AutoML - Praca domowa 2

Mieszko Mirgos, Anna Rutkiewicz

Styczeń 2024

1 Wstęp

Celem tej pracy domowej było zbudowanie modelu klasyfikującego do jednej z dwóch klas tak, aby uzyskać jak największą moc predykcyjną (wyznaczaną miarą zrównoważonej dokładności *balanced accuracy*) dla sztucznie wygenerowanego zbioru danych *artificial* z ukrytymi istotnymi zmiennymi.

2 Wykorzystane metody i wyniki eksperymentów

W ramach eksperymentu przygotowano dwa modele klasyfikujące. Pierwszy model stworzono ręcznie, samodzielnie dobierając zastosowane klasyfikatory oraz hiperparametry. Drugi model stworzono, wykorzystując poznane frameworki AutoML.

2.1 Model manualny

Zbudowano kilka modeli manualnych, wykorzystując różne metody optymalizacji hiperparametrów, składy ensembli, metody ekstrakcji istotnych cech oraz proporcje podziałów zbiorów na treningowe i testowe. Tabela 1 przedstawia najważniejsze modele oraz *balanced accuracy score* (w kolumnie *Test score*), które modele te uzyskały na wydzielonych zbiorach testowych.

Dwa pierwsze pipeline'y z tabeli, jako że osiągnęły najlepsze wyniki, przetestowano na udostępnionym API. Uzyskały one Balanced Accuracy Score odpowiednio 0.9667 oraz 0.93. Zdecydowano, że lepszym z ww. dwóch pipeline'ów będzie drugi z tabeli, jako że uzyskał lepszy wynik na zbiorze testowym.

Pipeline osiągający najlepsze wyniki zdefiniowany jest w następujący sposób:

```
pipe = Pipeline([
    ('reduce_dim', 'passthrough'),
    ('clf', VotingClassifier(
            voting='soft',
            verbose=True,
            estimators=[
            ('knn1', KNeighborsClassifier()),
            ('knn2', KNeighborsClassifier()),
            ('et1', ExtraTreesClassifier()),
            ('et2', ExtraTreesClassifier()),
            ('bc1', BaggingClassifier(KNeighborsClassifier(), random_state=4)),
            ('bc2', BaggingClassifier(KNeighborsClassifier(), random_state=4))
            ]
    ))
])
```

Jako moduł 'reduce_dim' zastosowano liniową redukcję wymiarów PCA.

Cały pipeline opakowany został w RandomizedSearchCV, aby znaleźć najlepsze zestawienie hiperparametrów. Określono następującą siatkę hiperparametrów:

Skład ensemble	Ensemble	Train-Test split	Preprocessing	Test score	Optymalizacja hiperparametrów
KNeighborsClassifier x2 ExtraTreesClassifier x2 BaggingClassifier(KNeighborsClassifier) x2	VotingClassifier	80-20	PCA	0.9025	RandomizedSearchCV
KNeighborsClassifier x2 ExtraTreesClassifier x2 BaggingClassifier(KNeighborsClassifier) x2	VotingClassifier	80-20	PCA XGBoost	0.9200	RandomizedSearchCV
KNeighborsClassifier x3 RandomForestClassifier x3 SVC x3 ExtraTreesClassifier x3 XGBClassifier x2	VotingClassifier	67-33	PCA SelectFpr SelectFdr	0.8621	RandomizedSearchCV
KNeighborsClassifier x3 RandomForestClassifier x3 SVC x3 ExtraTreesClassifier x3 XGBClassifier x2	VotingClassifier	80-20	PCA SelectFpr SelectFdr	0.8925	RandomizedSearchCV
KNeighborsClassifier x3 RandomForestClassifier x3 SVC x3 ExtraTreesClassifier x3 BaggingClassifier(KNeighborsClassifier) x3 DecisionTreeClassifier x3	VotingClassifier	67-33	PCA SelectFpr SelectFdr	0.8667	BayesSearchCV
KNeighborsClassifier x2 RandomForestClassifier x2 SVC x2 ExtraTreesClassifier x2 XGBClassifier x2 final estimator: VotingClassifier	StackingClassifier	80-20	PCA SelectFpr SelectFdr	0.8675	BayesSearchCV

Tabela 1: Tabela przedstawia wyniki osiągnięte przez wybrane modele manualne dla testowego zbioru danych wyodrębnionego z treningowego zbioru *artificial_train.data* za pomocą funkcji *train_test_split*.

```
    model_distributions=[
    {
        'clf__knn1__n_neighbors': [3,4,5,6,7],
        'clf__knn2__n_neighbors': [3,4,5,6,7],

        'clf__et1__n_estimators': list(range(100, 501)),
        'clf__et2__n_estimators': list(range(100, 501)),
        'clf__et1__max_depth': list(range(5, 13))+ [None],
        'clf__et2__max_depth': list(range(5, 13))+ [None],

        'clf__bc1__n_estimators': list(range(9,200)),
        'clf__bc2__n_estimators': list(range(9,200)),
        'clf__bc1__bootstrap': [True, False],
        'clf__bc2__bootstrap': [True, False],
    }
]
```

Dla liniowej redukcji PCA również zdefiniowano siatkę hiperparametrów dla RandomizedSearchCV do

przeszukania:

```
    dimgrid=[
    {
        'reduce_dim': [PCA(svd_solver='full')],
        'reduce_dim__n_components': [4,5,6,7,8,499],
    },
]
```

2.2 Model automatyczny

Przetestowane zostały frameworki AutoGluon, AutoSklearn, MlJar oraz TabPFN. Ze względu na słabe reultaty, AutoSklearn oraz MlJar zostały odrzucone na wczesnym etapie testów. Do wyboru kolumn zostały wykorzystane PCA oraz informacje preprocessor oparty o XGBoost. Sprawdzony został również wariant bez wyboru zmiennych ale jego wyniki były na tyle niezadowalające, że został pominięty.

Za najlepszy uznajemy TabPFN+PCA(5) bo pomimo takich samych wyników co AutoGluon+PCA(5) uczył się nieporównywalnie krócej.

Framework	Preprocessor	Train-Test split	Test score (balanced accuracy)	Learning time Limit (hours)
AutoGluon	PCA(5)	80-20	0.9025	4
AutoGluon	PCA(7)	80-20	0.8825	4
AutoGluon	PCA(20)	80-20	0.855	4
AutoGluon	PCA(50)	80-20	0.875	4
AutoGluon	XGBoost	80-20	0.89	4
AutoGluon	PCA(7)	67-33	0.87	4
AutoGluon	PCA(20)	67-33	0.86	4
AutoGluon	PCA(50)	67-33	0.83	4
TabPFN	PCA(5)	80-20	0.9025	None
TabPFN	PCA(5)	67-33	0.89	None
TabPFN	XGBoost	80-20	0.6725	None

Tabela 2: Tabela przedstawia wyniki osiągnięte przez wybrane frameworki, wybraliśmy najlepsze rezultaty *artificial_train.data* za pomocą funkcji *train_test_split*.

3 Wnioski

Na podstawie przeprowadzonych eksperymentów wyciągnięto następujące wnioski:

- Pomimo ich ogólnego dobrego działania, frameworki AutoMLowe nie zawsze są w stanie dorównać modelom stworzonym ręcznie za pomocą odpowiedniego tworzenia ensembli i optymalizacji hiperparametrów. Dla modeli opisanych powyżej, modele manualne osiągnęły znacznie lepsze rezultaty.
- Proporcja podziału zbioru treningowego na podzbiory: treningowy i testowy ma duże znaczenie (w przypadku testowanego zbioru, im większy był wydzielony podzbiór treningowy, tym lepsze wyniki uzyskiwał model).
- Ensemble składający się z mniejszej ilości modeli może dać lepsze wyniki niż bardziej rozbudowany ensemble.
- Dobry preprocessing ma bardzo duże znaczenie dla poprawności predykcji.

4 Bibliografia

1. https://github.com/automl/TabPFN