AutoML - raport nr 2

Aleksandra Buchowicz, Filip Pazio

1 Wstęp

Celem zadania było zaproponowanie metody o jak największej mocy predykcyjnej. Do dyspozycji był sztucznie wygenerowany zbiór danych, w którym zostały ukryte zmienne istotne.
Model należało przygotować:

- ręcznie, wybierając rodzaj modelu i hiperparametry,
- wykorzystując frameworki AutoMLowe.

Dokładności modeli miały być mierzone za pomocą miary `balanced accuracy`.

2 Dane

Dane, składały się na zbiory:

- artificial_train.data zawierający 2000 wierszy, 500 kolumn o wartościach o typie `int`, stanowiący dane treningowe;
- articial_train.labels zawierający 2000 wierszy, wartości $\{-1, 1\}$, stanowiący zmienną odpowiedzi dla danych treningowych;
- articial_test.data zawierający 600 wierszy, 500 kolumn o wartościach o typie `int`, stanowiący dane testowe, na których powinna zostać przeprowadzona predykcja.

3 Modele

3.1 Model AutoMLowy

Zdecydowaliśmy się na stworzenie modelu z pomocą frameworku AutoGluon. Zdecydowaliśmy się na niego, ponieważ jest on nam najbardziej znany oraz z uwagi na prostotę jego implementacji i dobre wyniki.
Przed przekazaniem danych do treningu modelu zapisaliśmy je w formie akceptowanej przez model `TabularPredictor`. Oto wyniki otrzymane przez modele w trakcie procesu treningu:

3.2 Model stworzony ręcznie

Konstruując własne modele, na początku wykorzystaliśmy surowe dane bez żadnej selekcji zmiennych, a następnie skorzystaliśmy z selekcji zmiennych z użyciem metody ANOVA. Selekcję hiperparametrów przeprowadziliśmy z wykorzystaniem metody `RandomizedSearchCV`.

3.2.1 ANOVA

Nasza implementacja polega na wykorzystaniu funkcji `SelectKBest`, oraz `f_classif`, obie z pochodzące z biblioteki `scikit-learn`. Funkcja `f_classif` używana jako funkcja celu, pozwala na wykorzystanie metody ANOVA. Zastosowanie selekcji zmiennych wpłynęło pozytywnie nie tylko na wartość `balanced accuracy`, ale i na czas wykonywania kodu.

model	score_val	pred_time_val	fit_time	stack_level	can_infer	fit_order
WeightedEnsemble_L2	0.86	0.55	140.73	2	True	14
XGBoost	0.85	0.02	13.85	1	True	11
CatBoost	0.84	0.01	43.72	1	True	7
LightGBM	0.84	0.01	11.92	1	True	4
LightGBMLarge	0.83	0.03	45.42	1	True	13
LightGBMXT	0.77	0.01	4.3	1	True	3
RandomForestEntr	0.7	0.13	7.81	1	True	6
ExtraTreesGini	0.69	0.12	1.86	1	True	8
KNeighborsDist	0.68	0.05	0.13	1	True	2
KNeighborsUnif	0.68	0.07	0.18	1	True	1
RandomForestGini	0.68	0.12	7.31	1	True	5
ExtraTreesEntr	0.66	0.12	2.84	1	True	9
NeuralNetTorch	0.57	0.02	5.78	1	True	12
NeuralNetFastAI	0.56	0.03	2.94	1	True	10

Tabela 1: Wyniki dla `TabularPredictor` z `eval_metric = balanced_accuracy`.

3.2.2 `GradientBoostingClassifier`

Dla tego modelu rozważaliśmy następującą siatkę parametrów:

Parametr	Typ	Wartości	Optymalne wartości
n_estimators	`int`	[1, 500]	458
learning_rate	`real`	[0.01, 0.99]	0.2891592793734936
max_leaf_nodes	`int`	[2, 50]	42
min_samples_leaf	`int`	[1, 50]	4
max_depth	`int`	[1, 15]	9

Tabela 2: Siatka parametrów dla algorytmu Gradient Boosting z użyciem ANOVA.

Warto wspomnieć, że ten algorytm jest jedynym z rozważanych przez nas algorytmów, który otrzymał równe wyniki przed i po zastosowaniu selekcji zmiennych metodą ANOVA.

3.2.3 `RandomForestClassifier`

Dla tego modelu rozważaliśmy następującą siatkę parametrów:

Parametr	Typ	Wartości	Optymalne wartości
bootstrap	`boolean`	-	`False`
max_features	`real`	[0.01, 1]	0.3732560346542585
min_samples_split	`real`	[0.01, 1]	0.06145870228764971
n_estimators	`int`	[1, 2001]	627

Tabela 3: Siatka parametrów dla algorytmu Random Forest z użyciem ANOVA.

3.2.4 `XGBClassifier`

Dla tego modelu rozważaliśmy następującą siatkę parametrów:

Parametr	Typ	Wartości	Optymalne wartości
n_estimators	`int`	[1, 5000]	427
learning_rate	`real`	[0, 10]	0.38721346767512266
max_depth	`int`	[1, 15]	5
min_child_weight	`real`	[0, 7]	0.6010107840065457

Tabela 4: Siatka parametrów dla algorytmu Gradient Boosting z użyciem ANOVA.

3.2.5 Ensamble

Zaimplementowaliśmy też ensamble opisanych powyżej modeli z wykorzystaniem klasyfikatora `VotingClassifier` z biblioteki `scikit-learn`. Uzyskał on wynik 0.9 na 5% zbioru testowego, co nie stanowiło poprawy w stosunku do wyników otrzymanych przez pojedyncze modele z selekcją zmiennych, z których każdy otrzymał wynik równy 0.9334.

4 Wyniki i wnioski

Poniżej przedstawiono wyniki uzyskane za zbiorze treningowym:

	Ensemble	RF	XGB	GB	AutoGluon
bez selekcji	-	0.836	0.7645	0.823	0.86
ANOVA	0.8385	0.794	0.8365	0.858	-

Tabela 5: Wyniki otrzymane na zbiorze treningowym.

Używając aplikacji do sprawdzania modeli otrzymaliśmy następujące wyniki:

	Ensemble	RF	XGB	GB	AutoGluon
bez selekcji	-	0.9	0.9	0.9334	0.8667
ANOVA	0.9	0.9334	0.9334	0.9334	-

Tabela 6: Wyniki otrzymane na 5% zbioru testowego.

Korzystając z ręcznie zbudowanych modeli udało nam się otrzymać lepsze wyniki, niż korzystając z frameworku AutoMLowego. Wśród nich według nas najlepiej sprawdził się Gradient Boosting z selekcją zmiennych ANOVA, z uwagi na wyniki osiągnięte na zbiorze testowym i treningowym oraz czasie wykonania, i ten model wybraliśmy do konkurencji z AutoGluon. Jednak prostota i szybkość użycia AutoGluona sprawia, że w niektórych sytuacjach może okazać się on preferowanym wyborem. Dla ręcznie stworzonych modeli, wykorzystanie selekcji zmiennych metodą ANOVA pozwoliło nie tylko uzyskać lepsze wyniki, ale i przyśpieszyć czas wykonywania kodu.

Literatura

[1] machinehack.com/story/using-anova-for-feature-selection-in-python