Mini-AutoML dla Danych Tabelarycznych

Jan Taran
Michał Syrkiewicz
Piotr Wysocki

23 stycznia 2026

1 Wstęp

Celem projektu było stworzenie uproszczonego systemu AutoML do automatycznej klasyfikacji binarnej danych tabelarycznych. System opiera się na predefiniowanym portfolio modeli, preprocessingu, mechanizmie selekcji oraz ensemblingu.

2 Etap 1: Budowa Portfolio Modeli

Decyzję o kształcie portfolio modeli oparto na wnioskach z artykułu Grinsztajna [1] oraz ogólnodostępnych benchmarkach. Główny trzon zestawienia stanowią modele drzewiaste oparte na algorytmach gradient boosting (LightGBM, CatBoost, XGBoost), które wykazują najwyższą skuteczność na danych tabelarycznych. W celu zdywersyfikowania zespołu i zwiększenia odporności na różnorodne charakterystyki danych, pulę uzupełniono o algorytmy: Random Forest, ExtraTrees, k-najbliższych sąsiadów (k-NN) oraz regresję logistyczną.

Ponadto, w oparciu o obserwacje heurystyczne poczynione w trakcie wstępnych eksperymentów na podanym zbiorze danych oraz własnych, zdecydowano się dodatkowo rozszerzyć portfolio o trzy warianty regresji: ElasticNet oraz modele z silniejszą regularyzacją L1 i L2. Zaobserwowano, że w dalszych etapach selekcji modele te osiągały satysfakcjonujące wyniki, a dzięki krótkiemu czasowi trenowania nie wpływały negatywnie na wydajność czasową całego potoku. Ze względu na wysoką złożoność obliczeniową, która znacząco wydłużyła proces (etap 3), zastosowano tylko dwie sieci neuronowe. Przy wyborze liczby konkretnych implementacji posiłkowano się rankingiem TabArena [2].

Konfiguracje hiperparametrów dobrano, korzystając z trzech źródeł, w celu zapewnienia szerokiego pokrycia przestrzeni poszukiwań:

Benchmark TabArena (AutoGluon): Dla podstawowych konfiguracji wybrano najlepsze ustawienia stosowane w pakiecie AutoGluon. Zdecydowano się na następującą liczebność: k-NN (1), regresja logistyczna (2), Random Forest (3), ExtraTrees (1), XGBoost (2), CatBoost (3), LightGBM (2) oraz NeuralNetTorch (2).

Pakiet FLAML: W celu dalszej dywersyfikacji wykorzystano bibliotekę FLAML, wybierając z niej najskuteczniejsze konfiguracje dla modeli: XGBoost (4) oraz LightGBM (3).

MementoML (Kaggle): Uzupełniająco skorzystano z wyników analizy MementoML na zbiorach danych z Kaggle. Obliczono średnie AUC dla modeli, uśredniając wyniki z różnych podziałów walidacyjnych. Na podstawie uzyskanego rankingu do portfolio dołączono po jednym modelu XGBoost i CatBoost z najlepszą siatką hiperparametrów.

Ostatecznie portfolio składa się z 28 modeli, które zostaną poddane dalszej selekcji w ramach metody fit() systemu autoML.

3 Etap 3: Metoda Selekcji i Implementacja

Główna klasa `MiniAutoML` realizuje metodę `fit`, która odpowiada za wybór najlepszego modelu dla nowo dostarczonych danych treningowych.

3.1 Procedura preprocessingu i inżynierii cech

W celu maksymalizacji jakości predykcji zaimplementowano zaawansowany moduł wstępnego przetwarzania danych (*preprocessing*), mający na celu ujawnienie ukrytych zależności oraz redukcję szumu.

Za proces ten odpowiada klasa `AutoMLPreprocessor`, implementująca standardowy interfejs biblioteki `scikit-learn` (metody `fit`, `transform`, `fit_transform`) oraz pomocniczą metodę `process`. Ta ostatnia automatyzuje cały potok przetwarzania, przyjmując surowy zbiór danych i zwracając gotowe podzbiory treningowe i testowe.

Proces przetwarzania danych przebiega w następujących krokach:

1. **Wstępne czyszczenie danych:**
   - Usunięcie kolumn o stałej wartości (zerowa wariancja).
   - Eliminacja zmiennych kategorycznych o zbyt wysokiej kardynalności (np. unikalne identyfikatory), które nie niosą wartości predykcyjnej.
   - Zachowanie kolumn dat do dalszego przetwarzania.
2. **Detekcja typów zmiennych:** System automatycznie klasyfikuje kolumny jako numeryczne, kategoryczne lub daty. Klasyfikacja opiera się na analizie typów danych oraz heurystyce badającej stosunek liczby unikalnych wartości do liczności próby (wykrywanie zmiennych dyskretnych ukrytych jako liczby).
3. **Przetwarzanie dat:**
   - Ekstrakcja podstawowych cech: rok, miesiąc, dzień, dzień tygodnia.
   - Transformacja cykliczna: zastosowanie funkcji trygonometrycznych (sinus, cosinus) dla cech okresowych (np. miesiąc, dzień tygodnia), co pozwala modelom zachować ciągłość między końcem a początkiem cyklu (np. grudzień blisko stycznia).
4. **Imputacja braków danych:**
   - *Dane numeryczne:* Zastosowano zaawansowaną metodę `IterativeImputer` (algorytm MICE), wykorzystującą `ExtraTreesRegressor` do predykcji brakujących wartości na podstawie korelacji z innymi cechami.
   - *Dane kategoryczne:* Ze względu na złożoność obliczeniową, zastosowano imputację modą. Analogiczne podejście zastosowano dla zmiennej celowej.
5. **Transformacja rozkładu i kodowanie:**
   - Dane numeryczne poddawane są standaryzacji oraz transformacji Yeo-Johnsona w celu zbliżenia ich rozkładu do normalnego.
   - Dane kategoryczne są kodowane metodą `OrdinalEncoder`.

3.2 Inżynieria cech (Feature Engineering)

W celu wzbogacenia zbioru danych zastosowano następujące techniki:

1. **Frequency Encoding:** Dodanie kolumn reprezentujących częstość występowania poszczególnych kategorii, co jest szczególnie efektywne dla modeli drzewiastych.
2. **Klasteryzacja KMeans:** Wykorzystanie algorytmu `MiniBatchKMeans` do stworzenia nowych cech reprezentujących przydział do klastra oraz dystans obserwacji od centroidów (inspiracja rozwiązaniem z biblioteki MLJAR).
3. **Interakcje arytmetyczne:** Wygenerowanie nowych cech na podstawie najważniejszych zmiennych numerycznych (wybranych przy użyciu `ExtraTrees`). Tworzone są kombinacje: A+B, A-B, A*B, A/B, B/A, log(A), sqrt(A). Aby zabezpieczyć się przed dzieleniem przez zero, chwilowo zera są podmieniane przez epsilon.

4. **Dyskretyzacja (Binning):** Zamiana wybranych zmiennych ciągłych na przedziały kwantylowe (np. 5 koszyków), co pomaga w redukcji szumu i wykrywaniu nieliniowości.

3.3 Selekcja cech

Proces redukcji wymiarowości składa się z trzech etapów:

1. **Usuwanie współliniowości:** Eliminacja zmiennych silnie skorelowanych, co stabilizuje modele liniowe i redukuje redundancję.
2. **Selekcja hybrydowa:**
   (a) **Ensemble Screening:** Wstępna selekcja wykorzystująca dwa estymatory: regresję logistyczną (z karą L1) oraz `ExtraTreesClassifier`. Cecha jest zachowywana, jeśli zostanie uznana za istotną przez przynajmniej jeden z tych modeli.
   (b) **PCA (Principal Component Analysis):** W przypadku, gdy po wstępnej selekcji liczba cech nadal przekracza ustalony próg, stosowana jest redukcja wymiarowości z zachowaniem 99.9% wariancji.
   (c) **Sequential Feature Selector (SFS):** Jeśli liczba cech jest umiarkowana, uruchamiany jest algorytm eliminacji wstecznej (*Backward Elimination*), wykorzystujący jako estymator szybki klasyfikator `KNeighborsClassifier` (z parametrem $k = 4$).

W przypadku zastosowania PCA, zbiór danych wynikowych składa się wyłącznie ze zmiennych numerycznych (komponentów głównych). W przeciwnym razie zachowywane są oryginalne typy kolumn (w tym kategoryczne), co pozwala na wykorzystanie ich potencjału przez modele takie jak CatBoost czy XGBoost.

3.4 Procedura selekcji modeli (Screening)

Po etapie inżynierii cech system przystępuje do ewaluacji zdefiniowanego portfolio modeli na przetworzonym zbiorze treningowym. Aby zapewnić rzetelność wyników i zminimalizować ryzyko przeuczenia, proces ten realizowany jest w schemacie **5-krotnej walidacji krzyżowej(Stratified 5-Fold CV)**. Gwarantuje to zachowanie proporcji klas w każdym podziale, co jest kluczowe dla stabilności oceny. Wyniki uśrednione z 5 powtórzeń służą do utworzenia wstępnego rankingu, który determinuje kandydatów do etapu łączenia modeli.

3.5 Strategie łączenia modeli (Ensembling)

W celu poprawy zdolności generalizacji, system generuje zespoły modeli wykorzystując predykcje typu **Out-of-Fold (OOF)**. Dzięki temu estymatory są łączone na podstawie ich zachowania na danych, których nie widziały podczas treningu, co eliminuje zjawisko wycieku danych.

3.5.1 Stacking Ensemble

System buduje meta-modele, które uczą się optymalnych wag dla predykcji modeli bazowych. Zastosowano dwa kryteria doboru bazy:

- **Top-3 Best:** Zespół złożony z trzech modeli o najwyższym wyniku walidacyjnym.
- **Top-3 Unique:** Zespół trzech najlepszych modeli należących do rozłącznych rodzin algorytmicznych.

3.5.2 Szybkie heurystyki (Fast Heuristics)

Równolegle testowane są metody agregacji niewymagające dodatkowego treningu, aplikowane do 5 czołowych modeli z rankingu:

- **Uśrednianie i Rangi):** Wyliczanie średniej z prawdopodobieństw lub ich rang. Metoda rangowania uniezależnia wynik od skali kalibracji poszczególnych estymatorów.
- **Champ + Rebel:** System wybiera lidera rankingu oraz model z czołowej piątki najsłabiej z nim skorelowany. Ostateczna predykcja jest ważoną średnią obu, co pozwala korygować specyficzne błędy lidera odmienną perspektywą "buntownika".

3.6 Wybór końcowy i Refitting

Wszystkie wygenerowane zespoły trafiają do wspólnego rankingu z modelami pojedynczymi. Konfiguracja o najwyższym średnim wyniku walidacyjnym zostaje uznana za zwycięską. W ostatnim kroku następuje procedura Refit:

1. Jeśli wygrał model pojedynczy, jest on trenowany ponownie na **całym dostępnym zbiorze treningowym**.
2. Jeśli wygrał Stacking, wszystkie jego modele bazowe są trenowane na pełnym zbiorze, natomiast wagi meta-modelu pozostają zachowane.

Pozwala to w pełni wykorzystać dostępną informację przed ewaluacją na zbiorze testowym.

4 Wnioski

Realizacja i testy systemu Mini-AutoML pozwoliły na sformułowanie kluczowych obserwacji dotyczących automatyzacji procesu uczenia maszynowego na danych tabelarycznych:

1. **Wpływ inżynierii cech na lidera rankingu**
   Zauważono silną zależność między zastosowanym przetwarzania danych a typem wygrywającego modelu. W zależności od preprocessingu wygrywały modele regresyjne, gradient boosting albo sieci neuronowe. W każdym przypadku ensemble modeli nie było znacząco gorsze od wygrywających modeli.
2. **Skuteczność dywersyfikacji w Stackingu**
   Eksperymenty potwierdziły zasadność implementacji strategii *Stacking Top-3 Unique*. Zespoły złożone z modeli o zbliżonej skuteczności, ale działających na różnych zasadach, osiągały przeważnie lepsze wyniki generalizacji niż zespoły złożone z trzech najlepszych, ale niemal identycznych wariantów tego samego algorytmu. Różnorodność błędów popełnianych przez modele składowe okazała się kluczowa dla skuteczności meta-modelu.

Literatura

[1] L. Grinsztajn, E. Oyallon, G. Varoquaux, *Why do tree-based models still outperform deep learning on typical tabular data?*, NeurIPS Datasets and Benchmarks Track, 2022.

[2] TabArena Team, *TabArena Leaderboard*, Hugging Face Space. Dostępne na: https://huggingface.co/spaces/TabArena/leaderboard

[3] AutoGluon, *TabArena Data Configurations*, GitHub Repository. Dostępne na: https://github.com/autogluon/tabarena/tree/main/data/configs

[4] Microsoft, *FLAML Default Binary Configurations*, GitHub Repository. Dostępne na: https://github.com/microsoft/FLAML/blob/main/flaml/default/all/binary.json

[5] MI2DataLab, *MementoML Dataset Parameters*, Kaggle Dataset. Dostępne na: https://www.kaggle.com/datasets/mi2datalab/mementoml?resource=download&select=parameters

[6] Płońska, A., & Płoński, P. (2021). *MLJAR: Automated Machine Learning for Humans*. Dostępne pod adresem: https://github.com/mljar/mljar-supervised [Dostęp: 2024].

[7] Van Buuren, S., & Groothuis-Oudshoorn, K. (2011). *mice: Multivariate Imputation by Chained Equations in R*. Journal of Statistical Software, 45(3), 1-67.

[8] Ferri, F. J., Pudil, P., Hatef, M., & Kittler, J. (1994). *Comparative study of techniques for large-scale feature selection*. In Pattern Recognition in Practice IV (pp. 403-413). Elsevier.