy model pojedynczy	knn4
Balanced Accuracy (single)	0.8151
Threshold (single)	0.30
Balanced Accuracy (ensemble)	0.7981
Wybrany wariant	SINGLE (knn4)
Czas całkowity `fit()` [min]	0.11

5 Wnioski

Eksperyment pokazał, że selekcja modelu realizowana bezpośrednio w metodzie `fit()` pozwoliła automatycznie wybrać najlepszą konfigurację z portfolio w krótkim czasie. Ranking modeli na walidacji wskazał, że w testach najwyższy wynik Balanced Accuracy osiągały modele kNN oraz metody zespołowe oparte na drzewach (ExtraTrees).

Zbudowany ensemble (soft voting) nie poprawił wyniku względem najlepszego modelu pojedynczego, co sugeruje, że w tym przypadku topowe modele miały podobne błędy lub dominował jeden bardzo mocny algorytm. Dodatkowo strojenie progu decyzyjnego (np. 0.30 dla kNN) pozwoliło zwiększyć Balanced Accuracy w porównaniu do standardowego progu 0.5, co potwierdza zasadność tego kroku w pipeline selekcji.