Raport projektu #1

z przedmiotu Zaawansowane Metody Uczenia Maszynowego

Piotr Podbielski,
grupa dr. Pawła Teisseyre'a

3 maja 2019

Streszczenie

Raport opisuje proces rozwiązywania problemu klasyfikacji danych pochodzących od jednej z firm telekomunikacyjnych do jednej z dwóch klas.

1 Wstęp

Niniejszy dokument powstał w wyniku pierwszego projektu z przedmiotu *Zaawansowane Metody Uczenia Maszynowego* wykładanego na kierunku *Inżynierii i Analizy Danych* w roku akademickim *2018/2019* na wydziale *Matematyki i Nauk Informacyjnych Politechniki Warszawskiej.*

Celem głównym projektu jest identyfikacja klientów, którzy korzystają z oferty handlowej (zmienna class == 1). Wartość class == 0 oznacza, że klient nie skorzystał z oferty.

W pliku *train.txt* znajdują się dane treningowe, które mogą być wykorzystane do budowy modelu. Należy dokonać predykcji dla danych ze zbioru *testx.txt*, każdemu klientowi przypisując prawdopodobieństwo skorzystania z oferty.

2 Zbiór danych

Najważniejsze informacje:

- zbiór treningowy liczy 40000 przykładów,
- zbiór ten posiada 231 cech,
- 164 cech ma conajmniej 15% braków (wartości *NA*),
- 38 cech to cechy będące zahashowanymi ciągami znaków (są to wartości unikalne typu imię, nazwisko, albo też zmienne kategoryczne – brak dokładniejszych informacji).

3 Opis postępowania

1. Załadowanie bibliotek,
2. wczytanie danych,
3. podział danych treningowych na porcje w celu użycia kros-walidacji,
4. transformacja cech,
5. wytrenowanie modeli,
6. wyliczenie prawdopodobieństw dla zbioru z pliku *testx.txt*.

3.1 Załadowanie bibliotek

Model został zaimplementowany w języku *Python* przy użyciu biblioteki *scikit-learn*. Użyto także innych bibliotek pomocniczych, które przedstawiono poniżej:

- `category_encoders` – różne sposoby enkodowania zmiennych kategorycznych,
- `matplotlib` – wizualizacja danych,
- `numpy` – obliczenia na macierzach,
- `pandas` – manipulacja ramkami danych,
- `seaborn` – wysoko-poziomowy interfejs dla `matplotlib`, który pozwala w prosty sposób wizualizować różne ciekawe informacje,
- `tqdm` – pasek postępu,
- `xgboost` – gradient boosting na drzewach.

3.2 wczytanie danych

Z plików `txt` załadowano zbiór zarówno treningowy, jak i ten, na którym ma zostać dokonana predykcja.

3.3 podział danych treningowych na porcje w celu użycia kros-walidacji

W celu późniejszego porównywania modeli zbiór treningowo podzielono na 10 części. Trenowanie i ewaluacja modelu odbywa się w przypadku kros-walidacji 10-krotnie. W każdym kroku pewne 9 części tworzy zbiór treningowe a pozostała część zbiór walidacyjny. Wyniki z wszystkich iteracji zostają uśrednione.

3.4 transformacja cech

W przypadku problemów z dużą liczbą cech, kluczowym elementem jest ich oczyszczenie, transformacja i wybranie.

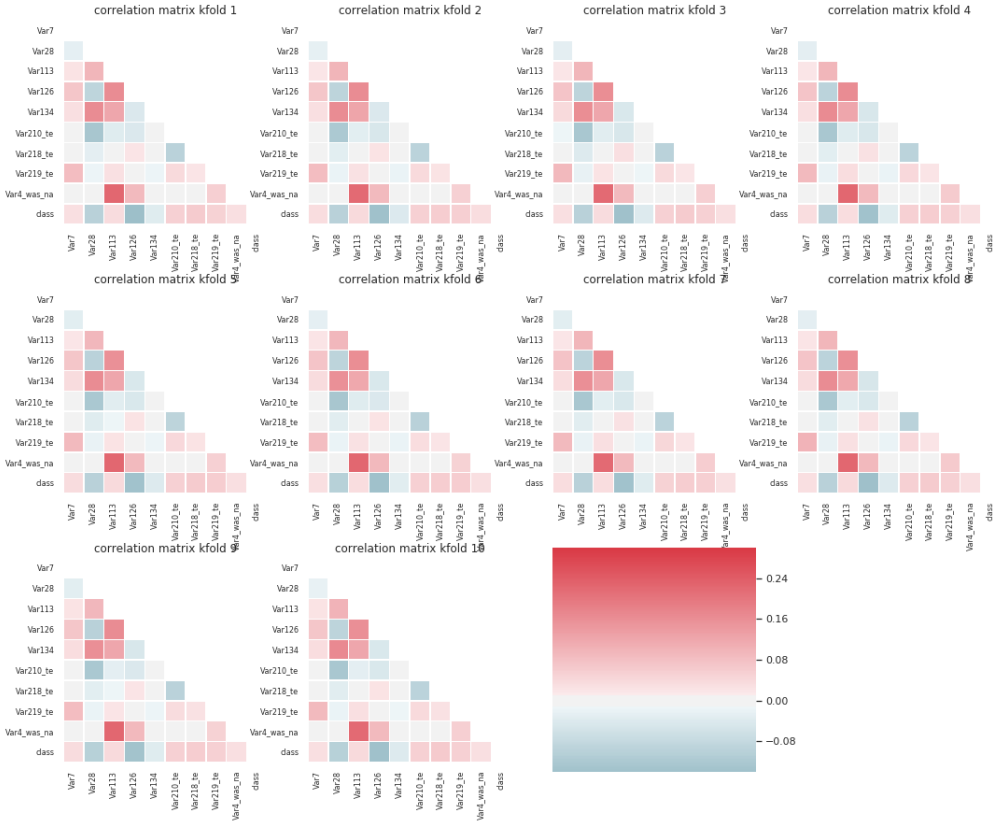
Metodą prób i błędów wyznaczono następujący sposób transformacji cech:

1. Usunięcie cech ze wszystkimi wartościami będącymi brakami (`NA`),
2. usunięcie cech ze zmiennością równą 0 – cechy z tylko jedną wartością zostaną usunięte,
3. usunięcie cech kategorycznych (będącymi hashami), które mają ponad 30 wartości,
4. zakodowanie pozostałych cech kategorycznych sposobem `TargetEncoding`. Cechy są zastępowane mieszanką prawdopodobieństwa aposteriori zmiennej celu pod warunkiem wartości danej cechy i prawdopodobieństwa apriori zmiennej celu względem wszystkich danych treningowych,
5. zamiana cech z frakcją braków powyżej 0.6 na flagę czy dana wartość cechy miała brak (1) czy nie (0),
6. normalizacja danych (odjęcie średniej i podzielenie przez odchylenie standardowe),
7. usunięcie cech, których korelacja ze zmienną celu wynosi mniej niż 0.03,
8. usunięcie z pary skorelowanych cech jednej cechy, której współczynnik korelacji z drugą wynosi ponad 0.25.

Powyższe postępowanie daje najlepsze wyniki.

Każda z porcji zbiorów dla kros-walidacji przechodzi przez powyższy *pipeline*, po czym usuwane są dodatkowo cechy, które nie zawierają się w każdej z porcji. Dzięki temu podczas budowania modeli udział biorą tylko cechy, które zostały wybrane podczas wszystkich z procesów transformacji na porcjach do kros-walidacji.

Docelowo użyte cechy oraz korelacje między nimi dla każdej z porcji kros-walidacji ukazano na rysunku 1.



Rysunek 1: Mapa ciepła przedstawiająca docelowo użyte cechy oraz korelacje między nimi dla każdej z porcji kros-walidacji.

3.5 wytrenowanie modeli

Zgodnie z założeniami zadania przetestowano 4 metody klasyfikacji. Poniżej przedstawiono ich nazwy, konfigurację i uzyskane wyniki (metryka LIFT_{10}) za pomocą kros-walidacji na zbiorach treningowych (9 części) i testowych (1 część):

- `LogisticRegression` – *solver* - `saga`, *C* (parametr regularyzacji) - 0.01, *max_iter* - 10^6, *penalty* (typ regularyzacji) - `l2`; zbiór treningowy - 0.262, zbiór walidacyjny - **0.26**,
- `MLPClassifier` – *early_stopping* - `True`, *hidden_layer_size* - (50), *activation* - `relu`, *max_iter* - 200; zbiór treningowy - 0.294, zbiór walidacyjny - **0.290**,
- `RandomForestClassifier` – *n_estimators* - 1000, *min_samples_split* (minimalna liczba elementów, aby dokonać kolejnego podziału) - 200, *max_depth* (maksymalna głębokość drzewa) - 3; zbiór treningowy - 0.406, zbiór walidacyjny - **0.401**,
- `XGBClassifier` – *objective* - `binary:logistic`, *n_estimators* - 50; zbiór treningowy - 0.400, zbiór walidacyjny - **0.396**.

Dla metod *lasu losowego* i *gradient boostingu* uzyskano podobne wyniki, aczkolwiek zdecydowano się na użycie drugiej z tych metod do predykcji klasy na zbiorze udostępnionym przez prowadzącego, ponieważ wytrenowana została ona z użyciem mniejszej liczby drzew i w krótszym czasie.

Warto zaznaczyć, że podczas trenowania modeli wykorzystano parametr umożliwiający przekazanie wagi danej obserwacji. Dzięki temu model zwracał większą uwagę na przykłady z klasą równą 1, które są w mniejszości.

3.6 wyliczenie prawdopodobieństw dla zbioru z pliku *testx.txt*

Po wytrenowaniu 10 modeli (metoda kros-walidacji) na zbiorze treningowym dokonano predykcji dla danych udostępnionych przez prowadzącego (dla owych 10 modeli) i uśredniono prawdopodobieństwa klasy `1` dla każdego z modelu. Prawdpodobieństwa te zapisano do pliku `PIOPOD.txt`.

4 Wnioski

Podczas rozwiązywania zadania natknięto się na kilka problemów, których rozwiązanie było kluczowe dla jego rozwiązania:

- poradzenie sobie z cechami, które mają dużo braków,
- zakodowanie zmiennych kategorycznych,
- poprawne zaimplementowanie metryki LIFT_{10},
- niezbalansowanie zbioru treningowego,
- zapewnienie niezależności zbiorów treningowych i walidacyjnych.

  Sukcesywne odkrywanie i rozwiązywanie powyższych problemów pozwoliło na uzyskanie całkiem zadowalającego wyniku, który ma dużą szansę zostać powtórzony na zbiorze testowym dostarczonym przez prowadzącego, ponieważ podczas trenowania modeli zwrócono uwagę, aby nie zostały one przetrenowane.