Raport - PD2

Agata Kopyt, Zuzanna Kotlinska

January 2024

1 Cel zadania

Projekt polega na przetestowaniu działania dwóch podejść: klasycznego modelowania i modelowania z użyciem frameworków AutoML-owych w problemie klasyfikacji binarnej. Celem eksperymentów jest zbudowanie modelu o jak największej mocy predykcyjnej, mierzonej za pomocą metryki *balanced accuracy*.

2 Dane

Eksperymenty przeprowadzone zostały na sztucznie wygenerowanych danych, zawierających 500 zmiennych objaśniających i 2000 obserwacji, w przypadku zbioru treningowego, oraz 600 obserwacji, w przypadku zbioru testowego.

3 Podejście "ręczne"

Pierwotny pomysł zakładał zmniejszenie ilości zmiennych objaśniających przed przystąpieniem do modelowania, co zostało zrobione na dwa sposoby:

- PCA, z wyjaśnieniem 80% wariancji,
- usunięcie zmiennych o współczynniku korelacji V Cramera większym niż 70%.

W przypadku PCA, zachowane zostały 282 zmienne, jednak wyniki modelowania na okrojonym zbiorze były niezadowalające i zdecydowanie gorsze niż w przypadku pełnych danych. W drugim przypadku, nie zostały znalezione żadne zmienne, które byłyby wysoko skorelowane.

W związku z tym, zdecydowano się przetestować wybór zmiennych objaśniających rekursywną eliminacją cech z walidacją krzyżową (RFECV). Ustalono minimalną liczbę zmiennych do pozostawienia na 400, a jako metrykę użyto emphbalanced_accuracy.

Testowane w przedstawianym podejściu modele to Decision Tree, Random Forest oraz XGBoost. Hiperparametry dostrajane są za pomocą metody *random search*. Poniżej przedstawione są siatki hiperparametrów dla każdego z modeli.

Hiperparametr	**Typ**	**Wartości**
max_depth	integer	[1, 30]
max_features	string	None, log2, sqrt
criterion	string	gini, entropy
splitter	string	best, random
min_samples_split	integer	[1, 60]
min_samples_leaf	integer	[1, 60]

Tabela 1: Zestaw hiperparametrów dla Decision Tree

Hiperparametr	**Typ**	**Wartości**
n_estimators	integer	[1, 2000]
criterion	string	gini, entropy
max_depth	integer	[3, 10]
min_samples_split	integer	[2, 10]
min_samples_leaf	integer	[1, 10]
bootstrap	logical	True, False
max_samples	numeric	[0, 1]

Tabela 2: Zestaw hiperparametrów dla Random Forest

Hiperparametr	**Typ**	**Wartości**
n_estimators	integer	[50, 300]
max_depth	integer	[3, 10]
learning_rate	numeric	[0.01, 0.2]
subsample	numeric	[0.6, 1]
colsample_bytree	numeric	[0.6, 1]
gamma	numeric	[0, 1]

Tabela 3: Zestaw hiperparametrów dla XGBoost

W przypadku modeli Decision Tree zostało sprawdzone 1000 różnych kombinacji hiperparametrów, natomiast w przypadku Random Forest i XGBoosta - 200.

Trening przeprowadzony został w dwóch wariantach:

- bez zastosowania doboru zmiennych objaśniających, dostrojono parametry klasyfikatora i przeprowadzono 5-cio krotną walidację krzyżową
- z doborem zmiennych objaśniających, dostrojono parametry klasyfikatora i przeprowadzono 5-cio krotną walidację krzyżową

W tabeli numer 4 przedstawione są uśrednione wartości metryki *balanced accuracy* uzyskane w walidacji krzyżowej dla każdego z trzech testowanych modeli.

Model	RFECV	uśrednione balanced accuracy
`Decision Tree`		0.781
`Decision Tree`	✓	0.788
`Random Forest`		0.697
`Random Forest`	✓	0.685
`XGBoost`		0.828
`XGBoost`	✓	0.833

Tabela 4: Wartości metryki balanced accuracy w zależności od modelu

Jak można było się spodziewać, najlepsze rezultaty osiągnął XGBoost. Dobór cech dla tego modelu poprawił wynik klasyfikacji o średnio 0.005. Jednak w przypadku testów na 5% zbioru walidacyjnego, model XGBoost bez RFECV osiągnął lepszy wynik, o wartości *balanced accuracy* na poziomie 0.93333 podczas gdy dla modelu z RFECV wartość ta wyniosła 0.9. Zatem za najlepszy klasyfikator przygotowany "ręcznie" wybrałyśmy XGBoost bez RFECV. W celu jego dodatkowego porównania z najlepszym klasyfikatorem utworzonym przy użyciu podejścia AutoMLowego, obliczono także uśrednioną wartość *F1* przy pomocy 5-cio krotnej walidacji krzyżowej: 0.829.

4 Podejście AutoML

W przypadku drugiego podejścia przetestowane zostały AutoML-owe frameworki `MLJAR` oraz `Autogluon`. Dobór zmiennych objaśniających, a także strojenie hiperparametrów i walidacja krzyżowa wykonywane są już przy budowie modelu. Porównując modele AutoGluona z modelami MLJAR zastosowano 2 metryki - *f1* i *balanced accuracy*. Docelową metryką, na której testowany będzie najlepszy model jest *balanced accuracy* dostępna przy trenowaniu modelu AutoGluon. W związku z tym zastosowano dwa warianty budowy modelu:

- z podziałem głównego zbioru treningowego na treningowy i walidacyjny,
- na pełnym zbiorze treningowym.

4.1 MLJAR

Pakiet `MLJAR` oferuje cztery różne tryby użycia, z których wybrany został tryb: *Compete* oraz *Explain*, którego użyto w początkowej fazie modelowania w celu orientacji w czasie treningu modelu i bazowych wynikach.

Tryb *Compete* zastosowano narzucając `eval_metric = 'f1'`, ponieważ pakiet nie posiada *balanced accuracy* w oferowanych metrykach ewaluacyjnych. Dobranym klasyfikatorem był komitet klasyfikatorów zawierający jedynie różne warianty modelu Catboost. W związku z tym zdecydowano wypróbować modyfikacje trybu *Compete*, aby skupić się na algorytmach drzewiastych ze wzmocnieniem gradientowym.

W modyfikacji trybu użyto następujących parametrów:

Parametr	Typ	Wartości
`algorithms`	`string`	"CatBoost", "Xgboost", "LightGBM"
`ml_task`	`string`	'binary classification'
`eval_metric`	`string`	"f1"
`start_random_models`	`integer`	15
`hill_climbing_steps`	`integer`	3
`top_models_to_improve`	`integer`	4

Tabela 5: Zestaw parametrów dla własnego trybu AutoML

W ten sposób najlepsze wyniki uzyskał komitet klasyfikatorów, tym razem zawierający różne warianty wszystkich testowanych algorytmów.

4.2 AutoGluon

Pakiet `AutoGluon` oferuje cztery różne tryby użycia, z których wybrany został tryb: *best_quality* oraz *medium_quality*, którego użyto w początkowej fazie modelowania w celu orientacji w czasie treningu modelu i bazowych wynikach.
Tryb *best_quality* zastosowano narzucając `eval_metric = 'balanced_accuracy'`. Dobranym klasyfikatorem był zarówno przy podziale zbioru treningowego jak i na pełnym zbiorze był `WeightedEnsemble_L2`. Jednak po sprawdzeniu członków komitetu, okazało się, że dla pełnego zbioru, klasyfikator `WeightedEnsemble_L2` był tożsamy z `CatBoost_BAG_L1`.

4.3 Wyniki

W tabeli 6 przedstawiono wartości metryki *f1* dla wytrenowanych modeli MLJARowych i AutoGluonowych oraz *balanced accuracy* dla modeli Autogluonowych.

Najlepsze rezultaty dla modelu tworzonego automatycznie osiągnieto przy użyciu MLJAR Modified Compete.W przypadku testów na próbce 5% zbioru walidacyjnego, osiągnął on wartości *balanced accuracy* na poziomie 0.9.

Model	pełny zbiór	CV f1	CV ba	test f1	test ba
MLJAR Explain	✓	0.813			
MLJAR Compete		0.725		0.920	0.922
MLJAR Compete	✓	0.881			
MLJAR Modif. Compete		0.737		0.992	0.993
MLJAR Modif. Compete	✓	0.887			
AutoGluon medium			0.841	0.806	0.809
AutoGluon medium	✓		0.860		
AutoGluon best			0.869	0.836	0.835
AutoGluon best	✓		0.867		

Tabela 6: Wartości metryk w zależności od modelu

5 Podsumowanie

Najlepszy klasyfikator uzyskano przy "ręcznym" przygotowaniu modelu. Osiągnął on wynik o 0.03333 lepszy dla *balanced accuracy* na próbce 5% zbioru walidacyjnego niż najlepszy klasyfikator przygotowany przez framework AutoMLowy.

Na tej podstawie wnioskujemy, że używanie frameworków AutoMLowych może być dobrym rozwiązaniem, ze względu na niewielką różnicę w wynikach, zwłaszcza gdy oba podejścia skutkują wartościami *balanced accuracy* na poziomie rzędu 0.9. Gdy jednak nawet niewielka poprawa wyniku ma szczególne znaczenie dla modelowanego problemu, "ręczne" przygotowanie modelu powinno być lepszym rozwiązaniem. Warto jednak w początkowej fazie modelowania zastosować framework AutoMLowy z takim trybem jak np. *Explain* w MLJAR, aby w szybki sposób uzyskać ogólny ogląd problemu.