Agata Pałdyna, 277019

ZMUM - **Projekt 1** – Raport

1. Cel projektu.

Celem projektu było praktyczne sprawdzenie metod klasyfikacji na rzeczywistych danych pochodzących od jednej z firm telekomunikacyjnych, wybór najlepszej z nich i dokonanie predykcji na zbiorze testowym. Zmienne zostały zanonimizowane ze względu na ochronę danych osobowych. Celem użycia metod klasyfikacji była identyfikacja klientów, którzy korzystają z oferty handlowej (zmienna `class=1`). Wartość `class=0` oznaczała, że klient nie skorzystał z oferty.

2. Opis przetwarzania danych.

Dane treningowe znajdowały się w pliku `train.txt`, a dane testowe w pliku `testx.txt`. W danych treningowych zaledwie około 7% obserwacji należy do klasy 1, zatem klasy są niezrównoważone. Na początku sprawdziłam procent występowania `NA` w poszczególnych zmiennych z podziałem na klasy. Okazało się, że zarówno w obserwacjach klasy 1, jak i klasy 0, jest podobny procent `NA` w poszczególnych zmiennych, zatem obecność `NA` nie wpływa znacząco na to, do której klasy należy obserwacja. Następnie dane z pliku `train.txt` zostały oczyszczone w następujący sposób:

- zmienne, w których ponad 97% wartości to były `NA`, zostały usunięte,
- w pozostałych zmiennych, za pomocą funkcji `impute()` z pakietu `e1071`, `NA` zostały zastąpione medianami wartości tych zmiennych,
- zmienne jakościowe, w których było ponad 51 różnych kategorii, zostały usunięte (ponieważ przy próbie trenowania modelu random forest występował błąd, że nie może być więcej niż 53 kategorie; większość z tych usuniętych zmiennych miała setki, a nawet tysiące różnych kategorii).

Usuwając wybrane kolumny z danych z pliku `train.txt`, jednocześnie usuwałam te same kolumny z danych z pliku `testx.txt`, aby ujednolicić oba te zbiory danych. Podobnie w kwestii uzupełniania braków danych medianami – zrobiłam to od razu w obu zbiorach.

Ponadto na potrzeby modeli, które można budować tylko na danych liczbowych, stworzyłam ‘liczbowy odpowiednik’ danych z pliku `train.txt` – za pomocą funkcji `as.numeric()` zamieniłam zmienne typu `factor` na zmienne liczbowe.

Do każdego treningu dzieliłam za pomocą kroswalidacji (metoda `createDataPartition()` z pakietu `caret`) oczyszczone dane na dwa zbiory:

- train – do trenowania modeli; tu 90% obserwacji,
- test – do testowania, który klasyfikator i z jakimi parametrami daje najlepszą predykcję; tu pozostałe 10% obserwacji.

Kolejnym krokiem było wybranie spośród analizowanych metod klasyfikacji najlepszej i dokonanie za jej pomocą predykcji dla danych ze zbioru `testx.txt`, przypisując każdemu klientowi prawdopodobieństwo skorzystania z oferty. Wyniki zostały zapisane w pliku `AGAPAL.txt`.

3. Podsumowanie eksperymentów.

Testowałam następujące metody klasyfikacji (w takiej kolejności jak zostały niżej wymienione):

1) las losowy – metoda `randomForest()` z pakietu `randomForest`,
2) adaboost – metoda `adaboost()` z pakietu `fastAdaboost` (próbowałam najpierw metody `boosting()` z pakietu `adabag`, ale uczenie trwało bardzo długo, natomiast używając metody `adaboost()` z pakietu `fastAdaboost` uczenia trwa około 50 razy szybciej),
3) xgboost – metoda `xgboost()` z pakietu `xgboost`,
4) drzewo – metoda `rpart()` z pakietu `rpart`.

Każdą z metod najpierw przetestowałam jednorazowo, w większości przypadków z domyślnymi parametrami, w celu zorientowania się, jakie mniej więcej wyniki dają poszczególne metody, oraz ile czasu zajmuje ich wykonanie. Dzięki temu mogłam zdecydować, na jak dużo iteracji mogę sobie pozwolić, oraz jak dużo parametrów mogę przetestować.

We wszystkich testach liczyłam trzy miary: precyzję (metoda `Precision()` z pakietu `MLmetrics`), AUC (metoda `AUC()` z pakietu `MLmetrics`), oraz precyzję @ 10% (własna implementacja na podstawie kodu z zajęć laboratoryjnych), jednakże wyboru najlepszych modeli spośród testowanych dokonywałam patrząc na precyzję @ 10%.

3.1 Las losowy.

W testach lasów losowych modyfikowałam parametr `ntree`. Zrobiłam tutaj kilka testów z różnymi wartościami tego parametru. Najlepszy wynik, wg miary prec10%, dał model dla `ntree = 24`. Poniższa tabelka przedstawia uśrednione wyniki poszczególnych miar dla tego parametru.

Precyzja	AUC	Precyzja @ 10%
95,37%	69,17%	47,16%

3.2 Adaboost.

W przypadku tej metody w testach zmieniałam parametr `nIter`. Ze względu na to, że uczenie tutaj trwało długo, przetestowałam tylko kilka wartości tego parametru. Najpierw przetestowałam wartości ze zbioru `{10, 30, 60, 100}`. Najlepszy wynik osiągnął wówczas model dla największej z testowanych wartości, tj. `nIter = 100` – precyzja @ 10% wyniosła w przybliżeniu 38,4%. Postanowiłam więc przetestować tą metodę dla większych parametrów, `{150, 200, 270}` (z dalszych testów tej metody zrezygnowałam, ponieważ ten drugi test trwał ok 30 godzin). I tutaj najlepszy wynik, wg miary prec10%, dał model dla `nIter = 150`. Poniższa tabelka przedstawia uśrednione wyniki poszczególnych miar dla tego parametru.

Precyzja	AUC	Precyzja @ 10%
95,54%	70,33%	39,32%

3.3 Xgboost.

W testach metody xgboost modyfikowałam parametry `nrounds` oraz `eta`. Zmienianie parametru `eta` nie wpływało na model, obliczane miary dawały takie same wyniki. Natomiast w przypadku parametru `nrounds` najlepszy wynik, wg miary prec10%, dał model dla `nrounds = 1`. Poniższa tabelka przedstawia uśrednione wyniki poszczególnych miar dla tego parametru.

Precyzja	AUC	Precyzja @ 10%
95,71%	71,61%	61,97%

3.4 Drzewo.

W przypadku drzew w testach zmieniałam parametry `minsplit` oraz `cp`. Zmienianie parametru `minsplit` nie wpływało na model, obliczane miary dawały takie same wyniki. Zmienianie parametru `cp` miało wpływ na model, ale tylko dla wartości `cp` z przedziału `[0.001, 0.002]` - dla kolejnych wartości tego parametru miary były stałe. Najlepszy wynik, wg miary prec10%, dał model dla `cp = 0.001`. Poniższa tabelka przedstawia uśrednione wyniki poszczególnych miar dla tego parametru.

Precyzja	AUC	Precyzja @ 10%
95,61%	70,76%	59,82%

4. Uzasadnienie wyboru końcowej metody.

Jak wspomniałam wcześniej, wyboru najlepszych modeli spośród testowanych dokonywałam patrząc na precyzję @ 10%. Następnie mając najlepszy model dla każdej z testowanych metod, wybrałam model, który dał najlepsze wyniki dla każdej obliczanej miary. Tym modelem jest **xgboost z parametrem `nrounds = 1`**. Miał on najlepszą precyzję, AUC oraz precyzję @ 10% spośród najlepszych modeli.