Raport

Filip Chrzuszcz Piotr Fic

1. Wstęp

Przedmiotem naszych analiz jest zbiór danych Bank Marketing. Zawiera on dane pochodzące z portugalskich banków, które opisują różne cechy klientów, którzy brali udział w kampanii telefonicznej banku. Celem jest przewidzenie czy klient założy lokatę w banku. Zbiór jest stosunkowo mały, zawiera około 4500 wierszy oraz 16 kolumn pomocniczych oraz 1 kolumnę celu. Nie zawiera żadnych danych brakujących, co z pewnością ułatwia pracę z tymi danymi.

2. Opis danych

W tej części pokrótce omówimy zmienne zawarte w zbiorze. Jednakże skupimy się głównie na tych, które okazały się najbardziej pomocne dla naszego modelu, o którym dokładnie napiszemy w dalszej części raportu.

Nasza zmienna celu jest kolumną binarną. Zawiera ona 4000 wartości negatywnych oraz 500 pozytywnych, co powoduje że nasz zbiór jest dość mocno niezbalansowany, co może rodzić problemy przy późniejszych predykcjach. Będzie również należało zwrócić uwagę na odpowiedni podział danych na treningowe oraz testowe, tak aby podobny procent tych dwóch zbiorów stanowiły obie klasy.

Przechodząc dalej do omówienia zmiennych pomocniczych, zdecydowanie warto zacząć od zmiennej ‘poutcome’. Już na etapie EDA widać, iż jest to zmienna nad którą warto się będzie uważnie pochylić, gdyż ukazuje ona wynik poprzedniej kampanii marketingowej prowadzonej przez bank na danym kliencie. Zawiera ona wartości failure, other oraz success. Okazuje się, iż ci klienci, którzy w poprzedniej kampanii pozytywnie na nią zareagowali (tzn, założyli lokatę), w tej kampanii również są w większości ku temu chętni.

Kolejnymi zmiennymi, które omówimy są ‘duration’ oraz ‘contact’. Obie mówią bezpośrednio o samej rozmowie odbywającej się w ramach kampanii i przedstawiają odpowiednio długość rozmowy oraz sposób w jaki skontaktowano się z klientem. Rozkład zmiennej ‘duration’ jest mocno skośny z medianą blisko 0, jednakże jest całkiem sporo outlierów świadczących o dłuższych rozmowach, które w zrozumiały sposób mogą się przyczynić do większych szans na lokatę. Mniej zrozumiały może być wpływ kolumny ‘contact’, jednakże z naszej analizy można wyciągnąć wniosek, iż ludzie wśród których forma kontaktu jest nieznana są bardzo mało chętni do założenia lokaty.

Reszta zmiennych nie miała zbyt wielkiego wpływu na działanie modelu, toteż nie będziemy poświęcali zbyt wiele miejsca na ich opisanie w tym raporcie. Warto jedynie wspomnieć, iż spora część zmiennych takie jak na przykład :

- previous (ilość kontaktów z klientem przed tą kampanią)

- pdays (ilość dni od ostatniego kontaktu)
- balance (roczny przychód)

ma bardzo skośny rozkład, co trzeba będzie zmienić przed przystąpieniem do modelowania.

3. Inżynieria cech

Aby móc rozpocząć modelowanie, należy najpierw odpowiednio przygotować dane. Rozpoczęliśmy ten proces od prostego zamienienia wszystkich wartości yes/no na odpowiednio 1/0. Następnie zdecydowaliśmy się na podział danych na treningowe oraz testowe, przykładając szczególną uwagę do tego, aby nie doszło do żadnego wycieku danych, który mogłby zafałszować nasz wynik.

Kolejnym krokiem było zmniejszenie skośności wyżej wymienionych kolumn. Zdecydowaliśmy się na użycie dwóch algorytmów, w zależności od wartości w danych kolumnach. Jeśli w kolumnie występowały wartości niedodatnie, to używaliśmy algorytmu ‘yeo -johnson’, a jeśli takowe nie występowały to ‘box-cox’.

Ostatnim krokiem na który zdecydowaliśmy się przed rozpoczęciem modelowania było zakodowanie zmiennych kategorycznych. Zdecydowaliśmy się to zrobić na dwa sposoby, a potem móc porównać je i wybrać najlepszy.

I. Pierwszy sposób zakłada zakodowanie kolumn w których występuje stosunkowo dużo unikalnych wartości, za pomocą Target Encodingu, a te w których unikalnych wartości jest mniej zakodować w sposób One Hot

II. Druga metoda zakłada zakodowanie wszystkich kolumn używając Target Encodera.

Przygotowawszy w ten sposób dane jesteśmy gotowi rozpocząć proces modelowania.

4. Modelowanie dla przygotowanych zbiorów

Po wykonaniu inżynierii cech dla oryginalnego zbioru danych oraz przygotowaniu dwóch wersji zbioru gotowego do modelowania możemy przystąpić do przygotowania modeli uczenia maszynowego. Istnieje wiele algorytmów dla problemu binarnej klasyfikacji. Zdecydowaliśmy się wybrać cztery konkretne, aby porównać osiągane przez nie rezultaty. Wybrane algorytmy to:

- Regresja logistyczna
- XGBoost
- Las losowy
- SVM (support vector machine)

Wcześniej dokonaliśmy podziału zbiorów na części treningową i testową. Teraz możemy użyć zbiorów treningowych do przeprowadzenia procesu kroswalidacji z dostrajaniem parametrów. Następnie modele z parametrami wybranymi w procesie kroswalidacji zostały ocenione na zbiorach testowych.

W tym momencie warto wspomnieć jakie miary są odpowiednie dla naszego zadania. Ze względu na niezbalansowaną zmienną celu, accuracy nie jest wystarczającą miarą. Musimy zwracać uwagę również na recall i precision. Patrząc na problem z biznesowego punktu widzenia, bardziej zależy nam na wysokiej wartości recall, akceptując spadek precision. Ponieważ błąd pominięcia klienta chętnego na subskrypcję jest poważniejszy niż zaproponowanie subskrypcji klientowi, który odrzuci ofertę.

Wnioski z doświadczenia:

- Modele uzyskały wyższe wyniki na zbiorze dla którego użyliśmy one-hot encoding wraz z target-encoding w zależności od zmiennej. Wynika stąd, że nasze przypuszczenia z eksploracji danych były słuszne.
- Różnice pomiędzy osiągami modeli były niewielkie. Patrząc wnikliwiej najlepiej z brakiem balansu danych poradził sobie XGBoost osiągając najwyższą miarę recall.

Na tym etapie do dalszego rozwoju projektu wybraliśmy oczywiście zbiór danych, który dał lepsze wyniki oraz algorytm XGBoost, gdyż uznaliśmy, że ma największy potencjał do dalszej poprawy.

5. Optymalizacja modelu

Wcześniej przeprowadziliśmy już klasyczne strojenie parametrów. Jednak dla wybranego algorytmu XGBoost zastosowaliśmy dodatkowo bardziej skomplikowaną optymalizację Bayesowską. Ostateczny model XGBoost ustawiliśmy według rezultatu tej metody.

6. Finalna ocena na zbiorze testowym

Na zbiorze testowym porównaliśmy miary wszystkich modeli.

	regresja logistyczna	XGBoost	las losowy	SVM
precision	0,60	0,48	0,65	0,48
recall	0,18	0,45	0,10	0,38
accuracy	0,89	0,88	0,89	0,88

Wyniki dla zbioru testowego

Wyniki wszystkich algorytmów okazały się bardzo podobne jeśli chodzi o miarę accuracy. Niestety miara recall okazała się nieakceptowalnie niska dla regresji logistycznej i lasu losowego. Nieco lepiej wypadł model SVM, najlepiej XGBoost zgodnie z przewidywaniami, ale jednak poniżej oczekiwań. Z tego powodu podjęliśmy kolejne działania w celu poprawy rezultatów.

7. Dopasowanie modelu

Modyfikując próg klasyfikacji dla modelu XGBoost możliwa jest poprawa miary recall. Pociąga to za sobą pogorszenie miary precision, jednak jak było wcześniej opisane, uwzględniając specyfikę zadania jest to akceptowalna metoda. Taką postać przyjmuje macierz pomyłek po przesunięciu progu decyzji.

Prawidłowa wartość	FALSE	686	115
	TRUE	26	78
		FALSE	TRUE
Predykcja			

Macierz pomyłek dla progu 0.1

Uzyskane w ten sposób miary modelu są następujące:

	XGBoost: próg=0.1
precision	0,40
recall	0,75
accuracy	0,84

Uważamy, że takie rozwiązanie jest przyzwoite biorąc pod uwagę jak słabe wyniki osiągnęły prostsze modele oraz trudności wynikające z braku balansu klas. Zdecydowaliśmy się jednak na podjęcie dodatkowych prób polepszenia wyników.

8. Stacking

Technika ta polega na użyciu kilku modeli, które kolejno ze swoich predykcji tworzą nowe zmienne w zbiorze. Finalną predykcję wylicza wybrany model, u nas na to miejsce wybrany został XGBoost, a pozostałe modele utworzyły pierwszą warstwę. Parametry modeli zostały ustawione na podstawie dostrajania przeprowadzonego wcześniej, kiedy działały pojedynczo.

Wyniki dla stack classifier:

	Stack classifier
precision	0,50
recall	0,40
accuracy	0,88

Okazało się, że metoda ta nie poprawiła wyników w oczekiwany sposób. Miary są na porównywalnym poziomie do osiąganych przez pojedyncze modele.

9. Over-sampling

Szansą na uporanie się z niezbalansowanym zbiorem danych jest over-sampling, czyli tworzenie sztucznych obserwacji dla mniej licznej klasy. Oczekuje się, że pomoże to modelowi lepiej nauczyć się zależności między zmiennymi. Spróbowaliśmy zatem jednej z metod over-samplingu: SMOTE. Wymaga ona jednak, aby dane były w pełni numeryczne już przed podziałem na zbiór treningowy i testowy, dlatego użycie inżynierii cech w dotychczasowej formie było niemożliwe. W zamian zastosowaliśmy one-hot encoding dla wszystkich zmiennych kategorycznych.

Dla zbioru przygotowanego metodą SMOTE przeprowadziliśmy kroswalidację z losowym szukaniem parametrów dla algorytmu XGBoost.

Pomimo, że model poprawił rezultaty podczas kroswalidacji, ostateczne wyniki na zbiorze testowym nie polepszyły się względem podejścia bez over-samplingu.

10. Znaczenie cech

Dla algorytmu XGBoost przeanalizowaliśmy jak poszczególne zmienne wpływały na decyzje podejmowane przez model. Okazało się, iż nasze przypuszczenia sformułowane na etapie eksploracji danych były trafne. Zmienne ‘poutcome’ oraz ‘duration’, na które zwróciliśmy uwagę analizując zbiór, były najczęściej wykorzystywane przez algorytm do podjęcia decyzji. Zaskakująco istotną zmienną okazała się forma kontaktu z klientem, gdyż w trakcie EDA nie zaobserwowaliśmy dla niej żadnych istotnych zależności.

11. Podsumowanie

Zbiór danych był ciekawym zadaniem do analizy i modelowania. Duże problemy sprawiał brak balansu zmiennej celu. Za ostateczne rozwiązanie uznajemy model XGBoost dostrojony optymalizacją bayesowską. Poprawienie miar, ważnych ze względu na specyfikę zadania, możemy uzyskać poprzez modyfikację progu decyzji modelu.