Automatyczne uczenie maszynowe – projekt 2

Magdalena Bartczak, Aleksandra Mach

Spis treści

1 Opis projektu i danych

1.1 Cel projektu

Celem projektu było zbudowanie dwóch klasyfikatorów: jednego modelu „ręcznego" oraz jednego modelu autoMLowego o jak najwyższej wartości miary zrównoważonej dokładności *balanced_accuracy*. Dane, na których budowałyśmy nasze modele to *artificial_train.data* – zbiór treningowy oraz *artificial_train.labels* – etykiety zbioru treningowego.

1.2 Preprocessing

Zbiór treningowy był rozmiaru 2000×501, przy czym ostatnia kolumna zawierała same puste wartości, więc została od razu usunięta. We wstępnej analizie przeprowadziłyśmy następujące kroki:

1. **Sprawdzenie typów zmiennych** – wszystkie zmienne objaśniające były typu *int64*.
2. **Liczba unikalnych wartości we wszystkich kolumnach.**
3. **Liczba braków danych** – w zbiorze treningowym nie pojawiły się żadne braki danych.
4. **Selekcja zmiennych** – wykorzystałyśmy kilka metod selekcji zmiennych.
   (i) W pierwszej kolejności skupiłyśmy się na zbadaniu **korelacji** między zmiennymi. Pojawiło się kilka par mocno skorelowanych zmiennych i postanowiłyśmy usunąć po jednej zmiennej z każdej pary, dla której korelacja co do wartości bezwzględnej była większa bądź równa 0.95. W ten sposób zostało usuniętych 12 zmiennych.
   (ii) **RFE (*Recursive Feature Elimination*)** – procedura selekcji cech, która iteracyjnie eliminuje najmniej istotne cechy z modelu, opierając się na ich wpływie na dokładność predykcji.
   (iii) **Metoda bazująca na informacji wzajemnej** – Klasa *MutualInformationFeatureSelector* wykorzystuje informacje wzajemne do selekcji cech, dobierając te, które mają największy wpływ na zmienną docelową. Wybór cech odbywa się poprzez iteracyjne obliczanie i porównywanie informacji wzajemnej, począwszy od cechy z najwyższą wartością, a następnie dodając kolejne, aż do osiągnięcia określonej liczby cech lub przekroczenia ustalonego progu.

2 Modele zbudowane ręcznie

Postanowiłyśmy przetestować działanie trzech następujących modeli: **Las losowy, XGBoost** oraz **ExtreTrees** po selekcji RFE i po doborze parametrów metodą Random Search. Każdy z modeli był dopasowywany na trzech różnych zbiorach zmiennych wynikających z różnych metod selekcji zmiennych, czyli

- brak selekcji,
- selekcja RFE,
- selekcja MI (informacja wzajemna).

Dla każdego z modeli oraz dla każdego z trzech wymienionych podzbiorów zmiennych treningowych, optymalne hiperparametry były znajdowane dwiema metodami: kroswalidacją przy użyciu funkcji *RandomizedSearchCV* oraz optymalizacją bayesowską. W obu przypadkach metryka, na podstawie której była oceniana jakość dopasowania, została ustawiona na *balanced_accuracy*. Dopasowanie tak wielu modeli z różnymi hiperparametrami miało na celu znalezienie tego, który zagwarantuje nam jak najwyższy wynik miary *balanced_accuracy*.

Wyniki miary *balanced_accuracy* na zbiorze testowym poszczególnych modeli zostały przedstawione w poniższej tabeli.

Tabela 1: Wyniki dla wszystkich ręcznie dopasowanych modeli

Model	**Selekcja**	**Wybór parametrów**	**balanced_accuracy**
Las losowy	brak	Random Search	0.795
XGBoost	brak	Random Search	0.805
ExtraTrees	brak	Random Search	0.84
Las losowy	RFE	Random Search	0.799
XGBoost	RFE	Random Search	0.808
ExtraTrees	RFE	Random Search	**0.852**
Las losowy	MI	Random Search	0.825
XGBoost	MI	Random Search	0.753
ExtraTrees	MI	Random Search	0.745
Las losowy	brak	Optymalizacja bayesowska	0.825
XGBoost	brak	Optymalizacja bayesowska	0.805
ExtraTrees	brak	Optymalizacja bayesowska	0.842
Las losowy	RFE	Optymalizacja bayesowska	0.84
XGBoost	RFE	Optymalizacja bayesowska	0.85
ExtraTrees	RFE	Optymalizacja bayesowska	0.84
Las losowy	MI	Optymalizacja bayesowska	0.817
XGBoost	MI	Optymalizacja bayesowska	0.827
ExtraTrees	MI	Optymalizacja bayesowska	0.84

Z tabeli widać, że obie metody selekcji zmiennych w większości przypadków poprawia wyniki. Z algorytmów najlepsze wyniki dają XGBoost i ExtraTrees, przy czym ExtraTrees średnio wypada lekko lepiej.

Na podstawie tych wyników wnioskujemy, że najlepszy pod względem miary *balanced_accuracy* jest model **ExtraTrees** i on też został wybrany jako ręczny model do przewidzenia prawdopodobieństw przynależności do klasy 1 na danych testowych.

3 Modele AutoMLowe

W ramach projektu wykorzystałyśmy również podejście AutoML, aby porównać jego skuteczność z ręcznie budowanymi modelami. Skorzystałyśmy z dwóch popularnych narzędzi AutoML:

- AutoGluon,
- Autosklearn.

3.1 AutoGluon

AutoGluon to biblioteka opracowana przez Amazon, która automatycznie dopasowuje modele, optymalizuje hiperparametry i przetwarza dane. Jest ona znana z efektywności w konkursach typu Kaggle oraz z szybkiego osiągania wysokiej jakości wyników z minimalnym nakładem pracy ze strony użytkownika.

W naszym projekcie wykorzystałyśmy AutoGluon do automatycznego trenowania różnych modeli, w tym głębokich sieci neuronowych, lasów losowych oraz innych algorytmów ensemble. AutoGluon automatycznie przeprowadził selekcję cech oraz tuning hiperparametrów, co pozwoliło na uzyskanie równie dobrych wyników w stosunkowo krótkim czasie.

3.2 AutoSklearn

AutoSklearn to kolejne narzędzie AutoML, które opiera się na bibliotece scikit-learn. Jest ono szczególnie znane z efektywnego wykorzystania technik meta-learningu oraz optymalizacji bayesowskiej do wyboru najlepszych modeli i ich hiperparametrów.

W naszym przypadku, autosklearn został użyty do automatycznego przetestowania szerokiej gamy modeli klasyfikacyjnych dostępnych w scikit-learn oraz do optymalizacji ich hiperparametrów. Dodatkowo, autosklearn automatycznie wykonał procesy takie jak imputacja brakujących danych i skalowanie cech, co jest kluczowe w przetwarzaniu zbiorów danych.

W celu dobrania jak najlepszych ustawień dla funkcji *AutoSklearnClassifier*, sprawdziłyśmy (ręcznie) różne wartości możliwych parametrów, a najwyższy wynik ustawionej metryki *metric* = "balanced_accuracy" udało nam się uzyskać dla *initial_configurations_via_metalearning* = 70, *resampling_strategy* = "cv-iterative-fit". I ten model został wybrany jako nasz najlepszy model AutoMLowy.

3.3 Wyniki modeli AutoMLowych i ogólne wnioski

Przeanalizowałyśmy wyniki uzyskane przez AutoGluon i Autosklearn, porównując je z ręcznie budowanymi modelami. W obu modelach ustawiłyśmy opcję, że miarą, na podstawie której ma być wybierany najlepszy model jest *balanced_accuracy*. Oba narzędzia AutoML wykazały się wysoką skutecznością, prezentując wyniki zbliżone lub przewyższające te osiągnięte przez modele ręczne. W szczególności, model wygenerowany przez AutoGluon osiągnął *balanced_accuracy* równą 0.83, podczas gdy model z Autosklearn uzyskał wynik 0.87.

Tabela 2: Wyniki modeli AutoMLowych

Model AutoML	**Wybrane modele**	**balanced_accuracy**
AutoGluon	WeightedEnsemble_L2	0.867
AutoSklearn	Random Forest, Gradient Boosting	**0.87**

Te wyniki pokazują, że podejście AutoML może być skuteczną alternatywą dla tradycyjnych metod budowy modeli, zwłaszcza w sytuacjach wymagających szybkiego prototypowania i minimalnego zaangażowania w tuning hiperparametrów. Wynik uzyskany przez AutoSklearn jest najwyższym wynikiem jaki udało nam się uzyskać. Mimo, że najlepszy model ręczny zwracał nam wynik niewiele

mniejszy, bo ok. 0.85, to wymagał większej ilości pracy przy preprocessingu, skonstruowania odpowiedniej siatki hiperparametrów i większego czasu potrzebnego do przeprowadzenia kroswalidacji, która te optymalne hiperparametry wybierze. Wobec tego, przy analizowanych przez nas danych, warto było wykorzystać modele AutoMLowe. Ogólnym wnioskiem z projektu też jest to, że warto pamiętać o możliwościach, które mogą nam dać modele AutoMLowe. Nawet jeśli nie zwracają idealnego wyniku, to na pewno są w stanie zasugerować jakie "ręczne" modele i ewentualnie jakie wartości ich parametrów mogłyby pasować do rozpatrywanych przez nas danych.