Katarzyna Spalińska

# Zaawansowane Metody Uczenia Maszynowego

# Projekt 1

## 1. Cel projektu

Celem projektu jest praktyczne przetestowanie 4 różnych metod klasyfikacji na rzeczywistych, zanonimizowanych danych telekomunikacyjnych. Celem klasyfikacji jest przewidywanie, z jakim prawdopodobieństwem klient skorzysta z oferty.

## 2. Wykorzystane narzędzia

Do zrealizowania projektu wykorzystano język programowania Python wraz z bibliotekami:

- pandas (obróbka zbiorów danych)
- numpy (obróbka zbiorów danych)
- sklearn (algorytmy uczenia maszynowego)
- xgboost (dodatkowa biblioteka implementująca algorytm XGBClassifier)

## 3. Przygotowanie danych

Początkowe wymiary zbioru treningowego wynosiły 231 kolumn x 40 tys. wierszy.

Pierwszym etapem było usunięcie kolumn niewnoszących żadnych informacji, czyli w całości wypełnionych brakami danych (NaN). Usunięcie ich ze zbioru danych spowodowało zmniejszenie się liczby kolumn do 213. Kolejnym krokiem było usunięcie również tych kolumn, w których występowało ponad 70% brakujących danych – to spowodowało zmniejszenie się treningowego zbioru danych do 75 kolumn.

W zbiorze danych nadal występowały braki danych, ale z ogólnej obserwacji wynikało, że występują one głównie jako brakujące prawie całe wiersze. Z tego względu wybrano sposób uzupełniania danych według metody „uzupełnienia ostatnią poprawną wartością”, dzięki czemu istnieje szansa, że prawie puste wiersze stały się bliskie rekordom rzeczywiście istniejącym. Nieodpowiednia w tej sytuacji wydawała się interpolacja liniowa, ponieważ nie ma wiedzy na temat tego, czy dane są w jakikolwiek sposób uporządkowane. Ponieważ przy wykorzystaniu tej metody pojedyncze początkowe rekordy nadal miały braki danych, zostały one po prostu usunięte, co zmniejszyło wymiary zbioru danych do 75 kolumn x 39 998 wierszy.

Kolejnym krokiem była zamiana zmiennych kategorycznych na liczbowe. Przetestowano dwie metody: Feature Encoding oraz Ordinal Encoding, i do dalszych eksperymentów wybrana została ta druga.

Metoda Feature Encoding okazała się dawać dużo gorsze wyniki (przedstawione w dodatkowym notatniku jupyterowym). Aby wykorzystać tę metodę poprawnie, konieczne jest aby podzielić zbiór danych treningowych na treningowy i walidacyjny przed kodowaniem danych. W przeciwnym przypadku wszystkie rekordy będą już „wstępnie dopasowane" funkcją fit_transform, która wykorzystuje zmienną „class" i prowadzi to do nadmiernego dopasowania (wcześniejsze wyniki modeli rzędu accuracy=1, prec@10=0.72 sugerowały błąd w przygotowaniu danych).

Metoda Ordinal Encoding nie wykorzystuje zmiennej „class", dlatego podział na zbiór treningowy i walidacyjny następuje w kolejnym kroku.

## 4. Wykorzystane metody uczenia maszynowego

Przy trenowaniu każdego z modeli została wykorzystana metoda dobierania hiperparametrów „random search". Wyjątkiem jest klasyfikator 2, Gaussian Naive Bayes, który posiada tylko jeden parametr.

### 4.1. Random Forest

Wybrane do zmieniania parametry:

- n_estimators [50:600; 50] – liczba estymatorów (drzew)
- max_depth [1:30] – maksymalna głębokość drzewa
- max_features [None, 'auto', 'log2'] – liczba zmiennych brana pod uwagę przy szukaniu najlepszego podziału drzewa (odpowiednio: wszystkie, pierwiastek, logarytm)

Najlepsze wyniki osiągnięte dla:

| n_estimators | max_depth | max_features | Prec@10 |
|---|---|---|---|
| 300 | 22 | log2 | 0.3985 |
| 50 | 8 | auto | 0.4023 |
| 300 | 27 | auto | 0.4038 |

### 4.2. Gaussian Naive Bayes

Wybrane do zmieniania parametry:

- var_smoothing – wartość dodawana do wariancji dla stabilności obliczeniowej

| var_smoothing | Prec@10 |
|---|---|
| 1e-13 | 0.1780 |
| 1e-12 | 0.1969 |
| 1e-11 | 0.2038 |

### 4.3. XGBoost

Wybrane do zmieniania parametry:

- n_estimators [50:600; 50] – liczba estymatorów (drzew)
- max_depth [1:30] – maksymalna głębokość drzewa
- learning_rate [0.05, 0.1, 0.2, 0.3, 0.4, 0.5] – współczynnik uczenia
- gamma - minimalna redukcja błędu wymagana do wykonania kolejnego podziału

Najlepsze wyniki osiągnięte dla:

| n_estimators | max_depth | learning_rate | gamma | Prec@10 |
|---|---|---|---|---|
| 100 | 4 | 0.1 | 2 | 0.4 |
| 50 | 2 | 0.2 | 4 | 0.4030 |
| 150 | 3 | 0.05 | 4 | 0.4038 |

### 4.4. AdaBoost

Wybrane do zmieniania parametry:

- n_estimators [50:500; 50] – liczba estymatorów (drzew)
- max_depth [1:20] – maksymalna głębokość drzewa
- algorithm [„SAMMA”,”SAMMA.R”] - algorytm używany przy boostingu
- learning_rate [0.05, 0.1, 0.2, 0.3, 0.4, 0.5] – współczynnik uczenia

Najlepsze wyniki osiągnięte dla:

| n_estimators | max_depth | algorithm | learning_rate | Prec@10 |
|---|---|---|---|---|
| 100 | 3 | SAMME | 0.1 | 0.3992 |
| 400 | 1 | SAMME | 0.4 | 0.4 |
| 50 | 2 | SAMME.R | 0.1 | 0.4038 |

Po wynikach eksperymentów okazuje się, że wszystkie trzy klasyfikatory oparte na drzewach (Random Forest, XGBoost, AdaBoost) osiągnęły tę samą maksymalną skuteczność według miary prec@10, jak również okazały się najlepsze. Poza GaussianNB przetestowano jeszcze kilka innych metod klasyfikacji, jednak wszystkie osiągały jeszcze gorsze wyniki.

Do otrzymania predykcji na zbiorze testowym zdecydowano się wykorzystać XGBoost, ponieważ wykonywał się najszybciej spośród trzech najlepszych modeli.

Warto zwrócić uwagę, że we wszystkich modelach nie pojawia się żadna zależność sugerująca, jakie wartości parametrów są skuteczniejsze niż inne.