

Optymalizacja hiperparametrów xgboost

Dokumentacja wstępna

Przemysław Stawczyk, Piotr Zmyślony

15 kwietnia 2020

Spis treści

1	Treść zadania	2
2	Dane testowe	2
2.1	Analiza danych	2
2.2	Uzupełnienie brakujących danych	3
3	Propozycja rozwiązania	3
4	Funkcja celu	3
5	Sposób mierzenia jakości rozwiązania	3
5.1	Weryfikacja rozwiązania na innych danych	4

1 Treść zadania

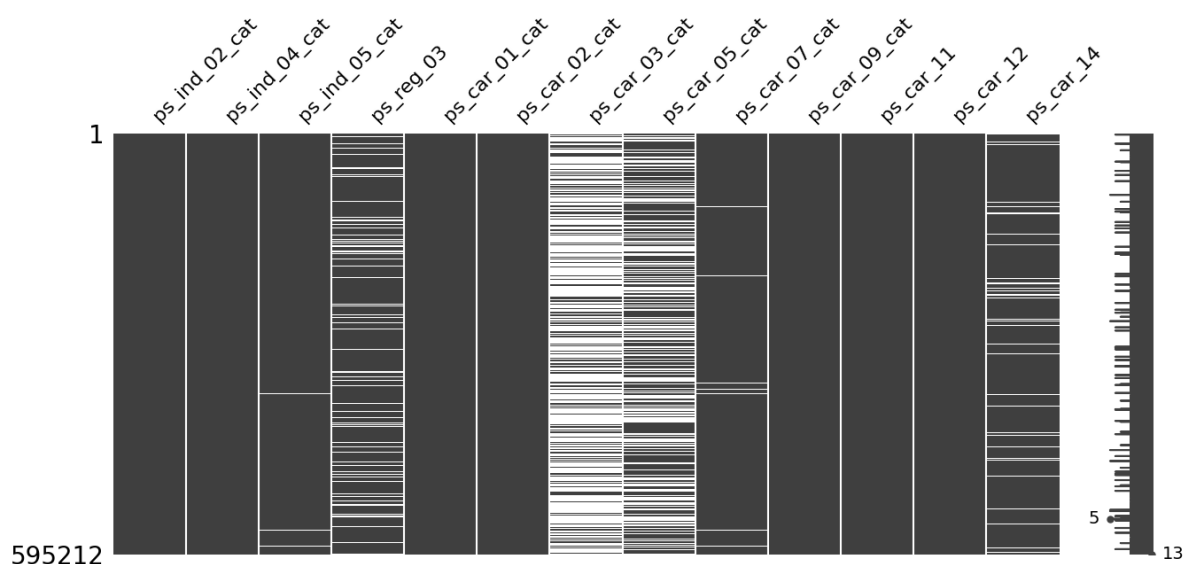
Naszym zadaniem jest przetestowanie różnych algorytmów heurystycznych/populacyjnych w kontekście problemu strojenia hiperparametrów algorytmu xgboost. Problem wyboru hiperparametrów wynika z ich bardzo dużej ilości, co często rozwiązane jest poprzez manualny dobór parametrów klasyfikatora.

Projekt zostanie zrealizowany w języku Python 3+.

2 Dane testowe

Jako dane na których będziemy trenować i testować klasyfikatory przyjęliśmy proponowany zestaw danych <https://www.kaggle.com/c/porto-seguro-safe-driver-prediction>. Zawiera on 57 atrybutów opisujących klientów firmy ubezpieczeniowej i jeden atrybut binarny sygnalizujący, czy w ciągu roku od zawarcia umowy, klient skorzystał z ubezpieczenia.

Rysunek 1: Brakujące atrybuty



2.1 Analiza danych

Po wstępnej analizie danych odkryliśmy, że w zbiorze danych posiadamy około 79% niekompletnych wierszy. Rysunek 1 przedstawia pokrycie niekompletnych atrybutów - jest ich jedynie 13, z czego większość jest wybrakowana w bardzo niewielkim stopniu.

Największym winowajcą jest atrybut binarny `ps_car_03_cat`, którego brakuje aż w 70% wierszy, oraz atrybut `ps_car_05_cat` (brakuje go w 44% przypadków).

Dodatkowo, występuje znaczna dysproporcja między klasami rekordów - tylko 3% wierszy opisuje klientów, którzy skorzystali z ubezpieczenia. Stąd niezbędna będzie interpolacja danych, tak aby ilość rekordów obu klas była równa.

2.2 Uzupełnienie brakujących danych

W związku z powyższym, planujemy uzupełnić brakujące atrybuty na bazie kompletnych wierszy danych. Do tego zastosujemy bibliotekę pythonową *impute*, ale nie będziemy analizować, jaka jest zależność między konkretnymi metodami interpolacji wybrakowanych atrybutów a hiperparametrami trenowanego klasyfikatora - ręcznie wybierzemy tą, która daje najlepsze (i najszybsze) rezultaty.

3 Propozycja rozwiązania

Planujemy zaimplementować następujące algorytmy:

- mutacyjny algorytm wspinaczkowy z tabu.
W 2 wariantach:
 - z prawdopodobieństwem P mutacji jednego (losowego) z parametrów
 - z prawdopodobieństwem P mutacji każdego z parametrów
(w szczególności mogą zmutować wszystkie)

Mutacja byłaby przeprowadzana losując jednostajnie ze zbioru elementów w zadanym promieniu od aktualnego (względny w stosunku do zakresu). W obu wariantach promień mutacji miałby maleć z czasem by od eksploracji przejść do eksploatacji maximum lokalnego.

- przegląd wyczerpujący hipersiatki jako metoda bazowa

Trenowane parametry. Forma hipersiatki

nazwa parametu	zakres
eta	0.2, 0.3, 0.4, 0.5
min_split_loss <i>gamma</i>	0, 1, 2, 3
max_depth	1, 2, 3 ... 16
min_child_weight	1, 2
max_delta_step	0, 1, 2
subsample	0.5, 0.6, 0.7, 0.8, 0.9, 1
colsample_bytree	0.6, 0.8, 1

4 Funkcja celu

Jako funkcję celu przybraliśmy *Average Precision Recall* obliczając wartość funkcji celu jako średnią arytmetyczną skuteczności przypisania predykcji. Planujemy wykorzystać implementację z pakietu *scikit-learn*.

Ta sama funkcja zostanie wykorzystana do oceny jakości finalnych wytrenowanych modeli na zbiorze testowym.

5 Sposób mierzenia jakości rozwiązania

Będziemy porównywać algorytm pod kątem czasu działania względem wyczerpującego przeglądu, analizując czy zysk z szybszego doboru parametrów jest wystarczająco duży, by go stosować dla różnych limitów przejranych kombinacji dla naszych algorytmów.

Zbiór danych zostanie podzielony na 2 podzbiory - uczenia i testowy *uczący odpowiednio większy*. Na zbiorze testowym nie będziemy uczyć i podejmować decyzji - zostanie on wykorzystany do zmierzenia działania algorytmu po dobraniu wszystkich parametrów. Zbiór uczący będzie działać z k-krotną walidacją krzyżową lub podzielony na zbiór uczenia i weryfikacyjny.

5.1 Weryfikacja rozwiązania na innych danych

Końcową wersję naszego algorytmu heurystycznego planujemy przetestować przy użyciu dodatkowego zbioru danych dot. przewidywania bankructwa polskich firm, który analizowaliśmy w [innym projekcie](#).