Wydział Matematyki
i Nauk Informacyjnych

POLITECHNIKA WARSZAWSKA

AutoML, Praca Domowa 2

Raport

Adam Majczyk 313420, Szymon Matuszewski 313435

Wersja 1.0

16.01.2024

Spis treści

1 Selekcja predyktorów

Pierwszym krokiem naszego rozwiązania było podzielenie zbioru treningowego na zbiór treningowy oraz walidacyjny w stosunku **4:1**. Podzielono równomiernie względem kolumny celu (*stratify* po kolumnie **label)**.

W zbiorze było 500 kolumn, zatem potrzebna była ekstrakcja cech aby zredukować liczbę wymiarów. Zdecydowaliśmy się na użycie dwóch algorytmów ekstrakcyjnych w trzech konfiguracjach:

1. Regresja logistyczna
2. Boruta
3. Regresja logistyczna + Boruta

1.1 Modele 1 zmiennej - regresja logistyczna

Uwaga - pierwotnie planowaliśmy zrobić to podejście jedynie dla modelu Regresji Logistycznej, stąd nazwa - regresja logistyczna. Finalnie użyto jednak 5 modeli, gdzie tylko 1 z nich to regresja. Dla każdej z 500 kolumn wytrenowano 5 modeli - **LogisticRegression**, **RandomForestClassifier**, **CatBoostClassifier**, **LGBMClassifier**, **XGBClassifier** na domyślnych hiperparametrach. Dla modeli, które wspierały, ustawiono parametr **eval_metric** na **balanced_accuracy**.

Dla każdego modelu wybrano zmienne o wartości **balanced_accuracy** wyższej niż 0.5. Następnie znaleziono część wspólną zmiennych nieodrzuconych przez modele. Finalnie podejście to zostawiło 73 zmienne.

1.2 Boruta

Na zbiorze treningowym przy 500 kolumnach zastosowaliśmy algorytm Boruta z wykorzystaniem *RandomForestClassifier* z parametrami:

- n_jobs=-1
- class_weight='balanced',
- max_depth=5

Algorytm ten pozwolił na ograniczenie zbioru do **20** kolumn.

1.3 Regresja logistyczna + Boruta

Przetestowaliśmy również podwójną ekstrakcję cech. Na **72** kolumny wyekstrachowane przy użyciu algorytmu opisanego w podpunkcie 1.1 nałożyliśmy ponownie algorytm Boruta o takich samych parametrach jak w podpunkcie 1.2. Pozwoliło to uzyskać zbiór treningowy składający się z **11** kolumn.

2 Optymalizacja (model manualny)

Modele z parametrami, które optymalizowaliśmy:

1. CatBoostClassifier - iterations, learning_rate, depth, l2_leaf_reg
2. LogisticRegression - C, penalty
3. RandomForestClassifier - n_estimators, max_depth, max_features
4. LGBMClassifier - n_estimators, learning_rate, max_depth, num_leaves
5. XGBClassifier - n_estimators, learning_rate, max_depth, gamma

Każdy model przeprocesowaliśmy dla 3 zestawów zmiennych objaśniających na dwóch algorytmach poszukiwań hiperparametrów: **Grid Search** oraz **optuna**. W ten sposób przetestowaliśmy 6 podejść dla 5 algorytmów predykcyjnych. Te podejścia to:

1. Regresja logistyczna + Grid Search
2. Boruta + Grid Search
3. Regresja logistyczna + Boruta + Grid Search
4. Regresja logistyczna + optuna
5. Boruta + optuna
6. Regresja logistyczna + Boruta + optuna

W algorytmach optymalizujących hiperparametry używaliśmy **crosswalidacji** oraz metryki ewaluacji **balanced_accuracy**.

3 AutoGluon

3.1 Wybór pakietu

W celu wytrenowania modelu za pomocą pakietu automatycznego uczenia maszynowego wybrano pakiet **AutoGluon**. Skorzystano z obiektu **TabularPredictor**.

3.2 Podział na zbiór treningowy i testowy

Dostarczony zbiór 2000 obserwacji podzielono na zbiór treningowy i testowy w proporcji 80-20. Upewniono się, że oba zbiory zawierają tą samą proporcję zmiennej objaśnianej (wykorzystano opcję **stratify** dla kolumny **label**).

3.3 Przygotowanie zmiennych

Nie dokonywano preprocessingu zmiennych.

3.4 Trenowanie modeli

Zbiór treningowy przekazano do **TabularPredictor**. Ustawiono metrykę ewaluacji na **balanced_accuracy**. Ograniczono czas treningu do 10 minut. Wytrenowano tak przygotowany obiekt **TabularPredictor**.

4 Wyniki

4.1 Optymalizacja (model manualny)

4.2 AutoGluon

Ranking modeli na zbiorze testowym przedstawiono w Tabeli 1

model	**balanced_accuracy**	**czas trenowania [s]**
WeightedEnsemble_L2	0.8525	15.7922
CatBoost	0.8475	2.6999
LightGBM	0.8350	2.7573
LightGBMLarge	0.8350	10.1192
LightGBMXT	0.7700	2.6904
KNeighborsUnif	0.7225	2.8266
KNeighborsDist	0.7225	0.0261
RandomForestEntr	0.7025	0.6494
RandomForestGini	0.6525	0.7759
ExtraTreesEntr	0.6425	0.3624
ExtraTreesGini	0.6100	0.3701
NeuralNetTorch	0.5875	4.7112
NeuralNetFastAI	0.5650	2.4301

Tabela 1: Wyniki dla automatycznego uczenia maszynowego

4.3 Model manualny

Najlepszym modelem pod względem metryki **balanced_accuracy** na zbiorze walidacyjnym okazał się model o następujących parametrach:

- Algorytm: CatBoostClassifier
- Algorytm selekcji zmiennych: Boruta
- Metoda optymalizacji hiperparametrów: optuna
- 'iterations': 322
- 'learning_rate': 0.15542656215841885
- 'depth': 8
- 'l2_leaf_reg': 9

Okazało się, że metryka **balanced_accuracy** jaką udało nam się uzyskać na zbiorze walidacyjnym wynosi **0.92**. Jest to wynik aż o 6 punktów procentowych lepszy od wyniku uzyskanego przy pomocy AutoGluona.

4.4 Wnioski

Okazuje się, że pakiety AutoML przy prostym wywołaniu bez ustawiania złożonych parametrów nie zawsze dają lepsze wyniki od modeli ręcznie optymalizowanych metodami State-of-Art. W celu realizacji założeń takie pakiety powinny szybko dostosowywać się do coraz lepszych metod redukcji wymiarów, aby takie algorytmy jak Boruta były w nich implementowane.