Raport

Jakub Bondyra

October 15, 2017

1 Wstęp

Niniejszy dokument jest opisem prac przeprowadzonych w celu rozwiązania zadanego problemu klasyfikacyjnego w ramach pierwszego projektu z przedmiotu *Warsztaty Badawcze*.

1.1 Zadanie problemu

Zadany był zbiór treningowy, liczący 50 000 rekordów oraz 50 zmiennych. Każdy rekord przyporządkowany był do jednej z dwóch klas - zatem mamy do czynienia z klasyfikacją binarną. Dostarczony został również zbiór testowy, składający się również z 50 000 rekordów. Celem projektu było zbudowanie modelu probabilistycznego, który będzie najlepszy wedle miary opisanej w następujący sposób:

- najpierw wyznaczane jest prawdopodobieństwo przynależności każdego rekordu ze zbioru testowego do klasy pozytywnej
- ze zbioru brane jest 10 000 rekordów o najwyższym prawdopodobieństwie (tzw. *score*)
- sprawdzany jest stosunek liczby rekordów z tego zbioru z rzeczywistą klasą pozytywną do wszystkich dziesięciu tysięcy rekordów tego zbioru.

Należy tutaj wspomnieć o tym, że klasy w zbiorze treningowym i testowym są zrównoważone, zatem w zbiorach na pewno znajduje się 10000 rekordów należących do klasy pozytywnej. Z tego też powodu miara ta jest dobrze zdefiniowana.

1.2 Ogólny opis prac

W swojej pracy stworzyłem wiele różnych modeli, które testowałem pod względem oczekiwanej wydajności wedle opisanej powyżej miary. Każdy, uwzględniony przeze mnie model, przygotowywałem w następujący sposób:

- najpierw dokonywałem selekcji zmiennych za pomocą algorytmów charakterystycznych dla analizowanego modelu
- następnie estymowałem wartości optymalne parametrów modelu, testując model dla różnych wartości poszczególnych parametrów, przy wykorzystaniu wielokrotnej kroswalidacji dla losowej próbki zbioru treningowego (tzw. *parameter tuning*)
- po wyznaczeniu optymalnych, przynajmniej lokalnie, parametrów modelu, testowałem skuteczność modelu zadaną powyżej miarą przy wykorzystaniu dziesięciokrotnej kroswalidacji - zatem dla zbiorów testowych wielkości 5000, sprawdzałem 1000 najlepszych *scores*.

Przed analizą modeli, zbiór treningowy oczyściłem, zamieniając predyktory kategoryczne na szereg predyktorów binarnych (tzw. *dummies*), usuwając predyktory skorelowane oraz predyktory, które zależały liniowo od innych predyktorów. W ten sposób uzyskałem 71 interesujących mnie predyktorów. *(uwaga - rozwijanie zmiennych kategorycznych tworzy więcej niż jeden nowy predyktor, dlatego też predyktorów jest więcej)*

Do pracy wykorzystałem środowisko R wraz z dodatkowymi pakietami, używałem w szczególności pakietu *caret* oraz pakietów zawierających różne modele klasyfikacyjne (np. *randomForest*, *naivebayes*, *xgboost*).

Całościowy opis mojej pracy zawarty jest w pliku jb_aux.ipynb, który powinien znajdować się w repozytorium.

1.3 Struktura dokumentu

W następnej sekcji opisuję najskuteczniejszy model, który jest moim proponowanym rozwiązaniem zadania. W sekcji trzeciej opisuję pokrótce inne rozwiązania problemu, które przeanalizowałem.

2 Rozwiązanie

Proponowanym przeze mnie modelem jest klasyfikator uzyskany z pakietu xgBoost. Jest to komitet drzew decyzyjnych, który został wytrenowany metodą gradient boostingu.

Przygotowując model, dokonałem selekcji zmiennych stosując metrykę ważności predyktorów charakterystyczną dla boostowanych drzew decyzyjnych. Wyznaczenie ważności predyktorów pozwoliło na pokazanie, że w boostowanych drzewach decyzyjnych istotne jest jedynie dziewięć predyktorów - pozostałe miały niemalże zerową wartość w metryce ważności. **Po więcej informacji odsyłam do pliku jb_aux.ipynb.**

Wedle mojej procedury postępowania, w dalszej kolejności dokonałem wyboru wartości hiperparametrów modelu (*parameter tuning*). W tym celu zbudowałem dużą liczbę modeli xgBoost na próbce zbioru treningowego i sprawdzałem ich skuteczność na zbiorze treningowym.

Specyficznie dla modelu xgBoost, losowa próbka używana do procedury *parameter tuning* była liczności 10 000, natomiast do oceny modelu stosowałem dziesięciokrotną kroswalidację zbioru treningowego.

W konsekwencji przeprowadzonej analizy hiperparametrów, ustaliłem następujące wartości parametrów:

- nrounds - 200
- max_depth - 1
- eta - 0.3
- gamma - 0.1
- colsample_bytree - 0.8
- min_child_weight - 0
- subsample - 0.7

Dla tak przygotowanego modelu zastosowałem dziesięciokrotną kroswalidację na zbiorze treninigowym i uzyskałem następujące wyniki:

- dla zwykłej miary skuteczności, uzyskałem średnią 72.5%
- dla miary zdefiniowanej w zadaniu problemu, uzyskałem średnią 84.75%

Warto zaznaczyć tutaj, że wyniki w każdej z miar dla każdego foldu były bardzo podobne - oznacza to zatem, że model dobrze się generalizuje. Dodatkowo, modele xgBoost odznaczały się bardzo dużą szybkością treningu (porównywalną nawet z szybkością naiwnego klasyfikatora Bayesa).

2.1 Wnioski

Model xgBoost został wybrany jako rozwiązanie, ponieważ osiągał najlepsze wyniki spośród wszystkich rozważanych modeli. Jest on również stosunkowo szybki, jak na stopień swojego skomplikowania. Warto wspomnieć również o tym, że odpowiednie dostrojenie hiperparametrów, z powodu ich mnogości, zajęło najwięcej czasu.

3 Inne rozwiązania

3.1 Regresja logistyczna

Analizując zastosowanie regresji logistycznej dla zadania, wykonywałem następujące operacje w celu selekcji ważnych zmiennych:

- wybrałem predyktory, dla których test na istotność osiągał p-wartość poniżej 0.1
- zastosowałem wsteczną metodę krokową BIC, przechodząc od modelu liniowego dla predyktorów wymienionych powyżej

W taki sposób uzyskałem niewielką liczbę ośmiu najważniejszych predyktorów.

Podobnie jak dla xgBoost, sprawdziłem działanie modelu dla wybranych predyktorów i dla dziesięciokrotnej kroswalidacji. Dla miary zdefiniowanej w zadaniu problemu uzyskałem średnią 72.92%, zatem dużo gorszą niż xgBoost.

3.1.1 Wnioski

Regresja logistyczna okazała się najgorszym modelem pod względem zadanej metryki dla zadanych danych.

3.2 Naiwny klasyfikator Bayesa

W problemach klasyfikacyjnych zawsze warto rozważać naiwny klasyfikator Bayesa, dlatego też spróbowałem również tego podejścia do danych.

Dla naiwnego klasyfikatora Bayesa, zastosowałem metrykę ważności predyktorów specyficzną dla tego klasyfikatora. Po wyznaczeniu ważności, do dalszych rozważań wybrałem tylko dwanaście predyktorów, arbitralnie obcinając pulę względem ważności (por. *jb_aux.ipynb*).

Podobnie jak dla xgBoost, sprawdziłem działanie modelu dla wybranych predyktorów i dla dziesięciokrotnej kroswalidacji. Dla miary zdefiniowanej w zadaniu problemu uzyskałem średnią 76.92%.

3.2.1 Wnioski

Skuteczność modelu okazuje się zaskakująco lepsza niż regresji logistycznej, mimo "naiwności" klasyfikatora. Warto wspomnieć również o tym, że ten model trenował się zdecydowanie najszybciej ze wszystkich.

3.3 Las losowy

W swoich rozważaniach wziąłem również pod uwagę las losowy, tworzony za pomocą pakietu *randomForest*.

Podobnie jak wcześniej, obliczyłem ważności predyktorów korzystając z metryki ważności predyktorów specyficznej dla lasów losowych. Arbitralnie obcinając pulę predyktorów pod względem tej wartości, uzyskałem 32 predyktory - pozostałe miały niemalże zerową ważność.

Wskutek procesu estymacji optymalnych wartości hiperparametrów, uzyskałem następujące wartości:

- mtry (liczba losowo wybieranych predyktorów) - 16
- ntree (liczba drzew) - 80

Dla tak zdefiniowanego modelu zastosowałem dziesięciokrotną kroswalidację i uzyskałem dla zadanej miary średnią 82.21%.

3.3.1 Wnioski

Lasy losowe dają jedne z lepszych wyników dla zadanej miary. Należy jednak zaznaczyć, że są one najwolniejszą rozważaną metodą.

3.4 Algorytm C5.0

W swoich rozważaniach badałem również inne, stosunkowo nowe metody oparte na drzewach decyzyjnych. Jednym z nich był model uzyskiwany z algorytmu Quinlana C5.0 - zmodyfikowanej wersji algorytmu C4.5, odznaczającej się lepszą efektywnością pamięciową i czasową. Metoda ta zapewnia również wsparcie dla boostingu, zatem model uzyskiwany z takiej metody wydaje się być ciekawą konkurencją dla modelu xgBoost.

W ramach selekcji predyktorów, zastosowałem do danych metrykę ważności predyktorów charakterystyczną dla boostowanych drzew decyzyjnych. Po arbitralnym obcięciu puli predyktorów, uzyskałem szesnaście najważniejszych predyktorów.

Wyznaczyłem również estymaty wartości optymalnych predyktorów, które wynosiły następująco:

- trials (iteracje boostingu) - 20
- model - rules (zatem modelem były reguły decyzyjne, a nie drzewo)
- winnow - FALSE (parametr dotyczący przesiewania predyktorów - niepotrzebny z powodu preselekcji predyktorów)

Dla zadanej miary i dla dziesięciokrotnej kroswalidacji uzyskałem średni wynik 84.08%.

3.4.1 Wnioski

Algorytm C5.0, który wspomaga się boostowaniem, osiąga wyniki zbliżone do wyników drzew xgBoost, jest on jednakże wolniejszy w trenowaniu.

3.5 Inne modele

Próbowałem również wytrenować dwa inne modele - maszynę wektorów podpierających z pakietu *e1071* oraz boostowane drzewa decyzyjne *adaBoost* z pakietu *fastAdaboost*. Obydwie te metody okazały się ekstremalnie niewydajne czasowo w treningu - z racji dostępności modeli o wystarczającej szybkości, eliminuje to tego typu modele z dalszych rozważań.

3.6 Zbiorcze wyniki modeli

Na zakończenie pokazuję tabelę podsumowującą własności wszystkich modeli - ich wyniki dla zadanej miary, liczbę wybranych do uczenia predyktorów oraz subiektywną ocenę szybkości uczenia (niestety, nie zmierzyłem tego żadną miarą).

Model	Wynik	Liczba uwzględnionych parametrów	Szybkość
Regresja logistyczna	72.92%	8	Duża
Naiwny Bayes	76.92%	12	Bardzo duża
Las losowy	82.21%	32	Niska
Reguły C5.0	84.08%	16	Średnia
xgBoost	84.75%	9	Średnia

Zbiorcze wyniki modeli