Raport do Pracy Domowej 2

Adrianna Wieczorek Joanna Witczak

16 stycznia 2024

Spis treści

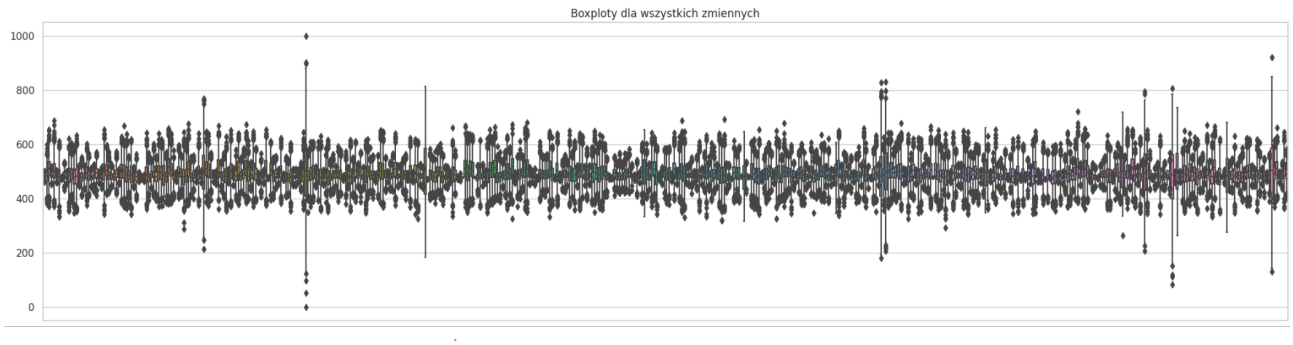
1 Wstęp

W ramach projektu porównałyśmy wyniki otrzymane przez model zbudowany za pomocą biblioteki służącej do automatycznej budowy klasyfikatora, z tymi otrzymanymi przy ręcznej kalibracji. Rozważany był problem klasyfikacji. Dane wykorzystane przy budowie modelu zostały w sposób sztuczny. Rozważanych było 500 zmiennych.

2 Model budowany ręcznie

2.1 Wstępna obróbka danych

Pracę rozpoczęłyśmy od szybkiej analizy eksploracyjnej poprzez wykresy skrzynkowe, aby sprawdzić, czy któraś zmienna dominuje wartościami nad innymi. Po analizie wykresów stwierdziłyśmy, że atrybuty mają podobne skale, więc zrezygnowałyśmy z standaryzacji.
Analiza skrzynkowych wykresów sugerowała obecność outlierów. Wykorzystałyśmy metodę rozstępu międzykwantylowego, co pozwoliło zidentyfikować 748 rekordów z obserwacjami odstającymi. Zamiast usuwać je, zdecydowałyśmy się zamienić skrajne wartości na odpowiednie kwantyle.



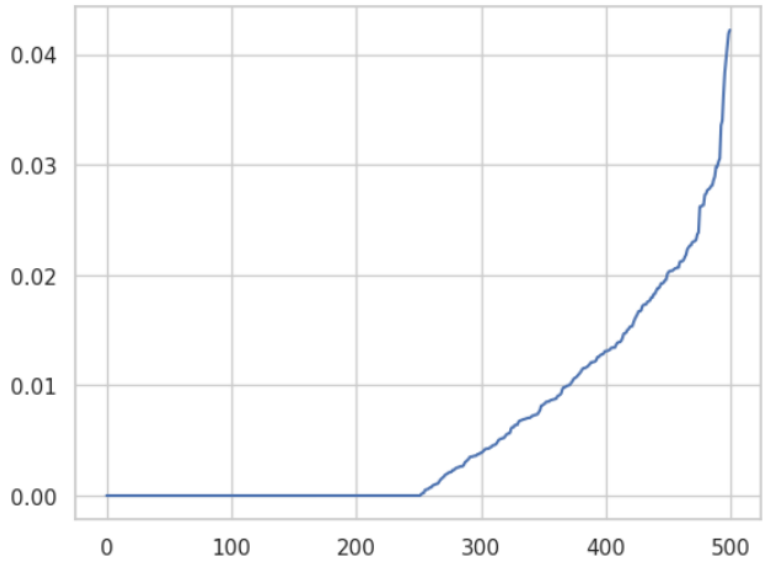
Rysunek 1: Boxploty dla części zmiennych

2.2 Selekcja zmiennych

Do selekcji zmiennych wykorzystałyśmy niezależnie od siebie kilka metod:

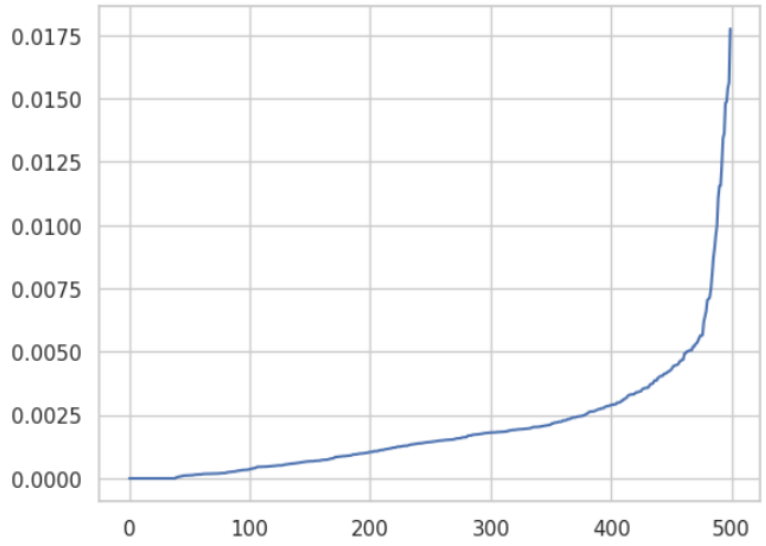
- Teoria informacji - Informacja wzajemna,
- Boruta,
- Współczynnik rang Spearmana,
- RandomForestRegressor
- XGBoost,
- SelectKBest.

Przy użyciu informacji wzajemnej ustaliłyśmy ograniczenie, wybierając tylko zmienne o informacji wzajemnej większej niż 0.025, co dało nam 25 zmiennych. Rozważałyśmy usunięcie zmiennych o wysokim współczynniku korelacji, ale takich nie było.



Rysunek 2: Wykres posortowanej rosnąco informacji wzajemnej dla zmiennych

Współczynnik rang Spearmana jest sposobem badania współczynnika korelacji w sposób czuły na zależności. Za pomocą tego algorytmu wybrałyśmy zmienne skorelowane ze zmienną odpowiedzi w stopniu silniejszym niż 0.1.
W przypadku algorytmu XGBoost postawiłyśmy warunek, że stopień istotności ma być większy niż 0.0075.



Rysunek 3: Wykres istotności zmiennych dla algorytmu XGBoost

Algorytm KBest wybrał 20 najlepszych zmiennych.
Ustalając progi dla większości metod oparłyśmy się na analizach graficznych, pozostałe dokonują selekcji zmiennych w sposób automatyczny.
Do budowy modelu wykorzystałyśmy te zmienne, które zostały wybrane przez co najmniej dwie metody - 17 atrybutów.

2.3 Budowa modelu

Przy budowie modelu rozważyłyśmy następujące klasyfikatory: RandomForest,GradientBoosting,SVM.
W kolejnym kroku przeprowadziłyśmy optymalizację hiperparametrów dla każdego algorytmu

za pomocą RandomizedSearchCV. W ramach naszego eksperymentu rozważyłyśmy przeszukiwanie siatki w kolejno 50, 50 i 100 krokach. Do oceny jakości modeli wykorzystywana była miara balanced accuracy.

Tabela 1: Siatka hiperparametrów dla Random Forest

Parametr	**Wartości**
model__n_estimators	[20, 50, 100, 200, 500, 700, 1000, 1500, 2000, 2100, 2500, 3000]
model__min_samples_split	[1e-5, 1e-4, 0.0001, 0.001, 0.01, 0.1]
model__warm_start	[True, False]
model__max_depth	[10, 30, 50, 70, 100, 150, 200, 500, 1000]
model__min_samples_leaf	[1,2, 3, 5, 10, 15, 20]

Dla Gradient Boosting rozważałyśmy następujące hiperparametry:

Tabela 2: Siatka hiperparametrów dla Gradient Boosting

Parametr	**Wartości**
model__n_estimators	[10, 50, 100, 150, 500, 1000, 1500, 2000, 2500]
model__subsample	[1e-5, 0.005, 0.01, 0.05, 0.1, 0.5, 0.7, 0.9, 1]
model__max_depth	[10, 15, 50, 100, 150, 500, 1000, 1500, 2000, 2500]
model__min_samples_leaf	[1, 3, 5, 10, 15, 20]
model__loss	['log_loss', 'exponential']

Tabela 3: Siatka hiperparametrów dla SVM

Parametr	**Wartości**
model__C	[1e-4, 1e-3, 0.001, 0.005, 0.01, 1, 10, 20, 50, 100, 200, 500, 1000]
model__kernel	['rbf', 'sigmoid', 'poly']
model__gamma	[1e-4, 1e-3, 0.01, 0.1, 0.5, 1, 5, 20, 50, 100]
model__tol	[1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 0.1]

2.4 Wyniki

Jako ostateczny model wybrałyśmy ten cechujący się najwyższym balanced accuracy wyliczonym na zbiorze testowym. Okazał się nim być RandomForest z parametrami 'warm_start': False, 'n_estimators': 2500, 'min_samples_split': 1e-05, 'min_samples_leaf': 1, 'max_depth': 200. Balanced accuracy obliczone dla tego modelu na zbiorze testowym wyniosło 0.88.

3 Model przy użyciu frameworka AutoML

Rozważyłyśmy następujące frameworki AutoML:

- AutoSklearn 1.0,
- AutoSklearn 2.0,
- AutoGluon.

Dla każdego ustawiłyśmy limit czasu 3600 sekund.

3.1 AutoSklearn 1.0

Do klasyfikatora zastosowałyśmy ziarno (seed) równe 42 i wartość parametru n_jobs równą 4. Dodatkowo zdecydowałyśmy się na resampling_strategy równe "holdout" - podział zbioru w proporcji 67:33 (treningowy:testowy).

AutoSklearn 1.0 osiągnął balanced accuracy w przybliżeniu równe 0.85 na zbiorze testowym. Stworzył zespół złożony z lasów losowych:

Tabela 4: Otrzymany zespół modeli AutoSklearn 1.0

Rank	**Ensemble Weight**	**Type**	**Cost**	**Duration**
16	0.04	Random Forest	0.153519	69.785877
11	0.04	Random Forest	0.144071	59.766010
17	0.06	Random Forest	0.168771	21.749009
7	0.02	Random Forest	0.144028	58.788337
9	0.04	Random Forest	0.144056	51.165194
5	0.12	Random Forest	0.138353	64.118033
3	0.02	Random Forest	0.134522	45.726043
1	0.16	Random Forest	0.128833	49.774744
13	0.02	Random Forest	0.145943	63.448480
10	0.08	Random Forest	0.144056	55.365121
4	0.02	Random Forest	0.138338	46.425425
6	0.08	Random Forest	0.138367	63.560133
14	0.06	Random Forest	0.145957	41.588484
12	0.10	Random Forest	0.145914	62.850531
15	0.02	Random Forest	0.151618	50.406034
2	0.06	Random Forest	0.132692	52.159780
8	0.06	Random Forest	0.144042	43.502542

3.2 AutoSklearn 2.0

Do klasyfikatora zastosowałyśmy ziarno (seed) równe 42 i wartość parametru n_jobs równą 4. AutoSklearn 2.0, w przeciwieństwie do AutoSklearn 1.0 automatycznie wybiera resampling_strategy. Została wybrana cv-iterative-fit z argumentem folds = 10.
AutoSklearn 2.0 osiągnął balanced accuracy równe 0.84 na zbiorze testowym. Wybrał 2 modele Extremely Randomized Trees, 1 model lasu losowego, 1 model SGD (Stochastic Gradient Descent) oraz 1 model passive_aggressive do końcowego zespołu:

Tabela 5: Otrzymany zespół modeli AutoSklearn 2.0

rank	**ensemble_weight**	**type**	**cost**	**duration**
3	0.02	extra_trees	0.188154	114.713903
2	0.02	extra_trees	0.178153	121.770025
5	0.86	sgd	0.458132	42.593864
1	0.04	random_forest	0.161208	1440.145208
4	0.02	passive_aggressive	0.414920	46.048432

3.3 AutoGluon

Do klasyfikatora użyłyśmy parametru presets = best_quality, aby otrzymać State-of-the-art (SOTA) jakość modelu.
AutoGluon w osiągnął balanced_accuracy równą w przybliżeniu 0.843. W poniższej tabeli prezentujemy fragment końcowych wyników:

Tabela 6: Wyniki AutoGluon

Model	**score_test**	**score_val**	**pred_time_test**	**pred_time_val**
CatBoost BAG L1	0.842659	0.861851	0.123305	0.182982
WeightedEnsemble_L2	0.842659	0.863104	0.690010	1.008118
LightGBM_r131_BAG_L1	0.842509	0.851230	0.111198	0.085011
CatBoost_r177_BAG_L1	0.837709	0.853749	0.124973	0.198139
LightGBM_BAG_L1	0.832458	0.835002	0.147030	0.076413
LightGBMLarge_BAG_L1	0.830033	0.836863	0.159840	0.092669
XGBoost BAG_L1	0.825033	0.829360	0.221251	0.155648

3.4 Wyniki

Jako ostateczny framework AutoML wybrałyśmy ten, cechujący się najwyższym balanced accuracy wyliczonym na zbiorze testowym. Okazał się nim być AutoSklearn 1.0. Balanced accuracy obliczone dla tego modelu na zbiorze testowym wyniosło w przybliżeniu 0.85.

4 Wnioski

Model zbudowany ręcznie osiągnął lepsze rezultaty niż ten stworzony przy użyciu frameworka AutoML.