Mini-AutoML

Jan Kwiecień, Filip Mieszkowski, Stanisław Kurzątkowski

1 Wstęp

Celem projektu była implementacja systemu Mini-AutoML, który realizuje zadanie automatycznego uczenia maszynowego opierając się na wcześniej wyselekcjonowanych modelach. System po wprowadzeniu danych ma za zadanie: dokonać selekcji najlepszego modelu lub modeli, wytrenować go oraz być wstanie zwrócić predykcje oraz prawdopodobieństwa przynależności do klas dla zbioru testowego.

2 Budowa portfolio modeli

Spośród różnych sposobów na tworzenie portfolio zdecydowaliśmy się na screening modeli. W tym celu wybraliśmy na początku sześć zbiorów danych, na których będziemy opierać dalszą analizę:

- **Adults** - przewidzenie, czy dana osoba zarabia powyżej 50000 USD rocznie ($> 50K$ lub $\leqslant 50K$). Składa się z danych demograficznych, takich jak wiek, wykształcenie (poziom i liczba lat), stan cywilny, zawód, rasa, płeć, kraj pochodzenia, zyski/straty kapitałowe,
- **Credit Default (Default of Credit Card Clients)** - przewidzenie, czy klient nie spłaci zobowiązania w następnym miesiącu. W zbiorze znajdują się same zmienne ilościowe,
- **APS Failure (Scania Trucks)** - przewidzenie czy awaria ciężarówki wynika z usterki systemu APS (klasa pozytywna), czy z innej przyczyny (klasa negatywna). Zbiór charakteryzuje się z dużą ilością braków danych i niezrównoważonymi klasami,
- **Breast Cancer** - klasyczny zbiór medyczny, w którym klasyfikujemy odmiany raka, łagodną lub guza złośliwej. Dane są tylko ilościowe, opisują cechy jąder komórkowych, takie jak: promień, tekstura, obwód, powierzchnia, gładkość, wklęsłość itp.
- **Bank Marketing** - przewidzenie, czy klient zdecyduje się na lokatę terminową. Występują zarówno zmienne jakościowe, jak i ilościowe, takie jak: dane klienta (wiek, praca, stan cywilny, wykształcenie, czy ma kredyt), dane ekonomiczne (wskaźnik zatrudnienia, EURIBOR, wskaźnik cen konsumpcyjnych),
- **Santander (Archive / Customer Satisfaction)** - identyfikacja niezadowolonych klientów na wczesnym etapie (0 - zadowolony, 1 - niezadowolony). W zbiorze mamy aż 200 cech ilościowych.

Następnie ustaliliśmy modele oraz hiperparametry, które będziemy testować. Zdecydowaliśmy się na

- regresję logistyczną ($C : [0.001, 0.01, 0.1, 1.0, 10.0]$, class_weight: [null, "balanced"]),
- linearSVC ($C : [0.001, 0.01, 0.1, 1.0, 10.0]$, class_weight: [null, "balanced"]),

- stochastic Gradient descent (loss: [”hinge”, ”log_loss”], alpha: [1e-06, 1e-05, 0.0001, 0.001], penalty: [”l2”, ”l1”, ”elasticnet”], l1_ratio: [0.15, 0.5]),
- las losowy (n_estimators: [100, 200], max_depth: [null, 5, 10], min_samples_split: [2, 5, 10], min_samples_leaf: [1, 2, 4], max_features: [”sqrt”, ”log2”, 0.5]),
- extratrees (n_estimators: [200], max_depth: [null, 10, 20], min_samples_split: [2, 5, 10], min_samples_leaf [1, 2, 4], max_features: [”sqrt”, ”log2”, 0.5]).

Dla tak wybranych zbiorów i zdefiniowanych modeli oraz ich siatek hiperparametrów, postępujemy dalej następująco:

1. dla każdego ze zbiorów i każdego z modeli przeszukujemy siatkę hiperparametrów (za pomocą RandomizedSearchCV korzystając z 20 iteracji tam gdzie siatka parametrów na to pozwala, w innych przypadkach mniej oraz trzykrotnej kroswalidacji).

   Tabela 1: Przykładowe wyniki walidacji dla regresji logistycznej i zbioru adult.

Class Weight	**C**	**Mean Balanced Accuracy**	**Rank**
balanced	0.1000	0.8186	1
balanced	10.0000	0.8183	2
balanced	1.0000	0.8179	3
balanced	0.0100	0.8087	4
balanced	0.0010	0.7933	5
None	1.0000	0.7667	6
None	10.0000	0.7657	7
None	0.1000	0.7586	8
None	0.0100	0.7358	9
None	0.0010	0.6051	10

2. następnie dla każdej z 30 konfiguracji (6 zbiorów po 5 modeli) wybieramy ten zestaw hiperparametrów, który spisał się najlepiej (największe balanced accuracy). Ponadto dobieramy do tego te zestawy hiperparametrów, których balanced accuracy wyniosło jedynie o dwie setne mniej. Jeśli takie nie występują, łącznie bierzemy 3 najlepsze.
3. następnie tworzymy odpowiednie kubełki, tak aby uzyskać jak najbardziej zróżnicowane modele. Ustaliliśmy następujące kryteria:
   - dla drzew (Random Forest, Extra Trees): depth < 10, depth $10 - 20$, depth $20 >$ oraz min_samples_leaf 1, min_samples_leaf > 1,
   - dla modeli liniowych (Logistic Regression, SVC): C < 0.1, C $0.1 - 1.0$, C > 1.0 oraz class_weight,
   - dla SGD: alpha $<$ 1e-5, alpha $<$ 1e-4, alpha $<$ 1e-3 oraz penalty.

   Po wyborze wszystkich możliwych modeli z punktu drugiego przechodzimy do przydziału do kubełków. Skrypt znajduje najlepszy model. Sprawdza jego kubełek (np. ”Płytki Las”). Dodaje do listy wybranych. Kontynuujemy ten proces, aż wyczerpią się unikalne kubełki lub osiągniemy limit per_pair_cap (czyli możemy mieć maksymalnie 6 modeli z każdej kombinacji zbiór×model). W ten sposób dostajemy zestaw modeli i hiperparametrów do ostatniego kroku,

4. w ostatnim kroku wybieramy po 6 modeli każdego rodzaju według klucza: najpierw na ilu zbiorach sprawdziły się najlepiej, w dalszej kolejności patrzymy na balanced_accuracy (otrzymujemy łącznie 30 modeli). Ostatnie 20 wybierzemy w ten sposób, że wybieramy po kolei najlepszą regresję logistyczną, potem najlepszy SVM itd. W ten sposób otrzymujemy bardzo zróżnicowany zbiór modeli i hiperparametrów. Dla każdego zbioru danych tworzony jest tzw. *fingerprint*.

3 Profilowanie Zbioru Danych i Fingerprinting

Kluczowym elementem meta-learningu jest stworzenie unikalnego wektora cech (tzw. *fingerprint*), który matematycznie opisuje nowy zbiór danych. Wektor ten składa się z 10 wyselekcjonowanych meta-cech.

3.1 Proste statystyki

- **n_samples** - liczba próbek (do obliczeń brany jest logarytm dziesiętny liczby próbek)
- **n_features** - liczba cech (do obliczeń brany jest logarytm dziesiętny liczby cech)
- **n_cat_features** - liczba cech kategorycznych (do obliczeń brany jest logarytm dziesiętny liczby cech kategorycznych)
- **missing_fraction_mean** - średni odsetek brakujących danych

3.2 Landmarkery

Landmarking to technika polegająca na uruchomieniu bardzo szybkich, prostych algorytmów (tzw. landmarkerów) na zbiorze danych i wykorzystaniu ich wyników jako meta-cech. Informują one system o geometrii przestrzeni cech. Użyliśmy trzech landmarkerów:

- **landmark_decision_stump** - landmark związany z najprostszym możliwym drzewem decyzyjnym
- **landmark_naive_bayes** - landmark związany z naiwnym klasyfikatorem bayesowskim
- **landmark_1nn** - landmark związany z algorytmem KNN dla $k = 1$.

3.3 Mary informacyjne

- **mean_mutual_information** - średnia informacja wzajemna
- **class_entropy** - entropia kolumny celu
- **pca_95_components_ratio** - jaki ułamek wszystkich cech jest potrzebny, aby wyjaśnić 95% wariancji

4 Meta-Learning i Wybór Modeli

Posiadając 10-wymiarowy wektor nowego zbioru, system porównuje go z bazą wiedzy. Obliczana jest odległość euklidesowa znormalizowanych wektorów (Z-score). Wagi przypisywane znanym zbiorom są odwrotnością tej odległości. Ostateczny ranking modeli z portfolio jest tworzony jako średnia ważona ich historycznych wyników na podobnych zbiorach.

System wybiera grupę najbardziej obiecujących modeli (ang. *shortlist*), a następnie trenuje je na aktualnym zbiorze treningowym. Tylko 5 najlepszych przechodzi do etapu łączenia (Ensemble).

5 Stacking i Architektura Zespołu

`MiniAutoML` nie poprzestaje na jednym modelu. Tworzy meta-architekturę, która łączy wyniki pięciu najlepszych klasyfikatorów. Aby uniknąć przeuczenia meta-modelu, system generuje macierz predykcji *Out-Of-Fold (OOF)* za pomocą 3-krotnej walidacji krzyżowej (`StratifiedKFold`).

5.1 Kandydaci do Stackingu (Meta-Learners)

Zamiast jednego algorytmu łączącego, testowane są cztery różnorodne meta-modele:

- **Regresja Logistyczna (C=1.0 i C=0.1)**: Klasyczne rozwiązanie dla stackingu. Niższy współczynnik C wymusza silniejszą regularyzację, co jest kluczowe, gdy modele bazowe są silnie skorelowane.
- **Random Forest (max_depth=3)**: Służy do wychwytywania nieliniowych relacji, w których jeden model myli się w specyficznych obszarach danych.
- **MLPClassifier (10 neuronów)**: Niewielka sieć neuronowa

W ostatnim kroku procedury `fit` system weryfikuje na zbiorze walidacyjnym trzy strategie: użycie najlepszego pojedynczego modelu, proste uśrednienie wyników (Average) oraz zastosowanie meta-modelu (Stacker). Strategia o najwyższym wyniku zostaje zapisana w atrybucie `ensemble_mode_` i to ona jest następnie wykorzystywana do ostatecznych predykcji.

6 Podsumowanie

Implementacja oraz działanie systemu Mini-AutoML składają się z dwóch głównych faz:

- **Faza tworzenia portfolio** - na podstawie sześciu wybranych zbiorów danych znajdowane są gotowe modele oraz ich hiperparametry - portfolio modeli
- **Faza działania systemu** - na podstawie tzw. *fingerprint* zbiorów danych wybierane są te arhitektury, które potencjalnie mają szansę dobrze poradzić sobie na dostarczonym zbiorze danych

7 Uwagi

Prawodpodobnie zwiększenie liczby zbiorów danych, na podstawie których tworzone jest portfolio mogłoby prowadzić do większej różnorodności modeli i lepszym działaniu systemu

Literatura

[1] Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. G. (2000). *Meta-learning by landmarking various learning algorithms*. W: Proceedings of the 17th International Conference on Machine Learning (ICML), s. 743–750.