Homework 2 raport

Marta Boratyn, Zuzanna Ostas

styczeń 2024

Spis treści

1 Wstęp

Celem projektu jest przygotowanie dwóch modeli klasyfikacyjnych o jak najwyższej mocy predykcyjnej. Modele miały być zbudowane w dwóch wersjach: jeden przygotowany "ręcznie", z samodzielnym wyborem typu modelu i jego hiperparametrów oraz jeden przygotowany z wykorzystaniem frameworków AutoMLowych. Zbiór danych do klasyfikacji zawiera 2000 obserwacji opisanych przez 500 zmiennych objaśniających, a zmienna objaśniana jest binarna, o wartościach 1 lub -1.

2 Przygotowanie danych

Dane składają się ze zmiennych numerycznych, w żadnej kolumnie nie występują braki danych. Z tego względu dane nie wymagają znacznego wstępnego przetwarzania.

Obserwacje podzielono na zbiór treningowy i testowy w proporcji 70:30. Następnie na zbiorze treningowym zbadano korelację zmiennych objaśniających i usunięto zmienne o korelacji większej niż 0.9. W ten sposób zmniejszono zbiór zmiennych jedynie o 10 kolumn.

3 Frameworki AutoMLowe

3.1 Autosklearn

Pierwszym frameworkiem wykorzystanym w projekcie jest funkcja AutoSklearnClassifier z pakietu AutoSklearn. W tym przypadku w analizie zostały uwzględnione wszystkie zmienne objaśniające. Rozważamy cztery modele zbudowane na danych z takimi samymi ustawieniami, jednak z czterema różnymi strategiami próbkowania:

- **holdout** – korzysta z podziału 67:22 (treningowy:testowy)
- **cv** – korzysta z kroswalidacji
- **holdout iterative fit** – holdout, ale dopasowuje iteracyjnie
- **cv iterative fit** – cv, ale dopasowuje iteracyjnie.

Ustawienia rozważane w modelu:

```
settings = {  'time':       300
              seed :        2024
              n jobs:       4
              metric: y     balanced accuracy}
```

resampling strategy	train	test	model
holdout	0.83	0.96	extra trees
cv	0.67	1	k nearest neighbors
holdout iterative	0.84	0.96	extra trees
cv iterative	0.89	1	random forest

Tabela 1: Porównanie wyników modeli w zależności od strategii próbkowania

Model ze strategią „cv iterative fit" osiągnął najwyższy balance accuracy na zbiorze walidacyjnym. Uzyskane predykcje 1 klasy zostały również przetestowane w aplikacji na zbiorze testowym, gdzie balanced accuracy wyszło równe 1. Z tego powodu wyniki końcowe zostały oparte na predykcjach tego modelu

4 AutoGluon

Drugim sprawdzonym frameworkiem był AutoGluon i funkcja TabularPredictor. W tym przypadku został zbudowany jeden predyktor z ustawieniami:
eval metric=„balanced accuracy", problem type=„binary".

Balanced accuracy modelu wyniosło 0.83, więc model nie został uznany za optymalny.

Bardzo możliwe, że zbudowanie modeli przy zmienionych parametrach, pozwoliłoby osiągnąć lepsze wyniki, jednak przez wysokie accuracy poprzedniego modelu, nie wgłębiałyśmy się bardziej w ten framework.

5 Modele ręczne

Przy ręcznej budowie przetestowano 3 rodzaje modeli: las losowy, model XGBoost i model ExtraTrees. Dla każdego z nich w pierwszym kroku zbudowano model z domyślnymi hiperparametrami, a następnie dokonano selekcji zmiennych na podstawie ich istotności wskazanych przez każdy z modeli.

5.1 Las losowy

Optymalny model w przypadku lasu losowego został wybrany poprzez losowe przeszukiwanie siatki parametrów metodą *RandomizedSearchCV* z pakietu *scikit-learn* ze względu na dużą liczbę testowanych parametrów. Przeszukiwano następującą siatkę parametrów:

```
param_grid = {   "n_estimators":      [50, 100, 150, 200],
                 "max_depth" :        [5,6,7,8,9,10],
                 "min_samples_leaf":  [50, 100],
                 "max_features":      [0.3, 0.6, 0.8, 1]}
```

5.2 XGBoost

Również dla modelu XGBoost zastosowano wybór hiperparametrów metodą *RandomizedSearchCV*. Testowane parametry:

```
param_grid = {  "n_estimators":      [50, 75, 100, 150],
                "eta" :              [0.1, 0.3, 0.5, 0.7],
                "gamma" :            [3, 5, 10, 15, 20, 30],
                "max_depth" :        [5,6,7,8,9],
                "subsample":         [0.5, 0.7, 0.8],
                "colsample_bytree":  [0.5, 0.6, 0.7, 0.8],
                "min_child_weight":  [0.02, 0.03, 0.04, 0.05]}
```

5.3 ExtraTreesClassifier

Dla modelu ExtraTreesClassifier przetestowano mniejszą liczbę parametrów, dlatego zastosowano metodę *GridSearchCV*. Testowane parametry:

```
param_grid = {  "n_estimators":      [100, 150],
                "max_depth" :        [7, 8, 9, 10],
                "min_samples_split": [10, 20]}
```

5.4 Wyniki

W poniższej tabeli przedstawiono najlepsze wyniki miary *balanced_accuracy* otrzymane dla każdego z modeli na zbiorach treningowych i testowych.

model	train	test
RandomForest	0.80	0.76
XGBoost	0.85	0.79
ExtraTreesClassifier	0.94	0.76

Tabela 2: Porównanie wyników modeli ręcznych

Pomimo nałożenia ograniczeń na parametry związane z "rozrastaniem się" drzew i testowaniu różnych siatek parametrów nie udało się całkowicie uniknąć przeuczenia modeli, przez co wyniki na zbiorach treningowych są lepsze niż na danych testowych.

Warto zaznaczyć, że modele przed selekcją zmiennych osiągały istotnie gorsze wyniki, co znaczy że duża część zmiennych powodowała duże zaszumienie, utrudniające budowę modelu. Prawdopodobnie dalsze testowanie metod selekcji zmiennych pozwoliłoby na osiągnięcie lepszych wyników.

Najlepszym z analizowanych modeli okazał się XGBoost. Został on zbudowany na 122 wyselekcjonowanych zmiennych, a wybrane przez algorytm *RandomizedSearchCV* hiperparametry to: *subsample*=0.8, *n_estimators*=150, *min_child_weight* = 0.02, *max_depth* = 5, *gamma* = 10, *eta* = 0.3, *colsample_bytree* = 0.5.

Alogrytm ten został przetestowany również na 5% zbioru walidacyjnego, zwracając wartośc *balanced_accuracy* = 0.8.

6 Wnioski

- FrameWorki AutoML-owe dają możliwość przetestowania wielu modeli z wieloma kombinacjami parametrów, co pozwala w krótkim czasie osiągnąć klasyfikacje o wysokiej dokładności.
- W przypadku AutoSklearna odpowiedni wybór parametru resampling strategy jest kluczowy. Na tych samych danych, dla tych samych ustawień różnica balanced accuracy różniła się o ponad 0.2 dla 'cv' i 'cv ierative fit'
- Dla Autosklearn drobne zmiany w danych powodowały brak zbudowania żadnego modelu w czasie 5 minut i losowe przypisanie do klas. Z tego powodu modele były budowane bez usunięcia obserwacji silnie skorelowanych.
- Autoskleran sprawdza wiele klasyfikatorów przez co można porównać jak różne metody uczenia maszynowego radzą sobie z tym problemem.
- Najczęściej wybieranym modelem był ExtraTreesClassifier, jednak najwyższa dokładność została osiągnięta dla modelu RandomForest.
- AutoGluon ma mniej parametrów do dostosowania i z tego powodu trudniej jest osiągnąć wysoką dokładności
- Ręczna budowa modeli okazała się mniej skuteczna niż frameworki AutoML-owe. Dalsze testowanie modeli i hiperparametrów, a także inny dobór zmiennych objaśniających pozwoliłby na poprawę wyników, jednak metody AutoML pozwoliły na szybsze znalezienie optymalnego rozwiązania.
- Istotnym czynnikiem poprawiającym wyniki przy ręcznej budowie modeli była selekcja zmiennych.