Optymalizacja hiperparametrów xgboost

Dokumentacja wstępna

Przemysław Stawczyk, Piotr Zmyślony

12 kwietnia 2020

Spis treści

1 Treść zadania

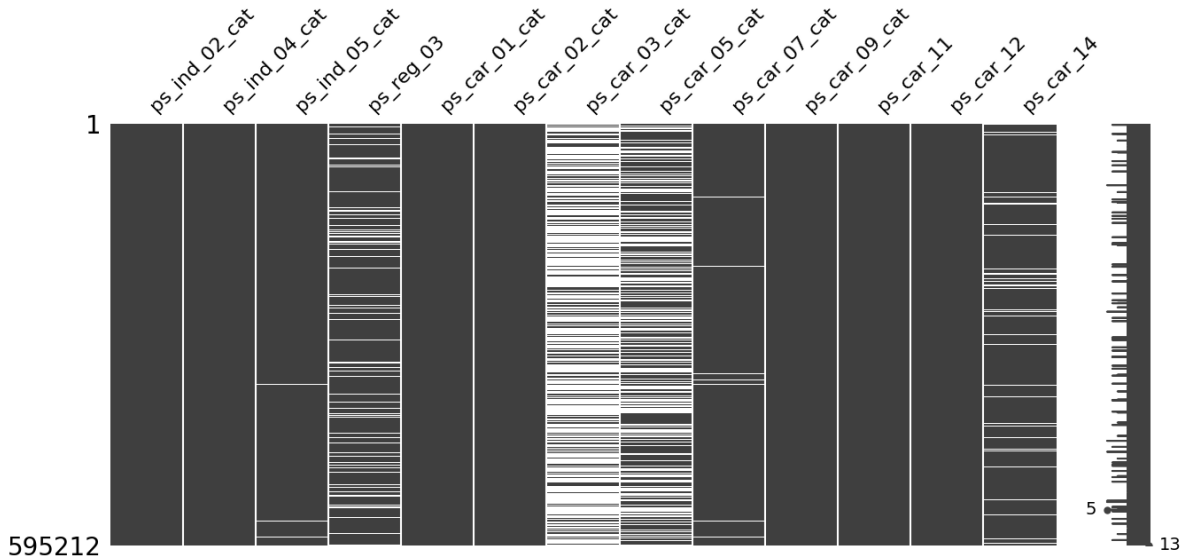
Naszym zadaniem jest przetestowanie różnych algorytmów heurystycznych/populacyjnych w kontekście problemu strojenia hiperparametrów algorytmu xgboost. Problem wyboru hiperparametrów wynika z ich bardzo dużej ilości, co często rozwiązyne jest poprzez manualny dobór parametrów klasyfikatora.

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, wystaępuje 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 *impyute*, 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.

2.3 Alternatywne dane

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

3 Propozycja rozwiązania

Planujemy zaimplementować 2 algorytmy heurystyczne:

- stochastyczny algorytm wspinaczkowy z tabu
- klasyczny algorytm wspinaczkowy z tabu
  [dla parametrów liczbowych przegląda sąsiednich]

oraz metodę bazową : przegląd wyczerpujący.

4 Funkcja celu

Jako funkcję celu planujemy wykorzystać *przewidywany koszt*

```
Expected cost = p(p) × [ p(tp) × benefit(tp) + p(fn) × cost(fn) ]
              + p(n) × [ p(tn) × benefit(tn) + p(fp) × cost(fp) ]
```

gdzie :

- *P(x)* to prawdopodonieństwo x
- *benefit(x)* to zysk z x
- *cost(x)* to koszt/kara za x
- *x* - może oznaczać :
  - *p* - pozytywną predykcję
  - *n* - negatywną predykcję
  - *tp* - pozytywną prawidłową predykcję
  - *fp* - pozytywną fałszywą predykcję
  - *t*n - negatywną prawidłową predykcję
  - *fn* - negatywną fałszywą predykcję

Gdzie przewidywane koszta i straty preedykcji będą możliwe do zmiany jako parametry uruchomienia.

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 przejrzanych kombinacji dla naszych algorytmów.