Raport

Zaawansowane metody uczenia maszynowego

Projekt 1.

Autor:
Rafał Kobiela

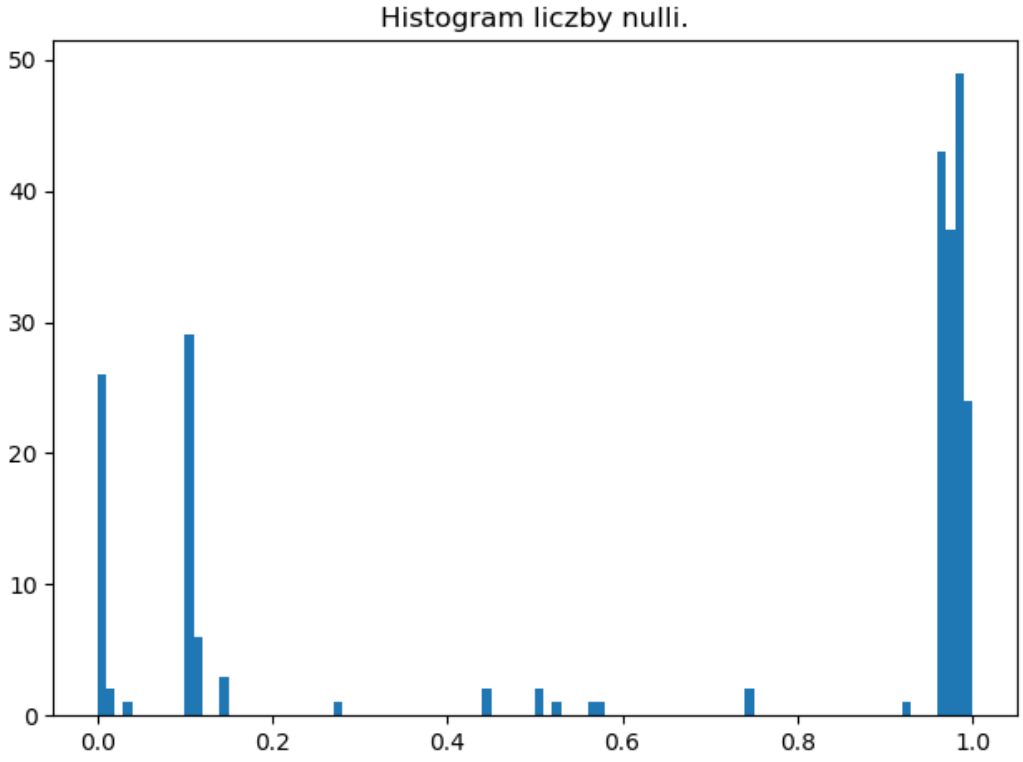
16 listopada 2018

1 Opis problemu

Celem projektu było zbudowanie jak najlepszego klasyfikatora który będzie przewidywał prawdopodobieństwo skorzystania z oferty handlowej. Był on oceniany według metryki "precyzja@10". Metryka ta jest o tyle ciekawa, że nie zalezy od wybranego thresholdu przynależności do danej klasy oraz w nejlepszym przypadku może nie wynosić 1.

2 Opis danych oraz ich przetwarzanie

Zbiór który otrzymaliśmyn do pracy miał 230 zmiennych z czego 193 były kategoryczny a reszta numeryczna. Zbiór treningowy miał 40000 obserwacji. Wiele zmiennych miało dużą liczbę brakujących obserwacji. Aby to sprawdzić sporządziłem histogram.



Rysunek 1: Rysunek prezentujący wykrywanie twarzy.

Patrząc na wykres zdecydowałem się zostawić zmienne, które mają mniej niż 30 procent wartości brakujących. Po tej operacji zostało już tylko 68 zmiennych. Na-

stępnie w braki danych z kolumn numerycznych zaimputowałem wartość średnią z kolumny. Tak przygotowane dane wyskalowałem, tak aby miały średnią 0 oraz wariancję równą 1. Wiadomo, że na tak przygotowanych danych klasyfikatory uzyskują lepsze wyniki niż na danych nieprzeskalowanych. W zmiennych kategorycznych z brakami danych poradziłem sobie wypełniając je najczęściej występującą wartością z danej kolumny. Jako, że do implementacji użyłem Pythona, który nie ma zaimlementowanej obsługi factorów, dane kategoryczne musialem przekształcić na formę zrozumiałą dla klasyfikatora, czyli One hot encodingi. Z uwagi na to, że niektóre zmienne miały po kilka tysięcy unikalnych wartości musiałem dokonać ich selekcji. Gdybym tego nie zrobił liczba kolumn zrobiłaby się większa niż licza obserwacji. Wybrałem kolumny które miały mniej niż 20 unikalnych wartości, które potem przekształciłem na One hot encodingi. Tak przygotowany zbiór danych miał ostatecznie 91 zmiennych.

3 Podejście

Na początku chciałem prostym sposobem wyznaczyć najlepszy klasyfikator do późniejszej optymalizacji. Były one sprawdzane podczas 5-krotnej kroswalidacji. Wyniki prezentują się następująco:

1. Logistic regression: 0.3584480600750939
2. Random forest: 0.37647058823529417
3. Gradient boosting: 0.3994993742177722
4. Adaboost: 0.39549436795994997
5. XGBoost: 0.3917521902377973
6. LightGBM: 0.4004181405917321

Jak widać najlepszy okazał się LightGBM z pakietu LightGBM oraz Gradient boosting z pakietu sklearn i to te klasyfikatory zdecydowałem się optymalizować w dalszej fazie projektu.

3.1 Optymalizacja

Na początku chciałem szukać optymalnych hiperparametrów Grid searchem, jednak szybko się okazało, że dużo lepszym rozwiązaniem będzie Random search. Parametry których szukałem dla Gradient boosting były z przedziałów:

1. learning rate : [0.1, 1]

2. n estimators: [100, 300]
3. min samples split: [2, 20]
4. max depth: [2, 10]

Parametry dla LightGBM:

1. learning rate : [0.1, 1],
2. num leaves : [20, 200],
3. max depth : [2, 20],
4. reg alpha : [0, 1],
5. reg lambda : [0, 1],
6. min child weight : [0, 0.1],
7. min child samples : [10, 100],
8. feature fraction : [0.3, 1],
9. bagging fraction : [0.3, 1]

Wyniki parametry podczas szukania były sprawdzane podczas 5-krotnej kroswalidacji. Najlepszym wynikiem jaki uzyskałem było 0.4062578 dla klasyfikatora LightGBM. Jednocześnie zdaję sobie sprawę z nie do końca poprawnego użycia kroswalidacji w tym przypadku. Przeprowadzając ją powinienem zbudować pipeline, który używałby wartości z części treningowej do skalowania oraz imputacji braków danych do części testowej. Tak jednak nie robiłem przez co wyniki testów mogą być nieco zawyżone.

4 Podsumowanie

Wybranym przeze mnie algorymem klasyfikacji został LightGBM z pakietu sklearn. Jest to implementacja Gradient boostingu z użyciem drzew decyzyjnych. Jest to jedna z najlepszych obecnie metod klasyfikacji. Jego niewątpliwą zaletą, nad na przykład XGBoostem, jest to, że trening jest bardzo szybki.