Warszawa, listopad 2023 r.

Automatyczne uczenie maszynowe

Praca domowa nr 2

Filip Skrzeczkowski
Filip Suchorab

Prowadzące: Anna Kozak, Katarzyna Woźnica

Streszczenie

Niniejsza praca stanowi raport z wykonania drugiej pracy domowej z przedmiotu Automatyczne uczenie maszynowe. Opisujemy w niej postawione przed nami wyzwanie oraz kroki podjęte celem jego realizacji. W szczególności omawiamy dwa modele: dostrojony ręcznie i za pomocą frameworka AutoMl, które udało nam się wytrenować oraz przedstawiamy wyniki ich działania.

1 Wstep

Celem pracy domowej było zaproponowanie metody klasyfikacji, która pozwoli zbudować model o jak największej mocy predykcyjnej. Problem klasyfikacji binarnej rozważano na zadanym wygenerowanym zbiorze danych w którym ukryto istotne zmienne. Jako miarę dokładności modelu przyjęto balanced accuracy.
Przygotowano dwa modele.

1. Model zbudowany ręcznie
2. Model zbudowany przez framework AutoMLowy

2 Zbiory danych

Zadany zbiór danych na którym prowadzono eksperymenty to sztucznie wygenerowany zbiór zawierający 2000 obserwacji. Zbiór zawiera 500 zmiennych objaśniających numerycznych o wartościach całkowitych.

3 Model AutoML

Do realizacji zadania w sposób automatyczny wykorzystano pakiet `mljar-supervised`. Najważniejszymi argumentami, który przemawiały za takim wyborem były prostota użycia oraz dobra jakość modeli.

3.1 Trenowanie

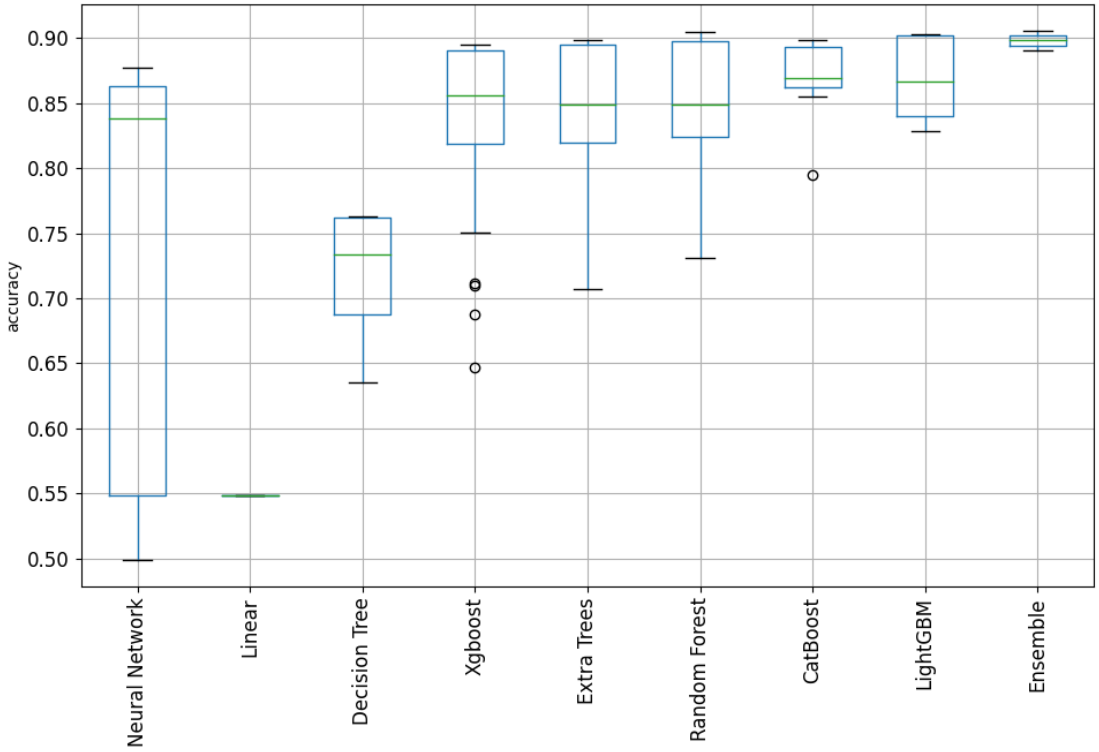
Trenowanie uruchomiono w trybie *compete*, zapewniającym najwyższą jakość modeli, z włączoną możliwością ich stackowania. Były to jedyne nasze ingerencje w działanie pakietu, a same dane nie były przed użyciem poddawane żadnemu preprocessingowi (oprócz tego związanego z samym wczytaniem). Mljar otrzymał budżet pięciu godzin, jednak po ok. dwóch godzinach działania sam zakończył pracę, nie mogąc już skonstruować lepszych modeli. Ze względu na fakt, że pakiet ten nie obsługuje metryki balanced accuracy, wykorzystaliśmy zwykłe accuracy, wiedząc, że dane treningowe są dobrze zbalansowane, co uprzednio sprawdziliśmy.

3.2 Wyniki

Najwyższą wartość accuracy w kroswalidacji uzyskał model `Ensemble_Stacked` - było to $0,906$. W aplikacji sprawdzającej model na 5% danych testowych uzyskał on wynik balanced accuracy na poziomie $0,933$. W tabeli 3.2 przedstawiono 10 najlepszych modeli, natomiast na rysunku 1 uwieczniono boxplot jakości modeli wytrenowanych przez AutoML.

Tabela 1: Ranking modeli wytrenowanych przez AutoML

Nazwa	typ modelu	accuracy	czas trenowania
Ensemble_Stacked	Ensemble	$0,906000$	$25,880000$
100_RandomForest_SelectedFeatures_Stacked	Random Forest	$0,904500$	$26,810000$
25_LightGBM_SelectedFeatures_Stacked	LightGBM	$0,903500$	$16,950000$
66_LightGBM_SelectedFeatures_Stacked	LightGBM	$0,903500$	$17,350000$
89_LightGBM_SelectedFeatures_Stacked	LightGBM	$0,903500$	$16,440000$
91_RandomForest_SelectedFeatures_Stacked	Random Forest	$0,903500$	$27,110000$
74_LightGBM_Stacked	LightGBM	$0,903000$	$23,430000$
25_LightGBM_Stacked	LightGBM	$0,903000$	$22,920000$
76_LightGBM_Stacked	LightGBM	$0,902000$	$23,980000$
27_LightGBM_Stacked	LightGBM	$0,902000$	$25,330000$



Rysunek 1: Boxplot jakości modeli wytrenowanych przez AutoML

4 Model zbudowany ręcznie

4.1 Testowane algorytmy

Do zadania wybrano trzy algorytmy uczenia maszynowego bazujące na gradient boostingu: XGBoost, CatBoost i LightGBM. Dla każdego z modeli dopasowano hiperparametry metodą Random Search. Hiperparametry otrzymane optymalne hiperparametry:

1. XGBoost
    - eval_metric=error
    - estimators=85
    - colsample_bytree=0.602
    - gamma=6,
    - learning_rate=0.0312,
    - max_depth=10,
    - min_child_weight=1,
    - subsample=0.97
2. CatBoost - domyślne parametry
3. LightGBM
    - objective=binary
    - num_leaves=60
    - num_iterations=150
    - min_data_in_leaf=10

- metric=binary_logloss
- max_bin=500
- learning_rate=0.03

4.2 Preprocessing

W ramach preprocessingu zastosowano selekcję zmiennych. Jako metodę selekcji zastosowano metodę rekurencyjnej eliminacji zmiennych. Liczbę zmiennych do selekcji wybrano eksperymentalnie. Selekcję przeprowadzono przy użyciu funkcji CatBoostClassifier.select_features dostępnej w pakiecie CatBoost. Funkcja ta wykorzystuje metodę rekurencyjnej eliminacji zmiennych, decydując którą zmienną usunąć na modelu wytrenowanym algorytmem CatBoost. Eksperymentalnie ustalono, że liczba zmiennych, które należy wybrać aby modele sprawowały się najlepiej wynosi około 20. Tym sposobem funkcją CatBoostClassifier.select_features oraz manualną selekcją wyłoniono 11 zmiennych, które służą jako wejście do modelu a resztę ze zbioru 500 zmiennych odrzucono.

4.3 Wyniki

Na zbiorze treningowym wykonano preprocessing i wytrenowano modele opisane w sekcji 4.1. W celu przetestowania jakości modelu zastosowano pięciokrotną kroswalidację. Jako miarę jakości predykcji modelu zastosowano średnią z balanced accuracy z kroswalidacji. Wyniki przedstawiono w tabeli 2.

Tabela 2: Jakość modeli

Model	Average Balanced Accuracy	
	Before preprocessing	After preprocessing
XGBoost	$0,837$	$0,817$
CatBoost	$0,812$	$0,890$
LightGBM	$0,812$	$0,866$
Ensemble z trzech powyższych	$0,804$	$0,868$

Podczas pracy nad modelami przetestowano także usuwanie outlierów w danych poprzez *quantile filtering* oraz tworzenie enseblingów modeli, jednak nie metody te nie doprowadziły do zauważalnej poprawy jakości modeli.

5 Wnioski

Wytrenowane modele finalnie oferują dosyć dobrą jakość, z przewagą na korzyść AutoMLa, przy wyczerpującym wykorzystaniu metod preprocessingu i dostrajania samych modeli.

Niniejsza praca domowa była cennym doświadczeniem zarówno w ręcznym tworzeniu modeli uczenia maszynowego, jaki i w korzystaniu z AutoMLa, a także pokazała jak trudno jest człowiekowi dorównać temu drugiemu.