Automatyczne uczenie maszynowe - projekt nr 2

Iza Danielewska, Patrycja Żak

Spis treści

1 Wprowadzenie

1.1 Cel projektu

Celem projektu jest opracowanie skutecznej metody klasyfikacji na sztucznie wygenerowanym zbiorze danych, gdzie istotne zmienne są ukryte, w celu zbudowania modelu o maksymalnej mocy predykcyjnej. Klasyfikacja ma obejmować dwie klasy, a jakość modelu będzie mierzona za pomocą zrównoważonej dokładności (*balanced accuracy*). Projekt zakłada przygotowanie dwóch wariantów modelu:

- ręczny, gdzie należy wybrać rodzaj modelu oraz hiperparametry,
- przy użyciu frameworków AutoMLowych.

1.2 Zbiory danych

W projekcie wykorzystano zbiór danych `artificial`, który został sztucznie wygenerowany oraz nie posiada żadnej interpretacji lub opisu zawartych zmiennych. Zestaw zawiera 500 zmiennych objaśniających oraz został odgórnie podzielony na zbiór treningowy i testowy. Pierwszy z nich zawiera 2000 wierszy, z którego 200 wykorzystano do zbioru walidacyjnego, zaś drugi z nich posiada 600 obserwacji. Eksperyment odbywał się bez etykiet zbioru testowego, ale korzystano z udostępnionej aplikacji do walidacji na 5% zbioru testowego.

2 Eksperyment

2.1 Model wykonany przez framework AutoMLowy

Do wykonania tej części eksperymentu wybrany został framework `AutoGluon`.
`AutoGluon` działa z surowymi danymi, co oznacza, że nie trzeba wykonywać żadnego wstępnego przetwarzania danych przed dopasowaniem modelu. Autorzy framework'a zdecydowanie zalecają unikanie wykonywania operacji takich jak przypisywanie brakujących wartości lub kodowanie typu one–hot, ponieważ `AutoGluon` ma dedykowaną logikę do automatycznego radzenia sobie z takimi sytuacjami.
Do wykonania modelu została użyta funkcja `TabularPredictor` ze względu na charakter podanych danych (dane tabelaryczne). Oczywiście, uprzednio dane zostały dostosowane do postaci przyjmowanej przez ową funkcję, poprzez połączenie zmiennych objaśniających ze zmienną objaśnianą i ustawienie nazwy zmiennej objaśnianej na `'500'`. W tym przypadku wykorzystywaliśmy cały zbiór treningowy do ucznia modelu, bez podziału na dodatkowy zbiór walidacyjny.
Parametry funkcji `TabularPredictor` zostały określone w celu lepszego dopasowania do danych. Parametr `problem_type`, określający typ problemu, ustawiony został na wartość `'binary'` ze względu na binarą naturę zmiennej objaśnianej, natomiast parametr odpowiedzialny za wybór miary dopasowania `eval_metric` został ustawiony na `'balanced_accuracy'`, zgodnie z założeniami ustalonymi w treści projektu. Dodatkowo, przy dopasowywaniu modelu został użyty parametr `presets` ustawiony na wartość `'best_quality'`, aby uzyskać możliwie najlepszą jakość predykcyjną.

2.2 Model wykonany ręcznie

W modelu ręcznym przeprowadzono szereg eksperymentów w celu odszukania najlepszej możliwej metody uczenia maszynowego. Od początku pracowano z algorytmem `XGBoost`, który uznano za najcelniejszy wybór.
Ze względu na prawie całkowity brak dostępu do etykiet zbioru testowego, ostrożnie wysuwano wnioski na podstawie wyników na zbiorze walidacyjnym. Jednym z pierwszych wyników na 5% zbioru testowego, jedynie po zastosowaniu metody `PCA` do pięciu komponentów, było 0,9. Mimo bardzo wysokiej jakości predykcji, to według autorek, nie gwarantuje to tak samo wysokiego wyniku na całym zbiorze. Postanowiono zagłębić się w dane i odkryć dokładniejszy sposób predykcji nieznanych danych. Przeprowadzono analizę, która zawierała:

- analizę istotności zmiennych,
- zastąpienie wartości odstających poprzez odpowiednie kwantyle (za wartości odstające uznano takie, które nie mieszczą się między 0,03 a 0,97 kwantylem),

- standaryzacje zmiennych
- redukcje wymiarowości poprzez zastosowanie `PCA`,
- regresje `Lasso` oraz `Ridge`.

Badanie istotności zmiennych zostało rozpoczęte od wykorzystania atrybutu `feature_importances` zawartego w funkcji `XGBoost`. Na podstawie otrzymanych wyników stwierdzono, że należy znacznie zmniejszyć liczbę zmiennych.
Przed rozpoczęciem dalszych działań wykonano badanie unikalności wartości zmiennych, występowania wartości odstających, a także wykonano standaryzację zmiennych.
Następnie wykonano szereg eksperymentów na zbiorach stworzonych przez przekształcenie pierwotnego zbioru. Eksperymenty te dotyczyły wykrycia istotnych zmiennych w modelu. Wykorzystano do tego regresję `Lasso`, a także metodę `PCA`. Metoda `Ridge` dawała mierne wynik, dlatego postanowiono z niej zrezygnować. Rozpatrywano także korelację zmiennych, jednak dawała ona mało informatywne wyniki.
Postanowiono zastosować także optymalizację bayesowską, aby dopasować optymalne hiperparametry w algorytmie `XGBoost` przy różnych kombinacjach przetworzenia zbioru danych, a także dla różnych metod zmniejszania wymiarowości danych. Hiperparametry wykorzystane w optymalizacji bayesowskiej zostały przedstawione w tabelce 1 wraz z zakresami ich wartości.

Tabela 1: Wartości parametrów użyte do optymalizacji bayesowskiej dla modelu `XGBoost`

Parametr	Minimalna wartość	Maksymalna wartość
`'max_depth'`	4	10
`'learning_rate'` (o rozkładzie 'log-uniform')	0,01	1
`'n_estimators'`	100	10000
`'gamma'`	0	1
`'min_child_weight'`	1	10

3 Wyniki eksperymentów

3.1 Wyniki modelu stworzonego przez framework `AutoGluon`

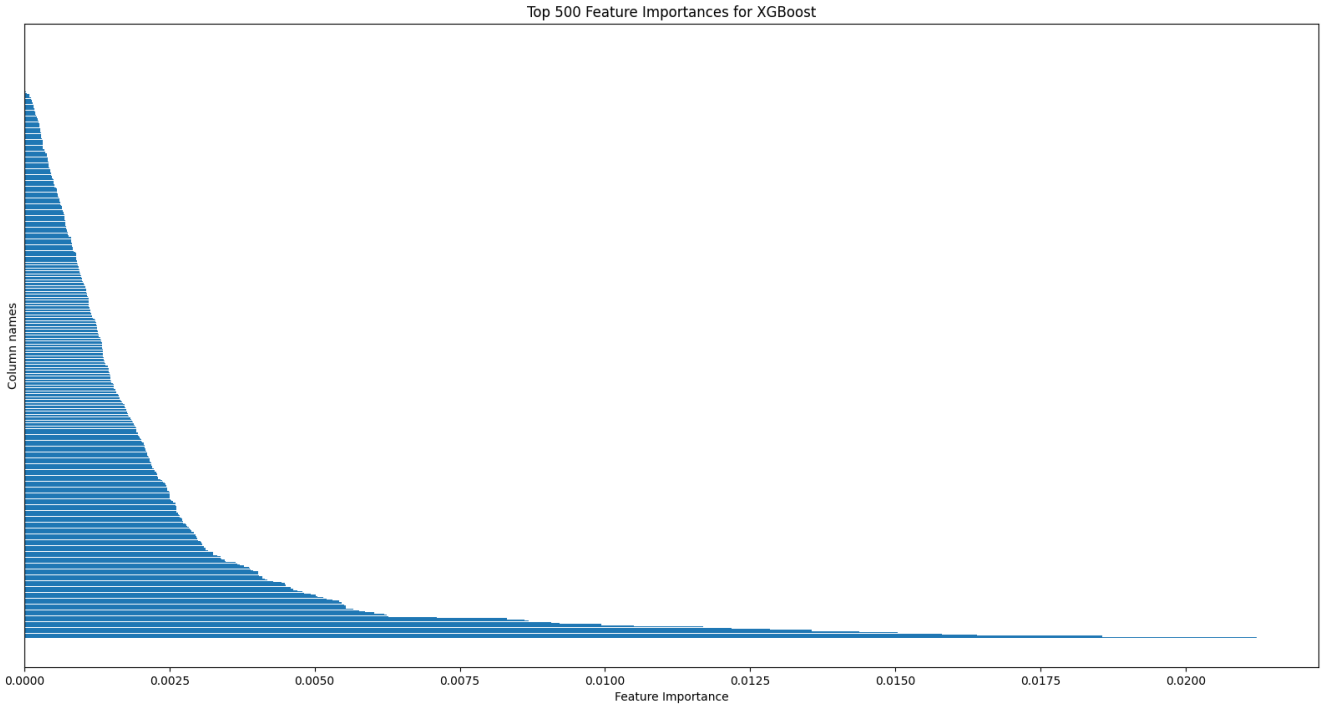
Po dopasowaniu modelu można prześledzić historię uczenia za pomocą atrybutu `fit_summary()`. Można również zauważyć, że najlepszą jakość predykcyjną przy uczeniu modelu uzyskano przez algorytmy m.in., `XGBoost`, `CatBoost` czy `LightGBM`. Można również prześledzić czas dopasowywania poszczególnych modeli w różnych warstwach, a także wszystkie modele wykorzystane w uczeniu. Według raportu najgorzej działają sieci neuronowe, które uzyskują niższe wartości miary dopasowania w porównaniu z innymi metodami.
Obliczając `balanced_accuracy` na zbiorze treningowym, na którym model był uczony otrzymujemy wartość 0,9515, co jest bardzo dobrym wynikiem. Tak wysoka wartość BA jest uzyskiwana dzięki ustawieniu parametru `presets` na `'best_quality'`, dzięki czemu model jest nastawiony na uzyskiwanie jak najlepszej jakości predykcji mierzonej przez miarę Balanced Accuracy.
Korzystając z udostępnionej aplikacji do obliczania BA na 5% zbioru testowego uzyskiwany jest wynik na poziomie 0,9333.

3.2 Wyniki modelu stworzonego ręcznie

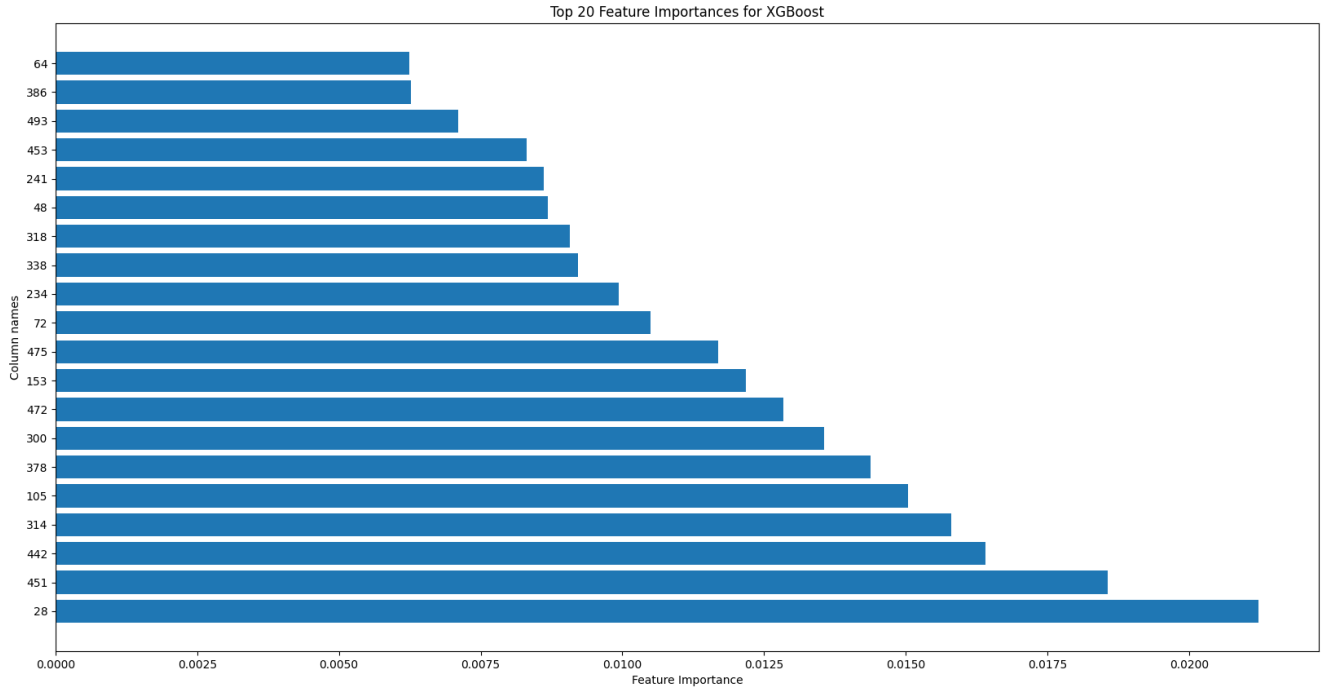
Eksperyment w tym przypadku rozpoczęto od analizy istotności zmiennych, którego wyniki został przedstawione na wykresach 1 oraz 2.

Wykres na rysunku 2 przedstawia 20 najbardziej istotnych zmiennych, wyznaczonych na podstawie atrybutu `feature_importances` w modelu `XGBoost`. Dla wszystkich zmiennych wykres był dość „gładki", co możemy zauważyć na rysunku 1.

Skoki można zauważyć po kilku najbardziej istotnych zmiennych. Może to świadczyć o tym, że do budowania modelu wystarczy o wiele mniej zmiennych niż zostało podanych. Na tej podstawie postanowiono wykonać selekcję. Zanim dokonano selekcji zbiór poddano obróbce poprzez standaryzację. Poza tym rozpatrzono kilka przypadków, w których selekcja była wykonywana na zbiorze, z którego usunięto outliery. Porównanie wyników zostało umieszczone w tabeli 2.



Rysunek 1: Wykres istotności zmiennych dla treningowego zbioru danych bez preprocessingu.



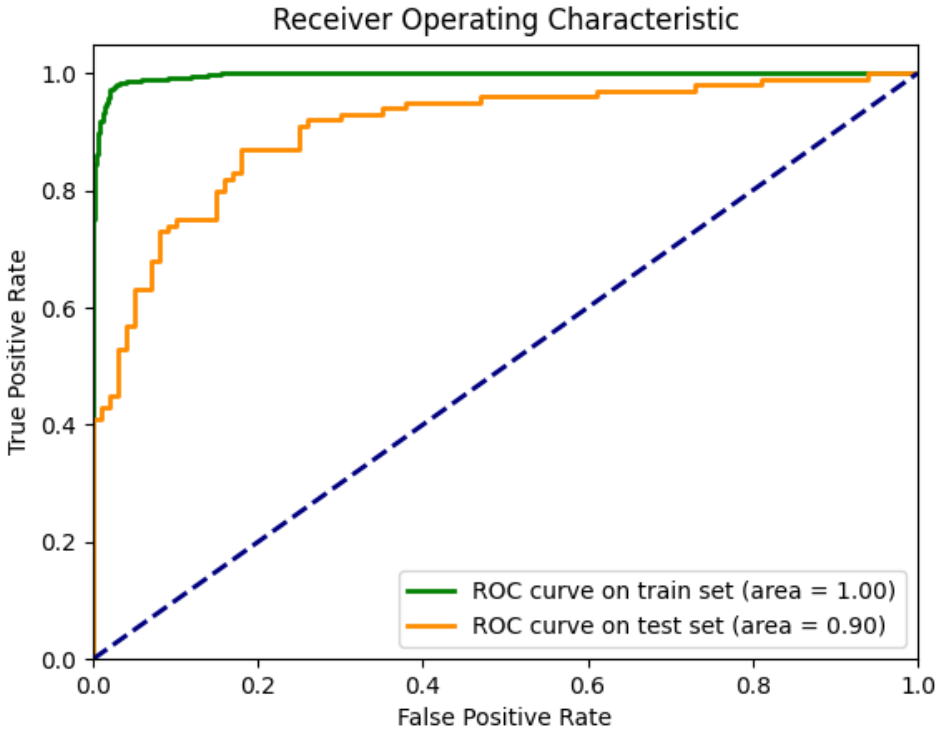
Rysunek 2: Wykres istotności 20 najistotniejszych zmiennych dla treningowego zbioru danych bez preprocessingu.

Możemy zauważyć, że najlepszy wynik uzyskiwany jest dla modelu, w którym wykorzystano `PCA` oraz standaryzację i optymalizację bayesowską. W tym przypadku rozpatrywano zbiór bez usuniętych outlierów. Może to świadczyć o istotnym wpływie wartości odstających na dopasowanie modelu do zmiennych.
Najwyższy wynik jaki uzyskano przy ustalonym stanie losowym, to 0,84. Mimo to zdecydowano, że końcowym modelem zostanie model nr 8. Wynika to z faktu, że przy użyciu aplikacji osiągnął on wynik 0,833, gdzie natomiast model nr 7 osiągnął wynik 0,8.

Tabela 2: Porównanie wartości BA (Balanced Accuracy) dla poszczególnych modeli, gdzie OB oznacza optymalizację bayesowską

Lp.	Model	Usunięte outliery	Standaryzacja	OB	BA
1	`XGBoost`	X	X	X	0,795
2	`XGBoost`	✓	✓	✓	0,800
3	`Lasso + XGBoost`	✓	✓	X	0,620
4	`Lasso + XGBoost`	✓	✓	✓	0,635
5	`PCA + XGBoost`	X	✓	X	0,810
6	`PCA + XGBoost`	✓	✓	X	0,815
7	`PCA + XGBoost`	X	✓	✓	**0,840**
8	`PCA + XGBoost`	✓	✓	✓	0,825

Dla modelu, który uzyskał najlepszy wynik, narysowano również krzywą `ROC`, która jest widoczna na rysunku 3. Wynik ten został uzyskany dla następującej kombinacji hiperparametrów metody `XGBoost`: `[('gamma', 0.0), ('learning_rate', 0.01), ('max_depth', 10), ('min_child_weight', 1), ('n_estimators', 2793)]`.



Rysunek 3: Wykres krzywej `ROC` na zbiorze testowym i treningowym dla modelu nr 7.

4 Podsumowanie

Wartości BA dla AutoMLowego modelu są bardzo dobre, co świadczy o tym, że modele budowane automatycznie potrafią bardzo dobrze dopasowywać się do zbiorów, ale mają jedną wadę jaką jest ich instalacja.
Wartości BA dla modelu budowanego ręcznie są zdecydowanie gorsze niż dla modeli budowanych automatycznie. Wynika to ze względu na złożoność modelu jaka została zastosowana w tym eksperymencie.
Model przy wykorzystaniu pakietu AutoML daje jednak nadzieję na poprawę wyniku modelu ręcznego, ale przy założeniu większej skali eksperymentów. Ze względu na specyfikę danych, bardzo mała zmiana w preprocessingu powodowała dużą różnicę w wyniku stosowanego modelu. To skłoniło do skupieniu się nad pracą i eksploracją tego problemu.
Zakończony projekt zostaje uznany za sukces, ponieważ uzyskano satysfakcjonujące modele o wysokiej predykcji.